

*Моей жене, Ирине Николаевне Шалыто,
без помощи которой эта книга
не увидела бы свет, посвящается*

Введение

В книге рассматриваются методы аппаратной и программной реализации алгоритмов логического управления, которое основано на истинности и ложности входных и выходных переменных.

Основные теоретические исследования в этой области были начаты в мире с конца 30-х годов XX века. В течение многих лет признанным лидером в этом направлении в СССР был член-корреспондент АН СССР Михаил Александрович Гаврилов (1903—1979), работавший в Институте проблем управления (Институт автоматики и телемеханики) АН СССР (Москва). Кроме научного направления он создал еще и уникальное явление — Школу по теории релейных устройств и конечных автоматов, которая существует по сей день вот уже в течение более трех десятилетий и за это время проводилась более **т р и д ц а т и** раз в различных городах страны. В работе Школы принимали участие сотни специалистов, и она сформировала большое число крупных ученых в области логического управления.

Автор начал заниматься исследованиями в области логического управления в Научно-производственном объединении (НПО) «Аврора» (Ленинград) (ныне Федеральный научно-производственный центр — Государственное унитарное предприятие «НПО „Аврора“» (Санкт-Петербург)) с середины жизни этого научного направления, с 1971 г., после окончания Ленинградского электротехнического института им. В. И. Ульянова (Ленина) (ныне Санкт-Петербургский государственный электротехнический университет (ЛЭТИ)). Это было время расцвета теоретических исследований в данной области и широкого внедрения комплексных систем управления сложными промышленными и транспортными объектами. Автор имел честь познакомиться с М. А. Гавриловым и стать участником его Школы. В эти годы возникало много теоретических и практических задач, в решении которых автор принимал участие.

Сказанное, а также общение с многочисленными слушателями Института повышения квалификации руководящих работников и специалистов судостроительной промышленности (Ленинград), в котором автор преподавал много лет, привели его к активным, почти тридцатилетним исследованиям в указанной области, результаты которых обобщены в настоящей книге.

За эти годы элементная база систем логического управления быстро изменялась, пройдя путь от релейно-контактных элементов, микросхем малого и среднего уровней интеграции к программным логическим устройствам, а затем к большим и сверхбольшим интегральным схемам, частным случаем которых являются микропроцессоры и микроконтроллеры, используемые в качестве вычислительного ядра, в том числе и в промышленных (управляющих) компьютерах, и в программируемых логических контроллерах.

При этом в настоящее время в крупных и давно существующих организациях весьма часто возникает ситуация, когда одновременно ведутся работы по проектированию, модернизации и модификации различных поколений систем логического управления технологическими процессами, в каждой из которых превалирует тот или иной тип элементов и устройств из указанного выше спектра.

Каждый тип элементной базы порождает свои критерии и ограничения, что приводит к постановке новых научных задач, решение которых позволяет выполнить выдвигаемые практикой требования.

Эти требования связаны с различными характеристиками систем логического управления, например такими, как надежность, живучесть, диагностируемость. Однако все эти годы научные интересы автора были связаны с другими аспектами построения систем этого класса, которые характеризуются такими терминами, как «логическое проектирование» или «логический синтез», и были посвящены разработке формализованных методов аппаратной и программной реализации алгоритмов логического управления, которые являлись актуальными на протяжении многих лет и являются актуальными в настоящее время.

Так, например, задача синтеза комбинационных схем в базисе априори заданных произвольных элементов, решаемая в настоящей работе, являлась актуальной при использовании логических элементов различной физической природы и сложности и не потеряла своей актуальности в настоящее время применительно к сверхбольшим интегральным схемам программируемой логики и базовым матричным кристаллам, основой которых являются функциональные преобразователи различной сложности.

Если при построении схем ставились задачи минимизации аппаратуры (внешних выводов у настраиваемых модулей и модулей в схемах) и ее унификации, предельным случаем которой является однородность, то при программной реализации оптимизировались

объем памяти и быстродействие при одних методах и «понятность» и «безошибочность» программ при других методах, что связано с большой ответственностью объектов, управляемых системами рассматриваемого класса. Последнее свойство было и остается актуальным и при построении схем любого уровня интеграции.

При разработке предлагаемых методов везде, где это было возможно, автором получены оценки сложности предстоящих реализаций. Отметим также, что если некоторые классы схем и программ изоморфны и поэтому могут строиться одинаковыми методами, то возможность работы с целыми числами и арифметическими операциями позволяет предложить методы программной реализации алгоритмов логического управления, которые не имеют аналогов в аппаратуре.

При этом для аппаратных реализаций автором предложены методы построения только комбинационных схем, а для программных реализаций рассматривались также и автоматы с памятью. Это связано с тем, что «синхронность» программных реализаций резко упрощает построение этого класса автоматов по сравнению с построением асинхронных схем с памятью, синтез которых в настоящее время является отдельным направлением в логическом проектировании.

Специфика комбинационных схем, рассматриваемых в настоящей работе, состоит в том, что для них может не решаться проблема риска и не обеспечиваться высокое быстродействие, так как в системах логического управления технологическими процессами они в большинстве случаев воздействуют на инерционные исполнительные механизмы. Другая особенность этих схем состоит в том, что они обычно реализуют булевы формулы с малой повторностью переменных. Такие формулы характерны также и для пометок дуг графов переходов, описывающих поведение автоматов с памятью, применяемых в системах этого класса.

В логическом проектировании кроме задач, непосредственно возникающих из потребностей практики, имеется также уже ставшая традиционной область теоретических задач, связанных с исследованиями как класса всех булевых функций, являющихся одним из важнейших объектов дискретной математики, так и таких его подклассов, которые образуют, например, линейные, симметрические, пороговые и самодвойственные функции. Большой интерес представляют также исследования булевых формул, являющихся одной из реализаций булевых функций, первичным заданием которых являются таблицы истинности. При этом, по мнению автора, особую важность имеют простейшие из булевых формул — неповторные (в определенном базисе), свойства которых, как показано в настоящей работе, связаны с такими фундаментальными понятиями математики (в основном дискретной), как например «эйлеровы графы», «клики в графах», «числа Фибоначчи» и «число e ».

Из изложенного вытекает круг вопросов, рассматриваемых в книге, которые детально перечислены в оглавлении.

При этом интересно отметить, что прошло почти 30 лет от момента постановки задачи до разработки мультиплексорного метода, который появился неожиданно в процессе написания этой работы и явился одним из важнейших результатов автора. Отметим также, что все методы, рассматриваемые в книге, иллюстрируются большим числом примеров, ибо в соответствии с приписываемым И. М. Гельфанду высказыванием: «Теории приходят и уходят, а примеры остаются» (В. Арнольд. Предисловие к книге: Р. Грехем, Д. Кнут, О. Паташник. Конкретная математика. Основание информатики. М.: Мир, 1998).

Ряд результатов, описываемых в настоящем издании, и в частности SWITCH-технология, были использованы и используются в настоящее время в НПО «Аврора» при проектировании систем логического управления судовыми техническими средствами и другими объектами, в том числе и с участием таких иностранных фирм, как «ABB Stromberg» (Финляндия), «Norcontrol» (Норвегия), «FF-Automation» (Финляндия).

Работа выполнена в Научно-производственном объединении «Аврора», Руководству которого автор глубоко признателен за предоставленную возможность для проведения исследований, результаты которых составили содержание настоящей книги.

Автор особо благодарен своему Учителю В. Л. Артюхову, который в течение многих лет учил меня «уму-разуму» и не только в вопросах логического управления. Помощь Б. П. Кузнецова, В. Н. Кондратьева, Н. И. Туккеля позволила совместно получить результаты, которые являются украшением данной работы.

Ряд результатов получен автором при выполнении исследований на кафедре «Компьютерные технологии» Санкт-Петербургского государственного института точной механики и оптики (технического университета), Руководству которого автор также искренне признателен. Содержание книги неоднократно апробировалось при чтении лекций студентам, проходящим обучение в магистратуре указанной кафедры, высокие способности которых в области точных наук обеспечили эффективную экспертизу результатов, полученных мною.

Я благодарю также Л. Я. Розенблюма, Д. А. Поспелова и О. П. Кузнецова за многолетний благожелательный интерес к моим исследованиям.

Я признателен Н. А. Абрамовой, Г. П. Агибалову, А. А. Амбарцумяну, В. И. Анисимову, В. В. Антипову, Ю. В. Баглюку, О. Л. Бандман, В. А. Богдановой, В. Н. Вагину, С. В. Вальчуку, В. И. Варшавскому, В. Н. Васильеву, В. В. Войтецкому, И. В. Герасимову, В. А. Годунову, А. К. Григорян, В. В. Девяткову, А. Д. Закревскому, В. Н. Захарову, Г. И. Иванову, В. М. Корчанову, Б. А. Кулику,

В. Г. Лазареву, В. Д. Малюгину, А. В. Марковскому, Г. И. Никитину, Г. И. Новикову, В. А. Осипову, В. Г. Парфенову, П. П. Пархоменко, Е. И. Пийль, И. В. Прангишвили, Д. В. Пузанкову, Е. И. Пупыреву, И. А. Рябинину, Ю. Л. Сагаловичу, А. С. Смирнову, Е. Н. Туруте, В. П. Чистову, Л. Б. Шипиловой, В. Б. Яковлеву и А. Е. Янковской, человеческая теплота которых в течение многих лет согревали меня и мои исследования.

Я благодарен за проявленное внимание А. Н. Берлину, Л. М. Гольденбергу, В. П. Иванникову, Ю. Л. Иваськиву, М. Б. Игнатьеву, Ф. Э. Келлеру, О. И. Кутузову, А. Х. Мурсаеву, К. В. Недялкову, А. Е. Сазонову, Б. Я. Советову, Влад. В. Сапожникову, А. Н. Терехову и Е. П. Угрюмову.

Я особо признателен за внимание Н. А. Семихатову.

Автор благодарен также редактору этой и предыдущей книг Л. С. Тихомировой, огромный труд которой несомненно привел к их улучшению.

Не могу не поблагодарить своих Родителей Абрама Израилевича и Валентину Георгиевну Шалыто, дочь Инну и сестру мою мамы — Маргариту Георгиевну Рубинову за все то, что они сделали для меня в жизни.

Я всегда буду помнить М. А. Гаврилова, Г. А. Копейкина, Л. М. Фишмана, В. А. Кондрашова, В. И. Евсеева, Б. Л. Овсевича, Э. В. Сергеева, И. Р. Фрейдзона, А. А. Первозванского и Н. К. Перкова.

Глава 1

Булевы формулы и булевы функции. Классификация, табулирование, свойства

Функция, у которой ее значения и значения аргументов двоичны, называется **булевой**.

Булевы функции (БФУ) задаются в виде **таблиц истинности (ТИ)**.

В настоящей работе, если это не оговаривается особо, рассматриваются одиночные, полностью определенные БФУ n переменных, заданные на 2^n входных наборах. Основной метрикой БФУ является число переменных n .

Аналитическая запись БФУ в виде формулы, использующей логические операции (базис формулы), называется **булевой формой (БФ)**.

В настоящей работе рассматриваются формулы в различных базисах, но основное внимание уделяется формулам в базисе И, ИЛИ, НЕ. Рассматриваются также формулы в таком нетрадиционном для дискретной математики [1] базисе, как И, ИЛИ, НЕРАВНОЗНАЧНОСТЬ, НЕ, который является избыточным по критерию функциональной полноты [2], но весьма удобным для сокращения длины формул и позволяющим легко декомпозировать формулы вследствие того, что для указанных двухместных операций выполняется сочетательный закон [3].

Зафиксируем базис логических операций. При этом основной метрикой БФ является число (h) символов переменных и их инверсий в правой ее части, которые будем называть буквами. Булевы формулы в фиксированном базисе из h букв могут быть разделены на два класса: неповторные, для которых $h = n$, и повторные, для которых $h > n$.

Повторная формула в одном базисе может быть неповторной в другом базисе, и наоборот. Например, повторная формула $f = !x_1 x_2 \vee x_1 !x_2$ в базисе $\{\&, \vee, !\}$, для которой $n = 2$, $h = 4$, явля-

ется неповторной в другом базисе: $f = x_1 \oplus x_2$. Здесь «!» — символ операции НЕ (инверсии).

Б е с п о в т о р н ы е булевы формулы (ББФ) в базисе $\{\&, \vee, !\}$ имеют важное значение, так как они

— широко используются на практике, и в том числе в алгоритмах управления судовыми техническими средствами [3—5];

— изоморфны неповторным параллельно-последовательным контактным схемам [5] и дихотомическим деревьям [6];

— их сравнительно «немного» [3];

— любая повторная формула из h букв может быть получена из однотипной с ней ББФ из h букв за счет переобозначений переменных, но не наоборот [3].

1.1. Классификация неповторных булевых формул в базисе И, ИЛИ, НЕ

PN-классификация. Классификация булевых формул по типам относительно двух операций эквивалентности (перестановка переменных и замена переменных их инверсиями) называется *PN-классификацией* (от английских слов «permutation» — перестановка и «negation» — отрицание).

Если в качестве представителей типов ББФ в базисе $\{\&, \vee, !\}$ выбрать положительно монотонные формулы, то им изоморфны соответствующие представители типов параллельно-последовательных контактных схем (П-схем) из нормально разомкнутых контактов.

В [5] выполнено подробное исследование этого класса схем и получена производящая функция для подсчета количества типов таких схем из h контактов. Количество типов T_1 этих схем и соответственно *PN*-типов ББФ при $h \leq 10$ приведено в табл. 1.1.

Т а б л и ц а 1.1

h	1	2	3	4	5	6	7	8	9	10
T_1	1	2	4	10	24	66	180	522	1532	4624

Имея при $h = \text{const}$ всех представителей типов, за счет перестановки входных переменных и расстановки инверсий над одиночными символами переменных могут быть получены все ББФ из h букв, из которых, в свою очередь, за счет отождествлений могут быть получены все БФ из h букв в рассматриваемом базисе.

В [3] было предложено для перечисления *PN*-типов ББФ использовать структурные полиномы, в которых числа изображают число букв в конъюнкциях, а умалчиваемые знаки произведения (в скобочных формах) и знаки суммы соответствуют операциям конъюнкции и дизъюнкции. При этом отметим, что эти полиномы отража-

ют только структуру формул и не предназначены для вычислений по ним. Например, формуле $f = (x_1 x_2 x_3 \vee !x_4 x_5) x_6 x_7$ соответствует полином $(3 + 2)2$.

Табулирование полиномов при $h \leq 8$ выполнено автором и приведено в [7]. Например, при $h = 3$ таких полиномов четыре: $3, 2 + 1, 1 + 1 + 1, (1 + 1)1$.

Среди всех PN -типов ББФ могут быть выделены PN -типы, соответствующие неповторным бесскобочным формулам — неповторным дизъюнктивным нормальным формам (ДНФ). Количество PN -типов T_2 таких формул при $h \leq 8$ приведено в табл. 1.2.

Т а б л и ц а 1.2

h	1	2	3	4	5	6	7	8
T_2	1	2	3	5	7	11	15	22

Эти типы совпадают с числом разбиений числа h на слагаемые, для подсчета которых Раманужаном была получена рекуррентная формула [8].

N -классификация. В [8] приведены производящие функции для определения числа Π -схем, снабженных различными ярлыками (различные индексы в обозначениях контактов). Эта классификация соответствует N -классификации ББФ. Число N -типов ББФ (T_3) при $h \leq 7$ приведено в табл. 1.3 [8].

Т а б л и ц а 1.3

h	1	2	3	4	5	6	7
T_3	1	2	8	52	472	5504	78146

При $h = 3$ представителями N -типов ББФ являются следующие формулы: $x_1 x_2 x_3$, $x_1 x_2 \vee x_3$, $x_1 x_3 \vee x_2$, $x_2 x_3 \vee x_1$, $x_1 \vee x_2 \vee x_3$, $(x_1 \vee x_2) x_3$, $(x_1 \vee x_3) x_2$, $(x_2 \vee x_3) x_1$, из которых за счет расстановки инверсий над одиночными символами переменных могут быть получены все ББФ при $h = 3$.

P -классификация. При P -классификации ББФ различают формулы, отличающиеся лишь структурой и расстановкой инверсий над одиночными символами переменных.

В [3] определено число P -типов ББФ (T_4) при $h \leq 6$ (табл. 1.4).

Т а б л и ц а 1.4

h	1	2	3	4	5	6
T_4	2	6	20	80	340	1582

При $h = 2$ представителями P -типов ББФ являются следующие формулы: $!x_1 !x_2$, $!x_1 x_2$, $x_1 x_2$, $!x_1 \vee !x_2$, $!x_1 \vee x_2$, $x_1 \vee x_2$, из которых за счет перестановки переменных могут быть получены все ББФ при $h = 2$.

Число неповторных булевых формул. В [3] при $h \leq 10$ определено число ББФ. В табл. 1.5 приведено их количество (T_5) при $h \leq 6$.

Т а б л и ц а 1.5

h	1	2	3	4	5	6
T_5	2	8	64	832	15104	352265

Перечислим ББФ при $h = 2$: $\neg x_1 \neg x_2$, $\neg x_1 x_2$, $x_1 \neg x_2$, $x_1 x_2$, $\neg x_1 \vee \neg x_2$, $\neg x_1 \vee x_2$, $x_1 \vee \neg x_2$, $x_1 \vee x_2$.

В [3] установлен весьма интересный факт, состоящий в том, что

$$B(h) = \frac{\log T_5}{\log T_1}$$

приблизительно равно трем (по крайней мере при $h \leq 9$). Этот показатель может использоваться в качестве инварианта класса ББФ в рассматриваемом базисе. Отметим, что здесь и в дальнейшем, если это не оговорено особо, применяются логарифмы по основанию два.

***NP*N-классификация.** Вводя третью операцию эквивалентности — замену формулы ее отрицанием, из представителей *PN*-типов ББФ можно получить представителей *NP*N-типов этих формул. Так как при $h \geq 2$ среди представителей *PN*-типов ББФ нет самодвойственных, то количество *NP*N-типов ББФ (T_6) определяется при $h \geq 2$ делением значений, приведенных в табл. 1.1, на два (табл. 1.6).

Т а б л и ц а 1.6

h	1	2	3	4	5	6	7	8
T_6	1	1	2	5	12	33	90	261

1.2. Деревья из двухвходовых элементов

1.2.1. Подсчет числа неизоморфных деревьев

В [3, 9] приведено рекуррентное соотношение для подсчета числа неизоморфных деревьев с h входами из $h - 1$ двухвходовых элементов, которое имеет следующий вид:

$$R = \begin{cases} \sum_{i=1}^{[h/2]} R_i R_{h-i} & \text{при } h = 2k + 1, \quad k \geq 1; \\ \sum_{i=1}^{h/2} R_i R_{h-i} & \text{при } h = 2k, \quad k = 1, 2, 3; \\ \sum_{i=1}^{h/2} R_i R_{h-i} & \text{при } h = 2k, \quad k \geq 4, \end{cases}$$

где $[h/2]$ — символ округления числа $h/2$ до ближайшего меньшего целого.

Значения R при $h \leq 12$, впервые полученные в [10], указаны в табл. 1.7.

Т а б л и ц а 1.7

h	1	2	3	4	5	6	7	8	9	10	11	12
R	1	1	1	2	3	6	11	23	46	98	207	451

На рис.1.1 приведено дерево, являющееся линейным каскадом при $h = 3$, а на рис. 1.2 — деревья при $h = 5$.

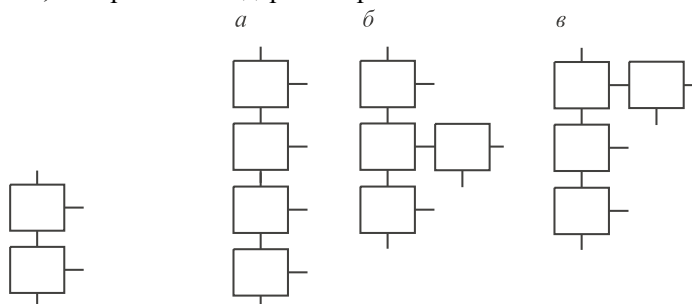


Рис. 1.1

Рис. 1.2

Рассмотренные деревья могут быть использованы в качестве более «крупной» классификации ББФ. Интересно отметить, что число типов в этой классификации, например для ДНФ из пяти букв, не может быть уменьшено.

1.2.2. Покрывающие деревья

Покрывающим называется дерево, которое за счет исключения из него одного или нескольких двухвходовых элементов позволяет получить любое дерево с h входами.

В табл. 1.8 при $h \leq 8$ приведено число элементов в покрывающих деревьях.

Т а б л и ц а 1.8

h	2	3	4	5	6	7	8
Э	1	2	4	5	8	10	15

На рис.1.3 приведено покрывающее дерево при $h = 5$.

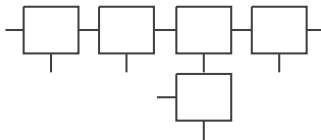


Рис. 1.3

1.2.3. Модули с простой структурой, универсальные в классе формул

Покрывающие деревья являются основой для построения модулей с простой структурой, универсальных в классе формул. Если при $h = \text{const}$ в соответствующем покрывающем дереве каждый элемент заменить трехходовым мажоритарным элементом (МЭ), то получится модуль, универсальный в классе формул в базисе $\{\&, \vee, !\}$ из h букв. При этом предполагается, что прямые и inverse выходы источников информации равнодоступны.

На рис. 1.4 приведен модуль, универсальный в рассматриваемом классе формул из пяти букв (5-универсальный модуль), настроенный в качестве примера на реализацию ББФ $f = (x_1 !x_2 \vee x_3) (x_4 \vee x_5)$.

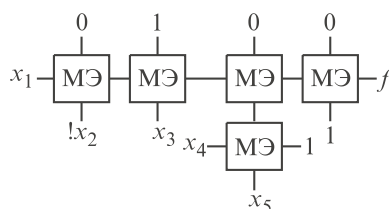


Рис. 1.4

Достоинства рассмотренных модулей состоят в простых структуре и настройке, а также в однотипности элементов, из которых они построены, а недостаток — в большом числе внешних выводов. Так, модуль на рис. 1.4 имеет 12 выводов — 11 входов и один выход.

Несколько более сложный метод построения модулей с простой структурой, основанный на наложении деревьев из элементов И и ИЛИ с различным числом входов и применении *NPN*-классификации, рассмотрен в [11]. Этот метод позволил, в частности, построить 5-универсальный модуль с 11 выводами — девятью входами и двумя выходами.

В заключение раздела отметим, что еще более сложный метод, излагаемый в гл. 7, позволяет за счет увеличения элементной сложности уменьшить число внешних выводов и построить, в частности, 5-универсальный модуль с девятью выводами [12].

1.3. *PN*-классификация неповторных формул в базисе И, ИЛИ, НЕРАВНОЗНАЧНОСТЬ, НЕ

В [13] установлено, что число *PN*-типов ББФ в этом базисе из h букв, реализуемых линейным каскадом из $h-1$ двухвходовых элементов И, ИЛИ, НЕРАВНОЗНАЧНОСТЬ, определяется соотношением

$$T_7 = F_{2h+1},$$

где F_{2h+1} — член с номером $2h+1$ ряда Фибоначчи, начинающегося с нуля [14].

В [3] отмечено, что при $h \geq 3$

$$T_7 < 3^{h-1}.$$

В [14] найдено при $h \leq 6$ число *PN*-типов формул рассматриваемого класса T_8 , а в [3] выполнено их табулирование при $h \leq 4$.

В [9] приведена верхняя оценка T_9 числа *PN*-типов этих формул для произвольных h , которая имеет следующий вид:

$$T_9 = RT_7.$$

В табл. 1.9 приведены значения T_7 , T_8 и T_9 при $h \leq 8$.

Т а б л и ц а 1.9

h	1	2	3	4	5	6	7	8
T_7	1	3	8	21	55	144	377	987
T_8	1	3	8	29	101	405	?	?
T_9	1	3	8	42	165	864	4147	22791

1.4. Свойства неповторных формул в базисе И, ИЛИ, НЕ

1.4.1. Ранг функций

Число единиц в столбце значений булевой функции называется ее **рангом**.

Приведем несколько утверждений относительно рангов функций:

— если БФУ имеет нечетный ранг, то она существенно зависит от всех своих переменных [15];

— ранг функции, несущественно зависящей от некоторых переменных, — четное число;

— ранг ББФ в базисе $\{\&, \vee, !\}$, существенно зависящей от всех переменных, нечетен, но не наоборот, так, например, повторная БФ $f = !x_1!x_2x_3 \vee x_1x_2$ имеет ранг три [3];

— существуют различные PN -типы ББФ в этом базисе, имеющие одинаковые ранги, например формулы $f_1 = x_1x_2 \vee x_3x_4$ и $f_2 = x_1(x_2 \vee x_3 \vee x_4)$ имеют ранг семь;

— самодвойственные функции, существенно зависящие от n переменных, не реализуются ББФ в указанном базисе, так как функции этого класса имеют четный ранг, равный 2^{n-1} [9].

1.4.2. Вычисление числа единиц по формуле

Таблично-аналитический метод подсчета числа единиц для бесповторных булевых формул предложен в [3].

В [16] для решения этой задачи используется одна из аксиом теории вероятностей. Так как в таблице истинности $P(x_i=0) = P(x_i=1) = 1/2$, определим вероятность того, что БФУ принимает значение, равное единице, в виде дроби

$$P = \frac{r}{2^n},$$

в которой числитель определяет ранг функции. Таким образом, «вероятностный» метод позволяет определить невероятностную характеристику БФУ.

В [16] было предложено преобразовывать ББФ в базисе $\{\&, \vee, !\}$ в ББФ в базисе И—НЕ, избавляясь тем самым от неортогонализированных операций ИЛИ, нарушающих аксиому о независимости событий. По преобразованной формуле записывается формула для вычисления искомой вероятности, что весьма трудоемко.

Более простой метод, использующий эту аксиому, был предложен автором и А.Ю.Еремеевым в [17], который позволяет применить ее непосредственно к заданной ББФ. При этом для подформулы вида $f_1 = x_i x_j$ записываются соотношения

$$P_{ij} = P_i P_j; \quad Q_{ij} = 1 - P_{ij},$$

а для подформулы $f_2 = x_i \vee x_j$ — соотношения

$$Q_{ij} = Q_i Q_j; \quad P_{ij} = 1 - Q_{ij},$$

а затем на основе этих соотношений по заданной ББФ определяются значения P и r .

Пример 1.1. Определить ранг функции, заданной ББФ $f = x_1x_2 \vee x_3x_4$.

Выполним следующие вычисления:

$$P_{12} = P_1 P_2 = \frac{1}{4}; \quad Q_{12} = 1 - P_{12} = \frac{3}{4};$$

$$P_{34} = P_3 P_4 = \frac{1}{4}; \quad Q_{34} = 1 - P_{34} = \frac{3}{4};$$

$$Q = Q_{12} Q_{34} = \frac{9}{16}; \quad P = 1 - Q = \frac{7}{16}.$$

Таким образом, $r = 7$.

Еще более простой метод решения сформулированной задачи может быть предложен на основе построения по ББФ, за счет ортогонализации всех дизъюнкций путем «домножения», ортогональной скобочной формулы (ОСФ). Для уменьшения трудоемкости метода ОСФ должна, по возможности, содержать минимальное число букв. По ОСФ за счет замены x_i на P_i , $\neg x_i$ на Q_i ($P_i = Q_i = 1/2$) строится искомая формула для подсчета вероятности числа единиц.

Пример 1.2. Решить задачу, рассмотренную в предыдущем примере.

Построим ОСФ:

$$f = x_1 x_2 \vee x_3 x_4 = x_1 x_2 \vee \neg(x_1 x_2) x_3 x_4 = x_1 x_2 \vee (\neg x_1 \vee \neg x_2) x_3 x_4 =$$

$$= x_1 x_2 \vee (\neg x_1 \vee x_1 \neg x_2) x_3 x_4.$$

При этом

$$P = P_1 P_2 + (Q_1 + P_1 Q_2) P_3 P_4 =$$

$$= \left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + \left(\left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)\left(\frac{1}{2}\right)\right)\left(\frac{1}{2}\right)\left(\frac{1}{2}\right) = \frac{7}{16}.$$

Следовательно, $r = 7$.

Пример 1.3. Подсчитать число единиц в таблице истинности для ББФ

$$f = (((x_1 \vee x_2) x_3 \vee x_4) x_5 \vee x_6) x_7 \vee x_8.$$

Построим ОСФ с минимальным числом букв:

$$f = (((x_1 \neg x_2 \vee x_2) x_3 \neg x_4 \vee x_4) x_5 \neg x_6 \vee x_6) x_7 \neg x_8 \vee x_8.$$

При этом

$$P = (((P_1 Q_2 + P_2) P_3 Q_4 + P_4) P_5 Q_6 + P_6) P_7 Q_8 + P_8 = \frac{171}{256}.$$

Таким образом, $r = 171$.

Для упрощения построения ОСФ для повторных в рассматриваемом базисе формул ортогонализацию дизъюнкций в них будем про-

водить не только за счет «домножения», но и за счет проведения разложения Шеннона (разд.3.1.1) по повторным переменным.

С целью получения в формуле для вычисления P величины равной 2^n при необходимости ОСФ корректируется: подформулы в ней умножаются на выражения вида $!x_i \vee x_i$, где x_i — отсутствующая в подформуле переменная, которая не позволяет получить в ортогональной ДНФ (ОДНФ) хотя бы одну конъюнкцию, состоящую из n букв. При этом отметим, что в формуле для подсчета P выражению $!x_i \vee x_i$ соответствует выражение $Q_i + P_i$, записываемое в окончательной форме не как единица, а как дробь $2/2$.

Пример 1.4. Определить ранг БФУ, заданной формулой $f = (x_1 \vee x_2) x_3 \vee x_1 x_2$.

Выполним разложение Шеннона по переменной x_3 :

$$f = !x_3 x_1 x_2 \vee x_3 (x_1 \vee x_2).$$

Построим ОСФ:

$$f = !x_3 x_1 x_2 \vee x_3 (x_1 \vee !x_1 x_2).$$

Если по ОСФ построить ОДНФ, то в ней две конъюнкции содержат по три буквы, и поэтому корректировка ОСФ не требуется. При этом

$$P = Q_3 P_1 P_2 + P_3 (P_1 + Q_1 P_2) = \frac{4}{8}.$$

Следовательно, $r = 4$.

Пример 1.5. Определить ранг БФУ, заданной формулой $f = (!x_1 \oplus x_2) \vee (x_3 \oplus !x_4)$.

Запишем БФУ в виде $f = !x_1 !x_2 \vee x_1 x_2 \vee !x_3 !x_4 \vee x_3 x_4$ и разложим ее по переменной x_1 :

$$f = !x_1 (!x_2 \vee !x_3 !x_4 \vee x_3 x_4) \vee x_1 (x_2 \vee !x_3 !x_4 \vee x_3 x_4).$$

Остаточные повторные формулы ортогонализуем «домножением»:

$$f = !x_1 (!x_2 \vee x_2 !x_3 !x_4 \vee x_2 x_3 x_4) \vee x_1 (x_2 \vee !x_2 !x_3 !x_4 \vee !x_2 x_3 x_4).$$

ОДНФ в этом случае содержит конъюнкции из четырех букв, и поэтому корректировка ОСФ не требуется. При этом

$$P = Q_1 (Q_2 + P_2 Q_3 Q_4 + P_2 P_3 P_4) + P_1 (P_2 + Q_2 Q_3 Q_4 + Q_2 P_3 P_4) = \frac{12}{16}.$$

Следовательно, $r = 12$.

Пример 1.6. Определить ранг БФУ, заданной формулой $f = !x_1 x_2 \vee !x_1 x_3 \vee x_2 x_4$.

Выполним разложение Шеннона по переменной x_2 :

$$f = !x_2 !x_1 x_3 \vee x_2 (!x_1 \vee x_4).$$

Построим ОСФ:

$$f = !x_2 !x_1 x_3 \vee x_2 (!x_1 \vee x_1 x_4).$$

ОДНФ этой функции не содержит конъюнкций из четырех букв, и поэтому откорректируем ОСФ, например следующим образом:

$$f = !x_2 !x_1 x_3 (!x_4 \vee x_4) \vee x_2 (!x_1 \vee x_1 x_4).$$

При этом

$$P = Q_2 Q_1 P_3 (!Q_4 + P_4) + P_2 (Q_1 + P_1 P_4) = \frac{8}{16}.$$

Следовательно, $r = 8$.

1.4.3. Свойства бесповторных пороговых формул

Для PN -типов бесповторных пороговых формул (БПФ) могут быть выбраны такие представители, у которых все единицы в столбцах значений расположены подряд, начиная со «дна» ТИ.

Инвариантами класса PN -типов БПФ, существенно зависящих от n переменных, являются нечетные числа, соответствующие их рангам: 1, 3, 5, ..., $2^n - 1$.

Число PN -типов БПФ, существенно зависящих от n переменных, при $n \geq 1$ равно 2^{n-1} .

Представитель любого PN -типа БПФ из h букв реализуется каскадом из $h - 1$ двухвходовых элементов И и ИЛИ, каскадом из $h - 1$ МЭ и каскадом из двухвходовых элементов И—НЕ (ИЛИ—НЕ), в котором число элементов определяется соотношением [9]:

$$h - 1 \leq \vartheta \leq 2(h - 1).$$

В [9] показано, что если функция $\Phi(x_1, \dots, x_{n-1})$ реализуется бесповторной пороговой формулой, то этому классу принадлежит также формула $f(x_1, \dots, x_n) = \Phi(x_1, \dots, x_{n-1}) * x_n$, где знак $*$ соответствует операциям $\{\&, \vee\}$.

Отсюда следует, что произвольная (в том числе скобочная) БПФ может быть представлена в виде:

$$f(x_1, \dots, x_n) = ((\dots (x_1 * x_2) * x_3 \dots * x_{n-2}) * x_{n-1}) * x_n.$$

Бесповторные пороговые формулы n переменных с наибольшей суммой весов и порога имеют вид [9]:

$$\begin{aligned} f_1(x_1, \dots, x_n) &= \\ &= (\dots (x_1 x_2 \vee x_3) x_4 \vee x_5) x_6 \vee \dots \vee x_{n-1}) x_n \quad \text{при } n = 2k; \\ f_2(x_1, \dots, x_n) &= \\ &= (\dots (x_1 \vee x_2) x_3 \vee x_4) x_5 \vee \dots \vee x_{n-1}) x_n \quad \text{при } n = 2k + 1. \end{aligned}$$

Пороговая реализация функций f_1 и f_2 имеет вид: $(p_1, p_2, \dots, p_n; T)$, где $p_i = F_i$; $T = F_{n+1}$; p_i — вес i -й переменной, T — порог функции; F_i — i -е число ряда Фибоначчи, $F_1 = F_2 = 1$. Этот результат был независимо получен в [9, 18].

Для БПФ f_1 и f_2 справедливо соотношение [9]

$$T + \sum_{i=1}^n p_i = F_{n+3} - 1.$$

Верхние оценки параметров БПФ удовлетворяют соотношениям:

$$\max p_i \leq F_n, \quad \max T \leq F_{n+1},$$

где максимум берется по всем формулам класса.

Простой метод определения весов и порога для произвольной БПФ по построенному для нее линейному бинарному графу с максимальным числом путей изложен в разд. 15.5. При этом значение пометки Акерса входа i -й условной вершины равно p_i , а число нулевых путей в графе равно T . Этот метод впервые описан в [19].

1.5. Отношения покрытия

Будем считать, что функция $Q_2(X)$ покрывает функцию $Q_1(X)$, если на каждом входном наборе X выполняется соотношение $Q_2 \geq Q_1$. При этом отношение покрытия будем обозначать так же $(Q_2 \geq Q_1)$.

Пусть имеется выражение

$$f = Q_0! \Phi_1! \Phi_2 \vee Q_1! \Phi_1 \Phi_2 \vee Q_2 \Phi_1! \Phi_2 \vee Q_3 \Phi_1 \Phi_2.$$

Вынесение за скобки незначительно уменьшает число букв и не исключает инверсий функций Φ_j :

$$f = !\Phi_1 (Q_0! \Phi_2 \vee Q_1 \Phi_2) \vee \Phi_1 (Q_2! \Phi_2 \vee Q_3 \Phi_2).$$

1. В [3] показано, что если выполняются отношения

$$Q_0 \leq Q_1, Q_0 \leq Q_2, Q_0 \leq Q_3, Q_1 \leq Q_3, Q_2 \leq Q_3,$$

то выражение для f даже в ДНФ может быть существенно упрощено и оно не будет содержать инверсий функций Φ_j :

$$f = Q_0 \vee Q_1 \Phi_2 \vee Q_2 \Phi_1 \vee Q_3 \Phi_1 \Phi_2.$$

Пример 1.7. Если $Q_0 = 0$, $Q_1 = !x_1 x_2$, $Q_2 = x_1 !x_2$, $Q_3 = 1$, то приведенные соотношения выполняются и

$$f = !x_1 x_2 \Phi_2 \vee x_1 !x_2 \Phi_1 \vee \Phi_1 \Phi_2 = (!x_1 x_2 \vee \Phi_1) \Phi_2 \vee x_1 !x_2 \Phi_1.$$

2. При выполнении отношений вида

$$Q_0 \leq Q_1 \leq Q_2 \leq Q_3$$

может появиться возможность дальнейшего упрощения.

Пример 1.8. Если $Q_0 = 0$, $Q_1 = x_1 x_2$, $Q_2 = x_1 \vee x_2$, $Q_3 = 1$, то последние соотношения выполняются и

$$f = x_1 x_2 \Phi_2 \vee (x_1 \vee x_2) \Phi_1 \vee \Phi_1 \Phi_2 = (x_1 \vee \Phi_2) \Phi_1 \vee x_2 \Phi_2.$$

3. Если $Q_0 = 0$, $Q_1 = !Q$, $Q_2 = Q$, $Q_3 = 1$, то

$$f = !Q \Phi_2 \vee Q \Phi_1 = \text{MX}(\Phi_2, \Phi_1; Q),$$

где MX — функция мультиплексирования.

4. Если выполняются отношения

$$Q_0 = Q_1 \leq Q_2 \leq Q_3,$$

то

$$f = Q_1 \vee \Phi_1 (Q_2 \vee Q_3 \Phi_2).$$

5. Если функция f содержит только три типа фрагментов — Q_1 , Q_2 , Q_3 , для которых выполняются отношения покрытия, то при $Q_0 = 0$ справедливо, что $f = Q_1 \Phi_2 \vee \Phi_1 (Q_2 \vee Q_3 \Phi_2)$, а при $Q_0 = Q_1$ получается более простое выражение $f = Q_1 \vee \Phi_1 (Q_2 \vee Q_3 \Phi_2)$.

6. Если функция f содержит три типа фрагментов, для которых

$$Q_0 = 0 \leq Q_1 = Q_2 = Q \leq Q_3 = 1,$$

то функция f является мажоритарной функцией трех переменных:

$$f = (\Phi_1 \vee \Phi_2) Q \vee \Phi_1 \Phi_2 = \Phi_1 \# \Phi_2 \# Q.$$

7. Если функция f содержит только два типа фрагментов — Q_0 и Q_1 , то

$$f = Q_0 !\Phi \vee Q_1 \Phi = \text{MX}(Q_0, Q_1; \Phi).$$

8. При $Q_0 \leq Q_1$ справедливо, что $f = Q_0 \vee Q_1 \Phi$. Например, при $Q_0 = x_1 x_2 \leq Q_1 = x_1 \vee x_2$ справедливо, что $f = x_1 x_2 \vee (x_1 \vee x_2) \Phi = x_1 \# x_2 \# \Phi$. При $Q_0 \geq Q_1$ справедливо, что $f = Q_0 ! \Phi \vee Q_1$.

9. Выполнение отношения покрытия между функциями Q_0 и Q_1 позволяет их реализовывать совместно. Например, при $Q_0 = !x_1 x_2 \leq Q_1 = x_1 \oplus x_2 = Q_0 \vee x_1 !x_2$, а при $Q_0 = !x_1 x_2 \leq Q_1 = x_1 \vee x_2 = Q_0 \vee x_1$.

Более подробно отношения покрытия, введенные в [3], рассмотрены в разд. 7.2.2.

1.6. Метод совместной реализации булевых функций

Известен [20] «гарвардский» метод совместной реализации булевых функций, заданных таблицей истинности. Пусть в качестве примера требуется реализовать две БФУ — f_1 и f_2 , зависящие от переменных $x_1, \dots, x_n$. Суть метода состоит в том, что столбец значений одной из этих функций, например f_1 , переносится в левую часть ТИ, а функция f_2 рассматривается как не полностью определенная БФУ, заданная на 2^n входных наборах из 2^{n+1} наборов. Минимизируя эту функцию, в ряде случаев удастся получить формулу

$$f_2 = \Phi_1(f_1, x_1, \dots, x_n),$$

существенно более простую, чем формула

$$f_2 = \Phi_1(x_1, \dots, x_n).$$

Обратим внимание, что сложность совместной реализации зависит от того, какая из двух функций переносится в левую часть ТИ, что и было отмечено при участии автора настоящей работы в [20].

Построим формулы для f_1 и f_2 в отдельности и перенесем в левую часть «простейшую» из них. Такой подход будем использовать в качестве эвристического правила для выбора БФУ, подлежащей переносу.

Пример 1.9. Реализовать «гарвардским» методом БФУ f_1 и f_2 , заданные табл. 1.10.

Из табл. 1.10 следует, что

$$\begin{aligned} f_1 &= (x_1 \vee x_2) x_3 \vee x_1 x_2 = x_1 \# x_2 \# x_3, \\ f_2 &= !x_1 x_2 x_3 \vee x_1 (x_2 \oplus x_3). \end{aligned}$$

Считая, что функция f_1 простейшая, перенесем столбец f_1 в левую часть ТИ (табл. 1.11).

Т а б л и ц а 1.10

x_1	x_2	x_3	f_1	f_2	f_3	f_4	f_5	f_6
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	1
0	1	0	0	0	0	0	1	1
0	1	1	1	1	0	1	0	1
1	0	0	0	0	0	0	1	1
1	0	1	1	1	0	1	0	1
1	1	0	1	1	0	1	0	1
1	1	1	1	0	1	1	1	0

Т а б л и ц а 1.11

x_1	x_2	x_3	f_1	f_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	0

Минимизируя функцию f_2 с применением карты Карно, получим:

$$f_2 = f_1 (!x_1 \vee !x_2 \vee !x_3) = f_1!(x_1 x_2 x_3).$$

Предложим для решения той же задачи менее трудоемкий метод, суть которого состоит в записи уравнения

$$f = f_1 \oplus f_2$$

и решении его, оставляя в правой части «простейшую» формулу, например f_1 . При этом

$$f_2 = f_1 \oplus f.$$

Пример 1.10. Реализовать БФУ f_1 и f_2 из предыдущего примера с помощью предлагаемого метода.

Используя табл. 1.10, построим БФУ $f_3 = f_1 \oplus f_2 = x_1 x_2 x_3$. Таким образом, $f_2 = f_1 \oplus x_1 x_2 x_3$. Эта БФ содержит меньшее число логических операций по сравнению с БФ, найденной в предыдущем примере.

Пример 1.11. Реализовать параллельный одноразрядный двоичный сумматор (функции f_4 и f_5 табл. 1.10).

При этом

$$f_4 = (x_1 \vee x_2) x_3 \vee x_1 x_2, \quad f_5 = x_1 \oplus x_2 \oplus x_3,$$

а суммарное число логических операций равно шести.

Применяя «гарвардский» метод, получим:

$$f_4 = (x_1 \vee x_2) x_3 \vee x_1 x_2, \quad f_5 = !f_4 (x_1 \vee x_2 \vee x_3) \vee x_1 x_2 x_3$$

или

$$f_5 = x_1 \oplus x_2 \oplus x_3, \quad f_4 = x_1 x_2 \vee x_3 !f_5.$$

В первом случае число логических операций равно одиннадцати (после факторизации — девять), а во втором — шести.

Используя предлагаемый метод, запишем $f_6 = f_4 \oplus f_5$ (табл.1.10). Так как функция f_6 является «зеркальной» (разд. 10.7), то из [21] следует, что

$$f_6 = (x_1 \oplus x_2) \vee (x_1 \oplus x_3).$$

Так как $f_4 = f_6 \oplus f_5$, то

$$\Phi = x_1 \oplus x_2, f_5 = \Phi \oplus x_3, f_4 = (\Phi \vee (x_1 \oplus x_3)) \oplus f_5.$$

Эти соотношения содержат лишь пять логических операций, из которых четыре однотипны.

Предложенный метод, как и «гарвардский», может применяться не только для двух, но и для большего числа булевых формул.

Пример 1.12. Использовать предлагаемый метод для реализации системы БФ (СБФ):

$$f_1 = x_1 x_2, f_2 = x_1 \vee x_2, f_3 = x_1 \oplus x_2.$$

Построим для заданной системы ТИ и сформируем по ней БФ $f = f_1 \oplus f_2 \oplus f_3$, которая равна нулю. Из соотношения $f_1 \oplus f_2 \oplus f_3 = 0$, в частности, следует, что $f_3 = f_1 \oplus f_2$.

Предложенный подход может быть также использован и для упрощения внесения изменений в реализованную СБФ [22]. Еще один метод совместной реализации БФУ изложен в разд.16.8.

1.7. Новые соотношения для преобразований булевых функций

В [23] получены соотношения:

$$f(x_1, \dots, x_n) = (!x_i f(x_i = 0) \vee x_i !f(x_i = 1)) \oplus x_i;$$

$$f(x_1, \dots, x_n) = (!x_i !f(x_i = 0) \vee x_i f(x_i = 1)) \oplus !x_i.$$

Эти соотношения названы в [24] преобразованиями А р т ю х о в а—Ш а л ы т о.

Они неоднократно применяются в настоящей работе при решении различных задач логического проектирования. Рассмотрим несколько из таких задач.

1. На основе использования полученных преобразований может быть предложен подход для упрощения настройки мультиплексоров (МХ) и постоянных запоминающих устройств (ПЗУ), программируемых по таблицам истинности.

Предположим, что настройка нулем предпочтительнее, чем настройка единицей, а столбец значений реализуемой полностью определенной функции f содержит q_0 нулей и q_1 единиц.

Если $q_0 > q_1$, то целесообразно применять соотношения:

$$f_1 = f, f_1 = f \oplus 0. \quad (1.1)$$

Если $q_0 < q_1$, то целесообразно использовать соотношения:

$$f_1 = !f, f_1 = f \oplus 1. \quad (1.2)$$

Предположим, что $q_0 = q_1$, но число нулей в верхней и нижней половинах различно.

Если при этом число нулей в верхней половине больше, чем в нижней, то целесообразно применять соотношения:

$$f_1 = !x_1 f(x_1 = 0) \vee x_1 !f(x_1 = 1), f = f_1 \oplus x_1, \quad (1.3)$$

а в противном случае — соотношения:

$$f_1 = !x_1 !f(x_1 = 0) \vee x_1 f(x_1 = 1), f = f_1 \oplus !x_1. \quad (1.4)$$

Применительно к МХ этот подход впервые описан в [23] и использован в [25]. Из изложенного следует, что все приведенные выше соотношения могут быть реализованы схемой на рис.1.5.

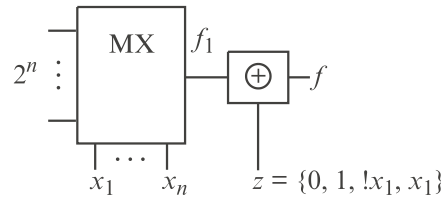


Рис. 1.5

Этот подход может быть применен для каждого из выходов ПЗУ с целью уменьшения «травмирования» этого устройства при программировании.

Пример 1.13. Реализовать мажоритарную функцию трех переменных «два и более из трех» $f(x_1, x_2, x_3) = |0001\ 0111|^T$ с помощью схемы на рис.1.5, в которой $n = 3$. Здесь T — символ операции «транспонирование».

При программировании МХ непосредственно по столбцу значений (соотношения (1.1)) число единиц равно четырём, а $z = 0$.

Так как в этом случае $q_0 = q_1$, а в верхней половине число нулей больше, чем в нижней, то программирование МХ выполним на основе соотношений (1.3). При этом $f_1 = |0001\ 1000|^T$, а $z = x_1$. Таким образом, в данном случае число единиц в коде настройки уменьшилось до двух.

2. Построить параллельный одноразрядный двоичный сумматор-вычитатель, в котором обеспечивается компромисс между однородностью используемых элементов и их числом.

Известно [26], что двоичный сумматор описывается системой БФ:

$$f_1 = x_1 (x_2 \vee x_3) \vee x_2 x_3, \quad f_2 = x_1 \oplus x_2 \oplus x_3,$$

а двоичный вычитатель — СБФ:

$$f_3 = !x_1 (x_2 \vee x_3) \vee x_2 x_3, \quad f_4 = x_1 \oplus x_2 \oplus x_3.$$

Наименее трудоемким является построение СБФ:

$$F_1 = (z \oplus x_1) (x_2 \vee x_3) \vee x_2 x_3, \quad F_2 = x_1 \oplus x_2 \oplus x_3,$$

где $z = \{0, 1\}$, которая, однако, не обеспечивает указанный компромисс.

Применяя к функции f_1 первое преобразование по переменной x_2 , получим:

$$f_1 = (!x_1 x_2 !x_3 \vee x_1 !x_2 x_3) \oplus x_2 = \Phi \oplus x_2.$$

Функция Φ является «зеркальной», и поэтому на основании [21]

$$\Phi = (x_1 \oplus x_2) (x_2 \oplus x_3).$$

Следовательно, $f_1 = (x_1 \oplus x_2) (x_2 \oplus x_3) \oplus x_2$.

Применяя к функции f_3 первое преобразование по переменной x_3 , получим:

$$f_3 = (!x_1 x_2 !x_3 \vee x_1 !x_2 x_3) \oplus x_3 = \Phi \oplus x_3.$$

Таким образом, сумматор-вычитатель может быть реализован СБФ:

$$F_1 = (x_1 \oplus x_2) (x_2 \oplus x_3) \oplus z, \quad F_2 = x_1 \oplus x_2 \oplus x_3,$$

где $z = \{x_2, x_3\}$.

Эта СБФ, полученная в [27], была использована в [28]. В этой работе заданное устройство реализуется схемой из пяти двухвходовых логических элементов, четыре из которых однотипны.

Обращая внимание на тот факт, что исходные формулы f_1 и f_3 отличаются на инверсию первой буквы, из изложенного выше можно записать:

$$F_1 = (z \oplus x_1 \oplus x_2) (x_2 \oplus x_3) \oplus x_2, \quad F_2 = x_1 \oplus x_2 \oplus x_3,$$

где $z = \{0, 1\}$.

По этой СБФ может быть построена схема из шести двухвходовых элементов, которая более проста, чем схема, приведенная в [29]. Полностью однородная схема из двухвходовых элементов И—НЕ (ИЛИ—НЕ) содержит более шести элементов.

3. Рассмотрим вопрос о применении предложенных преобразований с целью минимизации числа букв в БФ.

Пример 1.14. Булева функция «точно два из трех» $f(x_1, x_2, x_3) = |0001\ 0110|^T$ реализуется формулой:

— в базисе $\{\&, \vee, !\}$ из восьми букв, получаемой с использованием карты Карно,

$$f = !x_1 x_2 x_3 \vee x_1 (!x_2 x_3 \vee x_2 !x_3);$$

— в базисе $\{\&, \oplus\}$ из семи букв, получаемой на основе полинома Жегалкина [20],

$$f = x_1 (x_2 \oplus x_3 \oplus x_2 x_3) \oplus x_2 x_3;$$

— в базисе $\{\&, \vee, \oplus, !\}$ из шести букв, получаемой из первой БФ,

$$f = !x_1 x_2 x_3 \vee x_1 (x_2 \oplus x_3);$$

— в базисе $\{\&, \oplus, !\}$ из шести букв, получаемой из предыдущей формулы при замене операции $\vee$ на операцию $\oplus$ между ортогональными подформулами,

$$f = !x_1 x_2 x_3 \oplus x_1 (x_2 \oplus x_3);$$

— в базисе $\{\&, \vee, \oplus\}$ из пяти букв, получаемой с помощью разложения Рида (разд.3.1.1) [30] по переменной x_1 ,

$$f = x_1 (x_2 \vee x_3) \oplus x_2 x_3;$$

— в базисе $\{\&, \vee, \oplus, !\}$ из пяти букв, получаемой как с помощью разложения Рида по переменной $!x_3$, так и с помощью первого из предлагаемых преобразований, выполняемого последовательно по переменным x_2 и x_1 ,

$$f = (x_1 \vee x_2) !x_3 \oplus x_1 \oplus x_2.$$

Полученная формула может быть реализована каскадной схемой из двухвходовых элементов, в то время как предыдущая формула такой схемой не реализуется.

Пример 1.15. Булева функция $f(x_1, x_2, x_3, x_4) = |0100\ 0011\ 0011\ 1110|^T$ реализуется формулой:

— в базисе $\{\&, \vee, !\}$ из 13 букв, получаемой с применением карты Карно,

$$f = (!x_1 !x_2 x_4 \vee x_1 x_2) !x_3 \vee (!x_1 x_2 \vee x_1 !x_2 \vee x_2 !x_4) x_3;$$

— в базисе $\{\&, \oplus\}$ из 12 букв, получаемой на основе полинома Жегалкина,

$$f = x_1 ((x_2 \oplus x_3) \oplus x_2 x_3) (1 \oplus x_4) \oplus (1 \oplus x_1 \oplus x_2 \oplus x_3) x_4;$$

— в базисе $\{\&, \vee, \oplus, !\}$ из 11 букв, получаемой из первой БФ,

$$f = (!x_1 !x_2 x_4 \vee x_1 x_2) !x_3 \vee ((x_1 \oplus x_2) \vee x_2 !x_4) x_3;$$

— в базисе $\{\&, \vee, \oplus, !\}$ из 10 букв, получаемой из второй БФ,

$$f = x_1 ((x_2 \oplus x_3) \oplus x_2 x_3) !x_4 \oplus (!x_1 \oplus x_2 \oplus x_3) x_4;$$

— в базисе $\{\&, \vee, \oplus, !\}$ из 8 букв, получаемой с помощью разложения Рида по переменной $!x_1$,

$$f = !x_1 (!x_2 \oplus x_3) !x_4 \oplus ((x_2 \oplus x_3) \vee x_2 !x_4);$$

— в базисе $\{\&, \vee, \oplus, !\}$ из 6 букв, получаемой с помощью второго из предлагаемых преобразований, выполняемого по переменной $!x_1$,

$$f = !x_1 \oplus ((!x_1 \oplus x_3) x_4 \vee (x_2 \oplus x_3)).$$

Выводы

1. Определено количество типов неповторных булевых формул (ББФ) в базисе И, ИЛИ, НЕ в различных классификациях.

2. Выполнено табулирование представителей PN -типов этих формул при $h \leq 8$.

3. Определено рекуррентное соотношение для подсчета неизоморфных деревьев из двухвходовых элементов.

4. Рассмотрены покрывающие деревья и на их основе предложены модули, универсальные в указанном классе формул с простой структурой.

5. Рассмотрена PN -классификация ББФ в базисе И, ИЛИ, НЕ-РАВНОЗНАЧНОСТЬ, НЕ и выполнено их табулирование при $h \leq 4$.

6. Установлено, что ББФ в базисе И, ИЛИ, НЕ, существенно зависящие от всех переменных, имеют нечетный ранг.

7. Предложены методы подсчета числа единиц при формульном задании булевой функции.

8. Определены свойства неповторных пороговых формул и установлена их связь с числами Фибоначчи.

9. Определены отношения покрытия, выполнение которых позволяет записывать булевы формулы в форме, аналогичной разложению Шеннона, которая не содержит переменных с инверсиями.

10. Предложен метод совместной реализации булевых функций, обладающий меньшей трудоемкостью по сравнению с «гарвардским» методом.

11. Получены новые соотношения для преобразования булевых функций и на примерах показана целесообразность их применения при решении ряда задач логического проектирования.

Л и т е р а т у р а

1. Яблонский С. В. Введение в дискретную математику. М.: Наука, 1979.
2. Яблонский С. В., Гаврилов Г. П., Кудрявцев В. Б. Функции алгебры логики и классы Поста. М.: Наука, 1966.
3. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Настраиваемые модули для управляющих логических устройств. Л.: Энергоиздат, 1981.
4. Лазарев В. Г., Наумчук О. Ф., Саввин Г. Г., Сагалович Ю. Л. О переписи типовых релейно-контактных схем автоматической телефонии // Проблемы передачи информации. 1963. № 8.
5. Шеннон К. Э. Работы по теории информации и кибернетике. М.: Изд-во иностр. лит., 1963.
6. Кнут Д. Искусство программирования для ЭВМ. Т. 3. Сортировка и поиск. М.: Мир, 1978.
7. Артюхов В. Л. Логические методы синтеза дискретных систем. Л.: Ин-т повышения квалификации руководящих работников и специалистов судостроительной промышленности (ИПК СП), 1974.
8. Риордан Дж. Введение в комбинаторный анализ. М.: Изд-во иностр. лит., 1963.
9. Артюхов В. Л., Розенблум Л. Я., Шалыто А. А. Логические возможности некоторых типов каскадных структур // Сети связи и дискретные устройства управления. М.: Наука, 1976.
10. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. О классификации бесповторных булевых функций / Теория автоматов и ее приложения. Материалы советско-болгарского семинара. М.: Ин-т проблем передачи информации, 1973.
11. Попов Ю. А., Воронин А. Т., Сладков А. Б. Вопросы проектирования схем с высоким уровнем интеграции на основе КНС технологии // Дискретные системы. Т.1. Симпозиум IFAC. Рига: Зинатне, 1974.
12. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Вопросы выбора и применения многофункциональных логических модулей // Дискретные системы. Т.1. Симпозиум IFAC. Рига: Зинатне, 1974.
13. Levy S. Y., Winder R., Mott T. H. A note on tributary switching networks // IEEE Trans. on Computers. 1964. N 2.
14. Воробьев Н. Н. Числа Фибоначчи. М.: Наука, 1978.
15. Чистов В. П., Битюцкий В. П. Функциональная полнота в ленточных однородных структурах // Изв. АН СССР. Техн. кибернетика. 1971. № 3.
16. Пупырев Е. И. Использование аксиомы теории вероятностей в задачах анализа булевых функций // Автоматика и телемеханика. 1976. № 7.
17. Шалыто А. А. Реализация алгоритмов судовых управляющих логических систем при использовании микропроцессорной техники. Л.: ИПК СП, 1988.
18. Fridman A. D., Menon P. R. Theory and design of switching circuits. Computer Science Press., Inc., 1975.
19. Кондратьев В. Н., Шалыто А. А. Реализация булевых функций одним линейным арифметическим полиномом с маскированием // Автоматика и телемеханика. 1996. № 1.
20. Поспелов Д. А. Логические методы анализа и синтеза схем. М.: Энергия, 1974.
21. Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1283744 // Бюл. изобр. 1987. № 2.
22. Шалыто А. А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
23. Артюхов В. Л., Шалыто А. А. Судовые управляющие логические системы. Л.: ИПК СП, 1983.
24. Артюхов В. Л., Шалыто А. А. Реализация булевых формул однородными мультиплексорными и мажоритарными каскадами // Изв. РАН. Теория и системы управления. 1996. № 5.
25. Артюхов В. Л., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1149244 // Бюл. изобр. 1985. № 13.

26. Мищенко В. А., Козюминский В. Д., Семашко А. Н. Многофункциональные автоматы и элементная база цифровых ЭВМ. М.: Радио и связь, 1981.
27. Шалыто А. А. Программная реализация алгоритмов логического управления судовыми системами. Л.: ИПК СП, 1989.
28. Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1290290 // Бюл. изобр. 1987. № 6.
29. Шалыто А. А. Комбинационный двоичный сумматор-вычитатель. А. с. СССР № 1264166 // Бюл. изобр. 1986. № 38.
30. Reed I. S. Class of multiple error correcting codes and decoding scheme // IRE Trans. Inform. Theory. 1954. IT-4.

Глава 2

Формульный метод синтеза комбинационных схем из произвольных логических элементов

Одной из важнейших задач в области логического синтеза является задача построения комбинационных схем, реализующих булевы функции, из произвольных априори известных логических элементов.

Эта задача является достаточно сложной, и поэтому рассмотрим вопрос о реализации одной булевой функции.

Возможны два подхода к построению комбинационных схем: от выхода к входам и от входов к выходу.

Первый из этих подходов, названный мультиплексорным методом, рассмотрен в гл. 3. В настоящей главе излагается второй подход, названный *формульным методом*, который существенно отличается от предложенного в [1]. В формульном методе реализуемая булева функция задана нормальной булевой формулой, которая может содержать скобки произвольной глубины, а ее двухместные операции подчиняются сочетательному закону.

Выполнение этого закона позволяет весьма просто «разделять» формулу на подформулы [2]. При этом будут рассматриваться два базиса логических операций: И, ИЛИ, НЕ и И, ИЛИ, НЕРАВНОЗНАЧНОСТЬ, НЕ. Если это особо не оговаривается, то в дальнейшем будем предполагать, что применяются формулы, записанные в первом базисе, а прямые и инверсные выходы источников информации равнодоступны.

В отличие от традиционного подхода, используемого в логическом синтезе, в качестве основной метрики будем применять не число входных переменных n в БФУ, а число букв h в БФ при фиксированном базисе логических операций, которые используются при ее записи. При этом повторная БФ, зависящая от n переменных и содержащая h букв, может первоначально реализовываться как неповторная БФ той же структуры, которая зависит от h переменных, а затем за

счет отождествлений соответствующих переменных осуществляется обратный переход к заданной формуле. Например, если требуется реализовать БФ $f = (x_1 \vee x_2) x_3 \vee x_1 x_2$, для которой $n = 3$, $h = 5$, то первоначально вместо нее реализуется ББФ $f = (z_1 \vee z_2) z_3 \vee z_4 z_5$ с $n = h = 5$, а затем выполняются следующие отождествления переменных: $z_1 = z_4 = x_1$, $z_2 = z_5 = x_2$, $z_3 = x_3$.

Основная идея предлагаемого метода состоит в реализации произвольной БФ в рассматриваемом базисе из h букв схемой из модулей, универсальных в классе произвольных БФ в том же базисе из q и менее букв, где $h \geq q$. В дальнейшем будем называть такие модули q -универсальными. Из изложенного следует, что для универсальности в классе произвольных БФ требуется, чтобы модуль был универсален только в классе ББФ в том же базисе из фиксированного числа букв, т. е. мог реализовать путем настройки представителя каждого типа ББФ в этом базисе и из этого числа букв в выбранной классификации.

Рассматриваются два типа универсальных модулей. Модули первого типа названы *настраиваемыми*, а второго — *виртуальными*.

Для модулей первого типа характерно, что каждый из них реализует порождающую функцию, объединяющую представителей всех типов ББФ в выбранной классификации из определенного числа букв. После этого модуль может быть настроен на реализацию любого из указанных представителей типов. Метод построения таких модулей изложен в гл. 7.

Для модулей второго типа характерно, что представитель каждого типа ББФ в выбранной классификации из определенного числа букв реализуется с помощью известных или предлагаемых в настоящей работе методов схемой из заданных произвольных элементов. При этом все построенные схемы образуют «таблицу настройки» виртуального модуля.

После разработки метода синтеза схем на настраиваемых q -универсальных модулях он сначала может быть распространен на синтез схем из настраиваемых многофункциональных модулей, которые, обладая q -универсальностью, реализуют также представителей некоторых типов формул из числа букв большего q (эти модули могут не проектироваться специально как многофункциональные, являясь ими по сути), а затем на синтез схем из виртуальных модулей и тем самым на синтез схем из произвольных априори известных логических элементов.

Предлагаемый метод в общем случае не гарантирует оптимальности схем ни по сложности, ни по быстродействию, но в отличие от известных методов обеспечивает линейную зависимость числа весьма сложных элементов в схеме от числа букв в реализуемой БФ. Этот метод был впервые описан в [3—7], а комплекс программ, его реализующих, — в [8].

2.1. Синтез схем из модулей, универсальных в классе формул

2.1.1. Реализация булевых формул схемами из положительно монотонных q -универсальных модулей

Будем использовать PN -классификацию и представлять реализуемую формулу и формулы, реализуемые модулем путем настройки, в виде структурных полиномов. В [2, 9] показано, что оптимальная стратегия, обеспечивающая минимальную элементную сложность реализации ББФ, состоит в том, что синтез проводится от входов к выходу и у применяемого модуля на каждом шаге реализации формулы должно быть использовано максимально возможное число информационных входов из q возможных. Таким образом, в данном случае отдельные шаги при движении от входов к выходу схемы независимы и, принимая оптимальное решение на каждом шаге, мы не ухудшаем окончательного результата.

В [9] доказано, что эта стратегия обеспечивает выполнение следующего соотношения, определяющего число модулей в схеме:

$$\left\lceil \frac{h-1}{q-1} \right\rceil \leq L(h, q) \leq \left\lceil \frac{2(h-1)}{q} \right\rceil, \quad (2.1)$$

где $\lceil a \rceil$ — символ округления числа a до ближайшего большего целого.

Нижняя оценка числа шагов совпадает с нижней оценкой в соотношении (2.1), а верхняя оценка числа шагов равна верхней оценке в соотношении (2.1), умноженной на константу $\lceil q/2 \rceil$.

Обратим внимание, что при $q = 2$

$$L(h, 2) = h - 1.$$

Пример 2.1. Реализовать на 3-универсальных модулях формулу

$$f = (!x_1 x_2 \vee x_1 !x_2) x_3 \vee x_4 x_5 (x_6 \vee x_7) \vee x_8.$$

В данном случае $h = 10$, $q = 3$, и поэтому $5 \leq L(10, 3) \leq 6$. Из приведенного соотношения следует, что произвольная (в том числе и заданная) БФ в базисе $\{\&, \vee, !\}$ из десяти букв в зависимости от ее структуры реализуется схемой из пяти либо из шести таких модулей.

Представим заданную БФ в виде полинома $((2+2)1+2)(1+1)+1$.

Процедура реализации состоит в следующем.

1. Формула не содержит подформул из трех букв.
2. Выделяем подформулу 2. Остаток: $((1+2)+2)(1+1)+1$.
3. Выделяем подформулу $1+2$. Остаток: $(2+2)(1+1)+1$.
4. Остаточная формула не содержит подформул из трех букв.
5. Выделяем подформулу 2. Остаток: $(1+2)(1+1)+1$.

6. Выделяем подформулу $1 + 2$. Остаток: $1(1 + 1) + 1$.
 7. Выделяем подформулу $1(1 + 1)$. Остаток: $1 + 1$.
 8. Остаточная формула содержит менее трех букв.
 9. Выделяем подформулу $1 + 1$. Остаток: 1 .
- Построенная схема из шести модулей приведена на рис.2.1.

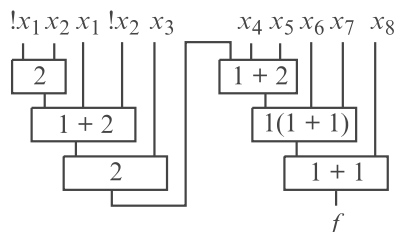


Рис. 2.1

Структура заданной БФ такова, что при $q = 3$ у трех из шести модулей удалось применить только два информационных входа из трех, что привело к построению схемы с числом модулей, определяемым верхней оценкой.

Пример 2.2. Реализовать формулу, рассмотренную в примере 2.1, на 4-универсальных модулях.

В этом случае $h = 10$, $q = 4$, и поэтому $3 \leq L(10, 4) \leq 5$.

Из сопоставления этого неравенства с приведенным в предыдущем примере следует, что при $h = 10$ усложнение модуля ($q = 4$ вместо $q = 3$) резко уменьшает оценку сложности: нижняя оценка при $q = 3$ становится верхней при $q = 4$.

Заданная БФ реализуется в данном случае следующим образом.

1. Выделяем подформулу $2 + 2$. Остаток: $(2 + 2)(1 + 1) + 1$.
2. Выделяем подформулу $2 + 2$. Остаток: $1(1 + 1) + 1$.
3. Выделяем подформулу $1(1 + 1) + 1$. Остаток: 1 .

Структура заданной формулы такова, что при $q = 4$ у всех трех модулей удалось применить все четыре информационных входа, что привело в этом случае к построению схемы с числом модулей, определяемым нижней оценкой. Таким образом, по сравнению с предыдущим примером усложнение модуля привело к двухкратному сокращению числа модулей в схеме. При этом, однако, нельзя сказать, что суммарная элементная сложность схемы уменьшилась, так как рассматриваемый модуль может быть реализован схемой из двух 3-универсальных модулей. Однако существует показатель, значение которого свидетельствует о том, что для заданной БФ применение 4-универсального модуля более эффективно. Это суммарное число логических выводов (внешние выводы без учета выводов для подачи питания) в модулях схемы. При $q = 4$ значение этого показателя равно $3 \cdot 7 = 21$, в то время как при $q = 2$ и $q = 3$ этот показатель равен $9 \cdot 4 = 36$ и $6 \cdot 5 = 30$ соответственно.

Пример 2.3. Реализовать формулу, рассмотренную в примере 2.1, на 5-универсальных модулях.

В этом случае $h = 10$, $q = 5$ и $3 \leq L(10, 5) \leq 4$. Из сравнения этого соотношения с приведенным в предыдущем примере следует, что за счет усложнения модуля верхняя оценка несколько снизилась, а нижняя не изменилась. Так как при $q = 4$ заданную ББФ удалось реализовать по нижней оценке, то реализовывать ее при $q = 5$ нецелесообразно.

Пример 2.4. Расширим базис двухместных операций, введя в него операцию НЕРАВНОЗНАЧНОСТЬ. При этом БФУ, рассмотренная в примере 2.1, может быть представлена в виде:

$$f = ((x_1 \oplus x_2) x_3 \vee x_4 x_5) (x_6 \vee x_7) \vee x_8.$$

В данном случае число букв уменьшилось до восьми. При использовании 3-универсального в базисе $\{\&, \vee, \oplus, !\}$ модуля выполняется соотношение $4 \leq L(8, 3) \leq 5$. Представим полученную ББФ в виде структурного полинома $((1 \oplus 1) 1 + 2)(1 + 1) + 1$. При этом процедура реализации состоит в следующем.

1. Выделяем подформулу $(1 \oplus 1) 1$. Остаток: $(2 + 2)(1 + 1) + 1$.
2. Остаточная формула не содержит подформулы из трех букв.
3. Выделяем подформулу 2. Остаток: $(1 + 2)(1 + 1) + 1$.
4. Выделяем подформулу $1 + 2$. Остаток: $1(1 + 1) + 1$.
5. Выделяем подформулу $1(1 + 1)$. Остаток: $1 + 1$.
6. Остаточная формула содержит менее трех букв.
7. Выделяем подформулу $1 + 1$. Остаток: 1.

Таким образом, рассматриваемая формула реализуется по верхней оценке сложности. Суммарное число логических выводов в этом случае $5 \cdot 6 = 30$. Следовательно, по этому критерию расширение базиса в данном случае нецелесообразно.

2.1.2. Минимизация числа q -универсальных модулей в схемах

Из соотношения (2.1) следует, что сокращение числа букв h при $q = \text{const}$, вообще говоря, приводит к сокращению числа модулей, однако для конкретных скобочных формул это может оказаться не так. Например, формула $f = x_1(x_2 x_3 x_4 \vee x_2 x_5 x_6 \vee x_7 x_8)$, содержащая девять букв, реализуется схемой из четырех 3-универсальных модулей, в то время как формула $f = x_1(x_2(x_3 x_4 \vee x_5 x_6) \vee x_7 x_8)$, получающаяся в результате минимизации предыдущей формулы и содержащая восемь букв, реализуется схемой из пяти таких модулей.

В [2] доказано, что, для того чтобы в результате минимизации числа букв в БФ от h до h число модулей в схеме не увеличивалось, должно выполняться соотношение

$$h_1 \leq \left\lceil 1 + \frac{h-1}{2} \frac{q}{q-1} \right\rceil, \quad (2.2)$$

а для того чтобы число модулей в схеме сокращалось, должно выполняться соотношение

$$h_1 \leq \left\lceil 1 + \frac{h-1}{2} \frac{q}{q-1} \right\rceil - \left\lfloor \frac{q}{2} \right\rfloor. \quad (2.3)$$

Таким образом, для гарантированного уменьшения числа модулей в схеме при больших h и $h \gg q$ число букв в формуле должно быть сокращено почти в два раза. При $h = 9$, $q = 3$ для уменьшения числа модулей должно выполняться неравенство $h_1 \leq 5$.

2.1.3. Реализация не полностью определенных булевых функций схемами из q -универсальных модулей

Получим на основании соотношения (2.1) оценку сложности реализации не полностью определенных БФУ. В [10] доказано, что для БФУ n переменных, равной нулю на t_0 входных наборах и единице — на t_1 входных наборах, может быть построена БФ в базисе $\{\&, \vee, !\}$ с числом букв

$$h \leq 2t_1 + t_0 - 2. \quad (2.4)$$

Таким образом, если

$$n > 2t_1 + t_0 - 2,$$

то по крайней мере от $(n - 2t_1 - t_0 + 2)$ переменных БФУ зависит несущественно.

В [11] доказано, что для БФУ, заданной на t входных наборах, может быть построена БФ в указанном базисе с числом букв

$$h \leq \frac{3}{2}t - 2 \text{ при } t \geq 2. \quad (2.5)$$

Таким образом, из соотношений (2.1), (2.4), (2.5) следует, что

$$L \leq \min \left(\left\lceil \frac{2(2t_1 + t_0 - 3)}{q} \right\rceil, \left\lceil \frac{3(t - 2)}{q} \right\rceil \right). \quad (2.6)$$

Пример 2.5. Реализовать не полностью определенную БФУ, заданную табл.2.1, на 3-универсальных модулях.

Т а б л и ц а 2.1

x_1	x_2	x_3	x_4	x_5	f
1	0	0	0	0	1
0	1	1	0	1	1
0	0	1	0	1	1
0	0	1	0	0	1
0	0	0	1	1	0
0	1	1	1	0	0
1	0	1	0	1	0
1	0	1	1	0	0

В данном случае $t_1 = t_0 = 4$, $t = 8$, $q = 3$, и поэтому из соотношения (2.6) следует, что $L \leq 6$. Для заданной БФУ с помощью алгоритма [10] может быть построена формула $f = x_1 ! x_3 \vee ! x_1 x_3 ! x_4$, которая формульным методом реализуется схемой из двух модулей.

2.1.4. Реализация булевых формул схемами из положительно монотонных модулей, универсальных в классе дизъюнктивных нормальных форм из q букв

Метод, изложенный в предыдущем разделе, может быть использован и в данном случае, однако уменьшение функциональных возможностей модулей приводит к повышению верхней оценки сложности. В [2] доказано, что для таких модулей справедливо соотношение

$$\left\lceil \frac{h-1}{q-1} \right\rceil \leq L_1(h, q) \leq \left\lceil \frac{h-1}{[(q/2)+1]} \right\rceil + \frac{c}{2}, \quad (2.7)$$

где c — число скобок в формуле.

Пример 2.6. Реализовать формулу, рассмотренную в примере 2.1, на модулях рассматриваемого класса при $q = 3$.

В данном случае $h = 10$, $q = 3$, и поэтому $5 \leq L_1(10, 3) \leq 8$, а заданная формула реализуется следующим образом.

1. Заданная БФ не содержит подформул, являющихся ДНФ из трех букв.
2. Выделяем подформулу 2. Остаток: $((1+2)1+2)(1+1)+1$.
3. Выделяем подформулу 1+2. Остаток: $(2+2)(1+1)+1$.
4. Остаточная формула не содержит подформул, являющихся ДНФ из трех букв.
5. Выделяем подформулу 2. Остаток: $(1+2)(1+1)+1$.

6. Выделяем подформулу $1 + 2$. Остаток: $1(1 + 1) + 1$.
 7. Остаточная формула не содержит подформул, являющихся ДНФ из трех букв.
 8. Выделяем подформулу $1 + 1$. Остаток: $2 + 1$.
 9. Выделяем подформулу $2 + 1$. Остаток: 1 .
- Заданная БФ реализуется схемой из шести рассматриваемых модулей.

Таким образом, эта БФ обладает такой структурой, что уменьшение функциональных возможностей модулей по сравнению с 3-универсальными модулями не привело к усложнению схемы.

Комплекс программ для выполнения синтеза в базе рассмотренных модулей, созданный при участии автора, описан в [36].

2.1.5. Реализация булевых формул схемами из немонотонных q -универсальных модулей

Если в качестве представителей типов формул, объединяемых в модуль, применяются немонотонные формулы, то изложенный в разд. 2.1.1 метод должен быть модифицирован. В этом случае синтез выполняется в два этапа.

На первом этапе представители типов формул, реализуемых модулем путем настройки, заменяются однотипными положительно монотонными формулами и от входов к выходу выполняется синтез с помощью метода, изложенного в разд. 2.1.1.

На втором этапе, который проводится от выхода к входам, выполняется корректировка схемы, построенной на первом этапе, с целью учета инверсий в представителях типов формул, реализуемых модулем. Для этого строится схема, найденная на первом этапе, без указания формул, которые реализуют модули. После этого формулы для пометки модулей выбираются на основе следующего правила.

Тип формулы выходного модуля, найденный на первом этапе, не изменяется, а выбор типов формул всех остальных модулей в схеме выполняется по правилу, названному основным: если от выхода рассматриваемого модуля до выхода схемы имеется четное число инверсий, то тип формулы, реализуемой этим модулем, не изменяется, а если число инверсий нечетно — тип формулы изменяется на дополнительный с точностью до расстановки инверсий и совпадает с одной из формул, выполняемых модулем за счет настройки.

Второй этап может быть выполнен всегда, так как q -универсальный модуль для любой формулы, реализуемой путем настройки, выполняет также и дополнительную ей формулу в рассматриваемой классификации. Для оценки числа шагов на втором этапе можно использовать соотношение (2.1). При этом точное число шагов на этом этапе равно числу модулей, определенных на первом этапе.

Приведенное выше правило распространяется также и на входные переменные.

Если на некотором шаге настройка модуля может быть выполнена неоднозначно, то применим следующее дополнительное эвристическое правило: будем выбирать такую настройку модуля, при которой изменяется настройка минимального числа модулей, подключенных к его входам.

При выполнении этого правила, если имеется возможность, то настройка модуля должна выбираться так, чтобы для модулей, связанных со входами схемы, число инверторов на этих входах было минимальным. Применение какого-либо другого правила влияет только на число инверторов на входах схемы и не изменяет числа модулей, определяемого соотношением (2.1).

Пример 2.7. Реализовать формулу, рассмотренную в примере 2.1, на 3-универсальных модулях, каждый из которых может быть настроен на реализацию формул:

$$z_1 = !ab \vee !c, z_2 = (!a \vee !b)!c,$$

$$z_3 = !a \vee !b \vee !c, z_4 = abc.$$

Этот модуль реализует также следующих представителей P -типов формул из двух букв: $!a!c, !ab, ab, !a \vee !b, b \vee !c$.

Первый этап построения схемы описан в примере 2.1. Выполним второй этап в обратном порядке.

6. Настроим шестой модуль на формулу $b \vee !c$.

5. Так как от выхода пятого модуля до выхода схемы нет инверторов, то сохраним тип настройки этого модуля: $!c (!a \vee !b)$.

4. Так как от выхода четвертого модуля до выхода схемы имеет-ся один инвертор, то заменим настройку этого модуля на дополнительную: $!c (!a \vee !b)$.

3. Так как от выхода третьего модуля до выхода схемы имеет-ся два инвертора, то, во-первых, сохраним тип настройки модуля, а во-вторых, сократим число инверсных переменных, подключенных ко входам модуля. Это обеспечивается при настройке ab .

2. Так как от выхода второго модуля до выхода схемы имеет-ся два инвертора, то, во-первых, сохраним тип настройки модуля, а во-вторых, сократим число инверсных переменных, подключенных ко входам модуля. Это обеспечивается при настройке $!c \vee b !a$.

1. Так как от выхода первого модуля до выхода схемы имеет-ся только два инвертора, то, во-первых, сохраним тип настройки модуля, а во-вторых, сократим число инверсных переменных, подключенных ко входам модуля. Это обеспечивается при настройке $!a b$.

Для каждого из входов схемы проверим выполнение основного правила с целью реализации заданной формулы. При этом восьмой, девятый и десятый входы схемы должны быть проинвертированы. Искомая схема приведена на рис. 2.2.

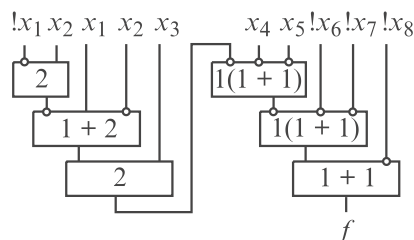


Рис. 2.2

Выполним верификацию построенной схемы. Для этого запишем по ней формулу, сопоставляя с каждым элементом пару круглых скобок, и приведем ее к нормальному виду, используя правило, предложенное в [2]: подсчитать число инверсий над каждой буквой и каждым знаком операции. Если число инверсий четно, то в формулу записывается буква без инверсии, а знак операции не изменяется. Если число инверсий нечетно, то буква записывается с инверсией, а знак изменяется на дополнительный: операция И изменяется на операцию ИЛИ, и наоборот. В данном случае

$$f = ((!x_1 x_2 \vee x_1 !x_2) x_3 \vee x_4 x_5) (x_6 \vee x_7) \vee x_8.$$

Таким образом, заданная формула реализована корректно.

2.1.6. Реализация булевых формул схемами из немонотонных q -универсальных модулей, использующих P -классификацию

Если модуль реализует путем настройки немонотонные формулы, но он q -универсален при использовании P -классификации, то синтез вновь становится одноэтапным, а на входах схемы нет необходимости устанавливать инверторы. При этом все модули соединяются друг с другом по прямым входам, а немонотонные формулы применяются лишь для того, чтобы не устанавливать инверторы на входах схемы. Соотношение (2.1) выполняется и для этой разновидности модулей.

2.2. Оценка эффективности многофункциональных логических модулей и реализация булевых формул схемами из этих модулей

До появления работы [2] многофункциональные логические модули (МЛМ) рассматривались как класс элементов, принципиально отличающийся от других классов логических элементов, что отражалось в обозначениях соответствующих микросхем.

В [2] автором было высказано утверждение, что любой логический элемент, имеющий более одного входа, является многофункциональным, так как он путем настройки реализует более одной функции.

Вопросы логической эффективности элементов рассматривались в [12—14]. Однако показатель эффективности (число подфункций в порождающей функции элемента), использованный в этих работах, не позволяет установить связь между сложностью элементов и числом этих элементов, требующихся для реализации БФ в используемом базисе из h букв.

Этого недостатка лишен коэффициент логической эффективности, или коэффициент многофункциональности, предложенный автором и определяемый соотношением [15]:

$$k_m = \sum_{i=1}^m \frac{\Phi(i)}{T_1(i)}, \quad (2.8)$$

где m — максимальное число букв, от которого модуль путем настройки реализует неповторные в рассматриваемом базисе формулы; $\Phi(i)$ — число PN -типов неповторных в том же базисе формул из i букв, реализуемых модулем путем настройки; $T_1(i)$ — число PN -типов неповторных в этом базисе формул из i букв, существующих теоретически (гл. 1).

Если модуль имеет $M \geq m$ входов, то физический смысл введенного коэффициента состоит в том, что при реализации большого массива формул в среднем в каждом из этих модулей k_m входов будут применяться как информационные, а остальные — как настроечные.

Для q -универсальных модулей $k_m = q$. При этом соотношение (2.1) связывает величину k_m с величиной L .

Предположим, что задан МЛМ, характеризующийся величиной k_m , который также q -универсален ($q \leq k_m$). Тогда

$$\left\lceil \frac{h-1}{m-1} \right\rceil \leq L_2(h, q) \leq \left\lceil \frac{2(h-1)}{q} \right\rceil \quad (2.9)$$

или в среднем:

$$\left\lceil \frac{h-1}{k_m-1} \right\rceil \lesssim L_3(h, k_m) \lesssim \left\lceil \frac{2(h-1)}{k_m} \right\rceil, \quad (2.10)$$

где $\lesssim$ — знак операции «приблизительно меньше».

При синтезе схем из МЛМ должна использоваться та же стратегия, что и при синтезе схем из универсальных модулей: на каждом шаге у каждого модуля должно применяться максимально возможное число информационных входов. Однако использование всех функциональных возможностей МЛМ весьма трудоемко, так как в этом случае не любая подформула из h_1 букв ($q < h_1 \leq m$) может быть реализована одним таким модулем.

При этом отметим, что любая подформула из q и менее букв реализуется одним таким модулем всегда. Таким образом, если для q -универсальных модулей нерезультативные шаги определяются структурой реализуемой БФ, то для МЛМ появляется также и другая причина — неуниверсальность модуля при $h_1 > q$.

Из изложенного следует, что для упрощения синтеза при фиксированном числе логических выводов должен проектироваться модуль не с максимальной величиной k_m , а с максимальной величиной q .

Пример 2.8. Определить значения k_m и q для элемента, структура которого описывается формулой $z = ab \vee cd$.

В этом случае $\Phi(1) = \Phi(4) = 1$, $\Phi(2) = \Phi(3) = 2$, $T_1(1) = 1$, $T_1(2) = 2$, $T_1(3) = 4$, $T_1(4) = 10$, и поэтому

$$k_m = \frac{1}{1} + \frac{2}{2} + \frac{2}{4} + \frac{1}{10} = 2.6; \quad q = 2.$$

Таким образом, можно считать, что этот элемент при реализации большого массива формул имеет как бы 2.6 информационных и 1.4 настроечных входов.

Этот элемент имеет пять логических выводов. В [2] показано, что при этом числе логических выводов может быть построен модуль с $q = 3$.

Пример 2.9. Определить значения k_m и q для микросхемы К1ЛР333, описываемой формулой

$$f = !(x_1 x_2 \vee x_3 x_4 \vee x_5 x_6 \vee x_7 x_8 x_9).$$

В [1] показано, что в данном случае $q = 3$, так как

$$k_m = \frac{1}{1} + \frac{2}{2} + \frac{4}{4} + \frac{8}{10} + \frac{10}{24} + \frac{12}{66} + \frac{9}{180} + \frac{4}{522} + \frac{1}{1532} = 4.46.$$

Из изложенного следует, что в этой микросхеме десять логических выводов применяются крайне неэффективно, так как известен 3-универсальный модуль с пятью логическими выводами [2].

Пример 2.10. Определить значения k_m и q для модуля, описываемого системой булевых формул:

$$\begin{aligned} f_1 = & x_1 x_2 x_3 x_4 x_5 x_6 x_7 !x_8 !x_9 !x_{10} \vee (x_1 x_2 \vee x_3 x_4 \vee x_5 x_6 \vee \\ & \vee x_7 x_8) x_9 !x_{10} \vee (x_1 x_2 x_3 x_4 \vee x_5 x_6 x_7 x_8) !x_9 x_{10} \vee \\ & \vee (x_1 x_2 x_3 \vee x_4 x_5 x_6 \vee x_7 \vee !x_8) x_9 x_{10}; \\ f_2 = & !f. \end{aligned}$$

Этот модуль, выпускаемый промышленностью в виде микросхемы К1ЖЛ081, в свое время (конец 60-х годов) обладал наивысшим уровнем интеграции среди логических микросхем, выпускаемых в мире [16]. Анализ функциональных возможностей этого элемента, выполненный в [17, 18], показал, что

$$k_m = \frac{1}{1} + \frac{2}{2} + \frac{4}{4} + \frac{10}{10} + \frac{24}{24} + \frac{58}{66} + \frac{50}{180} + \frac{26}{522} = 6.21; \quad q = 5.$$

При этом также было установлено, что в классе ДНФ этот элемент имеет

$$k_m = \frac{1}{1} + \frac{2}{2} + \frac{3}{3} + \frac{5}{5} + \frac{7}{7} + \frac{9}{15} + \frac{8}{22} = 6.964; \quad q = 6.$$

При этом отметим, что авторы этого элемента не знали всех его функциональных возможностей и при его создании ориентировались на то, чтобы модуль реализовал путем настройки 22 формулы из восьми букв, наиболее часто встречающиеся (по их мнению) в алгоритмах управления судовыми техническими средствами [16]. Ограниченность такого подхода и привела автора к разработке метода, излагаемого в настоящей главе.

Рассмотренный модуль имеет 12 логических выводов. В [19] показано, что этим числом выводов обладают в сумме 3- и 4-универсальные модули, которые образуют виртуальный модуль с $q = 6$.

Пример 2.11. Реализовать формулу из примера 2.1 схемой из микросхем К1ЖЛ081 (таблица настройки этой микросхемы приведена в [18]).

В данном случае $h = 10$, $m = 8$, $q = 5$, и поэтому из соотношения (2.9) следует, что $2 \leq L_2(10, 5) \leq 4$.

Так как эта микросхема путем настройки реализует в том числе и БФ с инверсиями, то синтез выполняется в два этапа.

Первый этап состоит в следующем.

1. В заданной БФ отсутствуют подформулы из восьми букв.
2. Подформула из семи букв $(2 + 2) 1 + 2$ не может быть реализована одной микросхемой.
3. В заданной БФ отсутствуют подформулы из шести букв.
4. Подформула из пяти букв $(2 + 2) 1$ реализуется одной микросхемой, так как она 5-универсальна. Остаток: $(1 + 2) (1 + 1) + 1$.

5. Остаток из шести букв реализуется одной микросхемой. Остаток: 1.

Таким образом, из-за неуниверсальности модуля при $q > 5$ из пяти шагов три не являются результативными.

Второй этап выполняется в обратном порядке.

2. Тип формулы, реализуемой последним элементом, не изменяется. В качестве представителя этого типа таблица настройки содержит формулу $f = (!g \vee !i !j) (!l \vee !m) \vee !n$.

1. Тип настройки первого элемента изменяется на дополнительный, так как от выхода этого элемента до выхода схемы расположено нечетное число инверторов. В качестве представителя этого типа таблица настройки содержит формулу $z = (!a \vee !b) (!c \vee !d) \vee !e$.

Заданная БФ реализуется при $a = !x_1, b = x_2, c = x_1, d = !x_2, e = x_3, g = z, i = !x_4, j = !x_5, l = !x_6, m = !x_7, n = !x_8$.

Отметим, что если использовать эту микросхему только в режиме 5-универсального модуля, трудоемкость синтеза уменьшается за счет усложнения схемы: число шагов на первом этапе сокращается на два, а число шагов на втором этапе увеличивается на один, так как в этом случае формула реализуется схемой из трех микросхем.

2.3. Построение универсальных микросборок и плат с высокой логической эффективностью

То обстоятельство, что 3-универсальный модуль с пятью логическими выводами [2] может быть реализован на одной микросхеме К1ЛР333 при $x_1 = x_7 = a, x_2 = x_4 = x_6 = d, x_3 = x_8 = b, x_5 = x_9 = c$, позволило автору решить для ВНИИ «Альтаир» (Москва) в рамках работы над отраслевым стандартом [20] следующую задачу: обеспечить максимальную логическую эффективность универсальной микросборки с 18 логическими выводами, в которой могут быть размещены четыре микросхемы малого уровня интеграции.

Из изложенного следует, что если применить три микросхемы К1ЛР333 и одну микросхему К1ЛБ333 с четырьмя двухвходовыми элементами И—НЕ, то на их основе могут быть построены входящие в состав микросборки [20] три 3-универсальных модуля с прямым и инверсным выходами каждый, у которых суммарное число логических выводов равно 18. Число этих микросборок M , необходимое для реализации произвольной булевой формулы в базисе $\{\&, \vee, !\}$ из h букв, определяется следующим соотношением [2]:

$$\left\lceil \frac{h-1}{6} \right\rceil \leq M \leq \left\lceil \frac{2(h-1)}{9} \right\rceil.$$

Этот же подход был использован автором в НПО «Аврора» при создании одной из систем управления судовыми техническими средствами для согласования большого числа логических выводов у микросхем, размещаемых на печатной плате, с относительно небольшим числом ее логических выводов при условии, что универсальная плата должна обладать максимальной логической эффективностью [2].

2.4. Синтез комбинационных схем из произвольных логических элементов

2.4.1. Синтез схем из двухвходовых элементов И—НЕ

Элемент И—НЕ реализует представителя одного из двух PN -типов ББФ в базисе $\{\&, \vee, !\}$ из двух букв, так как $z_1 = !(ab) = !a \vee !b$. Суперпозиция из двух таких элементов реализует ББФ $z = !((!ab) c) = ab \vee !c$. Следовательно, при $c = 1$ справедливо соотношение $z_2 = ab$, а при $b = 1$ — соотношение $z_3 = a \vee !c$.

Таким образом, построены виртуальный 2-универсальный модуль и виртуальный МЛМ с $k_m = 2.25$. Для приведенной выше формулы z из трех букв список формул, реализуемых виртуальным МЛМ, не содержит дополнительную к ней формулу, поэтому в дальнейшем для уменьшения трудоемкости двухэтапного синтеза будем применять только виртуальный 2-универсальный модуль. При этом метод в общем случае состоит из трех этапов, два из которых рассмотрены выше, а третий состоит в замене настроенных виртуальных модулей их реализациями из элементов И—НЕ.

Оценка сложности в этом случае имеет следующий вид:

$$h - 1 \leq L_4(h) \leq 2(h - 1).$$

Пример 2.12. Реализовать формулу, рассмотренную в примере 2.1, схемой из двухвходовых элементов И—НЕ.

На рис. 2.3 приведена схема, построенная в результате первого этапа синтеза, а на рис. 2.4 — схема, построенная на втором этапе.

Так как построенная схема (рис. 2.4) не содержит модулей с пометкой «2», выполнять третий этап нет необходимости. Она состоит из девяти двухвходовых элементов, что соответствует нижней оценке сложности при $h = 10$. Верификация этой схемы выполняется с помощью подхода, изложенного в разд. 2.1.5. Этот метод был впервые описан в [21].

2.4.2. Синтез схем из набора микросхем серии 133

В [22] перебором было построено более 90 схем из микросхем К1ЛБ333, К1ЛБ334, К1ЛБ331, К1ЛБ332, К1ЛР331, К1ЛР333, при-

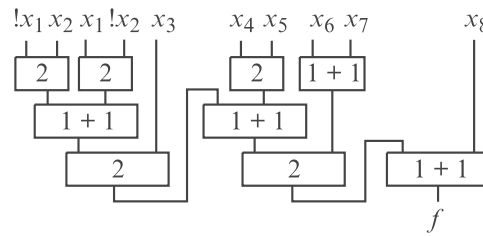


Рис. 2.3

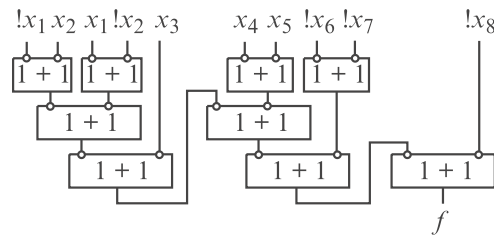


Рис. 2.4

чем суммарная сложность каждой из этих схем не превышала одного корпуса. Анализ этих схем показал, что для построенного виртуального МЛМ

$$k_m = \frac{1}{1} + \frac{2}{2} + \frac{4}{4} + \frac{10}{10} + \frac{24}{24} + \frac{35}{66} + \frac{16}{180} + \frac{5}{522} + \frac{1}{1532} = 5.63.$$

Таким образом, построен также и виртуальный 5-универсальный модуль.

Реализация БФ схемой из указанных микросхем выполняется в три этапа. На первых двух этапах синтез проводится на виртуальных МЛМ или виртуальных универсальных модулях, а на третьем — каждый из модулей в построенной структуре заменяется соответствующей схемой из заданных микросхем. При табличном задании БФУ число этапов увеличивается до четырех. При этом нулевой этап связан с построением БФ.

Из изложенного следует, что главная особенность предлагаемого метода состоит в предварительном исследовании всех функциональных возможностей заданных элементов в определенном классе формул, что резко упрощает последующий синтез.

Более подробно этот метод описан в [2,22]. Идея этого метода была использована при создании системы автоматизации проектирования дискретных устройств на микросхемах [23] и базовых матричных кристаллах [37].

Изложенный в настоящей главе формульный метод может применяться и при синтезе комбинационных схем в СБИС программи-

руемой логики [24]. Можно считать, что предлагаемый метод вошел в научный «обиход», так как он упоминается или описывается в [25–34].

Выводы

1. Предложен метод синтеза комбинационных схем из произвольных (в указанном выше смысле), априори известных логических элементов, названный формульным, так как он основан на реализации БФ в базисах $\{\&, \vee, !\}$ или $\{\&, \vee, \oplus, !\}$, для двухместных операций которых выполняется сочетательный закон.

2. Главная особенность предлагаемого метода состоит в предварительном исследовании всех функциональных возможностей заданных элементов в определенных выше классах БФ, что резко упрощает последующий синтез.

3. При формульном задании функции в зависимости от свойств элементов синтез может выполняться в один, два и три этапа, а при табличном задании функции число этапов увеличивается еще на один. При этом построение схемы по БФ на первом этапе всегда проводится от ее входов к выходу (снизу вверх).

4. Показано, что простой синтез обеспечивают модули, универсальные в классе произвольных БФ в указанных базисах из q букв (q -универсальные модули), а наиболее простой — положительно монотонные q -универсальные модули.

5. Предлагаемый метод отличается от известных тем, что он обеспечивает для практически произвольных логических элементов линейные оценки числа этих элементов, требующихся для реализации БФ в указанных базисах, от числа букв в ней, а для булевой функции — от числа заданных ее значений.

6. Показано, что при минимизации числа букв h в реализуемой БФ имеет место тенденция к сокращению числа весьма сложных q -универсальных модулей. Однако для конкретной БФ при сокращении числа букв может измениться ее структура, что в общем случае приводит к увеличению числа модулей. Установлены соотношения, при выполнении первого из которых сокращение числа букв в БФ до h_1 не увеличивает числа модулей, а при выполнении второго — с гарантией уменьшает их количество.

7. Предложен коэффициент многофункциональности логических элементов, позволяющий в отличие от известных показателей их логической эффективности установить линейную зависимость между числом букв в реализуемой булевой формуле и числом элементов в схеме, ее реализующей. Этот коэффициент для q -универсальных модулей равен q .

8. Предложен подход к формированию логически эффективных универсальных микросборок и плат, обеспечивающий согласование

относительно небольшого числа их логических выводов с большим числом этих выводов у микросхем, которые могут быть размещены на этих платах и в микросборках.

9. При использовании рассмотренного метода структура формулы (ее остаточной) «настраивает» применяемые элементы, в качестве которых, в частности, могут быть выбраны табличные преобразователи (постоянные запоминающие устройства) с q входами и мультиплексоры « 2^q в 1» (при настройке константами 0 и 1) или « 2^{q-1} в 1» (при настройке каждого из них константами 0 и 1, а также одной из переменных и ее инверсией). Использование элементов указанных типов, входящих обычно в состав логических блоков вентиляционных матриц, программируемых пользователями [35], делает каждый шаг декомпозиции результативным.

Л и т е р а т у р а

1. Степановская И. А., Ускач М. А. Представление булевых функций многофункциональными логическими элементами // Автоматика и вычисл. техника. 1973. № 4.
2. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Настраиваемые модули для управляющих логических устройств. Л.: Энергоиздат, 1981.
3. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. О многофункциональном использовании цифровых интегральных схем // Тез. докл. 5-й Всесоюз. науч.-техн. конференции «Проблемы создания систем управления судовыми техническими средствами». Л.: Судостроение, 1973.
4. Артюхов В. Л., Шалыто А. А. Многофункциональные модули из функциональных элементов // Рефераты докл. VI Всесоюз. совещ. по проблемам управления. М.: Наука, 1974.
5. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Вопросы выбора и применения многофункциональных логических модулей // Дискретные системы. Т.1. Симпозиум IFAC. Рига: Зинатне, 1974.
6. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Синтез комбинационных схем из многофункциональных логических модулей // Построение управляющих устройств и систем. М.: Наука, 1974.
7. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Оценки сложности предстоящих реализаций в задачах логического и технического проектирования // Теория релейных устройств. Труды XVI Всесоюз. школы-семинара. Челябинск: ЧПИ, 1976.
8. Артюхов В. Л., Иванов А. А., Шалыто А. А. Автоматизация проектирования комбинационных схем из универсальных модулей для управляющих систем. Л.: ИПК СП, 1987.
9. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Об оценках сложности реализации булевых формул древовидными схемами из настраиваемых модулей // Автоматика и телемеханика. 1981. № 11.
10. Кукинов А. М. Простой метод синтеза скобочных формул для недоопределенных булевых функций // Синтез дискретных автоматов и управляющих устройств. М.: Наука, 1968.
11. Редькин Н. П. О сложности реализации недоопределенных булевых функций // Автоматика и телемеханика. 1969. № 9.
12. Данем Б., Норт Д. Проблемы выбора логически эффективных основных ячеек // Синтез релейных структур. М.: Наука, 1965.
13. Бабичева Е. И., Прангшвили И. В., Ускач М. А., Шаптов Н. Некоторые критерии оценки эффективности логических модулей // Автоматика и телемеханика. 1968. № 3.

14. Майоров С. А., Павленко В. В., Петухов Г. А., Скорубский В. И. О выборе эффективных булевых функций // Изв. вузов. Приборостроение. 1970. № 1.
15. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Оценка логической эффективности интегральных микросхем // Автоматика и вычисл. техника. 1981. № 1.
16. Прангишвили И. В., Ускач М. А., Копейкин Г. А. Комплекс логических МДП-интегральных схем для систем автоматики и телемеханики // Приборы и системы управления. 1970. № 4.
17. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Применение многофункциональных модулей среднего уровня интеграции для построения комбинационных схем // Третье совещание «Логический синтез в дискретных однородных средах». М.: Ин-т проблем передачи информации, 1974.
18. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Применение многофункциональных модулей среднего уровня интеграции для построения комбинационных схем // Теория конечных автоматов и ее приложения. Рига: Зинатне, 1975. Вып. 5.
19. Артюхов В. Л., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1096637 // Бюл. изобр. 1984. № 21.
20. Микросхемы гибридные сложные — микросборки «Пакет». Руководство по применению. Отраслевой стандарт. ОСТ 5.8354—74.
21. Артюхов В. Л., Бородулин В. К., Каталажнов Ю. В., Шалыто А. А. Алгоритм построения логической схемы в базисе «И—НЕ» («ИЛИ—НЕ») по заданной формуле // Обмен опытом в радиопромышленности. 1973. № 10.
22. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Судовые управляющие логические системы. Л.: ИПК СП, 1977.
23. Лекарев М. Ф., Мелехин В. Ф. Автоматизация проектирования структур цифровых устройств. Л.: ЛПИ, 1984.
24. Антонов А. П., Мелехин В. Ф., Филиппов А. С. Обзор элементной базы фирмы ALTERA. СПб.: ЭФО, 1997.
25. Гаврилов М. А. Композиция и декомпозиция комбинационных автоматов // Теория автоматов. М.: Наука, 1976.
26. Якубайтис Э. А. Теория автоматов. Многофункциональные логические модули // Итоги науки и техники: Теория вероятностей. Математическая статистика. Теоретическая кибернетика. М.: ВИНТИ, 1976. № 13.
27. Гаврилов М. А., Девятков В. В., Путырев Е. И. Логическое проектирование дискретных автоматов. М.: Наука, 1977.
28. Малев В. А. Структурная избыточность в логических устройствах. М.: Связь, 1978.
29. Вирьянский З. Я. Проектирование логических устройств судовой автоматики. Л.: Судостроение, 1979.
30. Миценко В. А., Козюминский В. Д., Семашко А. Н. Многофункциональные автоматы и элементная база цифровых ЭВМ. М.: Радио и связь, 1981.
31. Колосов В. Г., Мелехин В. Ф. Проектирование узлов и систем автоматики и вычислительной техники. Л.: Энергоатомиздат, 1983.
32. Путырев Е. И. Перестраиваемые автоматы и микропроцессорные системы. М.: Наука, 1984.
33. Справочник по микропроцессорным устройствам / А. А. Молчанов, В. И. Корнейчук, В. П. Тарасенко, Д. А. Россошинский. Киев: Техника, 1987.
34. Угрюмов Е. П. Проектирование элементов и узлов ЭВМ. М.: Высшая школа, 1987.
35. Угрюмов Е. П., Грушвицкий Р. И., Альшевский А. Н. БИС/СБИС с репрограммируемой структурой. СПб.: Гос. электротех. ун-т, 1996.
36. Артюхов В. Л., Викентьев Л. Ф., Аляев Ю. А. Автоматизация синтеза комбинационных схем из настраиваемых модулей. Л.: ИПК СП, 1985.
37. Мелехин В. Ф., Душутин Е. В., Пышкин Е. В. Автоматизация синтеза при логическом проектировании цифровых устройств на основе базовых матричных кристаллов // Труды СПб ГТУ. 1994. № 449.

Г л а в а 3

Мультиплексорный метод реализации булевых функций схемами из произвольных логических элементов

Предлагаемый метод относится к классу декомпозиционных методов синтеза, так как он строит одновыходную комбинационную схему от выхода к входам на основе декомпозиции булевых функций (заданной и ее остаточных), выполняемой путем решения логических уравнений. Поэтому выполним обзор:

- декомпозиций булевых функций;
- декомпозиционных методов синтеза комбинационных схем;
- методов решения логических уравнений.

3.1. Обзор литературы

3.1.1. Декомпозиции булевых функций

Задача декомпозиции БФУ является одной из важнейших в логическом синтезе, и в настоящее время известно большое число работ по этой тематике [1]. К фундаментальным работам по этому вопросу можно отнести работы [2—8].

Частным случаем декомпозиций являются разложения БФУ по переменным. Наиболее известным является разложение Шеннона по переменной x_i [2], которое имеет следующий вид:

$$f = f(x_i = 0)!x_i \vee f(x_i = 1)x_i.$$

Так как в этом соотношении конъюнкции ортогональны, то

$$f = f(x_i = 0)!x_i \oplus f(x_i = 1)x_i.$$

Ввиду того что $\neg x_i = 1 \oplus x_i$, то

$$\begin{aligned} f &= f(x_i=0)(1 \oplus x_i) \oplus f(x_i=1)x_i = \\ &= f(x_i=0) \oplus (f(x_i=0) \oplus f(x_i=1))x_i. \end{aligned}$$

Ввиду того что $x_i = 1 \oplus \neg x_i$, то

$$\begin{aligned} f &= f(x_i=0)\neg x_i \oplus f(x_i=1)(1 \oplus \neg x_i) = \\ &= f(x_i=1) \oplus (f(x_i=0) \oplus f(x_i=1))\neg x_i. \end{aligned}$$

Эти соотношения называются разложениями Рида [4].

В настоящей работе из всего множества декомпозиций рассматриваются только разделительные декомпозиции, для построения которых предложен метод, названный мультиплексорным. Этот метод основан на решении логических уравнений вида

$$f(x_1, \dots, x_n) = F(\Phi_1, \dots, \Phi_m),$$

где f — декомпозируемая функция, а F — образ декомпозиции.

Определены условия, при выполнении которых такие уравнения имеют решения, которые весьма просто находятся. На основе этого метода декомпозиции предложен метод реализации БФУ схемами из произвольных, априори заданных логических элементов, названный мультиплексорным методом синтеза. Предлагаемый метод является универсальным и для большинства примеров, взятых из литературы, позволяет получать результаты не худшие, чем результаты, получаемые с помощью известных методов. Метод может весьма просто использоваться «вручную» при $n, m < 6$. При больших значениях n и m для снижения трудоемкости он может применяться после декомпозиции, выполняемой на основе других методов, например формульного.

3.1.2. Декомпозиционные методы синтеза комбинационных схем

Методы этой группы базируются на декомпозиции заданной БФУ и всех ее остаточных по образам функций, реализуемых используемыми элементами. Эти методы обеспечивают синтез схем от «выхода к входам», или «сверху вниз».

Аналитический метод синтеза, названный методом каскадов [9], основан на разложении Шеннона. Он может применяться для синтеза схем из переключаемых контактов и мультиплексоров. Элементы этих типов могут эффективно использоваться и на основе табличного метода, названного каноническим методом синтеза [10]. Этот метод был модифицирован для синтеза схем из элементов других типов [11]. Так, например, в [12, 13] излагается модифицированный канонический метод синтеза схем

на универсальных трехходовых элементах, отличных от мультиплексоров, которые предложены в [14] и названы WOS-модулями [15].

Декомпозиционный метод синтеза схем в простых базисах описан в [16]. Весьма специфическими являются методы синтеза схем на пороговых элементах [17—20]: до реализации заданной БФУ параметры пороговых элементов не известны, так как по определению произвольная пороговая функция или подфункция любого числа переменных реализуется одним пороговым элементом. При выполнении шага декомпозиции это обстоятельство не позволяет использовать логическое уравнение, так как его правая часть априори неизвестна. Для частного случая пороговых элементов — мажоритарных элементов [21] — параметры определены и известно большое число работ, посвященных методам синтеза схем в этом элементном базисе (в дальнейшем — в базисе), некоторые из которых указаны в гл. 4.

Наибольший интерес представляют методы синтеза схем в произвольном базисе. В [22—24] предложен метод синтеза схем, основанный на построении для заданной БФУ структурного графа, который в дальнейшем с помощью введенной коалгебры графов преобразуется в искомую схему. Метод является весьма трудоемким даже для весьма простых элементов.

Табличный метод, основанный на решении логических уравнений и названный методом Левенгейма, изложен в [25]. Однако его применение для сложных логических элементов весьма трудоемко.

Декомпозиционные методы синтеза схем в базисе сложных элементов впервые рассмотрены в [26, 27]. Эти методы получили дальнейшее развитие в работах [28—31], где они были названы *методами «направленного поиска»*, так как их идея заключается в выборе (на основе определенных оценочных критериев) переменных, подаваемых на входы элемента, используемого на рассматриваемом шаге декомпозиции. Эти методы весьма трудоемки, и их применение для «ручного» синтеза затруднительно.

Известны и другие работы, посвященные декомпозиционным методам синтеза, например [8, 32, 33], которые, однако, весьма специфичны.

Предлагаемый в настоящей работе метод позволяет устранить недостатки известных методов — специфичность для одних и трудоемкость для других методов.

3.1.3. Методы решения логических уравнений

Ряд методов декомпозиции и реализации БФУ основан на решении логических уравнений [34, 35].

Аналитический метод решения логических уравнений вида

$$f(x_1, \dots, x_n) = 0$$

рассмотрен в [36]. При этом было показано, что если $\Phi_1, \dots, \Phi_n$ — некоторое решение этого уравнения, то общее решение имеет вид:

$$x_i = \Phi_i f(u_1, \dots, u_n) \vee u_n ! f(u_1, \dots, u_n),$$

где $u_1, \dots, u_n$ — произвольные параметры.

В [37] рассмотрен метод, позволяющий в результате решения уравнения $f(x_1, \dots, x_n) = 1$ определить, на каких входных наборах заданная БФУ принимает единичные значения.

В [38] предлагается алгебраический метод решения уравнения

$$F(y_1, \dots, y_m, x_1, \dots, x_n) = 1$$

в виде системы частично определенных функций:

$$y_i = \Phi_j(x_1, \dots, x_n), \quad j = 1, \dots, m.$$

При этом общее решение получается как семейство систем параметрически заданных функций.

Логическое уравнение вида

$$f(x_1, \dots, x_n) = F(\Phi_1, \dots, \Phi_m)$$

может решаться за счет его замены эквивалентной системой из 2^n логических уравнений [39]. В общем случае эта система может быть сведена к эквивалентной системе линейных неравенств [40], к которой добавляются условия, ограничивающие допустимые значения неизвестных значениями 0 и 1, и решена с помощью весьма трудоемких целочисленных методов линейного программирования [41]. Специфика функций F и Φ_j позволяет упростить решение этой системы [42]. В разд. 3.4.3 показано, что, еще больше ограничив свойства функций Φ_j , удастся упростить процедуру решения системы уравнений, которая все равно остается весьма трудоемкой.

Отметим, что в [43] рассмотрен вопрос о решении систем дизъюнктивно-конъюнктивных уравнений. При этом показано, что решение дизъюнктивной (конъюнктивной) системы уравнений представимо в виде правила, аналогичного правилу Крамера, разработанного для решения обыкновенных линейных алгебраических уравнений.

В [44] на основе [45] был предложен матричный метод решения логических уравнений вида:

$$f(X) = F_1(\Phi_1(X_0), \dots, \Phi_m(X_0), X_1), \quad (3.1)$$

где X — множество из n входных переменных; X_0, X_1 — множества из n_0 и n_1 входных переменных соответственно, причем $X_0 \cap X_1 = \emptyset$. Этот метод, как показано в разд. 3.4.2, является чрезвычайно громоздким.

В настоящей работе предлагается существенно более простой метод решения таких уравнений.

3.2. Мультиплексорная декомпозиция и стандартные схемы для ее реализации

3.2.1. Основные определения

Декомпозицией булевой функции (БФУ) называется представление ее в виде суперпозиции других булевых функций.

Пусть БФУ зависит от множества n входных переменных. Предположим для определенности, что множество X_0 образуют переменные $x_1, \dots, x_k$, а множество X_1 — переменные $x_{k+1}, \dots, x_n$. При этом $n_0 = k$, $n_1 = n - k$.

Разделительной декомпозицией, или, более точно, m -кратной разделительной декомпозицией [46], называется ее представление в виде соотношения (3.1). В дальнейшем в этой и других декомпозициях внешнюю функцию в их правой части, как отмечалось выше, будем называть образом декомпозиции. Декомпозиция (3.1) универсальна, так как в таком виде может быть представлена произвольная БФУ.

Если заданная БФУ представлена в (3.1), то говорят, что она разложена по функциям $\Phi_1(X_0), \dots, \Phi_m(X_0)$.

Частным случаем рассмотренной декомпозиции является однократная, или простая, разделительная декомпозиция [46]:

$$f(X) = F_1(\Phi(X_0), X_1).$$

В [47] приведены условия существования для БФУ разделительной и простой разделительной декомпозиций и предложен метод их построения на основе карт декомпозиций. Недостаток этого метода состоит в том, что при его использовании определение функций Φ_j ненаглядно и не является аналитическим.

Введем в рассмотрение декомпозицию вида

$$f(X) = F_2(\Phi_1(X_0), \dots, \Phi_m(X_0), Q_0(X_1), \dots, Q_{2^m-1}(X_1)), \quad (3.2)$$

которую назовем функциональной разделительной декомпозицией.

Введем в рассмотрение также частный случай этой декомпозиции:

$$f(X) = Q_0(X_1) ! \Phi_1(X_0) \dots ! \Phi_m(X_0) \vee \\ \vee Q_1(X_1) ! \Phi_1(X_0) \dots \Phi_m(X_0) \vee \dots \\ \dots \vee Q_{2^m-1}(X_1) \Phi_1(X_0) \dots \Phi_m(X_0), \quad (3.3)$$

который назовем мультиплексорной декомпозицией по названию функции, соответствующей ее образу. В дальнейшем такую декомпозицию будем обозначать более компактно:

$$f(X) = MX(Q_0(X_1), \dots, Q_{2^m-1}(X_1); \Phi_1(X_0), \dots, \Phi_m(X_0)). \quad (3.4)$$

Декомпозиция (3.4) универсальна, так как может быть построена для произвольной БФУ.

Частным случаем рассмотренной является простая мультиплексорная декомпозиция

$$f(X) = MX(Q_0(X_1), Q_1(X_1); \Phi(X_0)),$$

Декомпозицию (3.4) можно рассматривать как булеву формулу, зависящую от $m + 2^m$ переменных $Q_i(X_1)$ и $\Phi_j(X_0)$, которую в общем случае можно проминимизировать. Это позволяет получить декомпозицию (3.2). Если в это соотношение подставить выражения для $Q_i(X_1)$, то можно получить декомпозицию (3.1). Однако в соотношение (3.2) можно подставить и выражения для $\Phi_j(X_0)$, что приводит к декомпозиции

$$f(X) = F_3(Q_0(X_1), \dots, Q_{2^m-1}(X_1), X_0), \quad (3.5)$$

которая в [46, 47] даже не упоминается.

Таким образом, для заданной БФУ может быть построено несколько разделительных декомпозиций и среди них выбрана простейшая по некоторому критерию сложности. При этом отметим, что в известных работах по декомпозиции БФУ, например в [46, 47], говоря о построении декомпозиций (3.1), на самом деле определяют условия существования и предлагают подход к построению декомпозиций (3.3). Видимо, по этой причине соотношение (3.1) в [47] названо дизъюнктивной декомпозицией. Из изложенного следует, что это определение некорректно, так как в (3.1) в качестве образа могут использоваться БФ, отличные от дизъюнкции. Это определение некорректно также и для декомпозиции (3.3), так как суть этой декомпозиции существенно более точно описывает понятие «мультиплексор», в котором 2^m -входы и m -входы «физически» разделены, в то время как, рассматривая соотношение (3.3) в качестве ДНФ, разделение функций Q_i и Φ_j не является столь наглядным.

3.2.2. Построение мультиплексорной декомпозиции на основе решения логических уравнений

Метод построения декомпозиции (3.4) для заданной функции f при фиксированных $n_0 = k$ и $n_1 = n - k$ состоит из четырех этапов.

1. Определяются по столбцу значений таблицы истинности (ТИ) непересекающиеся фрагменты G_t ($t = 0, 1, \dots, 2^k - 1$) длиной 2^{n-k} и формируются R групп ($R \leq t$), состоящих из одинаковых фрагментов $g_0, g_1, \dots, g_{R-1}$.

2. Вычисляется значение $m = \lceil \log R \rceil$, определяющее размерность функции МХ в соотношении (3.4).

3. Назначаются функции $Q_i(X_1)$.

4. Определяются функции $\Phi_j(X_0)$.

Рассмотрим третий этап более подробно.

При $R = 2^m$ функция $Q_i(X_1)$ формируется в результате ее приравнивания представителю группы фрагментов, причем каждой функции Q_i соответствует один представитель группы и наоборот. Число таких приравниваний (назначений функций Q_i) определяется соотношением $H_1 = 2^m!$ (здесь и далее символ $!$, стоящий после операнда, — операция «факториал»). Выбор варианта назначения определяет выбор функций Φ_j при фиксированном образе декомпозиции.

При $R \neq 2^m$ сначала формируются R функций Q_i , как и в случае $R = 2^m$. Остальные $(2^m - R)$ функций могут быть выбраны произвольно. При этом в качестве функции $Q_i(X_1)$ может быть выбрана произвольная БФУ, зависящая от n_1 переменных, в том числе и не являющаяся фрагментом реализуемой функции. Число вариантов назначений в этом случае H_2 , причем $H_2 > H_1$.

Обратим внимание на тот факт, что во втором случае возможны два варианта: $2^m = 2^k$ и $2^m < 2^k$. В первом из них, если критерием декомпозиции является простота функций Φ_j , то применение предлагаемого подхода нецелесообразно, так как того же результата можно достичь с помощью менее трудоемкого метода, излагаемого в следующем разделе. Для первого варианта при использовании других критериев и для второго варианта для построения декомпозиции (3.4) должен применяться предлагаемый метод.

Перейдем к рассмотрению четвертого этапа. Предположим, что назначение функций Q_i фиксировано. При этом если все функции Q_i различны (одинаковыми могут быть только функции Q_i при $R \neq 2^m, 2^m < 2^k$, которые не являются фрагментами функции f), то существует единственный набор функций Φ_j , удовлетворяющий соотношению (3.4), т. е. существует единственная декомпозиция этого вида. В противном случае существует определенное число таких декомпозиций, которое может быть вычислено.

Формирование функций Φ_j осуществляется таким образом, что оно позволяет расположить фрагменты, соответствующие функциям Q_i , в том порядке, в котором они размещены в столбце значений реализуемой функции.

В качестве модели для определения функций Φ_j и декомпозиции (3.4) в целом предлагается использовать стандартную схему, представленную на рис. 3.1.

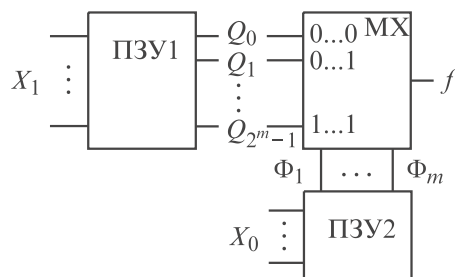


Рис. 3.1

В этой схеме применяются МХ « 2^m в 1» и два постоянных запоминающих устройства (ПЗУ).

ПЗУ1 формирует функции Q_i . При этом функция с номером i подается на вход МХ, адрес которого является двоичным эквивалентом номера i .

Каждая строка ПЗУ2 по значениям набора входных переменных множества X_0 формирует m -разрядный двоичный адрес 2^m -го входа МХ, по которому на выход МХ передается функция Q_i , соответствующая этому входному набору. При этом содержимое i -го столбца ПЗУ2 определяет функцию Φ_j .

Мультиплексор реализует образ декомпозиции (3.4).

Эта схема кроме теоретического значения для нахождения декомпозиций (3.4) имеет также практическое значение в качестве нового подхода к композиции ПЗУ.

Завершив изложение метода построения декомпозиции (3.4) для функции f , заданной ТИ, отметим, что если говорить о построении простейшей по выбранному критерию декомпозиции, то в общем случае имеют место три вида перебора, влияющих на ее сложность:

- по порядку расположения входных переменных в ТИ;
- по назначению функций Q_i ;
- по формированию функций $\Phi_j(X_0)$ при $R \neq 2^m$, $2^m < 2^k$.

Перебор минимален для симметрических функций с $R = 2^m$, так как для них может иметь место перебор только второго вида. Для таких функций с $R \neq 2^m$, $2^m < 2^k$, сложность декомпозиции не зависит только от порядка входных переменных в ТИ. Для произвольных БФУ изменение порядка входных переменных в об-

щем случае может так изменить величину R , что изменится и величина m .

Пример 3.1. Построить мультиплексорную декомпозицию для БФУ f (табл. 3.1) при $k=3$. При этом $X_0 = \{x_1, x_2, x_3\}$, $X_1 = \{x_4, x_5\}$.

Т а б л и ц а 3.1

x_1	x_2	x_3	x_4	x_5	f	g_i	G_r	Q_i	x_1	x_2	x_3	x_4	x_5	f	g_i	G_r	Q_i
0	0	0	0	0	0	g_0	G_0	Q_0	1	0	0	0	0	0	g_4	G_1	Q_1
0	0	0	0	1	0				1	0	0	0	1	0			
0	0	0	1	0	0				1	0	0	1	0	0			
0	0	0	1	1	0				1	0	0	1	1	1			
0	0	1	0	0	0	g_1	G_1	Q_1	1	0	1	0	0	0	g_5	G_2	Q_2
0	0	1	0	1	0				1	0	1	0	1	1			
0	0	1	1	0	0				1	0	1	1	0	1			
0	0	1	1	1	1				1	0	1	1	1	1			
0	1	0	0	0	0	g_2	G_1	Q_1	1	1	0	0	0	0	g_6	G_2	Q_2
0	1	0	0	1	0				1	1	0	0	1	1			
0	1	0	1	0	0				1	1	0	1	0	1			
0	1	0	1	1	1				1	1	0	1	1	1			
0	1	1	0	0	0	g_3	G_2	Q_2	1	1	1	0	0	0	g_7	G_1	Q_1
0	1	1	0	1	1				1	1	1	0	1	0			
0	1	1	1	0	1				1	1	1	1	0	0			
0	1	1	1	1	1				1	1	1	1	1	1			

1. Выделим в столбце значений ТИ $2^k = 8$ фрагментов длиной $2^{n-k} = 4$ и одинаковые фрагменты объединим в группы: $G_0 = g_0 = 0$, $G_1 = g_1 = g_2 = g_4 = g_7 = x_4 x_5$, $G_2 = g_3 = g_5 = g_6 = x_4 \vee x_5$. Таким образом, $R = 3$.

2. Так как $m = \lceil \log R \rceil$, $R \neq 2^m$, а $2^m < 2^k$, то будем искать декомпозицию

$$f(X) = MX(Q_0(X_1), Q_1(X_1), Q_2(X_1), Q_3(X_1); \Phi_1(X_0), \Phi_2(X_0)).$$

Схема, реализующая это соотношение, приведена на рис. 3.2.

3. Среди множества вариантов назначений функций Q_i выберем следующее: $Q_0 = G_0 = 0$, $Q_1 = G_1 = x_4 x_5$, $Q_2 = G_2 = x_4 \vee x_5$. В качестве функции Q_3 может быть выбрана произвольная функция, зависящая от переменных x_4, x_5 . Если $Q_3 \neq \{0, x_4 x_5, x_4 \vee x_5\}$, то вы-

бор функций Φ_j единствен. В противном случае при каждом назначении возможны несколько наборов функций Φ_1 и Φ_2 .

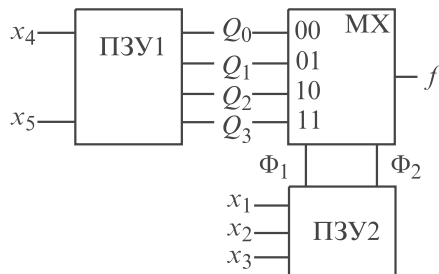


Рис. 3.2

4.1. Предположим, что $Q_3 = 1$. Тогда ПЗУ1 может быть запрограммировано, как показано на рис. 3.3.

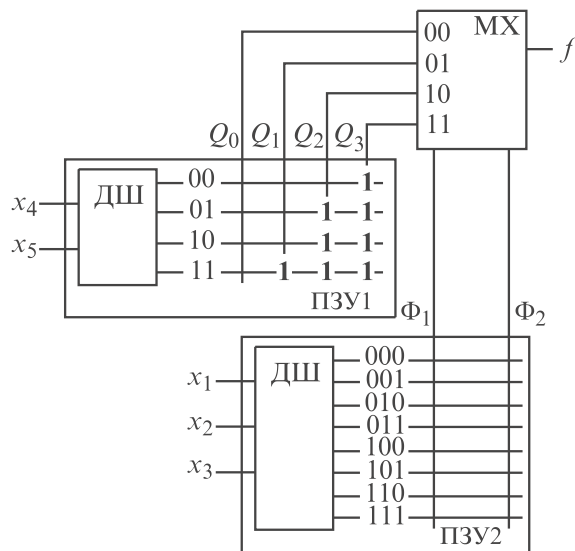


Рис. 3.3

Покажем, как формируются функции Φ_1 и Φ_2 (программируется ПЗУ2). Из рассмотрения табл. 3.1 и рис. 3.3 (ДШ — дешифратор) следует, что:

при $x_1 = 0$, $x_2 = 0$, $x_3 = 0$ на выход схемы должна передаваться функция Q_0 , подключенная к входу МХ адресом 00. Следовательно, $\Phi_1(0, 0, 0) = 0$, $\Phi_2(0, 0, 0) = 0$;

при $x_1=0, x_2=0, x_3=1$ на выход схемы должна передаваться функция Q_1 , подключенная к входу МХ адресом 01. Следовательно, $\Phi_1(0, 0, 1)=0, \Phi_2(0, 0, 1)=1$;

при $x_1=0, x_2=1, x_3=0$ на выход схемы должна передаваться функция Q_1 , подключенная к входу МХ адресом 01. Следовательно, $\Phi_1(0, 1, 0)=0, \Phi_2(0, 1, 0)=1$;

при $x_1=0, x_2=1, x_3=1$ на выход схемы должна передаваться функция Q_2 , подключенная к входу МХ адресом 10. Следовательно, $\Phi_1(0, 1, 1)=1, \Phi_2(0, 1, 1)=0$;

при $x_1=1, x_2=0, x_3=0$ на выход схемы должна передаваться функция Q_1 , подключенная к входу МХ адресом 01. Следовательно, $\Phi_1(1, 0, 0)=0, \Phi_2(1, 0, 0)=1$;

при $x_1=1, x_2=0, x_3=1$ на выход схемы должна передаваться функция Q_2 , подключенная к входу МХ адресом 10. Следовательно, $\Phi_1(1, 0, 1)=1, \Phi_2(1, 0, 1)=0$;

при $x_1=1, x_2=1, x_3=0$ на выход схемы должна передаваться функция Q_2 , подключенная к входу МХ адресом 10. Следовательно, $\Phi_1(1, 1, 0)=1, \Phi_2(1, 1, 0)=0$;

при $x_1=1, x_2=1, x_3=1$ на выход схемы должна передаваться функция Q_1 , подключенная к входу МХ адресом 01. Следовательно, $\Phi_1(1, 1, 1)=0, \Phi_2(1, 1, 1)=1$.

На этом завершается программирование ПЗУ2 (рис.3.4).

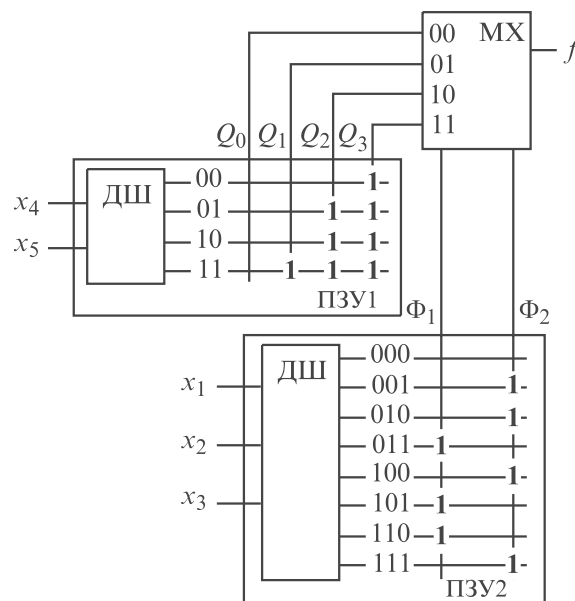


Рис. 3.4

Из рассмотрения ПЗУ2 следует, что

$$\Phi_1 = !x_1 x_2 x_3 \vee x_1(x_2 \oplus x_3), \quad \Phi_2 = x_1 \oplus x_2 \oplus x_3. \quad (3.6)$$

Таким образом, искомая декомпозиция построена.

Предложим аналитический метод определения функций Φ_1 и Φ_2 . Для этого составим и решим логическое уравнение

$$f(x_1, x_2, x_3, x_4, x_5) = MX(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2), \quad (3.7)$$

которое в мультиплексорной форме (МФ) имеет вид:

$$\begin{aligned} MX(Q_0, Q_1, Q_1, Q_2, Q_1, Q_2, Q_2, Q_1; x_1, x_2, x_3) = \\ = MX(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2). \end{aligned}$$

Это представление является частным случаем МФ, так как в нем все функции Q_i в правой части уравнения по содержанию различны. Это приводит к тому, что уравнение в данном случае имеет единственное решение, которое также будем искать в МФ.

Достоинство предлагаемого подхода состоит в том, что решение уравнения может быть найдено непосредственно из МФ за счет представления в двоичной форме индексов функций Q_i , расположенных в левой части уравнения в МФ, и записи представления каждого индекса в виде соответствующего столбца в МФ решения:

$$\begin{aligned} \Phi_1 &= MX(0, 0, 0, 1, 0, 1, 1, 0; x_1, x_2, x_3); \\ \Phi_2 &= MX(0, 1, 1, 0, 1, 0, 0, 1; x_1, x_2, x_3). \end{aligned} \quad (3.8)$$

Соотношения (3.6) являются формульной записью этого решения.

Декомпозиция (3.2) может быть получена, если учесть, что для функций Q_i выполняются отношения покрытия — неравенство $Q_0 \leq Q_1 \leq Q_2 \leq Q_3$. При этом

$$f = Q_0 \vee Q_1 \Phi_2 \vee Q_2 \Phi_1 \vee Q_3 \Phi_1 \Phi_2.$$

Так как $Q_0 = 0$, $Q_3 = 1$, $\Phi_1 \Phi_2 = 0$, то $f = Q_1 \Phi_2 \vee Q_2 \Phi_1$.

Декомпозиция (3.1) в данном случае имеет вид:

$$f = \Phi_1(x_4 \vee x_5) \vee \Phi_2 x_4 x_5,$$

а декомпозиция (3.5) соответствует следующему соотношению:

$$f = Q_1(x_1 \oplus x_2 \oplus x_3) \vee Q_2(!x_1 x_2 x_3 \vee x_1(x_2 \oplus x_3)).$$

4.2. Для случая $Q_3 = 0$, так как $Q_0 = Q_3$, то уравнение (3.7) может быть записано в виде:

$$f = Q_1 \Phi_1 !\Phi_2 \vee Q_2 \Phi_1 !\Phi_2.$$

Так как в данном случае в правой части уравнения в МФ две функции Q_i должны быть по содержанию одинаковы, а эти функции должны использоваться в левой его части, то уравнение (3.7) может быть записано в МФ, которую будем называть п а р а м е т р и ч е с к о й:

$$\begin{aligned} \text{MX}(Q_0?Q_3, Q_1, Q_1, Q_2, Q_1, Q_2, Q_2, Q_1; x_1, x_2, x_3) = \\ = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2). \end{aligned}$$

Это уравнение имеет два решения, первое из которых (3.6) определяется по соотношениям (3.8), а второе имеет вид:

$$\begin{aligned} \Phi_1 &= \text{MX}(1, 0, 0, 1, 0, 1, 1, 0; x_1, x_2, x_3) = \\ &= !x_1 \oplus x_2 \oplus x_3; \\ \Phi_2 &= \text{MX}(1, 1, 1, 0, 1, 0, 0, 1; x_1, x_2, x_3) = \\ &= !x_1 \# !x_2 \# !x_3 \vee x_1 x_2 x_3. \end{aligned} \tag{3.9}$$

Другим представлением МФ, которая может быть названа явной, в данном случае является следующее:

$$\begin{aligned} \text{MX}(0, x_4 x_5, x_4 x_5, x_4 \vee x_5, x_4 x_5, x_4 \vee x_5, x_4 \vee x_5, x_4 x_5; \\ x_1, x_2, x_3) = \text{MX}(0, x_4 x_5, x_4 \vee x_5, 0; \Phi_1, \Phi_2). \end{aligned}$$

Решение уравнения в этой форме выполняется следующим образом: для каждой функции в левой части МФ до точки с запятой выбирается одна из одинаковых с ней функций, расположенных в правой части МФ, и в решение в виде столбца записывается двоичный код номера выбранной позиции в правой части формы. Как и для предыдущего представления МФ, решениями этого уравнения являются соотношения (3.8) и (3.9).

В дальнейшем будет использоваться как параметрическая, так и явная мультиплексорная форма.

4.3. Для случая $Q_3 = x_4 \vee x_5$, так как $Q_2 = Q_3$, то уравнение (3.7) может быть записано в виде:

$$f = Q_1 \Phi_2 \vee Q_2 \Phi_1.$$

Запишем уравнение (3.7) в МФ в следующем виде:

$$\begin{aligned} \text{MX}(Q_0, Q_1, Q_1, Q_2?Q_3, Q_1, Q_2?Q_3, Q_2?Q_3, Q_1; x_1, x_2, x_3) = \\ = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2). \end{aligned}$$

Это уравнение имеет восемь решений. Во всех решениях функция Φ_1 одинакова и совпадает с первой функцией в соотношениях (3.8) и (3.6), а простейшая БФ для функции Φ_2 получается при

$$\Phi_2 = \text{MX}(0, 1, 1, 1, 1, 1, 1, 1; x_1, x_2, x_3) = x_1 \vee x_2 \vee x_3.$$

4.4. Для случая $Q_3 = x_4 x_5$, так как $Q_1 = Q_3$, то уравнение (3.7) может быть записано в виде:

$$f = Q_2 \Phi_1! \Phi_2 \vee Q_1 \Phi_2.$$

Запишем уравнение (3.7) в МФ в следующем виде:

$$\begin{aligned} \text{MX}(Q_0, Q_1?Q_3, Q_1?Q_3, Q_2, Q_1?Q_3, Q_2, Q_2, Q_1?Q_3; \\ x_1, x_2, x_3) = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2). \end{aligned}$$

Это уравнение имеет 16 решений, среди которых имеется следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, 0, 1, 0, 1, 1, 1; x_1, x_2, x_3) = x_1 \# x_2 \# x_3; \\ \Phi_2 &= \text{MX}(0, 1, 1, 0, 1, 0, 0, 1; x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3. \end{aligned}$$

Таким образом, решением уравнения в данном случае является «параллельный двоичный одноразрядный сумматор».

Пример 3.2. Построить мультиплексорную декомпозицию при $k = 2$ для БФУ

$$f(x_1, x_2, x_3, x_4) = |0000 \ 0001 \ 1111 \ 1111|^T.$$

1. Столбец значений этой функции состоит из трех типов фрагментов ($R = 3$) длиной четыре: $G_0 = g_0 = 0$, $G_1 = g_1 = x_3 \cdot x_4$, $G_2 = g_2 = g_3 = 1$.

2. Так как в данном случае $m = 2$, а $2^m = 2^k$, то простейшая реализация функций Φ_1 и Φ_2 достигается при использовании разложения Шеннона одновременно по переменным x_1 и x_2 .

Пример 3.3. Построить мультиплексорную декомпозицию при $k = 2$ для БФУ, рассмотренной в предыдущем примере, в таблице истинности которой порядок входных переменных изменен на противоположный:

$$f(x_4, x_3, x_2, x_1) = |0101 \ 0101 \ 0101 \ 0111|^T.$$

1. Столбец значений этой функции состоит из двух типов фрагментов ($R = 2$) длиной четыре: $G_0 = g_0 = g_1 = g_2 = x_1$, $G_1 = g_3 = x_1 \vee x_2$.

2. В данном случае $m = 1$, а $2^m \neq 2^k$. При этом исходное уравнение может быть записано в виде:

$$f = \text{MX}(Q_0, Q_1; \Phi).$$

3. В данном случае возможны два варианта назначений функций Q_i :

$$\begin{cases} Q_0 = x_1; \\ Q_1 = x_1 \vee x_2; \end{cases} \quad \begin{cases} Q_0 = x_1 \vee x_2; \\ Q_1 = x_1. \end{cases}$$

4. В первом случае

$$f = x_1 \vee x_2 \Phi;$$

$$\text{MX}(Q_0, Q_0, Q_0, Q_1; x_4, x_3) = \text{MX}(Q_0, Q_1; \Phi).$$

Это уравнение имеет единственное решение:

$$\Phi = \text{MX}(0, 0, 0, 1; x_4, x_3) = x_3 x_4.$$

Во втором случае

$$f = x_1 \vee x_2 !\Phi;$$

$$\text{MX}(Q_1, Q_1, Q_1, Q_0; x_4, x_3) = \text{MX}(Q_0, Q_1; \Phi).$$

Это уравнение имеет единственное решение:

$$\Phi = \text{MX}(1, 1, 1, 0; x_4, x_3) = !x_3 \vee !x_4.$$

Из изложенного следует, что в недекомпозированной форме заданная БФУ реализуется формулой $f = x_1 \vee x_2 x_3 x_4$.

В заключение раздела отметим, что в [49] приведена схема, аналогичная приведенной на рис.3.4, которая реализует конкретную БФУ пяти переменных при $n - k = 2$. Однако какой-либо математический аппарат для решения как этой задачи, так и в общем случае в этой работе отсутствует.

3.2.3. Использование карт декомпозиции при мультиплексорной декомпозиции

Пусть задана функция $f(x_1, \dots, x_n)$. При фиксированном значении $n - k$ эта функция может быть записана в виде: $f(x_1, \dots, x_k, x_{k+1}, \dots, x_n)$.

По столбцу значений ТИ этой функции легко определить номенклатуру фрагментов $G_l(x_{k+1}, \dots, x_n)$ длиной 2^{n-k} . Пусть число типов фрагментов равно R_1 .

Используя первые $n - k$ входных переменных для определения строк, а последние k входных переменных для определения столбцов, заполним по столбцу значений функции f карту декомпозиции [47]. Предположим, что эта карта имеет R_2 различных столбцов.

Если $R_1 < R_2$, то ТИ не преобразуется, а если $R_1 > R_2$, то по карте декомпозиции строится ТИ для БФУ $f(x_{k+1}, \dots, x_n, x_1, \dots, x_{n-k})$, столбец значений которой содержит R_2 различных фрагментов длиной 2^{n-k} , зависящих от переменных $x_1, \dots, x_{n-k}$.

Таким образом, по карте декомпозиции весьма просто определяется номенклатура фрагментов при любом значении $n - k$ для второго из возможных $n!$ вариантов расположения столбцов входных переменных в ТИ.

Пример 3.4. Для БФУ $f(x_1, x_2, x_3) = |0010 \ 0111|^T$, немонотонной по переменной x_3 , определить: не является ли она монотонной по переменной x_1 .

Построим карту декомпозиции при $k = 1$ (табл. 3.2).

Эта таблица не содержит столбца типа $!x_1$, и поэтому является

Т а б л и ц а 3.2

x_1	$x_2 x_3$			
	00	01	10	11
0	0	0	1	1
1	0	1	1	1

монотонной по переменной x_1 . При этом $f(x_2, x_3, x_1) = |0001 \ 1111|^T$.

Пример 3.5. Для БФУ

$$f(x_1, x_2, x_3, x_4) = |0000 \ 0110 \ 1001 \ 1111|^T$$

построить разделительную декомпозицию при $n - k = 2$.

В а р и а н т 1. Заданный столбец значений состоит из четырех типов фрагментов: $G_0 = 0$, $G_1 = x_3 \oplus x_4$, $G_2 = !(x_3 \oplus x_4)$, $G_3 = 1$. Назначим функции Q_i так, чтобы удовлетворялись отношения покрытия: $Q_0 \leq Q_1$, $Q_0 \leq Q_2$, $Q_0 \leq Q_3$, $Q_1 \leq Q_3$, $Q_2 \leq Q_3$. Эти соотношения выполняются, например, при $Q_0 = G_0$, $Q_1 = G_1$, $Q_2 = G_2$, $Q_3 = G_3$. Составим уравнение

$$f(x_1, x_2, x_3, x_4) = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2).$$

Это уравнение в МФ имеет вид:

$$\text{MX}(Q_0, Q_1, Q_2, Q_3; x_1, x_2) = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2).$$

При этом $\Phi_1 = x_1$, $\Phi_2 = x_2$.

Так как отношения покрытия выполняются, то из [48] следует, что

$$f = Q_0 \vee Q_1 \Phi_2 \vee Q_2 \Phi_1 \vee Q_3 \Phi_1 \Phi_2.$$

Учитывая, что $Q_0 = 0$, $Q_1 = !Q_2 = Q$, $Q_3 = 1$, получим:

$$f = \Phi_1 !Q \vee \Phi_2 Q.$$

В а р и а н т 2. Построим карту декомпозиции (табл. 3.3).

Т а б л и ц а 3.3

$x_1 x_2$	$x_3 x_4$			
	00	01	10	11
00	0	0	0	0
01	0	1	1	0
10	1	0	0	1
11	1	1	1	1

Из этой карты следует, что

$$f(x_3, x_4, x_1, x_2) = |0011 \ 0101 \ 0101 \ 0011|^T.$$

В данном случае имеется только два типа фрагментов: $G_0 = g_0 = g_3 = x_1$ и $G_1 = g_1 = g_2 = x_2$. Полагая $Q_0 = x_1$, $Q_1 = x_2$, запишем уравнение

$$f(x_3, x_4, x_1, x_2) = \text{MX}(Q_0, Q_1; \Phi),$$

которое в МФ имеет следующий вид:

$$\text{MX}(Q_0, Q_1, Q_1, Q_0; x_3, x_4) = \text{MX}(Q_0, Q_1; \Phi).$$

При этом $\Phi = \text{MX}(0, 1, 1, 0; x_3, x_4) = x_3 \oplus x_4$, а $f = x_1 !\Phi \vee x_2 \Phi$.

Таким образом, использование карты декомпозиции позволило найти декомпозицию менее трудоемко.

Пример 3.6. Построить мультиплексорную декомпозицию при $n - k = 2$ для булевой функции

$$\begin{aligned} f(x_4, x_5, x_1, x_2, x_3) = \\ = |0000 \ 0000 \ 0010 \ 1010 \ 1000 \ 1111 \ 1010 \ 1111|^T. \end{aligned}$$

В данном случае $R = 5$ и $m = 3$. Попытаемся уменьшить значения R и m за счет построения карты декомпозиции (табл. 3.4).

Т а б л и ц а 3.4

$x_4 x_5$	$x_1 x_2 x_3$							
	000	001	010	011	100	101	110	111
00	0	0	0	0	0	0	0	0
01	0	0	1	0	1	0	1	0
10	1	0	0	0	1	1	1	1
11	1	0	1	0	1	1	1	1

Из этой карты следует, что при записи БФУ в виде

$$f(x_1, x_2, x_3, x_4, x_5) = |0011 \ 0000 \ 0101 \ 0000 \ 0111 \ 0011 \ 0111 \ 0011|^T,$$

$R = 4$, а $m = 2$. Полагая $Q_0 = 0$, $Q_1 = x_5$, $Q_2 = x_4$, $Q_3 = x_4 \vee x_5$, выполняются отношения покрытия, и поэтому $f = x_4 \Phi_1 \vee x_5 \Phi_2$.

Для определения функций Φ_1 и Φ_2 составим уравнение в МФ:

$$\begin{aligned} \text{MX}(Q_2, Q_0, Q_1, Q_0, Q_3, Q_2, Q_3, Q_2; x_1, x_2, x_3) = \\ = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2). \end{aligned}$$

Это уравнение имеет единственное решение:

$$\Phi_1 = \text{MX}(1, 0, 0, 0, 1, 1, 1, 1; x_1, x_2, x_3) = x_1 \vee !x_2 !x_3;$$

$$\Phi_2 = \text{MX}(0, 0, 1, 0, 1, 0, 1, 0; x_1, x_2, x_3) = (x_1 \vee x_2) !x_3.$$

3.2.4. Разложения Шеннона по крайним левым входным переменным

Разложение по крайней левой входной переменной. Если в соотношении (3.4) выбрать $m = k - 1$, $Q_0 = f(x_1 = 0) = f(0)$, $Q_1 = f(x_1 = 1) = f(1)$, то $\Phi = x_1$.

При этом

$$f = \text{MX}(f(0), f(1); x_1).$$

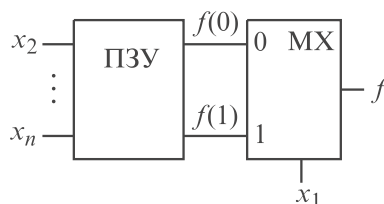


Рис. 3.5

Это соотношение может быть названо разложением Шеннона по крайней левой входной переменной. Стандартная схема, реализующая это соотношение, приведена на рис.3.5.

Разложение по k крайним левым входным переменным. Если в соотношении (3.4) выбрать $m = k$, $Q_0 = f(x_1 = 0, \dots, x_k = 0) = f_0(X_1)$, $Q_1 = f(x_1 = 0, \dots, x_k = 1) = f_1(X_1)$, ..., $Q_{2^k-1} = f(x_1 = 1, \dots, x_k = 1) = f_{2^k-1}(X_1)$, то функции Φ_j могут быть определены следующим образом:

$$\Phi_1 = x_1, \Phi_2 = x_2, \dots, \Phi_k = x_k.$$

При этом

$$f = \text{MX}(f_0(X_1), f_1(X_1), \dots, f_{2^k-1}(X_1); x_1, x_2, \dots, x_k).$$

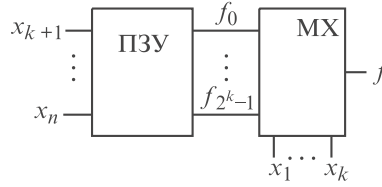


Рис. 3.6

Это соотношение может быть названо разложением Шеннона по k крайним левым входным переменным. Стандартная схема, реализующая это соотношение, приведена на рис. 3.6.

Разложение по $n - 1$ крайним левым входным переменным. Если в соотношении (3.4) выбрать $m = n - 1$, $Q_0 = f(x_1 = 0, \dots, x_{n-1} = 0, x_n) = f_0(x_n)$, ..., $Q_{2^{n-1}-1} = f(x_1 = 1, \dots, x_{n-1} = 1, x_n) = f_{2^{n-1}-1}(x_n)$, то функции Φ_j могут быть определены следующим образом:

$$\Phi_1 = x_1, \Phi_2 = x_2, \dots, \Phi_{2^{n-1}-1} = x_{n-1}.$$

При этом

$$f = \text{MX}(f_0(x_n), f_1(x_n), \dots, f_{2^{n-1}-1}(x_n); x_1, x_2, \dots, x_{n-1}),$$

где $f_i(x_n) = \{0, 1, !x_n, x_n\}$.

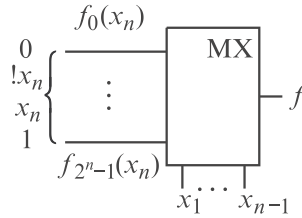


Рис. 3.7

Это соотношение может быть названо разложением Шеннона по $n - 1$ крайним левым входным переменным. Стандартная схема, реализующая это соотношение, приведена на рис. 3.7.

Разложение по всем входным переменным. Если в соотношении (3.4) выбрать $m = n$, $Q_0 = f(x_1 = 0, \dots, x_n = 0) = f_0, \dots, Q_{2^n-1} = f(x_1 = 1, \dots, x_n = 1) = f_{2^n-1}$, то функции Φ_j могут быть определены следующим образом:

$$\Phi_1 = x_1, \dots, \Phi_{2^n-1} = x_n.$$

При этом

$$f = \text{MX}(f_0, \dots, f_{2^n-1}; x_1, x_2, \dots, x_n), \text{ где } f_i = 0, 1.$$

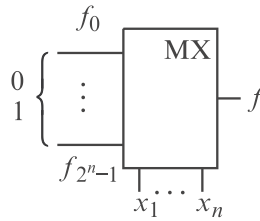


Рис. 3.8

Это соотношение может быть названо разложением Шеннона по всем входным переменным. Стандартная схема, реализующая это соотношение, приведена на рис. 3.8.

Методы реализации БФУ схемами, приведенными в этом разделе, хорошо известны и рассмотрены, например в [50].

3.2.5. Универсальные разложения булевых функций по крайним правым входным переменным

Разложение по крайней правой входной переменной. Рассмотрим соотношение

$$f = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2),$$

где $Q_i = \{0, 1, !x_n, x_n\}$, $Q_0 \neq Q_1 \neq Q_2 \neq Q_3$, а функции Φ_1 и Φ_2 не зависят от переменной x_n .

Покажем, что это соотношение порождает при различных назначениях функций Q_i все универсальные разложения произвольной БФУ на две остаточные функции по переменной x_n , причем каждое из них однотипно либо с разложением Шеннона, либо с одним из разложений Рида по этой переменной.

Тем самым доказывается, что не существует универсальных разложений по одной переменной, отличных от однотипных с разложениями Шеннона и Рида.

Для рассматриваемого случая существует $4! = 24$ (! — символ операции «факториал») варианта назначений функций Q_i в рас-

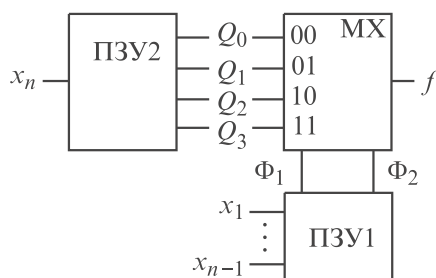


Рис. 3.9

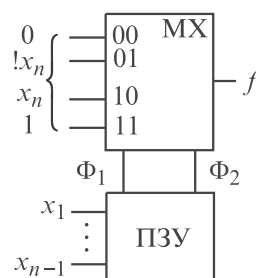


Рис. 3.10

сматриваемом соотношении, которому соответствует стандартная схема, приведенная на рис. 3.9. Эта схема может быть так же изображена, как показано на рис. 3.10.

Каждое из назначений приводит к определенной БФУ и соответствующей БФ (табл. 3.5).

Т а б л и ц а 3.5

N	Q_0	Q_1	Q_2	Q_3	f	N	Q_0	Q_1	Q_2	Q_3	f
1	0	1	$!x_n$	x_n	$\Phi_2 \oplus \Phi_1 !x_n$	13	$!x_n$	0	1	x_n	$! \Phi_2 !x_n \vee \Phi_1 x_n$
2	0	1	x_n	$!x_n$	$\Phi_2 \oplus \Phi_1 x_n$	14	$!x_n$	0	x_n	1	$! \Phi_2 !x_n \oplus \Phi_1$
3	0	$!x_n$	1	x_n	$\Phi_1 \oplus \Phi_2 !x_n$	15	$!x_n$	1	0	x_n	$! \Phi_1 !x_n \vee \Phi_2 x_n$
4	0	$!x_n$	x_n	1	$\Phi_2 !x_n \vee \Phi_1 x_n$	16	$!x_n$	1	x_n	0	$! \Phi_1 \oplus ! \Phi_2 x_n$
5	0	x_n	1	$!x_n$	$\Phi_1 \oplus \Phi_2 x_n$	17	$!x_n$	x_n	0	1	$! \Phi_1 !x_n \oplus \Phi_2$
6	0	x_n	$!x_n$	1	$\Phi_1 !x_n \vee \Phi_2 x_n$	18	$!x_n$	x_n	1	0	$! \Phi_2 \oplus ! \Phi_1 x_n$
7	1	0	$!x_n$	x_n	$! \Phi_2 \oplus \Phi_1 x_n$	19	x_n	0	1	$!x_n$	$\Phi_1 !x_n \vee ! \Phi_2 x_n$
8	1	0	x_n	$!x_n$	$\Phi_1 !x_n \oplus ! \Phi_2$	20	x_n	0	$!x_n$	1	$\Phi_1 \oplus ! \Phi_2 x_n$
9	1	$!x_n$	0	x_n	$! \Phi_1 \oplus \Phi_2 x_n$	21	x_n	1	0	$!x_n$	$\Phi_2 !x_n \vee ! \Phi_1 x_n$
10	1	$!x_n$	x_n	0	$! \Phi_1 !x_n \vee ! \Phi_2 x_n$	22	x_n	1	$!x_n$	0	$! \Phi_2 !x_n \oplus ! \Phi_1$
11	1	x_n	0	$!x_n$	$\Phi_2 !x_n \oplus \Phi_1$	23	x_n	$!x_n$	0	1	$\Phi_2 \oplus ! \Phi_1 x_n$
12	1	x_n	$!x_n$	0	$! \Phi_2 !x_n \vee ! \Phi_1 x_n$	24	x_n	$!x_n$	1	0	$! \Phi_1 !x_n \oplus ! \Phi_2$

Из рассмотрения этой таблицы следует, что восемь выражений в ней (4, 6, 10, 12, 13, 15, 19, 21) PN -однотипны с разложением Шеннона, восемь (2, 5, 7, 9, 16, 18, 20, 23) — PN -однотипны с разложением Риды по переменной x_n , а последние восемь (1, 3, 8, 11, 14, 17, 22,

24) — PN -однотипны с разложением Рида по переменной $!x_n$. Из изложенного следует, что не существует универсальных разложений на две остаточные функции, отличных от разложений Шеннона и Рида и PN -однотипных с ними, что до сих пор не было известно.

Каждое из этих выражений является универсальным, так как может использоваться в качестве образа декомпозиции произвольной булевой функции n переменных. Обратим внимание на тот факт, что при применении каждого из этих выражений существует единственный набор функций Φ_1 и Φ_2 .

Пример 3.7. Используя строку 6 табл. 3.5, выполнить декомпозицию функции

$$f(x_1, x_2, x_4, x_3) = |0101 \ 0100 \ 0110 \ 0111|^T.$$

Составим уравнение $f = \Phi_1 !x_3 \vee \Phi_2 x_3$ и запишем его в мультиплексной форме, применяя вместо функций Q_i их выражения:

$$\begin{aligned} MX(x_3, x_3, x_3, 0, x_3, !x_3, x_3, 1; x_1, x_2, x_4) = \\ = MX(0, x_3, !x_3, 1; \Phi_1, \Phi_2). \end{aligned}$$

Это уравнение имеет единственное решение:

$$\begin{aligned} \Phi_1 &= MX(0, 0, 0, 0, 0, 1, 0, 1; x_1, x_2, x_4) = x_1 x_4; \\ \Phi_2 &= MX(1, 1, 1, 0, 1, 0, 1, 1; x_1, x_2, x_4) = (!x_1 \oplus x_2) \vee !x_4. \end{aligned}$$

Эта декомпозиция соответствует разложению Шеннона по переменной x_3 , так как $f(x_3=0) = \Phi_1$, $f(x_3=1) = \Phi_2$. При этом $f = x_1 x_4 !x_3 \vee ((!x_1 \oplus x_2) \vee !x_4) x_3$.

Пример 3.8. Используя строку 21 табл. 3.5, выполнить декомпозицию функции, рассмотренной в предыдущем примере.

В этом случае $f = \Phi_2 !x_3 \vee \Phi_1 x_3$, и поэтому

$$\begin{aligned} MX(x_3, x_3, x_3, 0, x_3, !x_3, x_3, 1; x_1, x_2, x_4) = \\ = MX(x_3, 1, 0, !x_3; \Phi_1, \Phi_2); \\ \Phi_1 &= MX(0, 0, 0, 1, 0, 1, 0, 0; x_1, x_2, x_4) = (x_1 \oplus x_2) x_4; \\ \Phi_2 &= MX(0, 0, 0, 0, 0, 1, 0, 1; x_1, x_2, x_4) = x_1 x_4. \end{aligned}$$

Эта декомпозиция однотипна с разложением Шеннона по переменной x_3 . При этом $f = x_1 x_4 !x_3 \vee ((x_1 \oplus x_2) x_4) x_3$.

Пример 3.9. Применяя строку 1 табл. 3.5, выполнить декомпозицию функции, рассмотренной в примере 3.7.

В этом случае $f = \Phi_2 \oplus \Phi_1 !x_3$, и поэтому

$$\begin{aligned}
&MX(x_3, x_3, x_3, 0, x_3, !x_3, x_3, 1; x_1, x_2, x_4) = \\
&= MX(0, 1, !x_3, x_3; \Phi_1, \Phi_2); \\
\Phi_1 &= MX(1, 1, 1, 0, 1, 1, 1, 0; x_1, x_2, x_4) = !x_2 \vee !x_4; \\
\Phi_2 &= MX(1, 1, 1, 0, 1, 0, 1, 1; x_1, x_2, x_4) = (!x_1 \oplus x_2) \vee !x_4.
\end{aligned}$$

Эта декомпозиция однотипна с разложением Рида по переменной $!x_3$, так как $f(x_3=1) = \Phi_2$, $f(x_3=0) \oplus f(x_3=1) = \Phi_1$. При этом $f = (!x_1 \oplus x_2) \vee !x_4 \oplus (!x_2 \vee !x_4)!x_3$.

Пример 3.10. Используя строку 5 табл. 3.5, выполнить декомпозицию функции, рассмотренной в примере 3.7.

В этом случае $f = \Phi_1 \oplus \Phi_2 x_3$, и поэтому

$$\begin{aligned}
&MX(x_3, x_3, x_3, 0, x_3, !x_3, x_3, 1; x_1, x_2, x_4) = \\
&= MX(0, x_3, 1, !x_3; \Phi_1, \Phi_2); \\
\Phi_1 &= MX(0, 0, 0, 0, 0, 1, 0, 1; x_1, x_2, x_4) = x_1 x_4; \\
\Phi_2 &= MX(1, 1, 1, 0, 1, 1, 1, 0; x_1, x_2, x_4) = !x_2 \vee !x_4.
\end{aligned}$$

Эта декомпозиция является разложением Рида по переменной x_3 . В ней в отличие от предыдущих обе остаточные функции зависят от двух переменных. При этом $f = x_1 x_4 \oplus (!x_2 \vee !x_4) x_3$.

Пример 3.11. Применяя строку 20 табл. 3.5, выполнить декомпозицию функции

$$f(x_1, x_2, x_3, x_4) = |0011 \ 0010 \ 0110 \ 0111|^T.$$

В этом случае $f = \Phi_1 \oplus !\Phi_2 x_4$, и поэтому

$$\begin{aligned}
&MX(0, 1, 0, !x_4, x_4, !x_4, x_4, 1; x_1, x_2, x_3) = \\
&= MX(x_4, 0, !x_4, 1; \Phi_1, \Phi_2); \\
\Phi_1 &= MX(0, 1, 0, 1, 0, 1, 0, 1; x_1, x_2, x_3) = x_3; \\
\Phi_2 &= MX(1, 1, 1, 0, 0, 0, 0, 1; x_1, x_2, x_3) = !x_1 \oplus x_2 x_3.
\end{aligned}$$

Таким образом, $f = x_3 \oplus (!x_1 \oplus x_2 x_3) x_4$.

Булевы формулы, полученные в двух последних примерах, либо минимальны по числу букв, либо отличаются от минимальной не более чем на одну букву. При этом отметим, что БФУ для них существенно зависят от всех своих переменных, а их столбцы значений содержат четное число единиц и поэтому они не могут быть реализованы БФ, неповторными в базисе $\{\&, \vee, !\}$.

Из изложенного следует, что ответ на вопрос о нахождении простейшей декомпозиции при разложении по последней переменной,

как и обычно в логическом проектировании [11], остается открытым, так как этот ответ в общем случае связан с полным перебором по порядку расположения входных переменных в ТИ и назначением функций Q_i . Достоинство предлагаемого подхода состоит в том, что он позволяет весьма просто, с единых позиций проводить как разложение Шеннона, так и разложение Рида, а также разложения, однотипные с ними, что до сих пор не было известно.

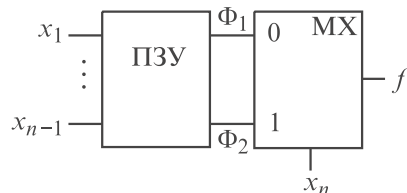


Рис. 3.11

Однако ответ на вопрос, «при каком назначении функций Q_i разложение по последней переменной наименее трудоемко?», находится в строке 6 табл. 3.5, так как при этом схема на рис. 3.10 трансформируется и резко упрощается, превращаясь в схему на рис. 3.11, структура которой аналогична структуре схемы на рис. 3.5, соответствующей разложению Шеннона по крайней левой входной переменной.

Для нахождения функций Φ_1 и Φ_2 (рис. 3.11) по заданной БФУ предлагается простой метод, состоящий в распаковывании (распределении) ее столбца значений. При этом функция Φ_1 (Φ_2) образует-

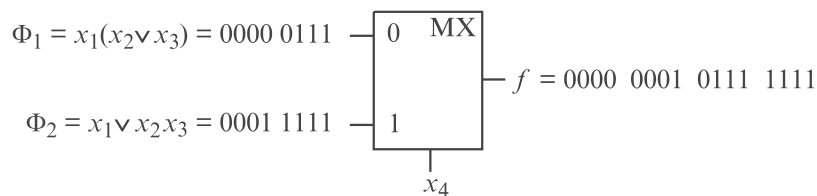


Рис. 3.12

ся из значений функции f , расположенных на четных (нечетных) позициях.

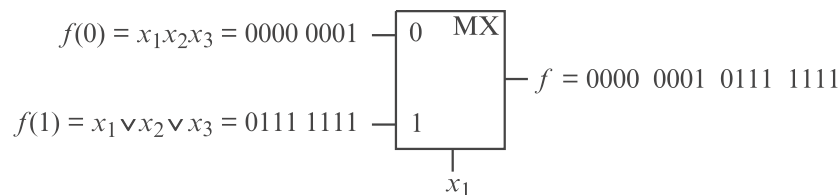


Рис. 3.13

Пример 3.12. Разложить по переменной x_4 булеву функцию

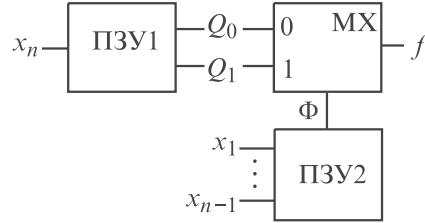


Рис. 3.14

$$f(x_1, x_2, x_3, x_4) = |0000 \ 0001 \ 0111 \ 1111|^T.$$

Искомое разложение приведено на рис. 3.12.

Традиционное разложение этой функции по переменной x_1 приведено на рис. 3.13.

Отметим, что если функция f содержит только два типа фрагментов длиной два, то стандартная схема на рис. 3.9 упрощается и превращается в схему на рис. 3.14.

Разложение по двум и более крайним правым входным переменным. Для построения стандартной схемы, обеспечивающей декомпозицию произвольной БФУ по функциям Q_i , зависящим от $n - k$ крайних правых переменных, множество этих функций должно состоять из всех функций $n - k$ переменных. При этом для $n - k = 2$ справедливо соотношение

$$f = \text{MX}(Q_0(X_1), \dots, Q_{15}(X_1); \Phi_1(X_0), \Phi_2(X_0), \Phi_3(X_0), \Phi_4(X_0)),$$

где $X_0 = \{x_1, \dots, x_{n-2}\}$, $X_1 = \{x_{n-1}, x_n\}$, а для $n - k = 3$ справедливо соотношение

$$f = \text{MX}(Q_0(X_1), \dots, Q_{255}(X_1); \Phi_1(X_0), \dots, \Phi_8(X_0)),$$

где $X_0 = \{x_1, \dots, x_{n-3}\}$, $X_1 = \{x_{n-2}, x_{n-1}, x_n\}$.

Сложность последнего выражения, а тем более выражений для $n - k \geq 3$, привела к тому, что в [49] был рассмотрен только один пример для случая $n - k = 2$. В этой работе вместо МФ использовалось весьма громоздкое выражение вида:

$$f = \bigvee_{i=0}^{15} Q_i(X_1) \tilde{\Phi}_1(X_0) \tilde{\Phi}_2(X_0) \tilde{\Phi}_3(X_0) \tilde{\Phi}_4(X_0),$$

где «~» — символ прямой или инверсной переменной (функции).

Для этого выражения в [49] было отмечено, что если функции $Q_i(X_1)$ зафиксировать, то нахождение декомпозиции сводится к

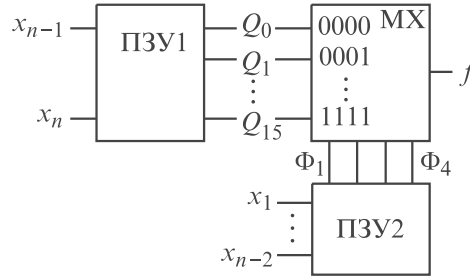


Рис. 3.15

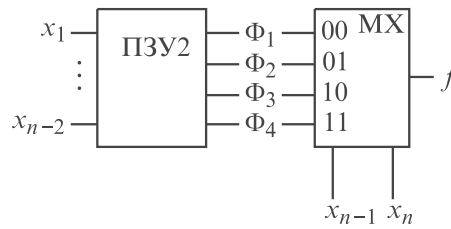


Рис. 3.16

определению функций $\Phi_j(X_0)$. Однако как выполнить фиксацию функций $Q_i(X_1)$, в этой работе не было предложено.

Решим этот вопрос сначала для $n - k = 2$, а затем и в общем случае.

Стандартная схема для $n - k = 2$, являющаяся частным случаем схемы на рис. 3.1, приведена на рис. 3.15. Автором был отмечен удивительный факт, состоящий в том, что среди $16!$ вариантов назначений 16 функций двух переменных Q_i на 16 входов мультиплексора «16 в 1» имеется один вариант, который приводит к замене этой схемы на существенно более простую стандартную схему (рис. 3.16).

Упрощение схемы на рис. 3.15 достигается в том случае, если функции Q_i на входах МХ расположить так, чтобы для каждой из них столбец значений совпадал с двоичным адресом входа МХ, к которому эта функция подключена. При этом приведенное выше

соотношение для $n - k = 2$ также резко упрощается и приобретает вид:

$$f = \text{MX}(\Phi_1(X_0), \Phi_2(X_0), \Phi_3(X_0), \Phi_4(X_0); x_{n-1}, x_n).$$

Таким образом, рассмотренная выше декомпозиция функции f

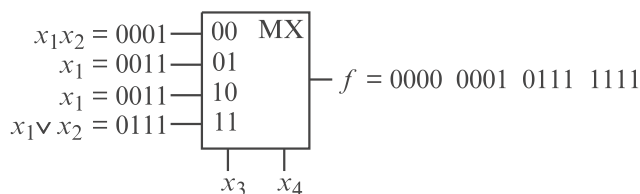


Рис. 3.17

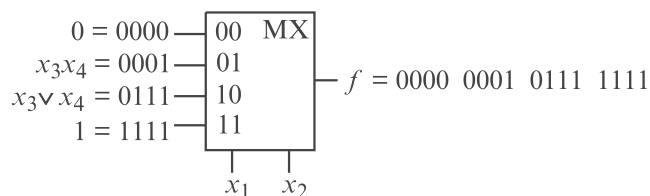


Рис. 3.18

свелась к ее разложению по двум крайним правым переменным — x_{n-1} и x_n . Предложим простейший способ определения функций Φ_j для БФУ, заданной ТИ: столбец значений этой функции «распределяется» так, что столбцы значений функций $\Phi_1, \Phi_2, \Phi_3, \Phi_4$ образуются из значений функции f с номерами 0, 4, 8, 12, ...; 1, 5, 9, 13, ...; 2, 6, 10, 14, ...; 3, 7, 11, 15, ... соответственно.

Пример 3.13. Разложить по двум последним переменным БФУ

$$f(x_1, x_2, x_3, x_4) = |0000 \ 0001 \ 0111 \ 1111|^T.$$

Искомое разложение приведено на рис. 3.17.

Традиционное разложение Шеннона при табличном задании БФУ приведено на рис. 3.18.

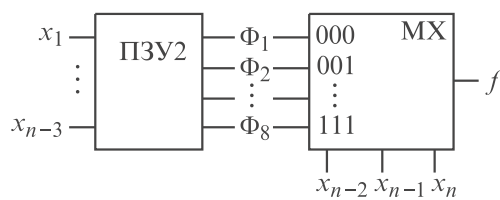


Рис. 3.19

Таким образом, при применении предлагаемого и известного подходов появляется возможность по заданной ТИ без изменения порядка входных переменных в ней получить два решения и выбрать простейшее, что до сих пор было не известно.

Рассмотренный подход, осуществляющий замену декомпозиции на разложение по крайним правым переменным, справедлив также и для произвольных значений $n - k$:

$$f = MX(\Phi_1(X_0), \Phi_2(X_0), \dots, \Phi_{2^{n-k}}(X_0); x_{k+1}, \dots, x_n).$$

При $n - k = 3$ предлагаемый подход позволяет в стандартной схеме, получаемой из схемы на рис. 3.1, исключить ПЗУ1 с тремя входами и 256 выходами и заменить МХ «256 в 1» на МХ «8 в 1» (рис. 3.19).

При разложении по $n - k$ последним переменным столбец значений заданной БФУ «распределяется» на 2^{n-k} функций по аналогии с тем, как это было изложено для $n - k = 1, 2$.

3.3. Использование мультиплексорной декомпозиции при реализации булевых функций

В разд. 3.2 для заданной БФУ при фиксированном значении k определялась декомпозиция по образу мультиплексорной функции относительно переменных Q_i и Φ_j . При этом после нахождения функций Q_i появлялась возможность определения образа декомпозиции относительно переменных Φ_j . Эта задача всегда имеет решение для произвольной БФУ, причем оно в общем случае не является единственным.

Теоретический и практический интерес представляет еще ряд задач:

- задан образ декомпозиции и требуется декомпозировать заданную БФУ по этому образу (определить функции Φ_j);
- задана порождающая функция (ПФ) модуля и требуется декомпозировать БФУ по образу ПФ;
- задана ПФ модуля и требуется декомпозировать БФУ по образу ПФ так, чтобы функции Φ_j обладали определенными свойствами;
- задана ПФ модуля и требуется реализовать БФУ в базисе заданного модуля;
- задана ПФ модуля и требуется реализовать БФУ в базисе заданного модуля так, чтобы число модулей в схеме было минимизировано.

Сложность решения этих задач увеличивается от первой задачи к пятой. При этом отметим, что решение третьей задачи не всегда существует. Так, например, в [42] показано, что существуют БФУ,

для которых простая мажоритарная декомпозиция не может быть построена.

В настоящей работе предлагается единый подход к решению этих задач. Этот подход может быть использован также и для случая применения разнотипных модулей.

3.3.1. Декомпозиция булевых функций по заданному образу

Пусть образ декомпозиции задан БФ вида:

$$F = F(\Phi_1, \dots, \Phi_m, X_1).$$

Задача декомпозиции БФУ $f(x_1, \dots, x_k, X_1)$ по заданному образу может быть сведена к решению относительно переменных $\Phi_1, \dots, \Phi_m$ логического уравнения

$$f(x_1, \dots, x_k, X_1) = F(\Phi_1, \dots, \Phi_m, X_1).$$

Решение этого уравнения, если оно существует, в общем случае имеет вид:

$$\Phi_1 = f_1(x_1, \dots, x_k), \dots, \Phi_m = f_m(x_1, \dots, x_k).$$

Предлагаемый метод состоит из следующих этапов:

— заданный образ декомпозиции записывается в МФ

$$F = MX(Q_0(X_1), \dots, Q_{2^m-1}(X_1); \Phi_1, \dots, \Phi_m),$$

в которой функции Q_i могут быть одинаковыми;

— определяется возможность записи декомпозируемой БФУ в виде:

$$f = MX(f_0(X_1), \dots, f_{2^k-1}(X_1); x_1, \dots, x_k),$$

где $f_t(X_1) = \{Q_0(X_1), \dots, Q_{2^m-1}(X_1)\}$;

— если БФУ записать в таком виде не удастся, то искомая декомпозиция отсутствует;

— если указанная форма представления БФУ существует, то исходное логическое уравнение записывается в МФ:

$$\begin{aligned} & 2^k \\ & MX(Q_q(X_1), \dots, Q_l(X_1); x_1, \dots, x_k) = \\ & = MX(f_0(X_1), \dots, f_{2^m-1}(X_1); \Phi_1, \dots, \Phi_m), \end{aligned}$$

где $q, \dots, l = \{0, \dots, 2^m - 1\}$.

Это уравнение всегда имеет решения, число которых при его записи в явной форме определяется соотношением

$$r = P_0^{t_0} P_1^{t_1}, \dots, P_{2^m-1}^{t_{2^m-1}},$$

где P_i — число функций $Q_i(X_1)$ в правой части уравнения; t_i — число функций $Q_i(X_1)$ в левой части уравнения.

Из этого соотношения следует, что если все функции Q в правой части уравнения различны, то решение единственно. При наличии нескольких решений среди них может быть выбрано одно, обладающее требуемыми свойствами. Предлагаемый подход, по крайней мере теоретически, позволяет проводить декомпозицию произвольной БФУ по произвольному образу. При $X_1 = 0$ декомпозиция всегда существует, но ее нахождение в случае, когда функции Φ_j должны обладать заданными свойствами, связано с огромным перебором.

Пример 3.14. Декомпонировать БФУ $f(x_1, x_2, x_3) = x_1 \# x_2 \# x_3$ по образу $F(\Phi, x_1, x_2, x_3) = \Phi(x_1 \vee x_2 \vee x_3) \vee x_1 x_2 x_3$.

Запишем уравнение

$$x_1 \# x_2 \# x_3 = \Phi(x_1 \vee x_2 \vee x_3) \vee x_1 x_2 x_3.$$

Запишем функцию F в МФ:

$$F = MX(Q_0, Q_1; \Phi),$$

где $Q_0 = x_1 x_2 x_3$, $Q_1 = x_1 \vee x_2 \vee x_3$.

В силу того что функция f не содержит фрагментов Q_0 и Q_1 , исходное уравнение не имеет решения.

Пример 3.15. Декомпонировать БФУ $f(x_1, x_2, x_3) = x_1 \# x_2 \# x_3$ по образу $F(\Phi_1, \Phi_2, F(\Phi_1, \Phi_2, x_2, x_3) x_2, x_3) = \Phi_1(\Phi_2 \vee x_2 \vee x_3) \vee \Phi_2 x_2 x_3$.

Запишем уравнение

$$x_1 \# x_2 \# x_3 = \Phi_1(\Phi_2 \vee x_2 \vee x_3) \vee \Phi_2 x_2 x_3.$$

Запишем это уравнение в МФ:

$$MX(x_2 x_3, x_2 \vee x_3; x_1) = MX(0, x_2 x_3, x_2 \vee x_3, 1; \Phi_1, \Phi_2).$$

Это уравнение имеет единственное решение:

$$\Phi_1 = MX(0, 1; x_1) = x_1;$$

$$\Phi_2 = MX(1, 0; x_1) = !x_1.$$

Пример 3.16. Декомпонировать БФУ $f(x_1, x_2, x_3) = x_1 \# x_2 \# x_3$ по образу $F(\Phi_1, \Phi_2, \Phi_3, \Phi_4) = \Phi_1(\Phi_2 \vee \Phi_3 \vee \Phi_4) \vee \Phi_2 \Phi_3 \Phi_4$.

Запишем уравнение

$$x_1 \# x_2 \# x_3 = \Phi_1(\Phi_2 \vee \Phi_3 \vee \Phi_4) \vee \Phi_2 \Phi_3 \Phi_4.$$

Запишем это уравнение в МФ:

$$\begin{aligned} & MX(0, 0, 0, 1, 0, 1, 1, 1; x_1, x_2, x_3) = \\ & = MX(0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4). \end{aligned}$$

Это уравнение имеет огромное число решений $8^4 \cdot 8^4 = 8^8$.

Среди этих решений, естественно, имеется и решение, найденное без перебора в предыдущем примере:

$$\Phi_1 = MX(0, 0, 0, 0, 1, 1, 1, 1; x_1, x_2, x_3) = x_1;$$

$$\Phi_2 = MX(1, 1, 1, 1, 0, 0, 0, 0; x_1, x_2, x_3) = !x_1;$$

$$\Phi_3 = MX(0, 0, 1, 1, 0, 0, 1, 1; x_1, x_2, x_3) = x_2;$$

$$\Phi_4 = MX(0, 1, 0, 1, 0, 1, 0, 1; x_1, x_2, x_3) = x_3.$$

3.3.2. Декомпозиция булевых функций в базисе заданного модуля

До проведения декомпозиции в этом случае необходимо сначала исследовать функциональные возможности модуля в части допустимых образов декомпозиции, а затем при максимально возможной величине $n - k$ выбрать для заданной БФУ ее образ и, если он существует, выполнить по нему декомпозицию с помощью метода, изложенного в предыдущем разделе.

Рассмотрим вопрос об исследовании функциональных возможностей заданного модуля. В [48] был предложен метод построения настраиваемых модулей, состоящий в том, что для N заданных БФУ, существенно зависящих от n переменных, в порождающую функцию (ПФ) модуля объединялись все разнотипные фрагменты Q_i длиной 2^{n-1} , образующие столбцы значений ТИ этих функций, причем все эти ТИ были записаны при одном и том же порядке переменных. При этом было предложено ПФ модуля строить в виде

$$F = MX(Q_0(x_2, \dots, x_n), \dots, Q_{R-1}(x_2, \dots, x_n); z_1, \dots, z_m),$$

где R — число разнотипных фрагментов; $m = \lceil \log R \rceil$.

Такой модуль при настройке $z_i = \{0, 1, !x_1, x_1\}$ может реализовать любую из N заданных функций, «собирая» ее из двух фрагментов.

Поэтому, памятуя поговорку, «что посеешь, то и пожнешь», естественным образом следует ожидать, что с помощью метода, изло-

женного в предыдущем разделе, можно декомпозировать только такие БФУ, столбцы значений которых образованы фрагментами Q_i , заложенными в ПФ модуля. Однако применительно к настраиваемым модулям имеет место ситуация, при которой «сжать можно больше, чем посеять», что позволяет расширить класс БФУ, декомпозлируемых с помощью рассматриваемого модуля. Например, порождающая функция модуля $F = a(b \vee c \vee d) \vee bcd$ в [48] строилась в результате объединения разнотипных фрагментов длиной четыре ($n - k = 2$) для представителей PN -типов пороговых функций трех переменных. Таким образом, этот модуль предполагалось использовать для построения декомпозиций по следующему образцу:

$$F = \Phi_1(\Phi_2 \vee x_{n-1} \vee x_n) \vee \Phi_2 x_{n-1} x_n,$$

который в МФ имеет вид:

$$F = \text{MX}(0, x_{n-1} x_n, x_{n-1} \vee x_n, 1; \Phi_1, \Phi_2).$$

Однако этой ПФ соответствуют еще три образа декомпозиции:

— при $n - k = 0$

$$\begin{aligned} F &= \Phi_1(\Phi_2 \vee \Phi_3 \vee \Phi_4) \vee \Phi_2 \Phi_3 \Phi_4 = \\ &= \text{MX}(0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4); \end{aligned}$$

— при $n - k = 1$

$$\begin{aligned} F &= \Phi_1(\Phi_2 \vee \Phi_3 \vee x_n) \vee \Phi_2 \Phi_3 x_n = \\ &= \text{MX}(0, 0, 0, x_n, x_n, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3); \end{aligned}$$

— при $n - k = 3$

$$\begin{aligned} F &= \Phi(x_{n-2} \vee x_{n-1} \vee x_n) \vee x_{n-2} x_{n-1} x_n = \\ &= \text{MX}(x_{n-2} x_{n-1} x_n, x_{n-2} \vee x_{n-1} \vee x_n; \Phi), \end{aligned}$$

которые расширяют класс положительно монотонных БФУ, декомпозлируемых с помощью этого модуля. Из приведенных соотношений следует, что если декомпозлируемая функция f при фиксированном $n - k$ состоит из фрагментов, входящих в мультиплексорную форму образа декомпозиции, то при $n - k = 0$ существует восемь в степени 2^n решений; при $n - k = 1$ — $3^{t_0} \cdot 3^{t_1} \cdot 2^{t_2}$ решений; при $n - k = 2, 3$ — единственное решение. При этом отметим, что в общем случае при $n - k = 3$ $\Phi = f(x_1, \dots, x_{n-3})$; при $n - k = 2$ $\Phi_j = f_j(x_1, \dots, x_{n-2})$; при $n - k = 1$ $\Phi_j = f_j(x_1, \dots, x_{n-1})$; при $n - k = 0$ $\Phi_j = f_j(x_1, \dots, x_n)$.

Отсюда следует, что простейшую декомпозицию по критерию гарантированной зависимости функций Φ_j от минимального числа переменных следует искать при максимально возможной величине $n - k$, что ограничивает или исключает перебор. Если при некотором значении $n - k$ декомпозиция найдена, то это не исключает того, что, уменьшая величину $n - k$ за счет увеличения перебора (числа решений), может быть найдено и более простое, например по числу букв, решение. При этом отметим, что среди этих решений находятся также и все решения, найденные при больших значениях $n - k$.

Из изложенного следует, что эффективное использование модуля связано с исследованием его функциональных возможностей в части определения всех образов декомпозиции, которые он порождает.

Будем называть образы декомпозиции, получаемые в результате замены входных переменных в ПФ переменными x_i и Φ_j , основными функциональными возможностями модуля. Полное исследование основных функциональных возможностей модуля может быть выполнено только при учете всех допустимых перестановок несимметричных входных переменных в таблице истинности ПФ модуля, что связано с перебором. Исследование дополнительных возможностей модуля осуществляется при условии, что кроме прямых переменных доступны также и их инверсии.

При этом для рассматриваемого модуля, в частности, удастся установить, что он позволяет при $n - k = 1$ декомпозировать отрицательно монотонные функции, применяя уравнение

$$f(x_1, \dots, x_n) = \Phi_1(\Phi_2 \vee \Phi_3 \vee !x_n) \vee \Phi_2 \Phi_3 !x_n,$$

которое в «полумультимплексорной» форме имеет вид:

$$f(x_1, \dots, x_n) = MX(0, 0, 0, !x_n, !x_n, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3).$$

Пример 3.17. Декомпозировать по ПФ рассматриваемого модуля булеву функцию

$$f(x_1, x_2, x_3, x_4) = |0001 \ 0101 \ 0101 \ 0111|^T.$$

Из рассмотрения столбца значений БФУ следует, что она не может быть декомпозирована при $n - k = 3, 2$, так как при этих значениях она содержит «запрещенные» для соответствующих декомпозиций фрагменты. Так как при $n - k = 1$ все фрагменты длиной два разрешены, то искомую декомпозицию будем определять в результате решения уравнения

$$f(x_1, x_2, x_3, x_4) = \Phi_1(\Phi_2 \vee \Phi_3 \vee x_4) \vee \Phi_2 \Phi_3 x_4,$$

которое в МФ имеет вид:

$$\begin{aligned} \text{MX}(0, x_4, x_4, x_4, x_4, x_4, x_4, 1; x_1, x_2, x_3) = \\ = \text{MX}(0, 0, 0, x_4, x_4, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3). \end{aligned}$$

Это уравнение имеет $3^1 \cdot 3^1 \cdot 2^6 = 576$ решений, среди которых отсутствуют такие, в которых все функции Φ_j равны переменным. Однако среди этих решений существует такое, в котором две функции Φ_j равны переменным:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, 1, 0, 1, 0, 1, 1; x_1, x_2, x_3) = x_1 \# x_2 \# !x_3; \\ \Phi_2 &= \text{MX}(0, 1, 0, 1, 0, 1, 0, 1; x_1, x_2, x_3) = x_3; \\ \Phi_3 &= \text{MX}(0, 1, 0, 1, 0, 1, 0, 1; x_1, x_2, x_3) = x_3. \end{aligned}$$

Если уменьшить величину $n - k$ до нуля, то

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= \Phi_1(\Phi_2 \vee \Phi_3 \vee \Phi_4) \vee \Phi_2 \Phi_3 \Phi_4; \\ \text{MX}(0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1; x_1, x_2, x_3, x_4) &= \\ = \text{MX}(0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4). \end{aligned}$$

Это уравнение имеет огромное число решений, равное 8^{16} , среди которых содержится простейшее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1; x_1, x_2, x_3) = x_4; \\ \Phi_2 &= \text{MX}(0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1; x_1, x_2, x_3) = x_1; \\ \Phi_3 &= \text{MX}(0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1; x_1, x_2, x_3) = x_2; \\ \Phi_4 &= \text{MX}(0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0; x_1, x_2, x_3) = x_3. \end{aligned}$$

Пример 3.18. Декомпонировать по ПФ рассматриваемого модуля БФУ из предыдущего примера, записав ее при другом порядке переменных:

$$f(x_4, x_1, x_2, x_3) = |0000 \ 0001 \ 0111 \ 1111|^T.$$

В этом случае минимальное решение может быть найдено уже при $n - k = 3$:

$$\begin{aligned} f(x_4, x_1, x_2, x_3) &= \Phi(x_1 \vee x_2 \vee x_3) \vee x_1 x_2 x_3; \\ \text{MX}(x_1 x_2 x_3, x_1 \vee x_2 \vee x_3; x_4) &= \text{MX}(x_1 x_2 x_3, x_1 \vee x_2 \vee x_3; \Phi). \end{aligned}$$

Это уравнение имеет единственное решение $\Phi = x_4$.

Из рассмотренных примеров следует, что для построения простейшей декомпозиции функции f по ПФ модуля возможны два подхода:

- исследование основных и дополнительных функциональных возможностей модуля и перебор по порядку расположения входных переменных в ТИ заданной функции f ;

- однократное исследование всех функциональных возможностей модуля, основанное на переборе входных переменных в ТИ порождающей функции модуля, и исключение перебора в ТИ заданной функции.

При «ручной» декомпозиции первый подход следует применять для достаточно сложных модулей, у которых ПФ несимметричны и зависят от пяти и более переменных. Для простых модулей, порождающие функции которых симметричны или не зависят более чем от четырех переменных, более целесообразен второй подход.

Исследование основных функциональных возможностей модуля осуществляется за счет определения мультиплексорной формы представления ПФ модуля при $n - k = 0, 1, \dots, m - 1$, где m — число входов модуля. Полное исследование основных возможностей модуля выполняется следующим образом:

- используя m входных переменных, строится исходная таблица истинности для ПФ модуля;

- по исходной ТИ за счет изменения порядка входных переменных строятся новые ТИ, столбцы значений которых отличаются и между собой, и от столбца значений исходной ТИ. Пусть, например, число ТИ с различными столбцами значений равно M ;

- в каждой из M таблиц истинности выполняется замена кортежа входных переменных на кортеж новых переменных $\Phi_1, \dots, \Phi_m$. При этом строятся M порождающих функций $F_1(\Phi_1, \dots, \Phi_m), \dots, F_M(\Phi_1, \dots, \Phi_m)$, имеющих различные столбцы значений. Этим порождающим функциям соответствует «сборка» реализуемой БФУ из фрагментов длиной $2^0 = 1$ ($n - k = 0$). Так как при перестановке входных переменных номенклатура фрагментов длиной один не изменяется, то из этих функций бывает целесообразно использовать только функцию $F_1(\Phi_1, \dots, \Phi_m)$;

- в каждой из M таблиц истинности выполняется замена кортежа входных переменных на кортеж новых переменных $\Phi_1, \dots, \Phi_{m-1}, x_n$. При этом строятся M порождающих функций $F_1(\Phi_1, \dots, \Phi_{m-1}, x_n), \dots, F_M(\Phi_1, \dots, \Phi_{m-1}, x_n)$, имеющих различные столбцы значений. Этим ПФ соответствует «сборка» реализуемой БФУ из фрагментов длиной 2^1 , зависящих от переменной x_n ($n - k = 1$).

Эти соотношения могут быть названы декомпозицией по функциям $\Phi_1, \dots, \Phi_{m-1}$, или разложением по одной (последней) переменной x_n .

Разложение по функции $Q(X_1)$ в этом случае строится при замене в этих соотношениях переменной x_n на эту функцию. Для рассмот-

ренных ПФ бывает целесообразным применять только те из них, которые различаются номенклатурой фрагментов длиной два. Например, функции $F_1 = (!\Phi_1 \oplus \Phi_2)x_n$ соответствует столбец значений с двумя типами фрагментов длиной два, а функциям $F_2 = (!\Phi_2 \oplus x_n)\Phi_1$ и $F_3 = (!\Phi_1 \oplus x_n)\Phi_2$ соответствуют различные столбцы значений с тремя одинаковыми типами фрагментов длиной два;

— увеличение величины $n - k$ продолжается до тех пор, пока не выполнится условие $n - k = m - 1$. При этом могут быть построены M порождающих функций $F_1(\Phi, x_{n-m+2}, \dots, x_n), \dots, F_M(\Phi, x_{n-m+2}, \dots, x_n)$, имеющих различные столбцы значений. Этим ПФ соответствует «сборка» реализуемой БФУ из фрагментов длиной 2^{m-1} , зависящих от переменных $x_{n-m+2}, \dots, x_n$.

На этом завершается исследование основных возможностей модуля. Исследование дополнительных возможностей не проводится при $n - k = 0$, а также при $n - k = 1$, если столбец значений образа декомпозиции содержит все четыре типа фрагментов длиной два: $0, 1, !x_n, x_n$. Исследование дополнительных возможностей в остальных случаях может быть заменено определением однотипности фрагментов заданной функции f и рассматриваемого образа декомпозиции при выбранном значении $n - k$. При этом при подстановке фрагментов функции f в мультиплексорную форму образа декомпозиции получаемая формула должна быть N -однотипной с булевой формулой образа.

Пример 3.19. Определить возможность декомпозиции при $n - k = 2$ по ПФ рассматриваемого модуля для БФУ

$$f(x_1, x_2, x_3, x_4) = |0000 \ 0100 \ 0111 \ 1111|^T.$$

Составим выражение $F_1 = MX(0, !x_3x_4, x_3 \vee x_4, 1; \Phi_1, \Phi_2)$. Так как формула $F_1 = \Phi_1(\Phi_2 \vee x_3 \vee x_4) \vee \Phi_2!x_3x_4$ не является N -однотипной с ПФ модуля $F = \Phi_1(\Phi_2 \vee x_3 \vee x_4) \vee \Phi_2x_3x_4$, то искомая декомпозиция не может быть построена.

Пример 3.20. Определить возможность декомпозиции при $n - k = 2$ по ПФ рассматриваемого модуля для БФУ

$$f(x_1, x_2, x_3, x_4) = |0000 \ 0100 \ 1101 \ 1101|^T.$$

Составим выражение $F_1 = MX(0, !x_3x_4, !x_3 \vee x_4, 1; \Phi_1, \Phi_2)$. Так как формула $F_1 = \Phi_1(\Phi_2 \vee !x_3 \vee x_4) \vee \Phi_2!x_3x_4$ является N -однотипной с ПФ модуля $F = \Phi_1(\Phi_2 \vee x_3 \vee x_4) \vee \Phi_2x_3x_4$, то искомая декомпозиция может быть построена:

$$\begin{aligned}
f(x_1, x_2, x_3, x_4) &= \Phi_1(\Phi_2 \vee !x_3 \vee x_4) \vee \Phi_2 !x_3 x_4; \\
MX(0, !x_3 x_4, !x_3 \vee x_4, !x_3 \vee x_4; x_1, x_2) &= \\
&= MX(0, !x_3 x_4, !x_3 \vee x_4, 1; \Phi_1, \Phi_2); \\
\Phi_1 &= MX(0, 0, 1, 1; x_1, x_2) = x_1; \\
\Phi_2 &= MX(0, 1, 0, 0; x_1, x_2) = !x_1 x_2.
\end{aligned}$$

3.3.3. Мультиплексорный метод реализации булевых функций

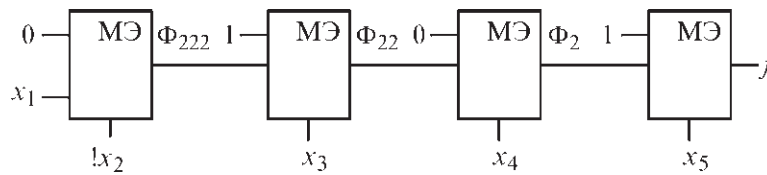


Рис. 3.20

Этот метод базируется на:

- исследовании функциональных возможностей применяемого элемента в части образов декомпозиции, которые он порождает;
- мультиплексорном разложении заданной БФУ по одному из образов декомпозиции, порождаемому элементом;
- мультиплексорной декомпозиции заданной БФУ с последующей реализацией образа декомпозиции и остаточных функций схемами из используемых элементов, которые строятся на основе мультиплексорного разложения.

В зависимости от свойств применяемого элемента этот метод может дополняться и другими приемами, упрощающими реализацию. Метод в отличие от формульного метода позволяет использовать повторность ПФ и ее остаточных.

Пример 3.21. Реализовать булеву функцию

$$f(x_1, \dots, x_5) = |0101 \ 0111 \ 0101 \ 0111 \ 0111 \ 0111 \ 0101 \ 0111|^T$$

схемой из трехвходовых мажоритарных элементов.

Эта БФУ содержит нечетное число единиц, что является необходимым условием бесповторности. Она реализуется ББФ $f = (x_1 !x_2 \vee \vee x_3) x_4 \vee x_5$.

1. Бесповторная каскадная схема, построенная формульным методом, приведена на рис. 3.20.

2. Так как заданная БФУ монотонна по переменной x_5 , то выполним мажоритарное разложение по этой переменной:

$$f(x_1, \dots, x_5) = \Phi_1 \# \Phi_2 \# x_5;$$

$$\text{MX}(x_5, x_5, x_5, 1, x_5, x_5, x_5, 1, x_5, 1, x_5, 1, x_5, x_5, x_5, 1;$$

$$x_1, x_2, x_3, x_4) = \text{MX}(0, x_5, x_5, 1; \Phi_1, \Phi_2).$$

Это уравнение имеет $2^{11} = 2048$ решений, одно из которых следующее:

$$\Phi_1 = \text{MX}(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1; x_1, x_2, x_3, x_4) = 1;$$

$$\Phi_2 = \text{MX}(0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1; x_1, x_2, x_3, x_4) =$$

$$= (x_1!x_2 \vee x_3)x_4.$$

Функция Φ_2 монотонна по переменной x_4 , и поэтому

$$\Phi_2(x_1, \dots, x_4) = \Phi_{21} \# \Phi_{22} \# x_4;$$

$$\text{MX}(0, x_4, 0, x_4, x_4, x_4, 0, x_4; x_1, x_2, x_3) =$$

$$= \text{MX}(0, x_4, x_4, 1; \Phi_{21}, \Phi_{22}).$$

Это уравнение имеет 32 решения, одно из которых следующее:

$$\Phi_{21} = \text{MX}(0, 0, 0, 0, 0, 0, 0, 0; x_1, x_2, x_3) = 0;$$

$$\Phi_{22} = \text{MX}(0, 1, 0, 1, 1, 1, 0, 1; x_1, x_2, x_3) = x_1!x_2 \vee x_3.$$

Функция Φ_{22} монотонна по переменной x_3 , и поэтому

$$\Phi_{22}(x_1, x_2, x_3) = \Phi_{221} \# \Phi_{222} \# x_3;$$

$$\text{MX}(x_3, x_3, 1, x_3; x_1, x_2) = \text{MX}(0, x_3, x_3, 1; \Phi_{221}, \Phi_{222}).$$

Это уравнение имеет восемь решений, одно из которых следующее:

$$\Phi_{221} = \text{MX}(1, 1, 1, 1; x_1, x_2) = 1;$$

$$\Phi_{222} = \text{MX}(0, 0, 1, 0; x_1, x_2) = x_1!x_2.$$

Функция Φ_{222} монотонна по переменной x_2 , и поэтому

$$\Phi_{222}(x_1, x_2) = \Phi_{2221} \# \Phi_{2222} \# !x_2;$$

$$\text{MX}(0, !x_2; x_1) = \text{MX}(0, !x_2, !x_2, 1; \Phi_{2221}, \Phi_{2222}).$$

Это уравнение имеет два решения, одно из которых следующее:

$$\Phi_{2221} = \text{MX}(0, 0; x_1) = 0;$$

$$\Phi_{2222} = \text{MX}(0, 1; x_1) = x_1.$$

В результате изложенного получается неповторная каскадная схема (рис. 3.20), которая построена формульным методом существенно менее трудоемко. Однако формульный метод в данном случае по заданной формуле позволяет построить только одну схему, в то время как мультиплексорный метод, используя таблицу истинности, позволяет даже без перестановки столбцов входных переменных в ней получать большое число решений (схем).

Пример 3.22. Реализовать БФ $f = x_1(x_2x_3 \vee x_4x_5)$ схемой из 3-универсальных модулей, структура каждого из которых описывается булевой формулой

$$F(a, b, c, d) = a(b \vee c \vee d) \vee bcd = |0000 \ 0001 \ 0111 \ 1111|^T.$$

По формуле f построим ТИ:

$$\begin{aligned} f(x_1, x_2, x_3, x_4, x_5) = \\ = |0000 \ 0000 \ 0000 \ 0000 \ 0001 \ 0001 \ 0001 \ 1111|^T. \end{aligned}$$

При $n - k = 2$ предположим, что $Q_0 = 0$, $Q_1 = x_4x_5$, $Q_2 = x_4 \vee x_5$, $Q_3 = 1$, и запишем уравнение:

$$\begin{aligned} x_1(x_2x_3 \vee x_4x_5) &= \Phi_1(\Phi_2 \vee x_4 \vee x_5) \vee \Phi_2x_4x_5; \\ \text{MX}(Q_0, Q_0, Q_0, Q_0, Q_1, Q_1, Q_1, Q_3; x_1, x_2, x_3) &= \\ &= \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2). \end{aligned}$$

Это уравнение имеет единственное решение:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, 0, 0, 0, 0, 0, 1; x_1, x_2, x_3) = x_1x_2x_3; \\ \Phi_2 &= \text{MX}(0, 0, 0, 0, 1, 1, 1, 1; x_1, x_2, x_3) = x_1. \end{aligned}$$

Для реализации функции Φ_1 вновь запишем и решим уравнение при $n - k = 2$, которое также имеет единственное решение:

$$\begin{aligned} x_1x_2x_3 &= \Phi_{11}(\Phi_{21} \vee x_2 \vee x_3) \vee \Phi_{21}x_2x_3; \\ \text{MX}(Q_0, Q_1; x_1) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_{11}, \Phi_{21}); \\ \Phi_{11} &= \text{MX}(0, 0; x_1) = 0; \\ \Phi_{21} &= \text{MX}(0, 1; x_1) = x_1. \end{aligned}$$

Таким образом, заданная БФ с помощью мультиплексорного метода, использующего на каждом шаге разделительную декомпозицию, реализуется неразделительной (повторной) схемой из двух модулей:

$$\begin{aligned} \Phi &= 0(x_1 \vee x_2 \vee x_3) \vee x_1x_2x_3 = x_1x_2x_3, \\ f &= \Phi(x_1 \vee x_4 \vee x_5) \vee x_1x_4x_5. \end{aligned}$$

При применении формульного метода эта БФ реализуется бесповторной схемой из трех модулей:

$$\begin{aligned}f_1 &= 0(1 \vee x_2 \vee x_3) \vee 1x_2x_3 = x_2x_3, \\f_2 &= f_1(1 \vee x_4 \vee x_5) \vee 1x_4x_5 = f_1 \vee x_4x_5, \\f &= 0(1 \vee x_1 \vee f_2) \vee 1x_1f_2 = x_1f_2.\end{aligned}$$

Следовательно, для рассмотренной формулы мультиплексорный метод является более эффективным по сравнению с формульным методом, так как в первом случае все четыре входа выходного модуля используются в качестве информационных входов, а во втором — у каждого модуля в схеме в качестве информационных применяется не более чем три входа. Формульный метод не позволяет реализовать формулу Φ , так как она отсутствует в заданной формуле.

Такую же сложность при использовании мультиплексорного метода имеет и формула $f_3 = x_1 \vee (x_2 \vee x_3)(x_4 \vee x_5)$, двойственная с заданной, которая с помощью формульного метода также реализуется схемой из трех модулей:

$$\begin{aligned}x_1 \vee (x_2 \vee x_3)(x_4 \vee x_5) &= \Phi_1(\Phi_2 \vee x_4 \vee x_5) \vee \Phi_2x_4x_5; \\MX(Q_0, Q_2, Q_2, Q_2, Q_3, Q_3, Q_3, Q_3; x_1, x_2, x_3) &= \\&= MX(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2); \\ \Phi_1 &= MX(0, 1, 1, 1, 1, 1, 1, 1; x_1, x_2, x_3) = x_1 \vee x_2 \vee x_3; \\ \Phi_2 &= MX(0, 0, 0, 0, 1, 1, 1, 1; x_1, x_2, x_3) = x_1.\end{aligned}$$

Для реализации функции Φ_1 вновь запишем и решим уравнение при $n - k = 2$, которое также имеет единственное решение:

$$\begin{aligned}x_1 \vee x_2 \vee x_3 &= \Phi_{11}(\Phi_{21} \vee x_2 \vee x_3) \vee \Phi_{21}x_2x_3; \\MX(Q_2, Q_3; x_1) &= MX(Q_0, Q_1, Q_2, Q_3; \Phi_{11}, \Phi_{21}); \\ \Phi_{11} &= MX(1, 1; x_1) = 1; \\ \Phi_{21} &= MX(0, 1; x_1) = x_1.\end{aligned}$$

Таким образом,

$$\begin{aligned}\Phi &= 1(x_1 \vee x_2 \vee x_3) \vee x_1x_2x_3 = x_1x_2x_3, \\f_3 &= \Phi(x_1 \vee x_4 \vee x_5) \vee x_1x_4x_5.\end{aligned}$$

Так как только эти два PN -типа формул в базисе $\{\&, \vee, !\}$ при $h = 5$ реализуются формульным методом схемой из трех таких модулей, то тем самым вместо соотношения

$$2 \leq L(5, 3) \leq 3$$

доказано, что

$$L(5, 3) = 2,$$

что до сих пор было не известно.

Особенности применения этого метода при синтезе схем в различных элементных базисах подробно рассмотрены в гл. 4—6.

Изложенный метод может быть модифицирован для реализации на его основе произвольной булевой функции n переменных схемой из табличных преобразователей [51], каждый из которых реализован, например, постоянным запоминающим устройством (ПЗУ) с m входами и одним выходом ($m < n$). Такое ПЗУ, несмотря на то что оно не имеет настроечных входов, может за счет программирования реализовать произвольную БФУ m и менее переменных. При этом отметим, что при синтезе схем из логических элементов для каждого из них априори известны число входов и его порождающая функция, которая может быть произвольной, в то время как при синтезе схем из ПЗУ для каждого из них априори известно только число его входов, а при синтезе схем из пороговых элементов каждого элемента априори не известно даже это число.

Предположим, что требуется реализовать БФУ, рассмотренную в примере 3.22 схемой из ПЗУ с $m \leq 4$. Если использовать схему из двух ПЗУ, аналогичную с приведенной в [51] для логического блока, применяемого в семействе микросхем XC4000 фирмы «Xilinx», то одно ПЗУ с $m = 3$ реализует разложение Шеннона по переменной x_1 , а второе ПЗУ с $m = 4$ — остаточную функцию от остальных переменных. Покажем, что модификация предлагаемого метода позволяет упростить эту схему, а также схему, построенную в примере 3.22, за счет использования более широких функциональных возможностей ПЗУ по сравнению с функциональными возможностями рассмотренного модуля.

Так как при $m = 4$ каждое ПЗУ можно запрограммировать на реализацию только одного типа фрагментов длиной 16, а ТИ заданной БФУ состоит из двух разнотипных фрагментов этой длины, то на одном ПЗУ эта функция не может быть реализована. Эта функция также не может быть декомпозирована по фрагментам длиной восемь, так как ее ТИ состоит из трех типов такой же длины, а ПЗУ может реализовать только два фрагмента этой же длины ($n - k = 1$).

Декомпозиция может быть выполнена по фрагментам длиной четыре, так как ТИ заданной функции состоит из трех разнотипных фрагментов такой же длины, а ПЗУ может реализовать четыре фрагмента этой же длины ($n - k = 2$), три из которых определяются номенклатурой фрагментов реализуемой функции, а четвертый может быть выбран произвольно. Возможность произвольного выбора этого фрагмента позволяет упростить по сравнению с примером 3.22

остаточную функцию, получаемую после первого шага декомпозиции.

Полагая, что $Q_0 = 0$, $Q_1 = x_4 x_5$, $Q_2 = 0$, $Q_3 = 1$, запишем уравнение:

$$\begin{aligned} x_1(x_2 x_3 \vee x_4 x_5) &= \Phi_2(\Phi_1 \vee x_4 x_5); \\ \text{MX}(Q_0? Q_2, Q_0? Q_2, Q_0? Q_2, Q_0? Q_2, Q_1, Q_1, Q_1, Q_3; x_1, x_2, x_3) &= \\ &= \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2). \end{aligned}$$

Из 16 решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, 0, 1, 0, 0, 0, 1; x_1, x_2, x_3) = x_2 x_3; \\ \Phi_2 &= \text{MX}(0, 0, 0, 0, 1, 1, 1, 1; x_1, x_2, x_3) = x_1. \end{aligned}$$

Таким образом, модификация предлагаемого метода, которая может быть названа мультиплексорным методом с настраиваемым образом декомпозиции, и более широкие функциональные возможности ПЗУ позволили построить подфункцию Φ_1 , зависящую только от двух переменных, и тем самым реализовать заданную БФУ схемой из двух ПЗУ — с $m = 4$ и $m = 2$.

Отметим, что в данном случае формульный метод строит из тех же элементов схему, реализующую другую декомпозицию: $\Phi = x_2 x_3 \vee x_4 x_5$, $f = x_1 \Phi$.

Отметим также, что уже при $m = 3$ постоянное запоминающее устройство может быть использовано для реализации разложения Шеннона заданной БФУ по крайней правой входной переменной ее таблицы истинности, так как при этом значении m в столбце значений ПЗУ могут быть «расположены» все четыре типа фрагментов длиной два.

Рассмотренная модификация мультиплексорного метода применима также и для реализации БФУ в базисе мультиплексоров « 2^m в 1».

Из изложенного следует, что при использовании ПЗУ и мультиплексоров с помощью изложенного метода при $m \geq 3$ каждый шаг декомпозиции является результативным, а настройка каждого из них определяется структурой столбца значений ТИ реализуемой БФУ (ее остаточной), в то время как при применении других типов элементов результативность шага декомпозиции определяется совпадением типов фрагментов длиной более одного в столбцах значений таблиц истинности реализуемой БФУ (ее остаточной) и порождающей функции элемента.

3.4. Сравнение мультиплексорного метода с известными методами

3.4.1. Мультиплексорный метод и декомпозиционный метод Миллера

В [32] на основе применения соотношения (3.1) рассмотрен декомпозиционный метод синтеза в двух постановках задачи:

- заданы функции f и F и требуется определить функции Φ_j ;
- заданы функции f и Φ_j и требуется определить функцию F .

При этом функция f может быть как полностью определенной, так и определенной лишь частично.

Этот метод базируется на весьма специфическом математическом аппарате, использующем нетрадиционные для булевой алгебры термины, такие, например, как частичное упорядочение функций, расширение функций, совместимость и несовместимость множеств, максимальная совместимость множеств и т.д. Это усложняет понимание и применение этого метода.

Покажем на примерах из [32], что предлагаемый существенно более простой мультиплексорный метод для многих БФУ приводит к тем же результатам.

Пример 3.23. При $m = 1$ и $n - k = 2$ построить функцию Φ , если F — мажоритарная функция трех переменных, а f — частично определенная функция вида

$$f(x_1, x_2, x_3, x_4) = |00 \text{ -- -- } -11 \text{ } 0111 \text{ } 0 \text{ -- } -1|^T.$$

1.1. Эта ТИ содержит четыре фрагмента длиной четыре, которые могут быть доопределены до трех типов фрагментов, обеспечивающих получение в качестве образа декомпозиции мажоранты (табл. 3.6).

Т а б л и ц а 3.6

x_3	x_4	g_1	g_1	g_2	g_3
0	0	0	— 1	0	0
0	1	0	— 1	1	— 1
1	0	— 0	1	1	— 1
1	1	— 0	1	1	1

1.2. При $Q_0 = 0$, $Q = Q_1 = Q_2 = x_3 \vee x_4$, $Q_3 = 1$ справедливо уравнение

$$f(x_1, x_2, x_3, x_4) = \Phi_1 \# \Phi_2 \# (x_3 \vee x_4),$$

которое в МФ имеет вид:

$$MX(0, 1, Q, Q; x_1, x_2) = MX(0, Q, Q, 1; \Phi_1, \Phi_2).$$

Это уравнение имеет четыре решения, два из которых неизоморфны:

$$\begin{cases} \Phi_1 = MX(0, 1, 0, 0; x_1, x_2) = !x_1 x_2; \\ \Phi_2 = MX(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2; \\ \Phi_1 = MX(0, 1, 0, 1; x_1, x_2) = x_2; \\ \Phi_2 = MX(0, 1, 1, 0; x_1, x_2) = x_1 \oplus x_2. \end{cases}$$

Таким образом, в данном случае

$$f = !x_1 x_2 \# (x_1 \vee x_2) \# (x_3 \vee x_4) \text{ или } f = x_2 \# (x_1 \oplus x_2) \# (x_3 \vee x_4).$$

2. Для заданной БФУ построим карту декомпозиции (табл. 3.7).

Т а б л и ц а 3.7

$x_1 x_2$	$x_3 x_4$			
	00	01	10	11
00	0	0	—	—
01	—	—	1	1
10	0	1	1	1
11	0	—	—	1

2.1. Из этой таблицы следует, что функция

$$f(x_3, x_4, x_1, x_2) = |0 - 00 \ 0 - 1 - \ - 11 - \ - 111|^T$$

также содержит четыре фрагмента длиной четыре, которые могут быть доопределены двумя способами до трех типов фрагментов, обеспечивающих получение мажоранты в качестве образа декомпозиции (табл. 3.8).

Т а б л и ц а 3.8

x_1	x_2	g_0	g_1	g_2	g_3	g_0	g_1	g_2	g_3
0	0	0	0	0	1	0	0	0	1
0	1	0	1	1	1	0	1	1	1
1	0	0	1	1	1	0	1	1	1
1	1	0	0	0	1	0	1	1	1

При $Q_0 = g_0, Q_1 = Q_2 = g_1, Q_3 = g_3$ в первом случае

$$f(x_3, x_4, x_1, x_2) = \Phi_1 \# \Phi_2 \# (x_1 \oplus x_2),$$

а во втором —

$$f(x_3, x_4, x_1, x_2) = \Phi_1 \# \Phi_2 \# (x_1 \vee x_2).$$

И в том и другом случае уравнение в МФ имеет вид:

$$MX(0, Q, Q, 1; x_3, x_4) = MX(0, Q, Q, 1; \Phi_1, \Phi_2).$$

Таким образом, $\Phi_1 = x_3, \Phi_2 = x_4$. Следовательно,

$$f = x_3 \# x_4 \# (x_1 \oplus x_2) \text{ или } f = x_3 \# x_4 \# (x_1 \vee x_2).$$

Именно эти решения и приведены в [32].

Пример 3.24. При $m = 1, n - k = 2, Q = \{x_1 x_2, x_1 \oplus x_2, x_1 \vee x_2\}$ построить функцию F для БФУ

$$f(x_1, x_2, x_3, x_4) = |0000 \ 0111 \ 0111 \ 0000|^T.$$

Используя карту декомпозиции, запишем заданную БФУ в виде:

$$f(x_3, x_4, x_1, x_2) = |0000 \ 0110 \ 0110 \ 0110|^T.$$

Полагая $Q_0 = 0, Q_1 = x_1 \oplus x_2$, запишем логическое уравнение:

$$f(x_3, x_4, x_1, x_2) = MX(Q_0, Q_1; \Phi),$$

которое в МФ имеет вид:

$$\begin{aligned} \text{MX}(\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_1, \mathcal{Q}_1; x_3, x_4) &= \text{MX}(\mathcal{Q}_0, \mathcal{Q}_1; \Phi); \\ \Phi &= \text{MX}(0, 1, 1, 1; x_3, x_4) = x_3 \vee x_4. \end{aligned}$$

Следовательно, $f = \mathcal{Q}_1(x_3 \vee x_4)$, $\mathcal{Q}_1 = x_1 \oplus x_2$, что совпадает с решением, приведенным в [32].

Пример 3.25. При $m = 1$, $n - k = 2$, $\mathcal{Q} = \{x_1 x_2, x_1 \oplus x_2, x_1 \vee x_2\}$ построить функцию F для БФУ

$$f(x_1, x_2, x_3, x_4) = |0011 \ -010 \ -010 \ -011|^T.$$

При заданном порядке переменных разрешенные функции $\mathcal{Q}$ не могут быть получены. Построим карту декомпозиции (табл. 3.9).

Т а б л и ц а 3.9

$x_1 x_2$	$x_3 x_4$			
	00	01	10	11
00	0	0	1	1
01	—	0	1	0
10	—	0	1	0
11	—	0	1	1

Доопределим эту функцию с новым порядком переменных следующим образом:

$$f(x_3, x_4, x_1, x_2) = |0110 \ 0000 \ 1111 \ 1001|^T.$$

При $\mathcal{Q}_0 = 0$, $\mathcal{Q}_1 = x_1 \oplus x_2$, $\mathcal{Q}_2 = !(x_1 \oplus x_2)$, $\mathcal{Q}_3 = 1$ справедливы соотношения:

$$\begin{aligned} f(x_3, x_4, x_1, x_2) &= \text{MX}(\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3; \Phi_1, \Phi_2); \\ \text{MX}(\mathcal{Q}_1, \mathcal{Q}_0, \mathcal{Q}_3, \mathcal{Q}_2; x_3, x_4) &= \text{MX}(\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3; \Phi_1, \Phi_2); \\ \Phi_1 &= \text{MX}(0, 0, 1, 1; x_3, x_4) = x_3; \\ \Phi_2 &= \text{MX}(1, 0, 1, 0; x_3, x_4) = !x_4. \end{aligned}$$

Таким образом, $f = \Phi_1 !\mathcal{Q} \vee \Phi_2 \mathcal{Q} = x_3 !\mathcal{Q} \vee !x_4 \mathcal{Q}$, где $\mathcal{Q} = x_1 \oplus x_2$. Применяя разложение Рида по переменной x_4 , получим БФ:

$$f = (x_3 \vee \mathcal{Q}) \oplus \mathcal{Q}x_4,$$

в которой устранены инверсии. Это решение найдено в [32] весьма трудоемко.

3.4.2. Мультиплексорный метод и декомпозиция булевых функций с помощью булевых матриц

В [45] был предложен метод решения логических уравнений на основе булевых матриц, а в [44] этот метод был использован для решения двух следующих задач:

- для заданной БФУ f определить значение m и функции F и Φ_j в декомпозиции (3.1);
- для заданных БФУ f и F определить функции Φ_j .

Рассмотрим применение этого метода для решения первой задачи. В [44] высказано утверждение, состоящее в том, что если при фиксированном значении $n - k$ фрагменты длиной 2^{n-k} у функций f и F совпадают, то эти задачи могут быть решены на основе составления и решения матричного уравнения, состоящего из трех булевых матриц:

$$|\mathbf{G}| * |\mathbf{X}| = |\mathbf{f}|,$$

где $|\mathbf{G}|$ — матрица фрагментов ПФ модуля размерности 2^{n-k} на 2^m ; $|\mathbf{X}|$ — матрица для определения функций Φ_j размерности 2^m на 2^k ; $|\mathbf{f}|$ — матрица фрагментов реализуемой функции f размерности 2^{n-k} на 2^k ; $*$ — операция умножения булевых матриц, отличающаяся от стандартной операции умножения матриц заменой операций сложения и умножения на операции дизъюнкции и конъюнкции.

При этом в [44] было сформулировано необходимое и достаточное условие того, чтобы это уравнение имело решение в виде псевдоединичной матрицы, состоящее в том, что для каждого столбца матрицы $|\mathbf{f}|$ существует идентичный столбец матрицы $|\mathbf{G}|$.

Решение матричного уравнения определяется в виде:

$$|\mathbf{X}| = \overline{|\mathbf{G}|^T * |\mathbf{f}|},$$

где $|\mathbf{G}|^T$ — транспонированная матрица $|\mathbf{G}|$; $\overline{|\mathbf{f}|}$ — дополнение матрицы $|\mathbf{f}|$.

Для нахождения искомой декомпозиции выполняются следующие этапы.

1. Строится матрица $|\mathbf{G}|$.
2. Строится матрица $|\mathbf{G}|^T$.
3. Строится матрица $|\mathbf{f}|$.
4. Строится матрица $\overline{|\mathbf{f}|}$.

5. Строится матрица $\overline{|\mathbf{X}|} = \overline{|\mathbf{G}|^T * |\mathbf{f}|}$.

6. Строится матрица $|\mathbf{X}| = \overline{\overline{|\mathbf{X}|}} = \overline{|\mathbf{G}|^T * |\mathbf{f}|}$.

7. Так как матрица $|\mathbf{X}|$ является верхней гранью [45] множества решений матричного уравнения, то строится псевдоединичная матрица $|\mathbf{E}|$, являющаяся нижней гранью множества решений уравнения, каждый столбец которой содержит только одну единицу.

8. По матрице $|\mathbf{E}|$ строится матрица $|\mathbf{L}|$ размерности $\lceil \log m \rceil$ на 2^k

по правилу: если $e_{ij} = 1$, то в $|\mathbf{L}|$ записывается столбец $\begin{vmatrix} i \\ j \end{vmatrix}$.

9. По матрице $|\mathbf{L}|$ строится матрица $|\Phi_d|$ размерности $\lceil \log m \rceil$ на 2^k по правилу:

$$\begin{vmatrix} i \\ j \end{vmatrix} \rightarrow \begin{vmatrix} j \\ i \end{vmatrix},$$

размещая значения j в порядке их возрастания.

10. Строится искомая матрица $|\Phi|$, столбцы которой формируются в результате представления чисел i в двоичной форме.

11. Каждая строка матрицы $|\Phi|$ заменяется искомой функцией Φ_j .

12. Используя столбцы матрицы $|\Phi|$, строится функция $F = f$.

Пример 3.26. Определить при $n - k = 2$ декомпозицию рассматриваемого вида для БФУ

$$\begin{aligned} f(x_1, x_2, x_3, x_4, x_5) = \\ = |0011 \ 0000 \ 0101 \ 0000 \ 0111 \ 0011 \ 0111 \ 0011|^T \end{aligned}$$

с помощью изложенного метода.

1. Столбец значений этой функции состоит из четырех типов фрагментов длиной четыре: $G_0 = g_0 = g_5 = g_7 = |0011|^T$, $G_1 = g_1 = g_3 = |0000|^T$, $G_2 = g_2 = |0101|^T$, $G_3 = g_4 = g_6 = |0111|^T$. Рассматривая эти фрагменты в качестве столбцов матрицы $|\mathbf{G}|$, в зависимости от их расположения может быть построено $4! = 24$ матрицы, каждой из которых соответствует определенная сложность функций F и Φ_j , получаемых в дальнейшем. Так как в [44, 45] не высказано никаких соображений по порядку расположения столбцов в матрице $|\mathbf{G}|$, то сформируем эту матрицу, например, следующим образом:

$$|\mathbf{G}| = \begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{vmatrix}$$

и введем следующие обозначения: $Q_0(x_4, x_5) = G_1 = 0$,
 $Q_1(x_4, x_5) = G_2 = x_5$, $Q_2(x_4, x_5) = G_0 = x_4$,
 $Q_3(x_4, x_5) = G_3 = x_4 \vee x_5$.

2. Построим матрицу

$$|\mathbf{G}|^T = \begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{vmatrix}.$$

3. По карте декомпозиции составим матрицу

$$|\mathbf{f}| = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{vmatrix}.$$

4. Построим матрицу

$$|\overline{\mathbf{f}}| = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{vmatrix}.$$

5. Построим матрицу

$$|\overline{\mathbf{X}}| = |\mathbf{G}|^T * |\overline{\mathbf{f}}| = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{vmatrix}.$$

6. Построим матрицу

$$|\mathbf{X}| = \overline{\overline{|\mathbf{X}|}} = \overline{|\mathbf{G}|^T * |\mathbf{f}|} = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{vmatrix}.$$

7. Построим матрицу

$$|\mathbf{E}| = \begin{vmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{vmatrix}.$$

8. Построим матрицу

$$|\mathbf{L}| = \begin{vmatrix} 0 & 0 & 1 & 2 & 2 & 2 & 3 & 3 \\ 1 & 3 & 2 & 0 & 5 & 7 & 4 & 6 \end{vmatrix}.$$

9. Построим матрицу

$$|\Phi_d| = \begin{vmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 0 & 1 & 0 & 3 & 2 & 3 & 2 \end{vmatrix}.$$

10. Построим искомую матрицу

$$|\Phi| = \begin{vmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{vmatrix}.$$

11. Таким образом,

$$\begin{aligned} \Phi_1(x_1, x_2, x_3) &= |1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1|^T = x_1 \vee !x_2 !x_3; \\ \Phi_2(x_1, x_2, x_3) &= |0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0|^T = (x_1 \vee x_2) !x_3. \end{aligned}$$

12. Построим функцию

$$\begin{aligned} F(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2) &= \\ &= Q_0 !\Phi_1 !\Phi_2 \vee Q_1 !\Phi_1 \Phi_2 \vee Q_2 \Phi_1 !\Phi_2 \vee Q_3 \Phi_1 \Phi_2. \end{aligned}$$

Подставляя в эту функцию функции Q_i , получим:

$$F(\Phi_1(X_0), \Phi_2(X_0), X_1) = \Phi_1 x_4 \vee \Phi_2 x_5 = f.$$

Таким образом, применение известного метода весьма трудоемко.

Пример 3.27. Определить при $n - k = 2$ с помощью мультиплексорного метода декомпозицию рассматриваемого вида для БФУ, приведенной в предыдущем примере.

В этом случае $G_0 = g_0 = g_5 = g_7 = x_4$, $G_1 = g_1 = g_3 = 0$, $G_2 = g_2 = x_5$, $G_3 = g_4 = g_6 = x_4 \vee x_5$.

Предположим, что $Q_0 = G_1 = 0$, $Q_1 = G_2 = x_5$, $Q_2 = G_0 = x_4$, $Q_3 = G_3 = x_4 \vee x_5$.

Составим и решим уравнение

$$f(x_1, x_2, x_3, x_4, x_5) = MX(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2);$$

$$\begin{aligned} MX(Q_2, Q_0, Q_1, Q_0, Q_3, Q_2, Q_3, Q_2; x_1, x_2, x_3) = \\ = MX(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2); \end{aligned}$$

$$\Phi_1 = MX(1, 0, 0, 0, 1, 1, 1, 1; x_1, x_2, x_3) = x_1 \vee !x_2 !x_3;$$

$$\Phi_2 = MX(0, 0, 1, 0, 1, 0, 1, 0; x_1, x_2, x_3) = (x_1 \vee x_2) !x_3.$$

Так как $Q_0 \leq Q_1$, $Q_0 \leq Q_2$, $Q_0 \leq Q_3$, $Q_1 \leq Q_3$, $Q_2 \leq Q_3$, то

$$\begin{aligned} f = Q_0 \vee Q_1 \Phi_2 \vee Q_2 \Phi_1 \vee Q_3 \Phi_1 \Phi_2 = \\ = x_5 \Phi_2 \vee x_4 \Phi_1 \vee (x_4 \vee x_5) \Phi_1 \Phi_2 = \Phi_1 x_4 \vee \Phi_2 x_5. \end{aligned}$$

Таким образом, предлагаемый метод решает поставленную задачу в «две строчки».

3.4.3. Мультиплексорный метод и метод решения логических уравнений

Задача декомпозиции функции $f(x_1, \dots, x_n)$ по функции $F(\Phi_1, \dots, \Phi_m)$ может быть сведена к решению логического уравнения

$$f(x_1, \dots, x_n) = F(\Phi_1, \dots, \Phi_m)$$

относительно функций $\Phi_1(x_1, \dots, x_n), \dots, \Phi_m(x_1, \dots, x_n)$.

Наиболее распространенным и универсальным методом решения поставленной задачи является сведение рассматриваемого логического уравнения к системе, состоящей из 2^n логических уравнений, зависящих в общем случае от $m2^n$ переменных.

Трудоемкость решения этой системы снижается при учете специфики функций F и Φ_j [42].

Рассмотрим применение этого метода на примере решения трех уравнений:

$$\begin{aligned}
x_1 x_2 \vee ! x_2 x_3 &= \Phi_1(x_1, x_2, x_3) \# \Phi_2(x_1, x_2, x_3) \# \Phi_3(x_1, x_2, x_3); \\
x_1 x_2 \vee ! x_2 x_3 &= \Phi_4(x_2, x_3) \# \Phi_5(x_1, x_3) \# \Phi_6(x_1, x_2); \\
x_1 x_2 \vee ! x_2 x_3 &= \Phi_7(x_1, x_2) \# \Phi_8(x_1, x_2) \# x_3.
\end{aligned}$$

Первое уравнение соответствует мажоритарной декомпозиции, второе — простой мажоритарной декомпозиции, а третье — мажоритарному разложению по переменной x_3 [46].

Для построения систем уравнений, эквивалентных каждому из приведенных уравнений, составим табл. 3.10.

Т а б л и ц а 3.10

x_1	x_2	x_3	f	Φ_1	Φ_2	Φ_3	Φ_4	Φ_5	Φ_6	Φ_7	Φ_8
0	0	0	0	a_{01}	a_{02}	a_{03}	a_{04}	a_{05}	a_{06}	a_{07}	a_{08}
0	0	1	1	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{06}	a_{07}	a_{08}
0	1	0	0	a_{21}	a_{22}	a_{23}	a_{24}	a_{05}	a_{16}	a_{17}	a_{18}
0	1	1	0	a_{31}	a_{32}	a_{33}	a_{34}	a_{15}	a_{16}	a_{17}	a_{18}
1	0	0	0	a_{41}	a_{42}	a_{43}	a_{04}	a_{25}	a_{26}	a_{27}	a_{28}
1	0	1	1	a_{51}	a_{52}	a_{53}	a_{14}	a_{35}	a_{26}	a_{27}	a_{28}
1	1	0	1	a_{61}	a_{62}	a_{63}	a_{24}	a_{25}	a_{36}	a_{37}	a_{38}
1	1	1	1	a_{71}	a_{72}	a_{73}	a_{34}	a_{35}	a_{36}	a_{37}	a_{38}

Первому уравнению соответствует система из восьми уравнений с 24 неизвестными:

$$\begin{aligned}
a_{01} \# a_{02} \# a_{03} &= 0; & a_{41} \# a_{42} \# a_{43} &= 0; \\
a_{11} \# a_{12} \# a_{13} &= 1; & a_{51} \# a_{52} \# a_{53} &= 1; \\
a_{21} \# a_{22} \# a_{23} &= 0; & a_{61} \# a_{62} \# a_{63} &= 1; \\
a_{31} \# a_{32} \# a_{33} &= 0; & a_{71} \# a_{72} \# a_{73} &= 1.
\end{aligned}$$

Это уравнение имеет 4^8 решений, и определение среди них такого, чтобы оно удовлетворяло заданным свойствам, связано с огромным перебором.

Второму уравнению соответствует система из восьми уравнений с 16 неизвестными:

$$\begin{aligned}
a_{04} \# a_{05} \# a_{06} &= 0; & a_{04} \# a_{25} \# a_{26} &= 0; \\
a_{14} \# a_{15} \# a_{06} &= 1; & a_{14} \# a_{35} \# a_{26} &= 1; \\
a_{24} \# a_{05} \# a_{16} &= 0; & a_{24} \# a_{25} \# a_{36} &= 1; \\
a_{34} \# a_{15} \# a_{16} &= 0; & a_{34} \# a_{35} \# a_{36} &= 1.
\end{aligned}$$

Процесс нахождения решения этой системы уравнений менее трудоемок, чем для предыдущей, но также является весьма сложным [42, 46].

Третьему уравнению соответствует система из восьми уравнений с восемью неизвестными:

$$\begin{aligned}
a_{07} \# a_{08} \# 0 &= 0; & a_{27} \# a_{28} \# 0 &= 0; \\
a_{07} \# a_{08} \# 1 &= 1; & a_{27} \# a_{28} \# 1 &= 1; \\
a_{17} \# a_{18} \# 0 &= 0; & a_{37} \# a_{38} \# 0 &= 1; \\
a_{17} \# a_{18} \# 1 &= 0; & a_{37} \# a_{38} \# 1 &= 1.
\end{aligned}$$

Эта система уравнений может быть записана существенно проще:

$$\begin{aligned}
a_{07} \& a_{08} &= 0; & a_{27} \& a_{28} &= 0; \\
a_{07} \vee a_{08} &= 1; & a_{27} \vee a_{28} &= 1; \\
a_{17} \& a_{18} &= 0; & a_{37} \& a_{38} &= 1; \\
a_{17} \vee a_{18} &= 0; & a_{37} \vee a_{38} &= 1;
\end{aligned}$$

Полученная система состоит из четырех не связанных между собой систем, каждая из которых просто решается:

$$\begin{aligned}
a_{07} = 0, \quad a_{08} = 1 \quad \text{или} \quad a_{07} = 1, \quad a_{08} = 0; \\
a_{17} = 0, \quad a_{18} = 0; \\
a_{27} = 0, \quad a_{28} = 1 \quad \text{или} \quad a_{27} = 1, \quad a_{28} = 0; \\
a_{37} = 1, \quad a_{38} = 1.
\end{aligned}$$

Таким образом, получены четыре решения исходного уравнения:

$$\begin{cases} \Phi_7(x_1, x_2) = |0001|^T = x_1 x_2; \\ \Phi_8(x_1, x_2) = |1011|^T = x_1 \vee !x_2; \\ \Phi_7(x_1, x_2) = |1001|^T = !(x_1 \oplus x_2); \\ \Phi_8(x_1, x_2) = |0011|^T = x_1; \end{cases}$$

$$\begin{cases} \Phi_7(x_1, x_2) = |0011|^T = x_1; \\ \Phi_8(x_1, x_2) = |1001|^T = !(x_1 \oplus x_2); \\ \Phi_7(x_1, x_2) = |1011|^T = x_1 \vee !x_2; \\ \Phi_8(x_1, x_2) = |0001|^T = x_1 x_2. \end{cases}$$

Из изложенного следует, что использование известного метода даже в этом случае весьма трудоемко.

Первое уравнение решается мультиплексорным методом столь же сложно, как и рассмотренным методом. Второе уравнение, основанное на неразделительной декомпозиции, предлагаемым методом сводится к первому уравнению и с приемлемым объемом перебора не решается. Третье уравнение мультиплексорным методом решается весьма просто:

$$\begin{aligned} x_1 x_2 \vee !x_2 x_3 &= \Phi_7(x_1, x_2) \# \Phi_8(x_1, x_2) \# x_3; \\ \text{MX}(x_3, 0, x_3, 1; x_1, x_2) &= \text{MX}(0, x_3, x_3, 1; \Phi_7, \Phi_8); \\ \begin{cases} \Phi_7 = \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2; \\ \Phi_8 = \text{MX}(1, 0, 1, 1; x_1, x_2) = x_1 \vee !x_2; \end{cases} \\ \begin{cases} \Phi_7 = \text{MX}(1, 0, 0, 1; x_1, x_2) = !(x_1 \oplus x_2); \\ \Phi_8 = \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1; \end{cases} \\ \begin{cases} \Phi_7 = \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1; \\ \Phi_8 = \text{MX}(1, 0, 0, 1; x_1, x_2) = !(x_1 \oplus x_2); \end{cases} \\ \begin{cases} \Phi_7 = \text{MX}(1, 0, 1, 1; x_1, x_2) = x_1 \vee !x_2; \\ \Phi_8 = \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2. \end{cases} \end{aligned}$$

В заключение главы отметим, что предложенный метод позволяет на каждом шаге выполнять декомпозицию при «совпадении» свойств столбцов значений реализуемой булевой функции и порождающей функции используемого элемента. Общая задача разложения булевой функции описана в [52, 53].

Выводы

1. Введены понятия «мультиплексорная декомпозиция» и «простая мультиплексорная декомпозиция». Эти виды декомпозиции

выполняются по образу мультиплексорной функции и являются разделительными.

2. Предложена стандартная схема, состоящая из мультиплексора и двух ПЗУ, реализующая мультиплексорную декомпозицию.

3. Предложен метод построения мультиплексорных декомпозиций на основе решения логических уравнений.

4. Предложены параметрическая и явная мультиплексорные формы для решения логических уравнений, позволяющие решать их существенно проще, чем известными методами. По уравнению, записанному в одной из таких форм, легко определяется число решений, и для каждой из этих форм существует конструктивный и наглядный подход к построению каждого из решений.

5. Показано, что построение карт декомпозиции позволяет весьма просто (в отличие от таблиц истинности) находить второе множество входных переменных с целью построения более простой мультиплексорной декомпозиции.

6. Предложен метод разложения произвольной БФУ, заданной таблицей истинности, по крайним правым входным переменным, в то время как традиционно при таком задании БФУ раскладывается по крайним левым переменным.

7. Рассмотрены все возможные 24 разложения произвольной БФУ на две остаточные функции по крайней правой входной переменной и показано, что все разложения по этой переменной либо совпадают с разложениями Шеннона и Рида, либо PN -однотипны с ними и других универсальных разложений на две остаточные функции не существует.

8. Показано, что при разложении произвольной БФУ по двум и более крайним правым входным переменным существует такое назначение функций на входах мультиплексора, что предложенная в п. 2 стандартная схема существенно упрощается за счет уменьшения размерности мультиплексора и исключения одного из двух ПЗУ. Предложен распределительный метод реализации БФУ, осуществляющий такие разложения.

9. Предложен мультиплексорный метод реализации БФУ на произвольных, априори заданных логических элементах, основанный на исследовании их функциональных возможностей и составлении и решении в мультиплексорной форме для каждого шага декомпозиции логического уравнения. Метод обеспечивает построение схем от выхода к входам (сверху вниз). Для БФУ, зависящих от числа переменных $n \geq 6$, и элементов с числом входов $m \leq 6$ предлагается использовать этот метод совместно с другими методами, например с формульным, для реализации подфункций меньшего числа переменных.

10. Приведены примеры схем, построенных мультиплексорным методом, которые реализуют БФ из пяти букв, неповторные в базе И, ИЛИ, НЕ с числом 3-универсальных модулей меньшим, чем

их число в схемах, построенных формульным методом. Это позволило для произвольных БФ в этом базисе из указанного числа букв снизить оценку числа таких модулей в схемах до двух.

11. Если при применении формульного метода выполняется синтез на настроенных модулях, что сокращает функциональные возможности модулей, то при использовании более трудоемкого мультиплексорного метода настройка каждого из них определяется на соответствующем шаге процесса синтеза. Это приводит к тому, что мультиплексорный метод строит схемы, сложность которых не превышает сложности схем, которые строятся формульным методом, обеспечивающим оценки сложности, линейные от числа букв в реализуемой формуле, записанной в базисе $\{\&, \vee, !\}$ или $\{\&, \vee, \oplus, !\}$.

Л и т е р а т у р а

1. Бибило П. Н. Декомпозиция булевых функций // Проектирование устройств логического управления. М.: Наука, 1984.
2. Shannon C. A symbolic analysis of relay and switching circuits // Trans. of American Inst. of Electrical engineers. 1938. N 57.
3. Поваров Г. Н. О функциональной разделимости булевых функций // Докл. АН СССР. 1954. № 5.
4. Reed I. S. Class of multiple-error correcting codes and decoding scheme // IRE Trans. Inform. Theory. 1954. IT-4.
5. Ashenurst R. I. The decomposition of switching functions // Annals of Computation Lab. of Harvard University. Mass. Cambridge. 1959. Vol. 29.
6. Curtis H. A. New approach to the design of switching circuits. N. J.: D. Van Nostrand Co, 1962.
7. Karp R. M. Functional decomposition and switching circuits design // J. Soc. Indust. Math. 1963. N 2.
8. Карп Р. М. Единый подход к функциональной декомпозиции // Синтез релейных структур. М.: Наука, 1965.
9. Поваров Г. Н. Математическая теория синтеза контактных $(1, k)$ -полюсников // Докл. АН СССР. 1955. № 5.
10. Блох А. Ш. Канонический метод синтеза контактных схем // Автоматика и телемеханика. 1961. № 6.
11. Блох А. Ш. Граф-схемы и их применение. Минск: Вышэйшая школа, 1975.
12. Блох А. Ш., Павловский А. И. Синтез логических схем на трехвходовом операторе // Докл. АН БССР. 1979. № 7.
13. Блох А. Ш., Павловский А. И. Логические сети из трехвходовых операторов // Автоматика и вычисл. техника. 1980. № 4.
14. Patt Y. N. A complex logic module for the synthesis of combinational switching circuits // Proc. 1967 AFIPS. Spring Joint Comput. Conf.
15. Bertrand J. C. Application des matrices booléennes a la synthèse modulaire des fonctions logiques // These de Decteur de 3-е Cycle. Toulouse, 1970.
16. Закурдаев Н. В. О некоторых правилах предпочтения в декомпозиционных методах синтеза комбинационных схем // Автоматика и телемеханика. 1972. № 10.
17. Варшавский В. И. Некоторые вопросы теории логических сетей, построенных из пороговых элементов // Вопросы теории математических машин. М.: Физматгиз, 1962.
18. Дертоузос М. Пороговая логика. М.: Мир, 1967.
19. Вавилов Е. Н., Егоров Б. М., Ланцев В. С., Тоценко В. Г. Синтез схем на пороговых элементах. М.: Сов. радио, 1970.

20. Бутаков Е. А. Методы синтеза релейных устройств из пороговых элементов. М.: Энергия, 1970.
21. Боголюбов И. Н., Овсевич Б. Л., Розенблум Л. Я. Синтез схем из пороговых и мажоритарных элементов // Сети передачи информации и их автоматизация. М.: Наука, 1965.
22. Горбатов В. А. Синтез логических схем в произвольном базисе // Теория дискретных автоматов. Рига: Зинатне, 1967.
23. Горбатов В. А. Схемы управления ЦВМ и графы. М.: Энергия, 1971.
24. Горбатов В. А. Основы дискретной математики. М.: Высш. школа, 1986.
25. Бивол Л. Г. О реализации булевых функций на произвольных элементах // Абстрактная и структурная теория релейных устройств. М.: Наука, 1972.
26. Пархоменко П. П. Синтез структур релейных устройств методом замены входных переменных // Автоматика и телемеханика. 1967. № 1.
27. Горовой В. Р. Синтез релейных структур методом замены выходных функций // Автоматика и телемеханика. 1967. № 1.
28. Гаврилов М. А., Копыленко В. М. Метод «переходных таблиц» синтеза многовыходных комбинационных структур на произвольных элементах. М.: Ин-т проблем управления, 1970.
29. Гаврилов М. А. Декомпозиция комбинационных автоматов // Математические структуры. Вычислительная математика. Математическое моделирование. София, 1975.
30. Гаврилов М. А. Композиция и декомпозиция комбинационных автоматов // Теория автоматов. М.: Наука, 1976.
31. Гаврилов М. А., Девятков В. В., Пупырев Е. И. Логическое проектирование дискретных автоматов. М.: Наука, 1977.
32. Миллер Р. Теория переключательных схем. Т. 1. Комбинационные схемы. М.: Наука, 1970.
33. Берлин А. Н. Синтез логических устройств в коммутационной технике. М.: Радио и связь, 1982.
34. Rudeanu S. Boolean functions and equations. Amsterdam; London: North-Holland Publ. Co., 1974.
35. Рудяну С. Локальные свойства булевых функций и экстремальные решения булевых уравнений // Теория автоматов. М.: Наука, 1976.
36. Lowenheim L. Über die Auflösung von Gleichungen in Logischen gebietkalkul // Math. Annal. 1910. Bd 68.
37. Закревский А. Д. Логические уравнения. Минск: Наука и техника, 1975.
38. Сериков Ю. А., Лившиц А. Н. Логические уравнения // Тез. докл. II Всесоюз. совещ. по теории релейных устройств и конечных автоматов. Рига, 1971.
39. Боголюбов И. Н., Овсевич Б. Л. Некоторые способы решения систем логических уравнений в произвольных базисах // Тез. докл. II Всесоюз. совещ. по теории релейных устройств и конечных автоматов. Рига, 1971.
40. Розенблум Л. Я. Некоторые вопросы синтеза логических схем из мажоритарных элементов. Дис. на соиск. ученой степени канд. техн. наук. Л.: ВЦ ЛО Мат. ин-та им. В. А.Стеклова АН СССР, 1965.
41. Юдин Д. Б., Гольштейн Е. Г. Линейное программирование. М.: Физматгиз, 1963.
42. Варшавский В. И. Мажоритарная декомпозиция // Автоматика и телемеханика. 1965. № 1.
43. Чистов В. П. О системах логических уравнений. М.: ВИНТИ. Деп. рук. № 2114-85.
44. Пандеф Е. Булевы матрицы // Булева алгебра и конечные автоматы. М.: Мир, 1969.
45. Ледли Р. Программирование и использование вычислительных машин. М.: Мир, 1966.
46. Поспелов Д. А. Логические методы анализа и синтеза схем. М.: Энергия, 1974.
47. Фридман А., Менон П. Теория и проектирование переключательных схем. М.: Мир, 1978.

48. *Артюхов В. Л., Копейкин Г. А., Шалыто А. А.* Настраиваемые модули для управляющих логических устройств. Л.: Энергоиздат, 1981.
49. *Li H. F.* Variable selection in logic synthesis using multiplexers // *Int. J. Electronics*. 1980. N 3.
50. *Артюхов В. Л., Копейкин Г. А., Шалыто А. А.* Судовые управляющие логические системы. Унифицированные логические схемы. Л.: ИПК СП, 1981.
51. *Угрюмов Е. П., Грушвицкий Р. И., Альшевский А. Н.* БИС/СБИС с репрограммируемой структурой. СПб.: Гос. электротех. ун-т, 1996.
52. *Лунанов О. Б.* Об одном методе синтеза схем // *Изв. вузов. Радиофизика*. 1958. № 1.
53. *Кобринский Н. Е., Трахтенброт Б. А.* Введение в теорию конечных автоматов. М.: Физматлит, 1962.
54. *Брусенцов Н. П., Владимирова Ю. С.* Решение булевых уравнений // *Методы математического моделирования. Труды ф-та ВМиК МГУ*. М.: Диалог—МГУ, 1998.
55. *Левченко В. С.* Общий вид решения булевых уравнений // *Автоматика и телемеханика*. 2000. № 2.

Глава 4

Реализация булевых функций схемами из мажоритарных элементов

Вопросы синтеза схем в этом базисе выделены в отдельную главу, так как практически никаким другим типам логических элементов не было посвящено столько теоретических исследований [1—19].

Мажоритарным элементом (МЭ) n переменных называется пороговый элемент с единичными весами и порогом $t = (n + 1)/2$, где $n \geq 3$ и нечетно.

В литературе и в настоящей работе в основном используются МЭ с $n = 3$. Рассматриваются также МЭ с $n = 5$.

4.1. Методы реализации булевых функций схемами из трехвходовых мажоритарных элементов

4.1.1. Основные соотношения

Трехвходовой мажоритарный элемент описывается БФУ $F(a, b, c) = |0001\ 0111|^T$. Эта БФУ является симметрической и может быть записана в виде:

$$F = S_{2,3}(a, b, c).$$

Являясь пороговой [20], эта БФУ может быть записана в виде:

$$F = \text{sign}(a + b + c - 2).$$

Ее арифметический полином [21] имеет вид:

$$F = ab + ac + bc - 2abc.$$

При использовании арифметического полинома с маскированием (разд. 20.1.3) эта БФУ может быть записана в виде:

$$F = r_2^2 (\text{bin} (a + b + c)).$$

Рассматриваемая БФУ может быть также записана в виде (разд. 20.1.3):

$$F = \text{sign} ((a + b) c + ab).$$

Наиболее употребительным является представление этой БФУ в виде БФ:

$$F = (a \vee b) c \vee ab.$$

Можно показать, что

$$F = (a \oplus b) c \vee ab.$$

Применяя разложение Рида, получим БФ:

$$F = (a \oplus b) c \oplus ab.$$

Так как эта БФУ является самодвойственной (гл. 10), то

$$F = (a \oplus b)(a \oplus c) \oplus a.$$

Будем обозначать рассматриваемую БФУ символом

$$F = a \# b \# c.$$

4.1.2. Обзор литературы

Выполним краткий обзор методов, известных из литературы.

Методы синтеза схем из трехходовых МЭ базируются на:

- аксиомах мажоритарной логики [1, 2];
- преобразованиях Кона—Линдемана [2], обеспечивающих за счет отождествления или антиотождествления переменных x_i и x_j (разложение сверткой переменных [15]) исключение одной переменной в остаточных функциях. Среди этих преобразований имеются, например, следующие:

$$f = (x_i \# x_j \# f(x_i = !x_j)) \# (x_i \# !x_j \# f(x_i = x_j)) \# !x_i;$$

$$f = (!x_i \# !x_j \# f(x_i = !x_j)) \# (!x_i \# x_j \# f(x_i = x_j)) \# x_i;$$

— разложении Шеннона в мажоритарном базисе (разложение фиксации переменных) [7]:

$$f = (f(x_i=0) \# !x_i \# 0) \# (f(x_i=1) \# x_i \# 0) \# 1;$$

— монотонности БФУ по переменной x_i или ее инверсии. При этом справедливы соотношения, первые три из которых получены в [4] и называются разложениями Акерса, а остальные — в [17], которые назовем разложениями Розенблюма:

$$f = \Phi_1 \# \Phi_2 \# x_i;$$

$$f = f(x_i=0) \# f(x_i=1) \# x_i;$$

$$f = f(x_i=0) \# f(x_i=1) \# !x_i;$$

$$f = f(x_i=x_j) \# f(x_i=!x_j) \# x_i;$$

$$f = f(x_i=x_j) \# f(x_i=!x_j) \# !x_i;$$

- самодвойственности и монотонности БФУ [5];
- пороговости и монотонности [6, 15];
- «повышении» монотонности БФУ за счет «временного» увеличения числа переменных [4];
- геометрических свойствах БФУ [5];
- функциональной разделимости БФУ [12];
- решении логических уравнений [14].

4.1.3. Формульный метод

Так как трехвходовой МЭ (в дальнейшем, если не оговаривается особо, МЭ) при $c = 0$ реализует формулу $F = a \vee b$, а при $c = 1$ — формулу $F = ab$, то этот элемент при равной доступности прямых и инверсных входных переменных универсален в классе БФ в базисе $\{\&, \vee, !\}$ из двух букв. Это обеспечивает возможность реализации произвольной БФ в указанном базисе из h букв древовидной схемой из $h - 1$ МЭ. Для неповторной БФ (ББФ) получающаяся схема является неповторной (по пометкам входов различных элементов). Повторные пороговые формулы (БПФ) могут быть реализованы особым видом древовидных схем — неповторными каскадными схемами без разветвлений (повторные каскадные схемы).

Пример 4.1. Реализовать непороговую ББФ

$$f = (x_1 !x_2 \vee x_3 x_4) x_5.$$

Реализуем эту формулу древовидной схемой из двухвходовых элементов И и ИЛИ (рис. 4.1) и заменим каждый из этих элементов одним МЭ с соответствующей настройкой (рис. 4.2). Полученная

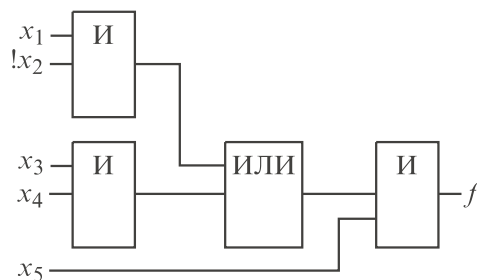


Рис. 4.1

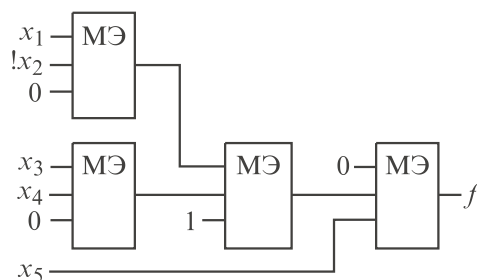


Рис. 4.2

схема является «естественной» для реализации ББФ, так как является древовидной, неповторной, монотонной (не содержит одновременно $!x_i$ и x_i) и состоит из $h - 1 = 4$ МЭ.

4.1.4. Графовый метод

Этот метод, описанный в [25], включает следующие этапы.

1. Заданная БФ в базисе $\{\&, \vee, !\}$ из h букв записывается в порядке, обеспечивающем минимальное число путей в линейном бинарном графе (ЛБГ) [26].

2. По преобразованной БФ строится планарный ЛБГ [26], состоящий из h условных и двух операторных вершин.

3. Каждая условная вершина графа, который рассматривается от выходов к входу, заменяется на МЭ, два входа которого подключаются как в графе, а на третий вход подается входная переменная, указанная в рассматриваемой условной вершине. В результате строится каскадная схема с минимальным числом разветвлений, состоящая из h МЭ.

4. Последний МЭ в схеме исключается, а соответствующая входная переменная подается на вход предпоследнего МЭ. Это обеспечивает построение каскадной схемы с минимальным числом разветвлений, состоящей из $h - 1$ МЭ. Имеющиеся в схеме длинные связи входов МЭ с константами и одной из входных переменных заменяются на короткие связи.

5. Верификация. Особенность этой схемы состоит в том, что на выходе каждого МЭ реализуется не подформула, как это имеет место в формульном методе, а фрагмент формулы, что впервые было установлено в [27].

Отметим, что для заданной БФУ в общем случае каскадная схема с разветвлениями не является единственной, так как при изменении порядка записи БФ число разветвлений в схеме может измениться или разветвления «переместятся» без изменения числа МЭ в схеме.

Графовый метод для произвольных ББФ строит неповторные каскадные схемы с разветвлениями, а для произвольных БПФ — неповторные каскадные схемы. Эти схемы являются монотонными. При этом новым является то, что, во-первых, для произвольной ББФ число МЭ в каскадной недревовидной схеме с разветвлениями и в древовидной схеме, строящейся формульным методом, совпадает, а, во-вторых, формуле, неповторной в базисе $\{\&, \vee, !\}$, соответствует схема с разветвлениями.

Пример 4.2. Реализовать каскадной схемой с разветвлениями из четырех МЭ непороговую неповторную формулу $f = (x_1!x_2 \vee \vee x_3x_4)x_5$.

1. Минимальное число путей в ЛБГ обеспечивается при записи БФ в виде $f = x_5(x_1!x_2 \vee x_3x_4)$.

2. Построим по этой формуле ЛБГ (рис. 4.3).

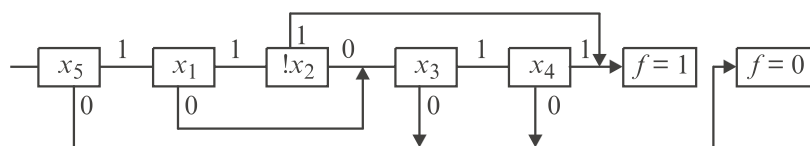


Рис. 4.3

3. Построим по ЛБГ каскадную схему с разветвлениями, состоящую из пяти МЭ (рис. 4.4).

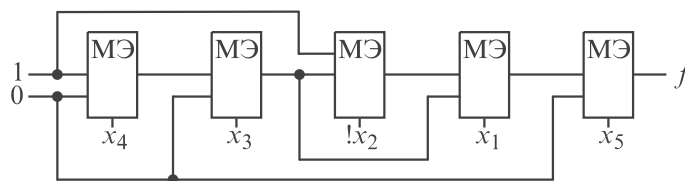


Рис. 4.4

4. Построим каскадную схему с разветвлениями, состоящую из четырех МЭ (рис. 4.5).

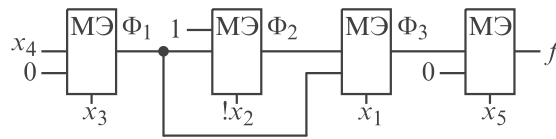


Рис. 4.5

5. Выполним верификацию построенной схемы:

$$\Phi_1 = x_3 \# x_4 \# 0 = x_3 x_4;$$

$$\Phi_2 = !x_2 \# \Phi_1 \# 1 = !x_2 \vee x_3 x_4;$$

$$\Phi_3 = x_1 \# \Phi_1 \# \Phi_2 = x_1 !x_2 \vee x_3 x_4;$$

$$f = x_5 \# 0 \# \Phi_3 = x_5 (x_1 !x_2 \vee x_3 x_4).$$

Если ЛБГ строить непосредственно по заданной ББФ, то схема изменится (рис. 4.6) без потери каскадности и изменения числа элементов в ней.

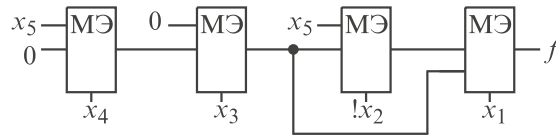


Рис. 4.6

В заключение раздела отметим, что изложенный метод базируется на соотношениях

$$MX(\Phi_0, \Phi_1; x_i) = \Phi_0 \# \Phi_1 \# x_i;$$

$$MX(\Phi_0, \Phi_1; x_i) = \Phi_0 \# \Phi_1 \# !x_i,$$

справедливых для функций, монотонных по переменным x_i и $!x_i$ соответственно [27].

4.1.5. Мультиплексорный метод

В данном случае этот метод включает в себя три составляющие:

- мажоритарные разложения;
- мультиплексорную декомпозицию, компоненты которой реализуются на МЭ;
- вспомогательные приемы, которые упрощают реализацию БФУ и дают возможность, в частности, проводить мажоритарные разложения.

Мультиплексорный метод позволяет с единых позиций проводить мажоритарные разложения и мультиплексорную декомпозицию, чередуя их при необходимости, в то время как в известных ра-

ботах такие разложения и функциональная декомпозиция проводятся различными методами.

Мажоритарные декомпозиции и мажоритарные разложения булевых функций. Мажоритарной декомпозицией [14] БФУ n переменных называется ее представление в виде

$$f(X) = \Phi_1(X) \# \Phi_2(X) \# \Phi_3(X).$$

Такая декомпозиция имеет смысл в тех случаях, когда функции Φ_j по какому-либо критерию проще, чем исходная функция f . При этом в качестве критерия может выступать не только число переменных, от которых эти функции зависят, но и другие характеристики. Например, при $n = 3$ такая декомпозиция бывает целесообразной, если по крайней мере одна функция Φ_j является мажорантой, существенно зависящей от всех переменных. Так как МЭ описывается БФУ, симметрической по всем переменным, то совместим исследование его функциональных возможностей с построением логических уравнений для проведения декомпозиций при различных значениях $n - k$.

1. При $n - k = 0$

$$\begin{aligned} f(x_1, \dots, x_n) &= \Phi_1 \# \Phi_2 \# \Phi_3; \\ f(x_1, \dots, x_n) &= \text{MX}(0, 0, 0, 1, 0, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3). \end{aligned}$$

Если и функцию f записать в МФ, то нахождение решения этого уравнения, обладающего заданными свойствами, связано с огромным перебором, так как число решений в этом случае равно $4^{t_0} \cdot 4^{t_1} = 4^{t_0+t_1} = 4^{2^n}$, где $t_0(t_1)$ — число нулей (единиц) в столбце значений функции f . Среди этих решений находятся и все те, которые могут быть получены любым из известных методов. Так как конструктивный метод сокращения перебора в рамках предлагаемого подхода автору не известен, то в дальнейшем такая декомпозиция не используется.

2. При $n - k = 1$, если столбец значений функции f состоит из фрагментов длиной два, входящих в множество $\{0, x_n, 1\}$, можно записать уравнение:

$$\begin{aligned} f(x_1, \dots, x_n) &= \Phi_1 \# \Phi_2 \# x_n; \\ f(x_1, \dots, x_n) &= \text{MX}(0, x_n, x_n, 1; \Phi_1, \Phi_2). \end{aligned}$$

Если и функцию f записать в МФ, то это уравнение имеет 2^{t_2} решений, где t_2 — число фрагментов x_n в столбце значений функции f .

Декомпозиция этого типа может быть названа мажоритарным разложением функции f по переменной x_n .

При этом разложения Акерса и Розенблюма являются частными случаями этого разложения. Это разложение впервые было рассмотрено в [3], однако метод его построения существенно более трудоемок по сравнению с предлагаемым решением уравнений в МФ.

3. При $n - k = 2$, если столбец значений функции f состоит из фрагментов длиной четыре двух типов — $x_{n-1}x_n$ и $x_{n-1} \vee x_n$, можно записать уравнение:

$$\begin{aligned} f(x_1, \dots, x_n) &= \Phi_1 \# x_{n-1} \# x_n; \\ f(x_1, \dots, x_n) &= \text{MX}(x_{n-1}x_n, x_{n-1} \vee x_n; \Phi). \end{aligned}$$

Если и функцию f записать в МФ, то это уравнение имеет единственное решение.

Декомпозиция этого типа может быть названа мажоритарным разложением функции f одновременно по двум переменным x_{n-1} и x_n .

4. Если столбец значений функции f состоит из следующих типов фрагментов: $\{0, Q(X_1), 1\}$, или $\{0, Q(X_1)\}$, или $\{Q(X_1), 1\}$, то можно записать уравнение:

$$\begin{aligned} f(x_1, \dots, x_n) &= \Phi_1 \# \Phi_2 \# Q(X_1); \\ f(x_1, \dots, x_n) &= \text{MX}(0, Q(X_1), Q(X_1), 1; \Phi_1, \Phi_2). \end{aligned}$$

Если и функцию f записать в МФ, то это уравнение имеет 2^{t_3} решений, где t_3 — число фрагментов $Q(X_1)$ в столбце значений функции f .

Декомпозиция этого типа может быть названа мажоритарным разложением функции f по подфункции $Q(X_1)$.

5. При наличии как прямых, так и инверсных входных переменных класс реализуемых функций расширяется.

При $n - k = 1$, если функция f состоит из фрагментов длиной два, входящих в множество $\{0, !x_n, 1\}$, можно записать уравнение:

$$\begin{aligned} f(x_1, \dots, x_n) &= \Phi_1 \# \Phi_2 \# !x_n; \\ f(x_1, \dots, x_n) &= \text{MX}(0, !x_n, !x_n, 1; \Phi_1, \Phi_2). \end{aligned}$$

Если и функцию f записать в МФ, то это уравнение имеет 2^{t_4} , где t_4 — число фрагментов $!x_n$ в столбце значений функции f .

Декомпозиция этого типа может быть названа мажоритарным разложением функции f по переменной $!x_n$.

При $n - k = 2$ для существования мажоритарного разложения должно выполняться соотношение

$$Q_0! \Phi \vee Q_1 \Phi = \Phi \# \tilde{x}_{n-1} \# \tilde{x}_n.$$

Пример 4.3. При $n - k = 2$ мажоритарно разложить БФУ

$$f(x_1, x_2, x_3, x_4) = |0100 \ 1101 \ 1101 \ 0100|^T.$$

В этом случае $Q_0 = !x_3 x_4$, $Q_1 = !x_3 \vee x_4$. Так как

$$!x_3 x_4 !\Phi \vee (!x_3 \vee x_4) \Phi = \Phi \# !x_3 \# x_4,$$

то составим и решим уравнение:

$$f(x_1, x_2, x_3, x_4) = \Phi \# !x_3 \# x_4;$$

$$\text{MX}(Q_0, Q_1, Q_1, Q_0; x_1, x_2) = \text{MX}(Q_0, Q_1; \Phi);$$

$$\Phi = \text{MX}(0, 1, 1, 0; x_1, x_2) = x_1 \oplus x_2.$$

Пример 4.4. Функция $f(x_1, x_2, x_3, x_4) = |0010 \ 1101 \ 1101 \ 0010|^T$ не может быть мажоритарно разложена при $n - k = 2$, так как при $Q_0 = x_3 !x_4$, $Q_1 = !x_3 \vee x_4$ справедливо соотношение

$$x_3 !x_4 !\Phi \vee (!x_3 \vee x_4) \Phi \neq \Phi \# \tilde{x}_3 \# \tilde{x}_4.$$

Метод реализации булевых функций на основе мультиплексорной декомпозиции. Этот метод состоит в следующем.

1. Зафиксируем значение $n - k$.
2. Если $n - k = k$, то строится карта декомпозиции для определения двух множеств разнотипных фрагментов, зависящих от $n - k$ первых и последних переменных.
3. Если $n - k < k$, то множество разнотипных фрагментов, зависящих от последних $n - k$ переменных, определяется по столбцу значений ТИ. Множество разнотипных фрагментов, зависящих от первых $n - k$ переменных, определяется по карте декомпозиции.
4. При декомпозиции по первым $n - k$ переменным таблица истинности должна быть перестроена.
5. Дальнейшие этапы декомпозиции выполняются для каждого из выделенных множеств.
6. Пусть число фрагментов равно R . Выберем $m = \lceil \log R \rceil$.
7. Запишем логическое уравнение

$$f = \text{MX}(Q_0(X_1), \dots, Q_{2^m-1}(X_1); \Phi_1(X_0), \dots, \Phi_m(X_0)).$$

8. Выполним назначение функций Q_i . Если это возможно, оно должно производиться таким образом, чтобы выполнялись отношения покрытия.

9. Исходное уравнение записывается в МФ и решается мультиплексорным методом относительно переменных Φ_j .

10. После этого могут быть записаны три типа соотношений:

$$f = F_1(Q_0(X_1), \dots, Q_{2^m-1}(X_1), \Phi_1(X_0), \dots, \Phi_m(X_0));$$

$$f = F_2(\Phi_1(X_0), \dots, \Phi_m(X_0), X_1);$$

$$f = F_3(Q_0(X_1), \dots, Q_{2^m-1}(X_1), X_0).$$

В первом случае в базисе МЭ должны быть реализованы функции F_1, Q_i, Φ_j , во втором — функции F_2 и Φ_j , а в третьем — функции F_3 и Q_i . При реализации этих функций должны в отдельности учитываться отношения покрытия для функций Q_i и Φ_j .

11. Выбирается простейшая реализация.

Пример 4.5. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0000 \ 1111 \ 0111 \ 0001|^T.$$

1. Предположим, что $n - k = 2$.

2. Так как в данном случае $n - k = k$, то построим карту декомпозиции (табл. 4.1).

Т а б л и ц а 4.1

x_1x_2	x_3x_4			
	00	01	10	11
00	0	0	0	0
01	1	1	1	1
10	0	1	1	1
11	0	0	0	1

Таким образом, заданная БФУ содержит четыре типа фрагментов, зависящих от переменных x_3, x_4 : $g_0=0, g_1=1, g_2=x_3 \vee x_4, g_3=x_3x_4$, и три типа фрагментов, зависящих от переменных x_1, x_2 : $g_0=\neg x_1x_2, g_1=x_1 \oplus x_2, g_2=x_1 \vee x_2$. При этом

$$f(x_3, x_4, x_1, x_2) = |0100 \ 0110 \ 0110 \ 0111|^T.$$

3. Используем фрагменты первого типа.

3.1. Назначим функции Q_i следующим образом: $Q_0=0, Q_1=x_3x_4, Q_2=x_3 \vee x_4, Q_3=1$.

3.2. Так как $Q_0 \leq Q_1 \leq Q_2 \leq Q_3$, то $f = Q_0 \vee Q_1 \Phi_2 \vee Q_2 \Phi_1 \vee Q_3 \Phi_1 \Phi_2$.

3.3. Запишем и решим уравнение:

$$f(x_1, x_2, x_3, x_4) = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2);$$

$$\text{MX}(Q_0, Q_3, Q_2, Q_1; \Phi_1, \Phi_2) = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2);$$

$$\Phi_1 = \text{MX}(0, 1, 1, 0; x_1, x_2) = x_1 \oplus x_2;$$

$$\Phi_2 = \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2.$$

3.4.1. Подставим функции Q_0 и Q_3 в упрощенное выражение для f . При этом $Q_1=x_3x_4, Q_2=x_3 \vee x_4, \Phi_1=x_1 \oplus x_2, \Phi_2=x_2$,

$f = \Phi_1 Q_2 \vee (\Phi_1 \vee Q_1) \Phi_2$. Эта декомпозиция может быть реализована схемой из $1 + 1 + 3 + 0 + 4 = 9$ МЭ.

3.4.2. Подставим все функции Q_i в упрощенное выражение для f . При этом $\Phi_1 = x_1 \oplus x_2$, $\Phi_2 = x_2$, $f = \Phi_1 (\Phi_2 \vee x_3 \vee x_4) \vee \Phi_2 x_3 x_4$. Эта декомпозиция может быть реализована схемой из $3 + 0 + 2 = 5$ МЭ.

3.4.3. Подставим все функции Φ_j в упрощенное выражение для f . При этом $Q_1 = x_3 x_4$, $Q_2 = x_3 \vee x_4$, $f = (!x_1 \vee Q_1) x_2 \vee Q_2 x_1 !x_2$. Эта декомпозиция может быть реализована схемой из $1 + 1 + 5 = 7$ МЭ.

4. Используем фрагменты второго типа.

4.1. Назначим функции Q_i следующим образом: $Q_0 = Q_1 = !x_1 x_2$, $Q_2 = x_1 \oplus x_2$, $Q_3 = x_1 \vee x_2$.

4.2. Так как $Q_0 = Q_1 \leq Q_2 \leq Q_3$, то $f = Q_0 \vee Q_2 \Phi_1 \vee Q_3 \Phi_1 \Phi_2$.

4.3. Запишем уравнение

$$f(x_3, x_4, x_1, x_2) = MX(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2);$$

$$MX(Q_0, Q_2, Q_2, Q_2; \Phi_1, \Phi_2) = MX(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2).$$

Это уравнение имеет два решения, из которых выберем следующее:

$$\Phi_1 = MX(0, 1, 1, 1; x_3, x_4) = x_3 \vee x_4;$$

$$\Phi_2 = MX(0, 0, 0, 1; x_3, x_4) = x_3 x_4.$$

4.4.1. При совместной реализации функций Q_0 и Q_2 получим: $Q_0 = !x_1 x_2$, $Q_2 = Q_1 \vee x_1 !x_2$, $Q_3 = x_1 \vee x_2$, $\Phi_1 = x_3 \vee x_4$, $\Phi_2 = x_3 x_4$, $f = Q_0 \vee \Phi_1 (Q_2 \vee Q_3 \Phi_2)$. Эта декомпозиция может быть реализована схемой из $1 + 2 + 1 + 1 + 1 + 4 = 10$ МЭ.

4.4.2. Подставим все функции Q_i в упрощенное выражение для f . При этом $\Phi_1 = x_3 \vee x_4$, $\Phi_2 = x_3 x_4$, $f = (!x_1 \vee \Phi_1 \Phi_2) x_2 \vee x_1 !x_2 \Phi_1$. Эта декомпозиция может быть реализована схемой из $1 + 1 + 6 = 8$ МЭ.

4.4.3. Подставим все функции Φ_j в упрощенное выражение для f . При этом $Q_0 = !x_1 x_2$, $Q_2 = x_1 \oplus x_2$, $Q_3 = x_1 \vee x_2$, $f = !x_1 x_2 \vee \vee x_1 \Phi_1 (\Phi_2 \vee !x_2)$. Эта декомпозиция может быть реализована схемой из $1 + 3 + 1 + 5 = 10$ МЭ.

Из изложенного следует, что сокращение числа типов фрагментов в заданной функции не привело к упрощению схемы.

5. Таким образом, простейшая схема, построенная на основе мультиплексорной декомпозиции, содержит пять МЭ.

6. Так как заданная БФУ реализуется БФ $f = !x_1 x_2 \vee x_1 x_3 x_4 \vee \vee x_1 !x_2 x_3 \vee x_1 !x_2 x_4$, то она может быть записана как $f = !x_1 x_2 \vee \vee x_1 (!x_2 \# x_3 \# x_4)$. Для этой формулы выполним эвристическую декомпозицию, которая реализуется схемой из четырех МЭ. Таким образом, для функции f существует более простая схема по сравнению со схемой, которую строит предлагаемый метод.

Вспомогательные приемы для упрощения реализации булевых функций. Перечислим основные из этих приемов.

1. Построение БФ в базисе $\{\&, \vee, !\}$ с минимизированным числом букв. Это позволяет:

- обнаружить, что БФУ реализуется неповторной формулой;
- обнаружить мажоранты (если они имеются) непосредственно по формуле;
- обнаружить (если они имеются) переменные, от которых БФУ не зависит;
- обнаружить (если они имеются) переменные, по которым БФУ монотонна;
- использовать формульный и графовый методы;
- провести эвристическую декомпозицию для снижения размерности решаемой задачи.

2. Построение карты декомпозиции для простого определения:

- при $n - k = k$ двух множеств разнотипных фрагментов, зависящих от $n - k$ первых или последних переменных;
- при $n - k < k$ множества разнотипных фрагментов, зависящих от $n - k$ первых переменных. В этом случае множество разнотипных фрагментов, зависящих от последних переменных, определяется непосредственно по ТИ.

3. Без построения БФ определить переменную, по которой БФУ монотонна, или изменить номенклатуру фрагментов в ТИ можно за счет перестановки столбцов входных переменных в ней. Эту перестановку целесообразно проводить с помощью метода Блоха [28], изложенного в разд. 5.1.1.

4. Назначение (по возможности) фрагментов Q_i таким образом, чтобы для них выполнялись отношения покрытия. Это позволяет строить образ декомпозиции без инверсий функций Φ_j и иметь возможность реализовывать функции Q_i не только раздельно, но и совместно.

5. Повышение монотонности [4] за счет «временного» увеличения числа переменных в реализуемой функции f . Если БФУ немонотонна по всем переменным, то ее монотонность может быть повышена, если «временно» все вхождения, например переменной $!x_n$ в БФ в базисе $\{\&, \vee, !\}$, заменить переменной x_{n+1} . Это формирует новую функцию f_1 . При этом появляется возможность мажоритарного разложения функции f_1 по переменной x_{n+1} , после проведения которого выполняется обратная замена этой переменной на переменную $!x_n$.

Пример 4.6. Выполнить мажоритарное разложение функции

$$f = x_1 \oplus x_2 = !x_1 x_2 \vee x_1 !x_2.$$

Так как эта функция немонотонна по каждой переменной, то требуемое разложение не может быть выполнено. Полагая $!x_2 = x_3$,

получим БФ $f_1 = !x_1 x_2 \vee x_1 x_3$, которая допускает искомое разложение по переменной x_3 :

$$\begin{aligned} !x_1 x_2 \vee x_1 x_3 &= \Phi_1 \# \Phi_2 \# x_3; \\ \text{MX}(0, 1, x_3, x_3; x_1, x_2) &= \text{MX}(0, x_3, x_3, 1; \Phi_1, \Phi_2). \end{aligned}$$

Среди четырех решений этого уравнения существует и такое:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2; \\ \Phi_2 &= \text{MX}(0, 1, 1, 0; x_1, x_2) = x_1 \oplus x_2, \end{aligned}$$

в котором функция Φ_2 совпадает с заданной функцией.

Выберем другое решение:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 1, 0, 0; x_1, x_2) = !x_1 x_2; \\ \Phi_2 &= \text{MX}(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2. \end{aligned}$$

Выполняя обратную замену $x_3 = !x_2$, получим:

$$x_1 \oplus x_2 = !x_1 x_2 \# (x_1 \vee x_2) \# !x_2.$$

Из изложенного следует, что для

$$f = !(x_1 \oplus x_2) = x_1 \oplus !x_2 = !x_1 !x_2 \# (x_1 \vee !x_2) \# x_2.$$

При этом отметим, что рассмотренные функции двух переменных реализуются схемами из трех МЭ:

$$\begin{aligned} x_1 \oplus x_2 &= (!x_1 \# x_2 \# 0) \# (x_1 \# x_2 \# 1) \# !x_2; \\ !(x_1 \oplus x_2) &= (!x_1 \# !x_2 \# 0) \# (x_1 \# !x_2 \# 1) \# x_2, \end{aligned}$$

в то время как остальные функции двух переменных реализуются одним МЭ.

6. Использование разложения Шеннона по немонотонной переменной x_i может повысить монотонность остаточных функций. Цена такого повышения монотонности — три МЭ, так как [10]

$$f = (f(x_i=0) \# !x_i \# 0) \# (f(x_i=1) \# x_i \# 1) \# 1.$$

Пример 4.7. Формула

$$f = !x_1 x_2 !x_3 \vee !x_1 x_3 !x_4 \vee x_1 !x_2 x_4 \vee x_1 x_2 x_3$$

немонотонна по всем переменным. Однако функции

$$f_1 = f(x_1=0) = x_2 !x_3 \vee x_3 !x_4 \text{ и } f_2 = f(x_1=1) = !x_2 x_4 \vee x_2 x_3$$

монотонны по двум переменным каждая.

7. Применение разложений Рида (разд. 3.1.1) по немонотонной переменной x_i или ее инверсии может повысить монотонность остаточных функций. Цена такого повышения монотонности также равна трем МЭ, так как в соотношениях

$$f = f_1 \oplus f_3 x_i \text{ и } f = f_2 \oplus f_3 !x_i,$$

как показано выше, операция «сумма по модулю два» реализуется схемой из трех МЭ.

Пример 4.8. Формула

$$f = !x_1 !x_2 x_3 \vee x_1 !x_3 \vee x_2 !x_3$$

немонотонна по всем переменным. Выполняя разложение Рида по переменной $!x_1$, получим $f_2 = f(x_1=1) = !x_3$, $f_3 = !x_2$. Формула $f = !x_3 \oplus !x_2 !x_1$ реализуется схемой из четырех МЭ.

8. Использование преобразований Артюхова—Шальто по немонотонной переменной x_i может повысить монотонность функций, реализуемых после преобразований, что также требует применения трех МЭ.

Пример 4.9. Формула

$$f = !x_1 !x_2 !x_3 \vee x_1 x_2 x_3$$

немонотонна по всем переменным. Выполняя преобразование по переменной x_3 , получим формулу $f = f_4 \oplus x_3$, в которой формула $f_4 = !x_1 \# !x_2 \# x_3$ монотонна по всем переменным и реализуется одним МЭ. Таким образом, заданная БФУ реализуется схемой из четырех МЭ.

4.1.6. Оценки сложности схем

Предположим, что прямые и инверсные входные переменные равнодоступны.

1. Выше было показано, что произвольная БФ в базисе $\{\&, \vee, !\}$ из h букв реализуется формульным методом пирамидальной схемой или графовым методом каскадной схемой с разветвлениями, число МЭ в которых определяется соотношением

$$L(h) = h - 1.$$

Это соотношение справедливо и для реализации произвольной БПФ из h букв каскадной схемой из МЭ.

2. Так как при реализации монотонных БФУ n переменных с помощью мажоритарного разложения каждый МЭ в пирамидальной

схеме исключает одну переменную, сложность реализации функций этого класса определяется соотношением

$$L_{\text{м}}(n) \leq 2^{n-1} - 1.$$

Таким образом, для монотонных БФУ справедливо соотношение

$$L_{\text{м}}(h, n) \leq \min(h - 1; 2^{n-1} - 1).$$

3. Так как произвольная БФУ n переменных может быть сведена к монотонной БФУ, зависящей от $2n$ переменных, то

$$L(n) \leq 2^{2n-1} - 1.$$

4. Так как произвольная БФУ n переменных может быть реализована пирамидальной схемой из мультиплексоров «2 в 1», число которых определяется соотношением

$$M(n) \leq 2^{n-1} - 1,$$

а каждый такой мультиплексор реализуется схемой из трех МЭ, то

$$L(n) \leq 3(2^{n-1} - 1).$$

5. Так как произвольная БФУ трех переменных реализуется схемой, содержащей не более четырех МЭ [5], то

$$L(n) \leq 3(2^{n-3} - 1) + 4 \cdot 2^{n-3} = 7 \cdot 2^{n-3} - 3.$$

6. Таким образом,

$$L_{\text{м}}(h, n) \leq \min(h - 1; 7 \cdot 2^{n-3} - 3).$$

7. В табл. 4.2 приведены значения функций $L_{\text{м}}(n)$ и $L(n)$ для $n = 3, \dots, 6$.

Т а б л и ц а 4.2

n	3	4	5	6
$L_{\text{м}}(n)$	3	7	15	31
$L(n)$	4	11	25	55

8. Из этой таблицы следует, что в каталоге реализаций всех NPN -типов БФУ четырех переменных схемами из МЭ [15] по край-

ней мере одна из схем избыточна, так как она содержит 13 МЭ. Эта функция $f = |1001\ 0110\ 0110\ 1000|^T$ реализуется формулой

$$f = !x_1 (!x_2 \oplus x_3 \oplus x_4) \vee x_1 (!x_2 (x_3 \oplus x_4) \vee x_2 !x_3 !x_4).$$

Из изложенного в разд. 4.1.9 следует, что при представлении этой формулы суперпозицией $f_1 = !x_2 \oplus x_3 \oplus x_4$, $f_2 = !x_2 (x_3 \oplus x_4) \vee x_2 !x_3 !x_4$, $f = !x_1 f_1 \vee x_1 f_2$ она реализуется схемой из десяти МЭ.

4.1.7. Реализация не полностью определенных булевых функций

Если на заданных входных наборах столбец значений БФУ не содержит одновременно фрагментов $!x_n$ и x_n , то это свойство должно быть сохранено и после доопределения. Если исходный столбец не содержит указанных фрагментов, то БФУ может быть доопределена так, что она не будет зависеть по крайней мере от переменной x_n .

Доопределение функции должно выполняться (если это возможно) так, чтобы ее можно было:

- разложить не только по одной переменной x_n , но и одновременно по двум переменным x_{n-1} и x_n ;
- разложить по подфункции $Q(X_1)$;
- мультиплексорно декомпозировать при выполнении отношений покрытия.

Пример 4.10. Пусть задана БФУ

$$f(x_1, x_2, x_3, x_4) = |00 - - 0 - -1\ 11 -1\ 0 -1 -|^T.$$

Доопределим эту функцию так, чтобы она не зависела от переменной x_4 и состояла из фрагментов трех типов 0, Q , 1:

$$f = |0000\ 0011\ 1111\ 0011|^T.$$

Так как $Q = x_3$, запишем уравнение:

$$f = \Phi_1 \# \Phi_2 \# x_3;$$

$$MX(0, x_3, 1, x_3; x_1, x_2) = MX(0, x_3, x_3, 1; \Phi_1, \Phi_2).$$

Это уравнение имеет четыре решения, из которых выберем следующее:

$$\Phi_1 = MX(0, 0, 1, 0; x_1, x_2) = x_1 !x_2;$$

$$\Phi_2 = MX(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2.$$

Таким образом, заданная БФУ реализуется схемой из четырех МЭ.

Пример 4.11. Пусть задана БФУ

$$f(x_1, x_2, x_3, x_4) = |000 - 0 - 01 \ 01 - 1 \ -111|^T.$$

Доопределим эту функцию так, чтобы она при $n - k = 2$ состояла из двух типов фрагментов длиной четыре: $f = |0001 \ 0001 \ 0111 \ 0111|^T$.

Полагая $Q_0 = x_3 x_4$, $Q_1 = x_3 \vee x_4$, запишем уравнение

$$\begin{aligned} f &= \Phi \# x_3 \# x_4; \\ \text{MX}(Q_0, Q_0, Q_1, Q_1; x_1, x_2) &= \text{MX}(Q_0, Q_1; \Phi); \\ \Phi &= \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1. \end{aligned}$$

Таким образом, заданная БФУ реализуется одним МЭ.

4.1.8. Реализация булевых функций, описываемых формулами, бесповторными в базисе И, ИЛИ, НЕ

Формульный и графовый методы для ББФ в базисе $\{\&, \vee, !\}$ из h букв строят бесповторные схемы из $h - 1$ МЭ. Покажем, что использование мультиплексорного метода позволяет для БФУ, описываемых такими формулами, строить повторные схемы, в том числе и из $h - 1$ МЭ, что весьма удивительно.

Пример 4.12. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0000 \ 0101 \ 0011 \ 0111|^T.$$

Эта БФУ содержит нечетное число единиц, что является необходимым условием бесповторности. Реализуем эту БФУ с помощью нескольких подходов.

1. Так как она монотонна по переменной x_4 , то выполним мажоритарную декомпозицию по этой переменной, применяя мультиплексорный метод:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= \Phi_1 \# \Phi_2 \# x_4; \\ \text{MX}(0, 0, x_4, x_4, 0, 1, x_4, 1; x_1, x_2, x_3) &= \text{MX}(0, x_4, x_4, 1; \Phi_1, \Phi_2). \end{aligned}$$

Это уравнение имеет восемь решений:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, ?, ?, 0, 1, ?, 1; x_1, x_2, x_3); \\ \Phi_2 &= \text{MX}(0, 0, ?, ?, 0, 1, ?, 1; x_1, x_2, x_3). \end{aligned}$$

Из рассмотрения функции Φ_1 следует, что она может быть доопределена до мажоранты. При этом функция Φ_2 доопределяется автоматически:

$$\Phi_1 = MX(0, 0, 0, 1, 0, 1, 1; x_1, x_2, x_3) = x_1 \# x_2 \# x_3;$$

$$\Phi_2 = MX(0, 0, 1, 0, 0, 1, 0; x_1, x_2, x_3) = !x_1 x_2 !x_3 \vee x_1 x_3.$$

Если функцию Φ_2 реализовать графовым методом, то функция f реализуется недревовидной схемой из шести МЭ (рис. 4.7). При использовании формульного метода для реализации функции Φ_2 строится повторная древовидная схема из того же числа МЭ (рис. 4.8).

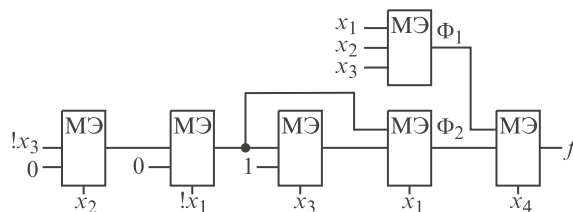


Рис. 4.7

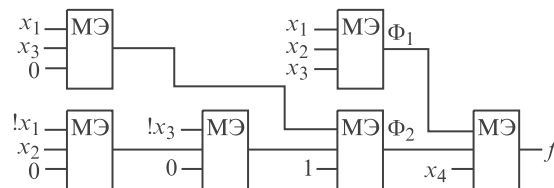


Рис. 4.8

Если не стремиться доопределить функцию Φ_1 до мажоранты, то может быть выбрано другое решение, например

$$\Phi_1 = MX(0, 0, 0, 0, 0, 1, 0; x_1, x_2, x_3) = x_1 x_3;$$

$$\Phi_2 = MX(0, 0, 1, 1, 0, 1, 1; x_1, x_2, x_3) = x_1 x_3 \vee x_2.$$

Так как $\Phi_2 = \Phi_1 \vee x_2$, то заданная БФУ реализуется недревовидной схемой из трех МЭ (рис. 4.9), которая изоморфна бесповторной каскадной схеме с разветвлениями (рис. 4.10).

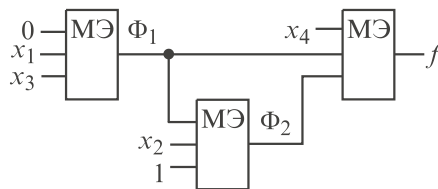


Рис. 4.9

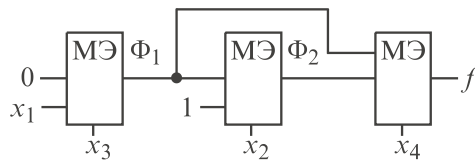


Рис. 4.10

2. Выполним мультиплексорную декомпозицию при $n - k = 2$:

$$f(x_1, x_2, x_3, x_4) = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2),$$

где $Q_0 = 0$, $Q_1 = x_4$, $Q_2 = x_3$, $Q_3 = x_3 \vee x_4$. Так как отношения покрытия выполняются, то

$$f = Q_0 \vee Q_1 \Phi_2 \vee Q_2 \Phi_1 \vee Q_3 \Phi_1 \Phi_2 = \Phi_1 x_3 \vee \Phi_2 x_4.$$

Для определения функций Φ_1 и Φ_2 запишем исходное уравнение в МФ и решим его:

$$\begin{aligned} \text{MX}(Q_0, Q_1, Q_2, Q_3; x_1, x_2) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2); \\ \Phi_1 &= x_1; \quad \Phi_2 = x_2. \end{aligned}$$

Таким образом, получили ББФ $f = x_1 x_3 \vee x_2 x_4$. Так как заданная БФУ реализуется **б е с п о в т о р н о й** БФ в рассматриваемом базисе, которая в силу этого является **м о н о т о н н о й**, то можно утверждать, что схемы на рис. 4.7, 4.8, построенные с помощью мультиплексорного метода, обладают **у д и в и т е л ь н ы м и** свойствами:

- они **п о в т о р н ы**;
- они **н е м о н о т н ы**, несмотря на то что МЭ монотонны;
- они содержат число МЭ большее, чем $h - 1$.

3. Формульный метод по полученной ББФ строит бесповторную древовидную схему из трех МЭ (рис. 4.11).

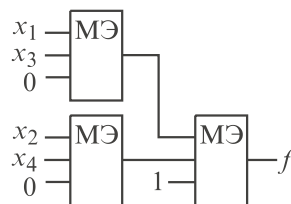


Рис. 4.11

4. При изменении порядка переменных в заданной БФУ

$$f(x_1, x_3, x_2, x_4) = |0001\ 0001\ 0001\ 1111|^T$$

появляется возможность выполнить мажоритарное разложение по функции $Q = x_2 x_4$:

$$\begin{aligned} f(x_1, x_3, x_2, x_4) &= \Phi_1 \# \Phi_2 \# Q; \\ \text{MX}(Q, Q, Q, 1; x_1, x_3) &= \text{MX}(0, Q, Q, 1; \Phi_1, \Phi_2). \end{aligned}$$

Среди восьми решений этого уравнения выберем следующие два:

$$\begin{cases} \Phi_1 = \text{MX}(1, 1, 0, 1; x_1, x_3) = !x_1 \vee x_3; \\ \Phi_2 = \text{MX}(0, 0, 1, 1; x_1, x_3) = x_1; \\ \Phi_1 = \text{MX}(1, 0, 1, 1; x_1, x_3) = x_1 \vee !x_3; \\ \Phi_2 = \text{MX}(0, 1, 0, 1; x_1, x_3) = x_3. \end{cases}$$

Таким образом, заданная БФУ может быть реализована также и повторными древовидными схемами из трех МЭ (рис. 4.12, 4.13).

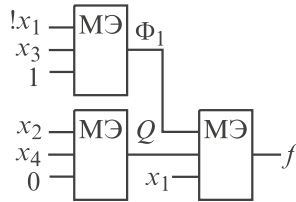


Рис. 4.12

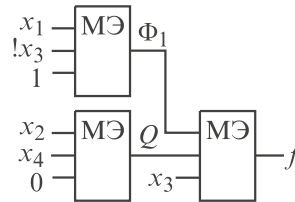


Рис. 4.13

5. Графовый метод строит по полученной ББФ неповторную каскадную схему с разветвлениями (рис. 4.14), с точностью до обозначений совпадающую со схемой на рис. 4.10.

Схема на рис. 4.10 реализует БФ при «естественном» порядке записи, а схема на рис. 4.14 — при противоположном порядке. И в том и в другом случае реализация выполняется не по подформулам, а по фрагментам формулы. Для схемы на рис. 4.14 справедливы соотношения:

$$\begin{aligned} \Phi_3 &= 0 \# x_4 \# x_2 = x_4 x_2; \quad \Phi_4 = \Phi_3 \# 1 \# x_3 = x_4 x_2 \vee x_3; \\ f &= \Phi_3 \# \Phi_4 \# x_1 = x_4 x_2 \vee x_3 x_1. \end{aligned}$$

При этом отметим, что в этой формуле подформула Φ_4 отсутствует.

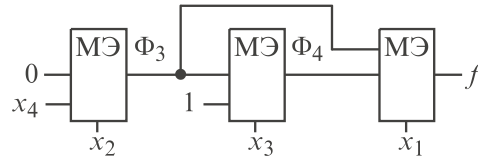


Рис. 4.14

Таким образом, для заданной БФУ построены пять типов схем: недревовидная, повторная древовидная, бесповторная древовидная, повторная древовидная и бесповторная каскадная с разветвлениями, причем схемы трех последних типов содержат по три МЭ.

Рассмотренная БФУ не порождает каскадные схемы без разветвлений (каскадные схемы). Это связано с тем, что она реализуется непороговой ББФ. Булевы функции, соответствующие бесповторным пороговым формулам, могут быть реализованы в том числе и бесповторными и повторными каскадными схемами.

Пример 4.13. Реализовать БФУ

$$f(x_1, \dots, x_5) = |0101 \ 0111 \ 0101 \ 0111 \ 0111 \ 0111 \ 0101 \ 0111|^T.$$

Эта БФУ содержит нечетное число единиц, что является необходимым условием бесповторности. Она реализуется БПФ $f = (x_1!x_2 \vee \vee x_3)x_4 \vee x_5$.

1. Бесповторная каскадная схема, построенная формульным методом, приведена на рис. 3.20.

2. Так как заданная БФУ монотонна по переменной x_5 , то выполним мажоритарное разложение по этой переменной:

$$f(x_1, \dots, x_5) = \Phi_1 \# \Phi_2 \# x_5;$$

$$\text{MX}(x_5, x_5, x_5, 1, x_5, x_5, x_5, 1, x_5, 1, x_5, 1, x_5, x_5, 1;$$

$$x_1, x_2, x_3, x_4) = \text{MX}(0, x_5, x_5, 1; \Phi_1, \Phi_2).$$

Это уравнение имеет $2^{11} = 2048$ решений, одно из которых следующее:

$$\Phi_1 = \text{MX}(0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1; x_1, x_2, x_3, x_4) = \\ = x_4;$$

$$\Phi_2 = \text{MX}(1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1; x_1, x_2, x_3, x_4) = \\ = x_1!x_2 \vee x_3 \vee!x_4.$$

Таким образом, в этом случае последний МЭ в схеме реализует БФУ

$$f = \Phi_2 \# x_4 \# x_5.$$

Это соотношение, внешне напоминающее мажоритарное разложение по переменным x_4 и x_5 , им не является, так как в данном случае функция Φ_2 зависит от переменной x_4 , т.е. по этой переменной декомпозиция является неразделительной.

Если функцию Φ_2 реализовать формульным методом, то получится повторная каскадная схема из четырех МЭ (рис. 4.15), реализующая неповторную БФ из пяти букв, что до сих пор было неизвестно. Удивительным является также и то, что построенная схема из $h - 1$ элементов «немонотонна» по переменной x_4 , несмотря на то что реализуемая БФУ по этой переменной монотонна.

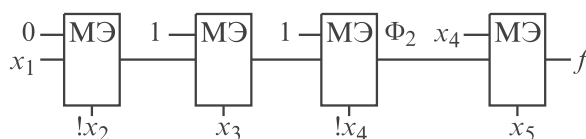


Рис. 4.15

Бесповторная каскадная схема для рассмотренной БФУ построена мультиплексорным методом в примере 3.20.

Из изложенного в настоящем разделе автор предполагает, что минимальное число МЭ, требующихся для реализации ББФ в базисе $\{\&, \vee, !\}$ из h букв, равно $h - 1$. Эта оценка обеспечивается в классе древовидных схем формульным методом, а переход к недревовидным схемам, видимо, не позволяет снизить эту оценку. Естественно, что для повторных БФ древовидные схемы во многих случаях не минимальны.

4.1.9. Реализация представителей всех PN -типов булевых функций трех переменных

Существуют 16 PN -типов БФУ, существенно зависящих от трех переменных. Определим для выбранных представителей каждого из PN -типов сложность простейшей схемы, получаемой с помощью предлагаемых подходов.

Пример 4.14. Булева функция $f_1(x_1, x_2, x_3) = |0001\ 0111|^T = x_1 \# \# x_2 \# x_3$ реализуется одним МЭ.

Пример 4.15. Булевы формулы, неповторные в базисе $\{\&, \vee, !\}$:

$$f_2 = x_1 x_2 x_3, \quad f_3 = x_1 x_2 \vee x_3, \quad f_4 = x_1 \vee x_2 \vee x_3, \quad f_5 = (x_1 \vee x_2) x_3,$$

реализуются, например, формульным или графовым методом, схемами из двух МЭ.

Пример 4.16. Реализовать БФУ

$$f_6(x_1, x_2, x_3) = |0001 \ 0100|^T = (!x_1 x_2 \vee x_1 !x_2) x_3.$$

1. Так как $h = 5$, то формульный и графовый методы строят схемы из четырех МЭ.

2. Формула $f_6 = (x_1 \oplus x_2) x_3$ реализуется схемой из четырех МЭ, так как подформула $f = x_1 \oplus x_2$ реализуется подсхемой из трех МЭ.

3. Так как БФУ монотонна по переменной x_3 , то на основе мультиплексорного метода

$$f_6 = \Phi_1 \# \Phi_2 \# x_3;$$

$$\text{MX}(0, x_3, x_3, 0; x_1, x_2) = \text{MX}(0, x_3, x_3, 1; \Phi_1, \Phi_2).$$

Это уравнение имеет четыре решения, одно из которых следующее:

$$\Phi_1 = \text{MX}(0, 0, 1, 0; x_1, x_2) = x_1 !x_2;$$

$$\Phi_2 = \text{MX}(0, 1, 0, 0; x_1, x_2) = !x_1 x_2.$$

Таким образом, заданная БФУ реализуется схемой из трех МЭ:

$$f_6 = (x_1 \# !x_2 \# 0) \# (!x_1 \# x_2 \# 0) \# x_3.$$

4. В [15] эта БФУ также реализована схемой из трех МЭ, построенной на основе неразделительной декомпозиции:

$$f_6 = (x_1 \# x_2 \# x_3) \# (!x_1 \# x_2 \# 1) \# 0.$$

Пример 4.17. Реализовать БФУ

$$f_7(x_1, x_2, x_3) = |0111 \ 1101|^T = !x_1 x_2 \vee x_1 !x_2 \vee x_3.$$

1. Так как $h = 5$, то формульный и графовый методы строят схемы из четырех МЭ.

2. Формула $f_7 = (x_1 \oplus x_2) \vee x_3$ реализуется схемой из четырех МЭ.

3. Так как БФУ монотонна по переменной x_3 , то на основе мультиплексорного метода

$$f_7 = \Phi_1 \# \Phi_2 \# x_3;$$

$$\text{MX}(x_3, 1, 1, x_3; x_1, x_2) = \text{MX}(0, x_3, x_3, 1; \Phi_1, \Phi_2).$$

Это уравнение имеет четыре решения, одно из которых следующее:

$$\begin{aligned}\Phi_1 &= \text{MX}(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2; \\ \Phi_2 &= \text{MX}(0, 1, 0, 0; x_1, x_2) = !x_1 \vee !x_2.\end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из трех МЭ:

$$f_7 = (x_1 \# x_2 \# 1) \# (!x_1 \# !x_2 \# 1) \# x_3.$$

4. В [15] эта БФУ также реализована схемой из трех МЭ, построенной на основе неразделительной декомпозиции:

$$f_7 = (!x_1 \# x_2 \# x_3) \# (x_1 \# !x_2 \# 0) \# 1.$$

Пример 4.18. Реализовать БФУ

$$\begin{aligned}f_8(x_1, x_2, x_3) &= |0110 \ 1001|^T = \\ &= !x_1 (!x_2 x_3 \vee x_2 !x_3) \vee x_1 (!x_2 !x_3 \vee x_2 x_3).\end{aligned}$$

1. Так как $h = 10$, то формульный и графовый методы строят схемы из девяти МЭ.

2. Формула $f_8 = x_1 \oplus x_2 \oplus x_3$ реализуется схемой из шести МЭ.

3. Так как БФУ немонотонна по всем переменным, то для проведения мажоритарного разложения выполним в БФ в базисе $\{\&, \vee, !\}$ замену $!x_3 = x_4$:

$$\begin{aligned}f(x_1, x_2, x_3, x_4) &= x_1 !x_2 x_4 \vee !x_1 x_2 x_4 \vee !x_1 !x_2 x_3 \vee x_1 x_2 x_3 = \\ &= |0011 \ 0101 \ 0101 \ 0011|^T.\end{aligned}$$

Для этой БФУ можно записать уравнение

$$f = \Phi_1 \# \Phi_2 \# x_4;$$

$$\text{MX}(0, 1, x_4, x_4, x_4, x_4, 0, 1; x_1, x_2, x_3) = \text{MX}(0, x_4, x_4, 1; \Phi_1, \Phi_2).$$

Среди 16 решений этого уравнения может быть выбрано следующее:

$$\begin{aligned}\Phi_1 &= \text{MX}(0, 1, 1, 1, 0, 0, 0, 1; x_1, x_2, x_3) = !x_1 \# x_2 \# x_3; \\ \Phi_2 &= \text{MX}(0, 1, 0, 0, 1, 1, 0, 1; x_1, x_2, x_3) = x_1 \# !x_2 \# x_3.\end{aligned}$$

Если выполнить обратную замену $x_4 = !x_3$, то получим искомую реализацию из трех МЭ:

$$f_8 = (!x_1 \# x_2 \# x_3) \# (x_1 \# !x_2 \# x_3) \# !x_3.$$

Если заменить переменную x_3 , то

$$f_8 = (x_1 \# !x_2 \# !x_3) \# (x_1 \# x_2 \# !x_3) \# x_3.$$

4. Аналогичные решения можно получить с помощью использования Кона—Линдемана [10].

Следствием полученных результатов является реализация функции «четность» трех переменных $f_{81} = !f_8$:

$$f_{81} = (!x_1 \# x_2 \# !x_3) \# (x_1 \# !x_2 \# !x_3) \# x_3;$$

$$f_{81} = (x_1 \# !x_2 \# x_3) \# (x_1 \# x_2 \# x_3) \# !x_3.$$

Пример 4.19. Реализовать БФУ

$$f_9(x_1, x_2, x_3) = |0011 \ 0101|^T = !x_1 x_2 \vee x_1 x_3.$$

1. В [10] эта БФУ реализована с помощью метода Акерса [3]. Она сводится к логически пассивной БФУ четырех переменных и реализуется схемой из пяти МЭ.

2. Так как $h = 4$, то формульный и графовый методы строят схемы из трех МЭ.

3. Так как БФУ монотонна по переменной x_3 , то выполним мажоритарное разложение по этой переменной:

$$f_9 = \Phi_1 \# \Phi_2 \# x_3;$$

$$MX(0, 1, x_3, x_3; x_1, x_2) = MX(0, x_3, x_3, 1; \Phi_1, \Phi_2).$$

Среди четырех решений этого уравнения выберем следующее:

$$\Phi_1 = MX(0, 1, 0, 0; x_1, x_2) = !x_1 x_2;$$

$$\Phi_2 = MX(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2.$$

Таким образом, заданная БФУ реализуется схемой из трех МЭ. Учет свойства $\Phi_1 \leq \Phi_2$ реализацию не упрощает.

Пример 4.20. Реализовать БФУ

$$f_{10}(x_1, x_2, x_3) = |0110 \ 1010|^T = !x_1 !x_2 x_3 \vee (x_1 \vee x_2) !x_3.$$

1. Так как $h = 6$, то формульный и графовый методы строят схемы из пяти МЭ.

2. Так как БФУ немонотонна по всем переменным, то выполним в БФ замену $!x_3 = x_4$:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= !x_1 !x_2 x_3 \vee (x_1 \vee x_2) x_4 = \\ &= |0011 \ 0101 \ 0101 \ 0101|^T. \end{aligned}$$

Проведем мажоритарное разложение по переменной x_4 :

$$\begin{aligned} f &= \Phi_1 \# \Phi_2 \# x_4; \\ \text{MX}(0, 1, x_4, x_4, x_4, x_4, x_4, x_4; x_1, x_2, x_3) &= \\ &= \text{MX}(0, x_4, x_4, 1; \Phi_1, \Phi_2). \end{aligned}$$

Среди 64 решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 1, 1, 1, 1, 1, 1, 1; x_1, x_2, x_3) = x_1 \vee x_2 \vee x_3; \\ \Phi_2 &= \text{MX}(0, 1, 0, 0, 0, 0, 0, 0; x_1, x_2, x_3) = !x_1 !x_2 x_3. \end{aligned}$$

Таким образом, после обратной замены $x_4 = !x_3$ заданная БФУ реализуется схемой из пяти МЭ. Учет свойства $\Phi_2 \leq \Phi_1$ реализацию не упрощает.

3. Выполним мультиплексорную декомпозицию при $n - k = 1$, $Q_0 = x_3$, $Q_1 = !x_3$:

$$f_{10}(x_1, x_2, x_3) = \text{MX}(Q_0, Q_1; \Phi).$$

При этом $f_{10} = x_3 !\Phi \vee !x_3 \Phi = \Phi \oplus x_3$. Для определения функции Φ запишем исходное уравнение в МФ и решим его:

$$\begin{aligned} \text{MX}(Q_0, Q_1, Q_1, Q_1; x_1, x_2) &= \text{MX}(Q_0, Q_1; \Phi); \\ \Phi &= \text{MX}(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2. \end{aligned}$$

Таким образом, $f_{10} = (x_1 \vee x_2) \oplus x_3$ и реализуется схемой из четырех МЭ.

4. Используем разложение Рида по переменной $!x_1$: $f_{10}(x_1 = 0) = x_2 \oplus x_3$, $f_{10}(x_1 = 1) = !x_3$, $f_{10}(x_1 = 0) \oplus f_{10}(x_1 = 1) = !x_2$, $f_{10} = !x_3 \oplus !x_2 !x_1$. Эта БФ реализуется схемой из четырех МЭ.

Пример 4.21. Реализовать БФУ

$$f_{11}(x_1, x_2, x_3) = |1100 \ 0001|^T = !x_1 !x_2 \vee x_1 x_2 x_3.$$

1. При использовании разложения Рида по переменной x_1 : $f_{12} = x_1 (!x_2 \vee x_3) \oplus !x_2$. Эта БФ реализуется схемой из пяти МЭ.

2. Так как $h = 5$, то формульный и графовый методы строят схемы из четырех МЭ.

3. Так как БФУ монотонна по переменной x_3 , то выполним мажоритарное разложение по этой переменной:

$$\begin{aligned} f_{11} &= \Phi_1 \# \Phi_2 \# x_3; \\ \text{MX}(1, 0, 0, x_3; x_1, x_2) &= \text{MX}(0, x_3, x_3, 1; \Phi_1, \Phi_2). \end{aligned}$$

Среди четырех решений этого уравнения выберем следующее:

$$\begin{aligned}\Phi_1 &= MX(1, 0, 0, 0; x_1, x_2) = !x_1 !x_2; \\ \Phi_2 &= MX(1, 0, 0, 1; x_1, x_2) = !(x_1 \oplus x_2).\end{aligned}$$

Независимая реализация этих функций приводит к подсхеме из четырех МЭ, но так как $\Phi_1 \leq \Phi_2$, то функции $\Phi_1 = !x_1 !x_2$, $\Phi_2 = \Phi_1 \vee \vee x_1 x_2$ реализуются подсхемой из трех МЭ. Таким образом, заданная БФУ реализуется схемой из четырех МЭ.

Пример 4.22. Реализовать БФУ

$$f_{12}(x_3, x_2, x_1) = |0111 \ 0110|^T = !x_1 x_2 \vee x_1 (!x_2 \vee !x_3).$$

1. Выполним мультиплексорную декомпозицию при $n - k = 2$, $Q_0 = x_1 \oplus x_2$, $Q_1 = x_1 \vee x_2$, $\Phi = !x_3$:

$$f_{12} = MX(Q_0, Q_1; \Phi).$$

Так как $Q_0 \leq Q_1$, то $f_{12} = Q_0 \vee Q_1 \Phi$. Таким образом, БФУ в этом случае реализуется схемой из шести МЭ.

2. Разложение Рида по переменной x_3 ($f_{12} = (x_1 \vee x_2) \oplus x_1 x_2 x_3$) также строит схему из шести МЭ.

3. Булева формула $f_{12} = (x_1 \oplus x_2) \vee x_1 !x_3$ реализуется схемой из пяти МЭ.

4. Так как БФУ $f_{12}(x_1, x_2, x_3) = |0011 \ 1110|^T$ монотонна по переменной $!x_3$, то

$$\begin{aligned}f_{12}(x_1, x_2, x_3) &= \Phi_1 \# \Phi_2 \# !x_3; \\ MX(0, 1, 1, !x_3; x_1, x_2) &= MX(0, !x_3, !x_3, 1; \Phi_1, \Phi_2).\end{aligned}$$

Из двух решений этого уравнения выберем следующее:

$$\begin{aligned}\Phi_1 &= MX(0, 1, 1, 0; x_1, x_2) = x_1 \oplus x_2; \\ \Phi_2 &= MX(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2.\end{aligned}$$

Таким образом, БФУ реализуется схемой из пяти МЭ. Учет свойства $\Phi_1 \leq \Phi_2$ в данном случае усложняет реализацию.

5. Так как $h = 5$, то формульный и графовый методы строят схемы из четырех МЭ.

6. В [15] эта БФУ также реализована схемой из четырех МЭ:

$$f_{12} = (x_1 \vee x_2) \# (!x_2 \vee !x_3) \# x_1 x_2.$$

Пример 4.23. Реализовать БФУ

$$f_{13}(x_1, x_2, x_3) = |1110 \ 0111|^T = !x_1!x_2 \vee x_2!x_3 \vee x_1x_3.$$

1. Так как БФУ «зеркальна» [24] по переменной $!x_1$, то $f_{13} = (!x_1 \oplus x_2) \vee (!x_1 \oplus x_3)$. Эта формула может быть реализована схемой из семи МЭ.

2. Так как БФУ немонотонна по всем переменным, то выполним в БФ замену $!x_3 = x_4$:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= !x_1!x_2 \vee x_1x_3 \vee x_2x_4 = \\ &= |1111 \ 0101 \ 0011 \ 0111|^T. \end{aligned}$$

Выполним мажоритарное разложение по переменной x_4 :

$$f = \Phi_1 \# \Phi_2 \# x_4;$$

$$\text{MX}(1, 1, x_4, x_4, 0, 1, x_4, 1; x_1, x_2, x_3) = \text{MX}(0, x_4, x_4, 1; \Phi_1, \Phi_2).$$

Среди восьми решений этого уравнения выберем следующее:

$$\Phi_1 = \text{MX}(1, 1, 0, 0, 0, 1, 0, 1; x_1, x_2, x_3) = !x_1!x_2 \vee x_1x_3;$$

$$\Phi_2 = \text{MX}(1, 1, 1, 1, 0, 1, 1, 1; x_1, x_2, x_3) = x_1 \vee x_2x_3.$$

Таким образом, после обратной замены $x_4 = !x_3$ заданная БФУ реализуется схемой из шести МЭ.

3. Так как $h = 6$, то формульный и графовый методы строят схемы из пяти МЭ.

4. Выполним мультиплексорную декомпозицию при $n - k = 2$, $Q_0 = !x_1 \vee !x_2$, $Q_1 = x_1 \vee x_2$, $\Phi = x_1$. При этом $f_{13} = \text{MX}(Q_0, Q_1; \Phi) = Q_0!x_1 \vee Q_1x_1$.

Следовательно, в этом случае также строится схема из пяти МЭ.

5. Используем преобразование Артюхова—Шалыто по переменной x_1 . При этом $f_{13} = (!x_1 \# !x_2 \# !x_3) \oplus x_1$. Таким образом, построена неразделительная декомпозиция, которая реализуется схемой из четырех МЭ.

Пример 4.24. Реализовать БФУ

$$f_{14}(x_1, x_2, x_3) = |1000 \ 0001|^T = !x_1!x_2!x_3 \vee x_1x_2x_3.$$

1. Так как БФУ «зеркальна» по переменной $!x_1$, то $f_{14} = (!x_1 \oplus x_2)(!x_1 \oplus x_3)$. Эта формула может быть реализована схемой из семи МЭ.

2. Так как БФУ немонотонна по всем переменным, то выполним в БФ замену $x_3 = x_4$:

$$f(x_1, x_2, x_3, x_4) = !x_1!x_2!x_3 \vee x_1x_2x_4 = \\ = |1100 \ 0000 \ 0000 \ 0101|^T.$$

Проведем мажоритарное разложение по переменной x_4 :

$$f = \Phi_1 \# \Phi_2 \# x_4;$$

$$\text{MX}(1, 0, 0, 0, 0, 0, x_4, x_4; x_1, x_2, x_3) = \text{MX}(0, x_4, x_4, 1; \Phi_1, \Phi_2).$$

Среди четырех решений этого уравнения выберем следующее:

$$\Phi_1 = \text{MX}(1, 0, 0, 0, 0, 0, 0, 0; x_1, x_2, x_3) = !x_1!x_2!x_3;$$

$$\Phi_2 = \text{MX}(1, 0, 0, 0, 0, 0, 1, 1; x_1, x_2, x_3) = !x_1!x_2!x_3 \vee x_1x_2.$$

Так как $\Phi_1 \leq \Phi_2$, то функции $\Phi_1 = !x_1!x_2!x_3$, $\Phi_2 = \Phi_1 \vee x_1x_2$ реализуются подсхемой из четырех МЭ. Таким образом, после обратной замены $x_4 = x_3$ заданная БФУ реализуется схемой из пяти МЭ.

3. Выполним мультиплексорную декомпозицию при $n - k = 1$, $Q_0 = 0$, $Q_1 = x_3$, $Q_2 = !x_3$, $Q_3 = 1$:

$$f_{14} = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2) = \Phi_1!x_3 \vee \Phi_2x_3.$$

Для определения функций Φ_1 и Φ_2 запишем исходное уравнение в МФ и решим его:

$$\text{MX}(Q_2, Q_0, Q_0, Q_1; x_1, x_2) = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2);$$

$$\Phi_1 = \text{MX}(1, 0, 0, 0; x_1, x_2) = !x_1!x_2;$$

$$\Phi_2 = \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1x_2.$$

Таким образом, БФУ реализуется схемой из пяти МЭ.

4. Так как $h = 6$, то формульный и графовый методы также строят схемы из пяти МЭ.

5. Используем преобразование Артюхова—Шалыто по переменной x_1 . При этом $f_{14} = (x_1 \# !x_2 \# !x_3) \oplus x_1$. Таким образом, построена неразделительная декомпозиция, которая реализуется схемой из четырех МЭ.

6. В [15] показано, что другая неразделительная декомпозиция

$$f_{14} = !x_1!x_3 \# (x_1 \vee !x_2) \# x_2x_3$$

также реализуется схемой из четырех МЭ.

Пример 4.25. Реализовать БФУ

$$f_{15}(x_1, x_2, x_3) = |1001 \ 0111|^T = !x_1!x_2!x_3 \vee (x_1 \vee x_2)x_3 \vee x_1x_2.$$

1. Так как $h = 8$, то формульный и графовый методы также строят схемы из семи МЭ.

2. Выполним мультиплексорную декомпозицию при $n - k = 2$, $Q_0 = !x_1 \oplus x_2$, $Q_1 = x_1 \vee x_2$, $\Phi = x_1$:

$$f_{15} = \text{MX}(Q_0, Q_1; \Phi) = Q_0! \Phi \vee Q_1 \Phi.$$

Таким образом, в этом случае также строится схема из семи МЭ.

3. Так как БФУ немонотонна по всем переменным, то выполним в БФ замену $x_3 = x_4$:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= !x_1!x_2!x_3 \vee (x_1 \vee x_2)x_4 \vee x_1x_2 = \\ &= |1100 \ 0101 \ 0101 \ 1111|^T. \end{aligned}$$

Выполним мажоритарное разложение по переменной x_4 :

$$\begin{aligned} f &= \Phi_1 \# \Phi_2 \# x_4; \\ \text{MX}(1, 0, x_4, x_4, x_4, x_4, 1, 1; x_1, x_2, x_3) &= \\ &= \text{MX}(0, x_4, x_4, 1; \Phi_1, \Phi_2). \end{aligned}$$

Среди 16 решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(1, 0, 1, 1, 0, 0, 1, 1; x_1, x_2, x_3) = !x_1x_3 \vee x_2; \\ \Phi_2 &= \text{MX}(1, 0, 0, 0, 1, 1, 1, 1; x_1, x_2, x_3) = x_1 \vee !x_2!x_3. \end{aligned}$$

Таким образом, после обратной замены $x_4 = x_3$ заданная БФУ реализуется схемой из пяти МЭ.

4. Преобразуем заданную БФ $f_{15} = !x_1!x_2!x_3 \vee (x_1 \# x_2 \# x_3)$. Эта формула реализуется схемой из четырех МЭ.

Пример 4.26. Реализовать БФУ

$$f_{16}(x_1, x_2, x_3) = |0010 \ 1001|^T = !x_1x_2!x_3 \vee x_1(!x_2!x_3 \vee x_2x_3).$$

1. Так как $h = 8$, то формульный и графовый методы также строят схемы из семи МЭ.

2. Схема из семи МЭ может быть построена и по БФ

$$f_{16} = !x_1x_2x_3 \vee x_1(!x_2 \oplus x_3).$$

3. Выполним мультиплексорную декомпозицию при $n - k = 1$, $Q_0 = 0$, $Q_1 = x_3$, $Q_2 = !x_3$, $Q_3 = 1$:

$$f_{16} = MX(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2) = \Phi_1 !x_3 \vee \Phi_2 x_3.$$

Для определения функций Φ_1 и Φ_2 запишем исходное уравнение в МФ и решим его:

$$MX(Q_0, Q_2, Q_2, Q_1; x_1, x_2) = MX(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2);$$

$$\Phi_1 = MX(0, 1, 1, 0; x_1, x_2) = x_1 \oplus x_2;$$

$$\Phi_2 = MX(0, 0, 0, 1; x_1, x_2) = x_1 x_2.$$

Таким образом, БФУ реализуется схемой из семи МЭ.

4. Так как БФУ немонотонна по всем переменным, то выполним в БФ замену $x_3 = x_4$:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= x_1 !x_2 !x_3 \vee !x_1 x_2 !x_3 \vee x_1 x_2 x_4 = \\ &= |0000 \ 1100 \ 1100 \ 0101|^T. \end{aligned}$$

Проведем мажоритарное разложение по переменной x_4 :

$$\begin{aligned} f &= \Phi_1 \# \Phi_2 \# x_4; \\ MX(0, 0, 1, 0, 1, 0, x_4, x_4; x_1, x_2, x_3) &= \\ &= MX(0, x_4, x_4, 1; \Phi_1, \Phi_2). \end{aligned}$$

Среди четырех решений этого уравнения выберем следующее:

$$\Phi_1 = MX(0, 0, 1, 0, 1, 0, 1, 1; x_1, x_2, x_3) = x_1 \# x_2 \# !x_3;$$

$$\Phi_2 = MX(0, 0, 1, 0, 1, 0, 0, 0; x_1, x_2, x_3) = (!x_1 x_2 \vee x_1 !x_2) !x_3.$$

Таким образом, после обратной замены $x_4 = x_3$ заданная БФУ реализуется схемой из шести МЭ. Учет свойства $\Phi_2 \leq \Phi_1$ в данном случае усложняет реализацию.

5. Разложение Риды по переменной x_1 строит БФ $f = x_2 !x_3 \oplus (x_2 \vee !x_3) x_1$, которая реализуется схемой из шести МЭ.

6. Для проведения мажоритарной декомпозиции при $n - k = 0$ запишем уравнение

$$\begin{aligned} f_{16} &= \Phi_1 \# \Phi_2 \# \Phi_3; \\ MX(0, 0, 1, 0, 1, 0, 0, 1; x_1, x_2, x_3) &= \\ &= MX(0, 0, 0, 1, 0, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3). \end{aligned}$$

Это уравнение имеет 4^8 решений, среди которых имеется следующее:

$$\begin{aligned}\Phi_1 &= \text{MX}(0, 1, 1, 1, 0, 0, 0, 1; x_1, x_2, x_3) = !x_1 \# x_2 \# x_3; \\ \Phi_2 &= \text{MX}(0, 0, 1, 0, 1, 0, 1, 1; x_1, x_2, x_3) = x_1 \# x_2 \# !x_3; \\ \Phi_3 &= \text{MX}(0, 0, 1, 0, 1, 1, 0, 0; x_1, x_2, x_3) = x_1 \# !x_2 \# 0.\end{aligned}$$

Это решение, которому соответствует схема из четырех МЭ, взято из [15], где эта БФУ реализована геометрическим методом [5].

Сформулируем выводы по материалу настоящего раздела.

1. Для формул f_2, f_3, f_4, f_5 , неповторных в базисе $\{\&, \vee, !\}$, простейшие схемы строят, например, формульный и графовый методы.

2. Простейшие схемы для функций f_6, f_7 построены с помощью мультиплексорного разложения по переменной.

3. Функция f_8 имеет простейшую реализацию при применении мажоритарного разложения по одной переменной, проводимого после увеличения числа переменных в функции.

4. Формула f_9 , повторная в базисе $\{\&, \vee, !\}$, имеет простейшую реализацию при использовании мажоритарного разложения по одной переменной и формульного или графового метода.

5. Функция f_{10} имеет простейшую реализацию при применении мультиплексорной декомпозиции и разложения Риде.

6. Формула f_{11} реализуется простейшими схемами с помощью формульного или графового метода и на основе мультиплексорного разложения.

7. Формула f_{12} , повторная в базисе $\{\&, \vee, !\}$, реализуется формульным или графовым методом простейшей схемой.

8. Функции f_{13}, f_{14} реализуются простейшими схемами на основе преобразования Артюхова—Шалыто.

9. Функция f_{15} реализуется простейшей схемой с помощью эвристической декомпозиции, формульного и графового методов.

10. В [5] утверждается, что произвольная БФУ трех переменных реализуется схемой, содержащей не более четырех МЭ. Из всех рассмотренных БФУ только функция f_{16} реализуется с помощью предлагаемых подходов схемой, содержащей более четырех МЭ.

4.1.10. Примеры реализации булевых функций четырех переменных

Пример 4.27. Реализовать БФУ

$$\begin{aligned}f(x_1, x_2, x_3, x_4) &= |0000 \ 0001 \ 0111 \ 1111|^T = \\ &= x_1(x_2 \vee x_3 \vee x_4) \vee x_2x_3x_4,\end{aligned}$$

которая является порождающей функцией модуля, универсального в классе пороговых функций трех переменных.

Столбец значений этой БФУ содержит только разрешенные фрагменты длиной два, и поэтому

$$\begin{aligned} x_1(x_2 \vee x_3 \vee x_4) \vee x_2 x_3 x_4 &= \Phi_1 \# \Phi_2 \# x_4; \\ \text{MX}(0, 0, 0, x_4, x_4, 1, 1, 1; x_1, x_2, x_3) &= \\ &= \text{MX}(0, x_4, x_4, 1; \Phi_1, \Phi_2). \end{aligned}$$

Среди четырех решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, 0, 1, 0, 1, 1, 1; x_1, x_2, x_3) = x_1 \# x_2 \# x_3; \\ \Phi_2 &= \text{MX}(0, 0, 0, 0, 1, 1, 1, 1; x_1, x_2, x_3) = x_1. \end{aligned}$$

Следовательно, $\Phi = x_1 \# x_2 \# x_3$, $f = x_1 \# \Phi \# x_4$.

Таким образом, заданная БФУ реализуется на основе мультиплексорного разложения минимальной схемой из двух МЭ (рис. 4.16), в то время как формульный и графовый методы строят схемы из шести МЭ.

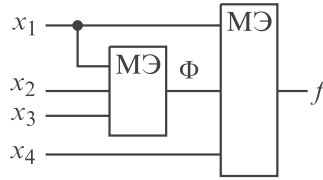


Рис. 4.16

Эта схема является основой для построения параллельного одноразрядного двоичного сумматора при его описании системой БФ:

$$P = x_1 \# x_2 \# x_3, \quad S = !P(x_1 \vee x_2 \vee x_3) \vee x_1 x_2 x_3,$$

так как

$$P = x_1 \# x_2 \# x_3, \quad \Phi = !P \# x_1 \# x_2, \quad S = !P \# \Phi \# x_3.$$

Таким образом, такой сумматор реализуется схемой из трех МЭ, один из которых имеет прямой и инверсный выходы [20].

Пример 4.28. Реализовать БФУ [13]

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= |1100 \ 0100 \ 1101 \ 1100|^T = \\ &= x_1!x_3 \vee !x_3 x_4 \vee !x_2!x_3 \vee x_1!x_2 x_4 = (x_1 \vee !x_2 \vee x_4)!x_3 \vee x_1!x_2 x_4. \end{aligned}$$

1. Так как для последней БФ $h = 7$, то формульный и графовый методы строят схемы из шести МЭ.

2. Использование преобразований Кона—Линдемана для одних переменных строят схему из семи МЭ [13], а для других переменных — схему из четырех МЭ [15].

3. Так как заданная БФУ монотонна по переменной x_4 , выполним мажоритарное разложение по этой переменной:

$$\begin{aligned}(x_1 \vee !x_2 \vee x_4)!x_3 \vee x_1!x_2x_4 &= \Phi_1 \# \Phi_2 \# x_4; \\ \text{MX}(1, 0, x_4, 0, 1, x_4, 1, 0; x_1, x_2, x_3) &= \\ &= \text{MX}(0, x_4, x_4, 1; \Phi_1, \Phi_2).\end{aligned}$$

Среди четырех решений этого уравнения выберем два разнотипных:

$$\begin{cases} \Phi_1 = \text{MX}(1, 0, 0, 0, 1, 0, 1, 0; x_1, x_2, x_3) = (x_1 \vee !x_2)!x_3; \\ \Phi_2 = \text{MX}(1, 0, 1, 0, 1, 1, 1, 0; x_1, x_2, x_3) = x_1!x_2 \vee !x_3; \\ \Phi_1 = \text{MX}(1, 0, 1, 0, 1, 0, 1, 0; x_1, x_2, x_3) = !x_3; \\ \Phi_2 = \text{MX}(1, 0, 0, 0, 1, 1, 1, 0; x_1, x_2, x_3) = x_1 \# !x_2 \# !x_3. \end{cases}$$

Первое решение совпадает с разложением Акерса [3] по переменной x_4 и соответствует схеме из пяти МЭ [13], а второе — приводит к построению схемы из двух МЭ.

4. Весьма сложный метод Акерса [4], учитывающий не только монотонность заданной БФУ, но и ее самодвойственность, также строит схему из двух МЭ.

Пример 4.29. Реализовать БФУ

$$\begin{aligned}f(x_1, x_2, x_3, x_4) &= |0010 \ 0010 \ 0010 \ 1011|^T = \\ &= x_1x_2(x_3 \vee !x_4) \vee x_3!x_4.\end{aligned}$$

1. Так как заданная БФУ монотонна по переменной $!x_4$, выполним мажоритарное разложение по этой переменной:

$$\begin{aligned}x_1x_2(x_3 \vee !x_4) \vee x_3!x_4 &= \Phi_1 \# \Phi_2 \# !x_4; \\ \text{MX}(0, !x_4, 0, !x_4, 0, !x_4, !x_4, 1; x_1, x_2, x_3) &= \\ &= \text{MX}(0, !x_4, !x_4, 1; \Phi_1, \Phi_2).\end{aligned}$$

Среди 16 решений этого уравнения выберем следующее:

$$\begin{aligned}\Phi_1 &= \text{MX}(0, 0, 0, 1, 0, 1, 1, 1; x_1, x_2, x_3) = x_1 \# x_2 \# x_3; \\ \Phi_2 &= \text{MX}(0, 1, 0, 0, 0, 0, 0, 1; x_1, x_2, x_3) = (!x_1 \oplus x_2)x_3.\end{aligned}$$

Так как функция Φ_2 реализуется подсхемой из четырех МЭ, то БФУ реализуется схемой из шести МЭ.

2. Так как для заданной булевой формулы $h = 6$, то формульный и графовый методы строят схемы из пяти МЭ.

3. Используем мультиплексорную декомпозицию при $n - k = 2$, $Q_0 = x_3!x_4$, $Q_1 = x_3 \vee !x_4$:

$$\begin{aligned} x_1 x_2 (x_3 \vee !x_4) \vee x_3!x_4 &= \text{MX}(Q_0, Q_1; \Phi); \\ \text{MX}(Q_0, Q_0, Q_0, Q_1; x_1, x_2) &= \text{MX}(Q_0, Q_1; \Phi); \\ \Phi &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2. \end{aligned}$$

Так как $Q_0 \leq Q_1$, то

$$f = Q_0 \vee Q_1 \Phi = \Phi \# x_3 \# !x_4,$$

и поэтому заданная БФУ реализуется схемой из двух МЭ.

Пример 4.30. Реализовать БФ [10]

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= x_1 x_2 \vee x_1 x_3 x_4 \vee x_2 x_3!x_4 = \\ &= x_1 (x_2 \vee x_3 x_4) \vee x_2 x_3!x_4. \end{aligned}$$

Так как заданная БФУ монотонна по переменным x_1, x_2, x_3 , то построим булеву функцию $f(x_4, x_3, x_2, x_1) = |0001 \ 0011 \ 0001 \ 0101|^T$.

1. Так как для скобочной булевой формулы $h = 7$, то формульный и графовый методы строят схемы из шести МЭ.

2. Выполним мажоритарное разложение по переменной x_1 :

$$\begin{aligned} x_1 (x_2 \vee x_3 x_4) \vee x_2 x_3!x_4 &= \Phi_1 \# \Phi_2 \# x_1; \\ \text{MX}(0, x_1, 0, 1, 0, x_1, x_1, x_1; x_4, x_3, x_2) &= \\ &= \text{MX}(0, x_1, x_1, 1; \Phi_1, \Phi_2). \end{aligned}$$

Среди 16 решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, 0, 1, 0, 1, 1, 1; x_4, x_3, x_2) = x_2 \# x_3 \# x_4; \\ \Phi_2 &= \text{MX}(0, 1, 0, 1, 0, 0, 0, 0; x_4, x_3, x_2) = x_2!x_4. \end{aligned}$$

Таким образом, БФУ реализуется схемой из трех МЭ.

3. В [10] схема из трех МЭ построена с помощью весьма трудоемкого метода решения 16 логических уравнений, соответствующих простой мажоритарной декомпозиции.

Пример 4.31. Реализовать БФУ [10]

$$\begin{aligned} f(x_4, x_3, x_2, x_1) &= |0101 \ 0100 \ 1100 \ 0100|^T = \\ &= x_1 (!x_2 \vee !x_3!x_4) \vee !x_2!x_3 x_4. \end{aligned}$$

1. Так как для булевой формулы $h = 7$, то формульный и графовый методы строят схемы из шести МЭ.

2. Так как БФУ монотонна по переменной x_1 , выполним мажоритарное разложение по этой переменной:

$$\begin{aligned} x_1 (!x_2 \vee !x_3 !x_4) \vee !x_2 !x_3 x_4 &= \Phi_1 \# \Phi_2 \# x_1; \\ \text{MX}(x_1, x_1, x_1, 0, 1, 0, x_1, 0; x_4, x_3, x_2) &= \\ &= \text{MX}(0, x_1, x_1, 1; \Phi_1, \Phi_2). \end{aligned}$$

Среди 16 решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(1, 1, 1, 0, 1, 0, 0, 0; x_4, x_3, x_2) = !x_2 \# !x_3 \# !x_4; \\ \Phi_2 &= \text{MX}(0, 0, 0, 0, 1, 0, 1, 0; x_4, x_3, x_2) = !x_2 x_4. \end{aligned}$$

Таким образом, БФУ реализуется схемой из трех МЭ.

3. В [10] схема из трех МЭ построена с помощью весьма трудоемкого метода Акерса [4].

Пример 4.32. Реализовать БФ [18]

$$\begin{aligned} f &= x_1 x_2 x_3 \vee !x_2 x_3 x_4 \vee x_1 !x_2 !x_3 \vee x_2 !x_3 x_4 = \\ &= x_1 (!x_2 !x_3 \vee x_2 x_3) \vee (!x_2 x_3 \vee x_2 !x_3) x_4. \end{aligned}$$

Так как заданная БФУ монотонна по переменным x_1 и x_4 , то построим БФУ

$$f(x_2, x_3, x_1, x_4) = |0011 \ 0101 \ 0101 \ 0011|^T.$$

1. Так как для скобочной булевой формулы $h = 10$, то формульный и графовый методы строят схемы из девяти МЭ.

2. Выполним мультиплексорную декомпозицию при $n - k = 2$, $Q_0 = x_1$, $Q_1 = x_4$:

$$f(x_2, x_3, x_1, x_4) = \text{MX}(Q_0, Q_1; \Phi) = x_1 !\Phi \vee x_4 \Phi.$$

Для определения функции Φ запишем и решим уравнение в МФ:

$$\begin{aligned} \text{MX}(Q_0, Q_1, Q_1, Q_0; x_2, x_3) &= \text{MX}(Q_0, Q_1; \Phi); \\ \Phi &= \text{MX}(0, 1, 1, 0; x_2, x_3) = x_2 \oplus x_3. \end{aligned}$$

Таким образом, в этом случае строится схема из шести МЭ, один из которых должен иметь прямой и инверсный выходы.

3. В [18], используя весьма трудоемкий метод решения 16 логических уравнений, построена простая мажоритарная декомпозиция, реализуемая схемой из четырех МЭ:

$$f = (x_1 \# !x_2 \# x_3) \# (x_2 \# x_3 \# x_4) \# (x_1 \# !x_3 \# x_4).$$

4. Выполним мажоритарное разложение по переменной x_4 :

$$\begin{aligned} x_1 (!x_2 !x_3 \vee x_2 x_3) \vee (!x_2 x_3 \vee x_2 !x_3) x_4 &= \Phi_1 \# \Phi_2 \# x_4; \\ \text{MX}(0, 1, x_4, x_4, x_4, x_4, 0, 1; x_2, x_3, x_1) &= \\ &= \text{MX}(0, x_4, x_4, 1; \Phi_1, \Phi_2). \end{aligned}$$

Среди 16 решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 1, 1, 1, 0, 0, 0, 1; x_2, x_3, x_1) = !x_1 \# x_2 \# x_3; \\ \Phi_2 &= \text{MX}(0, 1, 0, 0, 1, 1, 0, 1; x_2, x_3, x_1) = x_1 \# !x_2 \# x_3. \end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из трех МЭ.

Пример 4.33. Реализовать БФ [15]

$$f = x_2 x_3 \vee x_3 !x_4 \vee x_1 x_2 !x_4 \vee x_1 !x_3 x_4.$$

Из рассмотрения этой тупиковой ДНФ кажется, что БФУ немо-
нотонна по переменной x_4 или ее инверсии, однако если построить
ТИ

$$f(x_1, x_2, x_3, x_4) = |0010 \ 0011 \ 0011 \ 1111|^T,$$

то из ее рассмотрения следует, что БФУ монотонна по переменной
 $!x_4$. При этом $f = x_1 (x_2 \vee x_3) \vee (x_2 \vee !x_4) x_3$.

1. Так как для скобочной булевой формулы $h = 6$, то формуль-
ный и графовый методы строят схемы из пяти МЭ.

2. Выполним мажоритарное разложение по переменной $!x_4$:

$$\begin{aligned} x_1 (x_2 \vee x_3) \vee (x_2 \vee !x_4) x_3 &= \Phi_1 \# \Phi_2 \# !x_4; \\ \text{MX}(0, !x_4, 0, 1, 0, 1, 1, 1; x_1, x_2, x_3) &= \\ &= \text{MX}(0, !x_4, !x_4, 1; \Phi_1, \Phi_2). \end{aligned}$$

Среди двух решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, 0, 1, 0, 1, 1, 1; x_1, x_2, x_3) = x_1 \# x_2 \# x_3; \\ \Phi_2 &= \text{MX}(0, 1, 0, 1, 0, 1, 1, 1; x_1, x_2, x_3) = x_1 x_2 \vee x_3. \end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из четырех
МЭ.

В данном случае $\Phi_1 \leq \Phi_2$, и поэтому $\Phi_2 = \Phi_1 \vee !x_1 !x_2 x_3$. Однако
такая реализация усложняет схему.

3. Схема из четырех МЭ была построена также и в [15] с использованием весьма специфической универсальной декомпозиции [12]:

$$\begin{aligned} f(x_1, x_2, Y) &= f_1(x_1, Y) \# f_2(x_2, Y) \# f_3(x_1 \oplus x_2, Y) = \\ &= (x_1 \vee x_3) \# (x_2 \# x_3 \# !x_4) \# !x_3 x_4. \end{aligned}$$

Пример 4.34. Реализовать БФУ

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= |0001 \ 0000 \ 0001 \ 1111|^T = \\ &= (x_1 \vee !x_2) x_3 x_4 \vee x_1 x_2. \end{aligned}$$

1. Так как $h = 6$, то формульный и графовый методы строят схемы из пяти МЭ.

2. Выполним мажоритарное разложение по переменной x_4 :

$$\begin{aligned} (x_1 \vee !x_2) x_3 x_4 \vee x_1 x_2 &= \Phi_1 \# \Phi_2 \# x_4; \\ \text{MX}(0, x_4, 0, 0, 0, x_4, 1, 1; x_1, x_2, x_3) &= \\ &= \text{MX}(0, x_4, x_4, 1; \Phi_1, \Phi_2). \end{aligned}$$

Среди четырех решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 1, 0, 0, 0, 1, 1, 1; x_1, x_2, x_3) = x_1 x_2 \vee !x_2 x_3; \\ \Phi_2 &= \text{MX}(0, 0, 0, 0, 0, 0, 1, 1; x_1, x_2, x_3) = x_1 x_2. \end{aligned}$$

В данном случае $\Phi_2 \leq \Phi_1$, и поэтому $\Phi_1 = \Phi_2 \vee !x_2 x_3$.

Таким образом, заданная БФУ реализуется схемой из четырех МЭ.

3. Выполним мажоритарное разложение по функции $Q = x_3 x_4$:

$$\begin{aligned} (x_1 \vee !x_2) x_3 x_4 \vee x_1 x_2 &= \Phi_1 \# \Phi_2 \# Q; \\ \text{MX}(Q, 0, Q, 1; x_1, x_2) &= \text{MX}(0, Q, Q, 1; \Phi_1, \Phi_2). \end{aligned}$$

Среди четырех решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2; \\ \Phi_2 &= \text{MX}(1, 0, 1, 1; x_1, x_2) = x_1 \vee !x_2. \end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из четырех МЭ. Учет того, что в данном случае $\Phi_1 \leq \Phi_2$ и $\Phi_2 = \Phi_1 \vee !x_2$, сложность схемы не изменяет.

Пример 4.35. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0000 \ 0110 \ 0110 \ 1111|^T = \\ = x_1 x_2 \vee (x_1 \vee x_2) (!x_3 x_4 \vee x_3 !x_4).$$

1. Так как $h = 8$, то формульный и графовый методы строят схемы из семи МЭ.

2. Выполним мажоритарное разложение по функции $Q = x_3 \oplus x_4$:

$$x_1 x_2 \vee (x_1 \vee x_2) (!x_3 x_4 \vee x_3 !x_4) = \Phi_1 \# \Phi_2 \# Q; \\ MX(0, Q, Q, 1; x_1, x_2) = MX(0, Q, Q, 1; \Phi_1, \Phi_2); \\ \Phi_1 = x_1; \Phi_2 = x_2.$$

Таким образом, заданная БФУ реализуется схемой из четырех МЭ.

3. Так как БФ монотонна по переменным x_1 и x_2 , то построим

$$f(x_4, x_3, x_2, x_1) = |0001 \ 0111 \ 0111 \ 0001|^T.$$

Выполним мажоритарное разложение этой функции по переменным x_1 и x_2 :

$$x_1 x_2 \vee (x_1 \vee x_2) (!x_3 x_4 \vee x_3 !x_4) = \Phi \# x_1 \# x_2; \\ MX(x_1 x_2, x_1 \vee x_2, x_1 \vee x_2, x_1 x_2; x_4, x_3) = \\ = MX(x_1 x_2, x_1 \vee x_2; \Phi); \\ \Phi = MX(0, 1, 1, 0; x_4, x_3) = x_3 \oplus x_4.$$

Таким образом, заданная БФУ реализуется схемой из четырех МЭ.

4. Выполним мультиплексорную декомпозицию при $n - k = 2$, $Q_0 = x_1 x_2$, $Q_1 = x_1 \vee x_2$:

$$f(x_4, x_3, x_2, x_1) = MX(Q_0, Q_1; \Phi).$$

Так как $Q_0 \leq Q_1$, то $f = Q_0 \vee Q_1 \Phi = \Phi \# x_1 \# x_2$.

Для определения функции Φ запишем и решим уравнение в МФ:

$$MX(Q_0, Q_1, Q_1, Q_0; x_4, x_3) = MX(Q_0, Q_1; \Phi); \\ \Phi = MX(0, 1, 1, 0; x_4, x_3) = x_3 \oplus x_4.$$

Таким образом, заданная БФУ реализуется схемой из четырех МЭ.

Пример 4.36. Реализовать БФУ [9]

$$f = !x_1 !x_2 !x_4 \vee !x_3 x_4 \vee x_3 !x_4 = (!x_1 !x_2 \vee x_3) !x_4 \vee !x_3 x_4.$$

Построим таблицу истинности для БФУ

$$f(x_2, x_1, x_4, x_3) = |1110 \ 0110 \ 0110 \ 0110|^T.$$

1. Выполним мультиплексорную декомпозицию при $n - k = 2$,
 $Q_0 = x_3 \oplus x_4$, $Q_1 = !x_3 \vee !x_4$:

$$(!x_1!x_2 \vee x_3)!x_4 \vee !x_3x_4 = MX(Q_0, Q_1; \Phi).$$

Так как $Q_0 \leq Q_1$, то $f = Q_0 \vee Q_1 \Phi = !x_3(x_4 \vee \Phi) \vee !x_4(x_3 \vee \Phi)$.
 Для определения функции Φ запишем и решим уравнение в МФ:

$$\begin{aligned} MX(Q_1, Q_0, Q_0, Q_0; x_2, x_1) &= MX(Q_0, Q_1; \Phi); \\ \Phi &= MX(1, 0, 0, 0; x_2, x_1) = !x_1!x_2. \end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из шести МЭ.

2. Так как БФ монотонна по переменной $!x_2$, то построим

$$f(x_3, x_4, x_1, x_2) = |1000 \ 1111 \ 1111 \ 0000|^T.$$

Выполним мажоритарное разложение этой функции по переменной $!x_2$:

$$\begin{aligned} (!x_1!x_2 \vee x_3)!x_4 \vee !x_3x_4 &= \Phi_1 \# \Phi_2 \# !x_2; \\ MX(!x_2, 0, 1, 1, 1, 0, 0; x_3, x_4, x_1) &= \\ &= MX(0, !x_2, !x_2, 1; \Phi_1, \Phi_2); \\ \Phi_1 &= MX(0, 0, 1, 1, 1, 0, 0; x_3, x_4, x_1) = x_3 \oplus x_4; \\ \Phi_2 &= MX(1, 0, 1, 1, 1, 0, 0; x_3, x_4, x_1). \end{aligned}$$

Так как $\Phi_1 \leq \Phi_2$, то $\Phi_2 = \Phi_1 \vee !x_1!x_3!x_4$. Таким образом, функции Φ_1 и Φ_2 совместно реализуются подсхемой из шести МЭ.

Определим более простую их реализацию:

$$\begin{aligned} \Phi_2(x_3, x_4, x_1) &= \Phi_{21} \# \Phi_{22} \# !x_1; \\ MX(!x_1, 1, 1, 0; x_3, x_4) &= MX(0, !x_1, !x_1, 1; \Phi_{21}, \Phi_{22}); \\ \Phi_{21} &= MX(0, 1, 1, 0; x_3, x_4) = x_3 \oplus x_4; \\ \Phi_{22} &= MX(1, 1, 1, 0; x_3, x_4) = !x_3 \vee !x_4. \end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из шести МЭ.

3. Выполним мажоритарное разложение по функции $Q = !x_1!x_2$:

$$\begin{aligned} (!x_1!x_2 \vee x_3)!x_4 \vee !x_3x_4 &= \Phi_1 \# \Phi_2 \# Q; \\ MX(Q, 1, 1, 0; x_3, x_4) &= MX(0, Q, Q, 1; \Phi_1, \Phi_2); \\ \Phi_1 &= MX(0, 1, 1, 0; x_3, x_4) = x_3 \oplus x_4; \\ \Phi_2 &= MX(1, 1, 1, 0; x_3, x_4) = !x_3 \vee !x_4. \end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из шести МЭ. Учет свойства $\Phi_1 \leq \Phi_2$ в данном случае усложняет реализацию.

4. Так как $h = 6$, то формульный и графовый методы строят схемы из пяти МЭ.

5. В [9] эта БФУ реализуется четырьмя разными методами — схемами из 11, 9, 7 и 5 МЭ.

Пример 4.37. Реализовать БФУ [13]

$$\begin{aligned} f &= x_1!x_3x_4 \vee x_2!x_3x_4 \vee!x_2!x_3!x_4 \vee!x_2x_3x_4 \vee x_1x_2x_3!x_4 = \\ &= ((x_1 \vee x_2)!x_3 \vee!x_2x_3)x_4 \vee (x_1x_2x_3 \vee!x_2!x_3)!x_4. \end{aligned}$$

Так как БФУ монотонна по переменной x_1 , то построим ТИ для БФУ в виде:

$$f(x_2, x_4, x_3, x_1) = |1101 \ 0011 \ 0011 \ 0100|^T.$$

1. Так как $h = 12$, то формульный и графовый методы строят схемы из 11 МЭ.

2. При использовании разложения фиксации переменных константами могут быть построены схемы, содержащие в зависимости от порядка разложения семь и девять МЭ [13, 15].

3. Применение одного из преобразований Кона—Линдемана строит схему из семи МЭ [13].

4. Выполним мажоритарное разложение этой функции по переменной x_1 :

$$\begin{aligned} &((x_1 \vee x_2)!x_3 \vee!x_2x_3)x_4 \vee (x_1x_2x_3 \vee!x_2!x_3)!x_4 = \\ &= \Phi_1 \# \Phi_2 \# x_1; \\ &MX(1, x_1, 0, 1, 0, 1, x_1, 0; x_2, x_3, x_4) = \\ &= MX(0, x_1, x_1, 1; \Phi_1, \Phi_2). \end{aligned}$$

Это уравнение имеет четыре решения, из которых два разнотипны:

$$\begin{cases} \Phi_1 = MX(1, 0, 0, 1, 0, 1, 0, 0; x_2, x_3, x_4) = \\ \quad =!x_2(!x_3 \oplus x_4) \vee x_2!x_3x_4; \\ \Phi_2 = MX(1, 1, 0, 1, 0, 1, 1, 0; x_2, x_3, x_4) = \\ \quad =!x_2 \# !x_3 \# x_4 \vee x_2x_3!x_4; \\ \Phi_1 = MX(1, 1, 0, 1, 0, 1, 0, 0; x_2, x_3, x_4) =!x_2 \# !x_3 \# x_4; \\ \Phi_2 = MX(1, 0, 0, 1, 0, 1, 1, 0; x_2, x_3, x_4) = x_2 \oplus x_3 \oplus!x_4. \end{cases}$$

Первое решение соответствует разложению Акерса и является весьма сложным. Второе решение существенно проще и обеспечивает реализацию БФУ схемой из пяти МЭ.

5. Весьма трудоемкий метод Акерса [13] также строит схему из пяти МЭ.

Пример 4.38. Реализовать БФУ [18]

$$\begin{aligned} f &= x_1 x_2 x_3 \vee !x_1 !x_2 !x_3 \vee !x_1 !x_3 x_4 \vee !x_2 !x_3 x_4 = \\ &= x_1 x_2 x_3 \vee !x_3 (!x_1 (!x_2 \vee x_4) \vee !x_2 x_4). \end{aligned}$$

1. Эта БФУ монотонна по переменной x_4 . Многократное использование мажоритарного разложения по отдельным переменным, начиная с переменной x_4 , в этом случае крайне неэффективно, так как получается схема из 11 МЭ.

2. Так как $h = 9$, то формульный и графовый методы строят схемы из восьми МЭ.

3. Если мажоритарное разложение применить однократно, то может быть построена схема из семи МЭ, так как $f_1 = x_1 x_2 x_3$, $\Phi_1 = !x_1 !x_3 \vee f_1$, $\Phi_2 = !x_2 !x_3 \vee f_1$, $f = \Phi_1 \# \Phi_2 \# x_4$.

4. Используя эвристическую декомпозицию $f_1 = !x_1 \# !x_2 \# x_4$, $f = x_1 x_2 x_3 \vee !x_3 f_1$, может быть построена схема из пяти МЭ.

5. Эта БФУ в [18] после выбора порядка исключения переменных реализуется с помощью разложения Шеннона схемой из пяти МЭ.

Пример 4.39. Реализовать БФУ

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= |0110 \ 0000 \ 0110 \ 1111|^T = \\ &= x_1 x_2 \vee !x_2 (!x_3 x_4 \vee x_3 !x_4). \end{aligned}$$

1. Так как $h = 7$, то формульный и графовый методы строят схемы из шести МЭ.

2. Выполним мультиплексорную декомпозицию при $n - k = 2$, полагая $Q_0 = Q_1 = 0$, $Q_2 = x_3 \oplus x_4$, $Q_3 = 1$:

$$f = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2) = \Phi_1(Q_2 \vee \Phi_2).$$

Для определения функций Φ_1 и Φ_2 запишем и решим уравнение в МФ:

$$\begin{aligned} \text{MX}(Q_2, Q_0 \ ? \ Q_1, Q_2, Q_3; x_2, x_3) &= \\ &= \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2). \end{aligned}$$

Из двух решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(1, 0, 1, 1; x_1, x_2) = x_1 \vee !x_2; \\ \Phi_2 &= \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2. \end{aligned}$$

Полученная декомпозиция реализуется схемой из шести МЭ.

3. Из рассмотрения БФ следует, что она монотонна по переменной x_1 :

$$f(x_2, x_3, x_4, x_1) = |0011 \ 1100 \ 0101 \ 0101|^T.$$

Выполним мажоритарное разложение функции по этой переменной:

$$\begin{aligned} x_1 x_2 \vee !x_2 (!x_3 x_4 \vee x_3 !x_4) &= \Phi_1 \# \Phi_2 \# x_1; \\ \text{MX}(0, 1, 1, 0, x_1, x_1, x_1, x_1; x_2, x_3, x_4) &= \\ &= \text{MX}(0, x_1, x_1, 1; \Phi_1, \Phi_2). \end{aligned}$$

Это уравнение имеет 16 решений, среди которых имеется следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 1, 1, 0, 0, 1, 1, 0; x_2, x_3, x_4) = x_3 \oplus x_4; \\ \Phi_2 &= \text{MX}(0, 1, 1, 0, 1, 0, 0, 1; x_2, x_3, x_4) = x_2 \oplus x_3 \oplus x_4. \end{aligned}$$

Так как эта пара функций реализуется подсхемой из шести МЭ, то выберем другое решение:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 1, 1, 0, 1, 1, 1, 1; x_2, x_3, x_4); \\ \Phi_2 &= \text{MX}(0, 1, 1, 0, 0, 0, 0, 0; x_2, x_3, x_4) = !x_2 (x_3 \oplus x_4). \end{aligned}$$

Ввиду того что $\Phi_2 \leq \Phi_1$ и $\Phi_1 = \Phi_2 \vee x_2$, то подсхема в этом случае состоит из пяти МЭ, а БФУ в целом реализуется схемой из шести МЭ.

4. Выполним мажоритарное разложение по функции $Q = x_3 \oplus x_4$:

$$\begin{aligned} x_1 x_2 \vee !x_2 (!x_3 x_4 \vee x_3 !x_4) &= \Phi_1 \# \Phi_2 \# Q; \\ \text{MX}(Q, 0, Q, 1; x_1, x_2) &= \text{MX}(0, Q, Q, 1; \Phi_1, \Phi_2). \end{aligned}$$

Среди четырех решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2; \\ \Phi_2 &= \text{MX}(1, 0, 1, 1; x_1, x_2) = x_1 \vee !x_2. \end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из шести МЭ. Учет свойства $\Phi_1 \leq \Phi_2$ в данном случае схему не упрощает.

Сформулируем выводы по материалу настоящего раздела.

1. В примерах 4.27, 4.28, 4.30, 4.31, 4.32, 4.33, 4.37 мультиплексорное разложение по отдельным переменным строит простейшие схемы.

2. В примере 4.34 мультиплексорное разложение по переменной или по подфункции строит простейшую схему.

3. В примере 4.29 мультиплексорная декомпозиция строит простейшую схему.

4. В примерах 4.35, 4.39 все три перечисленных выше подхода строят схемы одинаковой сложности.

5. В примере 4.36 для БФ, повторной в базисе $\{\&, \vee, !\}$, простейшие схемы строят формульный и графовый методы.

6. В примере 4.38 простейшие схемы строят эвристическая декомпозиция, формульный и графовый методы.

4.1.11. Примеры реализации булевых функций пяти переменных

Пример 4.40. Реализовать БФУ [10]

$$f(x_1, x_2, x_3, x_4, x_5) = |0001\ 0001\ 0001\ 0111\ 0001\ 0111\ 0111\ 0111|^T.$$

1. Выполним разложение БФУ одновременно по двум переменным x_4 и x_5 :

$$\begin{aligned} f(x_1, x_2, x_3, x_4, x_5) &= \Phi_1 \# x_4 \# x_5; \\ \text{MX}(x_4 x_5, x_4 x_5, x_4 x_5, x_4 \vee x_5, x_4 x_5, x_4 \vee x_5, x_4 \vee x_5, x_4 \vee \\ &\vee x_5; x_1, x_2, x_3) = \text{MX}(x_4 x_5, x_4 \vee x_5; \Phi); \\ \Phi &= \text{MX}(0, 0, 0, 1, 0, 1, 1, 1; x_1, x_2, x_3) = x_1 \# x_2 \# x_3. \end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из двух МЭ.

2. В [10] эта схема построена с помощью весьма трудоемкого метода Акерса [4].

Пример 4.41. Реализовать БФУ [15]

$$\begin{aligned} f(x_2, x_1, x_5, x_4, x_3) &= \\ &= |1100\ 0000\ 0010\ 1011\ 0010\ 1011\ 0010\ 1011|^T = \\ &= (x_1 \vee x_2) ((!x_3 \vee x_4) x_5 \vee !x_3 x_4) \vee !x_1 !x_2 !x_4 !x_5. \end{aligned}$$

1. Эта БФУ монотонна по переменной $!x_3$. Проведение многократного разложения, начиная с этой переменной, крайне неэффективно, так как при этом строится схема из 11 МЭ.

2. Так как $h = 11$, то формульный и графовый методы строят схемы из десяти МЭ.

3. Схема из семи МЭ строится по БФ

$$f = (x_1 \vee x_2) (!x_3 \# !x_4 \# x_5) \vee !x_1 !x_2 !x_3 !x_5.$$

4. Выполним мультиплексорную декомпозицию при $n - k = 3$,
 $Q_0 = !x_4 !x_5$, $Q_1 = !x_3 \# x_4 \# x_5$:

$$f = MX(Q_0, Q_1; \Phi) = Q_0 !\Phi \vee Q_1 \Phi.$$

Для определения функции Φ запишем и решим уравнение в МФ:

$$\begin{aligned} MX(Q_0, Q_1, Q_1, Q_1; x_2, x_1) &= MX(Q_0, Q_1; \Phi); \\ \Phi &= MX(0, 1, 1, 1; x_2, x_1) = x_1 \vee x_2. \end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из шести МЭ, из которых один должен иметь прямой и инверсный выходы.

5. В [15] эта схема была получена при использовании функциональной декомпозиции.

Пример 4.42. Реализовать БФ [15]

$$f = (!x_1 \vee !x_3)(x_2 \vee !x_4)x_5 \vee x_2 !x_4.$$

Построим таблицу истинности по БФ

$$\begin{aligned} f(x_2, x_4, x_5, x_1, x_3) &= \\ &= | 0000 \ 1110 \ 0000 \ 0000 \ 1111 \ 1111 \ 0000 \ 1110 |^T. \end{aligned}$$

1. Так как $h = 7$, то формульный и графовый методы строят схемы из шести МЭ.

2. Выполним мажоритарное разложение по функции $Q = !x_1 \vee !x_3$:

$$\begin{aligned} (!x_1 \vee !x_3)(x_2 \vee !x_4)x_5 \vee x_2 !x_4 &= \Phi_1 \# \Phi_2 \# Q; \\ MX(0, Q, 0, 0, 1, 1, 0, Q; x_2, x_4, x_5) &= MX(0, Q, Q, 1; \Phi_1, \Phi_2). \end{aligned}$$

Это уравнение имеет четыре решения, среди которых выберем следующее:

$$\begin{aligned} \Phi_1 &= MX(0, 0, 0, 0, 1, 1, 0, 0; x_2, x_4, x_5) = x_2 !x_4; \\ \Phi_2 &= MX(0, 1, 0, 0, 1, 1, 0, 1; x_2, x_4, x_5) = x_2 \# !x_4 \# x_5. \end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из четырех МЭ. Учет свойства $\Phi_1 \leq \Phi_2$ в данном случае усложняет реализацию.

3. Выполним мажоритарное разложение этой функции по переменной $!x_3$:

$$\begin{aligned} (!x_1 \vee !x_3)(x_2 \vee !x_4)x_5 \vee x_2 !x_4 &= \Phi_1 \# \Phi_2 \# !x_3; \\ MX(0, 0, 1, !x_3, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, !x_3; x_2, x_4, x_5, x_1) &= \\ &= MX(0, !x_3, !x_3, 1; \Phi_1, \Phi_2). \end{aligned}$$

Это уравнение имеет четыре решения, среди которых выберем следующее:

$$\begin{aligned}\Phi_1 &= \text{MX}(0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0; x_2, x_4, x_5, x_1); \\ \Phi_2 &= \text{MX}(0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1; x_2, x_4, x_5, x_1) = \\ &= x_2 \# !x_4 \# x_5.\end{aligned}$$

Для функции Φ_1 проведем мультиплексорное разложение по функции $Q = !x_1 x_5$:

$$\begin{aligned}\Phi_1 &= \Phi_{11} \# \Phi_{12} \# Q; \\ \text{MX}(!x_1 x_5, 0, 1, !x_1 x_5; x_2, x_4) &= \\ &= \text{MX}(0, !x_1 x_5, !x_1 x_5, 1; \Phi_{11}, \Phi_{12}).\end{aligned}$$

Среди четырех решений этого уравнения выберем следующее:

$$\begin{aligned}\Phi_{11} &= \text{MX}(0, 0, 1, 1; x_2, x_4) = x_2; \\ \Phi_{12} &= \text{MX}(1, 0, 1, 0; x_2, x_4) = !x_4.\end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из четырех МЭ. Учет свойства $\Phi_1 \leq \Phi_2$ в данном случае усложняет реализацию.

4. В [16] схема из четырех МЭ построена за счет весьма сложного решения системы логических уравнений, соответствующей простой мажоритарной декомпозиции.

Пример 4.43. Реализовать БФУ [15]

$$\begin{aligned}f &= x_1 x_2 \vee x_1 x_3 \vee x_1 x_4 x_5 \vee x_2 x_3 x_4 = \\ &= x_1 (x_2 \vee x_3 \vee x_4 x_5) \vee x_2 x_3 x_4.\end{aligned}$$

Построим таблицу истинности по БФ:

$$\begin{aligned}f(x_1, x_2, x_3, x_4, x_5) &= \\ &= |0000 \ 0000 \ 0000 \ 0011 \ 0001 \ 1111 \ 1111 \ 1111|^T.\end{aligned}$$

1. Так как $h = 8$, то формульный и графовый методы строят схемы из семи МЭ.

2. Выполним многократно мажоритарное разложение по отдельным переменным:

$$\begin{aligned}x_1 (x_2 \vee x_3 \vee x_4 x_5) \vee x_2 x_3 x_4 &= \Phi_1 \# \Phi_2 \# x_5; \\ \text{MX}(0, 0, 0, 0, 0, 0, 0, 1, 0, x_5, 1, 1, 1, 1, 1; x_1, x_2, x_3, x_4) &= \\ &= \text{MX}(0, x_5, x_5, 1; \Phi_1, \Phi_2);\end{aligned}$$

$$\Phi_1 = \text{MX}(0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1; x_1, x_2, x_3, x_4);$$

$$\Phi_2 = \text{MX}(0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1; x_1, x_2, x_3, x_4).$$

$$\Phi_1 = \Phi_{11} \# \Phi_{12} \# x_4;$$

$$\text{MX}(0, 0, 0, x_4, 0, 1, 1, 1; x_1, x_2, x_3) =$$

$$= \text{MX}(0, x_4, x_4, 1; \Phi_{11}, \Phi_{12});$$

$$\Phi_{11} = \text{MX}(0, 0, 0, 0, 0, 1, 1, 1; x_1, x_2, x_3);$$

$$\Phi_{12} = \text{MX}(0, 0, 0, 1, 0, 1, 1, 1; x_1, x_2, x_3) = x_1 \# x_2 \# x_3.$$

$$\Phi_{11} = \Phi_{111} \# \Phi_{112} \# x_3;$$

$$\text{MX}(0, 0, x_3, 1; x_1, x_2) = \text{MX}(0, x_3, x_3, 1; \Phi_{111}, \Phi_{112});$$

$$\Phi_{111} = \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2;$$

$$\Phi_{112} = \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1.$$

$$\Phi_2 = \Phi_{21} \# \Phi_{22} \# x_4;$$

$$\text{MX}(0, 0, 0, x_4, x_4, 1, 1, 1; x_1, x_2, x_3) =$$

$$= \text{MX}(0, x_4, x_4, 1; \Phi_{21}, \Phi_{22});$$

$$\Phi_{21} = \text{MX}(0, 0, 0, 0, 1, 1, 1, 1; x_1, x_2, x_3) = x_1;$$

$$\Phi_{22} = \text{MX}(0, 0, 0, 1, 0, 1, 1, 1; x_1, x_2, x_3) = x_1 \# x_2 \# x_3.$$

Таким образом, заданная БФУ реализуется схемой из шести МЭ. Учет свойств $\Phi_1 \leq \Phi_2$, $\Phi_{11} \leq \Phi_{12}$, $\Phi_{111} \leq \Phi_{112}$ в данном случае усложняет реализацию.

3. Выполним мультиплексорную декомпозицию при $n - k = 2$, $Q_0 = 0$, $Q_1 = x_4 x_5$, $Q_2 = x_4$, $Q_3 = 1$:

$$f(x_1, x_2, x_3, x_4, x_5) = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2).$$

При этом $f = \Phi_1(\Phi_2 \vee x_4) \vee \Phi_2 x_4 x_5$, для которой справедливо соотношение

$$f(\Phi_1, \Phi_2, x_4, x_5) = |0000 \ 0001 \ 0011 \ 1111|^T.$$

Выполним мажоритарное разложение этой БФУ по переменной x_5 :

$$f(\Phi_1, \Phi_2, x_4, x_5) = \Phi_3 \# \Phi_4 \# x_5;$$

$$\text{MX}(0, 0, 0, x_5, 0, 1, 1, 1; \Phi_1, \Phi_2, x_4) =$$

$$= \text{MX}(0, x_5, x_5, 1; \Phi_3, \Phi_4);$$

$$\Phi_3 = \text{MX}(0, 0, 0, 0, 0, 1, 1, 1; \Phi_1, \Phi_2, x_4) = \Phi_1(\Phi_2 \vee x_4);$$

$$\Phi_4 = \text{MX}(0, 0, 0, 1, 0, 1, 1, 1; \Phi_1, \Phi_2, x_4) = \Phi_1 \# \Phi_2 \# x_4.$$

Таким образом, образ декомпозиции реализуется схемой из четырех МЭ. Для определения функций Φ_1 и Φ_2 запишем исходное уравнение в МФ и решим его:

$$\begin{aligned} & MX(Q_0, Q_0, Q_0, Q_2, Q_1, Q_3, Q_3, Q_3; x_1, x_2, x_3) = \\ & = MX(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2); \\ & \Phi_1 = MX(0, 0, 0, 1, 0, 1, 1, 1; x_1, x_2, x_3) = x_1 \# x_2 \# x_3; \\ & \Phi_2 = MX(0, 0, 0, 0, 1, 1, 1, 1; x_1, x_2, x_3) = x_1. \end{aligned}$$

Следовательно, суперпозиция $\Phi_1 = x_1 \# x_2 \# x_3$, $\Phi_2 = x_1$, $\Phi_3 = \Phi_1 (\Phi_2 \vee x_4)$, $\Phi_4 = \Phi_1 \# \Phi_2 \# x_4$, $f = \Phi_3 \# \Phi_4 \# x_5$ реализуется схемой из пяти МЭ. Верификация этой суперпозиции показывает, что найденное решение далеко от тривиального.

4. Если использовать весьма специфический метод, учитывающий, что заданная БФУ является пороговой, то она может быть реализована схемой из четырех МЭ [15].

Пример 4.44. Реализовать БФУ [14]

$$\begin{aligned} & f(x_3, x_2, x_1, x_5, x_4) = \\ & = |1101 \ 0100 \ 1111 \ 1110 \ 1000 \ 0000 \ 1101 \ 0100|^T = \\ & = ((!x_1 \vee !x_5) x_4 \vee x_2 (!x_4 \vee !x_5)) !x_3 \vee (!x_1 \vee !x_5) x_2 x_4 \vee !x_1 !x_4 !x_5. \end{aligned}$$

1. Так как $h = 14$, то формульный и графовый методы строят схемы из 13 МЭ.

2. Для суперпозиции $f_1 = !x_1 \vee !x_5$; $f = (f_1 x_4 \vee x_2 (!x_4 \vee !x_5)) !x_3 \vee f_1 x_2 x_4 \vee !x_1 !x_4 !x_5$ формульный и графовый методы строят схемы из 12 МЭ.

3. Выполним мультиплексорную декомпозицию при $n - k = 3$, $Q_0 = Q_1 = !x_1 !x_4 !x_5$, $Q_2 = !x_1 \# x_4 \# x_5$, $Q_3 = !x_1 \vee !x_4 \vee !x_5$:

$$\begin{aligned} & f = MX(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2) = Q_0 \vee \Phi_1 (Q_2 \vee Q_3 \Phi_2); \\ & MX(Q_2, Q_3, Q_0 ? Q_1, Q_2; x_3, x_2) = \\ & = MX(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2); \\ & \Phi_1 = MX(1, 1, 0, 1; x_3, x_2) = x_2 \vee !x_3; \\ & \Phi_2 = MX(0, 1, 0, 0; x_3, x_2) = x_2 !x_3. \end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из $2 + 1 + 2 + 4 + 1 + 1 = 11$ МЭ. Учет свойства $\Phi_2 \leq \Phi_1$ в данном случае усложняет реализацию.

4. В [14] приведена схема из девяти МЭ, реализующая заданную БФУ «методом последовательного исключения переменных с использованием ряда способов минимизации».

5. Из рассмотрения БФ следует, что она монотонна по переменным $!x_1, x_2, !x_3, !x_5$. Поэтому перестроим ТИ:

$$f(x_4, x_1, x_3, x_5, x_2) = \\ = |1101 \ 1100 \ 0101 \ 0000 \ 1111 \ 0101 \ 1100 \ 0100|^T.$$

Выполним мажоритарное разложение по переменной x_2 :

$$f(x_4, x_1, x_3, x_5, x_2) = \Phi_1 \# \Phi_2 \# x_2; \\ \text{MX}(1, x_2, 1, 0, x_2, x_2, 0, 0, 1, 1, x_2, x_2, 1, 0, x_2, 0; \\ x_4, x_1, x_3, x_5) = \text{MX}(0, x_2, x_2, 1; \Phi_1, \Phi_2).$$

Это уравнение имеет 64 решения:

$$\Phi_1 = \text{MX}(1, ?, 1, 0, ?, ?, 0, 0, 1, 1, ?, ?, 1, 0, ?, 0; x_4, x_1, x_3, x_5); \\ \Phi_2 = \text{MX}(1, ?, 1, 0, ?, ?, 0, 0, 1, 1, ?, ?, 1, 0, ?, 0; x_4, x_1, x_3, x_5).$$

С целью получения мажорант для функций Φ_1 и Φ_2 для придания направленности при выборе требуемого решения составим карту Карно для функции Φ_1 , которую удастся доопределить следующим образом:

$$\Phi_1 = \text{MX}(1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0; x_4, x_1, x_3, x_5) = \\ = !x_1 \# !x_3 \# !x_5.$$

При этом функция Φ_2 доопределяется автоматически:

$$\Phi_2 = \text{MX}(1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0; x_4, x_1, x_3, x_5).$$

Так как $\Phi_2 = !x_1 !x_4 !x_5 \vee x_1 !x_3 !x_4 x_5 \vee x_1 x_4 !x_5 \vee !x_1 !x_3 x_4 \vee !x_1 x_4 x_5$, то она монотонна по переменной $!x_3$, и поэтому

$$\Phi_2(x_1, x_4, x_5, x_3) = |1100 \ 1011 \ 0010 \ 1100|^T.$$

Выполним разложение этой БФУ по последней переменной:

$$\Phi_2 = \Phi_{21} \# \Phi_{22} \# !x_3; \\ \text{MX}(1, 0, !x_3, 1, 0, !x_3, 1, 0; x_1, x_4, x_5) = \\ = \text{MX}(0, !x_3, !x_3, 1; \Phi_{21}, \Phi_{22}).$$

Среди четырех решений этого уравнения выберем следующее:

$$\Phi_{21} = \text{MX}(1, 0, 1, 1, 0, 0, 1, 0; x_1, x_4, x_5) = !x_1 \# x_4 \# !x_5; \\ \Phi_{22} = \text{MX}(1, 0, 0, 1, 0, 1, 1, 0; x_1, x_4, x_5) = !x_1 \oplus x_4 \oplus x_5.$$

Таким образом, заданная БФУ реализуется схемой из семи МЭ.

6. В [14] заданная БФУ реализуется схемой из трех МЭ:

$$f = (!x_2 \# !x_3 \# !x_4) \# (!x_1 \# x_4 \# !x_5) \# x_2,$$

построенной за счет весьма трудоемкого решения системы логических уравнений, соответствующих простой мажоритарной декомпозиции

$$f = \Phi_1(x_2, x_3, x_4, x_5) \# \Phi_2(x_1, x_3, x_4, x_5) \# \Phi_3(x_1, x_2, x_4, x_5),$$

которая является неразделительной.

Пример 4.45. Реализовать БФУ [11]

$$f(x_2, x_1, x_5, x_4, x_3) = \\ = |1111\ 0000\ 1100\ 1100\ 0101\ 1010\ 0010\ 0111|^T = !x_1(!x_2 \vee \\ \vee x_3)!x_5 \vee ((!x_1 \vee x_4)!x_3 \vee x_1x_3)x_2x_5 \vee x_1(!x_2!x_4 \vee x_2!x_3x_4).$$

1. Из рассмотрения БФ следует, что она немонотонна по всем переменным, и поэтому без увеличения числа переменных с целью повышения монотонности при $n - k > 0$ она не может быть мажоритарно разложена. Так как БФУ зависит от сравнительно большого числа переменных, то их увеличение не проводилось.

2. Так как $h = 17$, то формульный и графовый методы строят схемы из 16 МЭ.

3. Выполним мультиплексорную декомпозицию при $n - k = 3$, $Q_0 = !x_5$, $Q_1 = !x_4$, $Q_2 = x_3 \oplus x_5$, $Q_3 = !x_3x_4 \vee x_3x_5$:

$$f(Q_0, Q_1, Q_2, Q_3; x_2, x_1) = MX(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2); \\ \Phi_1 = x_2; \Phi_2 = x_1.$$

Так как для функций Q_i условия покрытия не выполняются, то заданная БФУ реализуется схемой из $3 + 3 + 9 = 15$ МЭ.

4. Эта БФУ реализуется в [11] схемами из 31, 19, 13 и 11 МЭ, построенными четырьмя различными методами.

Сформулируем выводы по материалу настоящего раздела.

1. В примере 4.40 мультиплексорное разложение по двум переменным строит простейшую схему.

2. В примере 4.41 мультиплексорная декомпозиция строит простейшую схему.

3. В примере 4.42 мультиплексорное разложение по одной переменной или мультиплексорное разложение по подфункции строит простейшие схемы.

4. В примере 4.43 мультиплексорная декомпозиция строит схему, содержащую на один МЭ больше, чем известная схема.

5. В примере 4.44 мультиплексорное разложение строит схему из семи МЭ, в то время как известна схема из трех МЭ.

6. В примере 4.45 мультиплексорная декомпозиция строит схему из 15 МЭ, в то время как известна схема из 11 МЭ.

4.2. Реализация булевых функций схемами из пятиходовых мажоритарных элементов

4.2.1. Основные соотношения

Пятиходовой МЭ описывается БФУ вида

$$F(a, b, c, d, e) = |0000 \ 0001 \ 0001 \ 0111 \ 0001 \ 0111 \ 0111 \ 1111|^T.$$

Эта БФУ является симметрической

$$F = S_{3,4,5}(a, b, c, d, e).$$

Являясь пороговой, она может быть записана в виде:

$$F = \text{sign}(a + b + c + d + e - 3).$$

Так как эта БФУ является самодвойственной, то она может быть записана в виде [24]:

$$\begin{aligned} f_1 &= a \oplus b, \quad f_2 = a \oplus c, \quad f_3 = a \oplus d, \quad f_4 = a \oplus e, \\ F_1 &= f_1(f_2 \# f_3 \# f_4) \vee f_2 f_3 f_4, \quad F = F_1 \oplus a \end{aligned}$$

или

$$\begin{aligned} f_1 &= !a \oplus b, \quad f_2 = !a \oplus c, \quad f_3 = !a \oplus d, \quad f_4 = !a \oplus e, \\ F_1 &= (f_1 \vee f_2)(f_3 \vee f_4) \vee f_3 f_4, \quad F = F_1 \oplus !a. \end{aligned}$$

Используя переменные x_1, x_2, x_3, x_4, x_5 , запишем функцию F в виде:

$$\begin{aligned} F &= x_1 x_2 x_3 \vee x_1 x_2 x_4 \vee x_1 x_2 x_5 \vee x_1 x_3 x_4 \vee x_1 x_3 x_5 \vee x_1 x_4 x_5 \vee \\ &\vee x_2 x_3 x_4 \vee x_2 x_3 x_5 \vee x_2 x_4 x_5 \vee x_3 x_4 x_5 = x_1 (x_2 (x_3 \vee x_4 \vee x_5) \vee \\ &\vee (x_3 \# x_4 \# x_5)) \vee x_2 (x_3 \# x_4 \# x_5) \vee x_3 x_4 x_5. \end{aligned}$$

Применяя арифметический полином с маскированием [27], эту БФУ можно записать в виде:

$$F = r_3^3 \text{bin} \left(1 + \sum_{i=1}^5 x_i \right).$$

В дальнейшем для обозначения этой БФУ будем использовать обозначение

$$F = \text{Maj}(x_1, x_2, x_3, x_4, x_5).$$

Рассмотрим другие представления этой БФУ.

1. Используем мультиплексорную декомпозицию при $n - k = 2$, $Q_0 = 0, Q_1 = x_4 x_5, Q_2 = x_4 \vee x_5, Q_3 = 1$. Так как $Q_0 \leq Q_1 \leq Q_2 \leq Q_3$, то

$$\begin{aligned} F &= Q_0 \vee Q_1 \Phi_2 \vee Q_2 \Phi_1 \vee Q_3 \Phi_1 \Phi_2 = \\ &= \Phi_1 (\Phi_2 \vee x_4 \vee x_5) \vee \Phi_2 x_4 x_5; \\ \text{MX}(Q_0, Q_1, Q_1, Q_2, Q_1, Q_2, Q_2, Q_3; x_1, x_2, x_3) &= \\ &= \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2); \\ \Phi_1 &= \text{MX}(0, 0, 0, 1, 0, 1, 1, 1; x_1, x_2, x_3) = x_1 \# x_2 \# x_3; \\ \Phi_2 &= \text{MX}(0, 1, 1, 0, 1, 0, 0, 1; x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3. \end{aligned}$$

Так как функции Φ_1 и Φ_2 описывают параллельный одноразрядный двоичный сумматор (СУМ), а БФ для F совпадает с описанием 3-универсального логического модуля (УЛМЗ), то рассматриваемый элемент может быть реализован схемой из двух блоков (рис. 4.17).

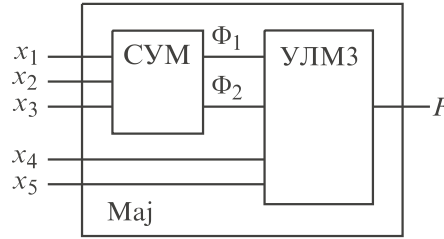


Рис. 4.17

Запишем БФ для функции Φ_2 иначе: $\Phi_2 = !\Phi_1(x_1 \vee x_2 \vee x_3) \vee \vee x_1 x_2 x_3$, тогда из примера 4.27 следует: $\Phi_1 = x_1 \# x_2 \# x_3$, $\Phi_3 = !\Phi_1 \# x_1 \# x_2$, $\Phi_2 = !\Phi_1 \# \Phi_3 \# x_3$, $\Phi_4 = \Phi_1 \# \Phi_2 \# x_4$, $F = \Phi_1 \# \Phi_4 \# x_5$.

Таким образом, пятивходовой МЭ реализуется схемой из пяти трехвходовых МЭ, один из которых должен иметь прямой и инверсный выходы.

2. Выполним мажоритарное разложение функции F по переменной x_5 :

$$\begin{aligned} F(x_1, x_2, x_3, x_4, x_5) &= \Phi_1 \# \Phi_2 \# x_5; \\ \text{MX}(0, 0, 0, x_5, 0, x_5, x_5, 1, 0, x_5, x_5, 1, x_5, 1, 1, 1; x_1, x_2, x_3, x_4) &= \\ &= \text{MX}(0, x_5, x_5, 1; \Phi_1, \Phi_2). \end{aligned}$$

Среди четырех решений этого уравнения выберем следующее:

$$\Phi_1 = \text{MX}(0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1; x_1, x_2, x_3, x_4);$$

$$\Phi_2 = \text{MX}(0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1; x_1, x_2, x_3, x_4).$$

При этом $\Phi_1 = x_1(x_2 \vee x_3 \vee x_4) \vee x_2x_3x_4$, $\Phi_2 = x_2 \# x_3 \# x_4$.

Тогда из примера 4.27 следует, что $\Phi = x_1 \# x_2 \# x_3$, $\Phi_1 = x_1 \# \Phi \# x_4$, $\Phi_2 = x_2 \# x_3 \# x_4$, $F = \Phi_1 \# \Phi_2 \# x_5$.

Таким образом, пятиходовой МЭ может быть построен из четырех трехходовых МЭ. Эта схема совпадает со схемами, построенными с помощью геометрического метода [5] и метода последовательного исключения переменных [16]. Из [8] известно, что эта схема является ответом на вопрос о простейшем решении поставленной задачи.

4.2.2. Исследование функциональных возможностей пятиходовых мажоритарных элементов

Так как такой элемент описывается БФУ, симметрической по всем переменным, то совместим исследование его функциональных возможностей с построением логических уравнений для проведения декомпозиций при различных значениях $n - k$.

1. При $n - k = 0$

$$f(x_1, \dots, x_n) = \text{Maj}(\Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5);$$

$$f(x_1, \dots, x_n) = \text{MX}(0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5).$$

Если и функцию f записать в МФ при этом значении $n - k$, то нахождение решения этого уравнения, обладающего определенными свойствами, связано с огромным перебором, так как число решений в этом случае равно $16^{t_0} \cdot 16^{t_1} = 16^{t_0+t_1} = 16^{2^n}$, где t_0, t_1 — число фрагментов 0 и 1 в столбце значений функции f . Среди этих решений находятся и все те, которые могут быть получены любым из известных методов. Так как конструктивный метод сокращения перебора автору неизвестен, то в дальнейшем такая декомпозиция не используется.

2. При $n - k = 1$, если столбец значений функции f состоит из фрагментов длиной два, входящих во множество $\{0, x_n, 1\}$, то можно записать уравнение в двух формах:

$$f(x_1, \dots, x_n) = \text{Maj}(\Phi_1, \Phi_2, \Phi_3, \Phi_4, x_n);$$

$$f(x_1, \dots, x_n) = \text{MX}(0, 0, 0, 0, x_n, 0, x_n, x_n, 1, 0, x_n, x_n, 1, x_n, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4).$$

Если и функцию f записать в МФ, то это уравнение имеет $5^{t_0+t_1} \cdot 6^{t_2}$ решений, где t_2 — число фрагментов x_n в столбце значений функции f . Эта декомпозиция может быть названа Маж-разложением функции f по переменной x_n .

3. При $n - k = 2$, если столбец значений функции f состоит из фрагментов длиной четыре, входящих в множество $\{G_0=0, G_1=x_{n-1}x_n, G_2=x_{n-1} \vee x_n, G_3=1\}$, то можно записать уравнение в двух формах:

$$\begin{aligned} f(x_1, \dots, x_n) &= \text{Maj}(\Phi_1, \Phi_2, \Phi_3, x_{n-1}, x_n); \\ f(x_1, \dots, x_n) &= \\ &= \text{MX}(G_0, G_1, G_1, G_2, G_1, G_2, G_2, G_3; \Phi_1, \Phi_2, \Phi_3). \end{aligned}$$

Это уравнение имеет $3^{t_3} \cdot 3^{t_4} = 3^{t_3+t_4}$ решений, где t_3 и t_4 — число фрагментов G_1 и G_2 в столбце значений функции f . Декомпозиция этого типа может быть названа Маж-разложением функции f по двум переменным x_{n-1} и x_n .

4. При $n - k = 3$, если столбец значений функции f состоит из фрагментов длиной восемь, входящих в множество $\{G_0=x_{n-2}x_{n-1}x_n, G_1=x_{n-2} \# x_{n-1} \# x_n, G_2=x_{n-2} \vee x_{n-1} \vee x_n\}$, то можно записать уравнение в двух формах:

$$\begin{aligned} f(x_1, \dots, x_n) &= \text{Maj}(\Phi_1, \Phi_2, x_{n-2}, x_{n-1}, x_n); \\ f(x_1, \dots, x_n) &= \text{MX}(G_0, G_1, G_1, G_2; \Phi_1, \Phi_2). \end{aligned}$$

Это уравнение имеет 2^{t_5} решений, где t_5 — число фрагментов G_1 в столбце значений функции f . Декомпозиция этого типа может быть названа Маж-разложением функции f по трем переменным x_{n-2} , x_{n-1} и x_n .

5. При $n - k = 4$, если столбец значений функции f состоит из фрагментов F_1 и F_2 , рассмотренных в предыдущем разделе, то можно записать уравнение в двух формах:

$$\begin{aligned} f(x_1, \dots, x_n) &= \text{Maj}(\Phi_1, x_{n-3}, x_{n-2}, x_{n-1}, x_n); \\ f(x_1, \dots, x_n) &= \text{MX}(F_1, F_2; \Phi). \end{aligned}$$

Это уравнение имеет единственное решение. Декомпозиция этого типа может быть названа Маж-разложением функции f по четырем переменным: x_{n-3} , x_{n-2} , x_{n-1} и x_n .

6. Если столбец значений функции f состоит из следующих типов фрагментов: $\{0, G(X_1), 1\}$, или $\{0, G(X_1)\}$, или $\{G(X_1), 1\}$, то можно записать уравнение в двух формах:

$$\begin{aligned} f(x_1, \dots, x_n) &= \text{Maj}(\Phi_1, \Phi_2, \Phi_3, \Phi_4, G(X_1)); \\ f(x_1, \dots, x_n) &= \text{MX}(0, 0, 0, G(X_1), 0, G(X_1), G(X_1), 1, \\ &0, G(X_1), G(X_1), 1, G(X_1), 1, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4). \end{aligned}$$

Если и функцию f записать в МФ, то это уравнение имеет $5^{t_0+t_1} \cdot 6^{t_6}$ решений, где t_6 — число фрагментов $G(X_1)$ в столбце значений функции f . Эта декомпозиция этого типа может быть названа *Маж-разложением* функции f по подфункции $G(X_1)$.

7. При наличии прямых и инверсных переменных класс реализуемых функций расширяется.

Так, например, при $n - k = 1$

$$\begin{aligned} f(x_1, \dots, x_n) &= \text{Maj}(\Phi_1, \Phi_2, \Phi_3, \Phi_4, !x_n); \\ f(x_1, \dots, x_n) &= \text{MX}(0, 0, 0, !x_n, 0, !x_n, !x_n, 1, 0, !x_n, !x_n, \\ &\quad 1, !x_n, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4). \end{aligned}$$

4.2.3. Примеры реализации булевых функций

В [15] было отмечено, что попытки Миаты [7] и других авторов предложить методы синтеза схем из пятивходовых МЭ, в какой-то степени аналогичные методам исключения переменных, предложенным для трехвходовых МЭ, не увенчались успехом, в чем Миата и признался в [7].

Методы синтеза схем в этом базисе, изложенные в [15], являются весьма трудоемкими.

Покажем, что элементы *Маж* могут использоваться с помощью мультиплексорного и формульного методов синтеза схем, а также на основе разложения Шеннона по одной переменной.

Пример 4.46. Реализовать БФУ [15] $f(x_1, x_2, x_3) = |0001\ 0111|^T$, которая описывает трехвходовой МЭ.

Запишем уравнение при $n - k = 2$, $G_0 = 0$, $G_1 = x_2 x_3$, $G_2 = x_2 \vee x_3$, $G_3 = 1$:

$$\begin{aligned} f(x_1, x_2, x_3) &= \text{Maj}(\Phi_1, \Phi_2, \Phi_3, x_2, x_3); \\ &\quad \text{MX}(G_1, G_2; x_1) = \\ &= \text{MX}(G_0, G_1, G_1, G_2, G_1, G_2, G_2, G_3; \Phi_1, \Phi_2, \Phi_3). \end{aligned}$$

Среди девяти решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0; x_1) = 0; \\ \Phi_2 &= \text{MX}(1, 1; x_1) = 1; \\ \Phi_3 &= \text{MX}(0, 1; x_1) = x_1. \end{aligned}$$

Таким образом, как и [15], $x_1 \# x_2 \# x_3 = \text{Maj}(0, 1, x_1, x_2, x_3)$.

Верификация найденного решения наиболее просто выполняется следующим образом:

$$\begin{aligned} f &= \text{sign}(0 + 1 + x_1 + x_2 + x_3 - 3) = \\ &= \text{sign}(x_1 + x_2 + x_3 - 2) = x_1 \# x_2 \# x_3. \end{aligned}$$

Из изложенного следует, что для рассматриваемого МЭ могут применяться все подходы, предложенные для трехвходовых МЭ.

Пример 4.47. Реализовать БФУ

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= |0000 \ 0001 \ 0111 \ 1111|^T = \\ &= x_1(x_2 \vee x_3 \vee x_4) \vee x_2 x_3 x_4, \end{aligned}$$

которая является порождающей функцией модуля, универсального в классе пороговых функций трех переменных.

Запишем уравнение при $n - k = 2$, $G_0 = 0$, $G_1 = x_3 x_4$, $G_2 = x_3 \vee x_4$, $G_3 = 1$:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= \text{Maj}(\Phi_1, \Phi_2, \Phi_3, x_3, x_4); \\ \text{MX}(G_0, G_1, G_2, G_3; x_1, x_2) &= \\ &= \text{MX}(G_0, G_1, G_1, G_2, G_1, G_2, G_2, G_3; \Phi_1, \Phi_2, \Phi_3). \end{aligned}$$

Среди девяти решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1; \\ \Phi_2 &= \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1; \\ \Phi_3 &= \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2. \end{aligned}$$

Таким образом, как и в [15],

$$x_1(x_2 \vee x_3 \vee x_4) \vee x_2 x_3 x_4 = \text{Maj}(x_1, x_1, x_2, x_3, x_4).$$

Верификация найденного решения наиболее просто выполняется при пороговом представлении БФУ [10]:

$$\begin{aligned} f &= \text{sign}(x_1 + x_1 + x_2 + x_3 + x_4 - 3) = \text{sign}(2x_1 + x_2 + x_3 + x_4 - 3) = \\ &= x_1(x_2 \vee x_3 \vee x_4) \vee x_2 x_3 x_4. \end{aligned}$$

Следовательно, этот модуль реализуется на одном рассматриваемом МЭ, в то время как при синтезе этого модуля на трехвходовых МЭ их требовалось два.

Из изложенного следует, что пятивходовой МЭ может использоваться с помощью формульного метода.

Пример 4.48. Реализовать БФ [13]

$$f = x_1!x_3 \vee !x_3x_4 \vee !x_2!x_3 \vee x_1!x_2x_4.$$

1. Так как эта БФ положительно монотонна по переменным x_1, x_4 , построим таблицу истинности:

$$f(x_2, x_3, x_1, x_4) = |1111 \ 0001 \ 0111 \ 0000|^T.$$

Ввиду того что при $n - k = 2$, $G_0 = 0$, $G_1 = x_1 x_4$, $G_2 = x_1 \vee x_4$, $G_3 = 1$ фрагменты функций f и Maj совпадают, составим уравнение:

$$\begin{aligned} x_1!x_3 \vee !x_3 x_4 \vee !x_2!x_3 \vee x_1!x_2 x_4 &= \text{Maj}(\Phi_1, \Phi_2, \Phi_3, x_3, x_4); \\ \text{MX}(G_3, G_1, G_2, G_0; x_2, x_3) &= \\ = \text{MX}(G_0, G_1, G_1, G_2, G_1, G_2, G_2, G_3; \Phi_1, \Phi_2, \Phi_3). \end{aligned}$$

Среди девяти решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(1, 0, 1, 0; x_2, x_3) = !x_3; \\ \Phi_2 &= \text{MX}(1, 0, 1, 0; x_2, x_3) = !x_3; \\ \Phi_3 &= \text{MX}(1, 1, 0, 0; x_2, x_3) = !x_2. \end{aligned}$$

Таким образом, $f = \text{Maj}(!x_3, !x_3, !x_2, x_1, x_4)$. Так как $f = !x_3(x_1 \vee !x_2 \vee x_4) \vee x_1!x_2 x_4$, то из предыдущего примера может быть получено другое решение: $f = \text{Maj}(!x_3, !x_3, x_1, !x_2, x_4)$.

2. В [13], применяя метод построения схем из МЭ с разными показателями мажорирования [4], эта БФУ реализована следующим образом:

$$f = \text{Maj}(x_1, !x_2, !x_3, !x_3, x_4).$$

Все приведенные решения эквивалентны, так как функция Maj симметрична по всем переменным.

Пример 4.49. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0001 \ 0001 \ 0001 \ 1111|^T = x_1 x_2 \vee x_3 x_4.$$

Запишем уравнение при $n - k = 2$, $G_0 = 0$, $G_1 = x_3 x_4$, $G_2 = x_3 \vee x_4$, $G_3 = 1$:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= \text{Maj}(\Phi_1, \Phi_2, \Phi_3, x_2, x_3); \\ \text{MX}(G_1, G_1, G_1, G_3; x_1, x_2) &= \\ = \text{MX}(G_0, G_1, G_1, G_2, G_1, G_2, G_2, G_3; \Phi_1, \Phi_2, \Phi_3). \end{aligned}$$

Среди 27 решений этого уравнения выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2; \\ \Phi_2 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2; \\ \Phi_3 &= \text{MX}(1, 1, 1, 1; x_1, x_2) = 1. \end{aligned}$$

Для реализации подфункции $\Phi = x_1 x_2$ первоначально настроим элемент Maj на функцию 3-универсального модуля $F = \text{Maj}(x_1, x_1, x_2, x_3, x_4)$, а затем настроим этот модуль на требуемую подфункцию

$$\Phi = \text{Maj}(x_1, x_1, x_2, 0, 0).$$

Таким образом, заданная БФУ реализуется схемой из двух элементов Maj .

Обратим внимание, что если заданную БФУ реализовать с помощью мультиплексорного метода при $n - k = 2$ на 3-универсальных модулях, то можно получить не 27, а только одно решение:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= \Phi_1(\Phi_2 \vee x_3 \vee x_4) \vee \Phi_2 x_3 x_4; \\ \text{MX}(G_1, G_1, G_1, G_3; x_1, x_2) &= \text{MX}(G_0, G_1, G_2, G_3; \Phi_1, \Phi_2); \\ \Phi_1 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2; \\ \Phi_2 &= \text{MX}(1, 1, 1, 1; x_1, x_2) = 1. \end{aligned}$$

Таким образом, при $n - k = 2$ элемент Maj имеет ту же номенклатуру фрагментов, что и модуль УЛМЗ, но из-за большего их количества увеличивается число вариантов для выбора более простых остаточных функций. Единственное решение для модуля УЛМЗ всегда существует и для элемента Maj .

Пример 4.50. Реализовать разложение Шеннона по переменной x_i :

$$f = !x_i f(x_i = 0) \vee x_i f(x_i = 1).$$

Из предыдущего примера следует, что

$$\begin{aligned} \Phi &= \text{Maj}(!x_i, !x_i, f(x_i = 0), 0, 0); \\ f &= \text{Maj}(\Phi, \Phi, 1, x_i, f(x_i = 1)). \end{aligned}$$

Таким образом, мультиплексор «2 в 1» реализуется схемой из двух элементов Maj .

4.2.4. Оценки сложности схем

Предположим, что прямые и инверсные входные переменные равнодоступны.

1. Так как элемент Maj может быть настроен на реализацию модуля УЛМЗ, то при использовании формульного метода для числа таких элементов справедливо соотношение

$$\left\lceil \frac{h-1}{2} \right\rceil \leq L(h) \leq \left\lceil \frac{2(h-1)}{3} \right\rceil.$$

Следовательно, при любом методе синтеза схем в рассматриваемом базисе $L(h) \leq \lceil 2(h-1)/3 \rceil$.

Так как в разд. 3.3.3 было показано, что произвольная БФ в базисе $\{\&, \vee, !\}$ из пяти букв реализуется схемой из двух модулей УЛМЗ, то и для рассматриваемого базиса $L(5) = 2$.

2. Из последнего примера следует, что мультиплексор «2 в 1» реализуется схемой из двух элементов Maj. Поэтому

$$L(n) \leq 2(2^{n-1} - 1) = 2^n - 2.$$

Эта оценка может быть снижена, так как в [15] показано, что $L(4) \leq 6$. При этом

$$L(n) \leq 2(2^{n-4} - 1) + 6 \cdot 2^{n-4} = 2^{n-1} - 2.$$

Из этого соотношения следует, что $L(5) \leq 14$. При этом отметим, что в [15] приведена схема из шести элементов Maj, реализующая функцию $f = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5$.

Из изложенного следует, что

$$L(h, n) \leq \min \left(\left\lceil \frac{2(h-1)}{3} \right\rceil, 2^{n-1} - 2 \right).$$

4.3. Реализация булевых функций схемами из трех- и пятиходовых мажоритарных элементов

Весьма сложный метод синтеза схем из МЭ с разными показателями мажорирования изложен в [5].

Использование мультиплексорного метода позволяет для многих БФУ весьма просто решать эту задачу. Так как пятиходовой МЭ «покрывает» трехходовой МЭ, а критерием качества схемы является минимальное число МЭ, то каждый шаг процедуры синтеза будем начинать с применения элемента Maj.

Пример 4.51. Реализовать БФУ

$$f(x_1, \dots, x_5) = |0001 \ 0111 \ 0000 \ 0001 \ 1111 \ 1111 \ 0000 \ 0111|^T.$$

Так как при $n - k = 2$ фрагменты заданной БФУ и элемента Maj совпадают, то составим уравнение в двух формах:

$$\begin{aligned} f(x_1, \dots, x_5) &= \text{Maj}(\Phi_1, \Phi_2, \Phi_3, x_4, x_5); \\ \text{MX}(G_1, G_2, G_0, G_1, G_3, G_3, G_0, G_2; x_1, x_2, x_3) &= \\ &= \text{MX}(G_0, G_1, G_1, G_2, G_1, G_2, G_2, G_3; \Phi_1, \Phi_2, \Phi_3). \end{aligned}$$

Это уравнение имеет 81 решение. Для определения решения составим карты Карно. Если выбрать $\Phi_1 = !x_2$, то функцию Φ_2 можно доопределить до мажоранты $\Phi_2 = x_1 \# !x_2 \# x_3$. При этом функция Φ_3 доопределяется автоматически: $\Phi_3 = x_1 !x_2 \vee x_2 x_3$. Корректность выбранного решения подтверждается его мультиплексорной формой:

$$\Phi_1 = \text{MX}(1, 1, 0, 0, 1, 1, 0, 0; x_1, x_2, x_3);$$

$$\Phi_2 = \text{MX}(0, 1, 0, 0, 1, 1, 0, 1; x_1, x_2, x_3);$$

$$\Phi_3 = \text{MX}(0, 0, 0, 1, 1, 1, 0, 1; x_1, x_2, x_3).$$

Реализуем функцию Φ_3 . Так как она содержит запрещенные фрагменты, то выполним в полученной формуле замену $!x_2 = x_4$: $\Phi_{31} = x_1 x_4 \vee x_2 x_3$. Из примера 4.47 следует, что $\Phi = x_1 \# x_4 \# 0$, $\Phi_{31} = \text{Maj}(\Phi, \Phi, 1, x_2, x_3)$.

Выполняя обратную замену, получим:

$$\Phi = x_1 \# !x_2 \# 0, \quad \Phi_3 = \text{Maj}(\Phi, \Phi, 1, x_2, x_3).$$

Таким образом, заданная БФУ реализуется схемой из двух трех-входовых и двух пятивходовых МЭ.

При использовании формульного метода для $h \geq 6$ следует помнить, что БФ в базисе $\{\&, \vee, !\}$ из двух букв реализуется одним трехвходовым МЭ, из трех букв — одним пятивходовым МЭ, а из пяти букв — двумя пятивходовыми МЭ. При этом $\lfloor (h-1)/2 \rfloor \leq L(h) \leq \lceil 2(h-1)/3 \rceil$.

Пример 4.52. Реализовать БФ

$$f = (x_1 x_2 \vee x_3 x_4)(x_5 x_6 \vee x_7 x_8) \vee \\ \vee (x_9 x_{10} \vee x_{11} x_{12})(x_{13} x_{14} \vee x_{15} x_{16}).$$

Из суперпозиции $f_1 = x_1 x_2$, $f_2 = f_1 \vee x_3 x_4$, $f_3 = f_2(x_5 x_6 \vee x_7 x_8)$, $f_4 = x_9 x_{10}$, $f_5 = f_4 \vee x_{11} x_{12}$, $f_6 = f_5(x_{13} x_{14} \vee x_{15} x_{16})$, $f = f_3 \vee f_6$ следует, что заданная БФ может быть реализована схемой из трех трехвходовых и шести пятивходовых МЭ.

Выводы

1. Предложен формульный метод реализации БФ в базисе $\{\&, \vee, !\}$ из h букв древовидными схемами из $h-1$ трехвходовых МЭ, которые строятся от входов к выходу (снизу вверх). Для пороговых неповторных БФ из h букв этот метод строит каскадные схемы из $h-1$ МЭ.

2. Предложен графовый метод реализации БФ в базисе $\{\&, \vee, !\}$ из h букв каскадными схемами с разветвлениями из $h-1$ трехвходовых

вых МЭ. Показано, что изменение порядка записи БФ в общем случае изменяет число разветвлений или их расположение в схеме.

3. Предложен более простой, чем известный, метод построения мажоритарных разложений по одной монотонной переменной. Рассмотрен вопрос о существовании таких разложений по двум монотонным переменным.

4. Предложен мультиплексорный метод реализации БФУ схемами из трехвходовых МЭ, основанный на:

- мажоритарных разложениях;
- мультиплексорной декомпозиции, компоненты которой реализуются на МЭ;
- вспомогательных приемах, упрощающих реализацию функций.

5. Предложены оценки сложности схем из трехвходовых МЭ, реализуемых этими методами.

6. Показано, как использовать формульный и мультиплексорный методы, а также разложение Шеннона по одной переменной для реализации БФУ схемами из пятивходовых МЭ. Получены оценки сложности таких схем.

7. Предложен мультиплексорный метод реализации БФУ схемами из трех- и пятивходовых МЭ.

8. Рассмотрение большого числа примеров, заимствованных из литературы, показало, что для многих из них предлагаемые методы строят схемы, сложность которых не превышает сложности схем, построенных с помощью известных методов. Для БФУ, допускающих неразделительные мажоритарные декомпозиции, мультиплексорный метод обычно неэффективен. При этом необходимо учесть то, что формульный и мультиплексорный методы универсальны и применимы для практически любых логических элементов, кроме пороговых, в то время как известные методы ориентированы в данном случае только на использование МЭ.

Л и т е р а т у р а

1. *Lindaman R.* A theorem for deriving majority-logic networks within an augmented Boolean algebra // IRE Trans. on Electronic Comput. 1960. N 3.
2. *Cohn M., Lindaman R.* Axiomatic majority-decision logic // IRE Trans. on Electronic Comput. 1961. N 1.
3. *Akers S. B.* A truth table method for the synthesis of combinational logic // IRE Trans. on Electronic Comput. 1961. N 4.
4. *Akers S. B.* Synthesis of combinational logic using threeinput majority gates // Switching circuit theory and logical design. Proc. 3rd annual sympos. AIEE publ. 1962.
5. *Muller H. S., Winder R. O.* Majority logic synthesis by geometric methods // IRE Trans. on Electronic Comput. 1962. N 1.
6. *Lefkowitz D.* An algorithm for decreasing inputs number and threshold // Proceedings of the IRE on the Electronic Comput. 1962. N 7.
7. *Miyata F.* Realization of arbitrary logical functions using majority elements // IEEE Trans. Electronic Comput. 1963. N 3.

8. Amarel S., Cooke G., Winder R. Majority gate networks // IEEE Trans. Electronic Comput. 1964. N 1.
9. Варшавский В. И., Розенблум Л. Я. О минимизации пирамидальных схем из мажоритарных элементов // Изв. АН СССР. Технич. кибернетика. 1964. № 3.
10. Боголюбов И. Н., Овсевич Б. Л., Розенблум Л. Я. Пороговая и мажоритарная логика // Информ. листок. 1964. № 2 (29).
11. Варшавский В. И., Розенблум Л. Я. Синтез пирамидальных схем из мажоритарных элементов // Изв. АН СССР. Технич. кибернетика. 1965. № 1.
12. Варшавский В. И., Розенблум Л. Я. Функциональная разделимость в мажоритарных схемах // Изв. АН СССР. Технич. кибернетика. 1965. № 4.
13. Боголюбов И. Н., Овсевич Б. Л., Розенблум Л. Я. Синтез схем из пороговых и мажоритарных элементов // Сети передачи информации и их автоматизация. 1965.
14. Варшавский В. И. Мажоритарная декомпозиция // Автоматика и телемеханика. 1965. № 9.
15. Розенблум Л. Я. Некоторые вопросы синтеза логических схем из мажоритарных элементов. Дис. на соиск. уч. степени канд. техн. наук. Л.: ВЦ ЛО математического института им. В.А. Стеклова, 1965.
16. Розенблум Л. Я. Об одном методе синтеза мажоритарных схем // Изв. высш. учебных заведений. Приборостроение. 1965. № 5.
17. Розенблум Л. Я. О минимизации схем в мажоритарном базисе // Кибернетика. 1968. № 1.
18. Поспелов Д. А. Логические методы анализа и синтеза схем. М.: Энергия, 1974.
19. Овсевич Б. Л., Розенблум Л. Я. Проектирование вычислительных и управляющих схем на мажоритарных элементах. Л.: Дом научно-технич. пропаганды, 1969.
20. Cohen S., Winder R. Threshold gate building blocks // IEEE Trans. on Comput. 1969. № 9.
21. Артюхов В. Л., Кондратьев В. Н., Шалыто А. А. Реализация булевых функций арифметическими полиномами // Автоматика и телемеханика. 1988. № 4.
22. Кондратьев В. Н., Шалыто А. А. Реализация булевых функций одним линейным арифметическим полиномом с маскированием // Автоматика и телемеханика. 1996. № 1.
23. Шалыто А. А. Switch-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
24. Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1283744 // Бюл. изобр. 1987. № 2.
25. Артюхов В. Л., Шалыто А. А. Реализация булевых формул однородными мультиплексорными и мажоритарными каскадами // Изв. РАН. Теория и системы управления. 1996. № 5.
26. Кузнецов Б. П., Шалыто А. А. Реализация булевых формул линейными бинарными графами. III. Оптимизация числа и суммарной длины путей // Изв. РАН. Теория и системы управления. 1995. № 5.
27. Артюхов В. Л., Шалыто А. А. Судовые управляющие логические системы. Л.: ИПК СП, 1984.
28. Блох А. III. Граф-схемы и их применение. Минск: Вышэйшая школа, 1975.

Глава 5

Реализация булевых функций схемами из мультиплексоров

5.1. Методы реализации булевых функций схемами из мультиплексоров «2 в 1»

5.1.1. Модификация канонического метода Блоха

Известен канонический метод Блоха [1], который основан на графической форме представления разложения Шеннона заданной функции и всех ее остаточных последовательно, по крайним левым входным переменным таблицы истинности (ТИ). Метод позволяет весьма эффективно строить схемы из переключаемых контактов.

Этот метод применим также и для построения схем из мультиплексоров (МХ) «2 в 1». Он состоит в следующем: по ТИ с фиксированным порядком входных переменных строится каноническая структура (КС); эта структура минимизируется за счет объединения кустов, помеченных одинаковыми цифрами; в минимизированной КС каждый куст заменяется мультиплексором.

Число мультиплексоров в построенной таким образом схеме, на входы которой подаются только безынверсные входные переменные, равно $N - 1$, где N — максимальный номер куста в КС. Эта схема может быть упрощена за счет замены переменной или ее инверсией каждого мультиплексора, на 2-входы которого подаются константы 0 и 1.

Предлагаемая модификация канонического метода позволяет непосредственно строить схемы, в которых на 2-входы мультиплексоров кроме указанных констант может подаваться также одна из переменных и (или) ее инверсия. Эта модификация состоит в изменении первого этапа канонического метода, т. е. в построении по ТИ с фиксированным порядком входных переменных модифицированной КС (МКС), кусты которой могут кодироваться не только цифрами, но и входными переменными и их инверсиями.

Число мультиплексоров в схеме, равное $N - 1$, как и в каноническом методе, может минимизироваться также за счет перестановки столбцов значений входных переменных в исходной таблице истинности.

Пример 5.1. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0000\ 1101\ 1011\ 0110|^T.$$

На рис. 5.1 приведена МКС, построенная по ТИ, а на рис. 5.2 — схема из шести мультиплексоров, изоморфная этой МКС.

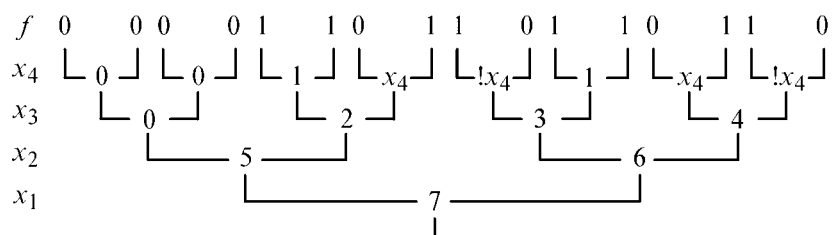


Рис. 5.1

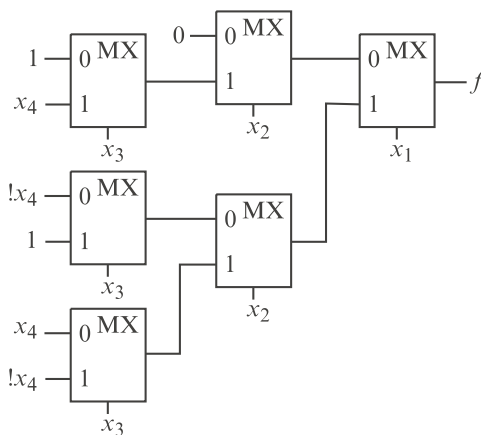


Рис. 5.2

Изменим порядок расположения в таблице истинности на «зеркальный». При этом для ее перестроения воспользуемся двухэтапным методом Блоха, который состоит в следующем:

— на первом этапе столбцы входных переменных переставляются в требуемом порядке без изменения столбца значений функции (табл. 5.1);

— на втором этапе строки измененной ТИ располагаются в лексикографическом порядке, образуя искомую ТИ (табл. 5.2).

Т а б л и ц а 5.1

x_4	x_3	x_2	x_1	f
0	0	0	0	0
1	0	0	0	0
0	1	0	0	0
1	1	0	0	0
0	0	1	0	1
1	0	1	0	1
0	1	1	0	0
1	1	1	0	1
0	0	0	1	1
1	0	0	1	0
0	1	0	1	1
1	1	0	1	1
0	0	1	1	0
1	0	1	1	1
0	1	1	1	1
1	1	1	1	0

Т а б л и ц а 5.2

x_4	x_3	x_2	x_1	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	1	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

На рис. 5.3 приведена МКС, построенная по преобразованной ТИ, а на рис. 5.4 — схема из четырех мультиплексоров, изоморфная МКС.

При использовании рассматриваемого метода число мультиплексоров «2 в 1», требующихся для реализации произвольной БФУ, существенно зависящей от n переменных, определяется соотношением

$$n-1 \leq L(n) \leq 2^{n-1} - 1.$$

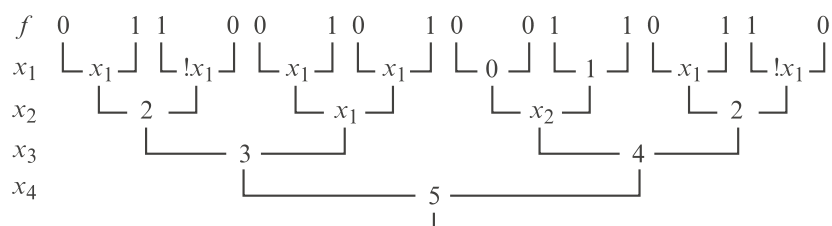


Рис. 5.3

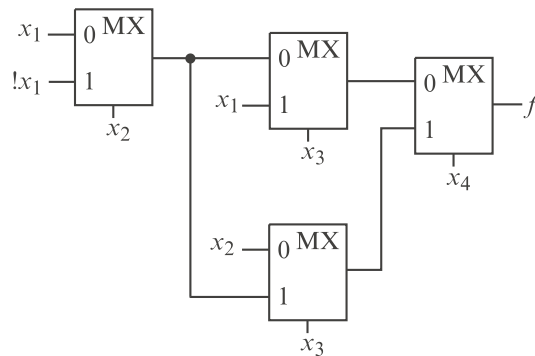


Рис. 5.4

Недостатки рассмотренного метода состоят в том, что
 — для заданной ТИ в ходе синтеза в каждом ярусе применяются одни и те же входные переменные;
 — мультиплексоры в схеме соединяются одинаково: выход предыдущего элемента соединяется с 2-входом по крайней мере одного из последующих элементов.

5.1.2. Распределительный метод

Этот метод является частным случаем мультиплексорного метода, но менее трудоемок.

Метод является графической формой представления разложения Шеннона заданной функции и всех ее остаточных последовательно, по крайним правым входным переменным ТИ.

Столбец значений произвольной БФУ n переменных в общем случае может состоять из четырех типов фрагментов — $\{0, !x_n, x_n, 1\}$. Из $4! = 24$ вариантов расположения этих символов на входах МХ «4 в 1» выберем следующий: $g_0=0$, $g_1=x_n$, $g_2=!x_n$, $g_3=1$. При этом уравнение, определяющее разложение функции f по переменной x_n , имеет вид:

$$f(x_1, \dots, x_n) = \text{MX}(0, x_n, !x_n, 1; \Phi_1, \Phi_2).$$

Это уравнение имеет единственное решение. У д и в и т е л ь - н ы м является тот факт, что

$$\text{MX}(0, x_n, !x_n, 1; \Phi_1, \Phi_2) = \text{MX}(\Phi_1, \Phi_2; x_n).$$

Из этого соотношения следует, что МХ «4 в 1» может быть заменен на МХ «2 в 1». Таким образом, исходное уравнение приобретает вид

$$f(x_1, \dots, x_n) = \text{MX}(\Phi_1, \Phi_2; x_n).$$

Это уравнение, имеющее единственное решение, может быть найдено весьма оригинально: функцию Φ_1 образуют значения заданной БФУ, расположенные на позициях с четными номерами, а функцию Φ_2 — значения заданной БФУ с нечетными номерами. Это позволяет, не записывая указанные уравнения, проводить распределение значений функции непосредственно в ходе построения схемы, выполняемого от выхода к входам. При этом входы различных элементов, имеющие одинаковые пометки, объединяются, а элементы, которые имеют одинаковую пометку 2-входов, исключаются.

Пример 5.2. Реализовать БФУ, рассмотренную в предыдущем примере.

На рис. 5.5 приведена схема, построенная по исходной ТИ, а на рис. 5.6 — по преобразованной (табл.5.2).

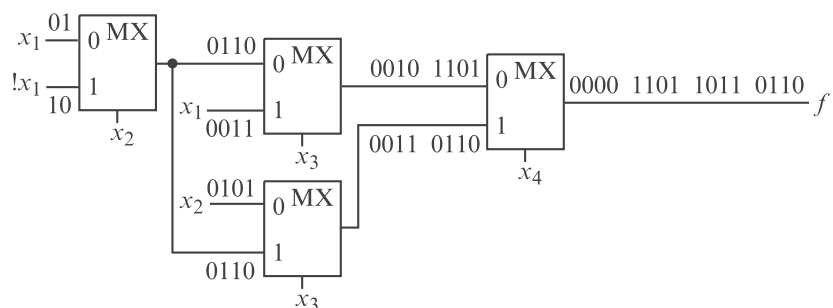


Рис. 5.5

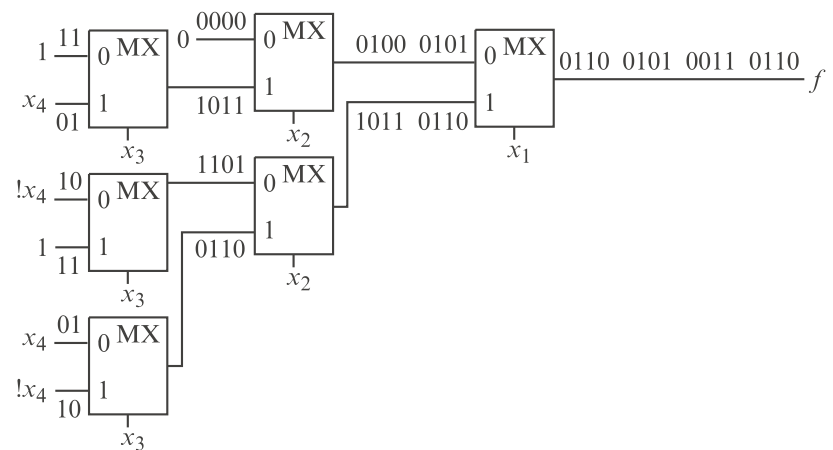


Рис. 5.6

Из приведенного примера следует, что для функции $f(x_1, \dots, x_n)$ этот метод строит ту же схему, что и модификация канонического метода по функции $f(x_n, \dots, x_1)$, и наоборот. Таким образом, вместо перестановки столбцов значений входных переменных в ТИ и построения с помощью любого из этих методов двух схем (по исходной и преобразованной ТИ с «зеркальным» порядком входных переменных) можно непосредственно по исходной ТИ построить две схемы, применяя к ней оба метода, и выбрать простейшую из них.

Предложенный метод обладает теми же недостатками, что и модификация канонического метода, и для него справедливы оценки, приведенные в предыдущем разделе.

Пример 5.3. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0001 \ 0001 \ 0001 \ 1111|^T.$$

На рис. 5.7 приведена схема, построенная по исходной ТИ. В этой схеме элемент, имеющий одинаковые пометки входов, может быть исключен (рис. 5.8).

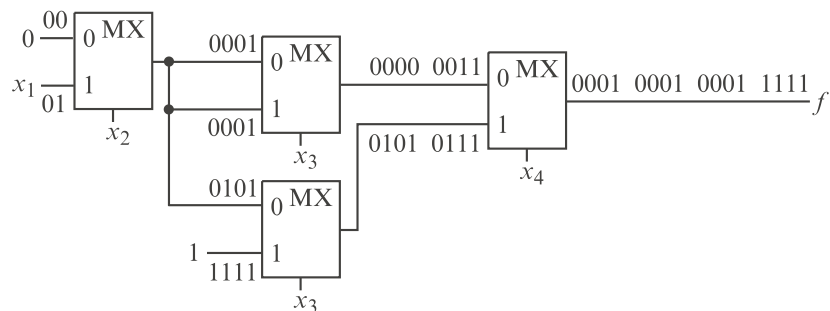


Рис. 5.7

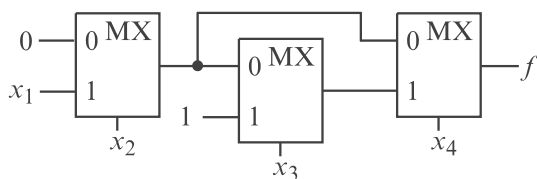


Рис. 5.8

Модификация канонического метода также строит в данном случае схему из трех мультиплексоров «2 в 1».

5.1.3. Мультиплексорный метод с настраиваемым образом декомпозиции

Метод состоит в том, что на i -м шаге построения схемы при минимально возможном значении $n - k \geq 1$ проводится мультиплексорное разложение на основе составления и решения относительно функции Φ_i уравнения вида

$$f_i = MX(g_{0i}, g_{1i}; \Phi_i),$$

где f_i — заданная БФУ или любая ее остаточная, столбец значений которой состоит только из двух типов фрагментов — g_{0i} и g_{1i} .

Рассмотрим сначала применение этого метода к классу функций n переменных, у которых столбец значений состоит только из двух типов фрагментов каждой из перечисленных длин: $j = 2, 4, \dots, 2^{n-1}$.

Для этого класса функций рассматриваемый метод строит однородные каскадные схемы из $n - 1$ мультиплексоров «2 в 1», которые будем называть каскадами первого типа, а указанное выше свойство столбца значений — условием каскадности [2]. Эти схемы являются также и неповторными, так как в них одна и та же входная переменная не подается на входы различных элементов, в то время как на входы одного элемента одновременно могут быть поданы входная переменная и ее инверсия. В этих схемах выход предыдущего элемента соединяется с 1-входом следующего элемента.

Пример 5.4. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0110 \ 1001 \ 1001 \ 0110|^T.$$

Так как в этой функции $g_{01} = x_4$, $g_{11} = !x_4$, то

$$\begin{aligned} f &= MX(x_4, !x_4; \Phi_1); \\ MX(x_4, !x_4, !x_4, x_4, !x_4, x_4, x_4, !x_4; x_1, x_2, x_3) &= \\ &= MX(x_4, !x_4; \Phi_1); \\ \Phi_1 &= MX(0, 1, 1, 0, 1, 0, 0, 1; x_1, x_2, x_3). \end{aligned}$$

Так как в этой функции $g_{02} = x_3$, $g_{12} = !x_3$, то

$$\begin{aligned} \Phi_1 &= MX(x_3, !x_3; \Phi_2); \\ MX(x_3, !x_3, !x_3, x_3; x_1, x_2) &= MX(x_3, !x_3; \Phi_2); \\ \Phi_2 &= MX(0, 1, 1, 0; x_1, x_2). \end{aligned}$$

Так как в этой функции $g_{03} = x_2$, $g_{13} = !x_2$, то

$$\begin{aligned}\Phi_2 &= \text{MX}(x_2, !x_2; \Phi_3); \\ \text{MX}(x_3, !x_3; x_1) &= \text{MX}(x_2, !x_2; \Phi_3); \\ \Phi_3 &= x_1.\end{aligned}$$

Используя правую часть каждого уравнения для определения настройки каждого мультиплексора «2 в 1», построим каскадную схему первого типа из трех таких элементов (рис. 5.9).

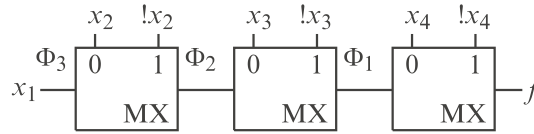


Рис. 5.9

Если снять ограничение, состоящее в том, что на каждом шаге построения схемы мультиплексорное разложение проводится при минимальном значении $n - k \geq 1$, то изложенный метод строит схемы, которые обладают для рассматриваемого класса БФУ некаскадной структурой. При этом

$$L(n) \geq n - 1.$$

Пример 5.5. Реализовать БФУ, рассмотренную в предыдущем примере.

Несмотря на то что, как было показано в предыдущем примере, для этой БФУ можно выполнить разложение при $n - k = 1$, выполним его при $n - k = 2$.

При этом так как $g_{01} = x_3 \oplus x_4$, $g_{11} = !g_{01}$, то

$$\begin{aligned}f &= \text{MX}(g_{01}, !g_{01}; \Phi); \\ \text{MX}(g_{01}, !g_{01}, !g_{01}, g_{01}; x_1, x_2) &= \text{MX}(g_{01}, !g_{01}; \Phi); \\ \Phi &= \text{MX}(0, 1, 1, 0; x_1, x_2).\end{aligned}$$

Реализуя функции g_{01} и Φ при $n - k = 1$, построим искомую схему из четырех мультиплексоров «2 в 1» (рис. 5.10).

Перейдем к рассмотрению вопроса о реализации произвольных БФУ с помощью рассматриваемого метода.

Пример 5.6. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0111 \ 1101 \ 0101 \ 1111|^T.$$

Этот столбец значений состоит из двух типов фрагментов длиной два, но четырех типов фрагментов длиной четыре. Поэтому он не может быть реализован каскадом первого типа.


$$f = \text{MX}(x_4, 1; \Phi_1);$$

$$\Phi_1 = \text{MX}(0, 1, 1, 0, 0, 0, 1, 1; x_1, x_2, x_3).$$

$$\Phi_1 = \text{MX}(g_{02}, x_2; \Phi_2);$$

$$\text{MX}(g_{02}, x_2; x_1) = \text{MX}(g_{02}, x_2; \Phi_2);$$

$$\Phi_2 = x_1.$$

$$g_{02} = \text{MX}(x_3, !x_3; \Phi_3);$$

$$\text{MX}(x_3, !x_3; x_2) = \text{MX}(x_3, !x_3; \Phi_3);$$

$$\Phi_3 = x_2.$$

Рис. 5.11

Пример 5.7. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0001 \ 0001 \ 0001 \ 1110|^T.$$

Столбец значений этой функции состоит из четырех типов фрагментов длиной два, и поэтому он не может быть реализован каскадной схемой первого типа.

Первый шаг декомпозиции этой функции может быть выполнен только при $n - k = 2$: $g_{01} = x_3 x_4$, $g_{11} = !x_3 \vee !x_4$. Составим и решим уравнение:

$$f = \text{MX}(g_{01}, g_{11}; \Phi);$$

$$\text{MX}(g_{01}, g_{01}, g_{01}, g_{11}; x_1, x_2) = \text{MX}(g_{01}, g_{11}; \Phi);$$

$$\Phi = \text{MX}(0, 0, 0, 1; x_1, x_2).$$

Остаточные функции g_{01} , g_{11} , Φ содержат по два типа фрагментов при $n - k = 1$: $\{0, x_4\}$, $\{1, !x_4\}$, $\{0, x_2\}$ соответственно. Таким образом, заданная БФУ реализуется схемой из четырех мультиплексов «2 в 1» (рис. 5.12). Так как $g_{11} = !g_{01}$, то так же может быть построена схема, приведенная на рис. 5.13.

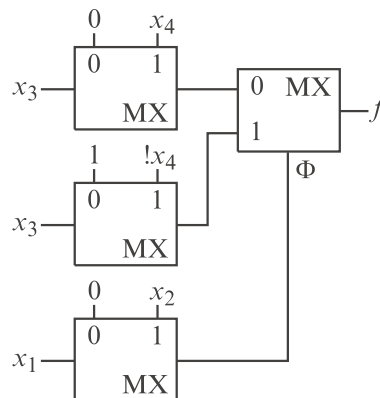


Рис. 5.12

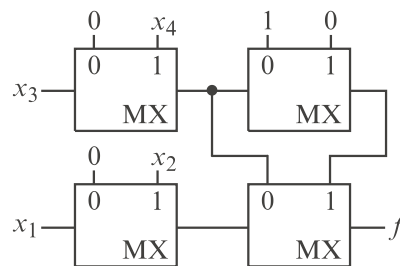


Рис. 5.13

Пример 5.8. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0111 \ 0101 \ 0000 \ 0011|^T.$$

Столбец значений этой функции состоит из трех типов фрагментов длиной два, и поэтому он не может быть реализован каскадной схемой первого типа.

Первый шаг декомпозиции этой функции может быть выполнен при $n - k = 3$: $g_{01} = !x_2 x_3 \vee x_4$, $g_{11} = x_2 x_3$. Составим и решим уравнение:

$$\begin{aligned} f &= \text{MX}(g_{01}, g_{11}; \Phi_1); \\ \text{MX}(g_{01}, g_{11}; x_1) &= \text{MX}(g_{01}, g_{11}; \Phi_1); \\ \Phi_1 &= x_1. \end{aligned}$$

Функция g_{01} может быть декомпозирована при $n - k = 1$ ($g_{02} = x_4$, $g_{12} = 1$):

$$\begin{aligned} g_{01} &= \text{MX}(x_4, 1; \Phi_2); \\ \text{MX}(x_4, 1, x_4, x_4; x_2, x_3) &= \text{MX}(x_4, 1; \Phi_2); \\ \Phi_2 &= \text{MX}(0, 1, 0, 0; x_2, x_3). \end{aligned}$$

Функция Φ_2 может быть декомпозирована при $n - k = 1$ ($g_{03} = x_3$, $g_{13} = 0$):

$$\begin{aligned} \Phi_2 &= \text{MX}(x_3, 1; \Phi_3); \\ \text{MX}(x_3, 0; x_2) &= \text{MX}(x_3, 1; \Phi_3); \\ \Phi_3 &= x_2. \end{aligned}$$

Функция g_{11} может быть декомпозирована при $n - k = 1$ ($g_{04} = 0$, $g_{14} = 1$):

$$\begin{aligned} g_{11} &= \text{MX}(0, 1; \Phi_4); \\ \text{MX}(0, 0, 0, 1; x_2, x_3) &= \text{MX}(0, 1; \Phi_4); \\ \Phi_4 &= \text{MX}(0, 0, 0, 1; x_2, x_3). \end{aligned}$$

Функция Φ_4 может быть декомпозирована при $n - k = 1$ ($g_{05} = 0$, $g_{15} = x_3$):

$$\begin{aligned} \Phi_4 &= \text{MX}(0, x_3; \Phi_5); \\ \text{MX}(0, x_3; x_2) &= \text{MX}(0, x_3; \Phi_5); \\ \Phi_5 &= x_2. \end{aligned}$$

Исходя из приведенных соотношений, может быть построена схема из четырех мультиплексоров «2 в 1» (рис. 5.14).

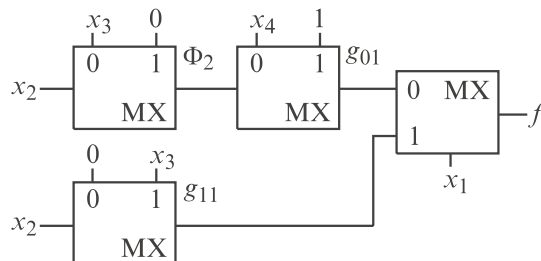


Рис. 5.14

Рассмотренный метод обладает свойством, которым не обладали два предыдущих метода: он позволяет для заданной ТИ проводить разложение не при фиксированном порядке входных переменных, а по любым крайним правым из этих переменных. Так, в последнем примере в первом ярусе используются переменные x_2, x_3 , во втором — переменная x_4 , а в третьем — x_1 , в то время как при применении модификации канонического метода в первом ярусе обязательными являются переменные x_4, x_3 , во втором — переменная x_2 , а в третьем — x_1 .

Метод не позволяет проводить разложение при любом порядке переменных, но отказ от фиксированного их порядка расширяет его оптимизационные возможности.

5.1.4. Смешанный метод

Этот метод основан на применении на каждом шаге декомпозиции одного из двух методов: если для столбца значений функции или подфункции выполняется условие каскадности, то используется мультиплексорный метод, в противном же случае — распределительный метод.

Пример 5.9. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0100 \ 0100 \ 1110 \ 1011|^T.$$

Так как столбец значений состоит из четырех типов фрагментов длиной два, применим распределительный метод по переменной x_4 :

$$\begin{aligned} g_1 &= \text{MX}(0, 0, 0, 0, 1, 1, 1, 1; x_1, x_2, x_3) = x_1; \\ g_2 &= \text{MX}(1, 0, 1, 0, 1, 0, 0, 1; x_1, x_2, x_3). \end{aligned}$$

Так как столбец значений функции g_2 состоит из двух типов фрагментов — длиной два и четыре, то используем мультиплексорный метод с настраиваемым образом декомпозиции по переменной x_3 :

$$\begin{aligned} g_2 &= \text{MX}(!x_3, x_3; g_3); \\ \text{MX}(!x_3, !x_3, !x_3, x_3; x_1, x_2) &= \text{MX}(!x_3, x_3; g_3); \\ g_3 &= \text{MX}(0, 0, 0, 1; x_1, x_2). \end{aligned}$$

Так как столбец значений функции g_2 также состоит из двух типов фрагментов, то применим мультиплексорный метод с настраиваемым образом декомпозиции по переменной x_2 :

$$\begin{aligned} g_3 &= \text{MX}(0, x_2; g_4); \\ \text{MX}(0, x_2; x_1) &= \text{MX}(0, x_2; g_4); \\ g_4 &= x_1. \end{aligned}$$

Искомая схема приведена на рис. 5.15.

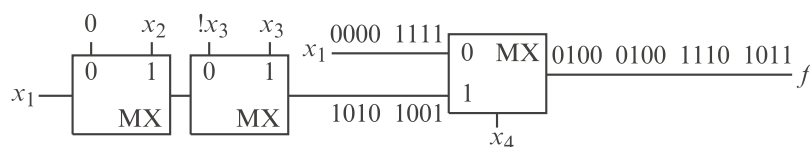


Рис. 5.15

5.1.5. Графовый метод

Этот метод имеет две разновидности, первая из которых состоит из этапов 1—5, а вторая — не содержит этапа 2.

1. По заданной БФ в базисе $\{\&, \vee, !\}$ на основе гл. 15 и [3] строится линейный бинарный граф (ЛБГ), состоящий из h условных (УВ) и двух операторных вершин (ОВ). При этом предполагается, что прямые и инверсные переменные равнодоступны.

2. Если в остове ЛБГ имеются дуги, помеченные символом «0», то пометки соответствующих условных вершин и исходящих из них дуг изменяются на противоположные. При этом строится ЛБГ, который может быть назван *нормированным*.

3. В ЛБГ, рассматривая его справа налево, каждая условная вершина заменяется на мультиплексор «2 в 1», что обеспечивает построение схемы из h мультиплексоров.

4. Последний элемент в схеме может быть заменен последним символом в реализуемой БФ, что обеспечивает построение схемы из $h - 1$ мультиплексоров «2 в 1». На этом этапе длинные связи в схеме могут быть укорочены.

5. Для восстановления без окончательной минимизации булевой формулы по построенной схеме, используя для этого соотношение, описывающее функционирование мультиплексоров «2 в 1», выполняется верификация схемы.

При этом отметим, что в построенной схеме на выходе каждого мультиплексора, несмотря на наличие в общем случае в БФ скобок произвольной глубины, реализуется фрагмент формулы [4], а не подформула, как это имеет место, например, при применении обратной польской записи.

Первая разновидность метода строит каскадную схему второго типа, а вторая — каскадную схему третьего типа. Эти схемы в отличие от каскадных схем первого типа могут содержать разветвления [3].

Пример 5.10. Реализовать БФ

$$f = !x_1 x_3 \vee x_1 !x_2 \vee x_2 !x_3.$$

1. На рис. 5.16 приведен ЛБГ с шестью условными вершинами, построенный по этой формуле.

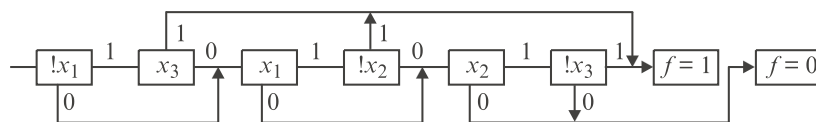


Рис. 5.16

2. На рис. 5.17 приведен нормированный ЛБГ, построенный по исходному графу.

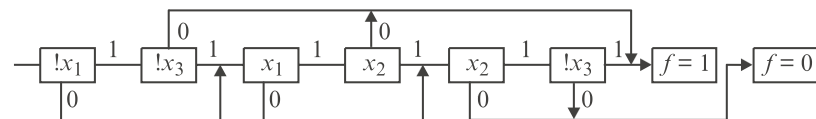


Рис. 5.17

3. На рис. 5.18 приведена каскадная схема второго типа с длинными связями из шести мультиплексоров «2 в 1».

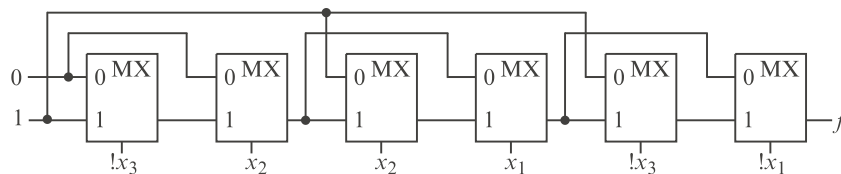


Рис. 5.18

4. На рис. 5.19 приведена каскадная схема второго типа с короткими связями из пяти мультиплексоров «2 в 1».

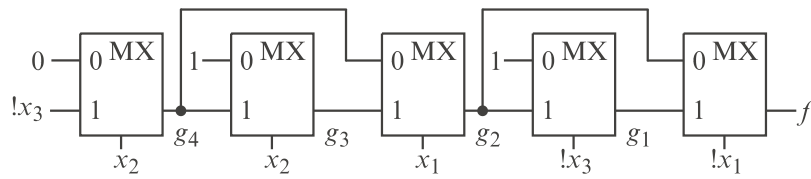


Рис. 5.19

5. Верификация построенной схемы выполняется следующим образом:

$$\begin{aligned}
 g_4 &= 0 \& !x_2 \vee !x_3 x_2 = x_2 !x_3; \\
 g_3 &= 1 \& !x_2 \vee g_4 x_2 = !x_2 \vee x_2 !x_3; \\
 g_2 &= g_4 !x_1 \vee g_3 x_1 = x_1 !x_2 \vee x_2 !x_3; \\
 g_1 &= 1 \& (!x_3) \vee g_2 !x_3 = x_3 \vee x_1 !x_2 \vee x_2 !x_3; \\
 f &= g_2 !x_1 \vee g_1 !x_1 = !x_1 x_3 \vee x_1 !x_2 \vee x_2 !x_3.
 \end{aligned}$$

Сокращение числа разветвлений в каскадной схеме в общем случае может быть обеспечено за счет построения ЛБГ с минимальным числом путей, что достигается изменением порядка записи формулы [3].

Рассмотренная в примере формула инвариантна к изменению порядка ее записи, и поэтому в данном случае число разветвлений не уменьшается.

На рис. 5.20 приведена повторная каскадная схема второго типа, построенная по БФ $f = (x_1 \vee x_2)x_3 \vee x_4$, а на рис. 5.21 — бесповторная каскадная схема второго типа, построенная по БФ $f = x_4 \vee x_3(x_2 \vee x_1)$.

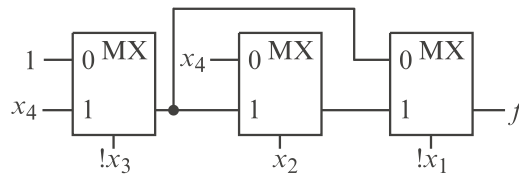


Рис. 5.20

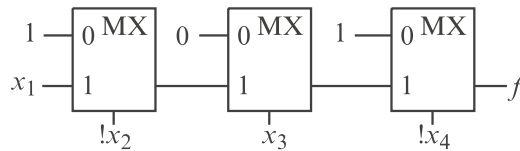


Рис. 5.21

Из изложенного в настоящем разделе и разд. 5.1.1 следует, что

$$L(h, n) \leq \min(h-1, 2^{n-1}-1).$$

Рассмотренный метод, в частности для неповторной БФ в базисе $\{\&, \vee, !\}$, зависящей от n переменных, строит в общем случае каскадные схемы второго и третьего типов из $n-1$ мультиплексоров «2 в 1». Однако существуют повторные формулы в этом базисе из h букв, зависящие от n переменных ($h > n$), которые могут быть реализованы, например с помощью модификации канонического метода, каскадными схемами второго и третьего типов без разветвлений из $n-1$ мультиплексоров «2 в 1».

На рис. 5.22 приведена построенная с помощью модификации канонического метода каскадная схема третьего типа из трех МХ, реализующая БФ $f = !x_1 x_2 \vee (x_1 \vee x_3) x_4$. На рис. 5.23 приведена каскадная схема второго типа, построенная в результате преобразования предыдущей схемы.

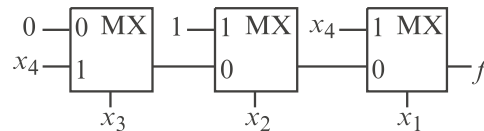


Рис. 5.22

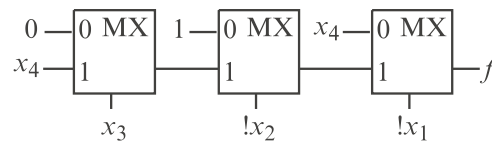


Рис. 5.23

В заключение раздела отметим, что схема из мультиплексоров «2 в 1», естественно, может строиться не только по ЛБГ, но и по бинарному графу, построенному по ТИ заданной функции (разд. 16.4.1, 16.4.2) за счет замены в нем каждой условной вершины мультиплексором, а операторной вершины $f=0$ ($f=1$) — константой 0 (1). Эта схема в дальнейшем может быть упрощена за счет замены переменной или ее инверсией каждого мультиплексора, на 2-входы которого подаются указанные выше константы.

5.1.6. Формульный метод

Если построена формула в базисе $\{\&, \vee, \oplus, !\}$ из h_1 букв, то по ней формульным методом может быть построена схема максимальной глубины из $h - 1$ двухвходовых элементов $\{\&, \vee, \oplus\}$. Если в этой схеме каждый элемент заменить на мультиплексор «2 в 1», то получим искомую схему. При этом

$$L(h_1) = h_1 - 1 + B,$$

где B — число двухвходовых элементов «неравнозначность» в исходной схеме, не связанных с ее входами.

Пример 5.11. Реализовать БФ

$$f = (!x_1 x_3 \oplus x_2) \vee x_4.$$

Схема максимальной глубины, реализующая эту формулу, является каскадной (рис. 5.24). Эта схема преобразуется в бесповторную однородную каскадную схему первого типа из мультиплексоров «2 в 1» (рис. 5.25).

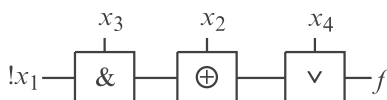


Рис. 5.24

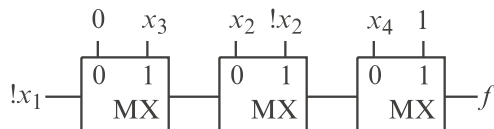


Рис. 5.25

Построение каскадной схемы по формуле проще, чем по таблице истинности, так как для последней определение выполнения условия каскадности связано с перебором по порядку расположения входных переменных в ней. В рассмотренном примере оно выполняется при x_1, x_3, x_2, x_4 и не выполняется при x_1, x_2, x_3, x_4 .

Пример 5.12. Реализовать БФУ

$$f(x_1, x_2, x_3) = |0001\ 0110|^T.$$

Для этой функции условие каскадности не выполняется ни при одном порядке переменных. Но так как эта БФУ с помощью преоб-

разования Артюхова—Шалыто (разд. 1.7 и [2]) может быть представлена повторной формулой

$$f = (x_1 \vee x_2) !x_3 \oplus x_1 \oplus x_2,$$

то она реализуется повторной однородной каскадной схемой первого типа, представленной на рис. 5.26.

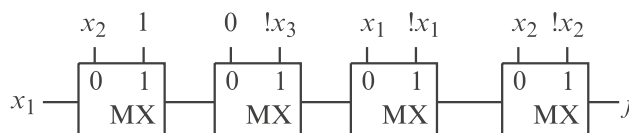


Рис. 5.26

Пример 5.13. Реализовать БФУ

$$f(x_1, x_2, x_3) = |0111\ 1110|^T.$$

Для этой функции условие каскадности не выполняется ни при одном порядке переменных. По карте Карно строится булева формула

$$f = !x_1 x_3 \vee x_1 !x_2 \vee x_2 !x_3,$$

которая реализуется древовидной схемой из пяти мультиплексоров «2 в 1».

Разложение Риды этой функции по переменной x_1 строит БФ

$$f = x_1 (!x_2 \oplus x_3) \oplus (x_2 \vee x_3),$$

которая реализуется недревовидной схемой из пяти мультиплексоров «2 в 1».

Преобразование Артюхова—Шалыто по переменной x_1 строит булеву формулу

$$f = ((!x_1 \vee x_2) x_3 \vee !x_1 x_2) \oplus x_1,$$

которая реализуется древовидной схемой из пяти мультиплексоров «2 в 1».

Учет того факта, что заданная БФУ является «зеркальной» (разд. 10.7), приводит к построению формулы $f = (x_1 \oplus x_2) \vee (x_1 \oplus x_3)$, которая реализуется древовидной схемой из трех элементов (рис. 5.27).

В заключение раздела отметим, что недревовидными схемами могут быть реализованы неповторные формулы в рассматриваемом базисе. Например, булева формула $f = x_1 x_2 \oplus x_3 x_4$ реализуется такой схемой, представленной на рис. 5.13.

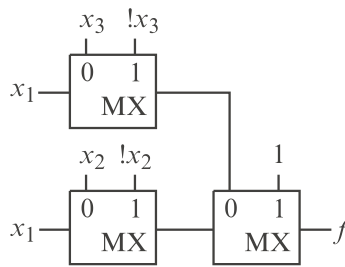


Рис. 5.27

5.1.7. Формульный метод в мультиплексорной форме

Рассмотренный в предыдущем разделе формульный метод ориентирован на построение простейшей по введенному критерию БФ. Однако если для применяемых элементов «родным» базисом является операция мультиплексирования «2 в 1», то при удачном выборе порядка переменных, по каждой из которых выполняется разложение Шеннона, может быть построена булева формула, использующая эту операцию и реализуемая формульным методом проще, чем булева формула, записанная в базисе $\{\&, \vee, \oplus, !\}$.

Пример 5.14. Реализовать БФУ, соответствующую формуле

$$f = !x_1!x_2x_3 \vee x_1x_2x_3 \vee !x_3x_4.$$

В базисе $\{\&, \vee, !\}$ простейшая формула имеет вид:

$$f = (!x_1!x_2 \vee x_1x_2)x_3 \vee !x_3x_4,$$

и реализуется формульным методом, древовидной схемой из шести мультиплексоров «2 в 1».

В базисе $\{\&, \vee, \oplus, !\}$ простейшая формула имеет вид:

$$f = (x_1 \oplus !x_2)x_3 \vee !x_3x_4,$$

и реализуется формульным методом, древовидной схемой из четырех мультиплексоров «2 в 1».

При применении операции мультиплексирования формула строится следующим образом:

$$\begin{aligned} f &= x_4!x_3 \vee (x_1 \oplus !x_2)x_3 = \\ &= \text{MX}(x_4, x_1 \oplus !x_2; x_3) = \text{MX}(x_4, \text{MX}(!x_2, x_2; x_1); x_3). \end{aligned}$$

Этой формуле соответствует древовидная схема из двух мультиплексоров «2 в 1» (рис. 5.28).

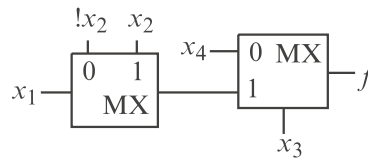


Рис. 5.28

Обратим внимание на то, что в этой схеме число элементов меньше чем $n - 1$.

5.1.8. Примеры сравнительного использования предложенных методов

Пример 5.15. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0110 \ 1001 \ 1001 \ 0110|^T.$$

1. Так как в данном случае БФ в базисе $\{\&, \vee, !\}$ состоит из 22 букв, то графовый метод строит каскадные схемы второго и третьего типов из 21 мультиплексора «2 в 1».

2. Модификация канонического метода строит по ТИ недревовидную схему из пяти мультиплексоров «2 в 1».

3. Распределительный метод также строит недревовидную схему из пяти мультиплексоров «2 в 1».

4. Мультиплексорный метод по таблице истинности и формульный метод по БФ

$$f = x_1 \oplus x_2 \oplus x_3 \oplus x_4$$

строят каскадную схему первого типа из трех мультиплексоров «2 в 1».

Пример 5.16. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |1011 \ 1110 \ 1110 \ 1011|^T.$$

1. Так как в данном случае БФ в базисе $\{\&, \vee, !\}$ состоит из 11 букв, то графовый метод строит каскадные схемы второго и третьего типов из 10 мультиплексоров «2 в 1».

2. Модификация канонического метода строит по ТИ недревовидную схему из пяти мультиплексоров «2 в 1».

3. Распределительный метод строит древовидную схему из четырех мультиплексоров «2 в 1».

4. Мультиплексорный метод по таблице истинности и формульный метод по БФ

$$f = (x_1 \oplus x_2 \oplus x_3) \vee !x_4$$

строят каскадную схему первого типа из трех мультиплексоров «2 в 1».

Пример 5.17. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0101 \ 0111 \ 0111 \ 0111|^T = \\ = (x_1 \vee x_2) x_3 \vee x_4.$$

1. Модификация канонического метода строит по ТИ недревовидную схему из трех мультиплексоров «2 в 1» (рис. 5.29), изоморфную каскадной схеме третьего типа с разветвлениями, которая строится графовым методом по заданной формуле (рис. 5.30).

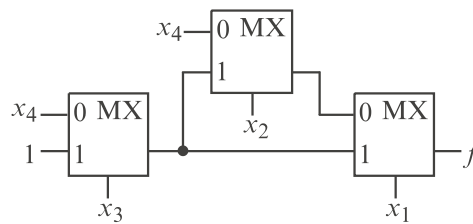


Рис. 5.29

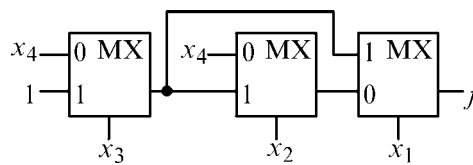


Рис. 5.30

2. Распределительный метод строит древовидную схему (рис. 5.31) из трех мультиплексоров «2 в 1», изоморфную каскадной схеме третьего типа без разветвлений (рис. 5.32), которая строится графовым методом по БФ

$$f = x_4 \vee x_3(x_2 \vee x_1).$$

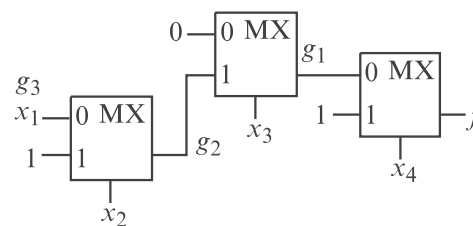


Рис. 5.31

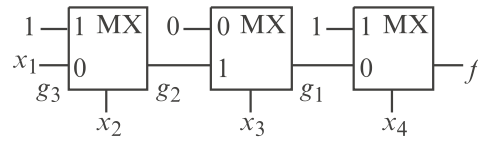


Рис. 5.32

3. Мультиплексорный метод и формульный метод строят каскадную схему первого типа из трех мультиплексоров «2 в 1» (рис. 5.33).

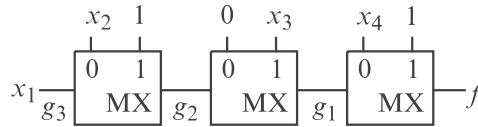


Рис. 5.33

Обратим внимание на удивительный факт, состоящий в том, что для данной БФ в каскадных схемах первого и третьего типов, построенных по формулам $f = (x_1 \vee x_2)x_3 \vee x_4$ и $f = x_4 \vee \vee x_3(x_2 \vee x_1)$ соответственно, подформулы g_i совпадают. Этим свойством обладают булевы формулы, бесповторные в рассматриваемом базисе, со структурой, соответствующей схеме Горнера. При $n = 4$ кроме рассмотренной БФ существует еще один тип формул, обладающий указанным свойством. Представителем этого типа является формула $f = (x_1 x_2 \vee x_3)x_4$.

4. Мультиплексорный метод (при $n - k = 2$ на первом шаге декомпозиции) строит схему из трех мультиплексоров «2 в 1» (рис. 5.34). В этой схеме $g = x_3 \vee x_4$, $\Phi = x_1 \vee x_2$.

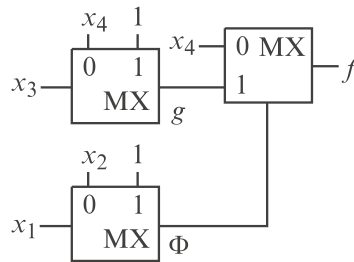


Рис. 5.34

Пример 5.18. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0111 \ 0100 \ 0100 \ 0111|^T.$$

1. Графовый метод строит по БФ $f = (!x_1!x_2 \vee x_1x_2)x_3 \vee !x_3x_4$ каскадные схемы второго и третьего типов из шести мультиплексоров «2 в 1».

2. Модификация канонического метода строит по ТИ недревовидную схему из пяти мультиплексоров «2 в 1» (рис. 5.35).

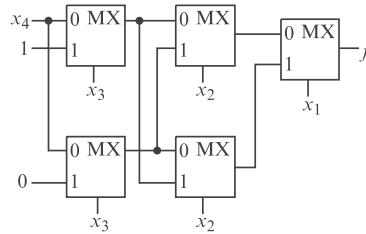


Рис. 5.35

3. Распределительный метод строит по ТИ недревовидную схему из четырех мультиплексоров «2 в 1» (рис. 5.36).

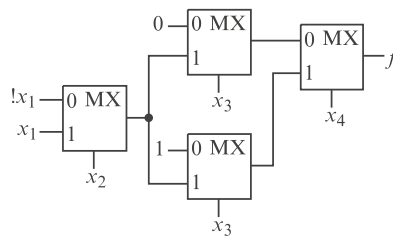


Рис. 5.36

4. Мультиплексорный метод строит древовидную схему из четырех мультиплексоров «2 в 1» (рис. 5.37).

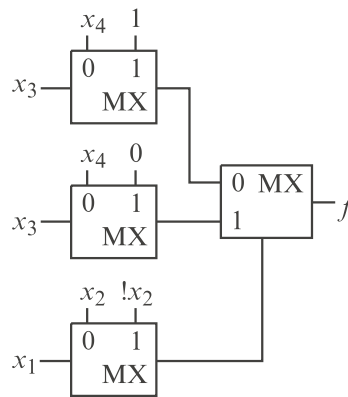


Рис. 5.37

5. Смешанный метод строит схему из четырех мультиплексоров «2 в 1» (рис. 5.38).

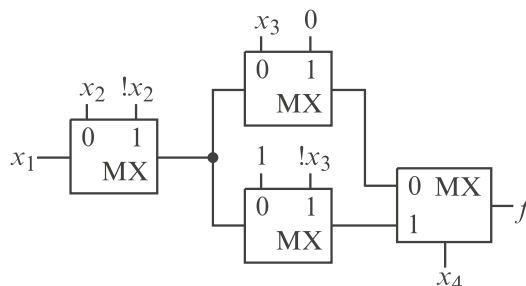


Рис. 5.38

6. Формульный метод строит по формуле $f = (!x_1 \oplus x_2)x_3 \vee !x_3x_4$ схему из четырех мультиплексоров «2 в 1» (рис. 5.39).

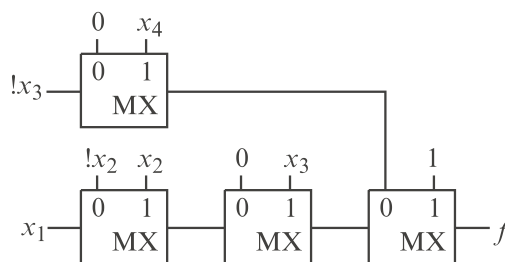


Рис. 5.39

7. Формульный метод в мультиплексорной форме строит схему из двух мультиплексоров «2 в 1» (рис. 5.28).

5.2. Методы реализации булевых функций схемами из мультиплексоров «2^k в 1»

Рассмотрим два таких метода, первый из которых основан на разложении Шеннона заданной БФУ по k крайним левым входным переменным в ТИ, а второй — по k крайним правым входным переменным в ней.

Первый метод известен из литературы [5], а второй является развитием распределительного метода, рассмотренного в разд. 5.1.2, и состоит в «распределении» значений функции или любой подфункции по 2^k направлениям.

Пример 5.19. Реализовать на мультиплексорах «4 в 1» функцию

$$f(x_1, x_2, x_3, x_4) = |0110 \ 1111 \ 0111 \ 1101|^T.$$

При использовании известного метода строится схема из четырех таких мультиплексоров, которая компактно изображена на рис. 5.40.

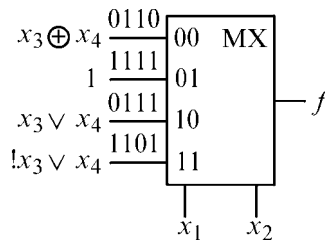


Рис. 5.40

При применении распределительного метода строится схема из трех мультиплексоров «4 в 1», которая компактно изображена на рис. 5.41.

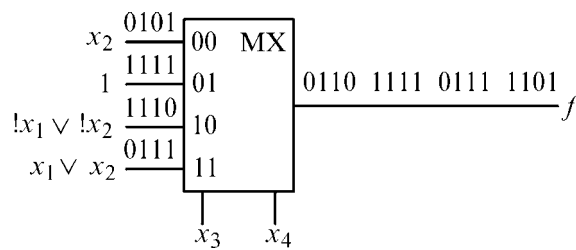


Рис. 5.41

Таким образом, для заданной ТИ без изменения порядка расположения столбцов значений входных переменных имеется возможность построения двух схем и выбора простейшей из них.

Выводы

1. Предложена модификация канонического метода Блоха, предназначенная для реализации БФУ схемами из мультиплексоров «2 в 1».
2. Предложен распределительный метод, предназначенный для реализации БФУ схемами из мультиплексоров.
3. Использование модификации канонического метода и распределительного метода позволяет по одной таблице истинности (без

перестановки столбцов значений входных переменных) получать два решения и выбирать простейшее из них.

4. Предложено применять мультиплексорный метод для реализации БФУ схемами из мультиплексоров «2 в 1». Определено условие, при выполнении которого строятся каскадные схемы первого типа из $n - 1$ мультиплексоров. Эти схемы являются однородными.

5. Предложен смешанный метод, основанный на использовании мультиплексорного и распределительного методов.

6. Предложен графовый метод реализации произвольной БФ в базисе $\{\&, \vee, !\}$ из h букв, основанный на замене в построенном для нее линейном бинарном графе каждой условной вершины на один мультиплексор «2 в 1». Метод строит каскадные схемы второго и третьего типов, состоящие из $h - 1$ мультиплексоров. Эти схемы в общем случае содержат разветвления.

7. Для реализации БФ в базисе $\{\&, \vee, \oplus, !\}$ предложены формульный метод и формульный метод в мультиплексорной форме.

8. Приведены примеры сравнительного применения предложенных методов.

Л и т е р а т у р а

1. Блох А. Ш. Граф-схемы и их применение. Минск: Высшая школа, 1975.
2. Артюхов В. Л., Шалыто А. А. Реализация булевых формул однородными мультиплексорными и мажоритарными каскадами // Изв. РАН. Теория и системы управления. 1996. № 5.
3. Артюхов В. Л., Кузнецов Б. П., Шалыто А. А. Настраиваемые логические устройства для судовых управляющих систем. Л.: ИПК СП, 1986.
4. Артюхов В. Л., Шалыто А. А. Судовые управляющие логические системы. Л.: ИПК СП, 1984.
5. Вау С., Танг С. Универсальные логические модули и их применение // Экспресс-информация. Вычислительная техника. 1970. № 27.

Глава 6

Использование мультиплексорного метода для реализации схем в различных элементных базисах

6.1. Реализация булевых функций схемами из двухвходовых элементов И—НЕ (ИЛИ—НЕ)

Этот элемент реализует булеву функцию $F(a, b) = |1110|^T = !(ab) = !a \vee !b$.

При $n - k = 0$ справедливы соотношения:

$$\begin{aligned} f(x_1, \dots, x_n) &= !(\Phi_1 \Phi_2); \\ f(x_1, \dots, x_n) &= \text{MX}(1, 1, 1, 0; \Phi_1, \Phi_2). \end{aligned}$$

При $n - k = 1$ для функции f , состоящей из двух типов фрагментов $\{1, !x_n\}$, справедливы соотношения:

$$\begin{aligned} f(x_1, \dots, x_n) &= !(\Phi x_n); \\ f(x_1, \dots, x_n) &= \text{MX}(1, !x_n; \Phi). \end{aligned}$$

Для булевых функций, состоящих из двух типов фрагментов $\{!Q, 1\}$, справедливы соотношения:

$$\begin{aligned} f(x_1, \dots, x_n) &= !(\Phi Q); \\ f(x_1, \dots, x_n) &= \text{MX}(1, !Q; \Phi). \end{aligned}$$

Пример 6.1. Декомпозировать БФУ

$$f(x_1, x_2, x_3) = |1011 \ 1110|^T$$

по образу рассматриваемого элемента.

Так как при $n - k = 1$ БФУ содержит только разрешенные фрагменты, то

$$\begin{aligned} f(x_1, x_2, x_3) &= !(\Phi x_3); \\ \text{MX}(!x_3, 1, 1, !x_3; x_1, x_2) &= \text{MX}(1, !x_3; \Phi); \\ \Phi &= \text{MX}(1, 0, 0, 1; x_1, x_2) = !(x_1 \oplus x_2). \end{aligned}$$

Пример 6.2. Декомпозировать БФУ

$$f(x_1, x_2, x_3, x_4) = |0001 \ 1111 \ 1111 \ 0001|^T$$

по образу рассматриваемого элемента.

При $n - k = 1$ БФУ не может быть декомпозирована, так как содержит три типа фрагментов $\{0, x_n, 1\}$. Однако в данном случае может быть выполнена декомпозиция по подфункции $Q = !(x_3 x_4)$:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= !(\Phi Q); \\ \text{MX}(!Q, 1, 1, !Q; x_1, x_2) &= \text{MX}(1, !Q; \Phi); \\ \Phi &= \text{MX}(1, 0, 0, 1; x_1, x_2) = !(x_1 \oplus x_2). \end{aligned}$$

Пример 6.3. Реализовать БФУ

$$f(x_1, x_2) = |0110|^T = !x_1 x_2 \vee x_1 !x_2 = x_1 \oplus x_2$$

схемой из рассматриваемых элементов.

Так как столбец значений БФУ содержит запрещенный фрагмент длиной два (тип фрагмента — x_2), от которого нельзя избавиться при перестановке переменных, то логическое уравнение составим при $n - k = 0$:

$$\begin{aligned} x_1 \oplus x_2 &= !(\Phi_1 \Phi_2); \\ \text{MX}(0, 1, 1, 0; x_1, x_2) &= \text{MX}(1, 1, 1, 0; \Phi_1, \Phi_2). \end{aligned}$$

Это уравнение имеет девять решений. В семи из них по крайней мере одна из остаточных является функцией «равнозначность», которая декомпозируется не проще заданной БФУ. Среди оставшихся двух решений выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(1, 1, 0, 1; x_1, x_2) = !x_1 \vee x_2; \\ \Phi_2 &= \text{MX}(1, 0, 1, 1; x_1, x_2) = x_1 \vee !x_2. \end{aligned}$$

Столбец значений функции Φ_1 содержит запрещенный фрагмент длиной два, поэтому для реализации этой функции составим уравнение при $n - k = 0$:

$$\begin{aligned} !x_1 \vee x_2 &= !(\Phi_{11} \Phi_{12}); \\ \text{MX}(1, 1, 0, 1; x_1, x_2) &= \text{MX}(1, 1, 1, 0; \Phi_{11}, \Phi_{12}). \end{aligned}$$

Это уравнение имеет 27 решений, среди которых выберем следующее:

$$\begin{aligned} \Phi_{11} &= \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1; \\ \Phi_{12} &= \text{MX}(1, 1, 1, 0; x_1, x_2) = !(x_1 x_2). \end{aligned}$$

Столбец значений функции Φ_2 содержит только разрешенные фрагменты длиной два, и поэтому составим и решим уравнение при $n - k = 1$:

$$\begin{aligned} x_1 \vee !x_2 &= !(\Phi x_2); \\ \text{MX}(!x_2, 1; x_1) &= \text{MX}(1, !x_2; \Phi); \\ \Phi &= \text{MX}(1, 0; x_1) = !x_1. \end{aligned}$$

Так как в данном случае переменная $!x_1$ недоступна, то заданная БФУ реализуется неминимальной схемой из пяти элементов. Это связано с видом использованной декомпозиции для функции Φ_2 . Для этой БФУ при $n - k = 2$ можно записать:

$$\begin{aligned} x_1 \vee !x_2 &= !(\Phi_{21} \Phi_{22}); \\ \text{MX}(1, 0, 1, 1; x_1, x_2) &= \text{MX}(1, 1, 1, 0; \Phi_{21}, \Phi_{22}). \end{aligned}$$

Это уравнение имеет 27 решений, среди которых выберем следующее:

$$\begin{aligned} \Phi_{21} &= \text{MX}(1, 1, 1, 0; x_1, x_2) = !(x_1 x_2); \\ \Phi_{22} &= \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2. \end{aligned}$$

Таким образом, заданная БФУ реализуется с помощью предлагаемого метода схемой из четырех элементов, такой же, как получена в [1] путем перебора.

Пример 6.4. Определить класс БФУ, которые оптимально реализуются только при применении соотношения $f = !(\Phi x_n)$.

При использовании этого соотношения может быть построена только каскадная схема.

Для булевой функции n переменных, реализуемой каскадно, известно [2], что столбец ее значений должен состоять не более чем из двух типов фрагментов длиной 2, 4, ..., $2^n - 1$. Из изложенного следует, что БФУ этого класса могут состоять только из таких фрагментов длиной два, как 1 и $!x_n$.

Рассмотрим каскад из двух элементов И—НЕ, для которого может быть записано соотношение:

$$f(\Phi, x_{n-1}, x_n) = |1010 \ 1011|^T = \Phi x_{n-1} \vee !x_n.$$

Таким образом, БФУ этого класса могут состоять из фрагментов длиной четыре только следующих двух типов: $!x_n, !x_{n-1} \vee !x_n$.

Рассмотрим каскад из трех элементов И—НЕ, для которого может быть записано соотношение

$$\begin{aligned} f(\Phi, x_{n-2}, x_{n-1}, x_n) &= |1011 \ 1011 \ 1011 \ 1010|^T = \\ &= (!\Phi \vee !x_{n-2}) x_{n-1} \vee !x_n. \end{aligned}$$

Таким образом, БФУ этого класса могут состоять из фрагментов длиной восемь только следующих двух типов: $x_{n-1} \vee !x_n, !x_{n-2} x_{n-1} \vee !x_n$.

Каскад из четырех элементов И—НЕ описывается формулой

$$f = (\Phi x_{n-3} \vee !x_{n-2}) x_{n-1} \vee !x_n.$$

БФУ этого класса могут состоять из фрагментов длиной 16 только следующих двух типов: $!x_{n-2} x_{n-1} \vee !x_n, (x_{n-3} \vee !x_{n-2}) x_{n-1} \vee !x_n$.

Продолжая этот анализ, можно показать, как это сделано в [3], что рассматриваемый каскад реализует только булевы формулы, в которых знаки $\&$ и $\vee$ чередуются, а чередование инверсий начинается после двух первых букв. При этом при нечетных n формулы начинаются с дизъюнкции инверсий, а при четных n — с конъюнкции.

Аналогичные результаты могут быть получены и для двухвходовых элементов ИЛИ—НЕ.

6.2. Реализация булевых функций схемами из трехвходовых монотонных модулей

Для функций, монотонных по переменной x_i , выполняется соотношение

$$f(x_i=0) \leq f(x_i=1).$$

При этом в [4] приведены следующие соотношения:

$$\begin{aligned} f(x_i=0) f(x_i=1) &= f(x_i=0); \\ f(x_i=0) \vee f(x_i=1) &= f(x_i=1); \\ f &= f(x_i=0) \vee x_i f(x_i=1). \end{aligned}$$

Для функций, монотонных по переменной x_i , выполняется соотношение

$$f(x_i=0) \geq f(x_i=1).$$

При этом в [4] приведены следующие соотношения:

$$\begin{aligned} f(x_i=0)f(x_i=1) &= f(x_i=1); \\ f(x_i=0) \vee f(x_i=1) &= f(x_i=0); \\ f = x_i f(x_i=0) \vee f(x_i=1). \end{aligned}$$

Простейший модуль рассматриваемого типа описывается БФУ

$$F(a, b, c) = |0001 \ 1111|^T = a \vee bc.$$

При перестановке входных переменных эта функция порождает еще два типа столбцов:

$$\begin{aligned} F(b, a, c) &= F(c, a, b) = |0011 \ 0111|^T; \\ F(c, b, a) &= F(b, c, a) = |0101 \ 0111|^T. \end{aligned}$$

1. При $n - k = 0$ справедливы уравнения:

$$f = \Phi_1 \vee \Phi_2 \Phi_3, f = \Phi_2 \vee \Phi_1 \Phi_3, f = \Phi_3 \vee \Phi_1 \Phi_2.$$

Каждое из этих уравнений имеет $3^{t_0} \cdot 5^{t_1}$ решений.

2. При $n - k = 1$ справедливы уравнения:

$$\begin{aligned} f &= \Phi_1 \vee \Phi_2 x_n, \quad f = \text{MX}(0, x_n, 1, 1; \Phi_1, \Phi_2); \\ f &= \Phi_2 \vee \Phi_1 x_n, \quad f = \text{MX}(0, 1, x_n, 1; \Phi_1, \Phi_2); \\ f &= x_n \vee \Phi_1 \Phi_2, \quad f = \text{MX}(x_n, x_n, x_n, 1; \Phi_1, \Phi_2). \end{aligned}$$

Первые два уравнения имеют 2^{t_1} решений, а третье — 3^{t_2} решений.

С помощью первых двух уравнений реализуются БФУ, столбцы которых состоят из трех типов фрагментов длиной два — $\{0, x_n, 1\}$, а с помощью третьего — БФУ, столбцы значений которых состоят только из двух типов фрагментов — $\{x_n, 1\}$.

Частным случаем для первых двух уравнений является разложение БФУ, монотонных по переменной x_n :

$$f = f(x_n=0) \vee x_n f(x_n=1).$$

3. При $n - k = 2$ справедливы уравнения:

$$\begin{aligned} f &= \Phi \vee x_{n-1} x_n, \quad f = \text{MX}(x_{n-1} x_n, 1; \Phi); \\ f &= x_{n-1} \vee \Phi x_n, \quad f = \text{MX}(x_{n-1}, x_{n-1} \vee x_n; \Phi); \\ f &= x_n \vee \Phi x_{n-1}, \quad f = \text{MX}(x_n, x_{n-1} \vee x_n; \Phi). \end{aligned}$$

С помощью этих уравнений реализуются БФУ, столбцы которых состоят соответственно из следующих типов фрагментов длиной четыре: $\{x_{n-1} x_n, 1\}$, $\{x_{n-1}, x_{n-1} \vee x_n\}$, $\{x_n, x_{n-1} \vee x_n\}$.

4. При $n - k = 1$ справедливы уравнения:

$$\begin{aligned} f &= \Phi_1 \vee \Phi_2 !x_n, \quad f = \text{MX}(0, !x_n, 1, 1; \Phi_1, \Phi_2); \\ f &= \Phi_2 \vee \Phi_1 !x_n, \quad f = \text{MX}(0, 1, !x_n, 1; \Phi_1, \Phi_2); \\ f &= !x_n \vee \Phi_1 \Phi_2, \quad f = \text{MX}(!x_n, !x_n, !x_n, 1; \Phi_1, \Phi_2). \end{aligned}$$

Первые два уравнения имеют 2^{t_1} решений, а третье — 3^{t_3} решений.

С помощью первых двух уравнений реализуются БФУ, столбцы которых состоят из трех типов фрагментов длиной два — $\{0, !x_n, 1\}$, а с помощью третьего — БФУ, столбцы значений которых состоят только из двух типов фрагментов — $\{!x_n, 1\}$.

Частным случаем для первых двух уравнений является разложение БФУ, монотонных по переменной $!x_n$:

$$f = !x_n f(x_n = 0) \vee f(x_n = 1).$$

5. При $n - k = 2$ по образу рассматриваемого элемента могут быть декомпозированы только БФУ, содержащие два типа фрагментов Q_0 и Q_1 длиной четыре, для которых выполняется одно из соотношений:

$$\begin{aligned} Q_0 !\Phi \vee Q_1 \Phi &= \Phi \vee \tilde{x}_{n-1} \tilde{x}_n; \\ Q_0 !\Phi \vee Q_1 \Phi &= \tilde{x}_{n-1} \vee \Phi \tilde{x}_n; \\ Q_0 !\Phi \vee Q_1 \Phi &= \tilde{x}_n \vee \Phi \tilde{x}_{n-1}. \end{aligned}$$

6. Рассматриваемый модуль позволяет также проводить декомпозицию по подфункции Q :

$$\begin{aligned} f &= \Phi_1 \vee \Phi_2 Q, \quad f = \text{MX}(0, Q, 1, 1; \Phi_1, \Phi_2); \\ f &= \Phi_2 \vee \Phi_1 Q, \quad f = \text{MX}(0, 1, Q, 1; \Phi_1, \Phi_2); \\ f &= Q \vee \Phi_1 \Phi_2, \quad f = \text{MX}(Q, Q, Q, 1; \Phi_1, \Phi_2). \end{aligned}$$

Пример 6.5. Декомпозировать БФУ

$$f(x_1, x_2, x_3) = |0001 \ 0111|^T$$

по образу рассматриваемого модуля.

При $n - k = 2$ БФУ не может быть декомпозирована.
 При $n - k = 1$ используем первое уравнение для переменной x_3 :

$$f = \Phi_1 \vee \Phi_2 x_3;$$

$$\text{MX}(0, x_3, x_3, 1; x_1, x_2) = \text{MX}(0, x_3, 1, 1; \Phi_1, \Phi_2).$$

Это уравнение имеет два решения:

$$\begin{aligned}\Phi_1 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2; \\ \Phi_2 &= \text{MX}(0, 1, 1, 0; x_1, x_2) = x_1 \oplus x_2; \\ \Phi_1 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2; \\ \Phi_2 &= \text{MX}(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2.\end{aligned}$$

Таким образом, $f = x_1 x_2 \vee (x_1 \oplus x_2) x_3$ и $f = x_1 x_2 \vee (x_1 \vee x_2) x_3$, причем в обоих решениях $\Phi_1 = f(x_3 = 0)$ и только во втором — $\Phi_2 = f(x_3 = 1)$. Так как во втором решении обе функции двух переменных монотонны, то заданная БФУ может быть реализована схемой из трех модулей.

Пример 6.6. Декомпозировать БФУ

$$f(x_1, x_2, x_3, x_4) = |1111 \ 0110 \ 0110 \ 0000|^T,$$

немонотонную по переменной x_4 , по образцу рассматриваемого модуля.

Выполним декомпозицию по функции $Q = x_3 \oplus x_4$:

$$\begin{aligned}f(x_1, x_2, x_3, x_4) &= \Phi_1 \vee \Phi_2 Q; \\ \text{MX}(1, Q, Q, 0; x_1, x_2) &= \text{MX}(0, Q, 1, 1; \Phi_1, \Phi_2).\end{aligned}$$

Среди двух решений этого уравнения выберем следующее:

$$\begin{aligned}\Phi_1 &= \text{MX}(1, 0, 0, 0; x_1, x_2) = !x_1 !x_2; \\ \Phi_2 &= \text{MX}(1, 1, 1, 0; x_1, x_2) = !x_1 \vee !x_2.\end{aligned}$$

6.3. Реализация булевых функций схемами из WOS-модулей

Известны работы [5, 6], авторы которых пытались определить наиболее логически эффективные БФУ трех переменных.

Так, в [5] на основе исследования композиций таких БФУ сделан вывод, что наиболее эффективной является функция

$$F(a, b, c) = |0010 \ 1101|^T = a \oplus b!c.$$

Авторы этой работы не обратили внимания на то, что столбец значений указанной БФУ состоит из четырех типов фрагментов

длиной два, и не изложили метод реализации БФУ схемами из элементов, описываемых такой функцией.

В [7] был рассмотрен подход к реализации БФУ, заданных ТИ, на однотипных трехходовых элементах, у которых столбец значений функции, описывающей функционирование каждого из них, состоит из четырех типов фрагментов длиной два. Подход состоит из трех этапов: преобразование заданной ТИ в новую, учитывающую свойства используемого элемента; построение по новой ТИ канонической таблицы, специфической для данного элемента; построение по этой таблице искомой схемы.

Недостатки этого подхода состоят в том, что, во-первых, для каждого из существующих 24 типов трехходовых элементов, обладающих указанным выше свойством, приходится разрабатывать свой метод синтеза, а во-вторых, он не применим для других типов элементов. В [8] такой метод разработан применительно к WOS-модулям, предложенным в [6].

В [8] в качестве WOS-модуля рассматривается элемент, который описывается БФ вида:

$$F(a, b, c) = |1011\ 0100|^T = 1 \oplus a \oplus c \oplus bc = !a \oplus !bc.$$

Используем предлагаемый универсальный мультиплексорный метод применительно к синтезу схем на этих модулях, специфика которых определяется в результате анализа их функциональных возможностей.

1. В данном случае существуют $3! = 6$ порядков записи этой функции (табл. 6.1).

Т а б л и ц а 6.1

№	Порядок переменных	Транспонированный столбец значений	Фрагменты
1	a, b, c	1011 0100	$!c\ 1\ c\ 0$
2	a, c, b	1101 0010	$1\ b\ 0\ !b$
3	b, a, c	1001 1100	$!c\ c\ 1\ 0$
4	b, c, a	1001 1010	$!a\ a\ !a\ !a$
5	c, a, b	1100 0110	$1\ 0\ b\ !b$
6	c, b, a	1010 0110	$!a\ !a\ a\ !a$

2. Если каждой строке этой таблицы для фрагментов длиной один ($n - k = 0$) сопоставить переменные Φ_1, Φ_2, Φ_3 , то получим шесть декомпозиций заданной функции f :

$$\begin{aligned} 1. f &= !\Phi_1 \oplus !\Phi_2 \Phi_3; \quad 2. f = !\Phi_1 \oplus !\Phi_3 \Phi_2; \quad 3. f = !\Phi_2 \oplus !\Phi_1 \Phi_3; \\ 4. f &= !\Phi_3 \oplus !\Phi_1 \Phi_2; \quad 5. f = !\Phi_2 \oplus !\Phi_3 \Phi_1; \quad 6. f = !\Phi_3 \oplus !\Phi_2 \Phi_1. \end{aligned}$$

Каждая из этих декомпозиций может быть найдена для произвольной БФУ, однако нахождение каждой такой декомпозиции с определенными свойствами функций Φ_1 , Φ_2 и Φ_3 связано с огромным перебором вариантов, число которых равно $4^{l_0+l_1} = 4^{2^n}$.

3. Если каждой строке этой таблицы для фрагментов длиной два ($n - k = 1$) сопоставить переменные Φ_1 , Φ_2 , x_n , то получим шесть декомпозиций заданной функции f :

1. $f = !\Phi_1 \oplus !\Phi_2 x_n = \text{MX}(!x_n, 1, x_n, 0; \Phi_1, \Phi_2);$
2. $f = !\Phi_1 \oplus !x_n \Phi_2 = \text{MX}(1, x_n, 0, !x_n; \Phi_1, \Phi_2);$
3. $f = !\Phi_2 \oplus !\Phi_1 x_n = \text{MX}(!x_n, x_n, 1, 0; \Phi_1, \Phi_2);$
4. $f = !x_n \oplus !\Phi_1 \Phi_2 = \text{MX}(!x_n, x_n, !x_n, !x_n; \Phi_1, \Phi_2);$
5. $f = !\Phi_2 \oplus !x_n \Phi_1 = \text{MX}(1, 0, x_n, !x_n; \Phi_1, \Phi_2);$
6. $f = !x_n \oplus !\Phi_2 \Phi_1 = \text{MX}(!x_n, !x_n, x_n, !x_n; \Phi_1, \Phi_2).$

Для соотношений 1, 2, 3, 5 существует единственное решение (набор функций Φ_1 и Φ_2) для произвольной БФУ.

Соотношения 4 и 6 применимы для БФУ, столбцы значений которых состоят только из фрагментов $!x_n$ и x_n . Число решений в этих случаях равно 3^{l_2} . Для рассматриваемого модуля при $n - k = 0, 1$ нет необходимости иметь в наличии инверсные входные переменные.

4. Если каждой строке этой таблицы для фрагментов длиной четыре ($n - k = 2$) сопоставить переменные Φ , x_{n-1} , x_n , то получим шесть декомпозиций заданной функции f :

1. $f = !\Phi \oplus !x_{n-1} x_n = \text{MX}(x_{n-1} \vee !x_n, !x_{n-1} x_n; \Phi);$
2. $f = !\Phi \oplus !x_n x_{n-1} = \text{MX}(!x_{n-1} \vee x_n, x_{n-1} !x_n; \Phi);$
3. $f = !x_{n-1} \oplus !\Phi x_n = \text{MX}(!(x_{n-1} \oplus x_n), !x_{n-1}; \Phi);$
4. $f = !x_n \oplus !\Phi x_{n-1} = \text{MX}(!(x_{n-1} \oplus x_n), !x_n; \Phi);$
5. $f = !x_{n-1} \oplus !x_n \Phi = \text{MX}(!x_n, x_{n-1} \oplus x_n; \Phi);$
6. $f = !x_n \oplus !x_{n-1} \Phi = \text{MX}(!x_{n-1}, x_{n-1} \oplus x_n; \Phi).$

Каждое из этих соотношений может быть применено к соответствующему классу БФУ. Так, например, первое из них применимо к БФУ, столбцы значений которых образованы из фрагментов только двух типов:

$$Q_0 = |1011|^T = x_{n-1} \vee !x_n, \quad Q_1 = |0100|^T = !x_{n-1} x_n.$$

Для каждого из этих классов решение, если оно существует, единственно.

Отметим, что i -е соотношение при $n - k = 2$ содержит в том числе и все решения, которые могут быть получены для i -х соотношений при $n - k = 1, 2$. При этом i -е соотношение при $n - k = 1$ содержит в том числе и все решения, которые могут быть получены для i -го соотношения при $n - k = 2$.

При $n - k = 2$ и доступности прямых и инверсных входных переменных нет необходимости запоминать разрешенные фрагменты длиной четыре, а вместо этого достаточно проверить два условия:

— число типов фрагментов в столбце значений должно быть равно двум;

— должно выполняться одно из трех соотношений:

$$Q_0! \Phi \vee Q_1 \Phi = ! \Phi \oplus \tilde{x}_{n-1} \tilde{x}_n;$$

$$Q_0! \Phi \vee Q_1 \Phi = ! x_{n-1} \oplus \tilde{\Phi} \tilde{x}_n;$$

$$Q_0! \Phi \vee Q_1 \Phi = ! x_n \oplus \tilde{\Phi} \tilde{x}_{n-1}.$$

5. Из соотношений при $n - k = 1$ могут быть получены также шесть соотношений для проведения декомпозиций по функции $Q(X_1)$:

$$1. f = ! \Phi_1 \oplus ! \Phi_2 Q = \text{MX}(!Q, 1, Q, 0; \Phi_1, \Phi_2);$$

$$2. f = ! \Phi_1 \oplus ! Q! \Phi_2 = \text{MX}(1, Q, 0, !Q; \Phi_1, \Phi_2);$$

$$3. f = ! \Phi_2 \oplus ! \Phi_1 Q = \text{MX}(!Q, Q, 1, 0; \Phi_1, \Phi_2);$$

$$4. f = ! Q \oplus ! \Phi_1 \Phi_2 = \text{MX}(!Q, Q, !Q, !Q; \Phi_1, \Phi_2);$$

$$5. f = ! \Phi_2 \oplus ! Q \Phi_1 = \text{MX}(1, 0, Q, !Q; \Phi_1, \Phi_2);$$

$$6. f = ! Q \oplus ! \Phi_2 \Phi_1 = \text{MX}(!Q, !Q, Q, !Q; \Phi_1, \Phi_2).$$

В заключение отметим, что при синтезе на каждом шаге декомпозиция должна выполняться начиная с максимальной величины $n - k$, что позволяет сократить перебор.

Пример 6.7. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |1001 \ 1010 \ 1010 \ 1001|^T$$

схемой из WOS-модулей.

Столбец значений этой БФУ состоит из двух типов фрагментов длиной четыре: $Q_0 = !(x_3 \oplus x_4)$, $Q_1 = !x_4$.

Используя четвертое соотношение при $n - k = 2$, составим и решим уравнение:

$$f = !x_4 \oplus !\Phi x_3;$$

$$\text{MX}(Q_0, Q_1, Q_1, Q_0; x_1, x_2) = \text{MX}(Q_0, Q_1; \Phi);$$

$$\Phi = \text{MX}(0, 1, 1, 0; x_1, x_2).$$

Для реализации функции Φ , применяя четвертое соотношение при $n - k = 1$, составим и решим уравнение:

$$\begin{aligned}\Phi &= !x_2 \oplus !\Phi_1 \Phi_2; \\ \text{MX}(x_2, !x_2; x_1) &= \text{MX}(!x_2, x_2, x_2, x_2; \Phi_1, \Phi_2); \\ \Phi_1 &= \text{MX}(0, 0; x_1) = 0; \\ \Phi_2 &= \text{MX}(1, 0; x_1) = !x_1.\end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из двух WOS-модулей:

$$\begin{aligned}a &= x_2, \quad b = 0, \quad c = !x_1, \quad \Phi = !x_2 \oplus !x_1 = x_1 \oplus x_2; \\ a &= x_4, \quad b = \Phi, \quad c = x_3, \quad f = !x_4 \oplus !\Phi x_3 = (\Phi \vee !x_3) \oplus x_4 = \\ &= ((x_1 \oplus x_2) \vee !x_3) \oplus x_4.\end{aligned}$$

Пример 6.8. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0101 \ 0110 \ 0110 \ 0101|^T$$

схемой из WOS-модулей.

Столбец значений этой БФУ состоит из двух типов фрагментов длиной четыре: $Q_0 = x_4$, $Q_1 = x_3 \oplus x_4$. Эти фрагменты являются разрешенными, так как выполняется соотношение

$$x_4 !\Phi \vee (x_3 \oplus x_4) \Phi = x_4 \oplus x_3 \Phi.$$

Составим и решим уравнение:

$$\begin{aligned}f &= x_4 \oplus x_3 \Phi; \\ \text{MX}(x_4, x_3 \oplus x_4, x_3 \oplus x_4, x_4; x_1, x_2) &= \text{MX}(x_4, x_3 \oplus x_4; \Phi); \\ \Phi &= \text{MX}(0, 1, 1, 0; x_1, x_2).\end{aligned}$$

Реализация функции Φ приведена в предыдущем примере. Таким образом, заданная БФУ реализуется схемой из двух WOS-модулей:

$$\begin{aligned}a &= !x_1, \quad b = 0, \quad c = x_2, \quad \Phi = x_1 \oplus x_2; \\ a &= !x_4, \quad b = !x_3, \quad c = \Phi, \quad f = !x_4 \oplus x_3 \Phi = (x_1 \oplus x_2) x_3 \oplus x_4.\end{aligned}$$

Пример 6.9. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |1001 \ 1010 \ 1010 \ 0101|^T$$

схемой из WOS-модулей.

Столбец значений этой БФУ при $n - k = 2$ не реализуется, так как он содержит три типа фрагментов длиной четыре. Выбирая первое соотношение при $n - k = 1$, составим и решим уравнение:

$$\begin{aligned} f &= !\Phi_1 \oplus !\Phi_2 x_4; \\ \text{MX}(!x_4, x_4, !x_4, !x_4, !x_4, !x_4, x_4, x_4; x_1, x_2, x_3) &= \\ &= \text{MX}(!x_4, 1, x_4, 0; \Phi_1, \Phi_2); \\ \Phi_1 &= \text{MX}(0, 1, 0, 0, 0, 0, 1, 1; x_1, x_2, x_3); \\ \Phi_2 &= \text{MX}(0, 0, 0, 0, 0, 0, 0, 0; x_1, x_2, x_3) = 0. \end{aligned}$$

Для реализации функции Φ_1 вновь выберем первое соотношение при $n - k = 1$:

$$\begin{aligned} \Phi_1 &= !\Phi_{11} \oplus !\Phi_{12} x_3; \\ \text{MX}(x_3, 0, 0, 1; x_1, x_2) &= \text{MX}(!x_3, 1, x_3, 0; \Phi_{11}, \Phi_{12}); \\ \Phi_{11} &= \text{MX}(1, 1, 1, 0; x_1, x_2); \\ \Phi_{12} &= \text{MX}(0, 1, 1, 1; x_1, x_2). \end{aligned}$$

Декомпозируем функцию Φ_{11} :

$$\begin{aligned} \Phi_{11} &= !\Phi_{111} \oplus !\Phi_{112} x_2; \\ \text{MX}(1, !x_2; x_1) &= \text{MX}(!x_2, 1, x_2, 0; \Phi_{111}, \Phi_{112}); \\ \Phi_{111} &= \text{MX}(0, 0; x_1) = 0; \\ \Phi_{112} &= \text{MX}(1, 0; x_1) = !x_1. \end{aligned}$$

Декомпозируем функцию Φ_{12} :

$$\begin{aligned} \Phi_{12} &= !\Phi_{121} \oplus !\Phi_{122} x_2; \\ \text{MX}(x_2, 1; x_1) &= \text{MX}(!x_2, 1, x_2, 0; \Phi_{121}, \Phi_{122}); \\ \Phi_{121} &= \text{MX}(1, 0; x_1) = !x_1; \\ \Phi_{122} &= \text{MX}(0, 1; x_1) = x_1. \end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из четырех WOS-модулей:

$$\begin{aligned} a &= !x_1, \quad b = x_1, \quad c = x_2, \quad \Phi_{12} = x_1 \oplus !x_1 x_2 = x_1 \vee x_2; \\ a &= 0, \quad b = !x_1, \quad c = x_2, \quad \Phi_{11} = 1 \oplus x_1 x_2 = !x_1 \vee !x_2; \\ a &= \Phi_{11}, \quad b = \Phi_{12}, \quad c = x_3, \quad \Phi_1 = !\Phi_{11} \oplus !\Phi_{12} x_3 = x_1 x_2 \vee !x_1 !x_2 x_3; \\ a &= \Phi_1, \quad b = 0, \quad c = x_4, \quad f = !\Phi_1 \oplus (!0) x_4 = \\ &= (!x_1 \vee !x_2)(x_1 \vee x_2 \vee !x_3) \oplus x_4. \end{aligned}$$

Пример 6.10. Декомпозировать мажоритарную функцию трех переменных $f(x_1, x_2, x_3) = |0001\ 0111|^T$ по образцу WOS-модулей.

Используя первое соотношение при $n - k = 1$, составим и решим уравнение:

$$\begin{aligned} f &= !\Phi_1 \oplus !\Phi_2 x_3; \\ \text{MX}(0, x_3, x_3, 1; x_1, x_2) &= \text{MX}(!x_3, 1, x_3, 0; \Phi_1, \Phi_2); \\ \Phi_1 &= \text{MX}(1, 1, 1, 0; x_1, x_2) = !x_1 \vee !x_2; \\ \Phi_2 &= \text{MX}(1, 0, 0, 1; x_1, x_2) = !(x_1 \oplus x_2). \end{aligned}$$

Таким образом, $f = x_1 x_2 \oplus (x_1 \oplus x_2) x_3$.

Пример 6.11. Реализовать БФУ [9]

$$f(x_1, x_2, x_3, x_4) = |0111\ 1011\ 0111\ 0111|^T$$

схемой из WOS-модулей.

Столбец значений этой БФУ при $n - k = 2$ не реализуется, так как он содержит три типа фрагментов длиной четыре. Выбирая первое соотношение при $n - k = 1$, составим и решим уравнение:

$$\begin{aligned} f &= !\Phi_1 \oplus !\Phi_2 x_4; \\ \text{MX}(x_4, 1, !x_4, 1, x_4, 1, x_4, 1; x_1, x_2, x_3) &= \\ &= \text{MX}(!x_4, 1, x_4, 0; \Phi_1, \Phi_2); \\ \Phi_1 &= \text{MX}(1, 0, 0, 0, 1, 0, 1, 0; x_1, x_2, x_3); \\ \Phi_2 &= \text{MX}(0, 1, 0, 1, 0, 1, 0, 1; x_1, x_2, x_3) = x_3. \end{aligned}$$

Для функции Φ_1 также не могут быть применены соотношения при $n - k = 2$. Вновь выберем первое соотношение при $n - k = 1$:

$$\begin{aligned} \Phi_1 &= !\Phi_{11} \oplus !\Phi_{12} x_3; \\ \text{MX}(!x_3, 0, !x_3, !x_3; x_1, x_2) &= \text{MX}(!x_3, 1, x_3, 0; \Phi_{11}, \Phi_{12}); \\ \Phi_{11} &= \text{MX}(0, 1, 0, 0; x_1, x_2); \\ \Phi_{12} &= \text{MX}(0, 1, 0, 0; x_1, x_2). \end{aligned}$$

Используем первое соотношение при $n - k = 2$ для функций $\Phi_{11} = \Phi_{12} = \Phi$:

$$\begin{aligned} \Phi &= !\Phi_0 \oplus !x_1 x_2; \\ \text{MX}(!x_1 x_2, !x_1 x_2; 0) &= \text{MX}(x_1 \vee !x_2, !x_1 x_2; \Phi_0); \\ \Phi_0 &= \text{MX}(1, 1; 0) = 1. \end{aligned}$$

Таким образом, предлагаемый универсальный метод строит схему из трех WOS-модулей, найденную в [9] с помощью весьма специфичного метода:

$$\begin{aligned} a &= 1, \quad b = x_1, \quad c = x_2, \quad \Phi = !x_1 x_2; \\ a &= \Phi, \quad b = \Phi, \quad c = x_3, \quad \Phi_1 = !\Phi !x_3 = (x_1 \vee !x_2) !x_3; \\ a &= \Phi_1, \quad b = x_3, \quad c = x_4, \quad f = !\Phi_1 \oplus !x_3 x_4 = \\ &= (!x_1 x_2 \vee x_3) \oplus !x_3 x_4 = (!x_1 x_2 \oplus x_4) \vee x_3. \end{aligned}$$

Из полученной БФ следует, что она монотонна по x_3 :

$$f(x_1, x_2, x_4, x_3) = |0111 \ 1101 \ 0111 \ 0111|^T.$$

Эта функция содержит два типа фрагментов длиной четыре, но они не являются разрешенными, так как функция $(x_3 \vee x_4) !\Phi \vee \vee (!x_3 \vee x_4) \Phi$ не сводится к желаемому образу.

Применяя второе соотношение при $n - k = 1$, составим и решим уравнение:

$$\begin{aligned} f &= !\Phi_1 \oplus !x_3 \Phi_2; \\ \text{MX}(x_3, 1, 1, x_3, x_3, 1, x_3, 1; x_1, x_2, x_4) &= \\ &= \text{MX}(1, x_3, 0, !x_3; \Phi_1, \Phi_2); \\ \Phi_1 &= \text{MX}(0, 0, 0, 0, 0, 0, 0, 0; x_1, x_2, x_4) = 0; \\ \Phi_2 &= \text{MX}(1, 0, 0, 1, 1, 0, 1, 0; x_1, x_2, x_4) = x_3. \end{aligned}$$

Для функции Φ_2 выполняется четвертое соотношение при $n - k = 2$:

$$\begin{aligned} \Phi_2 &= !x_4 \oplus !\Phi x_2; \\ \text{MX}(!x_2 \oplus x_4, !x_4; x_1) &= \text{MX}(!x_2 \oplus x_4, !x_4; \Phi); \\ \Phi &= \text{MX}(0, 1; x_1) = x_1. \end{aligned}$$

Таким образом, заданная БФУ с помощью предлагаемого метода может быть реализована схемой всего лишь из $2 \times 4 = 8$ WOS-модулей:

$$\begin{aligned} a &= x_4, \quad b = x_1, \quad c = x_2, \quad \Phi_2 = !x_1 x_2 \oplus !x_4; \\ a &= 0, \quad b = x_3, \quad c = \Phi_2, \quad f = 1 \oplus !x_3 (!x_1 x_2 \oplus !x_4) = \\ &= (!x_1 x_2 \oplus x_4) \vee x_3. \end{aligned}$$

Из приведенного примера следует, что и для данного типа модулей, реализующих произвольные БФУ при наличии только безынверсных входных переменных, использование свойства их монотонности по одной переменной обеспечило упрощение схемы.

6.4. Реализация булевых функций схемами из модулей, универсальных в классе пороговых функций трех переменных

В разд. 7.2.4 построена порождающая формула (ПФ) такого модуля, которая при равной доступности прямых и инверсных входных переменных имеет вид:

$$F = z_1 (z_2 \vee x_2 \vee x_3) \vee z_2 x_2 x_3.$$

При этом показано, что $z_1 = \{0, x_1, 1\}$, $z_2 = \{0, !x_1, x_1, 1\}$.

Так как этот модуль также универсален в классе произвольных БФ в базисе $\{\&, \vee, !\}$ из трех букв и в его ПФ объединены представители типов положительно монотонных бесповторных БФ в этом базисе, то число таких модулей, требующихся для реализации одноэтапным формульным методом произвольной БФ в указанном базисе из h букв, удовлетворяет соотношению

$$\left\lceil \frac{h-1}{2} \right\rceil \leq L(h, 3) \leq \left\lceil \frac{2(h-1)}{3} \right\rceil.$$

Из этого соотношения, в частности, следует, что $2 \leq L(5, 3) \leq 3$.

Для более эффективного использования этих модулей их настройка должна осуществляться не только константами, переменными и инверсиями переменных, но и функциями. В общем случае эти функции могут подаваться на любые входы модуля.

Ниже будет показано, что это достигается при применении мультиплексорного метода, который, в частности, обеспечивает $L(5, 3) = 2$ (пример 3.22).

Этот результат позволяет более эффективно реализовать БФ, которые формульным методом реализуются по приведенной выше верхней оценке. Так, например, при $h = 16$ справедливо соотношение $8 \leq L(16, 3) \leq 10$, а выражение $f = (x_1 x_2 \vee x_3 x_4) (x_5 x_6 \vee x_7 x_8) \vee (x_9 x_{10} \vee x_{11} x_{12}) (x_{13} x_{14} \vee x_{15} x_{16})$ реализуется формульным методом, схемой из десяти модулей. Совместное использование формульного и мультиплексорного методов позволяет в данном случае построить схему из девяти модулей.

Изменим обозначения входных переменных в порождающей формуле модуля [10—12]:

$$F(a, b, c, d) = a(b \vee c \vee d) \vee bcd = \\ = |0000 \ 0001 \ 0111 \ 1111|^T = |0 \ 1 \ 7 \ 15|^T.$$

Покажем, что более глубокое исследование функциональных возможностей модуля позволяет его настраивать на реализацию бо-

лее широкого класса функций по сравнению с заложенным при его проектировании.

1. На первый взгляд кажется, что модуль реализует только одну БФУ четырех переменных, которая приведена выше. Однако это не так, если учесть возможность переобозначения входных переменных и подачи на входы модуля не только переменных, но и их инверсий.

Для определения БФУ, порождаемых за счет переобозначения переменных, построим для ПФ модуля $4! = 24$ таблицы истинности, каждой из которых соответствует определенный порядок переменных a, b, c, d . Результаты этого построения объединены в табл. 6.2, в которой фрагменты длиной четыре транспонированных столбцов значений заданы десятичными эквивалентами.

Т а б л и ц а 6.2

№	Порядок переменных	Столбец значений	Тип столбца	№	Порядок переменных	Столбец значений	Тип столбца
1	a, b, c, d	0 1 7 15	1	13	c, a, b, d	0 7 1 15	2
2	a, b, d, c	0 1 7 15	1	14	c, a, d, b	0 7 1 15	2
3	a, c, b, d	0 1 7 15	1	15	c, b, a, d	1 3 3 7	3
4	a, c, d, b	0 1 7 15	1	16	c, b, d, a	1 5 5 7	4
5	a, d, b, c	0 1 7 15	1	17	c, d, a, b	1 3 3 7	3
6	a, d, c, b	0 1 7 15	1	18	c, d, b, a	1 5 5 7	4
7	b, a, c, d	0 7 1 15	2	19	d, a, b, c	0 7 1 15	2
8	b, a, d, c	0 7 1 15	2	20	d, a, c, b	0 7 1 15	2
9	b, c, a, d	1 3 3 7	3	21	d, b, a, c	1 3 3 7	3
10	b, c, d, a	1 5 5 7	4	22	d, b, c, a	1 5 5 7	4
11	b, d, a, c	1 3 3 7	3	23	d, c, a, b	1 3 3 7	3
12	b, d, c, a	1 5 5 7	4	24	d, c, b, a	1 5 5 7	4

Если еще раз изменить обозначение переменных и зафиксировать порядок расположения новых переменных, то каждой из этих таблиц истинности будет соответствовать своя функция (тип декомпозиций) четырех переменных.

Пусть при $n - k = 0$ эта БФУ зависит от переменных $\Phi_1, \Phi_2, \Phi_3, \Phi_4$, при $n - k = 1$ — от переменных $\Phi_1, \Phi_2, \Phi_3, x_n$, при $n - k = 2$ — от переменных $\Phi_1, \Phi_2, x_{n-1}, x_n$, а при $n - k = 3$ — от переменных $\Phi, x_{n-2}, x_{n-1}, x_n$.

2. При $n - k = 0$ поиск решений с требуемыми свойствами связан с огромным перебором, и поэтому в дальнейшем этот случай не рассматривается.

3. Заданный модуль не универсален при $n - k = 1$, так как ни одна из строк этой таблицы не содержит всех четырех типов фрагментов

длиной два. Так, например, строке 1 (табл. 6.2) соответствуют декомпозиции вида:

$$\begin{aligned} f &= MX(0, 0, 0, x_n, x_n, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3) = \\ &= \Phi_1(\Phi_2 \vee \Phi_3 \vee x_n) \vee \Phi_2 \Phi_3 x_n; \end{aligned}$$

строке 7 — декомпозиции вида:

$$\begin{aligned} f &= MX(0, 0, x_n, 1, 0, x_n, 1, 1; \Phi_1, \Phi_2, \Phi_3) = \\ &= \Phi_2(\Phi_1 \vee \Phi_3 \vee x_n) \vee \Phi_1 \Phi_3 x_n; \end{aligned}$$

строке 9 — декомпозиции вида:

$$\begin{aligned} f &= MX(0, x_n, 0, 1, 0, 1, x_n, 1; \Phi_1, \Phi_2, \Phi_3) = \\ &= \Phi_3(\Phi_1 \vee \Phi_2 \vee x_n) \vee \Phi_1 \Phi_2 x_n; \end{aligned}$$

строке 10 — декомпозиции вида:

$$\begin{aligned} f &= MX(0, x_n, x_n, x_n, x_n, x_n, x_n, 1; \Phi_1, \Phi_2, \Phi_3) = \\ &= (\Phi_1 \vee \Phi_2 \vee \Phi_3) x_n \vee \Phi_1 \Phi_2 \Phi_3. \end{aligned}$$

Первые три соотношения при их выполнении порождают $3^{t_0+t_1} \cdot 2^{t_2}$ декомпозиций, а четвертое — 6^{t_2} декомпозиций. Например, для БФУ $f = |0101\ 0011|^T$ все четыре соотношения порождают по 36 декомпозиций, а для БФУ $f_1 = |0101\ 0111|^T$ первые три соотношения порождают по 24 декомпозиции, а четвертое — 216 декомпозиций.

Каждое из приведенных соотношений содержит по три одинаковых типа фрагментов $\{0, x_n, 1\}$, и поэтому каждое из них может использоваться при декомпозиции произвольной БФУ, положительно монотонной по переменной x_n . Для заданной БФУ, принадлежащей указанному классу, из четырех приведенных выше соотношений должно выбираться то, которое порождает наиболее простые функции Φ_1, Φ_2, Φ_3 . Например, простейшую декомпозицию для БФУ $f_1(x_1, x_2, x_3)$ обеспечивает четвертое соотношение:

$$\Phi_1 = x_1, \Phi_2 = x_2, \Phi_3 = 1, f_1 = (\Phi_1 \vee \Phi_2 \vee \Phi_3) x_3 \vee \Phi_1 \Phi_2 \Phi_3.$$

Класс декомпозируемых функций при $n - k = 1$ может быть расширен в случае, если можно применять инверсию переменной x_n . При этом может декомпонироваться каждая БФУ, столбец значений которой образован фрагментами $(0, !x_n, 1)$, а во всех приведенных декомпозициях символ x_n должен быть заменен на символ $!x_n$.

Таким образом, модуль позволяет декомпонировать булевы функции, монотонные по переменной x_n или $!x_n$.

Для декомпозиции функций, не монотонных по этой переменной, можно построить БФ и по ней определить, имеет ли она монотонную переменную. При наличии такой переменной она должна использоваться в ТИ в качестве крайней правой.

4. При $n - k = 2$ строке 1 (табл. 6.2) соответствуют декомпозиции вида:

$$\begin{aligned} f &= \text{MX}(0, x_{n-1}x_n, x_{n-1} \vee x_n, 1; \Phi_1, \Phi_2) = \\ &= \Phi_1(\Phi_2 \vee x_{n-1} \vee x_n) \vee \Phi_2 x_{n-1}x_n; \end{aligned}$$

строке 7 — декомпозиции вида:

$$\begin{aligned} f &= \text{MX}(0, x_{n-1} \vee x_n, x_{n-1}x_n, 1; \Phi_1, \Phi_2) = \\ &= \Phi_2(\Phi_1 \vee x_{n-1} \vee x_n) \vee \Phi_1 x_{n-1}x_n; \end{aligned}$$

строке 9 — декомпозиции вида:

$$\begin{aligned} f &= \text{MX}(x_{n-1}x_n, x_{n-1}, x_{n-1}, x_{n-1} \vee x_n; \Phi_1, \Phi_2) = \\ &= x_{n-1}(\Phi_1 \vee \Phi_2 \vee x_n) \vee \Phi_1 \Phi_2 x_n; \end{aligned}$$

строке 10 — декомпозиции вида:

$$\begin{aligned} f &= \text{MX}(x_{n-1}x_n, x_n, x_n, x_{n-1} \vee x_n; \Phi_1, \Phi_2) = \\ &= (\Phi_1 \vee \Phi_2 \vee x_{n-1})x_n \vee \Phi_1 \Phi_2 x_{n-1}. \end{aligned}$$

Первым двум соотношениям соответствуют одинаковые фрагменты, и если решение существует, то оно единственно. Третье соотношение может породить 2^{t_3} решений (t_3 — число фрагментов x_{n-1} в булевой функции f), а четвертое — 2^{t_2} решений.

Из изложенного следует, что рассматриваемый модуль позволяет декомпозировать три класса функций, каждый из которых определяется следующей номенклатурой фрагментов длиной четыре в их столбцах значений: $\{0, x_{n-1}x_n, x_{n-1} \vee x_n, 1\}$, $\{x_{n-1}, x_{n-1}x_n, x_{n-1} \vee x_n\}$, $\{x_n, x_{n-1}x_n, x_{n-1} \vee x_n\}$, причем в каждом из этих классов могут быть выделены подклассы, каждый из которых определяется только некоторыми фрагментами из соответствующего множества фрагментов.

Отметим, что при построении ПФ этого модуля применялись первое соотношение и фрагменты только первого класса. Таким образом, модуль позволяет декомпозировать более широкий, чем предполагалось при его проектировании, класс БФУ при $n - k = 2$, которые могут быть реализованы схемами, содержащими меньшее число модулей.

Если кроме переменных x_{n-1} и x_n доступны также и их инверсии, то каждое из приведенных соотношений порождает еще по три

монотонных по этим переменным соотношения. Например, первое из них порождает соотношения:

$$\begin{aligned}
f &= \text{MX}(0, !x_{n-1}x_n, !x_{n-1} \vee x_n, 1; \Phi_1, \Phi_2) = \\
&= \Phi_1(\Phi_2 \vee !x_{n-1} \vee x_n) \vee \Phi_2 !x_{n-1}x_n; \\
f &= \text{MX}(0, x_{n-1}!x_n, x_{n-1} \vee !x_n, 1; \Phi_1, \Phi_2) = \\
&= \Phi_1(\Phi_2 \vee x_{n-1} \vee !x_n) \vee \Phi_2 x_{n-1}!x_n; \\
f &= \text{MX}(0, !x_{n-1}!x_n, !x_{n-1} \vee !x_n, 1; \Phi_1, \Phi_2) = \\
&= \Phi_1(\Phi_2 \vee !x_{n-1} \vee !x_n) \vee \Phi_2 !x_{n-1}!x_n.
\end{aligned}$$

Таким образом, при использовании рассматриваемого модуля по двум последним переменным, x_{n-1} и x_n , только БФУ, которые содержат фрагменты $x_{n-1} \oplus x_n$ и/или $!(x_{n-1} \oplus x_n)$, не могут быть разложены.

5. При $n - k = 3$ возможны четыре соотношения:

$$\begin{aligned}
f &= \text{MX}(x_{n-2}x_{n-1}x_n, x_{n-2} \vee x_{n-1} \vee x_n; \Phi) = \\
&= \Phi(x_{n-2} \vee x_{n-1} \vee x_n) \vee x_{n-2}x_{n-1}x_n; \\
f &= \text{MX}(x_{n-2}, x_{n-1} \vee x_n, x_{n-2} \vee x_{n-1}x_n; \Phi) = \\
&= x_{n-2}(x_{n-1} \vee x_n \vee \Phi) \vee x_{n-1}x_n\Phi; \\
f &= \text{MX}(x_{n-1}(x_{n-2} \vee x_n), x_{n-1} \vee x_{n-2}x_n; \Phi) = \\
&= x_{n-1}(x_{n-2} \vee x_n \vee \Phi) \vee x_{n-2}x_n\Phi; \\
f &= \text{MX}((x_{n-2} \vee x_{n-1})x_n, x_{n-2}x_{n-1} \vee x_n; \Phi) = \\
&= x_n(x_{n-2} \vee x_{n-1} \vee \Phi) \vee x_{n-2}x_{n-1}\Phi,
\end{aligned}$$

каждое из которых при равной доступности прямых и инверсных входных переменных порождает по семь соответствующих «монотонных» соотношений.

6. Кроме рассмотренных вариантов разложения БФУ по одной, двум и трем переменным, на основе соотношений для $n - k = 1$ при замене $x_n = Q$ могут быть построены соотношения для проведения разложений по подфункции Q . Это позволяет расширить класс БФУ, декомпозируемых по образу рассматриваемого модуля. Например, если БФУ при $n - k = 2$ содержит три типа фрагментов $\{0, x_{n-1} \oplus x_n, 1\}$, то такая функция может быть декомпозирована, в то время как разложение по двум переменным, x_{n-1} и x_n , не может быть выполнено.

7. Из изложенного следует, что рассматриваемый модуль при проведении декомпозиции мультиплексорным методом «сверху вниз» не позволяет декомпонировать при $n - k > 0$ произвольную БФУ, в то время как при ее проведении формульным методом «сни-

зу вверх» произвольная БФУ всегда может быть декомпозирована. Это объясняется тем, что при применении формульного метода производится изменение метрики: вместо числа переменных n используется число букв h .

Из изложенного также следует, что в тех случаях, когда мультиплексорный метод неприменим, а именно:

при $n - k = 0$ — из-за огромного объема перебора;

при $n - k > 0$ — из-за отсутствия в реализуемой БФУ «разрешенных» фрагментов;

при $n > 5$ — из-за громоздкости табличного определения функций Φ_j бывает целесообразно при ручном синтезе применять формульный метод для определения остаточных функций, которые могут быть реализованы мультиплексорным методом.

Пример 6.12. Определить с помощью мультиплексорного метода настройку модуля на реализацию функции $f = x_1 \# x_2 \# x_3$.

Используя первое соотношение при $n - k = 2$, составим и решим уравнение:

$$\begin{aligned} x_1 \# x_2 \# x_3 &= \Phi_1 (\Phi_2 \vee x_2 \vee x_3) \vee \Phi_2 x_2 x_3; \\ \text{MX}(x_2 x_3, x_2 \vee x_3; x_1) &= \text{MX}(0, x_2 x_3, x_2 \vee x_3, 1; \Phi_1, \Phi_2); \\ \Phi_1 &= \text{MX}(0, 1; x_1) = x_1; \\ \Phi_2 &= \text{MX}(1, 0; x_1) = !x_1. \end{aligned}$$

Таким образом, $a = x_1$, $b = !x_1$, $c = x_2$, $d = x_3$.

Еще более простое решение, которое не было заложено при проектировании модуля, может быть получено из четвертого соотношения при $n - k = 1$:

$$\begin{aligned} x_1 \# x_2 \# x_3 &= (\Phi_1 \vee \Phi_2 \vee \Phi_3) x_3 \vee \Phi_1 \Phi_2 \Phi_3; \\ \text{MX}(0, x_3, x_3, 1; x_1, x_2) &= \\ &= \text{MX}(0, x_3, x_3, x_3, x_3, x_3, x_3, 1; \Phi_1, \Phi_2, \Phi_3). \end{aligned}$$

Среди 36 решений этого уравнения имеется следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1; \\ \Phi_2 &= \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1; \\ \Phi_3 &= \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2. \end{aligned}$$

Таким образом, $a = x_3$, $b = x_1$, $c = x_1$, $d = x_2$ [11, 12].

Пример 6.13. Реализовать схемой из рассматриваемых модулей мажоритарную функцию пяти переменных

$$\begin{aligned} f(x_1, x_2, x_3, x_4, x_5) &= \text{MX}(0, x_4 x_5, x_4 x_5, x_4 \vee \\ &\vee x_5, x_4 x_5, x_4 \vee x_5, x_4 \vee x_5, 1; x_1, x_2, x_3). \end{aligned}$$

Применяя первое соотношение при $n - k = 2$, составим и решим уравнение:

$$\begin{aligned} f(x_1, x_2, x_3, x_4, x_5) &= \Phi_1(\Phi_2 \vee x_4 \vee x_5) \vee \Phi_2 x_4 x_5; \\ \text{MX}(0, x_4 x_5, x_4 x_5, x_4 \vee x_5, x_4 x_5, x_4 \vee x_5, x_4 \vee x_5, 1; x_1, x_2, x_3) &= \text{MX}(0, x_4 x_5, x_4 \vee x_5, 1; \Phi_1, \Phi_2); \\ \Phi_1 &= \text{MX}(0, 0, 0, 1, 0, 1, 1, 1; x_1, x_2, x_3) = x_1 \# x_2 \# x_3; \\ \Phi_2 &= \text{MX}(0, 1, 1, 0, 1, 0, 0, 1; x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3. \end{aligned}$$

Отметим, что полученное решение описывает двоичный одноразрядный сумматор.

Так как функция Φ_1 реализуется модулем путем настройки, а функция Φ_2 может быть представлена в виде:

$$\Phi_2 = !\Phi_1(x_1 \vee x_2 \vee x_3) \vee x_1 x_2 x_3,$$

то заданная БФУ реализуется схемой из трех модулей, один из которых должен иметь прямой и инверсный выходы.

6.5. Реализация булевых функций схемами из 3-универсальных немонотонных модулей

Незначительно усложним порождающую функцию модуля, универсального в классе пороговых функций трех переменных:

$$F(a, b, c, d) = a(!b \vee c \vee !d) \vee bcd = |0000 \ 0001 \ 1111 \ 1011|^T.$$

При этом сохраняется возможность реализации путем настройки представителей всех PN -типов формул в базисе $\{\&, \vee, !\}$ из трех букв:

$$\begin{aligned} a = 0 \quad F &= bcd, & a = !b \quad F &= cd \vee !b, \\ a = 1 \quad F &= !b \vee c \vee !d, & a = b \quad F &= (c \vee !d)b, \end{aligned}$$

а вместо реализации путем настройки мажоранты трех переменных появляется возможность разложения произвольной БФУ по крайней правой входной переменной в ее таблице истинности, так как

$$\begin{aligned} F(\Phi_1, \Phi_2, \Phi_3, x_n) &= \Phi_1(!\Phi_2 \vee \Phi_3 \vee !x_n) \vee \Phi_2 \Phi_3 x_n = \\ &= \text{MX}(0, 0, 0, x_n, 1, 1, !x_n, 1; \Phi_1, \Phi_2, \Phi_3). \end{aligned}$$

Модуль также обеспечивает реализацию разложения Шеннона по одной переменной, так как при $b = 1$ он выполняет функцию мультиплексирования $F = a!d \vee cd$.

Следовательно, этот модуль позволяет проводить синтез «сверху вниз» с помощью разложения по одной входной переменной, а также «снизу вверх» с помощью двухэтапного формульного метода, так как приведенные выше представители PN -типов содержат инверсии. Следовательно, в данном случае

$$L(h, n) \leq \left(\left\lceil \frac{2(h-1)}{3} \right\rceil, 2^{n-1} - 1 \right).$$

Не выполняя полного исследования функциональных возможностей модуля, приведем пример, демонстрирующий эффективность мультиплексорного метода.

Пример 6.14. Реализовать БФУ

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= |0000 \ 1011 \ 0001 \ 1111|^T = \\ &= (x_1 \vee x_3 \vee !x_4) x_2 \vee x_1 x_3 x_4. \end{aligned}$$

При $n - k = 1$ составим уравнение:

$$\begin{aligned} &(x_1 \vee x_3 \vee !x_4) x_2 \vee x_1 x_3 x_4 = \\ &= \Phi_1 (!\Phi_2 \vee \Phi_3 \vee !x_4) \vee \Phi_2 \Phi_3 x_4; \\ &MX(0, 0, !x_4, 1, 0, x_4, 1, 1; x_1, x_2, x_3) = \\ &= MX(0, 0, 0, x_4, 1, 1, !x_4, 1; \Phi_1, \Phi_2, \Phi_3); \end{aligned}$$

Среди $3^3 \cdot 3^3 = 729$ решений этого уравнения имеется следующее:

$$\begin{aligned} \Phi_1 &= MX(0, 0, 1, 1, 0, 0, 1, 1; x_1, x_2, x_3) = x_2; \\ \Phi_2 &= MX(0, 0, 1, 0, 0, 1, 1, 1; x_1, x_2, x_3) = x_1 x_3 \vee x_2 !x_3; \\ \Phi_3 &= MX(0, 0, 0, 0, 1, 1, 1, 1; x_1, x_2, x_3) = x_1. \end{aligned}$$

Функция Φ_2 выполняется модулем путем настройки при $a = x_2$, $b = 1$, $c = x_1$, $d = x_3$.

Таким образом, заданная БФУ реализуется схемой из двух модулей. При этом у первого модуля все входы используются как информационные ($a = x_2$, $b = \Phi_2$, $c = x_1$, $d = x_4$), а у второго только один вход является настроечным.

При применении двухэтапного формульного метода заданная БФ может быть реализована только схемой из трех модулей.

6.6. Реализация булевых функций схемами из трехмембранных пневматических реле Р-3Ф серии УСЭППА

Этот элемент реализует БФУ [13, 14]

$$\begin{aligned} F(a, b, c, d) &= |0001 \ 0000 \ 1101 \ 1111|^T = \\ &= |1 \ 0 \ 13 \ 15|^T = a(b \vee !c) \vee !bcd. \end{aligned}$$

Рассматриваемый элемент реализует путем настройки представителей всех PN -типов неповторных формул в базисе $\{\&, \vee, !\}$ из трех букв [10]:

$$\begin{aligned} a=0 \quad F &= !bcd, & a=b \quad F &= b \vee cd, \\ a=!b \quad F &= !b(!c \vee d), & a=1 \quad F &= b \vee !c \vee d, \end{aligned}$$

что позволяет при использовании двухэтапного формульного метода реализовать БФ в указанном базисе из h букв схемами, число модулей в которых удовлетворяет неравенству

$$L(h) \leq \left\lceil \frac{2(h-1)}{3} \right\rceil.$$

Этот элемент позволяет также реализовать при $d=!a \quad F = a \oplus !bc$, а при $b=0 \quad F = a!c \vee cd$. Так как на основе этих соотношений могут быть выполнены разложения Риди и Шеннона соответственно, то модуль является универсальным для построения на его основе схем, реализующих произвольные БФУ n переменных, для которых

$$L(n) \leq 2^{n-1} - 1.$$

Следовательно,

$$L(h, n) \leq \min \left(\left\lceil \frac{2(h-1)}{3} \right\rceil, 2^{n-1} - 1 \right).$$

Выполним более глубокое исследование функциональных возможностей этого элемента.

1. Для определения БФУ, порождаемых за счет переобозначения переменных, построим для ПФ элемента $4! = 24$ таблицы истинности (транспонированные столбцы значений), каждой из которых соответствует определенный порядок переменных a, b, c, d . Результаты этого построения объединены в табл. 6.3.

Т а б л и ц а 6.3

№	Порядок переменных	Столбец значений	Номер столбца и его тип	№	Порядок переменных	Столбец значений	Номер столбца и его тип
1	a, b, c, d	1 0 13 15	1/1	13	c, a, b, d	0 15 4 7	13/3
2	a, b, d, c	1 0 11 15	2/2	14	c, a, d, b	0 15 2 7	14/4
3	a, c, b, d	0 4 15 7	3/3	15	c, b, a, d	3 5 3 3	9/7
4	a, c, d, b	0 2 15 7	4/4	16	c, b, d, a	5 3 5 5	10/7
5	a, d, b, c	0 4 11 15	5/5	17	c, d, a, b	3 3 1 11	15/10
6	a, d, c, b	0 2 13 15	6/6	18	c, d, b, a	5 5 1 13	16/11
7	b, a, c, d	1 13 0 15	7/1	19	d, a, b, c	0 11 4 15	17/5
8	b, a, d, c	1 11 0 15	8/2	20	d, a, c, b	0 13 2 15	18/6
9	b, c, a, d	3 5 3 3	9/7	21	d, b, a, b	2 3 7 3	19/8
10	b, c, d, a	5 3 5 5	10/7	22	d, b, c, a	4 5 7 5	20/9
11	b, d, a, c	2 7 3 3	11/8	23	d, c, a, b	3 1 3 11	21/10
12	b, d, c, a	4 7 5 5	12/9	24	d, c, b, a	5 1 5 13	22/11

Если еще раз изменить обозначение переменных и зафиксировать порядок расположения новых переменных, то каждой из этих ТИ будет соответствовать своя функция (тип декомпозиций) четырех переменных.

Пусть при $n - k = 0$ эта БФУ зависит от переменных $\Phi_1, \Phi_2, \Phi_3, \Phi_4$, при $n - k = 1$ — от переменных $\Phi_1, \Phi_2, \Phi_3, x_n$, при $n - k = 2$ — от переменных $\Phi_1, \Phi_2, x_{n-1}, x_n$, а при $n - k = 3$ — от переменных $\Phi, x_{n-2}, x_{n-1}, x_n$.

2. При $n - k = 0$ поиск решений с требуемыми свойствами связан с огромным перебором, и поэтому в дальнейшем этот случай не рассматривается.

3. Заданный модуль универсален при $n - k = 1$, так как 2, 4, 5, 6, 7, 8, 11, 14, 17, 19, 20 и 21-я строки этой таблицы (всего 12 строк) содержат все четыре типа фрагментов длиной два. Так, например, строке 2 соответствует декомпозиция вида:

$$f = MX(0, x_n, 0, 0, !x_n, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3) = \\ = \Phi_1(!x_n \vee \Phi_2) \vee x_n! \Phi_2 \Phi_3.$$

Это позволяет разложить произвольную БФУ по крайней правой переменной x_n в ТИ. Частным случаем разложения по этой переменной является настройка элемента на реализацию функции мультиплексора «2 в 1». Используя строку 8 таблицы, получим:

$$f = \text{MX}(0, x_n, !x_n, 1, 0, 0, 1, 1; \Phi_1, \Phi_2, \Phi_3) = \\ \Phi_2(!x_n \vee \Phi_1) \vee x_n !\Phi_1 \Phi_3.$$

Эта БФУ при $\Phi_1 = 0$ приобретает вид: $f = \Phi_2 !x_n \vee \Phi_3 x_n$. При этом $a = \Phi_2$, $b = 0$, $c = x_n$, $d = \Phi_3$.

4. Если в соотношениях для $n - k = 1$ выполнить подстановку $x_n = Q$, то появляется возможность разложения заданной функции по подфункции Q , применяя рассматриваемый образ декомпозиции.

5. Переходя к случаю $n - k = 2$ и используя переменные Φ_1 , Φ_2 , x_{n-1} , x_n , можно получить 22 типа декомпозиций, из которых одиннадцать отличаются номенклатурой фрагментов длиной четыре (табл. 6.3). Например, первой строке таблицы соответствует декомпозиция вида:

$$f = \text{MX}(x_{n-1} x_n, 0, !x_{n-1} \vee x_n, 1; \Phi_1, \Phi_2) = \\ = \Phi_1(!x_{n-1} \vee \Phi_2) \vee x_{n-1} x_n !\Phi_2,$$

а девятой строке — декомпозиция вида:

$$f = \text{MX}(x_{n-1}, x_n, x_{n-1}, x_{n-1}; \Phi_1, \Phi_2) = \\ = x_{n-1}(\Phi_1 \vee !\Phi_2) \vee x_n !\Phi_1 \Phi_2.$$

Таким образом, рассматриваемый элемент позволяет декомпозировать 11 классов БФУ, каждый из которых определяется номенклатурой фрагментов длиной четыре в их столбцах значений:

$$\{0, x_{n-1} x_n, !x_{n-1} \vee x_n, 1\}, \{0, x_{n-1} x_n, x_{n-1} \vee !x_n, 1\}, \\ \{0, !x_{n-1} x_n, x_{n-1} \vee x_n, 1\}, \{0, x_{n-1} !x_n, x_{n-1} \vee x_n, 1\}, \\ \{0, !x_{n-1} x_n, x_{n-1} \vee !x_n, 1\}, \{0, x_{n-1} !x_n, !x_{n-1} \vee x_n, 1\}, \\ \{x_{n-1}, x_n\}, \{x_{n-1}, x_{n-1} !x_n, x_{n-1} \vee x_n\}, \{x_n, !x_{n-1} x_n, x_{n-1} \vee x_n\}, \\ \{x_{n-1}, x_{n-1} !x_n, x_{n-1} \vee !x_n\}, \{x_n, x_{n-1} x_n, !x_{n-1} \vee x_n\},$$

причем в каждом классе (за исключением седьмого) могут быть выделены подклассы, каждый из которых образован не всеми фрагментами соответствующего класса.

Из изложенного следует, что если столбец значений реализуемой БФУ содержит хотя бы один фрагмент длиной четыре, не принадлежащий множеству

$$\{0, x_{n-1} x_n, x_{n-1} !x_n, x_{n-1}, !x_{n-1} x_n, x_n, \\ x_{n-1} \vee x_n, x_{n-1} \vee !x_n, !x_{n-1} \vee x_n, 1\},$$

то эта функция неразложима по переменным x_{n-1} и x_n .

Если для этих переменных доступны также и их инверсии, то каждая из рассмотренных декомпозиций порождает по три декомпозиции, что с учетом повторений увеличивает число декомпозируемых классов БФУ с 11 до 26. Так, например, первая декомпозиция, соответствующая первой строке табл. 6.3, порождает три декомпозиции вида:

$$\begin{aligned} f &= \text{MX}(x_{n-1}!x_n, 0, !x_{n-1} \vee !x_n, 1; \Phi_1, \Phi_2) = \\ &= \Phi_1(!x_{n-1} \vee \Phi_2) \vee x_{n-1}!x_n!\Phi_2; \\ f &= \text{MX}(!x_{n-1}x_n, 0, x_{n-1} \vee x_n, 1; \Phi_1, \Phi_2) = \\ &= \Phi_1(x_{n-1} \vee \Phi_2) \vee !x_{n-1}x_n!\Phi_2 \end{aligned}$$

(совпадает по номенклатуре фрагментов со строкой 3 табл. 6.3);

$$\begin{aligned} f &= \text{MX}(!x_{n-1}!x_n, 0, x_{n-1} \vee !x_n, 1; \Phi_1, \Phi_2) = \\ &= \Phi_1(x_{n-1} \vee \Phi_2) \vee !x_{n-1}x_n!\Phi_2, \end{aligned}$$

а вторая декомпозиция, соответствующая строке 9 табл. 6.3, порождает:

$$\begin{aligned} f &= \text{MX}(!x_{n-1}, x_n, !x_{n-1}, !x_{n-1}; \Phi_1, \Phi_2) = \\ &= !x_{n-1}(\Phi_1 \vee !\Phi_2) \vee x_n! \Phi_1 \Phi_2, \\ f &= \text{MX}(x_{n-1}, !x_n, x_{n-1}, x_{n-1}; \Phi_1, \Phi_2) = \\ &= x_{n-1}(\Phi_1 \vee !\Phi_2) \vee !x_n! \Phi_1 \Phi_2, \\ f &= \text{MX}(!x_{n-1}, !x_n, !x_{n-1}, !x_{n-1}; \Phi_1, \Phi_2) = \\ &= !x_{n-1}(\Phi_1 \vee !\Phi_2) \vee !x_n! \Phi_1 \Phi_2. \end{aligned}$$

Это расширяет множество реализуемых фрагментов еще на четыре фрагмента: $\{!x_{n-1}, x_n, !x_{n-1}!x_n, !x_{n-1} \vee !x_n\}$.

Таким образом, неразложимыми по двум крайним правым переменным таблицы истинности в этом случае являются БФУ, в столбцах значений которых встречаются фрагменты $x_{n-1} \oplus x_n$ и $!(x_{n-1} \oplus x_n)$, а также БФУ, номенклатура фрагментов в столбцах значений которых не входит в 26 классов, указанных выше. При этом отметим, что если ТИ заданной БФУ при некотором порядке переменных не содержит «разрешенную» номенклатуру фрагментов, то не исключена возможность, что после построения ТИ с другим порядком переменных БФУ будет относиться к классу функций, разложимых по указанным переменным.

6. При $n - k = 3$ по табл. 6.3 также могут быть построены различные декомпозиции. Так, например, первой ее строке соответствует декомпозиция

$$\begin{aligned} f &= \text{MX}(!x_{n-2}x_{n-1}x_n, x_{n-2} \vee !x_{n-1} \vee x_n; \Phi) = \\ &= \Phi_1(x_{n-2} \vee !x_{n-1}) \vee x_{n-2}x_{n-1}x_n. \end{aligned}$$

При этом отметим, что с увеличением величины $n - k$ класс БФУ, декомпозируемых по заданному образцу, уменьшается.

Пример 6.15. Определить возможность декомпозиции БФУ

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= |1111 \ 0100 \ 0111 \ 0000|^T = \\ &= |15 \ 4 \ 7 \ 0|^T = (!x_1 \vee x_3 \vee x_4)!x_2 \vee !x_1!x_3x_4 \end{aligned}$$

по образцу рассматриваемого элемента.

Этот столбец значений состоит из фрагментов длиной четыре третьего типа (табл. 6.3). Используя, например, строку 3 этой таблицы, получим:

$$\begin{aligned} (!x_1 \vee x_3 \vee x_4)!x_2 \vee !x_1!x_3x_4 &= \Phi_1(x_3 \vee !\Phi_2) \vee !x_3x_4\Phi_2; \\ \text{MX}(1, !x_3x_4, x_3 \vee x_4, 0; x_1, x_2) &= \\ = \text{MX}(0, !x_3x_4, 1, x_3 \vee x_4; \Phi_1, \Phi_2); \\ \Phi_1 &= \text{MX}(1, 0, 1, 0; x_1, x_2) = !x_2; \\ \Phi_2 &= \text{MX}(0, 1, 1, 0; x_1, x_2) = x_1 \oplus x_2. \end{aligned}$$

Таким образом, $a = !x_2$, $b = x_3$, $c = x_1 \oplus x_2$, $d = x_4$.

Пример 6.16. Определить возможность декомпозиции БФУ

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= |1011 \ 0000 \ 1111 \ 1000|^T = \\ &= |11 \ 0 \ 15 \ 8|^T = (x_1 \vee x_3 \vee !x_4)!x_2 \vee x_1!x_3!x_4 \end{aligned}$$

по образцу рассматриваемого элемента.

При $n - k = 2$ только при «прямых» входных переменных набор фрагментов ($G_0 = x_3 \vee !x_4$, $G_1 = 0$, $G_2 = 1$, $G_3 = !x_3!x_4$) не является разрешенным. Однако если учесть, что в данном случае первой строке табл. 6.3 соответствуют фрагменты $Q_0 = x_3x_4$, $Q_1 = 0$, $Q_2 = !x_3 \vee x_4$, $Q_3 = 1$, то можно записать и решить уравнение:

$$\begin{aligned} (x_1 \vee x_3 \vee !x_4)!x_2 \vee x_1!x_3!x_4 &= \Phi_1(x_3 \vee \Phi_2) \vee !x_3!x_4!\Phi_2; \\ \text{MX}(x_3 \vee !x_4, 0, 1, !x_3!x_4; x_1, x_2) &= \\ = \text{MX}(!x_3!x_4, 0, x_3 \vee !x_4, 1; \Phi_1, \Phi_2); \\ \Phi_1 &= \text{MX}(1, 0, 1, 0; x_1, x_2) = !x_2; \\ \Phi_2 &= \text{MX}(0, 1, 1, 0; x_1, x_2) = x_1 \oplus x_2. \end{aligned}$$

Таким образом, $a = !x_2$, $b = x_1 \oplus x_2$, $c = !x_3$, $d = !x_4$.

Пример 6.17. Определить возможность декомпозиции БФУ

$$f(x_1, x_2, x_3, x_4) = |1000 \ 0000 \ 0000 \ 0001|^T = \\ = !x_1!x_2!x_3!x_4 \vee x_1x_2x_3x_4$$

по образу рассматриваемого элемента.

Разложение при $n - k = 2, 3$ не выполнить. Проведем разложение по переменной x_4 , применяя строку 2 табл. 6.3:

$$!x_1!x_2!x_3!x_4 \vee x_1x_2x_3x_4 = \Phi_1(!x_4 \vee \Phi_2) \vee x_4!\Phi_2\Phi_3; \\ MX(!x_4, 0, 0, 0, 0, 0, 0, 0, x_4; x_1, x_2) = \\ = MX(0, x_4, 0, 0, !x_4, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3).$$

Среди 729 решений этого уравнения выберем следующее:

$$\Phi_1 = MX(1, 0, 0, 0, 0, 0, 0, 0; x_1, x_2, x_3) = !x_1!x_2!x_3; \\ \Phi_2 = MX(0, 0, 0, 0, 0, 0, 0, 0; x_1, x_2, x_3) = 0; \\ \Phi_3 = MX(0, 0, 0, 0, 0, 0, 0, 1; x_1, x_2, x_3) = x_1x_2x_3.$$

Таким образом, $a = !x_1!x_2!x_3$, $b = 0$, $c = x_4$, $d = x_1x_2x_3$.
Обратим внимание, что в данном случае

$$f = \Phi_1!x_4 \vee \Phi_3x_4.$$

Пример 6.18. Реализовать БФУ, рассмотренную в предыдущем примере.

Настройка выходного элемента определена в предыдущем примере. Реализуем теперь функции Φ_1 и Φ_3 . Реализуем функцию Φ_1 , используя при $n - k = 2$ первую строку табл. 6.3 с учетом требуемых инверсий:

$$!x_1!x_2!x_3 = \Phi_{11}(x_2 \vee \Phi_2) \vee !x_2!x_3\Phi_{12}; \\ MX(!x_2!x_3, 0; x_1) = MX(!x_2!x_3, 0, x_2 \vee !x_3, 1; \Phi_{11}, \Phi_{21}); \\ \Phi_{11} = MX(0, 0; x_1) = 0; \\ \Phi_{12} = MX(0, 1; x_1) = x_1.$$

Таким образом, $a = 0$, $b = x_1$, $c = !x_2$, $d = !x_3$.

Так как формула Φ_3 отличается от формулы Φ_1 только расстановкой инверсий, то $a = 0$, $b = !x_1$, $c = x_2$, $d = x_3$.

Следовательно, мультиплексорный метод строит в данном случае схему из трех элементов, в то время как двухэтапный формульный метод реализует эту БФУ схемой из четырех элементов.

6.7. Реализация булевых функций схемами из модулей, универсальных в классе произвольных булевых функций трех переменных

Из изложенного в разд. 3.2.5 следует, что если при фиксированном значении $n - k$ порождающая функция модуля содержит в качестве фрагментов все БФУ $n - k$ переменных, то произвольная БФУ всегда может быть разложена по $n - k$ крайним правым входным переменным.

Поэтому можно утверждать, что любые модули, при построении которых применяются какие-либо классификации [10], не являются универсальными разложителями по крайним правым t входным переменным, так как любая классификация позволяет сократить номенклатуру фрагментов, объединяемых в модуль, чем обычно и пользуются для сокращения числа переменных в ПФ модуля.

При $t = 3$, например в [15], описан модуль, при построении которого использовалась PN -классификация, с пятью входами и одним выходом, а при $t = 4$, например в [16], описан модуль, который построен с применением NP -классификации, с семью входами и двумя выходами — прямым и инверсным.

Если при $n = t$ реализация БФУ на таких модулях осуществляется с помощью таблиц настройки, то при $n > t$ методы синтеза схем в базисе этих модулей в литературе не описывались.

Рассмотрим три таких метода на примере модуля с $t = 3$.

Первый из этих методов (двухэтапный формульный метод) состоит в использовании этих модулей как универсальных в классе произвольных БФ в базисе $\{\&, \vee, \oplus, !\}$, что приводит к следующей оценке сложности:

$$L(h) \leq \left\lceil \frac{2(h-1)}{3} \right\rceil.$$

Второй метод является модификацией двухэтапного формульного метода и основан на выделении в заданной БФ подформул, зависящих от трех переменных.

Третьим методом является мультиплексорный метод. Этот метод, так как порождающая функция модуля содержит фрагменты $\{0, !x_n, x_n, 1\}$, позволяет проводить с помощью такого модуля многократное разложение по крайней правой входной переменной заданной БФУ и всех ее остаточных, до тех пор пока они не будут зависеть от трех и менее переменных. При этом в худшем случае $2^{n-3} - 1$ элементов применяются в режиме разложителя, а 2^{n-3} элементов — в режиме модулей, универсальных в классе БФУ трех и менее переменных. Следовательно,

$$L(n) \leq 2^{n-2} - 1.$$

Из изложенного следует, что

$$L(h, n) \leq \min \left(\left\lceil \frac{2(h-1)}{3} \right\rceil, 2^{n-2} - 1 \right).$$

Для более эффективного использования модуля исследуем его функциональные возможности. Для упрощения это исследование будем проводить не при всех перестановках переменных на входах модуля, а только при фиксированном их порядке.

Запишем ТИ порождающей функции модуля [15] и выполним исследование ее функциональных возможностей:

$$F(a, b, c, d, e) = |0000 \ 0001 \ 1110 \ 1001 \ 0111 \ 1000 \ 0110 \ 1111|^T.$$

1. Так как при $n - k = 0$ выбор решения с требуемыми свойствами связан с огромным перебором, этот случай в дальнейшем не рассматривается.

2. При $n - k = 1$, $a = \Phi_1$, $b = \Phi_2$, $c = \Phi_3$, $d = \Phi_4$, $e = x_n$:

$$f = \text{MX}(0, 0, 0, x_n, 1, !x_n, !x_n, x_n, x_n, 1, !x_n, 0, x_n, !x_n, 1, 1; \\ \Phi_1, \Phi_2, \Phi_3, \Phi_4).$$

Это соотношение обеспечивает разложения произвольной булевой функции f по переменной x_n при применении заданного образа. При этом число решений равно $4^{t_0+t_1+t_2+t_3}$.

3. При $a = \Phi_1$, $b = \Phi_2$, $c = \Phi_3$, $d = \Phi_4$, $e = Q$:

$$f = \text{MX}(0, 0, 0, Q, 1, !Q, !Q, Q, Q, 1, !Q, 0, Q, !Q, 1, 1; \\ \Phi_1, \Phi_2, \Phi_3, \Phi_4).$$

Это соотношение обеспечивает разложения булевой функции f по подфункции Q при использовании заданного образа. При выделении подфункции Q число решений равно $4^{t_0+t_1+t_4+t_5}$.

4. При $n - k = 2$, $a = \Phi_1$, $b = \Phi_2$, $c = \Phi_3$, $d = x_{n-1}$, $e = x_n$:

$$f = \text{MX}(0, x_{n-1}x_n, !x_{n-1} \vee !x_n, !(x_{n-1} \oplus x_n), x_{n-1} \vee \\ \vee x_n, !x_{n-1}!x_n, x_{n-1} \oplus x_n, 1; \Phi_1, \Phi_2, \Phi_3).$$

Это соотношение обеспечивает единственное разложение каждой булевой функции f , фрагменты которой принадлежат указанному в этой мультиплексорной форме множеству фрагментов.

5. При $n - k = 3$, $a = \Phi_1$, $b = \Phi_2$, $c = x_{n-2}$, $d = x_{n-1}$, $e = x_n$:

$$f = \text{MX}(x_{n-2}x_{n-1}x_n, !x_{n-2} \# !x_{n-1} \# !x_n \vee x_{n-2}x_{n-1}x_n, !x_{n-2} \oplus \\ \oplus !x_{n-1}!x_n, x_{n-2} \vee (x_{n-1} \oplus x_n); \Phi_1, \Phi_2).$$

Это соотношение обеспечивает единственное разложение каждой булевой функции f , фрагменты которой принадлежат указанному в этой мультиплексорной форме множеству фрагментов.

6. При $n - k = 4$, $a = \Phi$, $b = x_{n-3}$, $c = x_{n-2}$, $d = x_{n-1}$, $e = x_n$:

$$f = \text{MX}(!x_{n-3}x_{n-2}x_{n-1}x_n \vee x_{n-3}(!x_{n-2} \# !x_{n-1} \# !x_n), \\ !x_{n-2}(!x_{n-3}x_{n-1} \vee (x_{n-1} \oplus x_n)) \vee x_{n-2}(x_{n-3} \vee !x_{n-1}!x_n); \Phi).$$

Это соотношение обеспечивает единственное разложение каждой булевой функции f , фрагменты которой принадлежат указанному в этой мультиплексорной форме множеству фрагментов.

Пример 6.19. Определить настройку модуля на реализацию функции мультиплексирования $f(x_1, x_2, x_3) = |0001 \ 1011|^T = x_1!x_3 \vee \vee x_2x_3$.

При $n - k = 1$ составим уравнение:

$$x_1!x_3 \vee x_2x_3 = F(\Phi_1, \Phi_2, \Phi_3, \Phi_4, x_3); \\ \text{MX}(0, x_3, !x_3, 1; x_1, x_2) = \\ = \text{MX}(0, 0, 0, x_3, 1, !x_3, !x_3, x_3, x_3, 1, !x_3, 0, x_3, !x_3, 1, 1; \\ \Phi_1, \Phi_2, \Phi_3, \Phi_4).$$

Это уравнение имеет 256 решений, среди которых выберем следующее:

$$\Phi_1 = \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1; \\ \Phi_2 = \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1; \\ \Phi_3 = \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2; \\ \Phi_4 = \text{MX}(1, 1, 1, 1; x_1, x_2) = 1.$$

Следовательно, $a = x_1$, $b = x_1$, $c = x_2$, $d = 1$, $e = x_3$.

Пример 6.20. Реализовать ПФ модуля, универсального в классе пороговых БФУ трех переменных,

$$f(x_1, x_2, x_3, x_4) = |0000 \ 0001 \ 0111 \ 1111|^T = \\ = x_1(x_2 \vee x_3 \vee x_4) \vee x_2x_3x_4.$$

При $n - k = 2$ составим и решим уравнение:

$$x_1(x_2 \vee x_3 \vee x_4) \vee x_2x_3x_4 = F(\Phi_1, \Phi_2, \Phi_3, x_3, x_4); \\ \text{MX}(0, x_3x_4, x_3 \vee x_4, 1; x_1, x_2) = \\ = \text{MX}(0, x_3x_4, !x_3 \vee !x_4, !(x_3 \oplus x_4), x_3 \vee x_4, !x_3!x_4, x_3 \oplus \\ \oplus x_4, 1; \Phi_1, \Phi_2, \Phi_3);$$

$$\begin{aligned}\Phi_1 &= \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1; \\ \Phi_2 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2; \\ \Phi_3 &= \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2.\end{aligned}$$

При $n - k = 1$ составим уравнение:

$$\begin{aligned}x_1 x_2 &= F(\Phi_1, \Phi_2, \Phi_3, \Phi_4, x_2); \\ \text{MX}(0, x_2; x_1) &= \text{MX}(0, 0, 0, x_2, 1, !x_2, !x_2, x_2, x_2, \\ &1, !x_2, 0, x_2, !x_2, 1, 1; \Phi_{21}, \Phi_{22}, \Phi_{23}, \Phi_{24}).\end{aligned}$$

Среди 16 решений этого уравнения выберем следующее:

$$\begin{aligned}\Phi_{21} &= \text{MX}(0, 1; x_1) = x_1; \\ \Phi_{22} &= \text{MX}(0, 0; x_1) = 0; \\ \Phi_{23} &= \text{MX}(0, 0; x_1) = 0; \\ \Phi_{24} &= \text{MX}(0, 0; x_1) = 0.\end{aligned}$$

Таким образом, заданная порождающая функция реализуется схемой из двух модулей, в то время как формульный метод строит в данном случае схему из трех модулей.

6.8. Реализация булевых функций схемами из модулей Препараты—Мюллера

В [17] предложены модули, универсальные в классе произвольных булевых функций t переменных, на информационные входы которых подаются переменные $x_1, \dots, x_t$, а на каждый из настроечных входов может быть подан один из следующих символов: $\{0, 1, !x_1, \dots, !x_t, x_1, \dots, x_t\}$.

Настройка этих модулей на реализацию заданной булевой функции t переменных производится по преобразованной ТИ [17, 18]. Однако методы реализации БФУ при $n > t$ схемами из таких модулей в литературе не описаны.

Рассмотрим три таких метода.

Первый из них (формульный метод) состоит в использовании этих модулей как универсальных в классе произвольных БФ в базе $\{\&, \vee, \oplus, !\}$, что приводит к следующей оценке сложности:

$$L(h) \leq \left\lceil \frac{2(h-1)}{t} \right\rceil.$$

Второй метод является модификацией двухэтапного формульного метода и основан на выделении в заданной БФ подформул, зависящих от t переменных.

Третьим методом является мультиплексорный метод. Предположим, что $t = 3$. Этот метод, так как ПФ модуля содержит фрагменты $\{0, !x_n, x_n, 1\}$, позволяет проводить с помощью этого элемента многократное разложение по крайней правой входной переменной заданной БФУ и всех ее остаточных, до тех пор пока они не будут зависеть от трех и менее переменных. При этом в худшем случае $2^{n-3} - 1$ элементов применяются в режиме разложителя, а 2^{n-3} элементов — в режиме модулей, универсальных в классе булевых функций трех и менее переменных. Следовательно,

$$L(n) \leq 2^{n-2} - 1.$$

Из изложенного следует, что при $t = 3$

$$L(h, n) \leq \min \left(\left\lceil \frac{2(h-1)}{3} \right\rceil, 2^{n-2} - 1 \right).$$

Для более эффективного применения модуля исследуем его функциональные возможности. Для упрощения это исследование будем проводить не при всех перестановках переменных на его входах, а только при фиксированном их порядке.

Запишем ТИ порождающей функции модуля [17, 18]:

$$\begin{aligned} F(a, b, c, d, e, f) &= |0000 \ 0000 \ 0001 \ 0001 \ 0110 \ 1000 \ 0111 \\ &\quad 1001 \ 1000 \ 0110 \ 1001 \ 0111 \ 1110 \ 1110 \ 1111 \ 1111|^T = \\ &= a(!d!e!f \vee d(e \oplus f)) \vee b(!d(e \oplus f) \vee d!e!f) \vee cef. \end{aligned}$$

1. Так как при $n - k = 0$ выбор решения с требуемыми свойствами связан с огромным перебором, этот случай в дальнейшем не рассматривается.

2. При $n - k = 1$, $a = \Phi_1$, $b = \Phi_2$, $c = \Phi_3$, $d = \Phi_4$, $e = \Phi_5$, $f = x_n$:

$$\begin{aligned} f_1 &= \text{MX}(0, 0, 0, 0, 0, x_n, 0, x_n, x_n, !x_n, !x_n, 0, x_n, 1, !x_n, x_n, !x_n, \\ &\quad 0, x_n, !x_n, !x_n, x_n, x_n, 1, 1, !x_n, 1, !x_n, 1, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5). \end{aligned}$$

Это соотношение обеспечивает разложения произвольной булевой функции f_1 по переменной x_n при использовании заданного образа. При этом число решений равно $8^{t_0+t_1+t_2+t_3}$.

3. При $a = \Phi_1$, $b = \Phi_2$, $c = \Phi_3$, $d = \Phi_4$, $e = \Phi_5$, $f = Q$:

$$\begin{aligned} f_1 &= \text{MX}(0, 0, 0, 0, 0, Q, 0, Q, Q, !Q, !Q, 0, Q, 1, !Q, Q, !Q, \\ &\quad 0, Q, !Q, !Q, Q, Q, 1, 1, !Q, 1, !Q, 1, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5). \end{aligned}$$

Это соотношение обеспечивает разложения булевой функции f_1 по подфункции Q при применении заданного образа. При выделении подфункции Q число решений равно $8^{t_0+t_1+t_4+t_5}$.

4. При $n - k = 2$, $a = \Phi_1$, $b = \Phi_2$, $c = \Phi_3$, $d = \Phi_4$, $e = x_{n-1}$, $f = x_n$:

$$f_1 = \text{MX}(0, 0, x_{n-1}x_n, x_{n-1}x_n, x_{n-1} \oplus x_n, !x_{n-1}!x_n, x_{n-1} \vee x_n, !(x_{n-1} \oplus x_n), !x_{n-1}!x_n, x_{n-1} \oplus x_n, !(x_{n-1} \oplus x_n), x_{n-1} \vee x_n, !x_{n-1} \vee !x_n, !x_{n-1} \vee !x_n, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4).$$

Это соотношение обеспечивает разложения каждой булевой функции f_1 , фрагменты которой принадлежат указанному в этой мультиплексорной форме множеству фрагментов.

5. При $n - k = 3$, $a = \Phi_1$, $b = \Phi_2$, $c = \Phi_3$, $d = x_{n-2}$, $e = x_{n-1}$, $f = x_n$:

$$f_1 = \text{MX}(0, x_{n-1}x_n, !x_{n-2}(x_{n-1} \oplus x_n) \vee x_{n-2}!x_{n-1}!x_n, !x_{n-2} \# x_{n-1} \# x_n \vee x_{n-2}!x_{n-1}!x_n, !x_{n-2}!x_{n-1}!x_n \vee x_{n-2}(x_{n-1} \oplus x_n), x_{n-2} \# x_{n-1} \# x_n \vee !x_{n-2}!x_{n-1}!x_n, !x_{n-1} \vee !x_n, 1; \Phi_1, \Phi_2, \Phi_3).$$

Это соотношение обеспечивает единственное разложение каждой булевой функции f_1 , фрагменты которой принадлежат указанному в этой мультиплексорной форме множеству фрагментов.

6. Так же могут быть записаны соотношения при $n - k = 4$, $a = \Phi_1$, $b = \Phi_2$, $c = x_{n-3}$, $d = x_{n-2}$, $e = x_{n-1}$, $f = x_n$ и при $n - k = 5$, $a = \Phi$, $b = x_{n-4}$, $c = x_{n-3}$, $d = x_{n-2}$, $e = x_{n-1}$, $f = x_n$.

Пример 6.21. Определить настройку модуля на реализацию БФУ

$$f_1(x_1, x_2, x_3) = |0110 \ 1001|^T = x_1 \oplus x_2 \oplus x_3.$$

Составим уравнение при $n - k = 2$:

$$\begin{aligned} x_1 \oplus x_2 \oplus x_3 &= F(\Phi_1, \Phi_2, \Phi_3, \Phi_4, x_2, x_3); \\ \text{MX}(x_2 \oplus x_3, !(x_2 \oplus x_3); x_1) &= \text{MX}(0, 0, x_2x_3, x_2x_3, x_2 \oplus x_3, !x_2!x_3, x_2 \vee x_3, !(x_2 \oplus x_3), !x_2!x_3, x_2 \oplus x_3, !(x_2 \oplus x_3), x_2 \vee x_3, !x_2 \vee !x_3, !x_2 \vee !x_3, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4). \end{aligned}$$

Это уравнение имеет четыре решения:

$$\begin{cases} \Phi_1 = \text{MX}(0, 0; x_1) = 0, \\ \Phi_2 = \text{MX}(1, 1; x_1) = 1, \\ \Phi_3 = \text{MX}(0, 1; x_1) = x_1, \\ \Phi_4 = \text{MX}(0, 1; x_1) = x_1; \end{cases} \quad \begin{cases} \Phi_1 = \text{MX}(0, 1; x_1) = x_1, \\ \Phi_2 = \text{MX}(1, 0; x_1) = !x_1, \\ \Phi_3 = \text{MX}(0, 1; x_1) = x_1, \\ \Phi_4 = \text{MX}(0, 0; x_1) = 0; \end{cases}$$

$$\left\{ \begin{array}{l} \Phi_1 = \text{MX}(1, 0; x_1) = !x_1, \\ \Phi_2 = \text{MX}(0, 1; x_1) = x_1, \\ \Phi_3 = \text{MX}(0, 1; x_1) = x_1, \\ \Phi_4 = \text{MX}(1, 1; x_1) = 1; \end{array} \right. \quad \left\{ \begin{array}{l} \Phi_1 = \text{MX}(1, 1; x_1) = 1, \\ \Phi_2 = \text{MX}(0, 0; x_1) = 0, \\ \Phi_3 = \text{MX}(0, 1; x_1) = x_1, \\ \Phi_4 = \text{MX}(1, 0; x_1) = !x_1. \end{array} \right.$$

При этом отметим, что известный метод [17, 18] позволяет определить только первое из этих решений (табл. 6.4).

Т а б л и ц а 6.4

j	x_1	x_2	x_3	f_1	Φ_j
1	0	0	0	0	0
	1	0	1	0	
	1	1	0	0	
2	0	0	1	1	1
	0	1	0	1	
	1	0	0	1	
3	0	1	1	0	x_1
	1	1	1	1	

Пример 6.22. Реализовать ПФ модуля, универсального в классе пороговых БФУ трех переменных,

$$f(x_1, x_2, x_3, x_4) = |0000 \ 0001 \ 0111 \ 1111|^T = \\ = x_1(x_2 \vee x_3 \vee x_4) \vee x_2 x_3 x_4.$$

При $n - k = 2$ составим и решим уравнение:

$$x_1(x_2 \vee x_3 \vee x_4) \vee x_2 x_3 x_4 = F(\Phi_1, \Phi_2, \Phi_3, \Phi_4, x_3, x_4); \\ \text{MX}(0, x_3 x_4, x_3 \vee x_4, 1; x_1, x_2) = \\ = \text{MX}(0, 0, x_3 x_4, x_3 x_4, x_3 \oplus x_4, !x_3 !x_4, x_3 \vee x_4, !(x_3 \oplus \\ \oplus x_4), !x_3 !x_4, x_3 \oplus x_4, !(x_3 \oplus x_4), x_3 \vee \\ \vee x_4, !x_3 \vee !x_4, !x_3 \vee !x_4, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4).$$

Это уравнение имеет 16 решений, среди которых нет решений, содержащих только одну функцию Φ_j , зависящую от переменных x_1 и x_2 . Поэтому выберем следующее решение:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2; \\ \Phi_2 &= \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1; \\ \Phi_3 &= \text{MX}(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2; \\ \Phi_4 &= \text{MX}(0, 0, 0, 0; x_1, x_2) = 0. \end{aligned}$$

Таким образом, заданная БФУ реализуется схемой из трех модулей.

Пример 6.23. Реализовать БФУ

$$\begin{aligned} f_1(x_1, x_2, x_3, x_4) &= |0111 \ 0001 \ 1110 \ 0110|^T = \\ &= !x_1 x_3 x_4 \vee x_1 ((!x_2 \vee x_4) !x_3 \vee x_3 !x_4 \vee !x_2 (x_3 \oplus x_4)). \end{aligned}$$

Модифицированный формульный метод строит в данном случае схему из пяти модулей:

$$\begin{aligned} f_0 &= (!x_2 \vee x_4) !x_3 \vee x_3 !x_4, \quad f_2 = !x_2 (x_3 \oplus x_4), \\ f_3 &= x_1 (f_0 \vee f_2), \quad f_4 = !x_1 x_3 x_4, \quad f_1 = f_3 \vee f_4. \end{aligned}$$

При $n - k = 2$ составим и решим уравнение:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= F(\Phi_1, \Phi_2, \Phi_3, \Phi_4, x_3, x_4); \\ \text{MX}(x_3 \vee x_4, x_3 x_4, !x_3 \vee !x_4, x_3 \oplus x_4; x_1, x_2) &= \\ = \text{MX}(0, 0, x_3 x_4, x_3 x_4, x_3 \oplus x_4, !x_3 !x_4, x_3 \vee x_4, !(x_3 \oplus \\ &\oplus x_4), !x_3 !x_4, x_3 \oplus x_4, !(x_3 \oplus x_4), x_3 \vee \\ &\vee x_4, !x_3 \vee !x_4, !x_3 \vee !x_4, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4). \end{aligned}$$

Это уравнение имеет 16 решений, среди которых выберем следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1; \\ \Phi_2 &= \text{MX}(1, 0, 1, 0; x_1, x_2) = !x_2; \\ \Phi_3 &= \text{MX}(1, 1, 0, 0; x_1, x_2) = !x_1; \\ \Phi_4 &= \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2. \end{aligned}$$

Таким образом, заданная БФУ четырех переменных реализуется одним модулем, универсальным в классе произвольных БФУ трех переменных.

Следовательно, мультиплексорный метод позволяет использовать кроме тех БФУ, которые закладывались в модуль при проектировании (все БФУ трех переменных), также и ряд других функций, которые модуль может реализовать путем настройки, так как его порождающая функция в этом случае зависит от шести переменных. Это объясняется тем, что мультиплексорный метод позволяет увеличить по сравнению с другими методами число входов модулей, применяемых в качестве информационных входов.

6.9. Реализация булевых функций схемами из 4-универсальных модулей

Этот модуль (разд. 8.2 и [19]) имеет порождающую функцию вида:

$$\begin{aligned} F(a, b, c, d, e, f) &= \\ &= |0000 \ 0000 \ 0000 \ 0001 \ 0000 \ 0111 \ 0111 \ 0111 \\ &\quad 0001 \ 0001 \ 0001 \ 1111 \ 0111 \ 1111 \ 1111 \ 1111|^T = \\ &= a(b(c \vee d \vee e \vee f) \vee cd \vee ef) \vee b(c \vee d)(e \vee f) \vee cdef. \end{aligned}$$

Модуль путем настройки ($a, b, c = \{0, x_1, 1\}$, $d = x_2$, $e = x_3$, $f = x_4$) реализует представителей всех десяти *PN*-типов БФ в базисе $\{\&, \vee, !\}$ из четырех букв.

При использовании формульного метода в данном случае справедливо соотношение:

$$\left\lceil \frac{h-1}{3} \right\rceil \leq L(h, 4) \leq \left\lceil \frac{h-1}{2} \right\rceil.$$

Для более эффективного применения модуля исследуем его функциональные возможности. Для упрощения это исследование будем проводить не при всех перестановках переменных на входах модуля, а только при фиксированном их порядке.

1. Так как при $n - k = 0$ выбор решения с требуемыми свойствами связан с огромным перебором, этот случай в дальнейшем не рассматривается.

2. При $n - k = 1$, $a = \Phi_1$, $b = \Phi_2$, $c = \Phi_3$, $d = \Phi_4$, $e = \Phi_5$, $f = x_n$ и доступности только «прямых» входных переменных модуль позволяет реализовать БФУ, у которых столбцы значений состоят из фрагментов из множества $\{0, x_n, 1\}$. Для булевой функции f_1 этого класса справедливы соотношения:

$$\begin{aligned} f_1(x_1, \dots, x_n) &= F(\Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5, x_n); \\ f_1(x_1, \dots, x_n) &= \text{MX}(0, 0, 0, 0, 0, 0, 0, x_n, 0, 0, x_n, 1, x_n, \\ &\quad 1, x_n, 1, 0, x_n, 0, x_n, 0, x_n, 1, 1, x_n, 1, 1, 1, 1, 1, 1; \\ &\quad \Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5). \end{aligned}$$

Для функций рассматриваемого класса это уравнение имеет $12^{t_0} \cdot 12^{t_1} \cdot 8^{t_3}$ решений.

3. При $a = \Phi_1$, $b = \Phi_2$, $c = \Phi_3$, $d = \Phi_4$, $e = \Phi_5$, $f = Q$ и доступности только «прямых» входных переменных модуль позволяет реализовать БФУ, у которых столбцы значений состоят из фрагментов

из множества $\{0, Q, 1\}$. Для булевой функции f_1 этого класса справедливо соотношение:

$$\begin{aligned} f(x_1, \dots, x_n) &= F(\Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5, Q); \\ f(x_1, \dots, x_n) &= \text{MX}(0, 0, 0, 0, 0, 0, 0, Q, 0, 0, Q, 1, Q, 1, Q, \\ &1, 0, Q, 0, Q, 0, Q, 1, 1, Q, 1, 1, 1, 1, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5). \end{aligned}$$

Для функций рассматриваемого класса это уравнение имеет $12^{t_0} \cdot 12^{t_1} \cdot 8^{t_5}$ решений.

4. При $n - k = 2$, $a = \Phi_1$, $b = \Phi_2$, $c = \Phi_3$, $d = \Phi_4$, $e = x_{n-1}$, $f = x_n$:

$$\begin{aligned} f_1 &= \text{MX}(0, 0, 0, x_{n-1}x_n, 0, x_{n-1} \vee x_n, x_{n-1} \vee x_n, x_{n-1} \vee x_n, \\ &x_{n-1}x_n, x_{n-1}x_n, x_{n-1}x_n, 1, x_{n-1} \vee x_n, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4). \end{aligned}$$

Это соотношение обеспечивает разложения каждой булевой функции f_1 , фрагменты которой принадлежат указанному в этой мультиплексорной форме множеству фрагментов. При этом число решений определяется соотношением $4^{t_0+t_1+t_6+t_7}$.

5. При $n - k = 3$, $a = \Phi_1$, $b = \Phi_2$, $c = \Phi_3$, $d = x_{n-2}$, $e = x_{n-1}$, $f = x_n$:

$$\begin{aligned} f_1 &= \text{MX}(0, x_{n-2}x_{n-1}x_n, x_{n-2}(x_{n-1} \vee x_n), x_{n-1} \vee x_n, x_{n-1}x_n, \\ &x_{n-2} \vee x_{n-1}x_n, x_{n-2} \vee x_{n-1} \vee x_n, 1; \Phi_1, \Phi_2, \Phi_3). \end{aligned}$$

Это соотношение обеспечивает единственное разложение каждой булевой функции f_1 , фрагменты которой принадлежат указанному в этой мультиплексорной форме множеству фрагментов.

6. При $n - k = 4$, $a = \Phi_1$, $b = \Phi_2$, $c = x_{n-3}$, $d = x_{n-2}$, $e = x_{n-1}$, $f = x_n$:

$$\begin{aligned} f_1 &= \text{MX}(x_{n-3}x_{n-2}x_{n-1}x_n, (x_{n-3} \vee x_{n-2})(x_{n-1} \vee x_n), \\ &x_{n-3}x_{n-2} \vee x_{n-1}x_n, x_{n-3} \vee x_{n-2} \vee x_{n-1} \vee x_n; \Phi_1, \Phi_2). \end{aligned}$$

Это соотношение обеспечивает единственное разложение каждой булевой функции f_1 , фрагменты которой принадлежат указанному в этой мультиплексорной форме множеству фрагментов.

7. При $n - k = 5$, $a = \Phi$, $b = x_{n-4}$, $c = x_{n-3}$, $d = x_{n-2}$, $e = x_{n-1}$, $f = x_n$:

$$\begin{aligned} f_1 &= \text{MX}(x_{n-3}x_{n-2}x_{n-1}x_n \vee x_{n-4}(x_{n-3} \vee x_{n-2})(x_{n-1} \vee x_n), \\ &(x_{n-3} \vee x_{n-2})(x_{n-1} \vee x_n) \vee x_{n-4}(x_{n-3} \vee x_{n-2} \vee x_{n-1} \vee x_n); \Phi). \end{aligned}$$

Это соотношение обеспечивает единственное разложение каждой булевой функции f_1 , фрагменты которой принадлежат указанному в этой мультиплексорной форме множеству фрагментов.

8. Если кроме переменных доступны также и их инверсии, то класс функций, реализуемых рассматриваемым модулем, существует

венно расширяется. Так, например, для отрицательно монотонных БФУ по переменной x_n их разложения могут определяться на основе решения уравнения:

$$\begin{aligned} f_1(x_1, \dots, x_n) &= F(\Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5, !x_n); \\ f_1(x_1, \dots, x_n) &= \\ &= \text{MX}(0, 0, 0, 0, 0, 0, 0, 0, !x_n, 0, 0, !x_n, 1, !x_n, 1, !x_n, 1, 0, !x_n, \\ &0, !x_n, 0, !x_n, 1, 1, !x_n, 1, 1, 1, 1, 1, 1, \Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5). \end{aligned}$$

Для функций рассматриваемого класса это уравнение имеет $12^{t_0} \cdot 12^{t_1} \cdot 8^{t_3}$ решений.

Пример 6.24. Реализовать ПФ модуля, универсального в классе пороговых БФУ трех переменных,

$$\begin{aligned} f_1(x_1, x_2, x_3, x_4) &= |0000 \ 0001 \ 0111 \ 1111|^T = \\ &= x_1(x_2 \vee x_3 \vee x_4) \vee x_2 x_3 x_4. \end{aligned}$$

При $n - k = 3$ составим и решим уравнение:

$$\begin{aligned} &x_1(x_2 \vee x_3 \vee x_4) \vee x_2 x_3 x_4 = \\ &= \Phi_1(\Phi_2(\Phi_3 \vee x_2 \vee x_3 \vee x_4) \vee \Phi_3 x_2 \vee x_3 x_4) \vee \\ &\vee \Phi_2(\Phi_3 \vee x_2)(x_3 \vee x_4) \vee \Phi_3 x_2 x_3 x_4; \\ \text{MX}(x_2 x_3 x_4, x_2 \vee x_3 \vee x_4; x_1) &= \text{MX}(0, x_2 x_3 x_4, x_2(x_3 \vee x_4), \\ &x_3 \vee x_4, x_3 x_4, x_2 \vee x_3 x_4, x_2 \vee x_3 \vee x_4, 1; \Phi_1, \Phi_2, \Phi_3); \\ \Phi_1 &= \text{MX}(0, 1; x_1) = x_1; \\ \Phi_2 &= \text{MX}(0, 1; x_1) = x_1; \\ \Phi_3 &= \text{MX}(1, 0; x_1) = !x_1. \end{aligned}$$

Таким образом, заданная БФУ реализуется при $a = x_1$, $b = x_1$, $c = !x_1$, $d = x_2$, $e = x_3$, $f = x_4$.

Следовательно, мультиплексорный метод позволил использовать дополнительные функциональные возможности модуля — реализовать на одном модуле БФ не из четырех, а из семи букв.

Формульный метод строит в данном случае схему из двух модулей.

Пример 6.25. Реализовать БФУ

$$\begin{aligned} f_1(x_1, x_2, x_3, x_4, x_5) &= \\ &= |0111 \ 0111 \ 0000 \ 0111 \ 0001 \ 0001 \ 0001 \ 1111|^T = \\ &= x_1 x_2 x_3 \vee (!x_1 !x_2 \vee x_2 x_3) x_4 \vee (!x_1 (!x_2 \vee x_3) \vee x_1 x_4) x_5. \end{aligned}$$

Формульный метод строит в данном случае схему из пяти модулей.

При $n - k = 3$ составим и решим уравнение:

$$\begin{aligned}
 & x_1 x_2 x_3 \vee (!x_1 !x_2 \vee x_2 x_3) x_4 \vee (!x_1 (!x_2 \vee x_3) \vee x_1 x_4) x_5 = \\
 & = \Phi_1 (\Phi_2 (\Phi_3 \vee x_3 \vee x_4 \vee x_5) \vee \Phi_3 x_3 \vee x_4 x_5) \vee \\
 & \quad \vee \Phi_2 (\Phi_3 \vee x_3) (x_4 \vee x_5) \vee \Phi_3 x_3 x_4 x_5; \\
 & MX(x_4 \vee x_5, x_3 (x_4 \vee x_5), x_4 x_5, x_3 \vee x_4 x_5; x_1, x_2) = \\
 & = MX(0, x_3 x_4 x_5, x_3 (x_4 \vee x_5), x_4 \vee x_5, x_4 x_5, \\
 & \quad x_3 \vee x_4 x_5, x_3 \vee x_4 \vee x_5, 1; \Phi_1, \Phi_2, \Phi_3); \\
 & \Phi_1 = MX(0, 0, 1, 1; x_1, x_2) = x_1; \\
 & \Phi_2 = MX(1, 1, 0, 0; x_1, x_2) = !x_1; \\
 & \Phi_3 = MX(1, 0, 0, 1; x_1, x_2) = !x_1 !x_2 \vee x_1 x_2.
 \end{aligned}$$

Так как формула Φ_3 содержит четыре буквы, то она может быть реализована одним модулем.

Таким образом, заданная БФУ может быть реализована схемой из двух модулей.

6.10. Реализация булевых функций схемами из одного типа функциональных преобразователей СБИС программируемой логики

Рассмотрим в качестве примера использование мультиплексорного метода для синтеза схем в базисе функционального преобразователя СБИС программируемой логики семейства АСТ2 фирмы «Actel» [20]. В этом преобразователе комбинационная схема описывается соотношением

$$F = MX(a, b, c, d; \Phi_1, \Phi_2),$$

где $\Phi_1 = ef, \Phi_2 = g \vee h$.

Пример 6.26. Декомпонировать по образу рассматриваемого преобразователя БФУ

$$\begin{aligned}
 f_1(x_1, x_2, x_3, x_4) &= |0000 \ 0101 \ 0111 \ 1110|^T = \\
 &= (!x_1 x_4 \vee x_1 !x_3) x_2 \vee x_1 (!x_2 x_4 \vee x_3 !x_4).
 \end{aligned}$$

При $n - k = 1$ предположим, что $Q_0 = 0, Q_1 = x_4, Q_2 = !x_4, Q_3 = 1$.

Запишем и решим уравнение в МФ:

$$\begin{aligned} \text{MX}(\mathcal{Q}_0, \mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_1, \mathcal{Q}_1, \mathcal{Q}_3, \mathcal{Q}_3, \mathcal{Q}_2; x_1, x_2, x_3) &= \\ &= \text{MX}(\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3; \Phi_1, \Phi_2); \\ \Phi_1 &= \text{MX}(0, 0, 0, 0, 0, 1, 1, 1; x_1, x_2, x_3) = x_1(x_2 \vee x_3); \\ \Phi_2 &= \text{MX}(0, 0, 1, 1, 1, 1, 1, 0; x_1, x_2, x_3) = !x_1x_2 \vee x_1(!x_2 \vee !x_3). \end{aligned}$$

Таким образом, заданная БФУ декомпозируется следующим образом: $a=0$, $b=x_4$, $c=!x_4$, $d=1$, $e=x_1$, $f=x_2 \vee x_3$, $g=!x_1x_2$, $h=x_1(!x_2 \vee !x_3)$.

При $n - k = 2$ предположим, что $\mathcal{Q}_0=0$, $\mathcal{Q}_1=x_4$, $\mathcal{Q}_2=x_3 \vee x_4$, $\mathcal{Q}_3=!x_3 \vee !x_4$.

Запишем и решим уравнение в МФ:

$$\begin{aligned} \text{MX}(\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3; x_1, x_2) &= \text{MX}(\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3; \Phi_1, \Phi_2); \\ \Phi_1 &= x_1, \quad \Phi_2 = x_2. \end{aligned}$$

Таким образом, заданная БФУ декомпозируется следующим образом: $a=0$, $b=x_4$, $c=x_3 \vee x_4$, $d=!x_3 \vee !x_4$, $e=x_1$, $f=1$, $g=x_2$, $h=0$.

Пример 6.27. Декомпонировать по образу рассматриваемого преобразователя БФУ

$$\begin{aligned} f_1(x_1, x_2, x_3, x_4) &= |0001 \ 0101 \ 0001 \ 1111|^T = \\ &= (x_1 \vee x_4)x_2 \vee x_3x_4. \end{aligned}$$

При $n - k = 2$ предположим, что $\mathcal{Q}_0=x_3x_4$, $\mathcal{Q}_1=x_4$, $\mathcal{Q}_2=x_3x_4$, $\mathcal{Q}_3=1$.

Запишем и решим уравнение в МФ:

$$\begin{aligned} \text{MX}(\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3; x_1, x_2) &= \text{MX}(\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3; \Phi_1, \Phi_2); \\ \Phi_1 &= x_1, \quad \Phi_2 = x_2. \end{aligned}$$

Таким образом, заданная БФУ декомпозируется следующим образом: $a=c=x_3x_4$, $b=x_4$, $c=x_3 \vee x_4$, $d=1$, $e=x_1$, $f=1$, $g=x_2$, $h=0$.

При $n - k = 1$ предположим, что $\mathcal{Q}_0=0$, $\mathcal{Q}_1=x_4$, $\mathcal{Q}_2=!x_4$, $\mathcal{Q}_3=1$.

Запишем и решим уравнение в МФ:

$$\begin{aligned} \text{MX}(\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_1, \mathcal{Q}_1, \mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_3, \mathcal{Q}_3; x_1, x_2, x_3) &= \\ &= \text{MX}(\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3; \Phi_1, \Phi_2); \\ \Phi_1 &= \text{MX}(0, 0, 0, 0, 0, 0, 1, 1; x_1, x_2, x_3) = x_1x_2; \\ \Phi_2 &= \text{MX}(0, 1, 1, 1, 0, 1, 1, 1; x_1, x_2, x_3) = x_2 \vee x_3. \end{aligned}$$

Таким образом, заданная БФУ не только декомпозируется, но и реализуется с помощью одного рассматриваемого преобразователя: $a=0, b=x_4, c=!x_4, d=1, e=x_1, f=x_2, g=x_2, h=x_3$.

6.11. Реализация булевых функций большого числа переменных

При большом числе переменных ($n \geq 6$) задание произвольной, полностью определенной БФУ с помощью ТИ весьма громоздко. Поэтому при таких значениях n булевы функции обычно задаются в виде булевых формул.

Мультиплексорный метод при таком задании БФУ может применяться после разложения Шеннона, проведения эвристической декомпозиции и/или использования формульного метода.

Пример 6.28. Реализовать схемой из 3-универсальных модулей БФ

$$f = (x_1 x_2 \vee x_3 x_4) x_5 \vee (x_6 \vee x_7) (x_8 \vee x_9).$$

Формульный метод строит в данном случае схему из пяти модулей: $f_1 = x_1 x_2, f_2 = f_1 \vee x_3 x_4, f_3 = x_6 \vee x_7, f_4 = f_3 (x_8 \vee x_9), f = f_2 x_5 \vee f_4$.

При применении 5-универсального модуля формульный метод строит схему из двух модулей:

$$f_1 = (x_1 x_2 \vee x_3 x_4) x_5, f = f_1 \vee (x_6 \vee x_7) (x_8 \vee x_9).$$

В примере 3.22 показано, что каждая из этих формул реализуется с помощью мультиплексорного метода схемой из двух 3-универсальных модулей.

Таким образом, совместное применение формульного и мультиплексорного методов позволяет реализовать заданную БФ схемой из четырех заданных модулей.

Пример 6.29. Реализовать схемой из модулей, универсальных в классе пороговых функций трех переменных, БФ

$$f = (x_1 \vee x_2) x_3 \vee x_1 x_2 \vee (x_4 \vee x_5) (x_6 \vee x_7).$$

Формульный метод строит в данном случае схему из пяти модулей:

$$f_1 = (x_1 \vee x_2) x_3, f_2 = f_1 \vee x_1 x_2, f_3 = x_4 \vee x_5, \\ f_4 = f_3 (x_6 \vee x_7), f = f_2 \vee f_4.$$

Эвристическая декомпозиция и формульный метод строят совместно схему из четырех модулей:

$$f_1 = x_1 \# x_2 \# x_3, f_2 = x_4 \vee x_5, f_3 = f_2 (x_6 \vee x_7), f = f_1 \vee f_3.$$

Эвристическая декомпозиция и мультиплексорный метод строят совместно схему из трех модулей:

$$f_1 = x_1 \# x_2 \# x_3, f_2 = f_1 \vee x_4 \vee x_5, \\ f = f_2 (f_1 \vee x_6 \vee x_7) \vee f_1 x_6 x_7.$$

6.12. Реализация автоматов с памятью схемами, использующими триггеры

Известен табличный метод реализации автоматов с памятью, использующий таблицы возбуждения триггеров [21].

Этот метод был усовершенствован в [22]. Однако и в этой редакции он является весьма громоздким. При этом формулы для возбуждения входов триггеров определяются в результате доопределения функций возбуждения и выбора значений констант в них [23].

В [21] для рассматриваемой задачи был предложен аналитический метод решения систем логических уравнений с одним и двумя неизвестными, однако уже при двух неизвестных этот метод весьма громоздок [24].

В [24, 25] был предложен более простой аналитический метод, позволяющий свести решение систем логических уравнений с m неизвестными к последовательному решению $k \leq m$ логических уравнений с одним неизвестным.

Пример 6.30. Решить уравнение $x_1 z \vee x_2 = S!z \vee !Rz$ относительно переменных R и S .

Решение этой задачи при применении метода, изложенного в [24, 25], может быть разбито на две части.

Полагая, что $z_1 = x_1 z \vee x_2$, решим уравнение $z_1 = S!z \vee !Rz$. В [25] показано, что

$$R = !z_1 z \vee h_1!z, \quad S = z_1!z \vee h_2 z,$$

где h_1 и h_2 — произвольные БФУ от переменных R и S .

Подставляя в эти соотношения выражение для z_1 , получим:

$$R = !(x_1 z \vee x_2) z \vee h_1!z = !x_1!x_2 z \vee h_1!z; \\ S = (x_1 z \vee x_2)!z \vee h_2 z = x_2!z \vee h_2 z.$$

Полагая $h_1 = !x_1!x_2$ и $h_2 = x_2$, получим: $R = !x_1!x_2$, $S = x_2$.

Достоинство изложенного метода состоит в том, что он является аналитическим, а недостаток — в относительной трудоемкости.

Для применения определенных типов триггеров могут быть использованы более простые подходы.

Так, в [26, 27] показано, что JK -триггер является универсальным, так как функция, описывающая его функционирование,

$$z_1 = J!z \vee !Kz = S!z \vee !Rz$$

может применяться для разложения Шеннона по переменной z .

Это свойство может использоваться для решения логических уравнений.

Пример 6.31. Решить уравнение $x_1 z \vee x_2 = S!z \vee !Rz$ относительно переменных R и S .

Полагая $z_1 = x_1 z \vee x_2$, получим, что при $z = 0$ $z_1 = x_2$, а при $z = 1$ $z_1 = x_1 \vee x_2$.

Таким образом,

$$x_2!z \vee (x_1 \vee x_2)z = S!z \vee !Rz.$$

Это уравнение имеет единственное решение:

$$S = x_2, \quad R = !x_1!x_2.$$

Из изложенного следует, что общее решение для JK -триггера имеет вид:

$$J = z_1(X, z = 0), \quad K = !z_1(X, z = 1).$$

В [27] показано, что для функций переходов, положительно монотонных по переменной z , универсальным является S -триггер, функционирование которого описывается соотношением

$$z_1 = S \vee !Rz.$$

Пример 6.32. Решить уравнение $x_1 z \vee x_2 = S \vee !Rz$ относительно переменных R и S .

Непосредственно из этого уравнения следует, что

$$R = !x_1, \quad S = x_2.$$

Соотношения для определения функций возбуждения T -триггеров приведены в [26, 27].

Для других типов триггеров найти простые аналитические решения, аналогичные приведенным для JK - и S -триггеров, не удастся.

Покажем, что мультиплексорный метод позволяет с единых позиций решать поставленную задачу по крайней мере тех типов триггеров, функции переходов которых приведены в табл. 6.5.

Т а б л и ц а 6.5

R	S	z	JK	S	R	E
			z_1	z_1	z_1	z_1
0	0	0	0	0	0	0
0	0	1	1	1	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	1

Из рассмотрения этой таблицы следует, что только столбец значений JK -триггера содержит все четыре типа фрагментов длиной два, что обеспечивает разложение произвольной функции переходов по крайней правой входной переменной. Остальные триггеры реализуют лишь монотонные функции переходов.

Пример 6.33. Реализовать автомат $z_1(x_1, x_2, z) = |0011\ 0111|^T = x_1 z \vee x_2$ на JK -триггере.

При $n - k = 1$ составим и решим уравнение:

$$x_1 z \vee x_2 = S!z \vee !Rz;$$

$$MX(0, 1, z, 1; x_1, x_2) = MX(z, 1, 0, !z; R, S);$$

$$R = MX(1, 0, 0, 0; x_1, x_2) = !x_1!x_2;$$

$$S = MX(0, 1, 0, 1; x_1, x_2) = x_2.$$

Пример 6.34. Реализовать автомат $z_1(x_1, x_2, z) = |0011\ 0111|^T = x_1 z \vee x_2$ на S -триггере.

При $n - k = 1$ составим уравнение:

$$x_1 z \vee x_2 = S \vee !Rz;$$

$$MX(0, 1, z, 1; x_1, x_2) = MX(z, 1, 0, 1; R, S).$$

Это уравнение имеет четыре решения:

$$R = MX(1, 0, 0, 0; x_1, x_2) = !x_1!x_2,$$

$$S = MX(0, 1, 0, 1; x_1, x_2) = x_2;$$

$$R = MX(1, 0, 0, 1; x_1, x_2) = !(x_1 \oplus x_2),$$

$$S = MX(0, 1, 0, 1; x_1, x_2) = x_2;$$

$$R = \text{MX}(1, 1, 0, 0; x_1, x_2) = !x_1,$$

$$S = \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2;$$

$$R = \text{MX}(1, 1, 0, 1; x_1, x_2) = !x_1 \vee x_2,$$

$$S = \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2.$$

Простейшим является третье решение. Таким образом, если подход, рассмотренный в примере 6.32, позволяет получить только одно решение (правда, простейшее), то мультиплексорный метод обеспечивает возможность получения большего числа решений.

Пример 6.35. Реализовать автомат $z_1(x_1, x_2, z) = |0011 \ 0111|^T = x_1 z \vee x_2$ на R -триггере.

При $n - k = 1$ составим уравнение:

$$x_1 z \vee x_2 = !R(S \vee z);$$

$$\text{MX}(0, 1, z, 1; x_1, x_2) = \text{MX}(z, 1, 0, 0; R, S).$$

Это уравнение имеет два решения, из которых первое является более простым:

$$R = \text{MX}(1, 0, 0, 0; x_1, x_2) = !x_1 !x_2,$$

$$S = \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2;$$

$$R = \text{MX}(1, 0, 0, 0; x_1, x_2) = !x_1 !x_2,$$

$$S = \text{MX}(1, 1, 0, 1; x_1, x_2) = !x_1 \vee x_2.$$

Пример 6.36. Реализовать автомат $z_1(x_1, x_2, z) = |0011 \ 0111|^T = x_1 z \vee x_2$ на E -триггере.

При $n - k = 1$ составим уравнение:

$$x_1 z \vee x_2 = !R \# S \# z;$$

$$\text{MX}(0, 1, z, 1; x_1, x_2) = \text{MX}(z, 1, 0, z; R, S).$$

Это уравнение имеет два решения:

$$R = \text{MX}(1, 0, 0, 0; x_1, x_2) = !x_1 !x_2,$$

$$S = \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2;$$

$$R = \text{MX}(1, 0, 1, 0; x_1, x_2) = !x_2,$$

$$S = \text{MX}(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2.$$

Пример 6.37. Построить булевы формулы для программной реализации счетного триггера на R -триггерах.

Известно, например, из [27], что счетный триггер может быть описан системой булевых формул, в которой переобозначение переменной z будет выполнено в дальнейшем:

$$z(x, y, z) = |0011 \ 0101|^T = !xy \vee xz;$$

$$y(x, y, z) = |0101 \ 1100|^T = !xy \vee x!z.$$

Обе формулы положительно монотонны по определенным переменным: первая — по переменным y и z , а вторая — по переменной y . Составим уравнения так, чтобы в каждом из них в правой части упоминалась та переменная, которая используется в обозначении функции в левой части соответствующего соотношения, приведенного выше:

$$!xy \vee xz = !R_1(S_1 \vee z);$$

$$!xy \vee x!z = !R_2(S_2 \vee y).$$

Решим первое уравнение, записав его в МФ:

$$MX(0, 1, z, z; x, y) = MX(z, 1, 0, 0; R_1, S_1).$$

Это уравнение имеет два решения:

$$R_1 = MX(1, 0, 0, 0; x, y) = !x!y,$$

$$S_1 = MX(0, 1, 0, 0; x, y) = !xy;$$

$$R_1 = MX(1, 0, 0, 0; x, y) = !x!y,$$

$$S_1 = MX(1, 1, 0, 0; x, y) = !x.$$

Решим второе уравнение, записав его в МФ:

$$MX(y, y, 1, 0; x, z) = MX(y, 1, 0, 0; R_2, S_2).$$

Это уравнение имеет два решения:

$$R_2 = MX(0, 0, 0, 1; x, z) = xz,$$

$$S_2 = MX(0, 0, 1, 0; x, z) = x!z;$$

$$R_2 = MX(0, 0, 0, 1; x, z) = xz,$$

$$S_2 = MX(0, 0, 1, 1; x, z) = x.$$

Выбирая вторые решения и вводя переобозначения переменной z , получим искомую СБФ:

$$\begin{aligned} R_1 &= !x!y, & S_1 &= !x, & z_1 &= !R_1(S_1 \vee z), \\ R_2 &= xz, & S_2 &= x, & y &= !R_2(S_2 \vee y), \\ z &= z_1. \end{aligned}$$

Выводы

1. Рассмотрено использование мультиплексорного метода для синтеза комбинационных схем в различных элементных базисах, известных из литературы. На примерах продемонстрирована эффективность этого метода. Мультиплексорный метод позволяет применять кроме тех булевых функций t переменных, которые закладывались в модуль при проектировании, также и ряд других функций, которые модуль может реализовать путем настройки, так как его порождающая функция зависит от числа переменных, большего t . Это объясняется тем, что мультиплексорный метод позволяет увеличить по сравнению с другими методами число входов модулей, используемых в качестве информационных входов, за счет увеличения количества решений, известного в каждом конкретном случае.

При этом, в частности, удалось доказать, что произвольная БФ в базисе И, ИЛИ, НЕ из пяти букв реализуется схемой из двух 3-универсальных модулей, в то время как для формульного метода верхняя оценка числа модулей в этом случае равна трем. На примере показано, что для модулей Препараты—Мюллера вместо получаемой известным методом единственной настройки на заданную БФУ предлагаемый метод позволяет найти большее число вариантов настройки.

2. Показано, как применять мультиплексорный метод для построения автоматов с памятью, использующих триггеры. Этот метод универсален и существенно менее трудоемок, чем известные также универсальные методы.

Л и т е р а т у р а

1. *Hellerman L.* A catalogue of three-variable or-invert and-invert logical circuits // IEEE Trans. on Electronic Computers. 1963. N 6.
2. *Артюхов В. Л., Шалыто А. А.* Реализация булевых формул однородными мультиплексорными и мажоритарными каскадами // Изв. РАН. Теория и системы управления. 1996. № 5.
3. *Артюхов В. Л., Розенблюм Л. Я., Шалыто А. А.* Логические возможности некоторых типов каскадных структур // Сети связи и дискретные устройства управления. М.: Наука, 1976.

4. Боголюбов И. Н., Овсевич Б. Л., Розенблюм Л. Я. Синтез схем из пороговых и мажоритарных элементов // Сети передачи информации и их автоматизация. М.: Наука, 1965.
5. Данем Б., Норт Д. Проблема выбора логически эффективных основных ячеек и их соединений // Синтез релейных структур. М.: Наука, 1965.
6. Patt Y. N. A complex logic module for the synthesis of combinational switching circuits // Proc. 1967 AFIPS, Spring Joint Comput. Conf.
7. Блох А. Ш. Граф-схемы и их применение. Минск: Вышэйшая школа, 1975.
8. Блох А. Ш., Павловский А. И. Логические сети из трехвходовых операторов // Автоматика и вычисл. техника. 1980. № 4.
9. Блох А. Ш., Павловский А. И. Синтез логических схем на трехвходовом операторе // Доклады АН БССР. 1979. № 7.
10. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Настраиваемые модули для управляющих логических устройств. Л.: Энергоиздат, 1981.
11. Cohen S., Winder R. Threshold gate building blocks // IEEE Trans. on Computers. 1969. № 9.
12. Артюхов В. Л., Копейкин Г. А., Шалыто А. А., Фишман Л. М. Многофункциональный логический модуль. А.с. СССР № 813787 // Бюл. изобр. 1979. № 10.
13. Вольский В. Е., Пушин Ю. Н., Юнг В. Н. Проектирование пневматических систем управления судовыми энергетическими установками. Л.: Судостроение, 1975.
14. Артюхов В. Л., Шалыто А. А., Вольский В. Е. Логический модуль. А. с. СССР № 798806 // Бюл. изобр. 1981. № 3.
15. Артюхов В. Л., Шалыто А. А., Фишман Л. М. Многофункциональный логический модуль. А. с. СССР № 760451 // Бюл. изобр. 1980. № 32.
16. Варавацкий В. И., Мараховский В. Б., Песчанский В. А. Многофункциональные модули, реализующие все функции трех и четырех переменных // II Всесоюз. совещ. по теории релейных устройств и конечных автоматов. Тез. докл. Рига, 1971.
17. Preparata F. P., Muller D. E. Generation of near-optimal universal boolean functions // J. Comp. and System Science. 1970. N 2.
18. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Судовые управляющие логические системы: Унифицированные логические схемы. Л.: ИПК СП, 1981.
19. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 758141 // Бюл. изобр. 1980. № 31.
20. Антонов А. П., Мелехин В. Ф., Филиппов А. С. Обзор элементной базы фирмы ALTERA. СПб.: ЭФО, 1997.
21. Фистер М. Логическое проектирование цифровых вычислительных машин. Киев: Техника, 1964.
22. Агаханян Т. М., Плеханов С. П. Интегральные триггеры устройств автоматизации. М.: Машиностроение, 1978.
23. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Судовые управляющие системы. Л.: ИПК СП, 1982.
24. Пухальский Г. И. Логическое проектирование цифровых устройств радиотехнических систем. Л.: ЛГУ, 1976.
25. Пухальский Г. И., Новосельцева Т. Я. Цифровые устройства. СПб.: Политехника, 1996.
26. Поспелов Д. А. Логические методы анализа и синтеза схем. М.: Энергия, 1968.
27. Шалыто А. А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.

Глава 7

Методы построения многофункциональных логических модулей

Известно [1], что мультиплексоры «2 в 1» при настройке константами 0 и 1 универсальны в классе произвольных булевых функций (БФУ) n переменных, а при настройке символами из множества $\{0, !x_n, x_n, 1\}$ они универсальны в классе произвольных БФУ $n + 1$ переменных.

Для сокращения числа входов модулей, универсальных в классе произвольных БФУ n переменных, в [2] был предложен метод их построения, называемый методом Препараты—Мюллера. Эти модули настраиваются символами из множества $\{0, !x_1, x_1, \dots, !x_n, x_n, 1\}$. Более подробно такие модули рассмотрены в гл. 11.

Дальнейшее сокращение числа входов модулей указанного типа достигается при объединении в модуль не самих функций, а представителей типов функций в той или иной классификации [3].

Другое направление в построении модулей связано с разработкой модулей, универсальных в классах булевых функций [4, 5]. Некоторые из таких модулей приведены в гл. 9, 10.

В настоящей главе рассматривается еще одно направление построения модулей, которые позволяют путем настройки реализовать каждую из N заданных БФУ n переменных. Такие модули называются многофункциональными логическими модулями (МЛМ).

В [6—11] исследуются различные аспекты построения МЛМ, однако методы построения таких модулей разработаны далеко не достаточно.

В настоящей главе излагается ряд методов для построения модулей из элементов с односторонней проводимостью (функциональные элементы). Модули из элементов с двусторонней проводимостью рассмотрены в гл. 13.

7.1. Основные характеристики многофункциональных логических модулей

К основным характеристикам МЛМ относятся:

- число переменных n в объединяемых булевых функциях (БФУ);
- число объединяемых БФУ;
- класс объединяемых БФУ;
- базис объединяемых булевых формул (БФ);
- число букв в объединяемых БФ;
- число объединяемых БФ;
- класс объединяемых БФ;
- наличие инверсий в объединяемых БФ;
- используемая классификация БФУ и БФ;
- разделимость входов на информационные, настроечные (управляющие) и информационно-настроечные;
- число входов модуля, или число входных переменных, в его порождающей функции (ПФ);
- число букв в порождающей формуле (ПФО) модуля;
- число символов в алфавите настройки;
- число выходов модуля;
- наличие прямого и инверсного выходов.

Излагаемые в настоящей главе методы позволяют строить модули с различными характеристиками. Основные усилия автора были направлены на разработку одновыходных модулей, универсальных в классе произвольных БФ в базисе $\{\&, \vee, !\}$ из q букв, у которых:

- информационные и информационно-настроечные входы разделены;
- число входов меньше величины $q + \lceil \log N \rceil$;
- число букв в ПФО минимизировано;
- ПФО не содержит инверсий;
- алфавит настройки: константы 0 и 1, переменные и их инверсии.

Построение этих и других типов модулей описано в примерах настоящей главы и в гл. 8. Большинство рассмотренных модулей защищены авторскими свидетельствами СССР, общее число которых равно 31 [12].

7.2. Мультиплексорный метод — метод объединения фрагментов

7.2.1. Основные положения

Пусть заданы N различных БФУ n переменных, которые необходимо объединить в порождающую функцию модуля.

1. Введем в рассмотрение переменную $k = \{0, 1, \dots, n - 1\}$ и зафиксируем ее значение. При этом число информационных вхо-

дов модуля равно $n - k$, а k переменных используется для его настройки.

Случай $k = n$ не рассматривается ввиду большой трудоемкости метода в этом случае. При этом значении k метод из аналитического превращается в переборный, так как в этом случае отсутствует разделение входов: они все являются информационно-настроечными.

При $k = n$ в [13, 14] были построены модули, универсальные в классе произвольных БФУ трех и четырех переменных. При $n = 3$ применялась PN -классификация и построены модули с пятью входами и одним выходом. При $n = 4$ применялась NPN -классификация и построены модули с семью входами и двумя выходами (прямым и инверсным).

В [14] доказано, что при $n = 3$ такие модули являются минимальными по числу внешних выводов, так как ни одна функция четырех переменных не может одновременно породить следующие три БФУ: $f_1 = x_1 x_2 x_3$, $f_2 = x_1 \# x_2 \# x_3$, $f_3 = x_1 \oplus x_2 \oplus x_3$. При этом отметим, что из соотношений для определения числа настроек МЛМ, приведенных в [15], также следует, что эти модули минимальны по числу внешних выводов.

Мультиплексорный метод также строит при $n = 3$ модуль с пятью входами и одним выходом. Модуль строится при $k = 1$ с дальнейшим переходом (при определении настройки для представителя одного из PN -типов) к $k = 2$, что позволяет для 15 типов функций иметь один информационный вход, а для одного типа функций — два информационных входа (разд. 8.4).

2. Выберем k переменных и проведем разложение Шеннона каждой из заданных БФУ одновременно по всем этим переменным. В результате формируется система соотношений вида:

$$f_t = MX(g_0^t, \dots, g_{2^k-1}^t; x_1, \dots, x_k),$$

где $t = 1, \dots, N$.

3. Выбор указанных переменных, а если возможно и самих функций (в случае, когда применяется какая-либо классификация), должен производиться так, чтобы число типов фрагментов G_j (различных функций g) во всех объединяемых БФУ было минимальным. Обозначим число типов фрагментов переменной R .

4. Определим величину $m = \lceil \log R \rceil$.

5. Выполним назначение функций Q_i , где $i = 0, \dots, 2^m - 1$.

Для этого при $R = 2^m$ сопоставим каждой из этих функций один из фрагментов G_j , и наоборот.

При $2^{m-1} < R < 2^m$ каждая из оставшихся $2^m - R$ функций Q_i сопоставляется либо с одним из фрагментов G_j , либо с любым другим фрагментом, зависящим от $n - k$ переменных.

Для построения порождающей формулы модуля с заданными свойствами при назначении функций Q_i , зависящих от $n - k$ переменных, должны выполняться отношения покрытия.

6. Для функции f_t построим мультиплексорную декомпозицию вида:

$$f_t = \text{MX}(Q_0, \dots, Q_{2^m-1}; \Phi_1^t, \dots, \Phi_m^t).$$

Этой декомпозиции соответствует схема, приведенная на рис. 7.1, которая состоит из мультиплексора и двух постоянных запоминающих устройств.

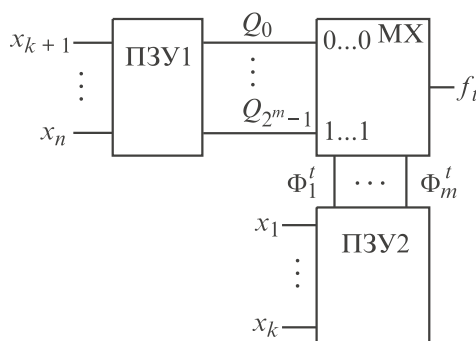


Рис. 7.1

7. Объединим функции Φ_j^t ($j = \text{const}, t = 1, \dots, M$), зависящие от $k < n$ переменных, в порождающую функцию, которую обозначим символом Φ_j . Эта функция в общем случае зависит от r информационных и p настроечных или информационно-настроечных входов. Значения переменных r и p зависят от свойств функций Φ_j^t и метода их объединения в порождающую функцию.

8. Построим булеву формулу вида:

$$F = \text{MX}(Q_0, \dots, Q_{2^m-1}; \Phi_1, \dots, \Phi_m), \quad (7.1)$$

которая является искомой порождающей формулой модуля.

Такому построению модуля соответствует стандартная схема, приведенная на рис. 7.2.

В этой и последующих схемах подсхема ПЗУ2 может быть заменена схемой из m многофункциональных логических модулей.

9. Если в ПФО (7.1) подставить выражения для функций Q_i , то

$$F = F_1(\Phi_1, \dots, \Phi_m, x_{k+1}, \dots, x_n).$$

При этом схема на рис. 7.2 преобразуется в схему, представленную на рис. 7.3, в которой КСх — комбинационная схема. В некоторых случаях эта схема может быть многофункциональным логическим модулем, который объединяет функции, зависящие от числа переменных, меньшего n (рис. 7.4).

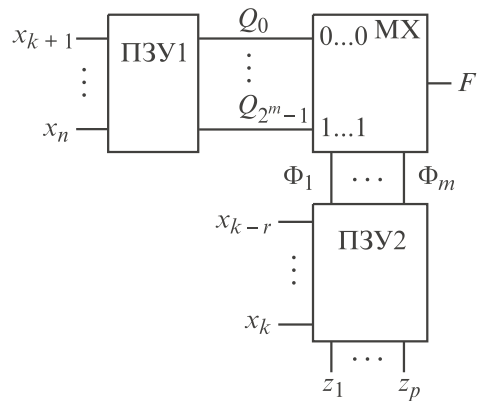


Рис. 7.2

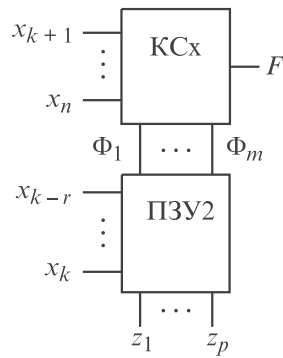


Рис. 7.3

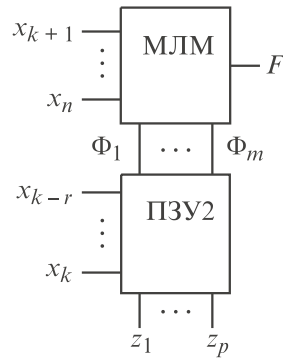


Рис. 7.4

10. Если в ПФО (7.1) подставить выражения для функций Φ_j , то

$$F = F_2(Q_0, \dots, Q_{2^m-1}, x_{k-r}, \dots, x_k, z_1, \dots, z_p).$$

11. Если в ПФО (7.1) подставить выражения для функций Q_i и Φ_j , то

$$F = F_3(x_{k-r}, \dots, x_k, \dots, x_n, z_1, \dots, z_p).$$

12. Рассмотрим частные случаи.

При $k = 0$ в ПФ модуля объединяются заданные БФУ. При этом стандартная схема на рис. 7.2 упрощается за счет исключения под-схемы ПЗУ2. В полученной схеме на входы Φ_j мультиплексора подаются константы 0 и 1.

При $k = 1$ в ПФ модуля объединяются различные «половинки» заданных БФУ. При этом стандартная схема на рис. 7.2 упрощается

за счет исключения подсхемы ПЗУ2. В полученной схеме на входы Φ_j мультиплексора подаются символы $0, !x_1, x_1, 1$.

При $k = 2, 3, \dots$ в ПФ модуля объединяются различные «четвертинки», «восьмушки» и т. д. заданных функций. В этих случаях на входы Φ_j мультиплексора в стандартной схеме (рис. 7.2) в общем случае подаются не константы, отдельные переменные и их инверсии, а булевы функции k переменных. При этом специфика объединяемых в модуль БФУ может позволить упростить схему.

Таким образом, при фиксированном значении k в ПФ модуля с помощью предлагаемого метода объединяют различные фрагменты, являющиеся функциями $n - k$ переменных. Эти фрагменты выбираются из порождающей функции для образования заданной БФУ в общем случае с помощью функций k переменных, подаваемых на Φ_j -входы мультиплексора.

7.2.2. Применение отношений покрытия для упрощения порождающих формул модулей

Из предыдущего раздела следует, что каждому назначению функций Q_i соответствует определенная порождающая функция. При $R = 2^m$ существуют $2^m!$ различных ПФ.

Для исключения перебора при назначении функций Q_i можно использовать отношения покрытия, рассмотренные в разд. 1.5, выполнение которых в данном случае обеспечивает построение ПФО модуля без инверсий переменных Φ_j , обеспечивающих его настройку. Покажем, как определяются эти отношения при $m = 1, 2, 3$.

1. При $m = 1$ запишем две БФ:

$$F = Q_0!z \vee Q_1 z, \quad F_1 = Q_0 \vee Q_1 z,$$

и определим условие, при выполнении которого $F = F_1$.

При $z = 0$ это равенство выполняется, если $Q_0 = Q_0$, а при $z = 1$ — если $Q_1 = Q_0 \vee Q_1$. Следовательно, $Q_0 \leq Q_1$.

2. При $m = 2$ запишем две БФ:

$$F = Q_0!z_1!z_2 \vee Q_1!z_1 z_2 \vee Q_2 z_1!z_2 \vee Q_3 z_1 z_2;$$

$$F_1 = Q_0 \vee Q_1 z_2 \vee Q_2 z_1 \vee Q_3 z_1 z_2,$$

и определим условия, при выполнении которых $F = F_1$.

При $z_1 = 0, z_2 = 0$ это равенство выполняется, если $Q_0 = Q_0$.

При $z_1 = 0, z_2 = 1$ — если $Q_1 = Q_0 \vee Q_1$. Таким образом, $Q_0 \leq Q_1$.

При $z_1 = 1, z_2 = 0$ — если $Q_2 = Q_0 \vee Q_2$. Следовательно, $Q_0 \leq Q_2$.

При $z_1 = 1, z_2 = 1$ — если $Q_3 = Q_0 \vee Q_1 \vee Q_2 \vee Q_3$. Таким образом, $(Q_0 \leq Q_3, Q_1 \leq Q_3, Q_2 \leq Q_3)$ или $(Q_0 \leq Q_3, Q_1 \vee Q_2 \leq Q_3)$.

Из изложенного следует, что для выполнения равенства $F = F_1$ должны выполняться либо пять условий:

$$Q_0 \leq Q_1, Q_0 \leq Q_2, Q_0 \leq Q_3, Q_1 \leq Q_3, Q_2 \leq Q_3,$$

либо четыре:

$$Q_0 \leq Q_1, Q_0 \leq Q_2, Q_0 \leq Q_3, Q_1 \vee Q_2 \leq Q_3.$$

Если для функций Q_i выполняются более жесткие условия, то порождающая формула модуля может быть значительно упрощена. Например, в случае, когда кроме приведенных выше условий выполняется также условие $Q_3 = Q_2$,

$$F_2 = Q_0 \vee Q_1 z_2 \vee Q_2 z_1.$$

В случае, когда $Q_0 = 0$, $Q_2 = Q_1$, $Q_3 = 1$,

$$F_3 = Q_1 \# z_1 \# z_2.$$

3. При $m = 3$ запишем две БФ:

$$\begin{aligned} F &= Q_0! z_1! z_2! z_3 \vee Q_1! z_1! z_2 z_3 \vee Q_2! z_1 z_2! z_3 \vee Q_3! z_1 z_2 z_3 \vee \\ &\vee Q_4 z_1! z_2! z_3 \vee Q_5 z_1! z_2 z_3 \vee Q_6 z_1 z_2! z_3 \vee Q_7 z_1 z_2 z_3; \\ F_1 &= Q_0 \vee Q_1 z_3 \vee Q_2 z_2 \vee Q_3 z_2 z_3 \vee Q_4 z_1 \vee Q_5 z_1 z_3 \vee \\ &\vee Q_6 z_1 z_2 \vee Q_7 z_1 z_2 z_3, \end{aligned}$$

и определим условия, при выполнении которых $F = F_1$.

При $z_1 = 0, z_2 = 0, z_3 = 0$ это равенство выполняется, если $Q_0 = Q_0$.

При $z_1 = 0, z_2 = 0, z_3 = 1$ — если $Q_1 = Q_0 \vee Q_1$. Таким образом, $Q_0 \leq Q_1$.

При $z_1 = 0, z_2 = 1, z_3 = 0$ — если $Q_2 = Q_0 \vee Q_2$. Следовательно, $Q_0 \leq Q_2$.

При $z_1 = 0, z_2 = 1, z_3 = 1$ — если $Q_3 = Q_0 \vee Q_1 \vee Q_2 \vee Q_3$. Таким образом, $(Q_0 \leq Q_3, Q_1 \leq Q_3, Q_2 \leq Q_3)$ или $(Q_0 \leq Q_3, Q_1 \vee Q_2 \leq Q_3)$.

При $z_1 = 1, z_2 = 0, z_3 = 0$ — если $Q_4 = Q_0 \vee Q_4$. Следовательно, $Q_0 \leq Q_4$.

При $z_1 = 1, z_2 = 0, z_3 = 1$ — если $Q_5 = Q_0 \vee Q_1 \vee Q_4 \vee Q_5$. Таким образом, $(Q_0 \leq Q_5, Q_1 \leq Q_5, Q_4 \leq Q_5)$ или $(Q_0 \leq Q_5, Q_1 \vee Q_4 \leq Q_5)$.

При $z_1 = 1, z_2 = 1, z_3 = 0$ — если $Q_6 = Q_0 \vee Q_2 \vee Q_4 \vee Q_6$. Следовательно, $(Q_0 \leq Q_6, Q_2 \leq Q_6, Q_4 \leq Q_6)$ или $(Q_0 \leq Q_6, Q_2 \vee Q_4 \leq Q_6)$.

При $z_1 = 1, z_2 = 1, z_3 = 1$ — если $Q_7 = Q_0 \vee Q_1 \vee Q_2 \vee Q_3 \vee Q_4 \vee Q_5 \vee Q_6 \vee Q_7$. Таким образом, $Q_0 \leq Q_7, Q_1 \leq Q_7, Q_2 \leq Q_7, Q_3 \leq Q_7, Q_4 \leq Q_7, Q_5 \leq Q_7, Q_6 \leq Q_7$.

Из изложенного следует, что для выполнения указанного равенства должны выполняться либо 19 условий:

$$\begin{aligned} Q_0 \leq Q_1, Q_0 \leq Q_2, Q_0 \leq Q_3, Q_0 \leq Q_4, Q_0 \leq Q_5, Q_0 \leq Q_6, Q_0 \leq Q_7; \\ Q_1 \leq Q_3, Q_1 \leq Q_5, Q_1 \leq Q_7; Q_2 \leq Q_3, Q_2 \leq Q_6, Q_2 \leq Q_7; Q_3 \leq Q_7; \\ Q_4 \leq Q_5, Q_4 \leq Q_6, Q_4 \leq Q_7; Q_5 \leq Q_7; Q_6 \leq Q_7, \end{aligned}$$

либо 16 условий:

$$\begin{aligned} Q_0 \leq Q_1, Q_0 \leq Q_2, Q_0 \leq Q_3, Q_0 \leq Q_4, Q_0 \leq Q_5, Q_0 \leq Q_6, Q_0 \leq Q_7; \\ Q_1 \vee Q_2 \leq Q_3, Q_1 \vee Q_4 \leq Q_5, Q_2 \vee Q_4 \leq Q_6; \\ Q_1 \leq Q_7, Q_2 \leq Q_7, Q_3 \leq Q_7, Q_4 \leq Q_7, Q_5 \leq Q_7, Q_6 \leq Q_7. \end{aligned}$$

Из изложенного также следует, что в общем случае могут быть найдены условия, при выполнении которых порождающая формула модуля может быть представлена в виде:

$$F_1 = Q_0 \vee Q_1 z_m \vee Q_2 z_{m-1} \vee Q_3 z_{m-1} z_m \vee \dots \vee Q_{2^m-1} z_1 \dots z_m.$$

В заключение раздела отметим, что если $R < 2^m$, то некоторые из функций Q_i могут иметь одинаковые выражения или не совпадать с функциями G . Выбор функций Q_i в этом случае проводится так, чтобы получить порождающую функцию модуля, минимизированную по числу букв.

7.2.3. Объединение заданных булевых функций

В этом случае $k = 0$, и поэтому схема на рис. 7.2 преобразуется в схему на рис. 7.5.

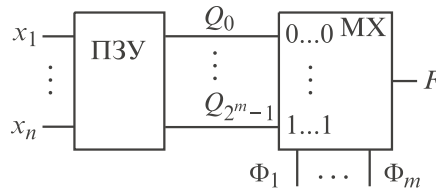


Рис. 7.5

В этой схеме $Q_i = f_i$, $m = \lceil \log N \rceil$. Выполним замену переменных $\Phi_j = z_j$. При этом

$$F_1 = \text{MX}(Q_0, \dots, Q_{2^m-1}; z_1, \dots, z_m).$$

Подставим выражения для функций Q_i в полученное соотношение, тогда

$$F_1 = F(x_1, \dots, x_n, z_1, \dots, z_m).$$

Это соотношение является ПФО модуля, число входов которого определяется следующим образом:

$$B_1 = n + \lceil \log N \rceil.$$

Алфавит настройки модуля — константы 0 и 1. При этом $z_i = \{0, 1\}$.

Этот частный случай является единственным известным аналитическим методом построения МЛМ [7].

Пример 7.1. При $k = 0$ построить логический модуль, универсальный в классе БФ в базисе $\{\&, \vee, !\}$ из трех букв.

Обозначим этот модуль как УЛМЗ. При его построении будем использовать PN -классификацию и выберем в качестве представителей типов следующие неповторные формулы:

$$\begin{aligned} f_1 &= x_1 x_2 x_3, f_2 = x_1 (x_2 \vee x_3), f_3 = x_1 \vee x_2 x_3, \\ f_4 &= x_1 \vee x_2 \vee x_3. \end{aligned}$$

В данном случае $B_1 = 5$. При $Q_i = f_{i-1}$ порождающая функция модуля имеет вид:

$$F_1(3) = MX(f_1, f_2, f_3, f_4; z_1, z_2).$$

Так как $f_1 \leq f_2 \leq f_3 \leq f_4$, то

$$\begin{aligned} F_1(3) &= f_1 \vee f_2 z_2 \vee f_3 z_1 \vee f_4 z_1 z_2 = \\ &= x_1 x_2 x_3 \vee x_1 (x_2 \vee x_3) z_2 \vee (x_1 \vee x_2 x_3) z_1 \vee \\ &\vee (x_1 \vee x_2 \vee x_3) z_1 z_2 = x_1 x_2 x_3 \vee x_1 x_2 z_2 \vee \\ &\vee x_1 x_3 z_2 \vee x_1 z_1 \vee x_2 x_3 z_1 \vee x_1 z_1 z_2 \vee x_2 z_1 z_2 \vee x_3 z_1 z_2 = \\ &= x_1 (x_2 x_3 \vee x_2 z_2 \vee x_3 z_2) \vee x_1 z_1 \vee z_1 (x_2 x_3 \vee x_2 z_2 \vee x_3 z_2) = \\ &= x_1 ((x_2 \vee x_3) z_2 \vee x_2 x_3) \vee x_1 z_1 \vee ((x_2 \vee x_3) z_2 \vee x_2 x_3) z_1 = \\ &= (x_1 \vee z_1) ((x_2 \vee x_3) z_2 \vee x_2 x_3) \vee x_1 z_1. \end{aligned}$$

Полученная БФ содержит девять букв. Изменим базис в ней:

$$\begin{aligned} F_1(3) &= (x_1 \vee z_1) (x_2 \# x_3 \# z_2) \vee x_1 z_1 = \\ &= z_1 \# x_1 \# (z_2 \# x_2 \# x_3). \end{aligned}$$

Эта формула из пяти букв реализуется схемой, состоящей из двух трехвходовых МЭ (рис. 7.6).

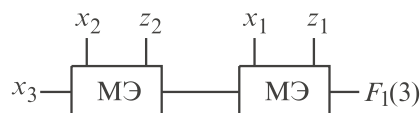


Рис. 7.6

В этом модуле три входа — x_1, x_2, x_3 являются информационными, а входы z_1 и z_2 — настроечными. Отметим, что рассмотренный модуль является однородным настраиваемым каскадом Макхопадхия из двух ячеек [3] и может быть построен также с помощью метода «покрывающих деревьев» (разд. 1.2.3).

Пример 7.2. При $k = 0$ построить еще один модуль УЛМЗ.

Выполним в бесскобочной формуле для $F_1(3)$ другое вынесение за скобки:

$$\begin{aligned} F_1(3) &= x_1(x_2x_3 \vee (x_2 \vee x_3)z_2) \vee \\ &\vee (x_1 \vee x_2x_3 \vee (x_2 \vee x_3)z_2)z_1 = \\ &= x_1(x_1z_1 \vee x_2x_3 \vee (x_2 \vee x_3)z_2) \vee \\ &\vee (x_1z_1 \vee x_2x_3 \vee (x_2 \vee x_3)z_2)z_1 = \\ &= (x_1 \vee z_1)(x_1z_1 \vee x_2x_3 \vee (x_2 \vee x_3)z_2). \end{aligned}$$

Полученная формула в базисе $\{\&, \vee, !\}$, так же как и предыдущая порождающая формула в этом базисе, содержит девять букв [16].

7.2.4. Объединение различных «половинок» заданных булевых функций

При $k = 1$ в порождающую функцию модуля объединяются разнотипные «половинки» заданных БФУ — разнотипные фрагменты длиной 2^{n-1} столбцов значений заданных булевых функций. Эти фрагменты являются БФУ, зависящими от $n - 1$ переменных. Некоторые из фрагментов могут зависеть от всех или от части этих переменных несущественно.

Выполним разложение Шеннона каждой из N заданных БФУ по одной и той же входной переменной, например x_1 . При этом формируются $2N$ остаточных функций (фрагментов) $g_0, \dots, g_{2N-1}$.

Предположим, что в этом случае число различных фрагментов (их типов) равно R , а $m = \lceil \log R \rceil$. Обозначим эти фрагменты переменными $G_0, \dots, G_{R-1}$. Если выполнить назначение $Q_i = G_l$, то и в этом случае может быть использована стандартная схема (рис. 7.5), в которой ПЗУ имеет $n - 1$ входов, помеченных символами $x_2, \dots, x_n$.

Выполним в этой схеме замену переменных $\Phi_j = z_j$. При этом

$$F_2 = \text{MX}(Q_0, \dots, Q_{2^m-1}; z_1, \dots, z_m).$$

Подставим выражения для функций Q_i в полученное соотношение, тогда

$$F_2 = F(x_2, \dots, x_n, z_1, \dots, z_m).$$

Это соотношение является порождающей формулой модуля, число входов которого определяется следующим образом:

$$B_2 = n - 1 + \lceil \log R \rceil.$$

Алфавит настройки модуля: $0, !x_1, x_1, 1$. При этом $z_i = \{0, !x_1, x_1, 1\}$. Это позволяет выбрать из ПФ модуля любые два фрагмента длиной 2^{n-1} и расположить их в требуемом порядке для образования каждой из заданных БФУ.

Сформулируем необходимое (но не достаточное) условие, при выполнении которого при $t = 1, \dots, N$ алфавит настройки может быть сокращен за счет исключения из него символа $!x_1$.

При $s < r$, если фрагмент Q_s находится в порождающей функции «левее» фрагмента Q_r , то и в функции f_t фрагмент Q_s должен находиться «левее» фрагмента Q_r :

$$f_t = !x_1 Q_s \vee x_1 Q_r.$$

Если для функций Q_i выполняются отношения покрытия, то

$$F_2 = Q_0 \vee Q_1 z_m \vee Q_2 z_{m-1} \vee Q_3 z_{m-1} z_m \vee \dots \vee Q_{R-1} z_1 \dots z_m.$$

Настройка модуля на реализацию функции f_t определяется в результате составления и решения мультиплексорным методом логического уравнения:

$$\begin{aligned} f_t &= F_2; \\ \text{MX}(Q_s, Q_r; x_1) &= \text{MX}(Q_0, \dots, Q_{R-1}; z_1, \dots, z_m); \\ z_1 &= \text{MX}(b_1, b_2; x_1); \\ &\vdots \\ z_m &= \text{MX}(b_{2^{m-1}}, b_{2^m}; x_1), \end{aligned}$$

где $b_1, \dots, b_{2^{m-1}} = \text{bin } s$; $b_2, \dots, b_{2^m} = \text{bin } r$.

Из изложенного следует, что для сокращения числа входов модуля значение R должно быть минимально. Это достигается:

- выбором соответствующих представителей классов;
- выбором переменной, по которой выполняется разложение Шеннона.

Изложенный метод строит одновыходные модули УЛМ q , число входов в которых определяется соотношением [3]

$$B_3 = q - 1 + \left\lceil \log \left(2 + \sum_{i=\lfloor q/2 \rfloor}^{q-1} T_1(i) \right) \right\rceil,$$

где $T_1(i)$ — число PN -типов формул, неповторных в рассматриваемом базисе из i букв.

Значения B_3 для $q = 2 \div 10$ приведены в табл. 7.1.

Т а б л и ц а 7.1

q	2	3	4	5	6	7	8	9	10
B_3	3	4	6	8	11	13	16	18	21

Метод построения МЛМ при $k = 1$ впервые был рассмотрен в [3, 17].

Пример 7.3. При $k = 1$ построить модуль УЛМЗ.

Выберем представителей PN -типов неповторных формул, выполним для каждого из них разложение Шеннона по переменной x_3 и представим их в мультиплексорной форме:

$$\begin{aligned} f_1 &= x_1 x_2 x_3 = \text{MX}(0, x_1 x_2; x_3), \\ f_2 &= x_1 (x_2 \vee x_3) = \text{MX}(x_1 x_2, x_1; x_3), \\ f_3 &= x_1 \vee x_2 x_3 = \text{MX}(x_1, x_1 \vee x_2; x_3), \\ f_4 &= x_1 \vee x_2 \vee x_3 = \text{MX}(x_1 \vee x_2, 1; x_3). \end{aligned}$$

Таким образом, $g_0 = 0$, $g_1 = x_1 x_2$, $g_2 = x_1 x_2$, $g_3 = x_1$, $g_4 = x_1$, $g_5 = x_1 \vee x_2$, $g_6 = x_1 \vee x_2$, $g_7 = 1$. Разнотипными фрагментами являются следующие: $G_0 = 0$, $G_1 = x_1 x_2$, $G_2 = x_1$, $G_3 = x_1 \vee x_2$, $G_4 = 1$.

Следовательно, в данном случае $R = 5$, $m = 3$, и поэтому (как и в случае $k = 0$) $B_3 = 5$.

С целью минимизации числа входов модуля разложим каждую из выбранных БФ по переменной x_1 :

$$\begin{aligned} f_1 &= x_1 x_2 x_3 = \text{MX}(0, x_2 x_3; x_1), \\ f_2 &= x_1 (x_2 \vee x_3) = \text{MX}(0, x_2 \vee x_3; x_1), \\ f_3 &= x_1 \vee x_2 x_3 = \text{MX}(x_2 x_3, 1; x_1), \\ f_4 &= x_1 \vee x_2 \vee x_3 = \text{MX}(x_2 \vee x_3, 1; x_1). \end{aligned}$$

Таким образом, $g_0=0$, $g_1=x_2x_3$, $g_2=0$, $g_3=x_2\vee x_3$, $g_4=x_2x_3$, $g_5=1$, $g_6=x_2\vee x_3$, $g_7=1$. Разнотипными фрагментами в данном случае являются следующие: $G_0=0$, $G_1=x_2x_3$, $G_2=x_2\vee x_3$, $G_4=1$. Следовательно, $R=4$, $m=2$, и поэтому $B_3=4$. В этом модуле входы x_2 и x_3 являются информационными, а остальные два входа — z_1 и z_2 — информационно-настроечными.

Выполним назначение функций Q_i следующим образом: $Q_i=G_i$. При этом таблица истинности ПФ модуля имеет вид:

$$F_2(z_1, z_2, x_2, x_3) = |0000 \ 0001 \ 0111 \ 1111|^T.$$

Запишем ПФ в мультиплексорной форме:

$$F_2(3) = \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2).$$

Запишем выбранные БФ в мультиплексорной форме, используя функции Q_i :

$$\begin{aligned} f_1 &= \text{MX}(Q_0, Q_1; x_1), & f_3 &= \text{MX}(Q_1, Q_3; x_1), \\ f_2 &= \text{MX}(Q_0, Q_2; x_1), & f_4 &= \text{MX}(Q_2, Q_3; x_1). \end{aligned}$$

Так как во всех этих БФУ номер первой остаточной функции меньше номера второй остаточной, то необходимое условие для получения безынверсных настроек выполняется. Применяя мультиплексорный метод, определим настройки модуля:

$$\begin{aligned} f_1 &= F_2(3), \\ \text{MX}(Q_0, Q_1; x_1) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2), \\ z_1 &= \text{MX}(0, 0; x_1) = 0, \\ z_2 &= \text{MX}(0, 1; x_1) = x_1; \end{aligned}$$

$$\begin{aligned} f_2 &= F_2(3), \\ \text{MX}(Q_0, Q_2; x_1) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2), \\ z_1 &= \text{MX}(0, 1; x_1) = x_1, \\ z_2 &= \text{MX}(0, 0; x_1) = 0; \end{aligned}$$

$$\begin{aligned} f_3 &= F_2(3), \\ \text{MX}(Q_1, Q_3; x_1) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2), \\ z_1 &= \text{MX}(0, 1; x_1) = x_1, \\ z_2 &= \text{MX}(1, 1; x_1) = 1; \end{aligned}$$

$$\begin{aligned}
f_4 &= F_2(3), \\
MX(Q_2, Q_3; x_1) &= MX(Q_0, Q_1, Q_2, Q_3; z_1, z_2), \\
z_1 &= MX(1, 1; x_1) = 1, \\
z_2 &= MX(0, 1; x_1) = x_1.
\end{aligned}$$

Таким образом, в данном случае необходимое условие для получения безынверсных настроек является также и достаточным: $z_i = \{0, x_1, 1\}$.

Построенная порождающая функция позволяет дополнительно реализовать БФУ $f_5 = x_1 \# x_2 \# x_3 = MX(Q_1, Q_2; x_1)$:

$$\begin{aligned}
f_5 &= F_2(3), \\
MX(Q_1, Q_2; x_1) &= MX(Q_0, Q_1, Q_2, Q_3; z_1, z_2), \\
z_1 &= MX(0, 1; x_1) = x_1, \\
z_2 &= MX(1, 0; x_1) = !x_1,
\end{aligned}$$

что делает его универсальным в классе пороговых функций трех переменных при $z_1 = \{0, x_1, 1\}$, $z_2 = \{0, !x_1, x_1, 1\}$. Таким образом, в данном случае указанное необходимое условие не является достаточным.

Модуль УЛМЗ с другими настройками на функцию f_5 описан в [5, 18, 19].

Определим порождающую формулу модуля. Так как $Q_0 \leq Q_1 \leq Q_2 \leq Q_3$, то

$$\begin{aligned}
F_2(3) &= Q_0 \vee Q_1 z_2 \vee Q_2 z_1 \vee Q_3 z_1 z_2 = \\
&= x_2 x_3 z_2 \vee (x_2 \vee x_3) z_1 \vee z_1 z_2 = z_1 (z_2 \vee x_2 \vee x_3) \vee z_2 x_2 x_3 = \\
&= MX(z_2 x_2 x_3, z_2 \vee x_2 \vee x_3; z_1).
\end{aligned}$$

Полученные порождающие формулы могут быть реализованы схемами из четырех и трех логических элементов (рис. 7.7, 7.8).

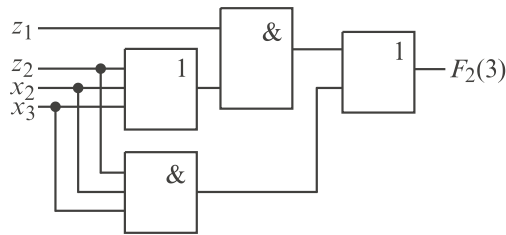


Рис. 7.7

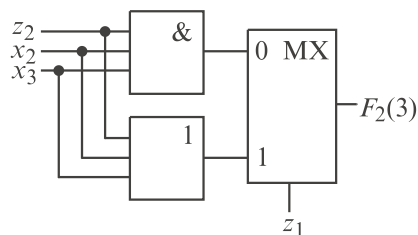


Рис. 7.8

Так как функция $F_2(3)$ является самодвойственной (разд. 10.10 и [20]), то

$$F_2(3) = (z_1 \oplus z_2)(z_1 \oplus x_2)(z_1 \oplus x_3) \oplus z_1.$$

Эта порождающая формула модуля реализуется схемой из пяти логических элементов (рис. 7.9). Построенная порождающая функция модуля может быть реализована мультиплексорным методом схемой из двух мажоритарных элементов (рис. 4.16). Для рассмотренного модуля в дальнейшем будем использовать обозначение, приведенное на рис. 7.10.

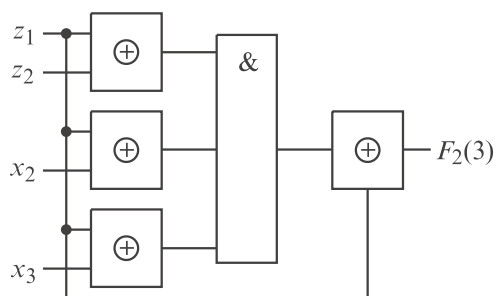


Рис. 7.9

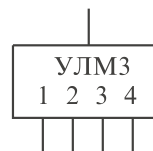


Рис. 7.10

При представлении порождающей формулы в дизъюнктивной нормальной форме модуль может быть реализован схемой (рис. 7.11).

Это позволило на пяти микросхемах малого уровня интеграции реализовать четыре модуля УЛМЗ с прямым и инверсным выходами каждый, которые образовали одну из микросборок среднего уровня интеграции с 20 внешними выводами. Это, в свою очередь, позволило согласовать большое число внешних выводов пяти микросхем, помещающихся в корпус микросборки, с существенно меньшим числом ее внешних выводов, обеспечивая при этом высокую логическую эффективность микросборки [21].

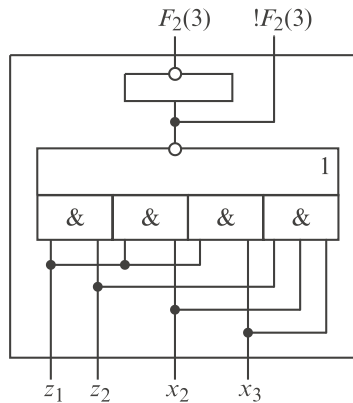


Рис. 7.11

7.2.5. Объединение различных «половинок» булевых функций, *PN*-однотипных с заданными функциями

Рассмотрим этот метод применительно к формулам, неповторным в базисе $\{\&, \vee, !\}$ из h букв.

Выполним разложение Шеннона по одной и той же переменной каждой из заданных БФ, так чтобы число разнотипных фрагментов длиной 2^{n-1} было минимальным.

Разобьем полученные разнотипные фрагменты на основе принципа двойственности на две группы так, чтобы:

— если в первой группе находится некоторый фрагмент, соответствующий неповторной пороговой формуле, то двойственный ему фрагмент должен находиться во второй группе;

— в первой группе должны оказаться все фрагменты, отличные от констант, соответствующие фрагментам представителей *NP*N-типов непороговых неповторных формул;

— в первой группе, по возможности, должны выполняться отношения покрытия.

Предположим, что G_l — некоторый фрагмент первой группы. Если выполнить назначение функций Q следующим образом:

$$Q_i = G_l, \text{ где } i = 0, \dots, 2^{n-1} - 1;$$

$$Q_r = !G_l, \text{ где } 2^{n-1} \leq r \leq i + 2^{n-1} \leq 2^n - 1,$$

и построить функцию V , которая объединяет фрагменты первой группы, то формула

$$F = z_1 \oplus V$$

может использоваться в качестве порождающей формулы модуля.

При этом, например, если $V = MX(Q_0, Q_1, Q_2, Q_3; z_2, z_3)$, то

$$F = MX(Q_0, \dots, Q_7; z_1, z_2, z_3),$$

где $Q_4 = !Q_0, Q_5 = !Q_1, Q_6 = !Q_2, Q_7 = !Q_3$.

Следовательно, если первая группа содержит R фрагментов, то модуль имеет число входов, определяемое соотношением

$$B_4 = h - 1 + \lceil \log R \rceil + 1 = h + \lceil \log R \rceil.$$

Изложенный метод базируется на следующем свойстве булевых функций: в разложении Шеннона по одной переменной заданной функции замена каждой остаточной функции, не являющейся константой, на антидвойственную функцию (разд. 10.5 и [20]) позволяет строить новую функцию, PN -однотипную с заданной.

При этом отметим, что для неповторных пороговых формул при разложении Шеннона по переменной с наибольшим весом одна из остаточных функций является константой. Для неповторных непороговых формул обе остаточные функции не являются константами.

Пример 7.4. Построить модуль УЛМЗ, используя предлагаемый метод.

Разобьем на основе принципа двойственности разнотипные фрагменты, найденные в предыдущем примере, на две группы: $\{G_0 = 0, G_1 = x_2 x_3\}$ и $\{G_2 = x_2 \vee x_3, G_3 = 1\}$.

Выполним назначение функций Q_i : $Q_0 = 0, Q_1 = x_2 x_3, Q_2 = !Q_0 = 1, Q_3 = !Q_1 = !x_2 \vee !x_3$.

При этом порождающая функция модуля имеет вид:

$$F_3(z_1, z_2, x_2, x_3) = \begin{vmatrix} 0000 & 0001 & 1111 & 1110 \end{vmatrix}^T.$$

Представим порождающую формулу модуля в мультиплексорной форме:

$$F_3(3) = MX(Q_0, Q_1, Q_2, Q_3; z_1, z_2).$$

Запишем заданные БФ или однотипные с ними через функции Q_i :

$$\begin{aligned} f_1 &= MX(Q_0, Q_1; x_1) = x_1 x_2 x_3, \\ f_{21} &= MX(Q_0, Q_3; x_1) = x_1 (!x_2 \vee !x_3), \\ f_3 &= MX(Q_1, Q_2; x_1) = x_1 \vee x_2 x_3, \\ f_{41} &= MX(Q_2, Q_3; x_1) = !x_1 \vee !x_2 \vee !x_3. \end{aligned}$$

Так как во всех этих функциях номер первой остаточной меньше номера второй остаточной, то необходимое условие для получения

безынверсных настроек выполняется. Применяя мультиплексорный метод, определим настройки модуля:

$$\begin{aligned} f_1 &= F_3(3), \\ \text{MX}(Q_0, Q_1; x_1) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2), \\ z_1 &= \text{MX}(0, 0; x_1) = 0, \\ z_2 &= \text{MX}(0, 1; x_1) = x_1; \end{aligned}$$

$$\begin{aligned} f_{21} &= F_3(3), \\ \text{MX}(Q_0, Q_3; x_1) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2), \\ z_1 &= \text{MX}(0, 1; x_1) = x_1, \\ z_2 &= \text{MX}(0, 1; x_1) = x_1; \end{aligned}$$

$$\begin{aligned} f_3 &= F_3(3), \\ \text{MX}(Q_1, Q_2; x_1) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2), \\ z_1 &= \text{MX}(0, 1; x_1) = x_1, \\ z_2 &= \text{MX}(1, 0; x_1) = !x_1; \end{aligned}$$

$$\begin{aligned} f_{41} &= F_3(3), \\ \text{MX}(Q_2, Q_3; x_1) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2), \\ z_1 &= \text{MX}(1, 1; x_1) = 1, \\ z_2 &= \text{MX}(0, 1; x_1) = x_1. \end{aligned}$$

Таким образом, в данном случае необходимое условие для получения безынверсных настроек не является достаточным и $z_i = \{0, !x_1, x_1, 1\}$.

Так как в данном случае $V = z_2 x_2 x_3$, то

$$F_3(3) = z_1 \oplus z_2 x_2 x_3.$$

Эта порождающая формула реализуется схемой из двух логических элементов (рис. 7.12).

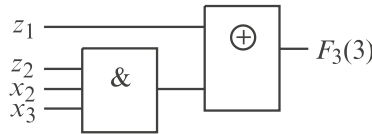


Рис. 7.12

Основы изложенного метода описаны в [22].

7.2.6. Объединение различных фрагментов, меньших чем «половинки» заданных булевых функций

Пример 7.5. При $k = 2$ построить модуль УЛМЗ.

В данном случае в порождающей функции модуля объединяются различные «четвертинки» столбцов значений заданных функций.

Выполним для каждой БФ разложение Шеннона одновременно по переменным x_1 и x_2 и представим эти формулы в мультиплексорной форме:

$$\begin{aligned}f_1 &= x_1 x_2 x_3 = \text{MX}(0, 0, 0, x_3; x_1, x_2); \\f_2 &= x_1 (x_2 \vee x_3) = \text{MX}(0, 0, x_3, 1; x_1, x_2); \\f_3 &= x_1 \vee x_2 x_3 = \text{MX}(0, x_3, 1, 1; x_1, x_2); \\f_4 &= x_1 \vee x_2 \vee x_3 = \text{MX}(x_3, 1, 1, 1; x_1, x_2).\end{aligned}$$

В данном случае $G_0 = 0$, $G_1 = x_3$, $G_2 = 1$ и $R = 3$.

Назначим функции Q_i , вводя функцию Q_2 , отсутствующую среди выделенных фрагментов: $Q_0 = 0$, $Q_1 = x_3$, $Q_2 = !x_3$, $Q_3 = 1$.

Для каждой из заданных БФ построим мультиплексорную декомпозицию вида:

$$f_t = \text{MX}(Q_0, \dots, Q_3; \Phi_1^t, \Phi_2^t),$$

определив функции Φ_1^t и Φ_2^t . Найденные функции подставим в правую часть этого выражения. При этом

$$\text{MX}(Q_0, Q_0, Q_0, Q_1; x_1, x_2) = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1^1, \Phi_2^1),$$

$$\begin{aligned}\Phi_1^1 &= \text{MX}(0, 0, 0, 0; x_1, x_2) = 0, \\ \Phi_2^1 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2, \\ f_1 &= \text{MX}(Q_0, Q_1, Q_2, Q_3; 0, x_1 x_2);\end{aligned}$$

$$\text{MX}(Q_0, Q_0, Q_1, Q_3; x_1, x_2) = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1^2, \Phi_2^2),$$

$$\begin{aligned}\Phi_1^2 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2, \\ \Phi_2^2 &= \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1, \\ f_2 &= \text{MX}(Q_0, Q_1, Q_2, Q_3; x_1 x_2, x_1);\end{aligned}$$

$$\text{MX}(Q_0, Q_1, Q_3, Q_3; x_1, x_2) = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1^3, \Phi_2^3),$$

$$\begin{aligned}\Phi_1^3 &= \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1, \\ \Phi_2^3 &= \text{MX}(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2, \\ f_3 &= \text{MX}(Q_0, Q_1, Q_2, Q_3; x_1, x_1 \vee x_2);\end{aligned}$$

$$MX(Q_1, Q_3, Q_3, Q_3; x_1, x_2) = MX(Q_0, Q_1, Q_2, Q_3; \Phi_1^4, \Phi_2^4),$$

$$\Phi_1^4 = MX(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2,$$

$$\Phi_2^4 = MX(1, 1, 1, 1; x_1, x_2) = 1,$$

$$f_4 = MX(Q_0, Q_1, Q_2, Q_3; x_1 \vee x_2, 1).$$

Из сопоставления построенных декомпозиций следует, что в качестве порождающей формулы в данном случае может быть выбрана БФУ вида:

$$F_4(3) = MX(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2),$$

где $\Phi_1 = x_1 \# z_1 \# z_2$, $\Phi_2 = x_1 \# z_3 \# z_4$, $z_1 = z_3 = \{0, 1, x_2\}$, $z_2 = z_4 = \{0, 1\}$. Если в эту порождающую формулу подставить выражения для функций Q_i , то

$$F_4(3) = MX(\Phi_1, \Phi_2; x_3).$$

Таким образом, в данном случае может быть построена схема, представленная на рис. 7.13.

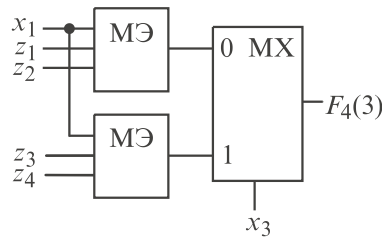


Рис. 7.13

Эта схема по всем показателям менее эффективна по сравнению со схемами, построенными с помощью предлагаемого метода при $k = 0, 1$.

Пример эффективного объединения «четвертинок» заданных функций при $n = 3$ приведен в следующем разделе (пример 7.8).

При $n = 3$ «восьмушки» заданных БФУ формируются при $k = n$ и поэтому в данном случае не рассматриваются.

Пример эффективного объединения «восьмушек» функций при $n = 5$ приведен в разд. 8.2.

7.3. Модифицированный мультиплексорный метод

1. При $k = \{0, \dots, n-1\}$ зафиксируем величину k .
2. Выберем k переменных и проведем разложение Шеннона каждой из заданных БФУ одновременно по всем этим переменным. В результате формируется система соотношений вида:

$$f_t = \text{MX}(g_0^t, \dots, g_{2^k-1}^t; x_1, \dots, x_k),$$

где $t = 1, \dots, N$.

3. В функции f_t выбираются разнотипные фрагменты G_j^t , где $j = \{0, \dots, R_t-1\}$; R_t — число разнотипных фрагментов в функции f_t .

4. Определяется величина $R = \max(R_1, \dots, R_N)$.

5. Определяется величина $m = \lceil \log R \rceil$.

6. Для функции f_t выполним назначение функций Q_i^t , где $i = 0, \dots, 2^m-1$.

7. Для функции f_t построим мультиплексорную декомпозицию вида:

$$f_t = \text{MX}(Q_0^t, \dots, Q_{2^m-1}^t; \Phi_1^t, \dots, \Phi_m^t).$$

8. Объединим функции Q_i^t ($i = \text{const}, t = 1, \dots, N$) в порождающую функцию, которую обозначим символом Q_i . Эта функция в общем случае зависит от d информационных и v настроечных или информационно-настроечных переменных. Значения переменных d и v зависят от свойств функций Q_i^t и метода их объединения в ПФ.

9. Объединим функции Φ_j^t ($j = \text{const}, t = 1, \dots, N$) в порождающую функцию, которую обозначим символом Φ_j . Эта функция в общем случае зависит от p информационных и w настроечных или информационно-настроечных переменных. Значения переменных p и w зависят от свойств функций Φ_j^t и метода их объединения в ПФ.

10. Построим БФ вида:

$$F = \text{MX}(Q_0, \dots, Q_{2^m-1}; \Phi_1, \dots, \Phi_m), \quad (7.2)$$

которая является искомой порождающей формулой модуля.

Такому построению модуля соответствует стандартная схема, приведенная на рис. 7.14.

Специфика заданных или выбранных в ходе построения модуля функций, однотипных с заданными, может резко упростить стандартную схему, например за счет исключения одного ПЗУ.

Пример 7.6. При $k = 1$ построить модуль УЛМЗ.

Предположим, что модуль должен реализовать путем настройки следующие формулы: $f_1 = x_1 x_2 x_3$, $f_2 = (!x_1 \vee x_2) x_3$, $f_3 = x_1 x_2 \vee x_3$, $f_4 = !x_1 \vee x_2 \vee x_3$.

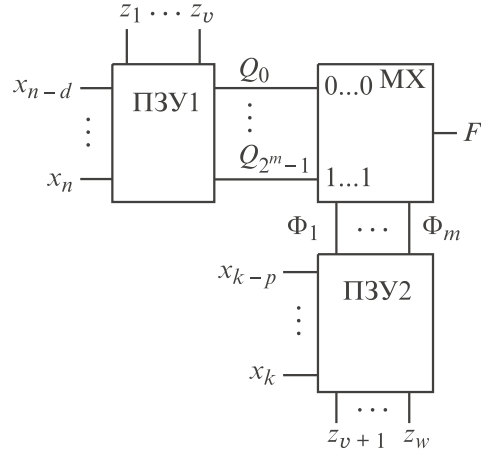


Рис. 7.14

Выполним для каждой из этих БФ разложение Шеннона по переменной x_1 :

$$\begin{aligned} f_1 &= x_1 x_2 x_3 = \text{MX}(0, x_2 x_3; x_1); \\ f_2 &= (!x_1 \vee x_2) x_3 = \text{MX}(x_3, x_2 x_3; x_1); \\ f_3 &= x_1 x_2 \vee x_3 = \text{MX}(x_3, x_2 \vee x_3; x_1); \\ f_4 &= !x_1 \vee x_2 \vee x_3 = \text{MX}(1, x_2 \vee x_3; x_1). \end{aligned}$$

На основе построенных декомпозиций следует, что

$$Q_0 = z_1, Q_1 = z_2 \# x_2 \# x_3, \Phi = x_1,$$

где $z_1 = \{0, x_3, 1\}$, $z_2 = \{0, 1\}$.

При этом порождающая формула модуля имеет вид:

$$F_5(3) = \text{MX}(Q_0, Q_1; \Phi).$$

Подставляя в это соотношение выражения для функций Q_i и Φ , получим искомую порождающую формулу:

$$F_5(3) = \text{MX}(z_1, z_2 \# x_2 \# x_3; x_1).$$

Этой формуле соответствует схема с пятью входами, состоящая из двух трехвходовых элементов (рис. 7.15).

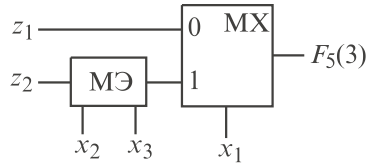


Рис. 7.15

Пример 7.7. При $k = 2$ построить модуль УЛМЗ.

Предположим первоначально, что модуль должен реализовать путем настройки следующие формулы: $f_1 = x_1 x_2 x_3$, $f_2 = x_1 (x_2 \vee x_3)$, $f_3 = x_1 \vee x_2 x_3$, $f_4 = x_1 \vee x_2 \vee x_3$.

Выполним для каждой из этих БФ разложение Шеннона одновременно по переменным x_1 и x_2 и представим эти формулы в мультиплексорной форме:

$$\begin{aligned} f_1 &= x_1 x_2 x_3 = \text{MX}(0, 0, 0, x_3; x_1, x_2); \\ f_2 &= x_1 (x_2 \vee x_3) = \text{MX}(0, 0, x_3, 1; x_1, x_2); \\ f_3 &= x_1 \vee x_2 x_3 = \text{MX}(0, x_3, 1, 1; x_1, x_2); \\ f_4 &= x_1 \vee x_2 \vee x_3 = \text{MX}(x_3, 1, 1, 1; x_1, x_2). \end{aligned}$$

Перечислим разнотипные фрагменты в каждой из формул: $\{0, x_3\}$, $\{0, x_3, 1\}$, $\{0, x_3, 1\}$, $\{x_3, 1\}$.

В данном случае $R_1 = R_4 = 2$, $R_2 = R_3 = 3$, и поэтому $R = 3$.

Заменяем формулы f_2 и f_3 на PN -однотипные формулы f_{21} и f_{31} , такие что $R_{21} = R_{31} = 2$. Это позволяет получить $R = 2$.

Таковыми формулами являются: $f_{21} = (x_1 \vee x_2) x_3$, $f_{31} = x_1 x_2 \vee x_3$.

При этом набор БФ, реализуемых модулем путем настройки, приобретает вид:

$$\begin{aligned} f_1 &= \text{MX}(0, 0, 0, x_3; x_1, x_2); \\ f_{21} &= \text{MX}(0, x_3, x_3, x_3; x_1, x_2); \\ f_{31} &= \text{MX}(x_3, x_3, x_3, 1; x_1, x_2); \\ f_4 &= \text{MX}(x_3, 1, 1, 1; x_1, x_2). \end{aligned}$$

Для каждой из этих формул построим мультиплексорную декомпозицию:

$$\begin{aligned} f_1 &= \text{MX}(0, x_3; \Phi^1), \\ \text{MX}(0, 0, 0, x_3; x_1, x_2) &= \text{MX}(0, x_3; \Phi^1), \\ \Phi^1 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2, \\ f_1 &= \text{MX}(0, x_3; x_1 x_2); \\ f_{21} &= \text{MX}(0, x_3; \Phi^2), \\ \text{MX}(0, x_3, x_3, x_3; x_1, x_2) &= \text{MX}(0, x_3; \Phi^2), \\ \Phi^2 &= \text{MX}(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2, \\ f_{21} &= \text{MX}(0, x_3; x_1 \vee x_2); \end{aligned}$$

$$\begin{aligned}
f_{31} &= \text{MX}(x_3, 1; \Phi^3), \\
\text{MX}(x_3, x_3, x_3, 1; x_1, x_2) &= \text{MX}(x_3, 1; \Phi^3), \\
\Phi^3 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2, \\
f_{31} &= \text{MX}(x_3, 1; x_1 x_2); \\
f_4 &= \text{MX}(x_3, 1; \Phi^4), \\
\text{MX}(x_3, 1, 1, 1; x_1, x_2) &= \text{MX}(x_3, 1; \Phi^4), \\
\Phi^4 &= \text{MX}(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2, \\
f_4 &= \text{MX}(x_3, 1; x_1 \vee x_2).
\end{aligned}$$

На основе построенных декомпозиций следует, что

$$Q_0 = z_1, \quad Q_1 = z_2, \quad \Phi = z_3 \# x_1 \# x_2,$$

где $z_1 = \{0, x_3\}$, $z_2 = \{x_3, 1\}$, $z_3 = \{0, 1\}$.

При этом порождающая формула модуля имеет вид:

$$F_6(3) = \text{MX}(Q_0, Q_1; \Phi).$$

Подставляя в это соотношение выражения для функций Q_i и Φ , получим искомую порождающую формулу:

$$F_6(3) = \text{MX}(z_1, z_2; z_3 \# x_1 \# x_2).$$

Этой формуле соответствует схема с пятью входами, состоящая из двух трехвходовых элементов (рис. 7.16).

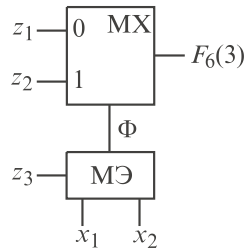


Рис. 7.16

Таким образом, переход от объединения «половинок» функций к объединению их «четвертинок» не упростил модуль.

Пример 7.8. При $k = 2$ построить модуль УЛМЗ.

С целью упрощения модуля вновь изменим набор объединяемых формул. При этом заменим формулы f_{21} и f_4 на PN -однотипные.

Таковыми формулами, не изменяющими значения $R = 2$, являются формулы $f_{22} = (!x_1 \vee !x_2) x_3$ и $f_{41} = !x_1 \vee !x_2 \vee x_3$.

Выполним для каждой БФ из нового множества разложение Шеннона одновременно по переменным x_1 и x_2 и представим эти формулы в мультиплексорной форме:

$$\begin{aligned} f_1 &= x_1 x_2 x_3 = \text{MX}(0, 0, 0, x_3; x_1, x_2); \\ f_{22} &= (!x_1 \vee !x_2) x_3 = \text{MX}(x_3, x_3, x_3, 0; x_1, x_2); \\ f_{31} &= x_1 x_2 \vee x_3 = \text{MX}(x_3, x_3, x_3, 1; x_1, x_2); \\ f_{41} &= !x_1 \vee !x_2 \vee x_3 = \text{MX}(1, 1, 1, x_3; x_1, x_2). \end{aligned}$$

Для каждой из этих формул построим мультиплексорную декомпозицию:

$$\begin{aligned} f_1 &= \text{MX}(0, x_3; \Phi^1), \\ \text{MX}(0, 0, 0, x_3; x_1, x_2) &= \text{MX}(0, x_3; \Phi^1), \\ \Phi^1 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2, \\ f_1 &= \text{MX}(0, x_3; x_1 x_2); \\ f_{22} &= \text{MX}(x_3, 0; \Phi^2), \\ \text{MX}(x_3, x_3, x_3, 0; x_1, x_2) &= \text{MX}(x_3, 0; \Phi^2), \\ \Phi^2 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2, \\ f_{22} &= \text{MX}(x_3, 0; x_1 x_2); \\ f_{31} &= \text{MX}(x_3, 1; \Phi^3), \\ \text{MX}(x_3, x_3, x_3, 1; x_1, x_2) &= \text{MX}(x_3, 1; \Phi^3), \\ \Phi^3 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2, \\ f_{31} &= \text{MX}(x_3, 1; x_1 x_2); \\ f_{41} &= \text{MX}(1, x_3; \Phi^4), \\ \text{MX}(1, 1, 1, x_3; x_1, x_2) &= \text{MX}(1, x_3; \Phi^4), \\ \Phi^4 &= \text{MX}(0, 0, 0, 1; x_1, x_2) = x_1 x_2, \\ f_{41} &= \text{MX}(1, x_3; x_1 x_2). \end{aligned}$$

На основе построенных декомпозиций следует, что

$$Q_0 = z_1, Q_1 = z_2, \Phi = x_1 x_2,$$

где $z_i = \{0, x_3, 1\}$.

При этом порождающая формула модуля имеет вид:

$$F_7(3) = MX(Q_0, Q_1; \Phi).$$

Подставляя в это соотношение выражения для функций Q_i и Φ , получим искомую порождающую формулу:

$$F_7(3) = MX(z_1, z_2; x_1 x_2).$$

Этой формуле соответствует схема, наиболее простая из известных схем модуля УЛМЗ с четырьмя входами (рис. 7.17). В этой схеме входы x_1 и x_2 являются информационными, а входы z_1 и z_2 — информационно-настроечными.

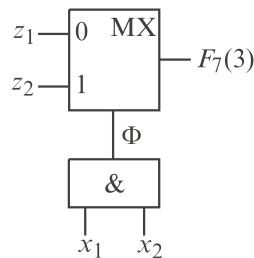


Рис. 7.17

Таким образом, модификация мультиплексорного метода позволила эффективно объединить «четвертинки» заданных БФУ, что не удавалось выполнить при непосредственном применении мультиплексорного метода.

Модификация мультиплексорного метода была впервые использована при построении модуля УЛМЗ [23], порождающая формула которого имеет вид:

$$F_8(3) = MX(z_1, z_2; !x_1 x_2).$$

Эта формула совпадает с порождающей формулой трехмембранного пневматического реле Р-3Ф, которое входит в состав серии пневматических элементов УСЭПИА [24, 25].

Основы изложенного в настоящем разделе метода описаны в [26].

7.4. Дополнительные методы

7.4.1. Использование преобразований булевых функций

В разд. 1.7 рассмотрены новые преобразования БФУ вида:

$$f = (!f(x_i=0)!x_i \vee f(x_i=1)x_i) \oplus !x_i = f_1 \oplus !x_i;$$

$$f = (f(x_i=0)!x_i \vee !f(x_i=1)x_i) \oplus x_i = f_2 \oplus x_i.$$

Из этих соотношений следует, что $f_1 = f \oplus !x_i$, $f_2 = f \oplus x_i$.

Известно также, что $f_3 = f = f \oplus 0$; $f_4 = !f = f \oplus 1$.

Таким образом, модуль с порождающей формулой

$$F = f \oplus z$$

при $z = \{0, 1\}$ может породить две функции, при $z = \{0, !x_i \text{ или } x_i, 1\}$ — три функции, при $z = \{0, !x_i, x_i, 1\}$ — четыре функции, а при $z = \{0, !x_1, x_1, \dots, !x_n, x_n, 1\}$ — 2^{n+1} функций.

Если не все заданные БФУ можно объединить в ПФО с помощью предложенного соотношения, то они могут быть разбиты на группы, для каждой из которых это соотношение можно использовать. После этого полученные формулы могут быть объединены в порождающую формулу модуля одним из изложенных методов.

Пример 7.9. Построить модуль УЛМЗ, применяя рассмотренный метод.

Выберем формулу $f = x_1 x_2 x_3$. Построим ПФО вида:

$$F_9(3) = z \oplus f = z \oplus x_1 x_2 x_3.$$

При $z=0$ $F_9(3) = x_1 x_2 x_3$, при $z=!x_1$ $F_9(3) = !x_1 \vee x_2 x_3$, при $z=x_1$ $F_9(3) = x_1 (!x_2 \vee !x_3)$, при $z=1$ $F_9(3) = !x_1 \vee !x_2 \vee !x_3$.

Таким образом, схема (рис. 7.18) с четырьмя входами, состоящая из двух логических элементов, является искомым модулем.

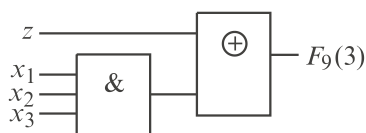


Рис. 7.18

При этом отметим, что построенный модуль [27] имеет три информационных и один настроечный вход, в то время как аналогичный модуль (рис. 7.12) имеет два информационных и два информационно-настроечных входа при одинаковых алфавитах настройки: $0, !x_1, x_1, 1$.

7.4.2. Использование NPN-классификации

Этот метод известен [13, 14] и приводится для общности.

Разобьем, исходя из принципа двойственности, заданное множество БФУ на два подмножества (группы). Построим одним из известных методов порождающую функцию для БФУ первой группы. Тогда инверсия ПФ позволяет получить булевы функции, однотипные с функциями второй группы. При этом соответствующий модуль может иметь либо два выхода (прямой и инверсный), либо один выход, соединенный с двухвходовым элементом НЕРАВНОЗНАЧНОСТЬ, один из входов которого подключен к схеме, реализующей ПФ, а на второй его вход могут подаваться константы 0 и 1.

Пример 7.10. Построить модуль УЛМЗ, применяя предлагаемый подход.

Сформируем на основе принципа двойственности две группы формул:

$$\{f_1 = x_1 x_2 x_3, \quad f_3 = x_1 \vee x_2 x_3\} \text{ и } \\ \{f_2 = x_1 (x_2 \vee x_3), \quad f_4 = x_1 \vee x_2 \vee x_3\}.$$

Построим методом «покрывающих деревьев» порождающую формулу для первой группы формул: $F = x_1 \# x_2 x_3 \# z_1$, где $z_1 = \{0, 1\}$. Тогда искомая порождающая формула может быть записана следующим образом:

$$F_{10}(3) = F \oplus z_2 = (x_1 \# x_2 x_3 \# z_1) \oplus z_2.$$

Эта порождающая формула реализуется схемой (рис. 7.19) с пятью входами, состоящей из трех логических элементов. При этом $z_i = \{0, 1\}$.

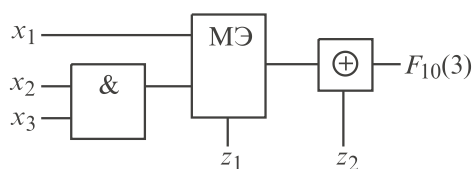


Рис. 7.19

Из приведенного примера следует, что использование этого метода в данном случае неэффективно.

В заключение главы отметим, что, применяя в методе «покрывающих деревьев» вместо трехвходовых мажоритарных элементов мультиплексоры «2 в 1», может быть построен однородный настраиваемый каскад Майтра [3] из двух ячеек (рис. 7.20), который явля-

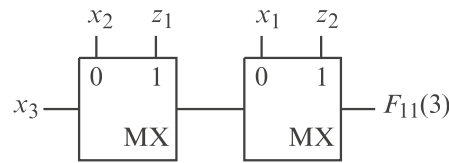


Рис. 7.20

ется модулем УЛМЗ как в рассмотренном базисе, так и в базисе $\{\&, \vee, \oplus, !\}$. В первом случае $z_i = \{0, 1\}$, а во втором — $z_1 = \{0, !x_2, 1\}$, $z_2 = \{0, !x_1, 1\}$.

В следующей главе рассмотренные выше методы применяются для построения более сложных модулей.

Выводы

1. Предложен мультиплексорный метод построения порождающих функций многофункциональных логических модулей из элементов с односторонней проводимостью (функциональных элементов) по заданному списку булевых функций n переменных. Этот метод является аналитическим и в отличие от известного также аналитического метода позволяет объединять в порождающую функцию не только сами БФУ, но и различные фрагменты их столбцов значений (например, «половинки»), что обычно позволяет сократить число переменных в порождающей функции, соответствующих входам модуля.

2. При фиксированном значении k предлагаемый метод позволяет объединять в порождающую функцию модуля различные фрагменты, являющиеся функциями $n - k$ переменных. Эти фрагменты выбираются из порождающей функции для образования заданной БФУ в общем случае с помощью функций k переменных, подаваемых на «нижние» входы мультиплексора, входящего в стандартные схемы модулей. При $k = 0$ на эти входы подаются константы 0 и 1, а при $k = 1$ кроме констант подается крайняя левая переменная таблицы истинности заданных БФУ, а возможно, и ее инверсия.

3. Предложено определять каждую настройку модуля на реализацию заданной БФУ с помощью решения логического уравнения в мультиплексорной форме.

4. Если в мультиплексорном методе при фиксированном k в порождающую функцию объединяются различные фрагменты, входящие во все заданные БФУ, то в его модификации для каждой БФУ определяются фрагменты и функции настройки на ее реализацию, а в порождающую функцию модуля мультиплексорно объединяются порождающие функции, построенные для фрагментов, расположенных на определенных позициях, и функций настройки для всех заданных БФУ.

5. Кроме универсального мультиплексорного метода и его модификации рассмотрены также дополнительные методы построения порождающих функций.

6. С помощью мультиплексорного метода построены порождающие функции большого числа модулей, которые защищены 26 авторскими свидетельствами СССР. Построение этих модулей описывается в гл. 8. Общее число авторских свидетельств, полученных автором на модули из функциональных элементов, равно 31.

Л и т е р а т у р а

1. Yau S. S., Tang C. K. Universal logic modules and their applications // IEEE Trans. on Computers. 1970. N 2.
2. Preparata F. P., Muller D. E. Generation of near-optimal universal boolean functions // J. Comp. and System Sciences. 1970. N 2.
3. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Настраиваемые модули для управляющих логических устройств. Л.: Энергоиздат, 1981.
4. Дуленов Е. Г. Многофункциональные симметрические схемы // Автоматика и телемеханика. 1970. № 5.
5. Cohen S., Winder R. O. Threshold gate building blocks // IEEE Trans. on Computers. 1969. N 9.
6. Попов Ю. А., Воронин А. Т., Сладков А. Б. Вопросы проектирования схем с высоким уровнем интеграции на основе КНС-технологии // Дискретные системы. Т. 1. Симпозиум IFAC. Рига: Зинатне, 1974.
7. Якубайтис Э. А. Логические автоматы и микромодули. Рига: Зинатне, 1975.
8. Якубайтис Э. А. Теория автоматов. Многофункциональные логические модули // Итоги науки и техники: Теория вероятностей. Математическая статистика. Теоретическая кибернетика. М.: ВИНТИ, 1976. № 13.
9. Малев В. А. Структурная избыточность в логических устройствах. М.: Связь, 1978.
10. Мищенко В. А., Козюминский В. Д., Семашко А. Н. Многофункциональные автоматы и элементная база цифровых ЭВМ. М.: Радио и связь, 1981.
11. Пуньрев Е. И. Перестраиваемые автоматы и микропроцессорные системы. М.: Наука, 1984.
12. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Судовые управляющие логические системы. Унифицированные логические схемы. Л.: ИПК СП, 1981.
13. Варшавский В. И., Песчанский В. А., Мараховский В. Б. Многофункциональные логические модули, реализующие все функции трех и четырех переменных // Тез. докл. II Всесоюз. совещ. по теории релейных устройств и конечных автоматов. Рига, 1971.
14. Stone H. E. Universal logic modules // Recent developments in switching theory / Ed. A. Makhopadhyay. New York: Acad. Press, 1971.
15. Стародубцев Н. А. Соотношения для числа настроек многофункциональных логических модулей // Изв. АН СССР. Техн. кибернетика. 1972. № 4.
16. Викентьев Л. Ф., Аляев Ю. А., Шалыто А. А. и др. Многофункциональный логический модуль. А. с. СССР № 1361539 // Бюл. изобр. 1987. № 47.
17. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Вопросы выбора и применения многофункциональных логических модулей // Дискретные системы. Т. 1. Симпозиум IFAC. Рига: Зинатне, 1974.
18. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. и др. Многофункциональный логический модуль. А. с. СССР № 813787 // Бюл. изобр. 1981. № 10.
19. Артюхов В. Л., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1070693 // Бюл. изобр. 1984. № 4.

20. *Шалыто А. А.* Многофункциональный логический модуль. А. с. СССР № 1283744 // Бюл. изобр. 1987. № 2.
21. *Микросхемы гибридные сложные* — микросборки «Пакет». Руководство по применению. Отраслевой стандарт. ОСТ 5.8354—74.
22. *Артюхов В. Л., Шалыто А. А.* Судовые управляющие логические системы. Л.: ИПК СП, 1983.
23. *Артюхов В. Л., Вольский В. Е., Шалыто А. А.* Логический модуль. А. с. СССР № 798806 // Бюл. изобр. 1981. № 3.
24. *Юдицкий С. А., Тагаевская А. А., Ефремова Т. К.* и др. Агрегатное построение пневматических систем управления. М.: Энергия, 1973.
25. *Вольский В. Е., Пушин Ю. Н., Юнг В. Н.* Проектирование пневматических систем управления судовыми энергетическими установками. Л.: Судостроение, 1975.
26. *Артюхов В. Л., Шалыто А. А.* Судовые управляющие логические системы. Л.: ИПК СП, 1984.
27. *Артюхов В. Л., Фишман Л. М., Шалыто А. А.* Многофункциональный логический модуль. А. с. СССР № 1096636 // Бюл. изобр. 1984. № 21.

Глава 8

Построение многофункциональных логических модулей

8.1. Модули, универсальные в классе дизъюнктивных нормальных форм из q букв

Используя PN -классификацию, построим одновыходные модули, универсальные в рассматриваемом классе формул из q букв ($q = 4, 5, 6$ букв).

1. $q = 4$.

В качестве представителей типов выберем следующие ДНФ:

$$\begin{aligned} f_1 &= x_1 x_2 x_3 x_4 = 0 \& x_1 \vee x_2 x_3 x_4 x_1; \\ f_2 &= x_1 \vee x_2 x_3 x_4 = x_2 x_3 x_4 !x_1 \vee 1 \& x_1; \\ f_3 &= x_1 x_2 \vee x_3 x_4 = x_3 x_4 !x_1 \vee (x_2 \vee x_3 x_4) x_1; \\ f_4 &= x_1 \vee x_2 \vee x_3 x_4 = (x_2 \vee x_3 x_4) !x_1 \vee 1 \& x_1; \\ f_5 &= x_1 \vee x_2 \vee x_3 \vee x_4 = (x_2 \vee x_3 \vee x_4) !x_1 \vee 1 \& x_1. \end{aligned}$$

1.1. Мультиплексорный метод при $k = 0$ строит одновыходной модуль с $4 + \lceil \log 5 \rceil = 7$ входами.

1.2. Применим этот метод при $k = 1$. При этом разнотипные фрагменты имеют вид: $G_0 = 0$, $G_1 = x_2 x_3 x_4$, $G_2 = 1$, $G_3 = x_3 x_4$, $G_4 = x_2 \vee x_3 x_4$, $G_5 = x_2 \vee x_3 \vee x_4$. Так как $R = 6$, то в этом случае можно построить модуль с $3 + \lceil \log 6 \rceil = 6$ входами. Ввиду того что $R < 2^m$, то при назначении функций Q_i некоторые из них могут быть одинаковыми, что может позволить упростить порождающую формулу модуля.

Назначим функции Q_i следующим образом: $Q_0 = 0$, $Q_1 = x_2 x_3 x_4$, $Q_2 = x_2 x_3 x_4$, $Q_3 = x_2 x_3 x_4$, $Q_4 = x_3 x_4$, $Q_5 = x_2 \vee x_3 x_4$, $Q_6 = x_2 \vee x_3 \vee x_4$, $Q_7 = 1$. При этом ПФО модуля имеет вид:

$$F_d(4) = \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3).$$

В качестве примера определим настройку модуля на реализацию формулы f_3 :

$$\begin{aligned} f_3 &= F_d(4); \\ \text{MX}(Q_4, Q_5; x_1) &= \\ &= \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3); \\ z_1 &= \text{MX}(1, 1; x_1) = 1; \\ z_2 &= \text{MX}(0, 0; x_1) = 0; \\ z_3 &= \text{MX}(0, 1; x_1) = x_1. \end{aligned}$$

Составим функции:

$$\begin{aligned} Q_{01} &= Q_0!z_3 \vee Q_1z_3 = z_3x_2x_3x_4; \\ Q_{11} &= Q_2!z_3 \vee Q_3z_3 = x_2x_3x_4; \\ Q_{21} &= Q_4!z_3 \vee Q_5z_3 = z_3x_2 \vee x_3x_4; \\ Q_{31} &= Q_6!z_3 \vee 1 \& z_3 = z_3 \vee x_2 \vee x_3 \vee x_4. \end{aligned}$$

Так как для этих функций выполняются условия покрытия, то

$$\begin{aligned} F_d(4) &= Q_{01} \vee z_2Q_{11} \vee z_1Q_{21} \vee z_1z_2Q_{31} = \\ &= z_1z_2(z_3 \vee x_2 \vee x_3 \vee x_4) \vee z_1(z_3x_2 \vee x_3x_4) \vee (z_2 \vee z_3)x_2x_3x_4. \end{aligned}$$

Построенный одновыходной модуль имеет шесть входов, из которых три являются информационными, а три — информационно-настроечными. Модуль описан в [1], где показано, что $z_i = \{0, x_1, 1\}$.

2. $q = 5$.

В качестве представителей типов выберем следующие ДНФ:

$$\begin{aligned} f_1 &= x_1x_2x_3x_4x_5 = 0 \& !x_1 \vee x_2x_3x_4x_5x_1; \\ f_2 &= x_1 \vee x_2x_3x_4x_5 = x_2x_3x_4x_5!x_1 \vee 1 \& x_1; \\ f_3 &= x_1x_2 \vee x_3x_4x_5 = x_3x_4x_5!x_1 \vee (x_2 \vee x_3x_4x_5)x_1; \\ f_4 &= x_1 \vee x_2 \vee x_3x_4x_5 = (x_2 \vee x_3x_4x_5)!x_1 \vee 1 \& x_1; \\ f_5 &= x_1 \vee x_2x_3 \vee x_4x_5 = (x_2x_3 \vee x_4x_5)!x_1 \vee 1 \& x_1; \\ f_6 &= x_1 \vee x_2 \vee x_3 \vee x_4x_5 = (x_2 \vee x_3 \vee x_4x_5)!x_1 \vee 1 \& x_1; \\ f_7 &= x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 = (x_2 \vee x_3 \vee x_4 \vee x_5)!x_1 \vee 1 \& x_1. \end{aligned}$$

2.1. Используя метод «наложения», можно построить простой одновыходной модуль с 10 входами:

$$F_{d1}(5) = z_1z_2z_3z_4z_5 \vee z_6z_7 \vee z_8 \vee z_9 \vee z_{10}.$$

Этот модуль описан в [2].

2.2. Мультиплексорный метод при $k = 0$ строит одновыходной модуль с $5 + \lceil \log 7 \rceil = 8$ входами.

2.3. Применим этот метод при $k = 1$. При этом разнотипные фрагменты имеют вид: $G_0=0$, $G_1=x_2x_3x_4x_5$, $G_2=1$, $G_3=x_3x_4x_5$, $G_4=x_2 \vee x_3x_4x_5$, $G_5=x_2x_3 \vee x_4x_5$, $G_6=x_2 \vee x_3 \vee x_4x_5$, $G_7=x_2 \vee x_3 \vee x_4 \vee x_5$. Так как $R = 8$, то в этом случае можно построить модуль с $4 + \lceil \log 8 \rceil = 7$ входами.

Назначим функции Q_i следующим образом: $Q_0=0$, $Q_1=x_2x_3x_4x_5$, $Q_2=x_3x_4x_5$, $Q_3=x_2 \vee x_3x_4x_5$, $Q_4=x_2x_3 \vee x_4x_5$, $Q_5=x_2 \vee x_3 \vee x_4x_5$, $Q_6=x_2 \vee x_3 \vee x_4 \vee x_5$, $Q_7=1$. При этом порождающая формула модуля имеет вид:

$$F_{\text{д2}}(5) = \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3).$$

В качестве примера определим настройку модуля на реализацию формулы f_3 :

$$\begin{aligned} f_3 &= F_{\text{д2}}(5); \\ \text{MX}(Q_2, Q_3; x_1) &= \\ &= \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3); \\ z_1 &= \text{MX}(0, 0; x_1) = 0; \\ z_2 &= \text{MX}(1, 1; x_1) = 1; \\ z_3 &= \text{MX}(0, 1; x_1) = x_1. \end{aligned}$$

Так как для этих функций выполняются условия покрытия, то

$$\begin{aligned} F_{\text{д2}}(5) &= Q_{01} \vee z_3 Q_1 \vee z_2 Q_2 \vee z_2 z_3 Q_3 \vee z_1 Q_4 \vee \\ &\vee z_1 z_3 Q_5 \vee z_1 z_2 Q_6 \vee z_1 z_2 z_3 Q_7. \end{aligned}$$

Построенный одновыходной модуль имеет семь входов, из которых четыре являются информационными, а три — информационно-настроечными. Модуль описан в [3], где показано, что $z_i = \{0, x_1, 1\}$.

3. $q = 6$.

В качестве представителей типов выберем следующие ДНФ:

$$\begin{aligned} f_1 &= x_1x_2x_3x_4x_5x_6 = 0 \& !x_1 \vee x_2x_3x_4x_5x_6x_1; \\ f_2 &= x_1 \vee x_2x_3x_4x_5x_6 = x_2x_3x_4x_5x_6!x_1 \vee 1 \& x_1; \\ f_3 &= x_1x_2 \vee x_3x_4x_5x_6 = x_3x_4x_5x_6!x_1 \vee (x_2 \vee x_3x_4x_5x_6)x_1; \\ f_4 &= x_1 \vee x_2 \vee x_3x_4x_5x_6 = (x_2 \vee x_3x_4x_5x_6)!x_1 \vee 1 \& x_1; \\ f_5 &= x_1x_2x_3 \vee x_4x_5x_6 = x_4x_5x_6!x_1 \vee (x_2x_3 \vee x_4x_5x_6)x_1; \\ f_6 &= x_1 \vee x_2x_3 \vee x_4x_5x_6 = (x_2x_3 \vee x_4x_5x_6)!x_1 \vee 1 \& x_1; \end{aligned}$$

$$\begin{aligned}
f_7 &= x_1 \vee x_2 \vee x_3 \vee x_4 x_5 x_6 = (x_2 \vee x_3 \vee x_4 x_5 x_6)!x_1 \vee 1 \& x_1; \\
f_8 &= x_1 x_2 \vee x_3 x_4 \vee x_5 x_6 = (x_3 x_4 \vee x_5 x_6)!x_1 \vee \\
&\quad \vee (x_2 \vee x_3 x_4 \vee x_5 x_6)x_1; \\
f_9 &= x_1 \vee x_2 \vee x_3 x_4 \vee x_5 x_6 = (x_2 \vee x_3 x_4 \vee x_5 x_6)!x_1 \vee 1 \& x_1; \\
f_{10} &= x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 x_6 = (x_2 \vee x_3 \vee x_4 \vee x_5 x_6)!x_1 \vee 1 \& x_1; \\
f_{11} &= x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6 = (x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6)!x_1 \vee \\
&\quad \vee 1 \& x_1.
\end{aligned}$$

3.1. Мультиплексорный метод при $k = 0$ строит одновыходной модуль с $6 + \lceil \log 11 \rceil = 10$ входами.

3.2. Используем этот метод при $k = 1$. При этом разнотипные фрагменты имеют вид: $G_0 = 0$, $G_1 = x_2 x_3 x_4 x_5 x_6$, $G_2 = 1$, $G_3 = x_3 x_4 x_5 x_6$, $G_4 = x_2 \vee x_3 x_4 x_5 x_6$, $G_5 = x_4 x_5 x_6$, $G_6 = x_2 x_3 \vee x_4 x_5 x_6$, $G_7 = x_2 \vee x_3 \vee x_4 x_5 x_6$, $G_8 = x_3 x_4 \vee x_5 x_6$, $G_9 = x_2 \vee x_3 x_4 \vee x_5 x_6$, $G_{10} = x_2 \vee x_3 \vee x_4 \vee x_5 x_6$, $G_{11} = x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6$. Так как $R = 12$, то в этом случае можно построить модуль с $5 + \lceil \log 12 \rceil = 9$ входами.

Назначим функции Q_i следующим образом: $Q_4 = 0$, $Q_5 = x_2 x_3 x_4 x_5 x_6$, $Q_6 = x_3 x_4 x_5 x_6$, $Q_7 = x_2 \vee x_3 x_4 x_5 x_6$, $Q_8 = x_4 x_5 x_6$, $Q_9 = x_2 x_3 \vee x_4 x_5 x_6$, $Q_{10} = x_3 x_4 \vee x_5 x_6$, $Q_{11} = x_2 \vee x_3 x_4 \vee x_5 x_6$, $Q_{12} = x_2 \vee x_3 \vee x_4 x_5 x_6$, $Q_{13} = x_2 \vee x_3 \vee x_4 \vee x_5 x_6$, $Q_{14} = x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6$, $Q_{15} = 1$.

Если функции $Q_0 \dots Q_3$ выбрать так, чтобы выполнялись отношения покрытия, то ПФО модуля может быть записана в виде:

$$F_d(6) = MX(Q_0, Q_1, \dots, Q_{15}; z_1, z_2, z_3, z_4).$$

В качестве примера определим настройку модуля на реализацию формулы f_7 :

$$\begin{aligned}
f_7 &= F_d(6); \\
MX(Q_{12}, Q_{15}; x_1) &= MX(Q_0, Q_1, \dots, Q_{15}; z_1, z_2, z_3, z_4); \\
z_1 &= MX(1, 1; x_1) = 1; \\
z_2 &= MX(1, 1; x_1) = 1; \\
z_3 &= MX(0, 1; x_1) = x_1; \\
z_4 &= MX(0, 1; x_1) = x_1.
\end{aligned}$$

Построенный одновыходной модуль имеет девять входов, из которых пять являются информационными, а четыре — информационно-настроечными. Модуль описан в [4], где показано, что $z_i = \{0, x_1, 1\}$.

8.2. Модули, универсальные в классе произвольных формул в базисе И, ИЛИ, НЕ из q букв

1. $q = 2$. Используется PN -классификация [5].

Выберем в качестве представителей типов следующие формулы:

$$f_1 = x_1 x_2, \quad f_2 = x_1 \vee x_2.$$

1.1. Применим мультиплексор «2 в 1» в качестве модуля УЛМ1(2) (рис. 8.1).

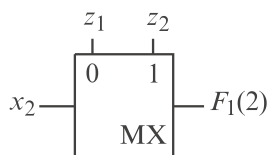


Рис. 8.1

Этот модуль описывается соотношением

$$F_1(2) = \text{MX}(z_1, z_2; x_2)$$

и при $z_1 = 0, z_2 = x_1$ реализует формулу $F_1(2) = x_1 x_2$, а при $z_1 = x_1, z_2 = 1$ — формулу $F_1(2) = x_1 \vee x_2$.

1.2. Мультиплексорный метод при $k = 0$ строит одновыходной модуль с $2 + \lceil \log 2 \rceil = 3$ входами, который описывается порождающей формулой вида:

$$F_2(2) = \text{MX}(f_1, f_2; z)$$

и при $z = 0$ реализует формулу $F_2(2) = x_1 x_2$, а при $z = 1$ — формулу $F_2(2) = x_1 \vee x_2$.

Так как $f_1 \leq f_2$, то $F_2(2) = x_1 \# x_2 \# z$ (рис. 8.2).

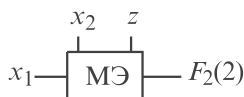


Рис. 8.2

1.3. Используем этот метод при $k = 1$. При этом разнотипные фрагменты имеют вид: $G_0 = 0, G_1 = x_2, G_2 = 1$. Так как $R = 3$, то в этом случае можно построить модуль с $1 + \lceil \log 3 \rceil = 3$ входами.

Назначим функции Q_i следующим образом: $Q_0 = Q_1 = 0, Q_2 = x_2, Q_3 = 1$. При этом

$$F_3(2) = \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2) = z_1 (x_2 \vee z_2).$$

При $Q_0=0, Q_1=Q_2=x_2, Q_3=1$

$$F_4(2) = MX(Q_0, Q_1, Q_2, Q_3; z_1, z_2) = z_1 \# z_2 \# x_2.$$

При $Q_0=0, Q_1=x_2, Q_2=Q_3=1$

$$F_5(2) = MX(Q_0, Q_1, Q_2, Q_3; z_1, z_2) = z_1 \vee x_2 z_2.$$

1.4. Применим метод, описанный в разд. 7.4.1. Выберем формулу $f = x_1 x_2$. Тогда

$$F_6(2) = x_1 x_2 \oplus z.$$

При $z=0$ $F_6(2) = x_1 x_2$, а при $z=1$ $F_6(2) = !x_1 \vee !x_2$.

2. $q = 3$.

2.1. Модули УЛМЗ, использующие *PN*-классификацию, рассмотрены в гл. 7.

2.2. Построим модуль УЛМЗ, использующий *P*-классификацию. Выберем в качестве представителей типов следующие формулы:

$$\begin{aligned} f_1 &= x_1 x_2 x_3 = 0 \& !x_1 \vee x_2 x_3 x_1; \\ f_2 &= !x_1 x_2 x_3 = x_2 x_3 !x_1 \vee 0 \& x_1; \\ f_3 &= !x_1 x_2 !x_3 = x_2 !x_3 !x_1 \vee 0 \& x_1; \\ f_4 &= !x_1 !x_2 !x_3 = !x_2 !x_3 !x_1 \vee 0 \& x_1; \\ f_5 &= x_1 (x_2 \vee x_3) = 0 \& !x_1 \vee (x_2 \vee x_3) x_1; \\ f_6 &= x_1 (x_2 \vee !x_3) = 0 \& !x_1 \vee (x_2 \vee !x_3) x_1; \\ f_7 &= !x_1 (x_2 \vee x_3) = (x_2 \vee x_3) !x_1 \vee 0 \& x_1; \\ f_8 &= x_1 (!x_2 \vee !x_3) = 0 \& !x_1 \vee (!x_2 \vee !x_3) x_1; \\ f_9 &= !x_1 (x_2 \vee !x_3) = (x_2 \vee !x_3) !x_1 \vee 0 \& x_1; \\ f_{10} &= !x_1 (!x_2 \vee !x_3) = (!x_2 \vee !x_3) !x_1 \vee 0 \& x_1; \\ f_{11} &= x_1 \vee x_2 x_3 = x_2 x_3 !x_1 \vee 1 \& x_1; \\ f_{12} &= x_1 \vee x_2 !x_3 = x_2 !x_3 !x_1 \vee 1 \& x_1; \\ f_{13} &= !x_1 \vee x_2 x_3 = 1 \& !x_1 \vee x_2 x_3 x_1; \\ f_{14} &= x_1 \vee !x_2 !x_3 = !x_2 !x_3 !x_1 \vee 1 \& x_1; \\ f_{15} &= !x_1 \vee x_2 !x_3 = 1 \& !x_1 \vee x_2 !x_3 x_1; \\ f_{16} &= !x_1 \vee !x_2 !x_3 = 1 \& !x_1 \vee !x_2 !x_3 x_1; \\ f_{17} &= x_1 \vee x_2 \vee x_3 = (x_2 \vee x_3) !x_1 \vee 1 \& x_1; \\ f_{18} &= !x_1 \vee x_2 \vee x_3 = 1 \& !x_1 \vee (x_2 \vee x_3) x_1; \\ f_{19} &= !x_1 \vee x_2 \vee !x_3 = 1 \& !x_1 \vee (x_2 \vee !x_3) x_1; \\ f_{20} &= !x_1 \vee !x_2 \vee !x_3 = 1 \& !x_1 \vee (!x_2 \vee !x_3) x_1. \end{aligned}$$

2.2.1. Мультиплексорный метод при $k = 0$ строит одновыходной модуль с $3 + \lceil \log 20 \rceil = 8$ входами.

2.2.2. Применим этот метод при $k = 1$. При этом разнотипные фрагменты имеют вид: $G_0 = 0$, $G_1 = x_2 x_3$, $G_2 = x_2 !x_3$, $G_3 = !x_2 !x_3$, $G_4 = x_2 \vee x_3$, $G_5 = x_2 \vee !x_3$, $G_6 = !x_2 \vee !x_3$, $G_7 = 1$. Так как $R = 8$, то в этом случае можно построить модуль с $2 + \lceil \log 8 \rceil = 5$ входами.

Назначим функции Q_i следующим образом: $Q_i = G_i$. При этом $z_i = \{0, !x_1, x_1, 1\}$. Для того чтобы упростить алфавит настройки ($z_i = \{0, x_1, 1\}$) число информационно-настроечных входов увеличим до четырех. Порождающую формулу модуля будем строить в виде:

$$F_P(3) = MX(Q_0, Q_1, \dots, Q_{15}; z_1, z_2, z_3, z_4),$$

где $Q_0 = 1$, $Q_1 = x_2 x_3$, $Q_2 = x_2 !x_3$, $Q_3 = !x_2 !x_3$, $Q_4 = x_2 \vee x_3$, $Q_5 = x_2 \vee !x_3$, $Q_6 = !x_2 \vee !x_3$, $Q_7 = 0$, $Q_8 = 0$, $Q_9 = !x_2 \vee !x_3$, $Q_{10} = x_2 \vee !x_3$, $Q_{11} = x_2 \vee x_3$, $Q_{12} = !x_2 !x_3$, $Q_{13} = x_2 !x_3$, $Q_{14} = x_2 x_3$, $Q_{15} = 1$.

В качестве примера определим настройку модуля на реализацию формулы f_{15} :

$$\begin{aligned} f_{15} &= F_P(3); \\ MX(Q_0, Q_2; x_1) &= MX(Q_0, Q_1, \dots, Q_{15}; z_1, z_2, z_3, z_4); \\ z_1 &= MX(0, 0; x_1) = 0; \\ z_2 &= MX(0, 0; x_1) = 0; \\ z_3 &= MX(0, 1; x_1) = x_1; \\ z_4 &= MX(0, 0; x_1) = 0. \end{aligned}$$

Другая настройка может быть получена, если учесть, что функция f_{15} может быть записана в МФ иначе:

$$\begin{aligned} f_{15} &= F_P(3); \\ MX(Q_0, Q_{13}; x_1) &= MX(Q_0, Q_1, \dots, Q_{15}; z_1, z_2, z_3, z_4); \\ z_1 &= MX(0, 1; x_1) = x_1; \\ z_2 &= MX(0, 1; x_1) = x_1; \\ z_3 &= MX(0, 0; x_1) = 0; \\ z_4 &= MX(0, 1; x_1) = x_1. \end{aligned}$$

Построенный одновыходной модуль имеет шесть входов, из которых два являются информационными, а четыре — информационно-настроечными. Модуль описан в [6].

Так как порождающая функция модуля $F_P(3)$ является «зеркальной» (разд. 10.7), то она может быть записана иначе:

$$F_P(3) = MX(Q_0, Q_1, \dots, Q_7; z_1 \oplus z_2, z_1 \oplus z_3, z_1 \oplus z_4).$$

2.3. Построим модуль, использующий N -классификацию. Выберем в качестве представителей типов следующие формулы:

$$\begin{aligned} f_0 &= x_1 x_2 \vee x_3, & f_2 &= x_1 \vee x_2 x_3, \\ f_4 &= x_1 x_3 \vee x_2, & f_1 &= (x_1 \vee x_3) x_2, & f_3 &= (x_1 \vee x_2) x_3, \\ f_5 &= x_1 (x_2 \vee x_3), & f_6 &= x_1 x_2 x_3, & f_7 &= x_1 \vee x_2 \vee x_3. \end{aligned}$$

Мультиплексорный метод при $k = 0$ строит модуль с $3 + \lceil \log 8 \rceil = 6$ входами (рис. 8.3).

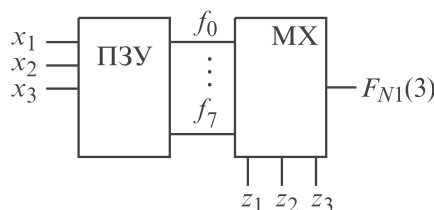


Рис. 8.3

Построим модуль, в котором постоянное запоминающее устройство (ПЗУ) заменено весьма простой схемой.

В [7] была предложена комбинационная схема (КСх1) с обратной связью, состоящая из двухвходовых элементов И и ИЛИ и реализующая функции f_0 — f_5 (рис. 8.4).

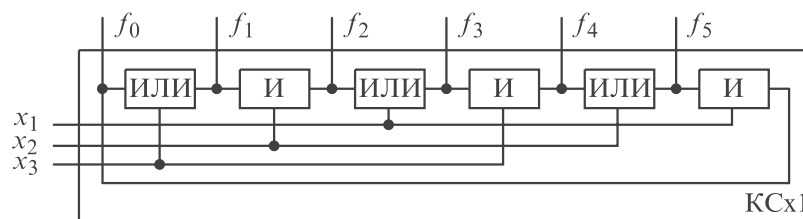


Рис. 8.4

Из изложенного следует, что искомым модулем является схема, приведенная на рис. 8.5. Этот модуль описан в [8].

2.4. Построим модуль, порождающая функция которого объединяет все 64 неповторные БФ в рассматриваемом базисе из трех букв.

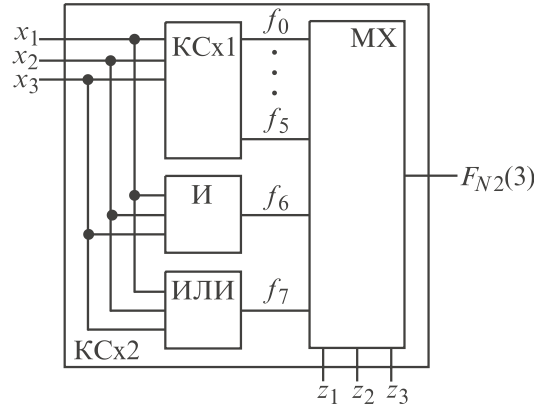


Рис. 8.5

Исходя из изложенного в предыдущем пункте, следует, что схема на рис. 8.6 является искомым модулем. Этот модуль описан в [9].

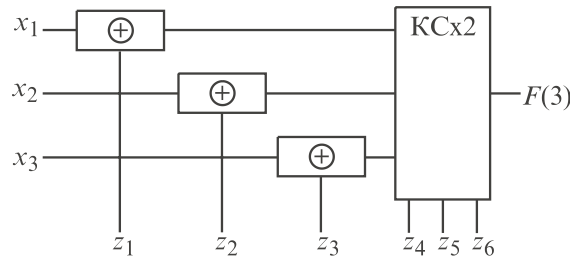


Рис. 8.6

3. $q = 4$.

В качестве представителей PN -типов неповторных пороговых формул выберем формулы, записанные в порядке убывания весов, и перечислим представителей всех типов неповторных формул, которые разложим по одной и той же переменной, обладающей для пороговых формул наибольшим весом, что обеспечивает минимальную номенклатуру остаточных формул:

$$\begin{aligned}
 f_1 &= x_1 x_2 x_3 x_4 = 0 \& !x_1 \vee x_2 x_3 x_4 x_1; \\
 f_2 &= x_1 x_2 (x_3 \vee x_4) = 0 \& !x_1 \vee x_2 (x_3 \vee x_4) x_1; \\
 f_3 &= x_1 (x_2 \vee x_3 x_4) = 0 \& !x_1 \vee (x_2 \vee x_3 x_4) x_1; \\
 f_4 &= x_1 (x_2 \vee x_3 \vee x_4) = 0 \& !x_1 \vee (x_2 \vee x_3 \vee x_4) x_1; \\
 f_5 &= x_1 \vee x_2 x_3 x_4 = x_2 x_3 x_4 !x_1 \vee 1 \& x_1; \\
 f_6 &= x_1 \vee x_2 (x_3 \vee x_4) = x_2 (x_3 \vee x_4) !x_1 \vee 1 \& !x_1;
 \end{aligned}$$

$$\begin{aligned}
f_7 &= x_1 \vee x_2 \vee x_3 x_4 = (x_2 \vee x_3 x_4)!x_1 \vee 1 \& x_1; \\
f_8 &= x_1 \vee x_2 \vee x_3 \vee x_4 = (x_2 \vee x_3 \vee x_4)!x_1 \vee 1 \& x_1; \\
f_9 &= x_1 x_2 \vee x_3 x_4 = x_3 x_4!x_1 \vee (x_2 \vee x_3 x_4)x_1; \\
f_{10} &= (x_1 \vee x_2)(x_3 \vee x_4) = x_2(x_3 \vee x_4)!x_1 \vee (x_3 \vee x_4)x_1.
\end{aligned}$$

3.1. Мультиплексорный метод при $k = 0$ строит одновыходной модуль с $4 + \lceil \log 10 \rceil = 8$ входами.

3.2. Применим метод построения порождающих функций модулей, описанный в разд. 7.4.1. Для этого построим три выражения:

$$\begin{aligned}
F_1 &= (x_1 \vee x_2 \vee x_3 \vee x_4) \oplus z = f_1 \oplus z; \\
F_2 &= (x_1 x_2 \vee x_3 \vee x_4) \oplus z = f_2 \oplus z; \\
F_3 &= (x_1 x_2 \vee x_3 x_4) \oplus z = f_3 \oplus z.
\end{aligned}$$

Первое из этих выражений порождает следующие формулы:

$$\begin{aligned}
F_1 &= x_1 \vee x_2 \vee x_3 \vee x_4 && \text{при } z = 0; \\
F_1 &= !x_1!x_2!x_3!x_4 && \text{при } z = 1; \\
F_1 &= !x_1!x_2!x_3 \vee x_4 && \text{при } z = !x_4; \\
F_1 &= (x_1 \vee x_2 \vee x_3)!x_4 && \text{при } z = x_4.
\end{aligned}$$

Второе выражение порождает формулы:

$$\begin{aligned}
F_2 &= x_1 x_2 \vee x_3 \vee x_4 && \text{при } z = 0; \\
F_2 &= (!x_1 \vee !x_2)!x_3!x_4 && \text{при } z = 1; \\
F_2 &= (!x_1 \vee !x_2)!x_3 \vee x_4 && \text{при } z = !x_4; \\
F_2 &= (x_1 x_2 \vee x_3)x_4 && \text{при } z = x_4.
\end{aligned}$$

Третье выражение порождает формулы:

$$\begin{aligned}
F_3 &= x_1 x_2 \vee x_3 x_4 && \text{при } z = 0; \\
F_3 &= (!x_1 \vee !x_2)(!x_3 \vee !x_4) && \text{при } z = 1.
\end{aligned}$$

Объединим формулы f_1 , f_2 и f_3 методом «наложения»:

$$\begin{aligned}
f &= x_1 z_1 \vee z_2 z_3 \vee z_4 \vee z_5. \\
\text{При } z_1 &= 1, \quad z_2 = x_2, \quad z_3 = 1, \quad z_4 = x_3, \quad z_5 = x_4 && \text{— } f = f_1; \\
\text{при } z_1 &= x_2, \quad z_2 = 0, \quad z_3 = 0, \quad z_4 = x_3, \quad z_5 = x_4 && \text{— } f = f_2; \\
\text{при } z_1 &= x_2, \quad z_2 = x_3, \quad z_3 = x_4, \quad z_4 = 0, \quad z_5 = 0 && \text{— } f = f_3.
\end{aligned}$$

Таким образом, одновыходной модуль с семью входами в данном случае описывается порождающей формулой вида:

$$F_1(4) = (x_1 z_1 \vee z_2 z_3 \vee z_4 \vee z_5) \oplus z_6.$$

3.3. При построении модуля используем *NPN*-классификацию. Выберем в качестве представителей типов — формулы f_4, f_6, f_7, f_8, f_9 . Применяя метод «наложения», получим формулу

$$F = z_1 \vee (z_2 \vee z_3) z_4 \vee z_5 z_6.$$

Модуль УЛМ4 с прямым и инверсным выходами, построенный на основе этой формулы, описан в [10], а одновыходной модуль с семью входами в данном случае имеет порождающую формулу вида:

$$F_2(4) = (z_1 \vee (z_2 \vee z_3) z_4 \vee z_5 z_6) \oplus z_7.$$

3.4. Используем этот метод при $k = 1$. При этом разнотипные фрагменты имеют вид: $G_0=0$, $G_1=x_2x_3x_4$, $G_2=x_2(x_3 \vee x_4)$, $G_3=x_2 \vee x_3x_4$, $G_4=x_2 \vee x_3 \vee x_4$, $G_5=1$, $G_6=x_3x_4$, $G_7=x_3 \vee x_4$. Так как $R=8$, то в этом случае можно построить модуль с $3 + \lceil \log 8 \rceil = 6$ входами.

Реализуем искомый модуль в виде стандартной схемы (рис. 8.7).

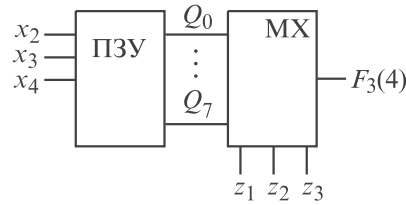


Рис. 8.7

Этой схеме соответствует соотношение

$$F_3(4) = \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3).$$

Для упрощения построения порождающей формулы преобразуем это соотношение к виду:

$$F_3(4) = \text{MX}(Q_{01}, Q_{11}, Q_{21}, Q_{31}; z_1, z_2).$$

Этому выражению соответствует схема на рис. 8.8.

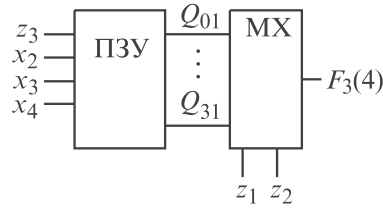


Рис. 8.8

Если назначить функции Q_i следующим образом: $Q_0=0$, $Q_1=x_2x_3x_4$, $Q_6=x_2\vee x_3\vee x_4$, $Q_7=1$, то $Q_{01}=!z_3Q_0\vee z_3Q_1=z_3x_2x_3x_4$, $Q_{31}=!z_3Q_6\vee z_3Q_7=z_3\vee x_2\vee x_3\vee x_4$. При этом существуют 4! варианта назначений функций Q_2, Q_3, Q_4, Q_5 , при которых удовлетворяются отношения покрытия. Назначим эти функции следующим образом: $Q_2=x_2(x_3\vee x_4)$, $Q_3=x_3\vee x_4$, $Q_4=x_3x_4$, $Q_5=x_2\vee x_3x_4$. При этом $Q_{11}=!z_3Q_2\vee z_3Q_3=(z_3\vee x_2)(x_3\vee x_4)$, $Q_{21}=!z_3Q_4\vee z_3Q_5=z_3x_2\vee x_3x_4$. Следовательно,

$$F_3(4) = z_1z_2(z_3\vee x_2\vee x_3\vee x_4)\vee z_1(z_3x_2\vee x_3x_4)\vee \\ \vee z_2(z_3\vee x_2)(x_3\vee x_4)\vee z_3x_2x_3x_4.$$

Эта порождающая функция состоит из 20 букв, а после вынесения за скобки переменной z_1 их количество уменьшается до 19. Для одновыходного модуля этого типа с тремя информационными и тремя информационно-настроечными входами в дальнейшем будем применять обозначение, приведенное на рис. 8.9.

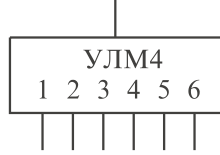


Рис. 8.9

В качестве примера определим настройку модуля на реализацию формулы f_{10} :

$$\begin{aligned} f_{10} &= F_3(4); \\ &= \text{MX}(Q_2, Q_3; x_1) = \\ &= \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3); \\ z_1 &= \text{MX}(0, 0; x_1) = 0; \\ z_2 &= \text{MX}(1, 1; x_1) = 1; \\ z_3 &= \text{MX}(0, 1; x_1) = x_1. \end{aligned}$$

Этот модуль при $z_1=x_1$, $z_2=x_1$, $z_3=!x_1$ реализует формулу $f = x_1(x_2\vee x_3\vee x_4)\vee x_2x_3x_4$, которая является порождающей формулой модуля УЛМ3. Рассмотренный модуль описан в [11], где показано, что при реализации неповторных формул $z_i = \{0, x_1, 1\}$.

3.5. Используя метод построения ПФ модулей, описанный в разд. 7.2.5, сформируем на основе принципа двойственности две группы фрагментов длиной восемь: $\{G_0=0, G_1=x_2x_3x_4, G_3=x_2\vee x_3x_4, G_4=x_3x_4\}$, $\{G_2=x_2(x_3\vee x_4), G_4=x_2\vee x_3\vee x_4, G_5=1,$

$G_7 = x_3 \vee x_4 \}$. В каждой из них находятся все фрагменты одной из двух непороговых формул.

Применим в качестве основных фрагменты второй группы. Назначим функции Q_i так, чтобы выполнялись отношения покрытия:

$$Q_0 = x_2(x_3 \vee x_4), Q_1 = x_3 \vee x_4, Q_2 = x_2 \vee x_3 \vee x_4, Q_3 = 1.$$

При этом

$$\begin{aligned} F = MX(Q_0, Q_1, Q_2, Q_3; z_2, z_3) &= Q_0 \vee Q_1 z_3 \vee Q_2 z_2 \vee Q_3 z_2 z_3 = \\ &= (z_2 \vee z_3 \vee x_2)(x_3 \vee x_4) \vee z_2(z_3 \vee x_2). \end{aligned}$$

Таким образом,

$$\begin{aligned} F_4(4) &= z_1 \oplus F = \\ &= MX(Q_0, Q_1, Q_2, Q_3, !Q_0, !Q_1, !Q_2, !Q_3; z_1, z_2, z_3) = \\ &= MX(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3) = \\ &= z_1 \oplus [(z_2 \vee z_3 \vee x_2)(x_3 \vee x_4) \vee z_2(z_3 \vee x_2)]. \end{aligned}$$

Если использовать в качестве основных фрагменты первой группы, то $Q_0 = 0, Q_1 = x_2 x_3 x_4, Q_2 = x_3 x_4, Q_3 = x_2 \vee x_3 x_4$. При этом

$$\begin{aligned} F_1 &= MX(Q_0, Q_1, Q_2, Q_3; z_2, z_3) = \\ &= Q_0 \vee Q_1 z_3 \vee Q_2 z_2 \vee Q_3 z_2 z_3 = (z_2 \vee z_2 x_2) x_3 x_4 \vee z_2 z_3 x_2. \end{aligned}$$

Таким образом,

$$\begin{aligned} F_5(4) &= z_1 \oplus F_1 = \\ &= MX(Q_0, Q_1, Q_2, Q_3, !Q_0, !Q_1, !Q_2, !Q_3; z_1, z_2, z_3) = \\ &= MX(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3) = \\ &= z_1 \oplus [(z_2 \vee z_3 x_2) x_3 x_4 \vee z_2 z_3 x_2]. \end{aligned}$$

Найденные порождающие формулы зависят от шести переменных и содержат по девять букв, что более чем в два раза меньше, чем число букв в порождающей формуле $F_3(4)$, также зависящей от шести переменных. Это упрощение достигнуто за счет расширения алфавита настройки — $\{0, !x_1, x_1, 1\}$ вместо $\{0, x_1, 1\}$.

Определим настройку модуля с порождающей формулой $F_5(4)$ на функцию, PN -однотипную с функцией $f_{10} = (x_1 \vee x_2)(x_3 \vee x_4) = !x_1 x_2 (x_3 \vee x_4) \vee x_1 (x_3 \vee x_4)$.

Заменим в этом разложении каждую из остаточных функций антидвойственной: $f_{101} = !x_1 !x_2 (!x_3 \vee !x_4) \vee x_1 (!x_3 \vee !x_4) = !x_1 !(x_2 \vee x_3 x_4) \vee x_1 !(x_3 \vee x_4) = !x_1 !Q_3 \vee x_1 !Q_2 = !x_1 Q_7 \vee x_1 Q_6 = (x_1 \vee !x_2) \& \& (!x_3 \vee !x_4)$.

При этом

$$\begin{aligned}
 f_{101} &= F_5(4); \\
 \text{MX}(Q_7, Q_6; x_1) &= \\
 &= \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3); \\
 z_1 &= \text{MX}(1, 1; x_1) = 1; \\
 z_2 &= \text{MX}(1, 1; x_1) = 1; \\
 z_3 &= \text{MX}(1, 0; x_1) = !x_1.
 \end{aligned}$$

Этот модуль описан в [12], а метод его построения — в [13].

3.6. Схема из двух модулей УЛМЗ (рис. 7.10) является одноходовым модулем УЛМ4 с семью входами (рис. 8.10).

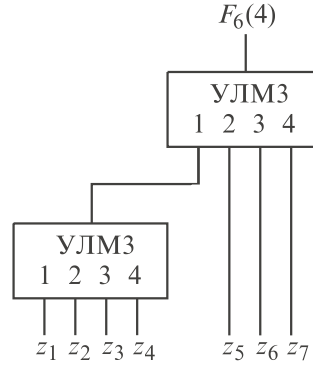


Рис. 8.10

Ниже в этом разделе будет показано, что этот модуль дополнительно реализует все непороговые формулы в базисе $\{\&, \vee, !\}$ из пяти букв. Этот модуль описан в [14].

4. $q = 5$.

$$\begin{aligned}
 f_1 &= x_1 x_2 x_3 x_4 x_5 = 0 \& !x_1 \vee x_2 x_3 x_4 x_5 x_1; \\
 f_2 &= x_1 x_2 x_3 (x_4 \vee x_5) = 0 \& !x_1 \vee x_2 x_3 (x_4 \vee x_5) x_1; \\
 f_3 &= x_1 x_2 (x_3 \vee x_4 x_5) = 0 \& !x_1 \vee x_2 (x_3 \vee x_4 x_5) x_1; \\
 f_4 &= x_1 x_2 (x_3 \vee x_4 \vee x_5) = 0 \& !x_1 \vee x_2 (x_3 \vee x_4 \vee x_5) x_1; \\
 f_5 &= x_1 (x_2 \vee x_3 x_4 x_5) = 0 \& !x_1 \vee (x_2 \vee x_3 x_4 x_5) x_1; \\
 f_6 &= x_1 (x_2 \vee x_3 (x_4 \vee x_5)) = 0 \& !x_1 \vee (x_2 \vee x_3 (x_4 \vee x_5)) x_1; \\
 f_7 &= x_1 (x_2 \vee x_3 \vee x_4 x_5) = 0 \& !x_1 \vee (x_2 \vee x_3 \vee x_4 x_5) x_1; \\
 f_8 &= x_1 (x_2 \vee x_3 \vee x_4 \vee x_5) = 0 \& !x_1 \vee (x_2 \vee x_3 \vee x_4 \vee x_5) x_1; \\
 f_9 &= x_1 \vee x_2 x_3 x_4 x_5 = x_2 x_3 x_4 x_5 !x_1 \vee 1 \& x_1;
 \end{aligned}$$

$$\begin{aligned}
f_{10} &= x_1 \vee x_2 x_3 (x_4 \vee x_5) = x_2 x_3 (x_4 \vee x_5)!x_1 \vee 1 \& x_1; \\
f_{11} &= x_1 \vee x_2 (x_3 \vee x_4 x_5) = x_2 (x_3 \vee x_4 x_5)!x_1 \vee 1 \& x_1; \\
f_{12} &= x_1 \vee x_2 (x_3 \vee x_4 \vee x_5) = x_2 (x_3 \vee x_4 \vee x_5)!x_1 \vee 1 \& x_1; \\
f_{13} &= x_1 \vee x_2 \vee x_3 x_4 x_5 = (x_2 \vee x_3 x_4 x_5)!x_1 \vee 1 \& x_1; \\
f_{14} &= x_1 \vee x_2 \vee x_3 (x_4 \vee x_5) = (x_2 \vee x_3 (x_4 \vee x_5))!x_1 \vee 1 \& x_1; \\
f_{15} &= x_1 \vee x_2 \vee x_3 \vee x_4 x_5 = (x_2 \vee x_3 \vee x_4 x_5)!x_1 \vee 1 \& x_1; \\
f_{16} &= x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 = (x_2 \vee x_3 \vee x_4 \vee x_5)!x_1 \vee 1 \& x_1; \\
f_{17} &= x_1 \vee x_2 x_3 \vee x_4 x_5 = (x_2 x_3 \vee x_4 x_5)!x_1 \vee 1 \& x_1; \\
f_{18} &= x_1 \vee (x_2 \vee x_3) (x_4 \vee x_5) = (x_2 \vee x_3) (x_4 \vee x_5)!x_1 \vee 1 \& x_1; \\
f_{19} &= x_1 (x_2 x_3 \vee x_4 x_5) = 0 \& !x_1 \vee (x_2 x_3 \vee x_4 x_5) x_1; \\
f_{20} &= x_1 (x_2 \vee x_3) (x_4 \vee x_5) = 0 \& !x_1 \vee (x_2 \vee x_3) (x_4 \vee x_5) x_1; \\
f_{21} &= x_1 x_2 \vee x_3 x_4 x_5 = x_3 x_4 x_5!x_1 \vee (x_2 \vee x_3 x_4 x_5) x_1; \\
f_{22} &= x_1 x_2 \vee x_3 (x_4 \vee x_5) = x_3 (x_4 \vee x_5)!x_1 \vee \\
&\quad \vee (x_2 \vee x_3 (x_4 \vee x_5)) x_1; \\
f_{23} &= (x_1 \vee x_2) (x_3 \vee x_4 x_5) = x_2 (x_3 \vee x_4 x_5)!x_1 \vee (x_3 \vee x_4 x_5) x_1; \\
f_{24} &= (x_1 \vee x_2) (x_3 \vee x_4 \vee x_5) = x_2 (x_3 \vee x_4 \vee x_5)!x_1 \vee \\
&\quad \vee (x_3 \vee x_4 \vee x_5) x_1.
\end{aligned}$$

4.1. Применяя метод «наложения» применительно к заданным БФ, можно построить одновыходной модуль УЛМ5 с 10 входами (рис. 8.11). Этот модуль описан в [12].

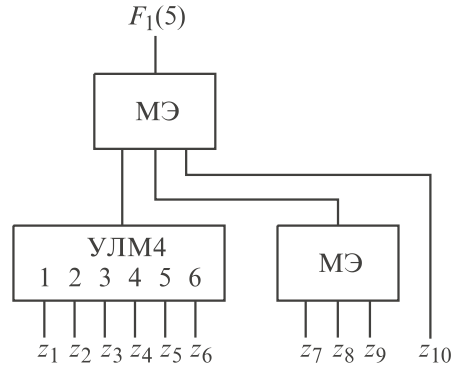


Рис. 8.11

4.2. Более простой модуль был предложен в [15]. При этом использовались метод «наложения» применительно к древовидным схемам в базисе $\{\&, \vee\}$, реализующим заданные БФ, и *NP*N-класси-

фикация. Этот одновыходной модуль имеет порождающую формулу вида:

$$F_2(5) = [(z_1 z_2 z_3 \vee z_4 z_5) z_6 z_7 \vee z_8 z_9] \oplus z_{10},$$

где $z_{10} = \{0, 1\}$.

4.3. Мультиплексорный метод при $k = 0$ строит одновыходной модуль с $5 + \lceil \log 24 \rceil = 10$ входами.

4.4. Применим этот метод при $k = 1$. При этом разнотипные фрагменты имеют вид: $G_0 = 0$, $G_1 = x_2 x_3 x_4 x_5$, $G_2 = x_2 x_3 (x_4 \vee x_5)$, $G_3 = x_2 (x_3 \vee x_4 x_5)$, $G_4 = x_2 (x_3 \vee x_4 \vee x_5)$, $G_5 = x_2 \vee x_3 x_4 x_5$, $G_6 = x_2 \vee x_3 (x_4 \vee x_5)$, $G_7 = x_2 \vee x_3 \vee x_4 x_5$, $G_8 = x_2 \vee x_3 \vee x_4 \vee x_5$, $G_9 = 1$, $G_{10} = x_2 x_3 \vee x_4 x_5$, $G_{11} = (x_2 \vee x_3) (x_4 \vee x_5)$, $G_{12} = x_3 x_4 x_5$, $G_{13} = x_3 (x_4 \vee x_5)$, $G_{14} = x_3 \vee x_4 x_5$, $G_{15} = x_3 \vee x_4 \vee x_5$. Так как $R = 16$, то в этом случае можно построить модуль с $4 + \lceil \log 16 \rceil = 6$ входами. Реализуем искомый модуль в виде стандартной схемы (рис. 8.12).

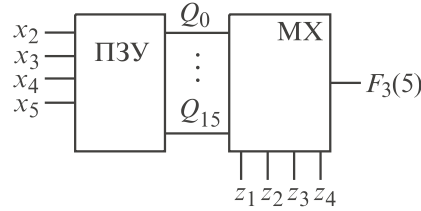


Рис. 8.12

Этой схеме соответствует соотношение

$$F_3(5) = \text{MX}(Q_0, Q_1, \dots, Q_{15}; z_1, z_2, z_3, z_4).$$

Для упрощения построения порождающей формулы преобразуем это соотношение к виду:

$$F_3(5) = \text{MX}(Q_{01}, Q_{11}, \dots, Q_{71}; z_1, z_2, z_3).$$

Этому выражению соответствует схема на рис. 8.13.

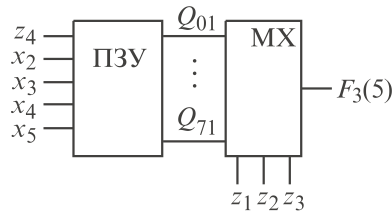


Рис. 8.13

Назначим функции Q_i и Q_{i1} следующим образом:

$$\begin{aligned}
Q_0 &= 0, & Q_1 &= x_2 x_3 x_4 x_5, & Q_{01} &= z_4 x_2 x_3 x_4 x_5; \\
Q_2 &= x_3 x_4 x_5, & Q_3 &= x_2 \vee x_3 x_4 x_5, & Q_{11} &= z_4 x_2 \vee x_3 x_4 x_5; \\
Q_4 &= x_2 x_3 (x_4 \vee x_5), & Q_5 &= x_3 (x_4 \vee x_5), & Q_{21} &= (z_4 \vee \\
& & & & & \vee x_2) x_3 (x_4 \vee x_5); \\
Q_6 &= (x_2 \vee x_3) (x_4 \vee & Q_7 &= x_2 \vee x_3 (x_4 \vee & Q_{31} &= z_4 x_2 \vee (x_2 \vee \\
& \vee x_5), & \vee x_5), & \vee x_3) (x_4 \vee x_5); \\
Q_8 &= x_2 (x_3 \vee x_4 x_5), & Q_9 &= x_3 \vee x_4 x_5, & Q_{41} &= (z_4 \vee x_2) (x_3 \vee \\
& & & & & \vee x_4 x_5); \\
Q_{10} &= x_2 x_3 \vee x_4 x_5, & Q_{11} &= x_2 \vee x_3 \vee x_4 x_5, & Q_{51} &= z_4 \# x_2 \# x_3 \vee \\
& & & & & \vee x_4 x_5; \\
Q_{12} &= x_2 (x_3 \vee x_4 \vee x_5), & Q_{13} &= x_3 \vee x_4 \vee x_5, & Q_{61} &= (z_4 \vee x_2) (x_3 \vee \\
& & & & & \vee x_4 \vee x_5); \\
Q_{14} &= x_2 \vee x_3 \vee x_4 \vee x_5, & Q_{15} &= 1, & Q_{71} &= z_4 \vee x_2 \vee x_3 \vee \\
& & & & & \vee x_4 \vee x_5.
\end{aligned}$$

Так как для функций Q_{i1} выполняются отношения покрытия, то

$$\begin{aligned}
F_3(5) &= Q_{01} \vee Q_{11} z_3 \vee Q_{21} z_2 \vee Q_{31} z_2 z_3 \vee Q_{41} z_1 \vee \\
& \vee Q_{51} z_1 z_3 \vee Q_{61} z_1 z_2 \vee Q_{71} z_1 z_2 z_3.
\end{aligned}$$

Если в эту формулу подставить выражения для функций Q_{i1} , приведенные выше, то она будет содержать 55 букв. В [5] показано, что эта формула может быть существенно упрощена. При этом булева формула

$$\begin{aligned}
F_3(5) &= (z_1 z_3 \vee z_4 x_2 x_3) x_4 x_5 \vee \{ [z_1 (z_4 \vee x_2 \vee x_3) \vee (z_1 \vee x_2 \vee \\
& \vee x_3) (x_4 \vee x_5)] z_2 \vee z_4 x_2 \vee x_3 x_4 x_5 \} z_3 \vee [z_1 (x_3 \vee x_4 x_5) \vee \\
& \vee (z_1 \vee x_3) (x_4 \vee x_5) z_2] (z_4 \vee x_2)
\end{aligned}$$

содержит 34 буквы. В качестве примера определим настройку модуля на реализацию формулы f_{18} :

$$\begin{aligned}
f_{18} &= F_3(5); \\
MX(Q_6, Q_{15}; x_1) &= MX(Q_0, Q_1, \dots, Q_{15}; z_1, z_2, z_3, z_4); \\
z_1 &= MX(0, 1; x_1) = x_1; \\
z_2 &= MX(1, 1; x_1) = 1; \\
z_3 &= MX(1, 1; x_1) = 1; \\
z_4 &= MX(0, 1; x_1) = x_1.
\end{aligned}$$

Этот одновыходной модуль с восемью входами описан в [16], где показано, что $z_i = \{0, x_1, 1\}$.

4.5. Используем метод, изложенный в разд. 7.2.5. При этом на основе принципа двойственности разобьем приведенные выше фрагменты на две группы. В первую группу включим следующие фрагменты, входящие в неповторные пороговые формулы: $G_0=0$, $G_1=x_2x_3x_4x_5$, $G_2=x_2x_3(x_4 \vee x_5)$, $G_5=x_2 \vee x_3x_4x_5$, $G_6=x_2 \vee x_3(x_4 \vee x_5)$. Из пары непороговых формул с постоянной остаточной функцией включим в первую группу фрагмент $G_{11}=(x_2 \vee x_3)(x_4 \vee x_5)$. Для оставшихся непороговых формул в первую группу включим фрагмент $G_{12}=x_3x_4x_5$, так как в этой группе уже имеется фрагмент $G_5(f_{21}=!x_1G_{12} \vee x_1G_5)$, и фрагмент $G_{13}=x_3(x_4 \vee x_5)$, так как в рассматриваемой группе имеется фрагмент $G_6(f_{22}=!x_1G_{13} \vee x_1G_6)$.

Так как $R=8$, то в данном случае может быть построен одновыходной модуль УЛМ5 с $5 + \lceil \log 8 \rceil = 8$ входами.

Назначим функции Q_i : $Q_0=0$, $Q_1=x_2x_3x_4x_5$, $Q_2=x_3x_4x_5$, $Q_3=x_2 \vee x_3x_4x_5$, $Q_4=x_2x_3(x_4 \vee x_5)$, $Q_5=x_3(x_4 \vee x_5)$, $Q_6=(x_2 \vee x_3)(x_4 \vee x_5)$, $Q_7=x_2 \vee x_3(x_4 \vee x_5)$. Объединим эти функции следующим образом:

$$F_2 = \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_2, z_3, z_4),$$

и построим ПФО модуля:

$$F_4(5) = z_1 \oplus F_2 = \text{MX}(Q_0, Q_1, \dots, Q_{15}; z_1, z_2, z_3, z_4),$$

где $Q_8=!Q_0$, $Q_9=!Q_1$, $Q_{10}=!Q_2$, $Q_{11}=!Q_3$, $Q_{12}=!Q_4$, $Q_{13}=!Q_5$, $Q_{14}=!Q_6$, $Q_{15}=!Q_7$.

Так как для функций $Q_0, \dots, Q_7$ выполняются отношения покрытия, то

$$\begin{aligned} F_4(5) &= z_1 \oplus (Q_0 \vee Q_1 z_4 \vee Q_2 z_3 \vee Q_3 z_3 z_4 \vee Q_4 z_2 \vee Q_5 z_2 z_4 \vee \\ &\vee Q_6 z_2 z_3 \vee Q_7 z_2 z_3 z_4) = z_1 \oplus \{z_2[z_3(x_2 \vee x_3) \vee (z_4 \vee \\ &\vee x_2)x_3](x_4 \vee x_5) \vee z_3 z_4 x_2 \vee (z_3 \vee z_4 x_2)x_3 x_4 x_5\}. \end{aligned}$$

Из изложенного следует, что функция $F_4(5)$ зависит от восьми переменных и реализуется булевой формулой из 19 букв, что почти в два раза меньше, чем число букв в формуле $F_3(5)$, построенной выше. Упрощение модуля достигается за счет расширения алфавита настройки, ($\{0, !x_1, x_1, 1\}$ вместо $\{0, x_1, 1\}$), и того, что некоторые БФ, реализуемые путем настройки, содержат инверсии.

Настройка этого модуля выполняется следующим образом.

Пусть требуется реализовать формулу

$$f_9 = x_1 \vee x_2 x_3 x_4 x_5 = x_2 x_3 x_4 x_5 !x_1 \vee 1 \& x_1 = Q_1 !x_1 \vee Q_8 x_1.$$

Составим и решим уравнение:

$$\begin{aligned} f_9 &= F_4(5); \\ \text{MX}(Q_1, Q_8; x_1) &= \text{MX}(Q_0, Q_1, \dots, Q_{15}; z_1, z_2, z_3, z_4); \\ z_1 &= \text{MX}(0, 1; x_1) = x_1; \\ z_2 &= \text{MX}(0, 0; x_1) = 0; \\ z_3 &= \text{MX}(0, 0; x_1) = 0; \\ z_4 &= \text{MX}(1, 0; x_1) = !x_1. \end{aligned}$$

Пусть требуется реализовать формулу

$$f = (x_1 \vee !x_2) (!x_3 \vee !x_4 \vee !x_5).$$

Она *PN*-однотипна с формулой $f_{24} = (x_1 \vee x_2) (x_3 \vee x_4 \vee x_5) = x_2 (x_3 \vee x_4 \vee x_5) !x_1 \vee (x_3 \vee x_4 \vee x_5) x_1$.

Запишем формулу

$$\begin{aligned} f &= !x_2 (!x_3 \vee !x_4 \vee !x_5) !x_1 \vee (!x_3 \vee !x_4 \vee !x_5) x_1 = \\ &= !(x_2 \vee x_3 x_4 x_5) !x_1 \vee (x_3 x_4 x_5) x_1 = \\ &= !Q_3 !x_1 \vee !Q_2 x_1 = Q_{11} !x_1 \vee Q_{10} x_1. \end{aligned}$$

Составим и решим уравнение:

$$\begin{aligned} f &= F_4(5); \\ \text{MX}(Q_{11}, Q_{10}; x_1) &= \text{MX}(Q_0, Q_1, \dots, Q_{15}; z_1, z_2, z_3, z_4); \\ z_1 &= \text{MX}(1, 1; x_1) = 1; \\ z_2 &= \text{MX}(0, 0; x_1) = 0; \\ z_3 &= \text{MX}(1, 1; x_1) = 1; \\ z_4 &= \text{MX}(1, 0; x_1) = !x_1. \end{aligned}$$

Этот модуль описан в [17].

4.6. Применим мультиплексорный метод при $k = 2$. При этом разложим по Шеннону каждый из приведенных выше 24 представителей *PN*-типов БФ по переменным x_1, x_2 и определим получающиеся при этом фрагменты, являющиеся «четвертинками». Например, для формулы f_{18} это разложение имеет вид:

$$\begin{aligned} f_{18} &= x_1 \vee (x_2 \vee x_3) (x_4 \vee x_5) = \\ &= \text{MX}(x_3 (x_4 \vee x_5), x_4 \vee x_5, 1, 1; x_1, x_2). \end{aligned}$$

Эта формула содержит следующие разнотипные фрагменты: $x_3 (x_4 \vee x_5), x_4 \vee x_5, 1$. Выпишем разнотипные фрагменты, которые

входят в состав всех формул: $G_0=0$, $G_1=x_3x_4x_5$, $G_2=x_3(x_4 \vee x_5)$, $G_3=x_4 \vee x_5$, $G_4=x_4x_5$, $G_5=x_3 \vee x_4x_5$, $G_6=x_3 \vee x_4 \vee x_5$, $G_7=1$.

Назначим функции Q_i следующим образом: $Q_i=G_i$. Объединим эти функции:

$$\begin{aligned} F_5(5) &= MX(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; \Phi_1, \Phi_2, \Phi_3) = \\ &= \Phi_1 \Phi_2 (\Phi_3 \vee x_3 \vee x_4 \vee x_5) \vee \Phi_1 (\Phi_3 x_3 \vee x_4 x_5) \vee \\ &\vee \Phi_2 (\Phi_3 \vee x_3) (x_4 \vee x_5) \vee \Phi_3 x_3 x_4 x_5. \end{aligned}$$

Эта БФ является порождающей формулой одного из модулей УЛМ4. Для t -й заданной БФ составим и решим уравнение

$$f_t = MX(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; \Phi_1^t, \Phi_2^t, \Phi_3^t).$$

Например, при $t = 18$:

$$\begin{aligned} &MX(Q_2, Q_3, Q_7, Q_7; x_1, x_2) = \\ &= MX(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; \Phi_1^{18}, \Phi_2^{18}, \Phi_3^{18}); \\ &\Phi_1^{18} = MX(0, 0, 1, 1; x_1, x_2) = x_1; \\ &\Phi_2^{18} = MX(1, 1, 1, 1; x_1, x_2) = 1; \\ &\Phi_3^{18} = MX(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2. \end{aligned}$$

В [13] показано, что функции $\Phi_j^t = \{0, x_1, x_1x_2, x_1 \vee x_2, 1\}$. Поэтому каждая функция Φ_j может быть реализована, например, трехвходовым мажоритарным элементом. Специфика функций Φ_j^t в данном случае такова [13], что по одному из входов все три МЭ могут быть объединены. Исходя из изложенного, в данном случае может быть построен одновыходной модуль УЛМ5 с десятью входами, обладающий блочной структурой (рис. 8.14).

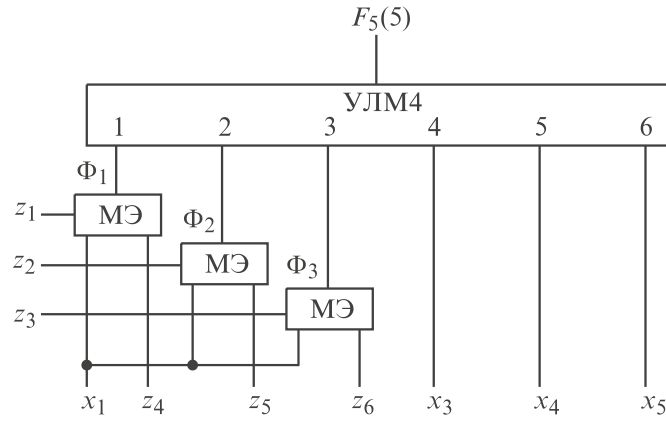


Рис. 8.14

Этот модуль при $z_1=0, z_2=1, z_3=1, z_4=1, z_5=1, z_6=x_2$ реализует $\Phi_1=x_1, \Phi_2=1, \Phi_3=x_1 \vee x_2$, и поэтому $F_5(5)=f_{18}$.

4.7. Используем мультиплексорный метод при $k=3$. При этом разложим по Шеннону каждый из приведенных выше 24 представителей PN -типов БФ по переменным x_1, x_2, x_3 и определим получающиеся при этом фрагменты, являющиеся «восьмушками». Например, для формулы f_{18} это разложение имеет вид:

$$\begin{aligned} f_{18} &= x_1 \vee (x_2 \vee x_3)(x_4 \vee x_5) = \\ &= \text{MX}(0, x_4 \vee x_5, x_4 \vee x_5, x_4 \vee x_5, 1, 1, 1; x_1, x_2, x_3). \end{aligned}$$

Эта формула содержит следующие разнотипные фрагменты: $0, x_4 \vee x_5, 1$. Выпишем разнотипные фрагменты, которые входят в состав всех формул: $G_0=0, G_1=x_4 x_5, G_2=x_4 \vee x_5, G_3=1$.

Назначим функции Q_i следующим образом: $Q_i=G_i$. Объединим эти функции:

$$\begin{aligned} F_6(5) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1, \Phi_2) = \\ &= \Phi_1(\Phi_2 \vee x_4 \vee x_5) \vee \Phi_2 x_4 x_5. \end{aligned}$$

Эта БФ является порождающей формулой одного из модулей УЛМЗ. Для t -й заданной БФ составим и решим уравнение

$$f_t = \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1^t, \Phi_2^t).$$

Например, при $t=18$:

$$\begin{aligned} &\text{MX}(Q_0, Q_2, Q_2, Q_2, Q_3, Q_3, Q_3, Q_3; x_1, x_2, x_3) = \\ &= \text{MX}(Q_0, Q_1, Q_2, Q_3; \Phi_1^{18}, \Phi_2^{18}); \\ \Phi_1^{18} &= \text{MX}(0, 1, 1, 1, 1, 1, 1; x_1, x_2) = x_1 \vee x_2 \vee x_3; \\ \Phi_2^{18} &= \text{MX}(0, 0, 0, 0, 1, 1, 1; x_1, x_2) = x_1. \end{aligned}$$

В [13] показано, что функции $\Phi_j^t = \{0, x_1, x_1 x_2, x_1 \vee x_2, x_1 x_2 x_3, x_1(x_2 \vee x_3), x_1 \vee x_2 x_3, x_1 \vee x_2 \vee x_3, 1\}$. Поэтому каждая функция Φ_j^t также может быть реализована модулем УЛМЗ. Специфика этих функций в данном случае [13] такова, что первые два одноименных входа двух модулей УЛМЗ могут быть объединены. Исходя из изложенного, может быть построен одновыходной модуль УЛМ5 с восемью входами, обладающий блочной структурой (рис. 8.15).

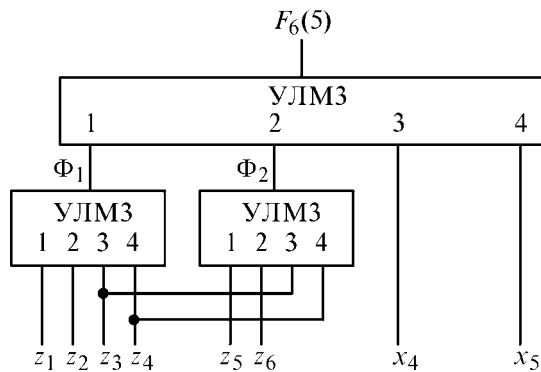


Рис. 8.15

Этот модуль при $z_1=1$, $z_2=x_1$, $z_3=x_2$, $z_4=x_3$, $z_5=x_1$, $z_6=x_1$ реализует $\Phi_1=x_1 \vee x_2 \vee x_3$, $\Phi_2=x_1$, и поэтому $F_6(5)=f_{18}$.

Построенный модуль является простейшим из известных одно-выходных модулей УЛМ5, так как содержит восемь входов и состоит из трех модулей УЛМ3.

Этот модуль описан в [18], где показано, что $z_i = \{0, x_1, x_2, x_3, 1\}$.

4.8. Построим еще один модуль, состоящий из трех модулей УЛМ3, который так же, как и предыдущий модуль УЛМ5, содержит девять логических выводов.

В разд. 3.3.3 показано, что произвольная неповторная БФ в рассматриваемом базисе из пяти букв может быть реализована схемой из двух модулей УЛМ3. При этом все 24 представителя PN -типов этих формул могут быть разбиты на два класса: непороговые (формулы $f_{17}—f_{24}$) и пороговые (формулы $f_1—f_{16}$). В [19] показано, что схема (рис. 8.16) является модулем УЛМ5.

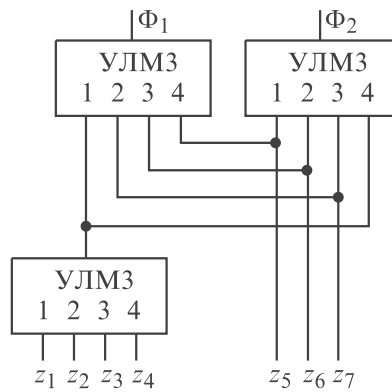


Рис. 8.16

В этом модуле на выходе Φ_1 путем настройки может быть реализована произвольная непороговая БФ рассматриваемого класса, а на выходе Φ_2 — произвольная пороговая БФ рассматриваемого класса. Этот модуль описан в [19].

4.9. Так как произвольная БФ в базисе $\{\&, \vee, !\}$ из пяти букв может быть реализована схемой из двух модулей УЛМ3, то такие два модуля с суммарным числом логических выводов равным 10 могут рассматриваться в качестве модуля УЛМ5.

5. $q = 6$.

5.1. В [15], применяя метод наложения древовидных схем, реализующих каждого из 33 представителей *NPN*-типов бесповторных БФ в базисе $\{\&, \vee, !\}$ из шести букв, построена ПФО одновыходного модуля УЛМ6 с 14 входами:

$$F_1(6) = \{[(z_1 z_2 z_3 \vee z_4 z_5 z_6) z_7 z_8 \vee z_9 z_{10}] z_{11} \vee z_{12} z_{13}\} \oplus z_{14},$$

где $z_{14} = \{0, 1\}$.

5.2. Мультиплексорный метод при $k = 0$ строит одновыходной модуль УЛМ6 с $6 + \lceil \log 66 \rceil = 13$ входами.

5.3. Из табл. 7.1 следует, что, используя мультиплексорный метод при $k = 1$, может быть построен одновыходной модуль УЛМ6 с 11 входами.

5.4. Построение такого модуля не выполнялось, так как в [20] показано, что два модуля — УЛМ3 и УЛМ4 (рис. 8.17) с суммарным числом логических выводов 12, реализуют путем суперпозиций произвольную БФ в рассматриваемом базисе из шести букв.

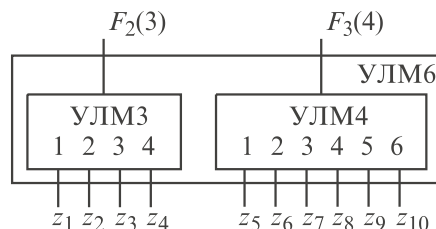


Рис. 8.17

Таким образом, схема на рис. 8.17 более эффективна, чем однокомпонентный модуль УЛМ6, так как при том же числе внешних выводов модули, входящие в эту схему, кроме совместного использования в указанных суперпозициях могут применяться также и независимо. Эта схема обладает наибольшей логической эффективностью среди всех известных логических схем с 12 логическими выводами. Схема описана в [20].

8.3. Модули, универсальные в классе произвольных формул в базисе И, ИЛИ, НЕРАВНОЗНАЧНОСТЬ, НЕ из q букв

При построении этих модулей используется PN -классификация.

1. $q = 3$.

Выберем в качестве представителей типов следующие формулы:

$$\begin{aligned} f_1 &= x_1 x_2 x_3, f_2 = x_1 (!x_2 \vee !x_3), f_3 = x_1 \vee x_2 x_3, \\ f_4 &= !x_1 \vee !x_2 \vee x_3, f_5 = x_1 \vee (!x_2 \oplus x_3), \\ f_6 &= \{x_1 \oplus x_2 x_3 \text{ или } x_1 \oplus (!x_2 \vee !x_3)\}, f_7 = x_1 \oplus x_2 \oplus x_3, \\ f_8 &= x_1 (x_2 \oplus x_3). \end{aligned}$$

1.1. Выберем в качестве порождающей следующую функцию пяти переменных:

$$\begin{aligned} F &= |0000 \ 0000 \ 0001 \ 1000 \ 1111 \ 1111 \ 1110 \ 0111|^T = \\ &= z_1 \oplus z_2 (z_3 \oplus z_4) (z_3 \oplus z_5). \end{aligned}$$

Применим переменные z_4 и z_5 в качестве информационных за счет следующих переобозначений: $z_4 = x_2, z_5 = x_3$. При этом

$$F_1 = z_1 \oplus z_2 (z_3 \oplus x_2) (z_3 \oplus x_3).$$

Составим и решим следующие уравнения:

$$\begin{aligned} f_1 &= F_1, \\ \text{MX}(Q_0, Q_2; x_1) &= \\ &= \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3), \\ z_1 &= \text{MX}(0, 0; x_1) = 0, \\ z_2 &= \text{MX}(0, 1; x_1) = x_1, \\ z_3 &= \text{MX}(0, 0; x_1) = 0; \end{aligned}$$

$$\begin{aligned} f_2 &= F_1, \\ \text{MX}(Q_0, Q_6; x_1) &= \\ &= \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3), \\ z_1 &= \text{MX}(0, 1; x_1) = x_1, \\ z_2 &= \text{MX}(0, 1; x_1) = x_1, \\ z_3 &= \text{MX}(0, 0; x_1) = 0; \end{aligned}$$

$$\begin{aligned}
f_3 &= F_1, \\
\text{MX}(Q_2, Q_4; x_1) &= \\
&= \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3), \\
z_1 &= \text{MX}(0, 1; x_1) = x_1, \\
z_2 &= \text{MX}(1, 0; x_1) = !x_1, \\
z_3 &= \text{MX}(0, 0; x_1) = 0;
\end{aligned}$$

$$\begin{aligned}
f_4 &= F_1, \\
\text{MX}(Q_4, Q_6; x_1) &= \\
&= \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3), \\
z_1 &= \text{MX}(1, 1; x_1) = 1, \\
z_2 &= \text{MX}(0, 1; x_1) = x_1, \\
z_3 &= \text{MX}(0, 0; x_1) = 0;
\end{aligned}$$

$$\begin{aligned}
f_5 &= F_1, \\
\text{MX}(Q_2, Q_7; x_1) &= \\
&= \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3), \\
z_1 &= \text{MX}(0, 1; x_1) = x_1, \\
z_2 &= \text{MX}(1, 1; x_1) = 1, \\
z_3 &= \text{MX}(0, 1; x_1) = x_1;
\end{aligned}$$

$$\begin{aligned}
x_1 \oplus x_2 x_3 &= F_1, \\
\text{MX}(Q_2, Q_6; x_1) &= \\
&= \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3), \\
z_1 &= \text{MX}(0, 1; x_1) = x_1, \\
z_2 &= \text{MX}(1, 1; x_1) = 1, \\
z_3 &= \text{MX}(0, 0; x_1) = 0;
\end{aligned}$$

$$\begin{aligned}
x_1 \oplus (!x_2 \oplus !x_3) &= F_1, \\
\text{MX}(Q_6, Q_2; x_1) &= \\
&= \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3), \\
z_1 &= \text{MX}(1, 0; x_1) = !x_1, \\
z_2 &= \text{MX}(1, 1; x_1) = 1, \\
z_3 &= \text{MX}(0, 0; x_1) = 0.
\end{aligned}$$

Используем в функции F в качестве информационной только переменную $z_5 = x_3$:

$$F_2 = z_1 \oplus z_2 (z_3 \oplus z_4) (z_3 \oplus x_3).$$

Составим и решим уравнения:

$$f_7 = F_2,$$

$$\text{MX}(Q_5, Q_6, Q_{13}, Q_{14}; x_1, x_2) = \text{MX}(Q_0, Q_1, \dots, Q_{15}; z_1, z_2, z_3, z_4),$$

$$z_1 = \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1,$$

$$z_2 = \text{MX}(1, 1, 1, 1; x_1, x_2) = 1,$$

$$z_3 = \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2,$$

$$z_4 = \text{MX}(1, 0, 1, 0; x_1, x_2) = !x_2;$$

$$f_8 = F_2,$$

$$\text{MX}(Q_1, Q_2, Q_5, Q_6; x_1, x_2) = \text{MX}(Q_0, Q_1, Q_2, \dots, Q_{15}; z_1, z_2, z_3, z_4),$$

$$z_1 = \text{MX}(0, 0, 0, 0; x_1, x_2) = 0,$$

$$z_2 = \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1,$$

$$z_3 = \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2,$$

$$z_4 = \text{MX}(1, 0, 1, 0; x_1, x_2) = !x_2.$$

Таким образом, в качестве порождающей формулы искомого одновыходного модуля можно применять формулу F_2 , в которой одна из переменных является информационной. Этот модуль описан в [21].

1.2. В гл. 7 показано, что модуль с порождающей формулой

$$F = z_1 z_2 z_3 \oplus z_4$$

реализует путем настройки первые четыре типа формул из трех букв. Покажем, что БФ

$$F = (z_0 \oplus z_1) z_2 z_3 \oplus z_4$$

является ПФО искомого модуля, так как он реализует также:

$$\text{при } z_3=1, z_4=1 \quad F = !z_2 \vee (!z_0 \oplus z_1);$$

$$\text{при } z_0=0, z_1=1 \quad F = z_2 z_3 \oplus z_4;$$

$$\text{при } z_2=1, z_3=1 \quad F = z_0 \oplus z_1 \oplus z_4;$$

$$\text{при } z_3=1, z_4=0 \quad F = (z_0 \oplus z_1) z_2.$$

Этот одновыходной модуль с пятью входами описан в [22].

2. $q = 4$.

В разд. 1.3 показано, что при этом значении q существуют 29 PN -типов формул.

2.1. Мультиплексорный метод при $k = 0$ строит одновыходной модуль с $4 + \lceil \log 29 \rceil = 9$ входами.

Такой модуль, построенный на основе модуля УЛМ4 (рис. 8.9), описан в [23].

2.2. Мультиплексорный метод при $k = 1$ строит одновыходной модуль с восемью входами. Этот модуль описан в [22].

8.4. Модуль, универсальный в классе произвольных булевых функций трех переменных

Запишем представителей всех 16 PN -типов функций, существенно зависящих от трех переменных — x_1, x_2, x_3 :

$$\begin{aligned}
 f_1 &= |0000 \ 0001|^T = 0 \& !x_1 \vee x_2 x_3 x_1 = x_1 x_2 x_3; \\
 f_2 &= |0000 \ 0111|^T = 0 \& !x_1 \vee (x_2 \vee x_3) x_1 = x_1 (x_2 \vee x_3); \\
 f_3 &= |0111 \ 1000|^T = (x_2 \vee x_3) !x_1 \vee !x_2 !x_3 x_1 = x_1 \oplus (x_2 \vee x_3); \\
 f_4 &= |0000 \ 0110|^T = 0 \& !x_1 \vee (x_2 \oplus x_3) x_1 = x_1 (x_2 \oplus x_3); \\
 f_5 &= |1001 \ 0110|^T = !(x_2 \oplus x_3) !x_1 \vee (x_2 \oplus x_3) x_1 = !(x_1 \oplus x_2 \oplus x_3); \\
 f_6 &= |0001 \ 0111|^T = x_2 x_3 !x_1 \vee (x_2 \vee x_3) x_1 = x_1 \# x_2 \# x_3; \\
 f_7 &= |0011 \ 0101|^T = x_2 !x_1 \vee x_3 x_1; \\
 f_8 &= |0001 \ 1000|^T = x_2 x_3 !x_1 \vee !x_2 !x_3 x_1 = (x_1 \oplus x_2) (x_1 \oplus x_3); \\
 f_9 &= |1001 \ 1000|^T = !(x_2 \oplus x_3) !x_1 \vee !x_2 !x_3 x_1 = !x_1 x_2 x_3 \vee !x_2 !x_3; \\
 f_{10} &= |1000 \ 0110|^T = !x_2 !x_3 !x_1 \vee (x_2 \oplus x_3) x_1; \\
 f_{11} &= |0111 \ 1111|^T = (x_2 \vee x_3) !x_1 \vee 1 \& x_1 = x_1 \vee x_2 \vee x_3; \\
 f_{12} &= |0001 \ 1111|^T = x_2 x_3 !x_1 \vee 1 \& x_1 = x_1 \vee x_2 x_3; \\
 f_{13} &= |0110 \ 1111|^T = (x_2 \oplus x_3) !x_1 \vee 1 \& x_1 = x_1 \vee (x_2 \oplus x_3); \\
 f_{14} &= |1110 \ 0111|^T = (!x_2 \vee !x_3) !x_1 \vee (x_2 \vee x_3) x_1 = \\
 &= !x_1 !x_2 \vee x_1 x_3 \vee x_2 !x_3 = !(x_1 \oplus x_2) \vee !(x_1 \oplus x_3); \\
 f_{15} &= |0111 \ 0110|^T = (x_2 \vee x_3) !x_1 \vee (x_2 \oplus x_3) x_1 = !x_1 x_2 \vee !x_2 x_3 \vee \\
 &\vee x_2 !x_3 = (!x_1 \vee x_2) !x_3 \vee !x_2 x_3 = !x_1 x_2 \vee (x_2 \oplus x_3);
 \end{aligned}$$

$$f_{16} = |1001 \ 0111|^T = (x_2 \oplus x_3)!x_1 \vee (x_2 \vee x_3)x_1 = \\ = !x_1!x_2!x_3 \vee (x_1 \# x_2 \# x_3).$$

Используем мультиплексорный метод при $k = 1$. Заданные БФУ содержат десять типов фрагментов длиной четыре: $G_0 = 0$, $G_1 = x_2x_3$, $G_2 = x_2 \vee x_3$, $G_3 = !x_2!x_3$, $G_4 = x_2 \oplus x_3$, $G_5 = !(x_2 \oplus x_3)$, $G_6 = x_2$, $G_7 = x_3$, $G_8 = 1$, $G_9 = !x_2 \vee !x_3$. При этом может быть построен модуль с $2 + \lceil \log 10 \rceil = 6$ входами, из которых два — информационных, а четыре — информационно-настроечных.

Для построения модуля УМЗ с пятью входами исключим первоначально фрагменты G_6 и G_7 , так как они встречаются реже других — только по одному разу в одной функции f_7 . Назначим функции Q_i следующим образом: $Q_0 = 0$, $Q_1 = x_2x_3$, $Q_2 = !x_2 \vee !x_3$, $Q_3 = !(x_2 \oplus x_3)$, $Q_4 = x_2 \vee x_3$, $Q_5 = !x_2!x_3$, $Q_6 = x_2 \oplus x_3$, $Q_7 = 1$.

Функция $F = \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3)$ является порождающей для любой функции из заданного списка, кроме функции f_7 . Функция F реализуется схемой, приведенной на рис. 8.18.

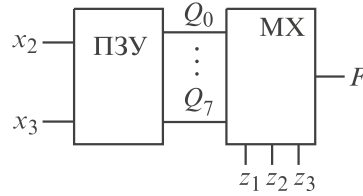


Рис. 8.18

Таблица истинности этого модуля имеет вид:

$$F(z_1, z_2, z_3, x_2, x_3) = \\ = |0000 \ 0001 \ 1110 \ 1001 \ 0111 \ 1000 \ 0110 \ 1111|^T.$$

Определим в качестве примера настройку модуля на реализацию функции f_5 :

$$f_5 = F; \\ \text{MX}(Q_3, Q_6; x_1) = \\ = \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3); \\ z_1 = \text{MX}(0, 1; x_1) = x_1; \\ z_2 = \text{MX}(1, 1; x_1) = 1; \\ z_3 = \text{MX}(1, 0; x_1) = !x_1.$$

Отметим, что в этом случае выполняется необходимое условие получения безынверсных настроек, которое, однако, не является достаточным.

Покажем, как откорректировать построенную БФУ для построения порождающей функции модуля УМЗ. Заменим в функции F переменную x_2 на переменную z_4 . При этом получим ПФО вида:

$$F(z_1, z_2, z_3, z_4, x_3) = \text{MX}(0, 0, 0, x_3, 1, !x_3, !x_3, x_3, x_3, 1, !x_3, 0, x_3, !x_3, 1, 1; z_1, z_2, z_3, z_4).$$

Так как столбец значений этой ПФ содержит все четыре типа фрагментов длиной два: $\{0, !x_3, x_3, 1\}$, то по переменной x_n может быть разложена произвольная функция n переменных. При $n = 3$ число таких разложений равно 4^8 . При реализации функции f_7 будем искать такое разложение, в котором $z_i = \{0, 1, !x_1, x_1, !x_2, x_2\}$.

Составим уравнение

$$\begin{aligned} f_7 &= F(z_1, z_2, z_3, z_4, x_3); \\ \text{MX}(0, 1, x_3, x_3; x_1, x_2) &= \text{MX}(0, 0, 0, x_3, 1, !x_3, \\ & !x_3, x_3, x_3, 1, !x_3, 0, x_3, !x_3, 1, 1; z_1, z_2, z_3, z_4). \end{aligned}$$

Одним из решений этого уравнения является следующее:

$$\begin{aligned} z_1 &= \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1; \\ z_2 &= \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2; \\ z_3 &= \text{MX}(0, 0, 0, 0; x_1, x_2) = 0; \\ z_4 &= \text{MX}(0, 0, 0, 0; x_1, x_2) = 0. \end{aligned}$$

Таким образом, одновыходная схема с пятью входами, представленная на рис. 8.19, является модулем УМЗ.

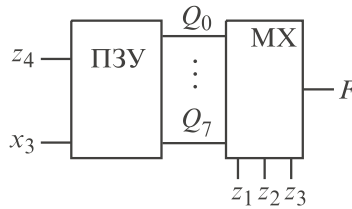


Рис. 8.19

Эта схема имеет один информационный и четыре информационно-настроечных входов. Для всех функций, кроме функции f_7 , $z_4 = x_2$, а для функции f_7 справедливо соотношение $z_4 = 0$. Этот модуль описан в [24].

8.5. Модули, настраиваемые на функции триггеров

1. Пусть требуется построить модуль, реализующий путем настройки три триггера:

$$Q_R(R, S, Q) = |0111 \ 0000|^T = !R(S \vee Q);$$

$$Q_E(R, S, Q) = |0111 \ 0001|^T = !R \# S \# Q;$$

$$Q_S(R, S, Q) = |0111 \ 0011|^T = !RQ \vee S.$$

1.1. Применим мультиплексорный метод при $k = 1$. При этом $G_0 = S \vee Q$, $G_1 = 0$, $G_2 = SQ$, $G_3 = S$. Полагая $Q_i = G_i$, получим:

$$\begin{aligned} Q_1(z_1, z_2, S, Q) &= |0111 \ 0000 \ 0001 \ 0011|^T = \\ &= \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2) = \\ &= !z_1!z_2(S \vee Q) \vee S(z_1 z_2 \vee !z_2 Q). \end{aligned}$$

Определим настройки этого модуля:

$$\begin{aligned} Q_R &= Q_1, \\ \text{MX}(Q_0, Q_1; R) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2), \\ z_1 &= \text{MX}(0, 0; R) = 0, \\ z_2 &= \text{MX}(0, 1; R) = R; \end{aligned}$$

$$\begin{aligned} Q_E &= Q_1, \\ \text{MX}(Q_0, Q_2; R) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2), \\ z_1 &= \text{MX}(0, 1; R) = R, \\ z_2 &= \text{MX}(0, 0; R) = 0; \end{aligned}$$

$$\begin{aligned} Q_S &= Q_1, \\ \text{MX}(Q_0, Q_3; R) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2), \\ z_1 &= \text{MX}(0, 1; R) = R, \\ z_2 &= \text{MX}(0, 1; R) = R. \end{aligned}$$

1.2. Используем мультиплексорный метод при $k = 2$. При этом $G_0 = Q$, $G_1 = 1$, $G_2 = 0$. Назначим функции Q_i следующим образом: $Q_0 = Q$, $Q_1 = Q_2 = Q_3 = 1$, $Q_4 = 0$, $Q_5 = Q$, $Q_6 = 0$, $Q_7 = 1$. При этом

$$\begin{aligned} Q_2(z_1, z_2, z_3, Q) &= |0111 \ 0000 \ 0001 \ 0011|^T = \\ &= \text{MX}(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3) = \\ &= (!z_1 \vee z_3)(z_2 \vee Q) \vee !z_1 z_3. \end{aligned}$$

Определим настройки этого модуля:

$$\begin{aligned}
Q_R &= Q_2, \\
MX(Q_0, Q_2, Q_4, Q_6; R, S) &= \\
= MX(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3), \\
z_1 &= MX(0, 0, 1, 1; R, S) = R, \\
z_2 &= MX(0, 1, 0, 1; R, S) = S, \\
z_3 &= MX(0, 0, 0, 0; R, S) = 0;
\end{aligned}$$

$$\begin{aligned}
Q_E &= Q_2, \\
MX(Q_0, Q_1, Q_4, Q_5; R, S) &= \\
= MX(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3), \\
z_1 &= MX(0, 0, 1, 1; R, S) = R, \\
z_2 &= MX(0, 0, 0, 0; R, S) = 0, \\
z_3 &= MX(0, 1, 0, 1; R, S) = S;
\end{aligned}$$

$$\begin{aligned}
Q_S &= Q_2, \\
MX(Q_0, Q_1, Q_6, Q_7; R, S) &= \\
= MX(Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7; z_1, z_2, z_3), \\
z_1 &= MX(0, 0, 1, 1; R, S) = R, \\
z_2 &= MX(0, 0, 1, 1; R, S) = R, \\
z_3 &= MX(0, 1, 0, 1; R, S) = S.
\end{aligned}$$

Этот модуль описан в [25].

2. Пусть требуется, чтобы модуль кроме перечисленных выше триггеров реализовал бы путем настройки еще один триггер

$$Q_{JK}(R, S, Q) = |0111\ 0010|^T = !RQ \vee S!Q.$$

Несмотря на то что появились дополнительные фрагменты $G_4 = SQ$ (при $k = 1$) и $G_3 = !Q$ (при $k = 2$), вводить их в найденную порождающую формулу модуля не будем, а построим ее в виде:

$$Q_3(z, R, S, Q) = |0111\ 0000\ 0111\ 0011|^T = !R(S \vee Q) \vee Sz.$$

При $z = 0$ — $Q_3 = Q_R$, при $z = Q$ — $Q_3 = Q_E$, при $z = 1$ — $Q_3 = Q_S$, а при $z = !Q$ — $Q_3 = Q_{JK}$. Этот модуль с прямым и инверсным выходами описан в [26].

3. Модуль, реализующий путем настройки функции шести триггеров, описан в [27].

8.6. Модули, настраиваемые на функции генерации

1. Построим модуль, который при отсутствии входных сигналов выдает ноль, при наличии первого входного сигнала генерирует прямоугольные импульсы, а при двух входных сигналах одновременно может функционировать в одном из трех режимов: фиксация состояния, в котором он находился; фиксация единицы; фиксация нуля.

Таким образом, модуль должен настраиваться на реализацию следующих функций:

$$S_1(x_1, x_2, S) = |0000 \ 1001|^T = x_1!(x_2 \oplus S);$$

$$S_2(x_1, x_2, S) = |0000 \ 1011|^T = x_1(x_2 \vee !S);$$

$$S_3(x_1, x_2, S) = |0000 \ 1000|^T = x_1!x_2!S.$$

Применим мультиплексорный метод при $k = 1$. При этом $G_0 = 0$, $G_1 = !(x_2 \oplus S)$, $G_2 = x_2 \vee !S$, $G_3 = !x_2!S$. Назначим функции Q_i следующим образом: $Q_0 = 0$, $Q_1 = !x_2!S$, $Q_2 = !(x_2 \oplus S)$, $Q_3 = x_2 \vee !S$. При этом порождающая формула модуля имеет вид:

$$\begin{aligned} S &= \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2) = \\ &= (z_1 \vee z_2)!x_2!S \vee z_1(z_2 \vee S)x_2. \end{aligned}$$

Определим настройки этого модуля:

$$\begin{aligned} S_1 &= S, \\ \text{MX}(Q_0, Q_2; x_1) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2), \\ z_1 &= \text{MX}(0, 1; x_1) = x_1, \\ z_2 &= \text{MX}(0, 0; x_1) = 0; \end{aligned}$$

$$\begin{aligned} S_2 &= S, \\ \text{MX}(Q_0, Q_3; x_1) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2), \\ z_1 &= \text{MX}(0, 1; x_1) = x_1, \\ z_2 &= \text{MX}(0, 1; x_1) = x_1; \end{aligned}$$

$$\begin{aligned} S_3 &= S, \\ \text{MX}(Q_0, Q_1; x_1) &= \text{MX}(Q_0, Q_1, Q_2, Q_3; z_1, z_2), \\ z_1 &= \text{MX}(0, 0; x_1) = 0, \\ z_2 &= \text{MX}(0, 1; x_1) = x_1. \end{aligned}$$

Этот модуль описан в [28].

2. Построим модуль, который при отсутствии входных сигналов выдает ноль, при наличии любого одного из двух входных сигналов генерирует прямоугольные импульсы, а при двух сигналах одновременно может функционировать в одном из четырех режимов: фиксация состояния, в котором он находился; фиксация единицы; фиксация нуля; фиксация состояния, инверсного относительно состояния, в котором он находился.

Таким образом, модуль должен настраиваться на реализацию следующих функций:

$$S_1(x_1, x_2, S) = |0010 \ 1001|^T = (x_1 \oplus x_2)!S \vee x_1 x_2 S;$$

$$S_2(x_1, x_2, S) = |0010 \ 1011|^T = (x_1 \vee x_2)!S \vee x_1 x_2 S;$$

$$S_3(x_1, x_2, S) = |0010 \ 1000|^T = !S(x_1 \oplus x_2);$$

$$S_4(x_1, x_2, S) = |0010 \ 1010|^T = (x_1 \vee x_2)!S.$$

Используя метод «наложения», построим ПФО:

$$S = [(z_1 \oplus !z_2 z_3) \vee z_2 !z_3]!S \vee z_1 z_3 S.$$

В этом случае

$$\begin{aligned} \text{при } z_1 = x_1, \ z_2 = 0, \ z_3 = x_2 & \text{ — } S = S_1; \\ \text{при } z_1 = x_1, \ z_2 = x_1, \ z_3 = x_2 & \text{ — } S = S_2; \\ \text{при } z_1 = 0, \ z_2 = x_1, \ z_3 = x_2 & \text{ — } S = S_3; \\ \text{при } z_1 = x_1, \ z_2 = x_2, \ z_3 = 0 & \text{ — } S = S_4. \end{aligned}$$

Этот модуль описан в [29].

8.7. Модули с параллельно-последовательной настройкой

Все рассмотренные ранее модули настраивались параллельно. В [30] был предложен модуль с последовательной настройкой, который с помощью входящего в него nm -разрядного статического регистра за nm тактов настраивает вторую входящую в него компоненту — модуль с параллельной настройкой, содержащий n информационных и nm настроечных входов.

В [31] в указанный модуль было введено nm -разрядных статических регистров, которые позволяют через информационные входы настроить входящий в него модуль с параллельной настройкой за m тактов.

В [32] был рассмотрен модуль с последовательной настройкой, который с помощью счетчика не более чем за 2^n тактов настраива-

ет модуль с отдельными n информационными и n настройчными входами.

Применяя этот модуль в качестве прототипа, в [33] было предложено настраиваемое устройство, в котором модуль с отдельными n информационными и m настройчными входами ($n \geq m$) за счет введения блока сброса и m -разрядного регистра может быть настроен через информационные входы за два такта.

Многие из рассмотренных в настоящей главе модулей описаны также в [34].

Выводы

1. Предложены модули, универсальные в классе дизъюнктивных нормальных форм из q букв ($q = 4, 5, 6$).

2. Предложены модули, универсальные в классе произвольных формул в базисе И, ИЛИ, НЕ из q букв ($q = 2, 3, 4, 5$).

3. Предложены модули, универсальные в классе произвольных формул в базисе И, ИЛИ, НЕРАВНОЗНАЧНОСТЬ, НЕ из q букв ($q = 3, 4$).

4. Предложен модуль, универсальный в классе произвольных булевых функций трех переменных. Этот модуль имеет то же число внешних выводов, что и известные модули, однако методы их построения в литературе не описаны.

5. Предложены модули, настраиваемые на функции триггеров.

6. Предложены модули, настраиваемые на функции генерации.

7. Предложены модули с параллельно-последовательной настройкой.

8. Большинство из этих модулей построены с помощью предлагаемого мультиплексорного метода — метода объединения фрагментов и защищены 26 авторскими свидетельствами СССР.

Л и т е р а т у р а

1. Артюхов В. Л., Шалыто А. А. Многофункциональный логический модуль. А.с. СССР № 703803 // Бюл. изобр. 1979. № 46.

2. Артюхов В. Л., Шалыто А. А. Логический модуль. А. с. СССР № 1126947 // Бюл. изобр. 1984. № 44.

3. Артюхов В. Л., Копейкин Г. А., Шалыто А. А., Фишман Л. М. Многофункциональный логический модуль. А. с. СССР № 577925. Зарегистрировано в Госреестре изобретений СССР 28.06.1977.

4. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Многофункциональный модуль. А. с. СССР № 643866 // Бюл. изобр. 1979. № 3.

5. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Настраиваемые модули для управляющих логических устройств. Л.: Энергоиздат, 1981.

6. Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 798802 // Бюл. изобр. 1981. № 3.

7. Rivest R. L. The necessity of feedback in minimal monotone combinational circuits // IEEE Trans. on Computers. 1977. N 6.

8. Артюхов В. Л., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1077052 // Бюл. изобр. 1984. № 8.
9. Артюхов В. Л., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1151949 // Бюл. изобр. 1985. № 15.
10. Артюхов В. Л., Шалыто А. А. Логический многофункциональный модуль. А. с. СССР № 739525 // Бюл. изобр. 1980. № 21.
11. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 758141 // Бюл. изобр. 1980. № 31.
12. Мищенко В. А., Аспидов А. И., Гурьянов А. И. Многофункциональный логический модуль. А. с. СССР № 913365 // Бюл. изобр. 1982. № 10.
13. Артюхов В. Л., Шалыто А. А. Судовые управляющие логические системы. Л.: ИПК СП, 1984.
14. Артюхов В. Л., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 807271 // Бюл. изобр. 1981. № 7.
15. Попов Ю. А., Воронин А. Т., Сладков А. Б. Вопросы проектирования схем с высоким уровнем интеграции на основе КНС-технологии // Дискретные системы. Т.1. Симпозиум IFAC. Рига: Зинатне, 1974.
16. Артюхов В. Л., Копейкин Г. А., Шалыто А. А., Фишман Л. М. Многофункциональный логический модуль. А. с. СССР № 746500 // Бюл. изобр. 1980. № 25.
17. Артюхов В. Л., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1084783 // Бюл. изобр. 1984. № 3.
18. Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1264161 // Бюл. изобр. 1986. № 38.
19. Артюхов В. Л., Ушкарева К. М., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 744550 // Бюл. изобр. 1980. № 24.
20. Артюхов В. Л., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1096637 // Бюл. изобр. 1984. № 21.
21. Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1290290 // Бюл. изобр. 1987. № 6.
22. Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1322253 // Бюл. изобр. 1987. № 25.
23. Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1273914 // Бюл. изобр. 1986. № 44.
24. Артюхов В. Л., Фишман Л. М., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 760451 // Бюл. изобр. 1980. № 32.
25. Артюхов В. Л., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 838699 // Бюл. изобр. 1981. № 22.
26. Артюхов В. Л., Шалыто А. А. Настраиваемый модуль. А. с. СССР № 826338 // Бюл. изобр. 1981. № 16.
27. Викентьев Л. Ф., Аляев Ю. А., Шалыто А. А. и др. Настраиваемый функциональный модуль. А. с. СССР № 1361538 // Бюл. изобр. 1987. № 47.
28. Артюхов В. Л., Бутенко Т. А., Шалыто А. А. Генератор импульсов. А. с. СССР № 851750 // Бюл. изобр. 1981. № 28.
29. Артюхов В. Л., Киселев В. В., Шалыто А. А. Настраиваемое устройство. А. с. СССР № 834696 // Бюл. изобр. 1981. № 20.
30. Якубайтис Э. А. Структура и эффективность многофункционального элемента // Автоматика и вычисл. техника. 1972. № 5.
31. Артюхов В. Л., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 838700 // Бюл. изобр. 1981. № 22.
32. Якубайтис Э. А. Универсальные логические элементы // Автоматика и вычисл. техника. 1973. № 5.
33. Артюхов В. Л., Шалыто А. А. Настраиваемое устройство. А. с. СССР № 890388 // Бюл. изобр. 1981. № 46.
34. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Судовые управляющие логические системы. Л.: ИПК СП, 1982.

Глава 9

Модули, универсальные в классе симметрических функций

9.1. Обзор литературы

В классе всех булевых функций n переменных важное значение имеют функции упорядоченного типа, специфические свойства которых обеспечивают простоту их реализации [1—3].

Среди функций этого типа одними из наиболее «интересных» являются полностью симметрические БФУ, которые в дальнейшем будем называть **симметрическими функциями (СФ)**.

Известно большое число работ, посвященных изучению свойств и реализации СФ, например [4—15].

Стандартная схема для реализации СФ в базисе релейно-контактных элементов, названная **симметрическим многополюсником**, предложена в [4].

Стандартная схема из функциональных элементов, построенная на основе принципа локального кодирования [16] и предназначенная для реализации путем настройки произвольных СФ n переменных, предложена в [17, 18]. Эта схема может быть названа **симметрическим модулем (СМ)**. Будем обозначать такой модуль символом $СМ_n$.

В [19] исследованы функциональные возможности модуля $СМ_n$ при использовании некоторых входных и настроечных алфавитов и предложен весьма трудоемкий метод определения его настройки на реализацию СФ.

В настоящей работе рассматриваются два типа СМ, первый из которых построен на основе принципа локального кодирования, а второй — на основе симметрического многополюсника Шеннона, реализованного на мультиплексорах (МХ).

Излагаются более простые методы реализации СФ в базисе таких модулей, чем известный метод, изложенный в [19].

Понятие «симметрическая реализация» [4] распространяется на реализацию несимметрических функций, в том числе пороговых и произвольных.

Вводятся понятия «симметрическое разложение булевой функции по крайним правым входным переменным» и «многократная разделительная симметрическая декомпозиция булевой функции». Предложены методы построения таких разложений и декомпозиций.

Изложены также методы синтеза комбинационных схем в базе симметрических модулей.

9.2. Основные определения

Функция $f(x_1, \dots, x_n)$ называется симметрической, если она не изменяется при любой перестановке входных переменных.

Число i ($i = 0, \dots, n$) называется рабочим числом симметрической функции, если она принимает единичные значения на всех входных наборах, содержащих i единиц.

Симметрическая функция n переменных может содержать от 0 до n рабочих чисел.

Поэтому число симметрических функций n переменных определяется соотношением

$$N_1(n) = 2^{n+1}.$$

Функция, которая зависит от переменных $x_1, \dots, x_n$ и имеет только одно рабочее число t , называется элементарной СФ (ЭСФ) и обозначается символом $S_t^p(x_1, \dots, x_n)$, а функция, которая, например, имеет три рабочих числа — t, k, l , обозначается символом $S_{t,k,l}^p(x_1, \dots, x_n)$.

При этом справедливо соотношение

$$S_{t,k,l}^p = S_t^p \vee S_k^p \vee S_l^p.$$

Представление СФ через ее рабочие числа будем называть симметрической реализацией (СР) первого типа и использовать в ней для отличия верхний индекс «р».

Симметрическая реализация для СФ n переменных может быть записана также и через код функции [16], который будем называть также кодом настройки. Такую реализацию будем называть СР второго типа и применять в ней верхний индекс «к».

Симметрические реализации могут быть построены не только для симметрических функций, но и для БФУ, не являющихся симметрическими. Для таких функций в СР могут использоваться при

одних методах построения как рабочие числа, так и код настройки, а при других — только код настройки, который по этой причине является более универсальной характеристикой симметрических реализаций. Существование определенной СР для булевой функции обеспечивает возможность ее реализации соответствующим симметрическим модулем при подаче на его информационные входы переменных из входного алфавита, а на настроечные — кода настройки, состоящего из символов настроечного алфавита. При этом симметрическая реализация для БФУ всегда существует [9], но может быть не единственной, что влияет на сложность соответствующего ей симметрического модуля.

Если в перечне входных переменных в СР каждая переменная встречается только один раз, то такую СР будем называть неповторной и повторной — в противном случае.

9.3. Два типа модулей, универсальных в классе симметрических функций

1. Модуль $СМ_n$ первого типа [17] приведен на рис. 9.1.

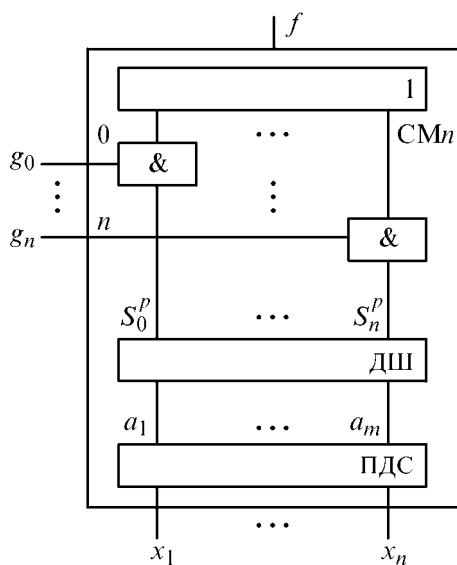


Рис. 9.1

В этой схеме применяются следующие обозначения: ПДС — параллельный двоичный счетчик; ДШ — дешифратор. ПДС определяет двоичный код числа единиц во входном наборе, а ДШ преобразу-

ет этот код в элементарную СФ. Структура остальной части схемы описывается соотношением

$$f = \bigvee_{i=0}^n g_i S_i^p(x_1, \dots, x_n),$$

где $g_i = 0, 1$. При этом отметим, что кортеж значений переменных g_i является кодом настройки модуля.

Число выходов в ПДС определяется соотношением

$$w = \begin{cases} \lfloor \log n \rfloor & \text{при } n \neq 2^k; \\ \lfloor \log n \rfloor + 1 & \text{при } n = 2^k, \end{cases}$$

где $k = 1, 2, 3, \dots$

При $n = 2$ ПДС является одноразрядным двоичным полусумматором (ПΣ) (рис. 9.2), в котором $a_1 = x_1 x_2 = S_2^p(x_1, x_2)$, $a_2 = x_1 \oplus x_2 = S_1^p(x_1, x_2)$.

При $n = 3$ ПДС является одноразрядным двоичным сумматором (рис. 9.3), в котором $a_1 = (x_1 \vee x_2) x_3 \vee x_1 x_2 = S_{2,3}^p(x_1, x_2, x_3)$, $a_2 = x_1 \oplus x_2 \oplus x_3 = S_{1,3}^p(x_1, x_2, x_3)$.

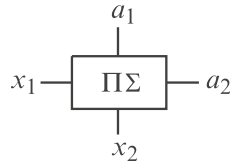


Рис. 9.2

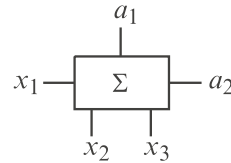


Рис. 9.3

ПДС при $n = 3$ может быть реализован блочной схемой из трех полусумматоров (рис. 9.4).

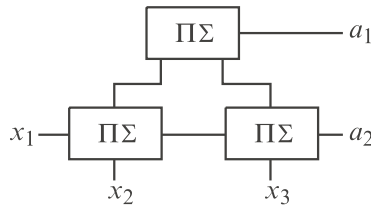


Рис. 9.4

При $n = 4$ ПДС описывается системой булевых формул (СБФ) вида:

$$a_1 = S_4^p(x_1, \dots, x_4), a_2 = S_{2,3}^p(x_1, \dots, x_4), a_3 = S_{1,3}^p(x_1, \dots, x_4).$$

Эта СБФ может быть реализована блочной схемой, представленной на рис. 9.5.

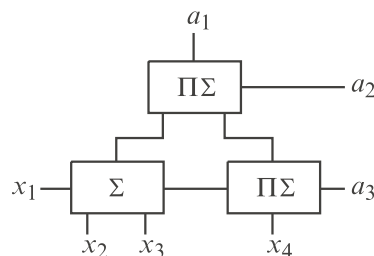


Рис. 9.5

ПДС при $n > 4$ строятся аналогично. Полными (состоящими только из сумматоров) являются ПДС при $n = 2^{k+1} - 1$. Число сумматоров в полных ПДС определяется соотношением

$$\begin{aligned} C(n) &= 1 + 3 + 7 + \dots + ((n-1)/2) = \\ &= 2^1 - 1 + 2^2 - 1 + 2^3 - 1 + \dots + 2^{\log((n+1)/2)} - 1 = \\ &= 2^0 + 2^1 + 2^2 + \dots + 2^{\log((n+1)/2)} - \log((n+1)/2) - 1 = \\ &= 2^{1+\log((n+1)/2)} - \log((n+1)/2) - 2. \end{aligned}$$

Из этого соотношения следует, что при $n = 3$ $C(3) = 1$, при $n = 7$ $C(7) = 4$, при $n = 15$ $C(4) = 11$. На рис. 9.6 в качестве примера приведен ПДС при $n = 15$.

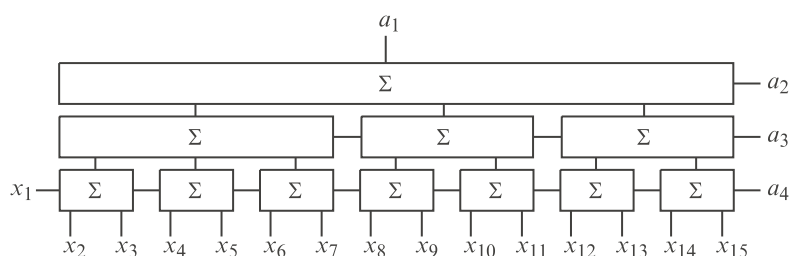


Рис. 9.6

Так как каждый сумматор может быть построен из трех полусумматоров (рис. 9.4), то число последних в полных ПДС определяется соотношением

$$C_1(n) = 3C(n).$$

Отметим, что подход к построению СМ, аналогичный с рассмотренным, изложен в [29].

2. CM второго типа имеют итеративную структуру и состоят только из мультиплексоров «2 в 1». Такие модули строятся в результате замены каждого переключательного контакта в симметрической структуре [4] на мультиплексор «2 в 1». При этом число мультиплексоров в схеме модуля CM n определяется соотношением

$$T(n) = \frac{n(n+1)}{2}.$$

Модуль CM1 реализуется одним мультиплексором «2 в 1» (рис. 9.7).

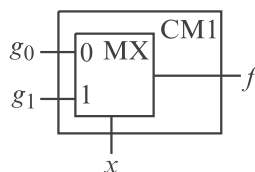


Рис. 9.7

Модуль CM2 реализуется схемой из трех мультиплексоров «2 в 1» (рис. 9.8) либо одним мультиплексором «4 в 1», в котором объединены средние 2^n -входы.

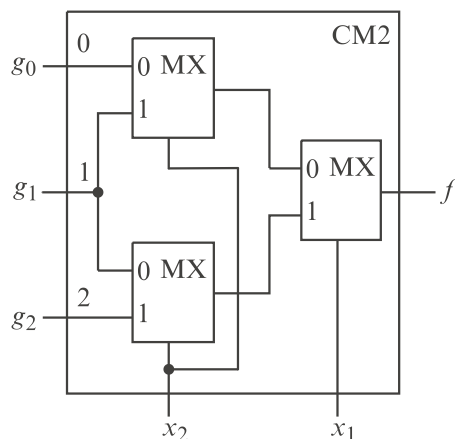


Рис. 9.8

На рис. 9.9 приведен модуль CM3, состоящий из шести мультиплексоров «2 в 1».

3. Модули CM n любого из рассмотренных двух типов имеют n информационных и $n + 1$ настроечных входов, и поэтому общее число входов в них определяется соотношением

$$V_1(n) = 2n + 1.$$

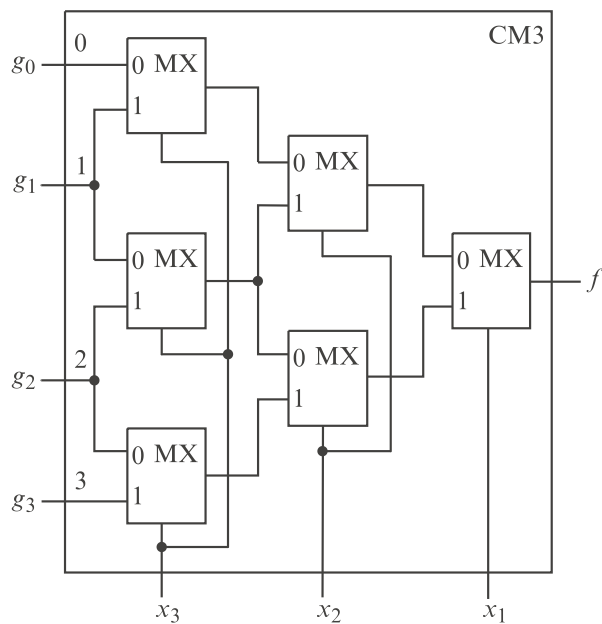


Рис. 9.9

Обратим внимание на то, что на информационные входы СМ входные переменные могут подаваться в произвольном порядке. В дальнейшем для модуля СМ n любого типа будем использовать обозначение, приведенное на рис. 9.10.

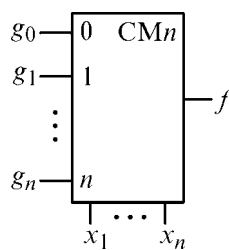


Рис. 9.10

9.4. Методы построения симметрических реализаций булевых функций

1. Входной и настроенный алфавиты. В настоящей работе для симметрических модулей рассматриваются девять классов входных (A^B) и семь классов настроенных (A^H) алфавитов.

В х о д н ы е а л ф а в и т ы:

$$\begin{aligned} A_1^B &= x_1, \dots, x_n; \\ A_2^B &= !x_1, \dots, !x_n; \\ A_3^B &= \tilde{x}_1, \dots, \tilde{x}_n, \text{ где } \tilde{x}_i = !x_i, x_i; \\ A_4^B &= x_1, \dots, x_{n-1}; \\ A_5^B &= x_1, \dots, x_m, \text{ где } 2 \leq m \leq n-2; \\ A_6^B &= \Phi(x_1, \dots, x_{n-1}); \\ A_7^B &= \Phi_1(x_1, \dots, x_{n-1}), \Phi_2(x_1, \dots, x_{n-1}); \\ A_8^B &= \Phi_1(x_1, \dots, x_{n-1}), \dots, \Phi_3(x_1, \dots, x_{n-1}); \\ A_9^B &= \Phi_1(x_1, \dots, x_{n-1}), \dots, \Phi_{R-1}(x_1, \dots, x_{n-1}). \end{aligned}$$

Н а с т р о е ч н ы е а л ф а в и т ы:

$$\begin{aligned} A_1^H &= 0, 1; \\ A_2^H &= 0, 1, x_1, \dots, x_n; \\ A_3^H &= 0, 1, x_1, \dots, x_n, !x_1, \dots, !x_n; \\ A_4^H &= g_0(x_{m+1}, \dots, x_n), \dots, g_m(x_{m+1}, \dots, x_n); \\ A_5^H &= g_0(x_1, \dots, x_n), \dots, g_n(x_1, \dots, x_n); \\ A_6^H &= 0, 1, !x_n, x_n; \\ A_7^H &= g_0(x_{m+1}, \dots, x_n), \dots, g_{R-1}(x_{m+1}, \dots, x_n). \end{aligned}$$

2. Симметрические реализации СФ при использовании алфавитов A_1^B, A_1^H . В этом случае предлагается по таблице истинности (ТИ) заданной БФУ построить таблицу настройки (ТН), содержащую $n + 3$ столбцов и 2^n строк. В ней первый столбец соответствует рабочим числам i , следующие n столбцов соответствуют значениям входных переменных, далее располагаются столбец значений БФУ и столбец, определяющий код настройки. При заполнении этой таблицы предлагается изменить порядок расположения строк в ней относительно порядка их расположения в ТИ по аналогии с тем, как это делается при построении таблицы настройки для модулей Препараты—Мюллера (разд. 11.3 и [20]).

При этом входные наборы разбиваются на $n + 1$ групп, в каждую из которых включается по C_n^i наборов, содержащих i единиц. Группы располагаются в таблице настройки в порядке возрастания значений переменной i , а наборы внутри каждой группы — в порядке возрастания их десятичных эквивалентов.

При $n = 2$ порядок расположения строк в таблице настройки совпадает с их порядком в таблице истинности.

По таблице настройки для заданной БФУ можно определить, является ли она симметрической, и при положительном ответе на этот вопрос (для всех входных наборов с одинаковым значением i значения функции одинаковы) — определить ее рабочие числа и код настройки.

Заполнение столбца кода настройки выполняется на основе соотношения

$$g_i = f^i,$$

где f^i — фрагмент столбца значений ТИ заданной БФУ, соответствующий рабочему числу i .

Пример 9.1. Определить имеет ли функция

$$f_1(x_1, x_2, x_3) = |0001 \ 0111|^T = (x_1 \vee x_2)x_3 \vee x_1x_2$$

симметрическую реализацию.

Построим таблицу настройки (табл. 9.1).

Т а б л и ц а 9.1

i	x_1	x_2	x_3	f_1	g_i^1	f_2	g_i^2	f_3	g_i^3
0	0	0	0	0	0	0	0	0	0
1	0	0	1	0		0		0	
	0	1	0	0	0	1	x_2	1	x_2
	1	0	0	0		0		0	
2	0	1	1	1		1		0	
	1	0	1	1	1	1	x_3	0	$\neg x_3$
	1	1	0	1		0		1	
3	1	1	1	1	1	1	1	1	1

Определим по этой таблице для заданной функции рабочие числа (2, 3) и код настройки (0011), указанный в столбце g_1^i , что позволяет получить симметрические реализации двух типов:

$$f_1 = S_{2,3}^p(x_1, x_2, x_3) = S_{0,0,1,1}^k(x_1, x_2, x_3).$$

Полученные симметрические реализации бесповторны (в каждую реализацию каждая входная переменная входит один раз), в то время как заданная БФ повторна.

На рис. 9.11 приведен модуль СМЗ, настроенный на реализацию заданной функции.

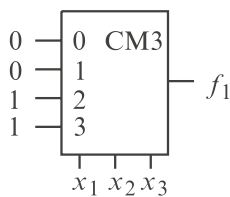


Рис. 9.11

Если для заданной функции f найдена СР, то для ее инверсии такая реализация может быть найдена без построения таблицы настройки, так как в [4] показано, что у функции $!f$ рабочими числами являются все числа от 0 до n , не являющиеся таковыми для функции f .

При этом код настройки функции $!f$ является инверсным коду настройки для функции f .

Пример 9.2. Определить симметрическую реализацию для БФ

$$f = (!x_1 \vee !x_2)!x_3 \vee !x_1!x_2.$$

Так как функция f_1 является самодвойственной, то $f = !f_1$, и поэтому

$$f = S_{0,1}^p(x_1, x_2, x_3) = S_{1,1,0,0}^k(x_1, x_2, x_3).$$

3. Симметрические реализации СФ при использовании алфавитов A_2^B, A_1^H . Возможность такой реализации определяется соотношением [10]

$$f(x_1, \dots, x_n) = S_{p, \dots, l}^p(x_1, \dots, x_n) = S_{n-p, \dots, n-l}^p(!x_1, \dots, !x_n).$$

Пример 9.3. Определить симметрическую реализацию в указанных алфавитах для функции f_1 .

Из изложенного выше следует, что

$$f_1 = S_{0,1}^p(!x_1, !x_2, !x_3).$$

4. Симметрические реализации БФУ при использовании алфавитов A_1^H, A_2^H . В этом случае модуль СМ2 с пятью входами является модулем, универсальным в классе произвольных БФУ двух переменных [19], который обозначим символом УМ2.

Это объясняется тем, что в данном случае

$$2^{C_2^1} = n + 2.$$

В левой части этого соотношения записано число возможных типов фрагментов длиной два в столбце ТИ при $i = 1$, а в правой его части — число символов в алфавите настройки A_2^H при $n = 2$. Для этого модуля $g_0, g_2 = 0, 1; g_1 = 0, 1, x_1, x_2$.

Модуль СМ2 является одним из одновыходных модулей УМ2, при построении которых не применяются какие-либо классификации [21]. Модулями УМ2 являются также:

— модуль СМ1 (МХ «2 в 1») с тремя входами и алфавитами A_4^B, A_6^H ;

— модуль Препараты—Мюллера с четырьмя входами и алфавитами A_1^B, A_3^H ;

— мультиплексор «4 в 1» с шестью входами и алфавитами A_1^B, A_1^H .

При $n > 2$ и алфавитах A_1^B, A_2^H модули СМ n не являются универсальными в классе произвольных БФУ, однако позволяют реализо-

вать кроме симметрических также и некоторые несимметрические функции n переменных. Общее число БФУ n переменных, реализуемых модулем СМ n , в данном случае определяется соотношением [19]

$$N_2(n) = 4(n+2)^{n-1}.$$

Так, модуль СМ3 реализует 100 из 256 функций трех переменных, а модуль СМ4 — 864 из 65 536 функций четырех переменных, в то время как $N_1(3) = 16$, а $N_1(4) = 32$.

Коды настройки модулей в данном случае могут быть определены с помощью таблицы настройки.

Пример 9.4. Определить, существует ли симметрическая реализация для БФУ

$$f_2(x_1, x_2, x_3) = |0011\ 0101|^T.$$

Из рассмотрения столбца f_2 (табл. 9.1) следует, что эта БФУ не является симметрической. Однако для нее может быть построена симметрическая реализация, код настройки которой приведен в столбце g_i^2 этой таблицы. При этом

$$f_2 = S_{0, x_2, x_3, 1}(x_1, x_2, x_3).$$

Полученная СР бесповторна, в то время как БФ для заданной БФУ повторна: $f_2 = !x_1 x_2 \vee x_1 x_3$.

5. Симметрические реализации БФУ при использовании алфавитов A_1^B, A_3^H . Дальнейшее расширение алфавита настройки позволяет еще больше расширить класс функций, реализуемых симметрическими модулями. Так как при $n = 3$

$$2^{C_3^1} = 2^{C_3^2} = 2n + 2,$$

то модуль СМ3 в данном случае универсален в классе произвольных БФУ трех переменных [19]. Этот модуль (рис. 9.9) имеет семь входов и один выход. Для этого модуля $g_0, g_3 = 0, 1; g_1, g_2 = 0, 1, x_1, x_2, x_3, !x_1, x_2, !x_3$ и он

— имеет меньшее число входов по сравнению с мультиплексором «8 в 1», который является модулем УМ3 с алфавитами A_1^B, A_1^H ;

— сложнее, чем мультиплексор «4 в 1» с шестью входами, который является модулем УМ3 с алфавитами A_4^B, A_6^H ;

— имеет более регулярную структуру, но и большее число входов по сравнению с модулем Препараты—Мюллера с шестью входами, который является модулем УМ3 с алфавитами A_1^B, A_3^H .

Число функций n переменных, реализуемых модулем СМ n в данном случае, определяется соотношением [19]

$$N_3(n) = 4(2n+2)^{n+1}.$$

Например, модуль СМ4 реализует 4000 БФУ, из которых только 32 являются симметрическими.

Определение кода настройки предлагается и в данном случае проводить по таблице настройки.

Пример 9.5. Определить возможность симметрической реализации для БФУ

$$f_3(x_1, x_2, x_3) = |0010 \ 0011|^T.$$

Из рассмотрения столбца f_3 (табл. 9.1) следует, что эта БФУ не является симметрической. Однако для нее может быть построена симметрическая реализация, код настройки которой приведен в столбце g_i^3 этой таблицы. При этом

$$f_3 = S_{0, x_2, !x_3, 1}^k(x_1, x_2, x_3).$$

Полученная симметрическая реализация неповторна, как и неповторна БФ для заданной БФУ:

$$f_3 = (x_1 \vee !x_3)x_2.$$

Пример 9.6. Определить возможность симметрической реализации для БФУ [19]

$$f_4(x_1, x_2, x_3, x_4) = |1000 \ 0111 \ 1000 \ 1000|^T.$$

Код настройки модуля СМ4 на реализацию заданной БФУ приведен в столбце g_i^4 табл. 9.2.

Т а б л и ц а 9.2

i	x_1	x_2	x_3	x_4	f_4	g_i^4	f_5	g_i^5	f_6	g_i^6	f_9	g_i^9
0	0	0	0	0	1	1	1	1	0	0	1	1
1	0	0	0	1	0	x_1	1	$!x_3$	0	0	0	$x_2 \oplus x_3$
	0	0	1	0	0		0		0		1	
	0	1	0	0	0		1		0		1	
	1	0	0	0	1		1		0		0	
2	0	0	1	1	0	x_2	1	x_3	1	$x_1x_2 \vee x_3x_4$ или $!x_1!x_2 \vee !x_3!x_4$	1	x_3
	0	1	0	1	1		0		0		0	
	0	1	1	0	1		1		0		1	
	1	0	0	1	0		0		0		0	
	1	0	1	0	0		1		0		1	
	1	1	0	0	1		0		1		0	
3	0	1	1	1	1	$!x_1$	1	x_3	1	1	1	x_3
	1	0	1	1	0		1		1		1	
	1	1	0	1	0		0		1		0	
	1	1	1	0	0		1		1		1	
4	1	1	1	1	0	0	0	0	1	1	0	0

При этом

$$f_4 = S_{1, x_1, x_2, !x_1, 0}^K(x_1, x_2, x_3, x_4).$$

Эта реализация с помощью предлагаемого метода найдена существенно проще, чем в [19].

Полученная СР повторна, в то время как БФ бесповторна:

$$f_4 = !x_1 x_2 \oplus !x_3 !x_4.$$

Пример 9.7. Определить возможность симметрической реализации для БФУ [22]

$$f_5(x_1, x_2, x_3, x_4) = |1101\ 1011\ 1011\ 0010|^T.$$

Код настройки модуля СМ4 на реализацию заданной БФУ приведен в столбце g_i^5 табл. 9.2.

При этом

$$f_5 = S_{1, !x_3, x_3, x_3, 0}^K(x_1, x_2, x_3, x_4).$$

Полученная СР существенно проще, чем БФ

$$f_5 = !x_1 (!x_2 x_4 \vee x_2 x_3 \vee !x_3 !x_4) \vee x_1 (!x_2 (x_3 \vee !x_4) \vee x_3 !x_4).$$

6. Симметрические реализации БФУ при использовании алфавитов A_3^B, A_1^H . Расширение входного алфавита увеличивает и класс функций, реализуемых СМ, так как некоторые несимметрические БФУ в алфавите A_1^B являются СФ в алфавите A_3^B .

Весьма трудоемкий метод построения симметрических реализаций для этого случая изложен в [22].

Отметим, что для алфавита A_3^B в отличие от алфавита A_1^B для БФУ может существовать несколько симметрических реализаций.

Пример 9.8. Определить возможность симметрической реализации для БФУ, рассмотренной в предыдущем примере.

В [22] показано, что

$$f_5 = S_{1, 2}^P(x_1, x_2, !x_3, x_4) = S_{2, 3}^P(!x_1, !x_2, x_3, !x_4).$$

Сравнение этих реализаций с найденной в примере 9.7 показывает, что в алфавите A_3^B заданная БФУ является симметрической и поэтому может быть записана через рабочие числа, в то время как в алфавите A_1^B она несимметрична, но имеет достаточно просто определяемую симметрическую реализацию, использующую код настройки.

7. Стандартные повторные симметрические реализации произвольных БФУ при использовании алфавитов A_1^B, A_1^H . Методы, рассмотренные выше, не позволяют реализовать произвольные БФУ n переменных одним модулем СМ n . В [19] показано, что при $A^B = 0, x_1, \dots, x_n$ и A_1^H модуль СМ n универсален в классе произвольных БФУ m переменных, где $m = \lfloor \log(n+1) \rfloor$.

Предложим простой метод построения стандартной повторной симметрической реализации для произвольной БФУ n переменных, применяя при этом модуль $CM(2^n - 1)$. Такая реализация соответствует теореме, изложенной в [9], в соответствии с которой произвольную БФУ n переменных всегда можно представить в виде симметрической функции, зависящей от $2^n - 1$ переменных.

Предлагаемая стандартная повторная СР базируется на размножении входных переменных [22]. При этом если переменную x_j ($j=1, \dots, n$) повторить 2^{n-j} раз, то для произвольной БФУ n переменных при $g_i = 0, 1$ справедливо соотношение

$$f_1(x_1, \dots, x_n) = \bigvee_{i=0}^{2^n-1} g_i S_i^p(\underbrace{x_1, \dots, x_1}_{2^{n-1}}, \dots, \underbrace{x_n, \dots, x_n}_{2^0}).$$

Такому представлению БФУ соответствует расширенная ТИ, содержащая $2^{n-1} + 2$ столбцов и 2^n строк. В этой таблице в крайнем левом столбце в порядке возрастания указаны рабочие числа от 0 до $2^n - 1$, затем следуют $2^n - 1$ столбцов, соответствующих входным переменным. Крайним правым столбцом таблицы является столбец значений функции. Назовем построенную таким образом таблицу полной. Столбец значений полной таблицы непосредственно (без увеличения числа строк или их перестановки) задает код настройки, подаваемый на настроечные входы модуля, на информационные входы которого переменная x_j подается 2^{n-j} раз.

Пример 9.9. Определить стандартную повторную симметрическую реализацию для функции

$$f_3(x_1, x_2, x_3) = |0010 \ 0011|^T.$$

Используя предлагаемый метод, построим для этой БФУ полную таблицу (табл. 9.3).

Т а б л и ц а 9.3

i	x_1	x_1	x_1	x_1	x_2	x_2	x_3	f_3
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0
2	0	0	0	0	1	1	0	1
3	0	0	0	0	1	1	1	0
4	1	1	1	1	0	0	0	0
5	1	1	1	1	0	0	1	0
6	1	1	1	1	1	1	0	1
7	1	1	1	1	1	1	1	1

Из рассмотрения этой таблицы следует, что

$$\begin{aligned} f_3 &= S_{2,6,7}^p(x_1, x_1, x_1, x_1, x_2, x_2, x_3) = \\ &= S_{0,0,1,0,0,0,1,1}^k(x_1, x_1, x_1, x_1, x_2, x_2, x_3). \end{aligned}$$

Рассмотренный метод реализации произвольных БФУ является беспереборным и строит для них стандартные (одинаковые по сложности применяемых СМ) симметрические реализации, которые для многих БФУ, в том числе симметрических, являются избыточными. Например, СФ, рассмотренная в примере 9.1, реализуется данным методом весьма сложно:

$$f_1 = S_{3,5,6,7}^p(x_1, x_1, x_1, x_1, x_2, x_2, x_3).$$

Поэтому рассмотрим модификации этого метода, которые во многих случаях позволяют уменьшить, а для СФ — исключить повторность входных переменных.

8. Повторные симметрические реализации произвольных БФУ при использовании алфавитов A_1^b , A_1^n и последовательном заполнении полной таблицы. Излагаемый метод строит СР для произвольной БФУ n переменных не более чем за $2^n - n$ шагов.

На первом шаге для заданной БФУ строится остов полной таблицы, в которой заполняются крайние столбцы, а для переменных x_j — крайние правые столбцы. Заданная БФУ проверяется на симметричность на основе подсчета значения переменной i для каждой строки и сравнения значений БФУ при одинаковых значениях переменной i . Если для каждого значения i все значения БФУ совпадают, то она является симметрической, в противном случае она таковой не является. При положительном результате проверки процесс завершается, а иначе осуществляется переход к шагу 2.

На втором шаге заполняется второй столбец с пометкой x_{n-1} за счет дублирования в него содержимого первого столбца с этой пометкой. После этого подсчитываются новые значения переменной i для каждой строки и заполняется крайний левый столбец. Если эта таблица задает симметрическую реализацию, то процесс заканчивается, в противном случае выполняется переход к следующему шагу.

На каждом последующем шаге заполняется слева направо один из столбцов полной таблицы и осуществляется проверка получаемой реализации на симметричность.

Этот процесс является сходящимся, так как не позднее чем за $2^n - n$ шагов искомая реализация будет найдена. При этом в крайнем случае будут заполнены все столбцы полной таблицы, как это имеет место при применении стандартной повторной симметрической реализации, рассмотренной в предыдущем пункте.

Пример 9.10. Определить повторную симметрическую реализацию для функции f_3 , рассмотренной в предыдущем примере.

Построим для заданной БФУ остов полной таблицы (табл. 9.4 без столбца f_5).

Т а б л и ц а 9.4

i	x_1	x_1	x_1	x_1	x_2	x_2	x_3	f_3	f_5
0				0		0	0	0	1
1				0		0	1	0	x_4
1				0		1	0	1	$!x_4$
2				0		1	1	0	1
1				1		0	0	0	$!x_4$
2				1		0	1	0	1
2				1		1	0	1	0
3				1		1	1	1	$!x_4$

Из рассмотрения остова таблицы следует, что заданная БФУ не является СФ, так как имеет различные значения при $i = \text{const}$.

Заполним методом дублирования второй столбец с пометкой x_2 этой таблицы (табл. 9.5).

Т а б л и ц а 9.5

i	x_1	x_1	x_1	x_1	x_2	x_2	x_3	f_3	f_5
0				0	0	0	0	0	1
1				0	0	0	1	0	x_4
2				0	1	1	0	1	$!x_4$
3				0	1	1	1	0	1
1				1	0	0	0	0	$!x_4$
2				1	0	0	1	0	1
3				1	1	1	0	1	0
4				1	1	1	1	1	$!x_4$

Таблица вновь не определяет симметрическую реализацию.

Заполним методом дублирования второй столбец с пометкой x_1 (табл. 9.6). Таблица снова не определяет симметрическую реализацию.

Заполним методом дублирования третий столбец с пометкой x_1 (табл. 9.7).

Т а б л и ц а 9.6

i	x_1	x_1	x_1	x_1	x_2	x_2	x_3	f_3	f_5
0			0	0	0	0	0	0	1
1			0	0	0	0	1	0	x_4
2			0	0	1	1	0	1	$!x_4$
3			0	0	1	1	1	0	1
2			1	1	0	0	0	0	$!x_4$
3			1	1	0	0	1	0	1
4			1	1	1	1	0	1	0
5			1	1	1	1	1	1	$!x_4$

Т а б л и ц а 9.7

i	x_1	x_1	x_1	x_1	x_2	x_2	x_3	f_3
0		0	0	0	0	0	0	0
1		0	0	0	0	0	1	0
2		0	0	0	1	1	0	1
3		0	0	0	1	1	1	0
3		1	1	1	0	0	0	0
4		1	1	1	0	0	1	0
5		1	1	1	1	1	0	1
6		1	1	1	1	1	1	1

Из этой таблицы следует, что

$$f_3 = S_{2,5,6}^p(x_1, x_1, x_1, x_2, x_2, x_3).$$

Эта реализация проще полученной в примере 9.9.

Из изложенного следует, что предложен конструктивный метод, позволяющий уточнить формулировку теоремы из [9], приведенной в п. 7: произвольную БФУ n переменных всегда можно представить в виде СФ, зависящей не более чем от $2^n - 1$ переменных. Таким образом, произвольная БФУ n переменных может быть реализована на одном симметрическом модуле, имеющем не более чем $2^n - 1$ информационных входов.

Пример 9.11. Определить повторную симметрическую реализацию для функции f_5 , рассмотренной в примере 9.7.

Применяя предлагаемый метод, получим СР вида:

$$f_5 = S_{0,1,3,4,6,7,10}^p(x_1, x_1, x_1, x_1, x_2, x_2, x_2, x_2, x_3, x_3, x_4).$$

Эта реализация существенно проще стандартной, которая при $n = 4$ имеет восемь вхождений переменной x_1 . Однако сравнение этой реализации с приведенными в примерах 9.7 и 9.8 показывает, что расширенные алфавиты, использованные в этих примерах, позволяют построить более простые симметрические реализации.

Найденная реализация повторна, как повторна и БФ для этой БФУ, приведенная в примере 9.7.

Пример 9.12. Определить повторную симметрическую реализацию для БФУ

$$f_6(x_1, x_2, x_3, x_4) = |0001\ 0001\ 0001\ 1111|^T.$$

Применяя предлагаемый метод, получим симметрическую реализацию вида:

$$f_6 = S_{3,7,8,9,10,11}^p(x_1, x_1, x_1, x_1, x_2, x_2, x_2, x_2, x_3, x_3, x_4).$$

Эта реализация существенно проще стандартной при $n = 4$. Она повторна, в то время как формула $f_6 = x_1 x_2 \vee x_3 x_4$ бесповторна.

9. Влияние повышения повторности симметрических реализаций на увеличение числа БФУ, для которых такие реализации существуют. Покажем для $n = 3$, как введение каждой новой переменной расширяет класс функций, допускающих симметрические реализации.

Обозначим значение функции $f(x_1, x_2, x_3)$ на входном наборе с номером w (десятичный эквивалент) символом $f(w)$.

Запишем $n + 1 = 4$ условия, при выполнении которых заданная БФУ является СФ: $f(1) = f(2)$, $f(1) = f(4)$, $f(3) = f(5)$, $f(3) = f(6)$.

Этим условиям удовлетворяют 16 из 256 БФУ трех переменных, так как в этом случае существуют четыре, не связанные между собой, группы значений функции: $f(0)$, $f(1) = f(2) = f(4)$, $f(3) = f(5) = f(6)$, $f(7)$.

Для того чтобы симметрической была функция $f(x_1, x_2, x_2, x_3)$ требуется выполнение трех условий: $f(1) = f(4)$, $f(2) = f(5)$, $f(3) = f(6)$. Этим условиям удовлетворяют 32 БФУ, так как число не связанных между собой групп значений функции в этом случае равно пяти: $f(0)$, $f(1) = f(4)$, $f(2) = f(5)$, $f(3) = f(6)$, $f(7)$.

При этом отметим, что в состав этих функций входят не все функции, которые при их записи в виде $f(x_1, x_2, x_3)$ были симметрическими. Например, функция f_1 не является симметрической при повторении переменной x_2 .

Для того чтобы функция $f(x_1, x_1, x_2, x_2, x_3)$ была симметрической, должны выполняться только два условия: $f(2) = f(4)$, $f(3) = f(5)$. Этим условиям удовлетворяют 64 БФУ, так как в этом

случае несвязанными являются шесть групп: $f(0), f(1), f(2)=f(4), f(3)=f(5), f(6), f(7)$.

Для того чтобы функция $f(x_1, x_1, x_1, x_2, x_2, x_3)$ была симметрической, должно выполняться только одно условие — $f(3) = f(4)$. Этому условию удовлетворяют 128 БФУ.

Для представления заданной БФУ в виде $f(x_1, x_1, x_1, x_1, x_2, x_2, x_3)$ условия отсутствуют: так может быть представлена любая из 256 функций трех переменных.

Пример 9.13. Определить условия, при выполнении которых получается симметрическая реализация, приведенная в примере 9.10.

Из рассмотрения ТИ для функции f_3 следует, что

$$\begin{aligned} f_3(0) = f_3(1) = f_3(3) = f_3(4) = f_3(5) = 0, \\ f_3(2) = f_3(6) = f_3(7) = 1. \end{aligned}$$

Таким образом, для заданной БФУ выполняется только одно из приведенных выше условий — $f_3(3) = f_3(4)$. Это определяет симметрическую реализацию с дополнительно введенными одной переменной x_2 и двумя переменными x_1 .

10. Повторные симметрические реализации произвольных БФУ при использовании алфавитов A_1^B, A_1^H и переборе вариантов при заполнении полной таблицы. Последовательный характер и направление заполнения полной таблицы в рассмотренном в п. 8 методе не позволяют в общем случае находить наилучшие (в рамках полной таблицы) решения.

Если эти ограничения снять, то такие решения за счет перебора могут быть найдены.

Пример 9.14. Определить простейшую (в рамках полной таблицы) симметрическую реализацию для БФУ

$$f_7(x_1, x_2, x_3) = |0000 \ 0111|^T.$$

Повторные СР этой БФУ, построенные без перебора, имеют вид:

$$\begin{aligned} f_7 &= S_{5,6,7}^P(x_1, x_1, x_1, x_1, x_2, x_2, x_3) = \\ &= S_{4,5,6}^P(x_1, x_1, x_1, x_2, x_2, x_3). \end{aligned}$$

Проводя перебор в рамках полной таблицы, получим:

$$f_7 = S_{3,4}^P(x_1, x_1, x_2, x_3).$$

Все приведенные симметрические реализации повторны, в то время как формула $f_7 = x_1(x_2 \vee x_3)$ бесповторна.

11. Простейшие повторные симметрические реализации произвольных БФУ при использовании алфавитов A_1^B, A_1^H . Находясь в рамках полной таблицы, для ряда БФУ n переменных мини-

мальные решения не могут быть найдены, так как структура этой таблицы ограничивает объем перебора. Так, например, для функции $f = (x_1 \vee x_2)x_3$ простейшая симметрическая реализация

$$f = S_{3,4}^p(x_1, x_2, x_3, x_3)$$

в рамках полной таблицы не может быть найдена ни при каком объеме перебора, так как в полной таблице переменная x_n не может быть дублирована.

Минимальные реализации в рамках рассматриваемого подхода определяются в результате полного перебора кортежей входных переменных и их дубликатов.

Пример 9.15. Определить симметрическую реализацию для БФУ

$$f_8(x_1, x_2, x_3) = |1101\ 0100|^T = !x_1(!x_2 \vee x_3) \vee !x_2x_3.$$

Используя подходы, изложенные в предыдущих пунктах, получим:

$$\begin{aligned} f_8 &= S_{0,1,3}^p(x_1, x_1, x_2, x_2, x_3) = \\ &= S_{0,3,4}^p(x_1, x_2, x_3, x_3, x_3). \end{aligned}$$

Применяя метод, изложенный в п. 14, в данном случае можно построить и неповторную симметрическую реализацию:

$$f_8 = S_{2,3}^p(!x_1, !x_2, x_3).$$

12. Симметрические реализации положительных (безынверсных) пороговых функций при использовании алфавитов A_1^B , A_1^H . Если для заданной БФУ известно, что она является пороговой [23], то ее минимальная повторная симметрическая реализация может быть найдена без перебора.

Естественно, что нахождение пороговых реализаций (представление БФУ через веса и порог) даже для положительных пороговых функций (ППФ) весьма трудоемко [23]. Однако при $n \leq 6$ существуют каталоги таких реализаций, например в [24], а для положительных неповторных пороговых формул в разд. 15.5 и в [25] предложен простой метод их построения.

Изложим метод построения СР для рассматриваемого класса функций. Этот метод состоит в следующем:

— для заданной ППФ строится пороговая реализация (ПР):

$$f(x_1, \dots, x_n) = \text{sign}(p_1x_1 + \dots + p_nx_n - T),$$

где p_i — вес переменной x_i ; T — порог функции;

- определяется значение $P = \sum_{i=1}^n p_i$;
- строится СФ «голосование T и более из P » вида:

$$f = S_{T, \dots, P}^P(\underbrace{x_1, \dots, x_1}_{p_1}, \dots, \underbrace{x_n, \dots, x_n}_{p_n}),$$

которая является искомой.

При этом отметим, что из [24] следует, что при $n \geq 5$ существуют ППФ, для которых веса всех переменных не равны единице, что приводит для этих функций к повторным симметрическим реализациям, отличным от получаемых с помощью методов, изложенных в пунктах 7—10, при использовании которых переменная x_n не размножается.

Пример 9.16. Определить симметрическую реализацию для положительной бесповторной пороговой формулы f_7 , рассмотренной в примере 9.14.

Из [25] следует, что $f_7 = \text{sign}(2x_1 + x_2 + x_3 - 3)$.

Так как в этом случае $P = 4$, $T = 3$, то симметрическая реализация имеет вид: $S_{3,4}^P(x_1, x_1, x_2, x_3)$. Эта симметрическая реализация была получена в примере 9.14 за счет перебора в рамках полной таблицы.

Так как для функции

$$f^D = !f(!x_1, \dots, !x_n)$$

справедливо соотношение [23]

$$T^D = P - T + 1,$$

то пороговая реализация функции f^D , двойственной (дуальной) заданной функции f , имеет вид [23]:

$$f^D(x_1, \dots, x_n) = \text{sign}(p_1 x_1 + \dots + p_n x_n - (P - T + 1)).$$

Таким образом,

$$f^D = S_{P-T+1, \dots, P}^P(\underbrace{x_1, \dots, x_1}_{p_1}, \dots, \underbrace{x_n, \dots, x_n}_{p_n}).$$

Пример 9.17. Определить симметрическую реализацию для формулы $f = x_1 \vee x_2 x_3$.

Так как $f = f_7^D$, то из предыдущего примера следует, что $P - T + 1 = 2$, и поэтому

$$f = S_{2,3,4}^P(x_1, x_1, x_2, x_3).$$

Если функция является самодвойственной ($f = f^D$), то для нее справедливо соотношение $T = (P + 1) / 2$.

Пример 9.18. Определить симметрическую реализацию для самодвойственной БФУ

$$f = x_1 (x_2 \vee x_3 \vee x_4) \vee x_1 x_2 x_3.$$

Из каталога [24] следует, что $f = \text{sign} (2x_1 + x_2 + x_3 + x_4 - 3)$.

При этом $p_1 = 2, p_2 = p_3 = p_4 = 1, P = 5, T = 3$ и

$$f = S_{3,4,5}^P(x_1, x_1, x_2, x_3, x_4).$$

Эта функция реализуется модулем СМ5 с 11 входами, в то время как модуль, универсальный в классе самодвойственных функций четырех переменных, имеет 12 входов [26].

13. Симметрические реализации отрицательных пороговых функций при использовании алфавитов A_1^B, A_1^H . Из изложенного в п. 2 следует, что для отрицательных пороговых функций может быть предложен указанный ниже метод построения симметрических реализаций:

— определяется пороговая реализация для положительной пороговой функции, инверсной заданной;

— искомой является реализация вида:

$$f = S_{0, \dots, T-1}^P(\underbrace{x_1, \dots, x_1}_{p_1}, \dots, \underbrace{x_n, \dots, x_n}_{p_n}).$$

Пример 9.19. Определить симметрическую реализацию для булевой формулы $f = !x_1 \vee !x_2 !x_3$.

Так как $f = !f_7$, то из изложенного выше и примера 9.16 следует, что

$$f = S_{0,1,2}^P(x_1, x_1, x_2, x_3).$$

14. Симметрические реализации пороговых функций при использовании алфавитов A_3^B, A_1^H . Произвольная пороговая функция (ПФ) может быть реализована следующим образом:

— для заданной ПФ строится БФ;

— строится симметрическая реализация для положительной ПФ, однотипной с заданной;

— инверсные переменные, входящие в БФ, заменяют в построенной симметрической реализации безынверсные переменные.

Пример 9.20. Определить симметрическую реализацию для булевой формулы $f = !x_1 (x_2 \vee !x_3)$.

Из изложенного выше и примера 9.16 следует, что

$$f = S_{3,4}^P(!x_1, !x_1, x_2, !x_3).$$

Таким образом, из изложенного выше (начиная с п. 12) следует, что при реализации ПФ симметрическим модулем его число входов определяется соотношением

$$V_2(P) = 2P + 1. \quad (9.1)$$

Е. Гото [23] показал, что среди пороговых функций n переменных существует такая функция, для которой

$$C \cdot 2^{n+2} \leq P,$$

где $C = 0.1163171\dots$

Следовательно,

$$C \cdot 2^{n+3} \leq V_2(n).$$

В табл. 9.8 для $n = 2, \dots, 6$ на основе [24] приведены максимальные значения P для класса ПФ и двух его подклассов — самодвойственных ПФ (СПФ) и бесповторных ПФ (БПФ).

Т а б л и ц а 9.8

Класс функций	n				
	2	3	4	5	6
СПФ	—	3	5	9	17
БПФ	2	4	7	12	20
ПФ	2	4	8	16	33

В [27] было показано, что максимальные значения P для БПФ удовлетворяют соотношению

$$P = F_{n+2} - 1, \quad (9.2)$$

где F_{n+2} — $(n + 2)$ -е число Фибоначчи; $F_1 = F_2 = 1$.

Из соотношений 9.1 и 9.2, в частности, следует, что самая сложная БПФ при $n = 4$ может быть реализована модулем СМ7, так как $F_6 = 8$, в то время как непороговая бесповторная БФ (ББФ), рассмотренная в примере 9.12, реализуется модулем СМ11, что свидетельствует о том, что непороговые ББФ реализуются СМ весьма сложно.

15. Стандартные повторные симметрические реализации произвольных БФУ при использовании алфавитов A_4^B, A_3^H . В [19] показано, что в этом случае модуль СМ n универсален в классе

произвольных БФУ $m + 1$ переменных, где $m = [\log (n + 1)]$. При этом настройка модуля определяется достаточно сложно.

Предложим стандартную симметрическую реализацию, для которой настройка модуля может быть найдена сравнительно просто. При этом для заданной БФУ n переменных будем искать настройку модуля $CM(2^{n-1} - 1)$.

Метод состоит в следующем. В ТИ заданной БФУ исключается столбец, соответствующий переменной x_n , а столбец ее значений преобразуется с помощью правил:

$$|00|^T = 0; |01|^T = x_n; |10|^T = !x_n; |11|^T = 1.$$

Полученную ТИ будем называть сокращенной. По такой таблице строится полная таблица, рассмотренная в п. 7, по которой и определяется настройка модуля.

Пример 9.21. Определить симметрическую реализацию для функции f_5 из примера 9.7.

Сокращенная ТИ может быть записана в виде:

$$f_5(x_1, x_2, x_3) = |1, x_4, !x_4, 1, !x_4, 1, 0, !x_4|^T.$$

Поэтому

$$f_5 = S_{1, x_4, !x_4, 1, !x_4, 1, 0, !x_4}(x_1, x_1, x_1, x_1, x_2, x_2, x_2, x_3).$$

16. Повторные симметрические реализации произвольных БФУ при использовании алфавитов A_4^B, A_3^H . Для поиска более простых сокращенной ТИ заданной БФУ по сравнению с предыдущим случаем симметрических реализаций предлагается применять метод, изложенный в п. 8.

Пример 9.22. Определить симметрическую реализацию для функции f_5 , приведенной в предыдущем примере.

Из рассмотрения табл. 9.4—9.6 следует, что симметрическая реализация строится на третьем шаге (табл. 9.6). При этом

$$f_5 = S_{1, x_4, !x_4, 1, 0, !x_4}(x_1, x_1, x_2, x_2, x_3).$$

Увеличение объема перебора, как и для случая A_1^B, A_1^H (пункты 10, 11), позволяет для ряда БФУ находить симметрические реализации с меньшей повторностью переменных, чем это обеспечивает метод, изложенный в настоящем пункте.

17. Симметрические реализации БФУ при использовании алфавитов A_5^B, A_4^H . Стандартная схема СС1, в которой применя-

ется постоянное запоминающее устройство (ПЗУ), приведена на рис. 9.12.

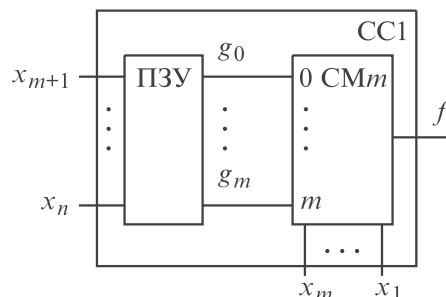


Рис. 9.12

Эта схема позволяет реализовать БФУ n переменных, для которых в таблице настройки модуля СМ m каждому рабочему числу i соответствует одна функция $g_0(x_{m+1}, \dots, x_n)$.

Пример 9.23. Определить симметрическую реализацию для функции f_5 из примера 9.7.

Построим для этой функции таблицу настройки при $m = 2$ (табл. 9.9).

Т а б л и ц а 9.9

i	x_1	x_2	g_i^5
0	0	0	$\neg x_3 \vee x_4$
1	0	1	$x_3 \vee \neg x_4$
2	1	1	$x_3 \neg x_4$

Из рассмотрения этой таблицы следует, что

$$f_5 = S_{\neg x_3 \vee x_4, x_3 \vee \neg x_4, x_3 \neg x_4}^K(x_1, x_2).$$

18. Симметрические реализации произвольных БФУ при использовании алфавитов A_1^B, A_5^H . Стандартная схема СС2 приведена на рис. 9.13. В этой схеме комбинационная схема (КСх) является более простой, чем постоянное запоминающее устройство (ПЗУ). Это ограничение является важным, так как в ПЗУ $(n, 1)$ может быть реализована произвольная БФУ n переменных, что делает применение рассматриваемой стандартной схемы при использовании ПЗУ нецелесообразным.

Схема СС2 позволяет реализовать БФУ n переменных, для которых в таблице настройки модуля СМ n каждому рабочему числу i

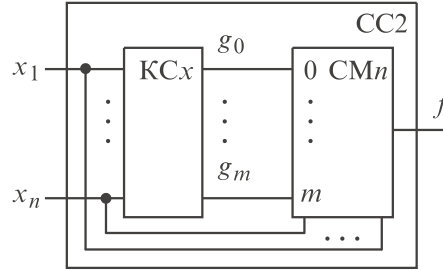


Рис. 9.13

соответствует одна функция $g_i(x_1, \dots, x_n)$. Так как это имеет место для любой функции n переменных, то рассматриваемая схема универсальна в классе БФУ n переменных.

В таблице настройки функция g_i задана на C_n^i наборах. Булева формула, реализующая эту функцию, строится в результате ее доопределения, например с помощью карт Карно. Возможны два исхода доопределений: каждая функция g_i зависит менее чем от n переменных; одна или несколько функций g_i зависят от n переменных.

При первом исходе применение схемы CC2 целесообразно, а при втором, если $g_i(x_1, \dots, x_n) = f(x_1, \dots, x_n)$, — бессмысленно.

Если $g_i(x_1, \dots, x_n) \neq f(x_1, \dots, x_n)$, то вопрос о применении схемы может обсуждаться.

Пример 9.24. Определить симметрическую реализацию для БФУ

$$f_9(x_1, x_2, x_3, x_4) = |1011 \ 1011 \ 0011 \ 0010|^T.$$

Настройка модуля СМ4 на реализацию этой функции приведена в столбце g_i^9 (табл. 9.2). При этом

$$f_9 = S_{1, x_2 \oplus x_3, x_3, x_3, 0}^k(x_1, x_2, x_3, x_4).$$

Пример 9.25. Определить симметрическую реализацию для БФУ

$$f_6(x_1, x_2, x_3, x_4) = |0001 \ 0001 \ 0001 \ 1111|^T.$$

Настройка модуля СМ4 на реализацию этой функции приведена в столбце g_i^6 (табл. 9.2). В этом случае $g_0^6 = g_1^6 = 0$, $g_2^6 = \{x_1 x_2 \vee x_3 x_4, !x_1 !x_2 \vee !x_3 !x_4\}$, $g_3^6 = g_4^6 = 1$.

Так как $f = x_1 x_2 \vee x_3 x_4$, то при $g_2^6 = x_1 x_2 \vee x_3 x_4$ использование схемы СС2 бессмысленно. При $g_2^6 = !x_1 !x_2 \vee !x_3 !x_4$ заданная БФУ весьма экзотично реализуется по функции, ей антидвойственной (разд. 10.5) и [26].

19. Симметрические реализации БФУ при использовании алфавитов A_6^B, A_6^H . Стандартная схема СС3 приведена на рис. 9.14.

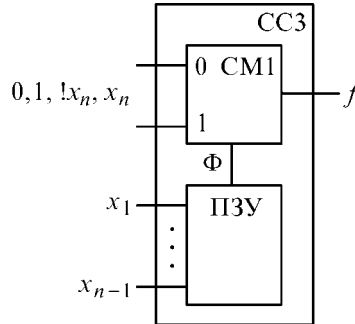


Рис. 9.14

Она позволяет реализовать БФУ n переменных, в каждой из которых столбец значений содержит два типа фрагментов длиной два. Отметим, что для случая $|00|^T = 0$ и $|11|^T = 1$ рассматриваемую схему применять нецелесообразно, так как заданная БФУ не зависит от переменной x_n . В этой схеме ПЗУ программируется (выбирается функция Φ) таким образом, чтобы обеспечить порядок расположения фрагментов в соответствии со столбцом значений реализуемой БФУ. Для каждой БФУ этого класса решение неоднозначно, так как используемые два символа из множества $\{0, 1, !x_n, x_n\}$ могут подключаться к настроечным входам модуля СМ1 двумя способами.

Функцию Φ предлагается определять с помощью мультиплексорного метода.

Пример 9.26. Определить симметрическую реализацию для БФУ

$$f_{10}(x_1, x_2, x_3) = |0101 \ 1001|^T.$$

Столбец значений этой функции состоит из двух типов фрагментов длиной два. Назначим функции g_i следующим образом: $g_0 = x_3, g_1 = !x_3$.

Запишем уравнение

$$f = S_{x_3, !x_3}^k(\Phi),$$

которое в мультиплексорной форме имеет вид:

$$MX(x_3, x_3, !x_3, x_3; x_1, x_2) = MX(x_3, !x_3; \Phi);$$

$$\Phi = MX(0, 0, 1, 0; x_1, x_2) = x_1!x_2.$$

Таким образом,

$$f = S_{x_3, !x_3}^{\kappa} (x_1!x_2).$$

Аналогично может быть получено второе соотношение

$$f = S_{!x_3, x_3}^{\kappa} (!x_1 \vee x_2).$$

Рассмотренный подход является первым частным случаем «симметрического разложения БФУ по переменной x_n » (п. 21). Для функций рассматриваемого класса это разложение порождает одну остаточную функцию меньшего числа переменных.

Так как модуль CM1 совпадает с мультиплексором «2 в 1», то для рассматриваемого класса функций симметрическое разложение по переменной x_n совпадает с разложением Шеннона [28] по функции Φ .

20. Симметрические реализации БФУ при использовании алфавитов A_7^B , A_6^H . Стандартная схема CC4 приведена на рис. 9.15. Она позволяет реализовать БФУ n переменных, в каждой из которых столбец значений содержит три типа фрагментов длиной два. Функции Φ_1 и Φ_2 в этой схеме определяются с помощью мультиплексорного метода.

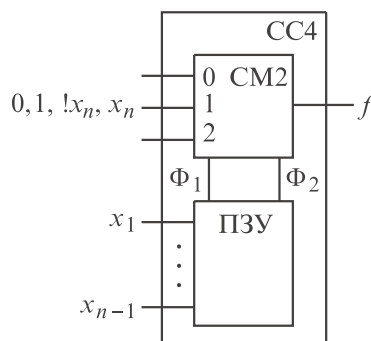


Рис. 9.15

Пример 9.27. Определить симметрическую реализацию для функции f_3 из примера 9.5.

Столбец значений этой функции содержит три типа фрагментов длиной два. Назначим функции g_i следующим образом: $g_0 = 0$, $g_1 = !x_3$, $g_2 = 1$.

Запишем уравнение

$$f_3 = S_{0,!x_3,1}^{\kappa}(\Phi_1, \Phi_2),$$

которое в мультиплексорной форме имеет вид:

$$MX(0,!x_3,0,1;x_1,x_2) = MX(0,!x_3,!x_3,1;\Phi_1,\Phi_2).$$

Это уравнение имеет два решения:

$$\Phi_1 = MX(0,0,0,1;x_1,x_2) = x_1x_2,$$

$$\Phi_2 = MX(0,1,0,1;x_1,x_2) = x_2;$$

$$\Phi_1 = MX(0,1,0,1;x_1,x_2) = x_2,$$

$$\Phi_2 = MX(0,0,0,1;x_1,x_2) = x_1x_2.$$

Таким образом,

$$f = S_{0,!x_3,1}^{\kappa}(x_1x_2, x_2) = S_{0,!x_3,1}^{\kappa}(x_2, x_1x_2).$$

Из сравнения этих симметрических реализаций с полученной в примере 9.4 следует, что усложнение функций, подаваемых на информационные входы модуля, позволило реализацию упростить.

Рассмотренный подход является вторым частным случаем «симметрического разложения БФУ по переменной x_n ».

21. Симметрические реализации произвольных БФУ при использовании алфавитов A_8^B, A_6^H . Стандартная схема СС5 приведена на рис. 9.16.

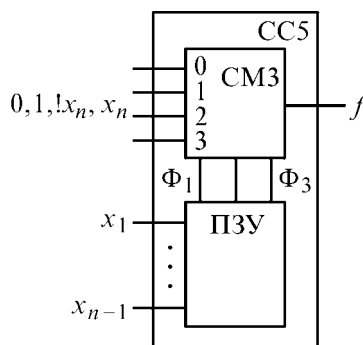


Рис. 9.16

Она позволяет реализовать БФУ n переменных, в каждой из которых столбец значений содержит до четырех типов фрагментов длиной два. Так как это имеет место для любой БФУ n переменных,

то рассматриваемая структура универсальна в классе произвольных БФУ указанного числа переменных. Функции Φ_1, Φ_2, Φ_3 в этой схеме определяются с помощью мультиплексорного метода.

Пример 9.28. Определить симметрическую реализацию для БФУ

$$f_4(x_1, x_2, x_3, x_4) = |1000 \ 0111 \ 1000 \ 1000|^T.$$

Столбец значений этой функции содержит четыре типа фрагментов длиной два. Назначим функции g_i следующим образом: $g_0 = !x_4, g_1 = 0, g_2 = x_4, g_3 = 1$.

Запишем уравнение

$$f_4 = S_{!x_4, 0, x_4, 1}^{\kappa}(\Phi_1, \Phi_2, \Phi_3),$$

которое в мультиплексорной форме имеет вид:

$$\begin{aligned} &MX(!x_4, 0, x_4, 1, !x_4, 0, !x_4, 0; x_1, x_2, x_3) = \\ &= MX(!x_4, 0, 0, x_4, 0, x_4, x_4, 1; \Phi_1, \Phi_2, \Phi_3). \end{aligned}$$

Это уравнение имеет $3^3 \cdot 3 = 81$ решение:

$$\begin{aligned} \Phi_1 &= MX(0, ?, ?, 1, 0, ?, 0, ?; x_1, x_2, x_3); \\ \Phi_2 &= MX(0, ?, ?, 1, 0, ?, 0, ?; x_1, x_2, x_3); \\ \Phi_3 &= MX(0, ?, ?, 1, 0, ?, 0, ?; x_1, x_2, x_3). \end{aligned}$$

Среди этих решений выберем следующее:

$$\begin{aligned} \Phi_1 &= MX(0, 0, 0, 1, 0, 0, 0, 1; x_1, x_2, x_3) = x_2 x_3; \\ \Phi_2 &= MX(0, 1, 1, 1, 0, 0, 0, 0; x_1, x_2, x_3) = !x_1 (x_2 \vee x_3); \\ \Phi_3 &= MX(0, 0, 1, 1, 0, 1, 0, 0; x_1, x_2, x_3) = !x_1 x_2 \vee x_1 !x_2 x_3. \end{aligned}$$

Таким образом,

$$f_4 = S_{!x_4, 0, x_4, 1}^{\kappa}(x_2 x_3, !x_1 (x_2 \vee x_3), !x_1 x_2 \vee x_1 !x_2 x_3).$$

Из сравнения этой симметрической реализации с полученной в примере 9.4 следует, что усложнение функций, подаваемых на информационные входы модуля, позволило упростить реализацию.

Рассмотренный подход может рассматриваться в качестве метода построения нового типа разложения произвольных булевых функций по одной переменной. Оно может быть названо «симметрическим разложением произвольной БФУ n переменных по переменной x_n ». В результате этого разложения в общем случае формируются три остаточные

функции меньшего числа переменных. С учетом частных случаев, рассмотренных в двух предыдущих пунктах, можно утверждать, что предлагаемое разложение порождает не более трех остальных функций, в то время как разложение Шеннона по одной переменной — не более двух.

22. Симметрические реализации произвольных БФУ при использовании алфавитов A_8^B, A_9^H . Стандартная схема СС6 приведена на рис. 9.17. Она позволяет реализовать БФУ n переменных, в каждой из которых столбец значений содержит R разнотипных фрагментов длиной 2^{n-m} . При этом ПЗУ1 реализует эти фрагменты, а ПЗУ2 обеспечивает порядок их расположения в соответствии с ТИ реализуемой БФУ. Функции $\Phi_1, \dots, \Phi_{R-1}$ в этой схеме определяются с помощью мультиплексорного метода.

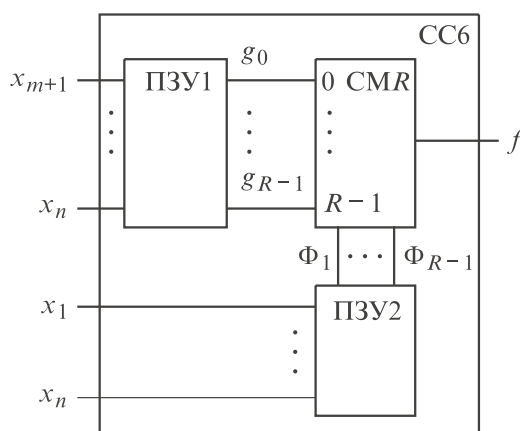


Рис. 9.17

Пример 9.29. Определить симметрическую реализацию для функции f_4 , рассмотренной в предыдущем примере.

При $m = 2$ столбец значений этой функции содержит два типа фрагментов длиной четыре. Назначим функции g_i следующим образом: $g_0 = !x_3 !x_4$, $g_1 = x_3 \vee x_4$.

Запишем уравнение

$$f_4 = S_{g_0, g_1}^{\kappa}(\Phi),$$

которое в мультиплексорной форме имеет вид:

$$MX(g_0, g_1, g_0, g_0; x_1, x_2) = MX(g_0, g_1; \Phi).$$

Это уравнение имеет одно решение:

$$\Phi = MX(0, 1, 0, 0; x_1, x_2) = !x_1 x_2.$$

Таким образом,

$$f_4 = S_{!x_3!x_4, x_3 \vee x_4}^{\kappa} (!x_1 x_2).$$

Из сравнения найденной симметрической реализации с полученной в предыдущем примере следует, что усложнение функций, подаваемых на настроечные входы модуля, позволило резко упростить реализацию: использовать модуль СМ1 вместо модуля СМ3.

Пример 9.30. Определить симметрическую реализацию для функции f_9 , рассмотренной в примере 9.24.

При $m = 2$ столбец значений этой функции содержит три типа фрагментов длиной четыре. Назначим функции g_i следующим образом: $g_0 = x_3 \vee !x_4$, $g_1 = x_3$, $g_2 = x_3 !x_4$.

Запишем уравнение

$$f_9 = S_{g_0, g_1, g_2}^{\kappa} (\Phi_1, \Phi_2),$$

которое в мультиплексорной форме имеет вид:

$$MX(g_0, g_0, g_1, g_2; x_1, x_2) = MX(g_0, g_1, g_1, g_2; \Phi_1, \Phi_2).$$

Это уравнение имеет два решения, из которых выберем следующее:

$$\Phi_1 = MX(0, 0, 0, 1; x_1, x_2) = x_1 x_2;$$

$$\Phi_2 = MX(0, 0, 1, 1; x_1, x_2) = x_1.$$

Таким образом,

$$f_9 = S_{x_3 \vee !x_4, x_3, x_3 !x_4}^{\kappa} (x_1 x_2, x_1).$$

Изложенный подход может рассматриваться в качестве метода построения нового типа функциональной декомпозиции произвольных булевых функций. Эта декомпозиция, используя термины, применяемые при функциональной декомпозиции [28], может быть названа $(R - 1)$ -кратной разделительной симметрической декомпозицией.

9.5. Исследование функциональных возможностей симметрических модулей

Рассмотренные в предыдущем разделе методы применимы для модулей СМ n при любых n .

При каждом значении n может быть выполнено исследование функциональных возможностей модулей на основе подхода, изложенного в гл. 3, что позволяет получить решения, в том числе и от-

личные от решений, которые строятся рассмотренными выше методами.

Так как порождающая функция симметрического модуля не является симметрической функцией от числа переменных, равного числу входов модуля, то полное исследование его функциональных возможностей является весьма трудоемким уже при $n = 2$. Поэтому для рассматриваемого в качестве примера модуля СМ2 выполним частичное исследование его функциональных возможностей.

Порождающая функция этого модуля имеет вид:

$$F(a, b, c, d, e) = S_{c, d, e}^k(a, b) = !a!bc \vee (a \oplus b)d \vee abe = \\ = |0000 \ 1111 \ 0011 \ 0011 \ 0011 \ 0011 \ 0101 \ 0101|^T.$$

Выполним следующие замены переменных: $\Phi_1 = a$, $\Phi_2 = b$, $\Phi_3 = c$, $\Phi_4 = d$, $\Phi_5 = e$.

При $n - k = 0$ справедливо соотношение

$$F = S_{\Phi_3, \Phi_4, \Phi_5}^k(\Phi_1, \Phi_2) = \\ = !\Phi_1!\Phi_2\Phi_3 \vee (\Phi_1 \oplus \Phi_2)\Phi_4 \vee \Phi_1\Phi_2\Phi_5 = \\ = \text{MX}(0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, \\ 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5).$$

При $n - k = 1$ справедливо соотношение

$$F = S_{\Phi_3, \Phi_4, x_n}^k(\Phi_1, \Phi_2) = \\ = !\Phi_1!\Phi_2\Phi_3 \vee (\Phi_1 \oplus \Phi_2)\Phi_4 \vee \Phi_1\Phi_2x_n = \\ = \text{MX}(0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, x_n, x_n, x_n, x_n; \\ \Phi_1, \Phi_2, \Phi_3, \Phi_4).$$

Рассмотренное соотношение является симметрическим разложением БФУ по одной крайней правой входной переменной при выбранном образе декомпозиции.

При $n - k = 2$ справедливо соотношение

$$F = S_{\Phi_3, x_{n-1}, x_n}^k(\Phi_1, \Phi_2) = \\ = !\Phi_1!\Phi_2\Phi_3 \vee (\Phi_1 \oplus \Phi_2)x_{n-1} \vee \Phi_1\Phi_2x_n = \\ = \text{MX}(0, 1, x_{n-1}, x_{n-1}, x_{n-1}, x_{n-1}, x_n, x_n; \Phi_1, \Phi_2, \Phi_3).$$

Рассмотренное соотношение является симметрическим разложением БФУ по двум крайним правым входным переменным при выбранном образе декомпозиции.

При $n - k = 3$ справедливо соотношение

$$\begin{aligned} F &= S_{x_{n-2}, x_{n-1}, x_n}^{\kappa} (\Phi_1, \Phi_2) = \\ &= !\Phi_1 !\Phi_2 x_{n-2} \vee (\Phi_1 \oplus \Phi_2) x_{n-1} \vee \Phi_1 \Phi_2 x_n = \\ &= \text{MX}(x_{n-2}, x_{n-1}, x_{n-1}, x_n; \Phi_1, \Phi_2). \end{aligned}$$

Рассмотренное соотношение является симметрическим разложением БФУ по трем крайним правым входным переменным при выбранном образе декомпозиции.

При $n - k = 4$ справедливо соотношение

$$\begin{aligned} F &= S_{x_{n-2}, x_{n-1}, x_n}^{\kappa} (\Phi_1, x_{n-3}) = \\ &= !\Phi_1 !x_{n-3} x_{n-2} \vee (\Phi_1 \oplus x_{n-3}) x_{n-1} \vee \Phi_1 x_{n-3} x_n = \\ &= \text{MX}(!x_{n-3} x_{n-2} \vee x_{n-3} x_{n-1}, !x_{n-3} x_{n-1} \vee x_{n-3} x_n; \Phi). \end{aligned}$$

Рассмотренное соотношение является симметрическим разложением БФУ по четырем крайним правым входным переменным при выбранном образе декомпозиции.

Из изложенного, в частности, следует, что при $n - k = 1$ модуль СМ2 позволяет проводить разложение по переменной $\tilde{x}_n$ функций, столбцы значений которых содержат не более трех типов фрагментов длиной два: $\{0, 1, \tilde{x}_n\}$.

Пример 9.31. Построить симметрическую реализацию для мажоритарной функции f_1 из примера 9.1.

Максимальное значение $n - k$, при котором фрагменты БФУ и одного из образов декомпозиции совпадают, равно единице. Поэтому запишем уравнение

$$\begin{aligned} x_1 \# x_2 \# x_3 &= S_{\Phi_3, \Phi_4, x_3}^{\kappa} (\Phi_1, \Phi_2); \\ x_1 \# x_2 \# x_3 &= !\Phi_1 !\Phi_2 \Phi_3 \vee (\Phi_1 \oplus \Phi_2) \Phi_4 \vee \Phi_1 \Phi_2 x_3; \\ &= \text{MX}(0, x_3, x_3, 1; x_1 x_2) = \\ &= \text{MX}(0, 0, 1, 1, 0, 1, 0, 1, 0, 1, x_3, x_3, x_3, x_3; \\ &\quad \Phi_1, \Phi_2, \Phi_3, \Phi_4). \end{aligned}$$

Это уравнение имеет $6 \cdot 4^2 \cdot 6 = 576$ решений, среди которых выберем, например, следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 1, 1, 1; x_1, x_2) = x_1 \vee x_2; \\ \Phi_2 &= \text{MX}(1, 1, 1, 0; x_1, x_2) = !x_1 \vee !x_2; \\ \Phi_3 &= \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2; \\ \Phi_4 &= \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2. \end{aligned}$$

Таким образом,

$$f_1 = S_{x_2, x_2, x_3}^k (x_1 \vee x_2, !x_1 \vee !x_2).$$

Отметим, что полученная декомпозиция в отличие от предыдущих является неразделительной.

Покажем, что если изменить порядок переменных в ТИ порождающей функции $(a, b, c, d, e \rightarrow a, b, c, e, d)$, то найденная СР может быть упрощена. При этом $F(a, b, c, e, d) = |0000 \ 1111 \ 0101 \ 0101 \ 0101 \ 0101 \ 0011 \ 0011|^T$.

Выполним следующие замены переменных: $\Phi_1 = a, \Phi_2 = b, \Phi_3 = c, \Phi_4 = e, \Phi_5 = d$.

При $n - k = 0$ справедливо соотношение

$$\begin{aligned} F &= S_{\Phi_3, \Phi_5, \Phi_4}^k (\Phi_1, \Phi_2) = \\ &= !\Phi_1 !\Phi_2 \Phi_3 \vee (\Phi_1 \oplus \Phi_2) \Phi_5 \vee \Phi_1 \Phi_2 \Phi_4 = \\ &= \text{MX}(0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, \\ &\quad 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5). \end{aligned}$$

При $n - k = 1$ справедливо соотношение

$$\begin{aligned} F &= S_{\Phi_3, x_n, \Phi_4}^k (\Phi_1, \Phi_2) = \\ &= !\Phi_1 !\Phi_2 \Phi_3 \vee (\Phi_1 \oplus \Phi_2) x_n \vee \Phi_1 \Phi_2 \Phi_4 = \\ &= \text{MX}(0, 0, 1, 1, x_n, x_n, x_n, x_n, x_n, x_n, x_n, x_n, 0, 1, 0, 1; \\ &\quad \Phi_1, \Phi_2, \Phi_3, \Phi_4). \end{aligned}$$

Пример 9.32. Построить симметрическую реализацию для мажоритарной функции f_1 из примера 9.1.

Максимальное значение $n - k$, при котором фрагменты БФУ и одного из образов декомпозиции совпадают, равно единице. Поэтому запишем уравнение

$$\begin{aligned} x_1 \# x_2 \# x_3 &= S_{\Phi_3, x_3, \Phi_4}^k (\Phi_1, \Phi_2); \\ x_1 \# x_2 \# x_3 &= !\Phi_1 !\Phi_2 \Phi_3 \vee (\Phi_1 \oplus \Phi_2) x_3 \vee \Phi_1 \Phi_2 \Phi_4; \\ &\text{MX}(0, x_3, x_3, 1; x_1, x_2) = \\ &= \text{MX}(0, 0, 1, 1, x_3, x_3, x_3, x_3, x_3, x_3, x_3, x_3, 0, 1, 0, 1; \\ &\quad \Phi_1, \Phi_2, \Phi_3, \Phi_4). \end{aligned}$$

Это уравнение имеет $4 \cdot 8^2 \cdot 4 = 1024$ решения, среди которых выберем, например, следующее:

$$\begin{aligned} \Phi_1 &= \text{MX}(0, 0, 1, 1; x_1, x_2) = x_1; \\ \Phi_2 &= \text{MX}(0, 1, 0, 1; x_1, x_2) = x_2; \\ \Phi_3 &= \text{MX}(0, 0, 0, 0; x_1, x_2) = 0; \\ \Phi_4 &= \text{MX}(1, 1, 1, 1; x_1, x_2) = 1. \end{aligned}$$

Таким образом,

$$f_1 = S_{0, x_3, 1}^{\kappa}(x_1, x_2).$$

Это решение может быть получено и существенно проще, если учесть, что мультиплексор «4 в 1» при объединении его средних 2^n -входов является модулем СМ2.

Рассмотренная перестановка переменных не позволяет расширить класс функций, которые могут быть разложены по переменной $\tilde{x}_n$, по сравнению с исходным порядком расположения входных переменных в ТИ.

Рассмотрим следующую перестановку переменных:

$$a, b, c, d, e \rightarrow c, d, e, a, b.$$

При этом $F(c, d, e, a, b) = |0000\ 0001\ 0110\ 0111\ 1000\ 1001\ 1110\ 1111|^T$.

Выполним следующие замены переменных: $\Phi_1 = c, \Phi_2 = d, \Phi_3 = e, \Phi_4 = a, \Phi_5 = b$.

При $n - k = 0$ справедливо соотношение

$$\begin{aligned} F &= S_{\Phi_1, \Phi_2, \Phi_3}^{\kappa}(\Phi_4, \Phi_5) = \\ &= \Phi_1! \Phi_4! \Phi_5 \vee \Phi_2(\Phi_4 \oplus \Phi_5) \vee \Phi_3 \Phi_4 \Phi_5 = \\ &= \text{MX}(0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, \\ &\quad 1, 1, 1, 0, 1, 1, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5). \end{aligned}$$

При $n - k = 1$ справедливо соотношение

$$\begin{aligned} F &= S_{\Phi_1, \Phi_2, \Phi_3}^{\kappa}(\Phi_4, x_n) = \\ &= \Phi_1! \Phi_4! x_n \vee \Phi_2(\Phi_4 \oplus x_n) \vee \Phi_3 \Phi_4 x_n = \\ &= \text{MX}(0, 0, 0, x_n, x_n, !x_n, x_n, 1, !x_n, 0, !x_n, x_n, \\ &\quad 1, !x_n, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4). \end{aligned}$$

Это соотношение может применяться для определения симметрической реализации произвольной БФУ по переменной x_n .

Пример 9.33. Определить симметрическую реализацию для БФУ $f_4 = !x_1 x_2 \oplus !x_3!x_4$, рассмотренной в примере 9.28.

Запишем уравнение

$$\begin{aligned} !x_1 x_2 \oplus !x_3!x_4 &= S_{\Phi_1, \Phi_2, \Phi_3}^{\kappa}(\Phi_4, x_4); \\ !x_1 x_2 \oplus !x_3!x_4 &= \Phi_1! \Phi_4! x_4 \vee \Phi_2(\Phi_4 \oplus x_4) \vee \Phi_3 \Phi_4 x_4; \\ \text{MX}(!x_4, 0, x_4, 1, !x_4, 0, !x_4, 0; x_1, x_2, x_3) &= \\ = \text{MX}(0, 0, 0, x_4, x_4, !x_4, x_4, 1, !x_4, 0, !x_4, x_4, \\ &\quad 1, !x_4, 1, 1; \Phi_1, \Phi_2, \Phi_3, \Phi_4). \end{aligned}$$

Это уравнение имеет $4^3 \cdot 4 \cdot 4 \cdot 4^3 = 65\,536$ решений, среди которых выберем следующее:

$$\begin{aligned}\Phi_1 &= \text{MX}(1, 0, 0, 0, 1, 0, 0, 0; x_1, x_2, x_3) = !x_2!x_3; \\ \Phi_2 &= \text{MX}(0, 0, 0, 1, 0, 0, 1, 0; x_1, x_2, x_3) = (x_1 \oplus x_3)x_2; \\ \Phi_3 &= \text{MX}(0, 0, 1, 1, 0, 0, 0, 0; x_1, x_2, x_3) = !x_1x_2; \\ \Phi_4 &= \text{MX}(0, 0, 1, 1, 0, 0, 1, 1; x_1, x_2, x_3) = x_2.\end{aligned}$$

Таким образом,

$$f_4 = S_{!x_2!x_3, (x_1 \oplus x_3)x_2, !x_1x_2}^k(x_2, x_4).$$

Рассмотрим случай $n - k = 2$ при выбранном порядке переменных:

$$\begin{aligned}F &= S_{\Phi_1, \Phi_2, \Phi_3}^k(x_{n-1}, x_n) = \\ &= \Phi_1!x_{n-1}!x_n \vee \Phi_2(x_{n-1} \oplus x_n) \vee \Phi_3x_{n-1}x_n = \\ &= \text{MX}(0, x_{n-1}x_n, x_{n-1} \oplus x_n, x_{n-1} \vee x_n, !x_{n-1}!x_n, !(x_{n-1} \oplus x_n), \\ &\quad !x_{n-1} \vee !x_n, 1; \Phi_1, \Phi_2, \Phi_3).\end{aligned}$$

Пример 9.34. Определить симметрическую реализацию для БФУ, рассмотренной в предыдущем примере.

Составим уравнение

$$\begin{aligned}!x_1x_2 \oplus !x_3!x_4 &= S_{\Phi_1, \Phi_2, \Phi_3}^k(x_3, x_4); \\ !x_1x_2 \oplus !x_3!x_4 &= \Phi_1!x_3!x_4 \vee \Phi_2(x_3 \oplus x_4) \vee \Phi_3x_3x_4; \\ \text{MX}(!x_3!x_4, x_3 \vee x_4, !x_3!x_4, !x_3!x_4; x_1, x_2) &= \\ &= \text{MX}(0, x_3x_4, x_3 \oplus x_4, x_3 \vee x_4, !x_3!x_4, !(x_3 \oplus x_4), \\ &\quad !x_3 \vee !x_4, 1; \Phi_1, \Phi_2, \Phi_3).\end{aligned}$$

Это уравнение имеет единственное решение:

$$\begin{aligned}\Phi_1 &= \text{MX}(1, 0, 1, 1; x_1, x_2) = x_1 \vee !x_2; \\ \Phi_2 &= \text{MX}(0, 1, 0, 0; x_1, x_2) = !x_1x_2; \\ \Phi_3 &= \text{MX}(0, 1, 0, 0; x_1, x_2) = !x_1x_2.\end{aligned}$$

Таким образом,

$$f_4 = S_{x_1 \vee !x_2, !x_1x_2, !x_1x_2}^k(x_3, x_4).$$

Полученное решение входит в состав множества решений при $n - k = 1$.

Следовательно, функция f_4 реализована на модулях СМ4 (пример 9.6), СМ3 (пример 9.28), СМ2 (пример 9.34) и СМ1 (пример 9.29) при использовании различных информационных и настроечных алфавитов и различных методов реализации.

9.6. Реализация булевых функций схемами из симметрических модулей

В тех случаях, когда БФУ не может быть реализована одним симметрическим модулем с фиксированным числом информационных входов, возникает традиционная для логического проектирования [30] задача — реализовать произвольную БФУ n переменных схемой из модулей СМ k , где $k < n$.

Предлагаемый метод синтеза основан на возможности многократного применения изложенных в настоящей главе симметрического разложения по переменной и/или симметрической декомпозиции до тех пор, пока остаточные функции не будут зависеть от двух и менее переменных (при наличии констант и входных переменных) или от трех и менее переменных (при наличии констант, входных переменных и их инверсий).

При этом в силу универсальности модулей СМ k в классе произвольных функций k переменных ($k \leq 3$) каждая из остаточных функций может быть реализована одним модулем.

Пример 9.35. Реализовать функцию f_4 , рассмотренную в примере 9.6, схемой из модулей СМ при $k \leq 3$ при наличии констант, входных переменных и их инверсий.

1. Используя результаты примера 9.28, построим схему из модулей СМ3. Для этого построим таблицу настройки модулей, реализующих остаточные функции (табл. 9.10).

Т а б л и ц а 9.10

i	x_1	x_2	x_3	Φ_1	g_i^1	Φ_2	g_i^2	Φ_3	g_i^3
0	0	0	0	0	0	0	0	0	0
1	0	0	1	0		1		0	
	0	1	0	0	0	1	$\neg x_1$	1	x_2
	1	0	0	0		0		0	
2	0	1	1	1		1		1	
	1	0	1	0	$\neg x_1$	0	$\neg x_1$	1	x_3
	1	1	0	0		0		0	
3	1	1	1	1	1	0	0	0	0

Искомая схема приведена на рис. 9.18.

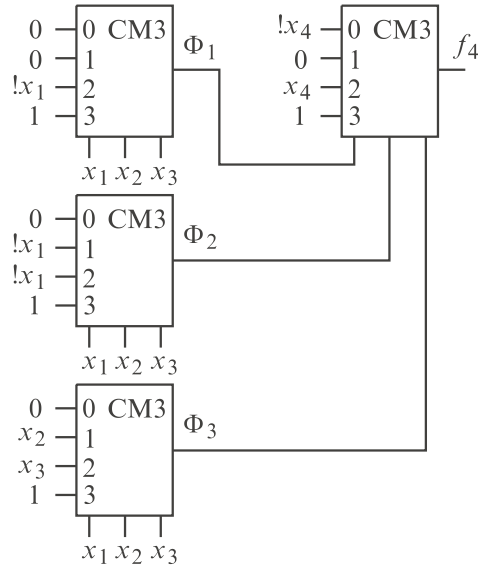


Рис. 9.18

2. Применяя результаты примера 9.34, построим схему рис. 9.19.

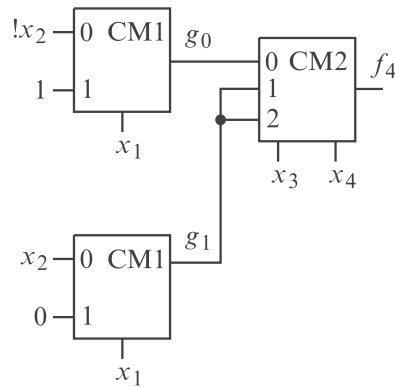


Рис. 9.19

Другая схема той же сложности может быть построена, если учесть, что $g_0 = !g_1$.

3. Используя результаты примера 9.29, построим схему (рис. 9.20).

Другая схема той же сложности может быть построена, если учесть, что $g_0 = !g_1$.

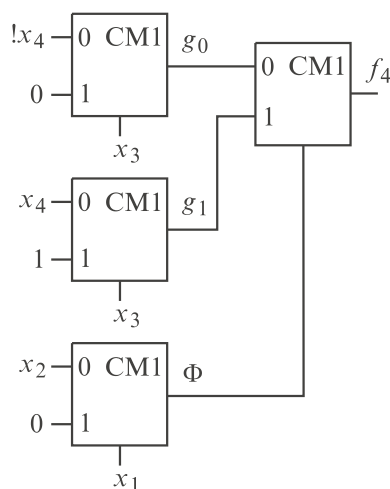


Рис. 9.20

Выводы

1. Рассмотрены два типа симметрических модулей (СМ), первый из которых известен и построен на основе принципа локального кодирования, а второй — на основе симметрического многополюсника Шеннона.

2. Предложены методы реализации симметрических функций в базисе СМ, более простые, чем известный метод.

3. Понятие «симметрическая реализация» распространяется на реализацию несимметрических функций, в том числе пороговых и произвольных. При этом в общем случае симметрические реализации могут быть построены, применяя либо рабочие числа, либо код настройки, причем код настройки является более универсальной характеристикой реализации. Форма описания симметрических реализаций булевых функций с помощью кода настройки является новой.

4. Для произвольной БФУ n переменных предложены конструктивные методы представления ее симметрической функцией, зависящей не более чем от $2^n - 1$ переменных, что несколько уточняет формулировку теоремы, приведенной в [9].

5. Показано, что повторные БФ могут иметь неповторные симметрические реализации, а неповторные БФ — повторные симметрические реализации, причем одна и та же БФ (БФУ) может иметь несколько симметрических реализаций, как повторных, так и неповторных.

6. Введены понятия «симметрическое разложение БФУ по крайним правым входным переменным» и «многократная разделитель-

ная симметрическая декомпозиция БФУ». Предложены методы построения этих разложений и декомпозиций.

7. Предложены методы синтеза комбинационных схем в базисе симметрических модулей.

Л и т е р а т у р а

1. Гаврилов М. А. Определение числа контактов в схемах релейно-контактных дешифраторов и их распределение по реле // Изв. АН СССР. Отделение техн. наук. 1945. № 12.
2. Поваров Г. Н. К изучению контактных схем упорядоченного типа // Проблемы передачи информации. 1959. Вып. 4.
3. Сагалович Ю. Л. Мера упорядоченности булевых функций // Проблемы передачи информации. 1961. Вып. 8.
4. Шеннон К. Работы по теории информации и кибернетике. М.: Изд-во иностр. литер., 1963.
5. Колдуэлл С. Логический синтез релейных устройств. М.: Изд-во иностр. литер., 1962.
6. Поваров Г. Н. К изучению симметрических булевых функций с точки зрения теории релейно-контактных схем // Докл. АН СССР. 1955. № 2.
7. Epstein G. Synthesis of electronic circuits for symmetric functions // IRE Trans. Electronic Computers. 1958. N 3.
8. Поваров Г. Н. Математико-логическое исследование синтеза контактных схем с одним входом и k выходами // Логические исследования. М.: Наука, 1959.
9. Kautz W. H. The realization of symmetric switching functions with linear-input logical elements // IRE Trans. Electronic Computers. 1961. N 10.
10. Миллер Р. Теория переключательных схем. Т.1. М.: Наука, 1970.
11. Borr R. C., Scidmore A. Transformation of switching functions to completely symmetric switching function // IEEE Trans. on Computers. 1968. № 6.
12. Das S., Sheng C. On detection total or partial symmetry of switching functions // IEEE Trans. on Computers. 1971. N 3.
13. Yau S., Tang J. Transformation of an arbitrary switching function to a totally symmetric function // IEEE Trans. on Computers. 1971. № 12.
14. Dalberg B. On symmetric functions with redundant variables wighed functions // IEEE Trans. on Computers. 1973. N 5.
15. Lee D., Hong S. An algorithm for transformation of an arbitrary switching function to a completely symmetric function // IEEE Trans. on Computers. 1976. N 11.
16. Лупанов О. Б. Об одном подходе к синтезу управляющих систем — принципе локального кодирования // Проблемы кибернетики. 1965. № 14.
17. Дулепов Е. Г. Базовый симметрический многополюсник. А. с. СССР № 264777 // Бюл. изобр. 1970. № 9.
18. Дулепов Е. Г. Многофункциональные симметрические схемы // Автоматика и телемеханика. 1970. № 5.
19. Дулепов Е. Г. Функциональные возможности универсальных симметрических логических элементов // Автоматика и вычисл. техника. 1974. № 4.
20. Preparata F. P., Muller D. E. Generation of near-optimal universal boolean functions // J. Comp. and System Sciences. 1970. N 2.
21. Артюхов В. Л., Конейкин Г. А., Шалыто А. А. Настраиваемые модули для управляющих логических устройств. Л.: Энергоиздат, 1981.
22. Горбатов В. А., Останков Б. Л., Фролов С. А. Регулярные структуры автоматного управления. М.: Машиностроение, 1980.
23. Бутаков Е. А. Методы синтеза релейных устройств из пороговых элементов. М.: Энергия, 1970.
24. Боголюбов И. Н., Овсиевич Б. Л., Розенблюм Л. Я. Пороговая и мажоритарная логика. Информационный листок. 1964. № 2 (29).

25. Кондратьев В. Н., Шалыто А. А. Реализация булевых функций одним линейным арифметическим полиномом с маскированием // Автоматика и телемеханика. 1996. № 1.
26. Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1283744 // Бюл. изобр. 1987. № 2.
27. Артюхов В. Л., Розенблюм Л. Я., Шалыто А. А. Логические возможности некоторых типов каскадных структур // Сети связи и дискретные устройства управления. М.: Наука, 1976.
28. Поспелов Д. А. Логические методы анализа и синтеза схем. М.: Энергия, 1974.
29. Фудзисава Т., Касами Т. Математика для радиоинженеров. М.: Радио и связь, 1984.
30. Гаврилов М. А., Девятков В. В., Пупырев Е. И. Логическое проектирование дискретных автоматов. М.: Наука, 1977.

Глава 10

Модули, универсальные в классе самодвойственных функций и в «близких» к ним классах

10.1. Обзор литературы

Одной из важнейших задач в теории булевых функций (БФУ) является их систематизация и классификация.

При систематизации БФУ определяются свойства, которые позволяют объединять функции в классы. При этом наиболее известными являются следующие классы БФУ: произвольные [1], линейные [1], монотонные [1], самодвойственные [1], симметрические [2], пороговые [3, 4], особенные [5], неповторные [6].

При классификации БФУ перечисляются типы функций, образующих класс. При этом к одному типу относят функции, инвариантные относительно выбранных преобразований. Наиболее известными (разд. 1.1) являются:

— P -классификация (инвариантность относительно перестановок входных переменных);

— N -классификация (инвариантность относительно инверсий входных переменных);

— PN -классификация (инвариантность относительно перестановок и инверсий входных переменных) [7];

— NPN -классификация (инвариантность относительно перестановок и инверсий входных переменных, а также инверсий функций) [8].

Для сокращения числа типов произвольных БФУ в [9] было предложено использовать линейные и аффинные преобразования.

Для пороговых функций была предложена SD -классификация (от английского словосочетания «self-dual» — самодвойственная) [8]. Преобразования, применяемые в этой классификации, подробно рассмотрены в [10]. Эта классификация была использована также и

для произвольных БФУ. При применении этой классификации определяются типы самодвойственных функций $n + 1$ переменных, которые порождают заданные функции n переменных.

В [11, 12] были предложены модули, которые путем настройки реализуют всех представителей самодвойственных функций четырех переменных, в свою очередь порождающих всех представителей PN -типов функций трех переменных.

Наиболее «крупной» для БФУ n переменных ($n \geq 4$) является самодвойственная классификация [13], которая в отличие от SD -классификации является групповой [14, 15]. При применении этой классификации определяют типы функций n переменных, при подстановке в которые самодвойственных функций удастся породить произвольные функции того же числа переменных. При этом возможны две разновидности такой классификации: без учета инвертирования функций и с учетом их инвертирования [16].

На основе этой классификации были построены модули, универсальные в классе всех БФУ четырех [17, 18] и n переменных [19—21].

Из изложенного следует, что при использовании как SD -, так и самодвойственной классификации весьма актуальной является разработка модулей, универсальных в классе самодвойственных функций n переменных.

Интерес представляют также модули, универсальные в классе самоантидвойственных функций, и модули, которые путем настройки реализуют одновременно две функции — произвольную БФУ и ее антидвойственную, что может рассматриваться так же как вычисление произвольной БФУ одновременно на двух противоположных наборах [22].

Указанные выше модули описываются в настоящей главе. В ней приведены также модули, которые путем настройки одновременно реализуют БФУ и ее двойственную. Показано, что модули, универсальные в указанных классах, имеют меньшее число входов по сравнению с соответствующими модулями, универсальными в классе произвольных функций. Это объясняется тем, что для функций указанных классов удастся определить такие их свойства, которые позволяют кодировать эти функции более просто (иметь более «короткие» коды настройки для соответствующих модулей), чем произвольные функции [23].

10.2. Реализация булевых функций мультиплексорами

В случае, когда в качестве алфавита настройки применяются константы 0 и 1, простейшим модулем, универсальным в классе произвольных функций, является мультиплексор (МХ) с n информационными и 2^n настроечными входами (рис. 10.1), называемый мультиплексором « 2^n в 1» [24].

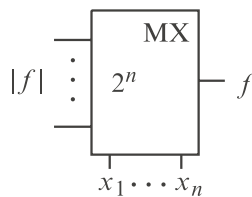


Рис. 10.1

При подаче на информационные входы модуля входных переменных $x_1, x_2, \dots, x_n$ в порядке, совпадающем с их порядком в левой части ТИ реализуемой функции f , его настройка осуществляется «подачей» столбца значений ТИ, обозначаемого символом $|f|$, на настроечные входы модуля.

Из изложенного следует, что число входов мультиплексора в этом случае определяется соотношением

$$M_1(n) = 2^n + n.$$

10.3. Реализация булевых функций, инверсных заданным

Функция f^n называется **инверсной** функции f , если

$$f^n(x_1, x_2, \dots, x_n) = \neg f(x_1, x_2, \dots, x_n).$$

Несмотря на то что мультиплексор « 2^n в 1» может реализовать как функцию f , так и функцию f^n , будем использовать схему (рис. 10.2), которая реализует функцию f^n по столбцу значений $|f|$.

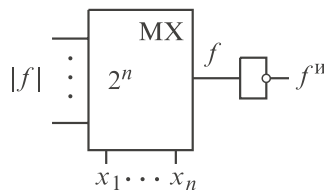


Рис. 10.2

10.4. Реализация булевых функций с инвертированным столбцом значений или его верхней, или нижней половиной

Известно [25], что

$$f = f \oplus 0; \quad f^n = f^n \oplus 1.$$

При этом схема на рис. 10.3 при подаче на вход z констант 0 и 1 позволяет по ТИ заданной функции f реализовать как функцию, так и ее инверсию.

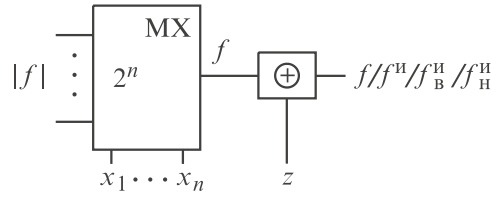


Рис. 10.3

В [26] были предложены два преобразования БФУ (разд. 1.7):

$$f = (!x_i!f(x_i=0) \vee x_i f(x_i=1)) \oplus !x_i;$$

$$f = (!x_i f(x_i=0) \vee x_i!f(x_i=1)) \oplus x_i.$$

Из этих соотношений следует, что

$$f_B^n = !x_1!f(x_1=0) \vee x_1 f(x_1=1) = f \oplus !x_1; \quad (10.1)$$

$$f_H^n = !x_1 f(x_1=0) \vee x_1!f(x_1=1) = f \oplus x_1.$$

Таким образом, схема на рис. 10.3 при подаче на вход z констант 0 и 1, переменной x_1 и ее инверсии $!x_1$ позволяет по столбцу значений $|f|$ реализовать заданную функцию f , ее инверсию f^n , а также функции f_B^n и f_H^n [27].

Из приведенных соотношений или разложений Рида следует, что если

$$f(x_i=1) = !f(x_i=0), \text{ то } f = f(x_i=1) \oplus !x_i;$$

$$f(x_i=0) = !f(x_i=1), \text{ то } f = f(x_i=0) \oplus x_i.$$

Эти функции могут быть реализованы схемой на рис.10.3, в которой МХ « 2^n в 1» заменен на МХ « 2^{n-1} в 1». Для таких функций выполняется соотношение

$$f(0, x_2, \dots, x_n) = f^n(1, x_2, \dots, x_n).$$

10.5. Реализация булевых функций, антидвойственных заданным

Функция $f^{ад}$ называется антидвойственной [13] функции f , если

$$f^{ад}(x_1, x_2, \dots, x_n) = f(!x_1, !x_2, \dots, !x_n).$$

Несмотря на то что мультиплексор « 2^n в 1» может реализовать как функцию f , так и функцию $f^{ад}$, в данном случае будем приме-

нять схему (рис. 10.4), которая реализует функцию $f^{\text{ад}}$ по столбцу значений $|f|$.

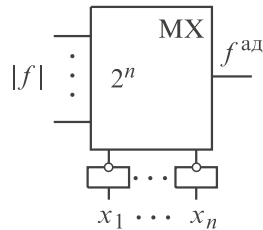


Рис. 10.4

10.6. Реализация булевых функций и им антидвойственных

Схема на рис. 10.5 при $z = 0$ реализует произвольную заданную функцию f , а при $z = 1$ — функцию $f^{\text{ад}}$.

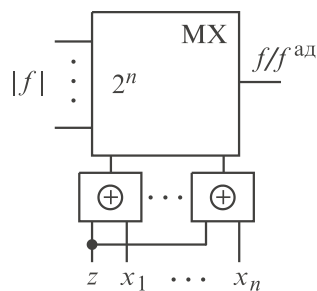


Рис. 10.5

10.7. Реализация самоантидвойственных (зеркальных) функций

Функция f называется самоантидвойственной [13], если

$$f(x_1, x_2, \dots, x_n) = f(!x_1, !x_2, \dots, !x_n).$$

Из этого соотношения следует, что функции этого класса имеют одинаковые значения на противоположных входных наборах. Будем обозначать такие функции символом $f^{\text{сад}}$. Функции этого класса могут быть названы также зеркальными, так как в каждой из них верхняя и нижняя половины столбца «зеркальны» друг другу.

Для функций этого класса справедливо соотношение

$$f(0, x_2, \dots, x_n) = f^{\text{ад}}(1, x_2, \dots, x_n). \quad (10.2)$$

Так как самоантидвойственные функции обладают специфическим свойством (соотношение 10.2), то появляется возможность реализации функций этого класса схемой, настраиваемой константами и содержащей мультиплексор меньшей размерности, чем он требуется для реализации произвольных функций того же числа переменных.

Для нахождения такой реализации рассмотрим первоначально в качестве примера схему, представленную на рис. 10.6.

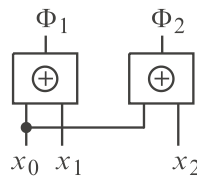


Рис. 10.6

Функционирование этой схемы описывает ТИ (табл. 10.1).

Т а б л и ц а 10.1

x_0	x_1	x_2	Φ_1	Φ_2
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	0
1	1	0	0	1
1	1	1	0	0

Из рассмотрения этой таблицы следует, что, во-первых, значения функций Φ_1 и Φ_2 располагаются в верхней половине ТИ в лексикографическом порядке, а, во-вторых, верхняя и нижняя половины каждой из этих функций «зеркальны» друг другу.

Таковыми же свойствами обладают ТИ для функций

$$\Phi_1 = x_0 \oplus x_1, \dots, \Phi_n = x_0 \oplus x_n.$$

Из изложенного следует, что схема на рис. 10.7 реализует заданную функцию $f^{\text{сад}}(x_0, x_1, \dots, x_n)$ при подаче на ее настроеч-

ные входы верхней половины столбца значений заданной функции $|f_{\text{в}}^{\text{сад}}|$.

Пример 10.1. Реализовать с помощью схемы на рис. 10.7 зеркальную функцию $f = !x_0x_1x_2 \vee x_0!x_1!x_2$, для которой $|f_{\text{в}}^{\text{сад}}| = |0001|^T$.

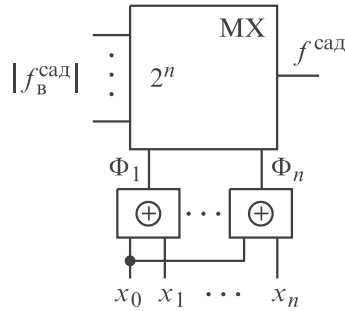


Рис. 10.7

Искомая схема приведена на рис. 10.8. Так как из рассмотрения этой схемы следует, что $f = \Phi_1\Phi_2$, то полученная схема может быть упрощена (рис. 10.9).

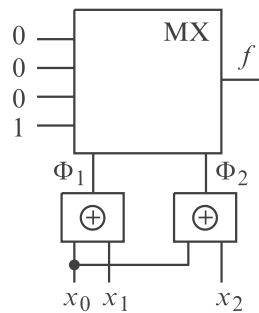


Рис. 10.8

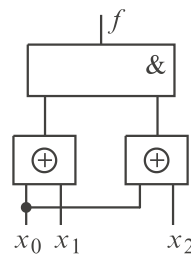


Рис. 10.9

Таким образом, в данном случае справедливо соотношение

$$!x_0x_1x_2 \vee x_0!x_1!x_2 = (x_0 \oplus x_1)(x_0 \oplus x_2).$$

Аналогично можно показать, что если в схеме на рис. 10.7 заменить переменную x_0 на ее инверсию $!x_0$, то новая схема (рис. 10.10) реализует заданную функцию $f^{\text{сад}}(x_0, x_1, \dots, x_n)$ при подаче на ее настроечные входы нижней половины столбца значений заданной функции $|f_{\text{н}}^{\text{сад}}|$.

Пример 10.2. Реализовать с помощью схемы на рис. 10.10 зеркальную функцию $f = !x_0!x_1!x_2 \vee x_0x_1x_2$, для которой $|f_{\text{н}}^{\text{сад}}| = |0001|^T$.

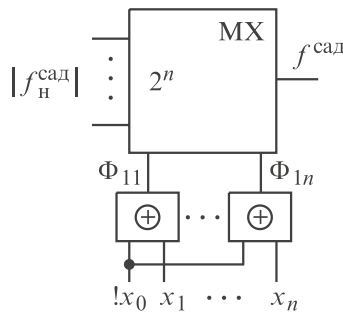


Рис. 10.10

Выполняя построение и преобразование, аналогичные приведенным в предыдущем примере, получим искомую схему (рис. 10.11),

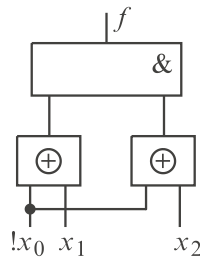


Рис. 10.11

для которой

$$!x_0 !x_1 !x_2 \vee x_0 x_1 x_2 = (!x_0 \oplus x_1) (!x_0 \oplus x_2).$$

Из изложенного следует, что при настройке константами модуль, универсальный в классе зеркальных функций n переменных, имеет число входов, определяемое соотношением

$$M_2(n) = 2^{n-1} + n.$$

Предложенный подход можно рассматривать и несколько с другой стороны: он позволяет по заданной БФУ n переменных $f(x_1, \dots, x_n)$ строить зеркальную функцию $n+1$ переменных $f^{\text{сад}}(x_0, x_1, \dots, x_n)$, в которой половина столбца значений совпадает со столбцом значений заданной функции.

В заключение раздела отметим, что из изложенного выше следует, что схема на рис. 10.5 реализует:

при $z = 0$ заданную функцию $f(x_1, \dots, x_n)$;

при $z = 1$ функцию $f^{\text{ад}}(x_1, \dots, x_n)$;

при $z = x_0$ функцию $f^{\text{сад}}(x_0, x_1, \dots, x_n)$, верхняя половина столбца значений которой совпадает со столбцом значений заданной функции f ;

при $z = !x_0$ функцию $f^{\text{сдл}}(x_0, x_1, \dots, x_n)$, нижняя половина столбца значений которой совпадает со столбцом значений заданной функции f .

10.8. Реализация булевых функций, двойственных заданным

Функция $f^{\text{д}}$ называется двойственной [4] функции f , если

$$f^{\text{д}}(x_1, x_2, \dots, x_n) = !f(!x_1, !x_2, \dots, !x_n).$$

Несмотря на то что мультиплексор « 2^n в 1» при настройке константами может реализовать как функцию f , так и функцию $f^{\text{д}}$ (если предварительно построить ее ТИ), в данном случае будем применять схему на рис. 10.12, построенную на основе схем на рис. 10.2, 10.4 и реализующую функцию $f^{\text{д}}$ по столбцу значений $|f|$.

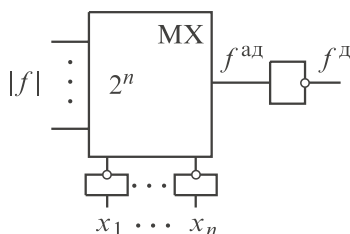


Рис. 10.12

10.9. Реализация булевых функций и их двойственных

Схема на рис. 10.13 при $z = 0$ реализует заданную функцию f , а при $z = 1$ — функцию $f^{\text{д}}$.

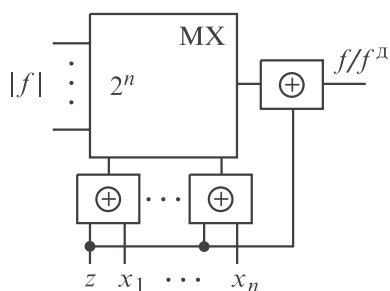


Рис. 10.13

10.10. Реализация самодвойственных функций

Функция f называется с а м о д в о й с т в е н н о й [13], если

$$f(x_1, x_2, \dots, x_n) = !f(!x_1, !x_2, \dots, !x_n).$$

Из этого соотношения следует, что функции этого класса принимают противоположные значения на противоположных входных наборах. Будем обозначать такие функции символом f^{cd} . Из приведенного соотношения следует, что для функций этого класса

$$!f(x_1, x_2, \dots, x_n) = f(!x_1, !x_2, \dots, !x_n).$$

Самодвойственными являются, например, такие функции, как

$$f_1 = x_0(x_1 \vee x_2) \vee x_1 x_2; \quad f_2 = x_0 \oplus x_1 \oplus x_2;$$

$$f_3 = x_0(x_1 \vee x_2 \vee x_3) \vee x_1 x_2 x_3.$$

Для функций этого класса ранг (число единиц в столбце значений) $R = 2^{n-1}$, а индекс (число пар противоположных входных наборов, на которых функция принимает единичные значения) $I = 0$.

Для самодвойственных функций справедливо равенство [8]

$$f(0, x_2, \dots, x_n) = f^1(1, x_2, \dots, x_n). \quad (10.3)$$

Сопоставим соотношение (10.3) для самодвойственных функций с соотношением (10.2) для самоантидвойственных функций. Из этих соотношений следует, что если в каждом из этих классов выбрать по одной функции, у которых верхние половины столбцов значений $f(0, x_2, \dots, x_n)$ совпадают, то их нижние половины будут инверсными.

Для инвертирования нижней половины столбца значений воспользуемся вторым равенством в (10.1).

Исходя из изложенного следует справедливость соотношений для функций рассматриваемых классов, у которых верхние половины столбцов значений совпадают:

$$f^{cd}(x_1, x_2, \dots, x_n) = f^{cad}(x_1, x_2, \dots, x_n) \oplus x_1; \quad (10.4)$$

$$f^{cad}(x_1, x_2, \dots, x_n) = f^{cd}(x_1, x_2, \dots, x_n) \oplus x_1. \quad (10.5)$$

Пример 10.3. Пусть заданы самодвойственная функция $f_0 = x_0(x_1 \vee x_2) \vee x_1 x_2$ и самоантидвойственная функция $f = !x_0 x_1 x_2 \vee x_0 !x_1 !x_2$. Определить взаимосвязь этих функций.

На основании соотношений (10.4) и (10.5) справедливы следующие равенства:

$$x_0(x_1 \vee x_2) \vee x_1 x_2 = (!x_0 x_1 x_2 \vee x_0 !x_1 !x_2) \oplus x_0;$$

$$!x_0 x_1 x_2 \vee x_0 !x_1 !x_2 = (x_0(x_1 \vee x_2) \vee x_1 x_2) \oplus x_0.$$

Несмотря на то, что каждая из функций $f^{\text{сд}}(x_1, x_2, \dots, x_n)$ и $f^{\text{сад}}(x_1, x_2, \dots, x_n)$ может быть реализована на мультиплексоре « 2^n в 1», приведем схемы (рис. 10.14, 10.15), реализующие соотношения (10.4) и (10.5).

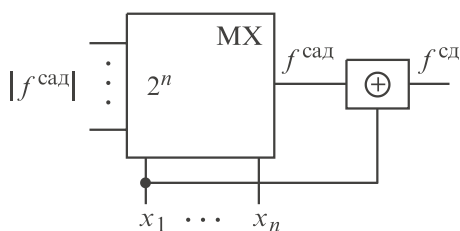


Рис. 10.14

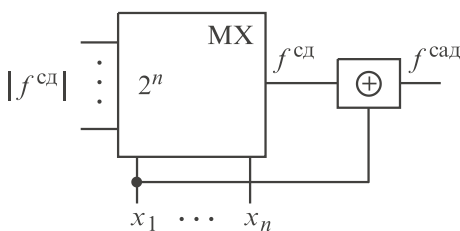


Рис. 10.15

Так как самодвойственные функции обладают специфическим свойством (соотношение 10.3), то появляется возможность реализации функций этого класса схемой, настраиваемой константами и содержащей мультиплексор меньшей размерности, чем он требуется для реализации произвольных функций того же числа переменных. Для этого запишем соотношение (10.4) для самодвойственных функций $n + 1$ переменных:

$$f^{\text{сд}}(x_0, x_1, x_2, \dots, x_n) = f^{\text{сад}}(x_0, x_1, x_2, \dots, x_n) \oplus x_0. \quad (10.6)$$

Учитывая тот факт, что для каждой самоантидвойственной функции может быть найдена самодвойственная функция, для которых справедливо равенство

$$|f^{\text{сад}}_в| = |f^{\text{сд}}_н|,$$

то на основании соотношения (10.6) и схемы на рис. 10.7 может быть построен модуль, универсальный в классе самодвойственных

функций $n + 1$ переменных, настраиваемый константами, который содержит мультиплексор « 2^n в 1» (рис. 10.16).

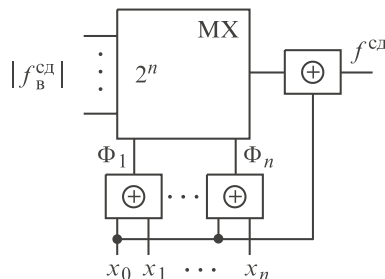


Рис. 10.16

Отметим также, что если в этом модуле заменить x_0 на $\neg x_0$, то новая схема (рис. 10.17) реализует заданную функцию $f^cd(x_0, x_1, x_2, \dots, x_n)$ при подаче на ее настроечные входы нижней половины столбца значений заданной функции $|f_H^cd|$.

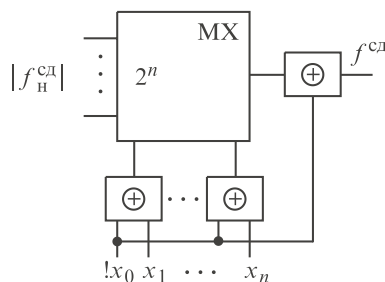


Рис. 10.17

Из изложенного следует, что при настройке константами модуль, универсальный в классе самодвойственных функций n переменных, имеет число входов, определяемое соотношением

$$M_3(n) = 2^{n-1} + n.$$

Пример 10.4. Реализовать с помощью схемы на рис. 10.16 функцию $f_1 = x_0(x_1 \vee x_2) \vee x_1x_2$, для которой $|f_{1B}^cd| = |0001|^T$.

При подаче на настроечные входы мультиплексора «4 в 1» столбца значений $|f_{1B}^cd|$ получим на его выходе функцию $g = \Phi_1\Phi_2$. Из изложенного следует, что заданная функция f может быть реализована схемой на рис. 10.18.

Из рассмотрения этой схемы следует, что

$$x_0(x_1 \vee x_2) \vee x_1x_2 = (x_0 \oplus x_1)(x_0 \oplus x_1) \oplus x_0.$$

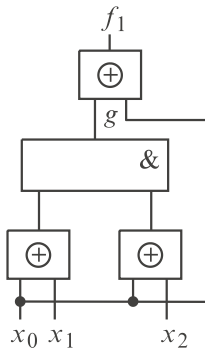


Рис. 10.18

Пример 10.5. Реализовать параллельный одноразрядный двоичный сумматор, функционирование которого описывается формулами f_1 и f_2 , схемой из двухвходовых элементов И и НЕРАВНОЗНАЧНОСТЬ.

Искомая схема, построенная с использованием схемы, приведенной в предыдущем примере, представлена на рис. 10.19 [28].

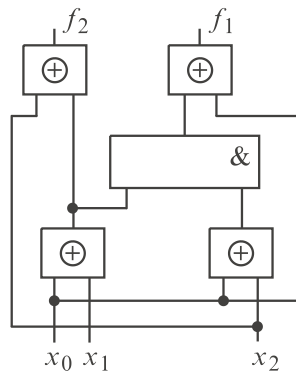


Рис. 10.19

Пример 10.6. Построить многофункциональный модуль, реализующий путем настройки систему функций, первая из которых является функцией переноса (f^n), а вторая — функцией заема (f^3) параллельного одноразрядного двоичного сумматора-вычитателя.

Рассмотрим несколько вариантов решения этой задачи.

1. Реализуем рассматриваемый модуль, настраиваемый константами, с помощью мультиплексора «4 в 1» и двухвходовых элементов И и ИЛИ.

Выберем для реализации формулы следующего вида:

$$f^n = !x_0 x_1 x_2 \vee x_0 (x_1 \vee x_2); \quad f^3 = !x_0 (x_1 \vee x_2) \vee x_0 x_1 x_2.$$

Полагая $g_1 = x_1 x_2$, $g_2 = x_1 \vee x_2$, объединим заданные формулы следующим образом: $F_1 = !z f^n \vee z f^3 = !z !x_0 g_1 \vee !z x_0 g_2 \vee z !x_0 g_2 \vee z x_0 g_1$ (рис. 10.20).

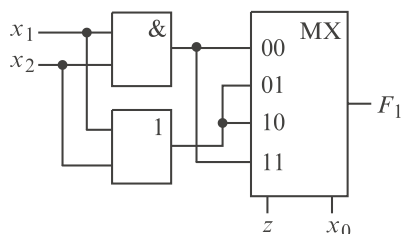


Рис. 10.20

Таким образом, при $z = 0$ $F^n = f_1$, а для $z = 1$ $F^3 = f_1$.

2. Реализуем этот модуль, настраиваемый отождествлениями, с помощью мультиплексора «8 в 1» и двухвходового элемента НЕ-РАВНОЗНАЧНОСТЬ.

Выберем для реализации формулы следующего вида:

$$f^n = (x_0 \vee x_1) x_2 \vee x_0 x_1; \quad f^3 = (x_0 \vee x_1) !x_2 \vee x_0 x_1.$$

Преобразуем первую из этих формул на основе соотношения (10.6) по переменной x_0 , а вторую — на основе того же соотношения по переменной x_1 :

$$f^n = (!x_0 x_1 x_2 \vee x_0 !x_1 !x_2) \oplus x_0;$$

$$f^3 = (!x_0 x_1 x_2 \vee x_0 !x_1 !x_2) \oplus x_1.$$

Таким образом, $F_2 = (!x_0 x_1 x_2 \vee x_0 !x_1 !x_2) \oplus z$. Следовательно, при $z = x_0$ $F_2 = f^n$, а при $z = x_1$ $F_2 = f^3$. Схема модуля, построенная на основе структуры, приведенной на рис. 10.3, в которой мультиплексор настроен константами, показана на рис. 10.21.

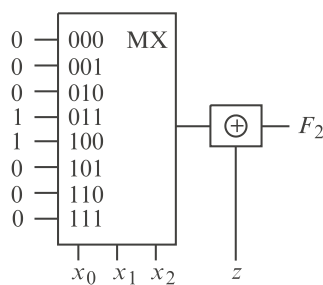


Рис. 10.21

3. Реализуем этот модуль, настраиваемый отождествлениями, с помощью двухвходовых элементов И и НЕРАВНОЗНАЧНОСТЬ.

Выберем для реализации формулы, приведенные в предыдущем пункте. Так как они являются самодвойственными, то

$$f^n = (x_0 \oplus x_1)(x_0 \oplus x_2) \oplus x_0; \quad f^3 = (x_0 \oplus x_1)(x_0 \oplus x_2) \oplus x_1.$$

Таким образом, $F_3 = (x_0 \oplus x_1)(x_0 \oplus x_2) \oplus z$. Следовательно, при $z = x_0$ $F_3 = f^n$, а при $z = x_1$ $F_3 = f^3$. Схема этого модуля приведена на рис. 10.22 [29].

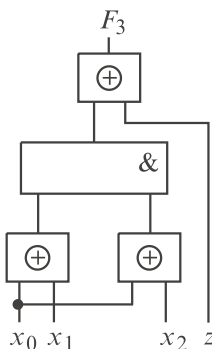


Рис. 10.22

4. Реализуем этот модуль, настраиваемый константами, с помощью двухвходовых элементов И, НЕРАВНОЗНАЧНОСТЬ и мультиплексора «2 в 1».

Искомая схема (рис. 10.23) построена на основе схемы на рис. 10.22.

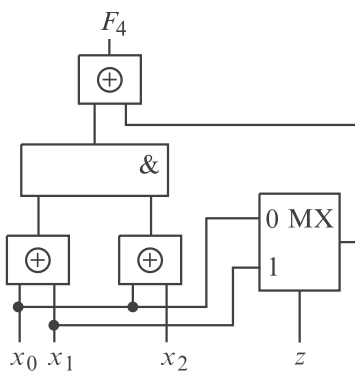


Рис. 10.23

5. Реализуем этот модуль, настраиваемый переменной и ее инверсией, с помощью двухвходовых элементов И и НЕРАВНОЗНАЧНОСТЬ.

Вновь выберем для реализации формулы следующего вида:

$$f^n = !x_0 x_1 x_2 \vee x_0 (x_1 \vee x_2); \quad f^3 = !x_0 (x_1 \vee x_2) \vee x_0 x_1 x_2.$$

Преобразуем эти формулы следующим образом:

$$f^n = (!x_0 x_1 x_2 \vee x_0 !x_1 !x_2) \oplus x_0 = g_0 \oplus x_0;$$

$$f^3 = (!x_0 !x_1 !x_2 \vee x_0 x_1 x_2) \oplus !x_0 = g_1 \oplus !x_0.$$

Так, столбцы значений зеркальных функций g_0 и g_1 состоят из одинаковых фрагментов половинной длины, расположенных в противоположном порядке, реализуем первую из них схемой на рис. 10.9, а вторую — схемой на рис. 10.11. При этом

$$\begin{aligned} f^n &= (x_0 \oplus x_1)(x_0 \oplus x_2) \oplus x_0; \\ f^3 &= (!x_0 \oplus x_1)(!x_0 \oplus x_2) \oplus !x_0. \end{aligned} \quad (10.7)$$

Таким образом, $F_5 = (z \oplus x_1)(z \oplus x_2) \oplus z$. Следовательно, при $z = x_0$ $F_5 = f^n$, а при $z = !x_0$ $F_5 = f^3$. Схема этого модуля приведена на рис. 10.24.

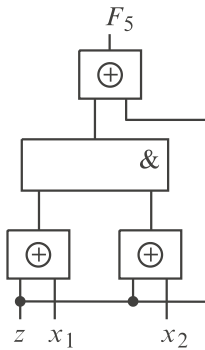


Рис. 10.24

6. Реализуем модуль, настраиваемый константами, с помощью двухвходовых элементов И и НЕРАВНОЗНАЧНОСТЬ.

Из соотношений 10.7 следует, что модуль может быть описан следующим образом: $g = z \oplus x_0$, $F_6 = (g \oplus x_1)(g \oplus x_2) \oplus g$. Таким образом, при $z = 0$ $F_6 = f^n$, а при $z = 1$ $F_6 = f^3$. Схема модуля приведена на рис. 10.25.

Если схему на рис. 10.25 дополнить двумя двухвходовыми элементами НЕРАВНОЗНАЧНОСТЬ, реализующими функцию «сумма» («разность»), описываемую соотношением $f = x_0 \oplus x_1 \oplus x_2$, то получается параллельный одноразрядный двоичный сумматор-вычитатель, содержащий шесть двухвходовых элементов НЕРАВНО-

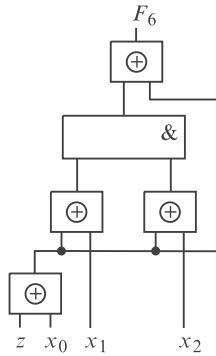


Рис. 10.25

ЗНАЧНОСТЬ и один двухвходовой элемент И [30] и который однороднее или проще известных схем [31].

Пример 10.7. Реализовать с помощью схемы на рис. 10.16 положительную самодвойственную пороговую функцию f четырех переменных, для которой $|f_{\text{в}}^{\text{сд}}| = |0000 \ 0001|^T$, и показать, что она порождает представителей всех пяти PN -типов пороговых функций трех переменных.

При этом используем схему на рис. 10.16 при $n = 3$, которая в результате настройки преобразуется в схему на рис. 10.26, описываемую формулой

$$F = (x_0 \oplus x_1)(x_0 \oplus x_2)(x_0 \oplus x_3) \oplus x_0.$$

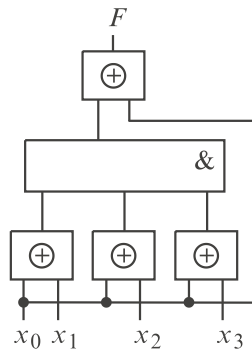


Рис. 10.26

Эта схема при $x_0 = 0$ реализует формулу $F = x_1 x_2 x_3$, при $x_0 = 1$ — $F = x_1 \vee x_2 \vee x_3$, при $x_1 = 0$ — $F = x_0(x_2 \vee x_3)$, при $x_1 = 1$ — $F = x_0 \vee x_2 x_3$, а при $x_3 = x_2$ — $F = x_0(x_1 \vee x_2) \vee x_1 x_2$. Та-

ким образом, построенная схема порождает представителей всех пяти *PN*-типов пороговых функций трех переменных.
 Этот модуль, построенный по формуле

$$f = x_0(x_1 \vee x_2 \vee x_3) \vee x_1x_2x_3,$$

описан в [32].

**10.11. Модуль, универсальный
 в рассмотренных классах булевых функций**

На рис. 10.27 приведена схема такого модуля.

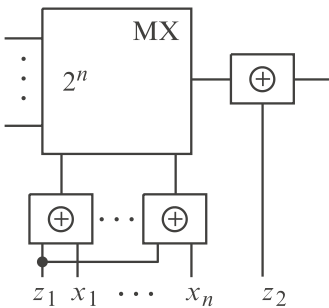


Рис. 10.27

Табл. 10.2 является таблицей настройки этого модуля на схемы, рассмотренные выше.

Т а б л и ц а 10.2

2^n -входы	z_1	z_2	Реализуемая схема	Реализуемые функции
$ f $	0	0	Рис. 10.1	f
$ f $	0	z	Рис. 10.3	$f/f''/f_v''/f_n''$
$ f $	1	0	Рис. 10.4	$f^{ад}$
$ f $	z	0	Рис. 10.5	$f/f^{ад}$
$ f_v^{сад} $	x_0	0	Рис. 10.7	$f^{сад}$
$ f $	z	z	Рис. 10.13	$f/f^д$
$ f_v^{сд} $	x_0	x_0	Рис. 10.16	$f^{сд}$

10.12. Одновременная реализация булевой функции и функции, ей антидвойственной (двойственной)

Предположим, что задана ТИ булевой функции $f(X)$, где $X = \{0, 1, \dots, 2^n - 1\}$. Тогда антидвойственная ей функция строится на основе соотношения

$$f^{\text{ад}}(X) = f(2^n - 1 - X).$$

В табл. 10.3 в качестве примера приведена функция f_4 и построенная по ней с помощью приведенного соотношения антидвойственная ей функция $f_4^{\text{ад}}$.

Т а б л и ц а 10.3

X	x_0	x_1	x_2	f_4	$f_4^{\text{ад}}$	f_5	f_6	$f_4^{\text{д}}$
0	0	0	0	0	1	0	1	0
1	0	0	1	0	1	0	1	0
2	0	1	0	0	1	0	1	0
3	0	1	1	0	0	0	0	1
4	1	0	0	0	0	0	0	1
5	1	0	1	1	0	0	1	1
6	1	1	0	1	0	0	1	1
7	1	1	1	1	0	0	1	1

Построим схему (рис. 10.28), основанную на объединении двух схем (рис. 10.7), предназначенных для реализации зеркальных функ-

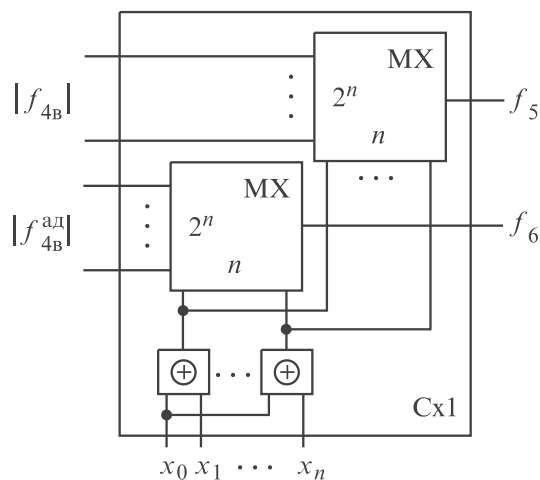


Рис. 10.28

ций от переменных $x_0, x_1, \dots, x_n$. На настроечные входы первой из этих схем подадим столбец $|f_{4в}|$, а на настроечные входы второй схемы — столбец $|f_{4в}^{ад}|$.

В табл. 10.3 приведены зеркальные функции f_5 и f_6 , реализуемые построенной схемой.

Дополним построенную схему коммутатором, состоящим из двух мультиплексоров «2 в 1», первый из которых строит заданную функцию f из верхней половины столбца значений функции f_5 и нижней половины столбца значений функции f_6 , а второй строит функцию $f^{ад}$ из верхней половины столбца значений функции f_6 и нижней половины столбца значений функции f_5 (рис. 10.29).

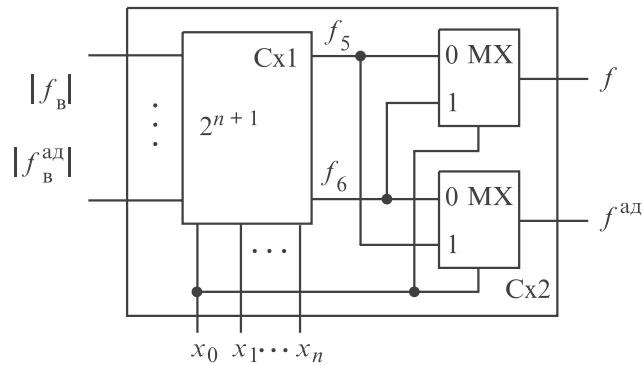


Рис. 10.29

Так как справедливо равенство

$$f^д = !f^{ад},$$

то схема на рис. 10.30 позволяет реализовать одновременно функцию f и двойственную ей функцию $f^д$.

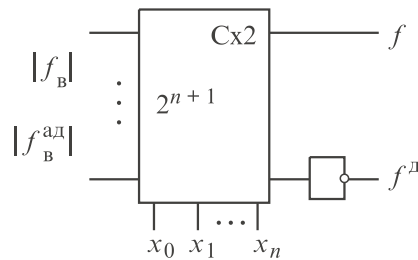


Рис. 10.30

Из изложенного следует, что при настройке константами модуль с двумя выходами, реализующий одновременно произвольную

БФУ n переменных и функцию, ей антидвойственную (двойственную), имеет число входов, определяемое соотношением

$$M_4(n) = 2^n + n.$$

На рис. 10.31 приведена схема, которая при $z = 0$ реализует функции f и $f^{\text{ад}}$, а при $z = 1$ — функции f и $f^{\text{д}}$ [33].

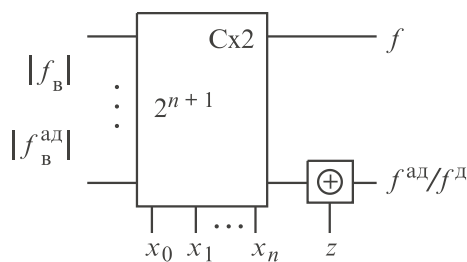


Рис. 10.31

Выводы

1. Предложены модули, универсальные в классе самодвойственных функций n переменных.
2. Предложены модули, универсальные в классе самоантидвойственных функций n переменных.
3. Показано, что описанные модули имеют меньшее число входов по сравнению с соответствующими модулями, универсальными в классе произвольных БФУ n переменных, так как функции, которые они объединяют, удастся «локально закодировать».
4. Предложены модули, реализующие путем настройки одновременно две функции — произвольную БФУ и функцию, ей антидвойственную, что может рассматриваться так же как вычисление произвольной БФУ одновременно на двух противоположных наборах.
5. Предложены модули, которые реализуют путем настройки одновременно БФУ и функцию, ей двойственную.

Литература

1. Яблонский С. В., Гаврилов Г. П., Кудрявцев В. Б. Функции алгебры логики и классы Поста. М.: Наука, 1966.
2. Шеннон К. Э. Работы по теории информации и кибернетике. М.: Изд-во иностр. литер., 1963.
3. Боголюбов И. Н., Овсевич Б. Л., Розенблом Л. Я. Синтез схем из пороговых и мажоритарных элементов // Сети передачи информации и их автоматизация. М.: Наука, 1965.
4. Бутаков Е. А. Методы синтеза релейных устройств из пороговых элементов. М.: Энергия, 1970.

5. Поваров Г. Н. К изучению контактных схем упорядоченного типа // Проблемы передачи информации. 1959. № 4.
6. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Настраиваемые модули для управляющих устройств. Л.: Энергоиздат, 1981.
7. Синтез электронных вычислительных и управляющих схем. М.: Изд-во иностр. литер., 1954.
8. Goto E., Takahasi H. Some theorems useful in threshold logic for enumerating boolean functions // Information Proceedings. IFIP Congress 62. Amsterdam, 1963.
9. Harrison M. A. On the classification of boolean functions by the general linear and affine groups // J. Soc. Indust. Appl. Math. 1964. N 2.
10. Дертоуэс М. Пороговая логика. М.: Мир, 1967.
11. Сычев А. В., Кириченко Н. В., Калмыков В. А. Многофункциональный логический модуль. А. с. СССР № 500525 // Бюл. изобр. 1976. № 3.
12. Попов В. А., Скибенко И. Т., Василенко А. С. и др. Логическая ячейка. А. с. СССР № 520585 // Бюл. изобр. 1976. № 25.
13. Розенблум Л. Я. Самодвойственная классификация и некоторые классы булевых функций // Автоматика и вычисл. техника. 1971. № 2.
14. Страздинь И. Э. Группа самодвойственных преобразований булевых функций // Теория дискретных автоматов. Рига: Зинатне, 1967.
15. Страздинь И. Э. Группы автоморфизмов алгебры булевых функций и классы Поста // Труды Рижского алгебраического семинара. Рига: Изд-во ЛатвГУ, 1969.
16. Страздинь И. Э. Самодвойственные типы булевых функций до пяти переменных // Автоматика и вычисл. техника. 1969. № 4.
17. Лысенко Э. В., Попов В. А., Скибенко И. Т. и др. Логическое устройство. А. с. СССР № 657432 // Бюл. изобр. 1979. № 14.
18. Дергачев В. А., Артеменко М. Н., Балалаев В. А. и др. Логический модуль. А. с. СССР № 1242929 // Бюл. изобр. 1986. № 25.
19. Лысенко Э. В., Попов В. А., Скибенко И. Т. и др. Универсальный логический модуль. А. с. СССР № 813410 // Бюл. изобр. 1981. № 10.
20. Дергачев В. А., Артеменко М. Н., Балалаев В. А. и др. Универсальный логический модуль. А. с. СССР № 1242931 // Бюл. изобр. 1986. № 25.
21. Дергачев В. А., Губка С. А., Балалаев В. А. и др. Универсальный логический модуль. А. с. СССР № 1290289 // Бюл. изобр. 1987. № 6.
22. Лупанов О. Б. Об одном подходе к синтезу управляющих систем — принципе локального кодирования // Проблемы кибернетики. М.: Наука, 1965. Вып. 14.
23. Лупанов О. Б. Асимптотические оценки сложности управляющих систем. М.: Изд-во МГУ, 1984.
24. Вау С., Танг С. Универсальные логические модули и их применение // Экспресс-информация. Вычисл. техника. 1970. № 27.
25. Поспелов Д. А. Логические методы анализа и синтеза схем. М.: Энергия, 1974.
26. Артюхов В. Л., Шалыто А. А. Судовые управляющие логические системы. Л.: ИПК СП, 1983.
27. Артюхов В. Л., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1149244 // Бюл. изобр. 1985. № 13.
28. Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1290290 // Бюл. изобр. 1987. № 6.
29. Шалыто А. А. Программная реализация алгоритмов логического управления судовыми системами. Л.: ИПК СП, 1989.
30. Шалыто А. А. Комбинационный двоичный сумматор-вычитатель. А. с. СССР № 1264166 // Бюл. изобр. 1986. № 38.
31. Мищенко В. А., Козюминский В. Д., Семашко А. Н. Многофункциональные автоматы и элементная база. М.: Радио и связь, 1981.
32. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. и др. Многофункциональный логический модуль. А. с. СССР № 813787 // Бюл. изобр. 1981. № 10.
33. Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1283744 // Бюл. изобр. 1987. № 2.

Глава 11

Модули, универсальные в классе всех булевых функций

Известно большое число работ [1—14], посвященных построению модулей, универсальных в классе всех булевых функций (БФУ) n переменных, называемых универсальными логическими модулями n переменных и обозначаемых УЛМ $_n$.

При этом наибольшее распространение на практике получили модули, в которых информационные и управляющие входы разделены.

В [1, 7] описаны универсальные логические модули n переменных без парафазных входных переменных, настройка которых осуществляется за счет подачи на управляющие входы констант 0 и 1.

В [1, 7] описаны также универсальные логические модули n переменных с одной парафазной входной переменной, настройка которых осуществляется за счет подачи на управляющие входы констант 0, 1, переменной x_n и ее инверсии $!x_n$. Это позволяет сократить число входов модулей по сравнению с предыдущим классом модулей.

Дальнейшее сокращение числа входов достигается при использовании универсальных логических модулей n переменных с n парафазными входными переменными, настройка которых осуществляется за счет подачи на управляющие входы констант 0, 1, переменных $x_1, x_2, \dots, x_n$ и их инверсий $!x_1, !x_2, \dots, !x_n$ [2, 7].

Из изложенного следует, что в настоящее время в теории универсальных логических модулей имеется пробел: до сих пор не рассматривались универсальные логические модули n переменных с m парафазными входами ($n > m$).

При определенной структуре модули этого класса являются обобщением трех классов модулей, перечисленных выше: первого — при $m = 0$, второго — при $m = 1$ и третьего — при $n = m$.

Материал настоящей главы, посвященный исследованию указанного выше нового класса модулей, призван восполнить указанный пробел.

11.1. Универсальные логические модули n переменных (УЛМ n) без парафазных входных переменных

Модулями этого класса являются мультиплексоры «2 в 1», рассматриваемые как одна компонента или композиция мультиплексоров меньшего числа переменных. Число входов таких модулей удовлетворяет соотношению

$$M_1(n) = n + 2^n.$$

Значения $M_1(n)$ при $n = 2—7$ приведены в столбце с $m = 0$ табл. 11.1. Для модулей этого типа будем применять обозначение УЛМ $n, 0$.

Т а б л и ц а 11.1

n	m							
	0	1	2	3	4	5	6	7
2	6	3	3	—	—	—	—	—
3	11	6	6	6	—	—	—	—
4	20	11	11	10	10	—	—	—
5	37	20	20	17	17	16	—	—
6	70	37	37	30	30	28	28	—
7	135	70	70	55	55	51	51	39

11.2. Модули УЛМ n с одной парафазной входной переменной

Модулями этого класса являются мультиплексоры « 2^{n-1} в 1», рассматриваемые как одна компонента или композиция мультиплексоров меньшего числа переменных. Число входов таких модулей удовлетворяет соотношению

$$M_2(n) = n + 2^{n-1} - 1.$$

Значения $M_2(n)$ при $n = 2—7$ приведены в столбце с $m = 1$ табл. 11.1. Для модулей этого типа будем использовать обозначение УЛМ $n, 1$.

На рис. 11.1 приведен модуль УЛМ3,1, реализованный мультиплексором «4 в 1», а на рис. 11.2 этого модуль построен как композиция трех мультиплексоров «2 в 1».

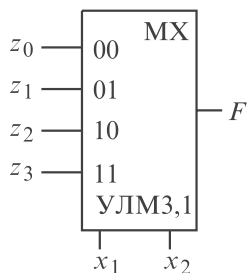


Рис. 11.1

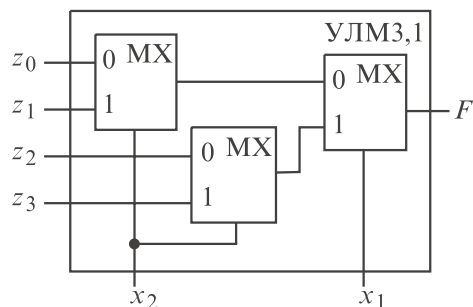


Рис. 11.2

В этих схемах реализация произвольной БФУ трех переменных — x_1, x_2, x_3 осуществляется за счет подачи на информационные входы переменных x_1, x_2 , а на управляющие входы $z_0, \dots, z_3$ — констант 0, 1, переменной x_3 и ее инверсии $\neg x_3$.

На рис. 11.3 приведен модуль УЛМ4,1, состоящий из двух мультиплексоров «4 в 1» и одного мультиплексора «2 в 1».

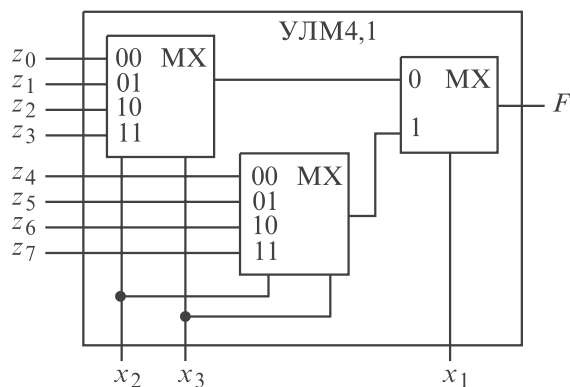


Рис. 11.3

11.3. Модули УЛМ n с n парафазными входными переменными (УЛМ n, n)

Структура модуля n переменных, настраиваемого за счет подачи символов 0, 1, $x_i, \neg x_i$ ($i = 1, \dots, n$) на управляющие входы $z_1, z_2, \dots, z_i$, приведена на рис. 11.4. Для модулей этого класса будем применять обозначение УЛМ n, n .

Этот модуль содержит блоки (Б), каждый из которых реализует подмножество конъюнкций n переменных. Конъюнкции в каждом подмножестве удовлетворяют условию избирательности [2]: с по-

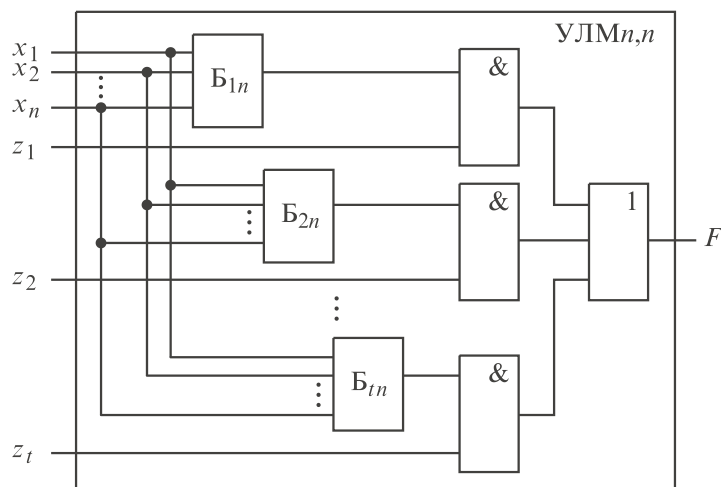


Рис. 11.4

мощью управляющего входа (связанного с блоком) за счет подачи на него символов 0, 1, x_i , $\neg x_i$ могут быть реализованы любые комбинации (дизъюнкции) конъюнкций, образующих это подмножество, а также отсутствие конъюнкций.

Число конъюнкций, входящих в блок, должно удовлетворять неравенству

$$s \leq [1 + \log(n+1)].$$

Число настроечных входов модуля определяется соотношением

$$t(n) = \left\lceil \frac{2^n}{[1 + \log(n+1)]} \right\rceil.$$

При этом число входов модуля при $n \geq 3$ определяется следующим образом:

$$M_3(n) = n + t = n + \left\lceil \frac{2^n}{[1 + \log(n+1)]} \right\rceil.$$

При $n = 2$ в качестве модуля УЛМ2,2 может применяться мультиплексор «2 в 1», структура которого описывается формулой $F = \neg x_1 z_0 \vee x_1 z_1$.

Значения $M_3(n)$ при $n = 2$ —7 приведены в крайней правой диагонали значений табл. 11.1.

При фиксированном n блоки, входящие в состав модуля, могут формироваться различными способами. Табл. 11.2 описывает четы-

ре варианта блоков модулей УЛМ3,3 (рис. 11.5). В этой таблице $a, b = \{0, 1\}$.

Т а б л и ц а 11.2

1			2			3			4		
x_1	x_2	x_3	x_1	x_2	x_3	x_1	x_2	x_3	x_1	x_2	x_3
a	0	1	a	0	0	a	0	0	a	0	0
a	1	0	$!a$	1	0	$!a$	0	1	$!a$	0	1
$!a$	1	1	a	1	1	a	1	1	$!a$	1	0
b	0	0	b	0	1	b	1	0	b	1	1

Выбирая для определенности третий вариант, перечислим входные наборы, образующие блоки модуля УЛМ3, 3 :

$$B_{13} = \begin{Bmatrix} 000 \\ 101 \\ 011 \end{Bmatrix}; B_{23} = \begin{Bmatrix} 100 \\ 001 \\ 111 \end{Bmatrix}; B_{33} = \begin{Bmatrix} 010 \\ 100 \end{Bmatrix}.$$

Из сравнения модуля УЛМ3,1 (рис. 11.2) и модуля УЛМ3,3 (рис. 11.5) следует, что первый из них обладает меньшей элементной сложностью при одинаковом числе входов.

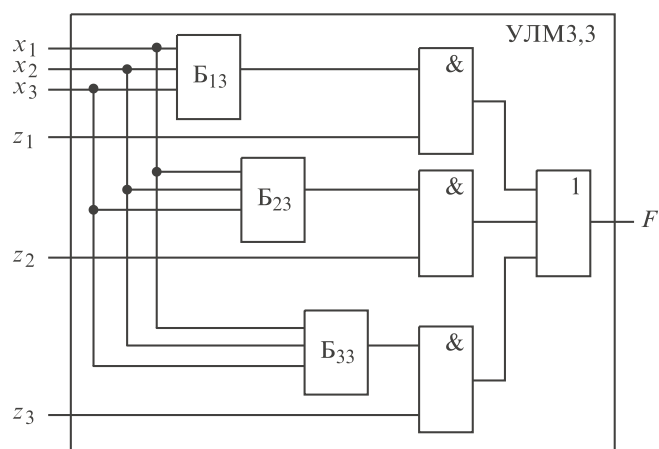


Рис. 11.5

При $n > 3$ модули УЛМ n,n имеют меньшее число входов. Кроме того, как будет показано в следующем разделе, блоки модулей УЛМ t,t могут применяться при построении модулей УЛМ n,n , где $n > t$.

При $n = 4, 5$ блоки могут быть сформированы на основе табл. 11.3 [2, 14]. В этой таблице $c, d = \{0, 1\}$.

Т а б л и ц а 11.3

$n = 4$				$n = 5$				
x_1	x_2	x_3	x_4	x_1	x_2	x_3	x_4	x_5
a	0	0	b	0	0	0	b	c
$\neg a$	0	1	b	$\neg a$	0	1	b	c
a	1	1	b	a	1	1	b	c
0	1	0	b	d	1	0	0	0
1	1	0	b	$\neg d$	1	0	0	1
				d	1	0	1	1
				0	1	0	1	0
				1	1	0	1	0

Перечислим входные наборы, входящие в соответствующие блоки. Для модуля УЛМ4, 4:

$$B_{14} = \begin{Bmatrix} 0000 \\ 1010 \\ 0110 \end{Bmatrix}; B_{24} = \begin{Bmatrix} 0001 \\ 1011 \\ 0111 \end{Bmatrix}; B_{34} = \begin{Bmatrix} 1000 \\ 0010 \\ 1110 \end{Bmatrix};$$

$$B_{44} = \begin{Bmatrix} 1001 \\ 0011 \\ 1111 \end{Bmatrix}; B_{54} = \begin{Bmatrix} 0100 \\ 1100 \end{Bmatrix}; B_{64} = \begin{Bmatrix} 0101 \\ 1101 \end{Bmatrix}.$$

Для модуля УЛМ5, 5:

$$B_{15} = \begin{Bmatrix} 00000 \\ 10100 \\ 01100 \end{Bmatrix}; B_{25} = \begin{Bmatrix} 00001 \\ 10101 \\ 01101 \end{Bmatrix}; B_{35} = \begin{Bmatrix} 00010 \\ 10110 \\ 01110 \end{Bmatrix}; B_{45} = \begin{Bmatrix} 00011 \\ 10111 \\ 01111 \end{Bmatrix};$$

$$B_{55} = \begin{Bmatrix} 10000 \\ 00100 \\ 11100 \end{Bmatrix}; B_{65} = \begin{Bmatrix} 10001 \\ 00101 \\ 11101 \end{Bmatrix}; B_{75} = \begin{Bmatrix} 10010 \\ 00110 \\ 11110 \end{Bmatrix}; B_{85} = \begin{Bmatrix} 10011 \\ 00111 \\ 11111 \end{Bmatrix};$$

$$B_{95} = \begin{Bmatrix} 01000 \\ 11001 \\ 01011 \end{Bmatrix}; B_{105} = \begin{Bmatrix} 11000 \\ 01001 \\ 11011 \end{Bmatrix}; B_{115} = \begin{Bmatrix} 01010 \\ 11010 \end{Bmatrix}.$$

Модуль УЛМ3, 3, в котором минимизируется элементная сложность, описан в [15].

11.4. Использование блоков модулей УЛМ t, t при построении модулей УЛМ n, n

Эту задачу рассмотрим на примере построения модуля УЛМ $5,5$ с применением блоков модуля УЛМ $3,3$.

Анализ входных наборов, образующих блоки модуля УЛМ $5,5$ и приведенных в предыдущем разделе, показывает, что первые че-

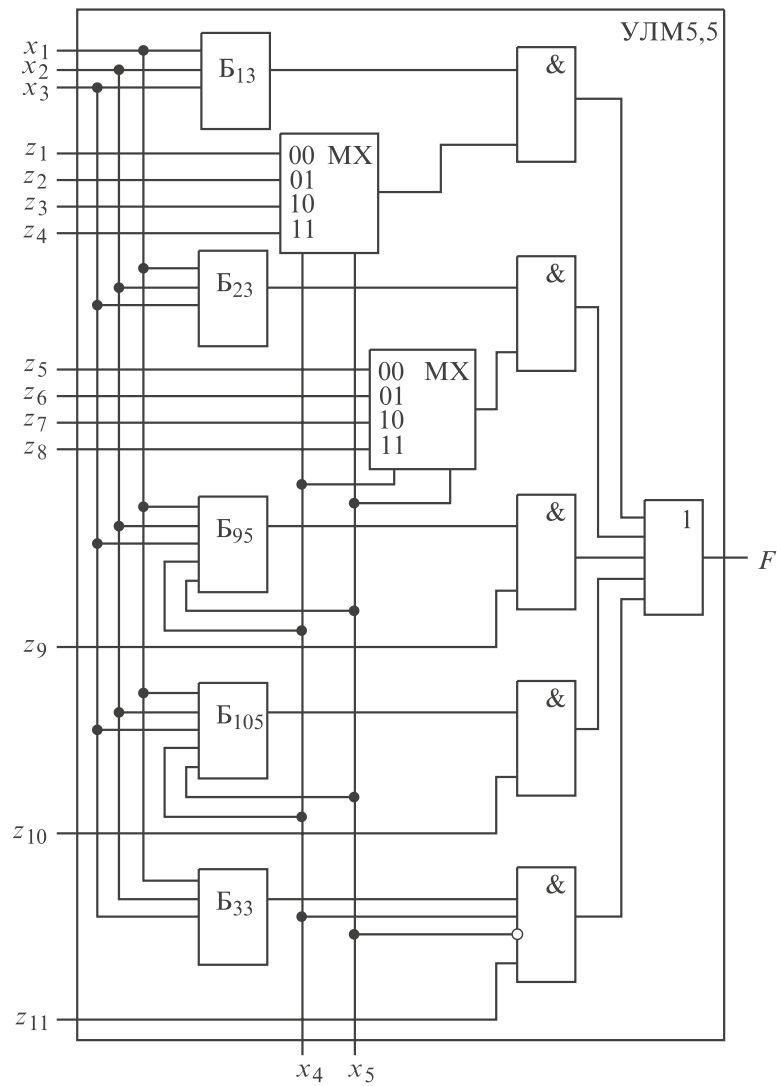


Рис. 11.6

тыре блока ($B_{15}, \dots, B_{45}$) могут быть построены на базе блока B_{13} модуля УЛМ3,3, блоки с пятого (B_{55}) по восьмой (B_{85}) — на базе блока B_{23} модуля УЛМ3,3, девятый и десятый блоки (B_{95}, B_{105}) должны использоваться в «естественном» виде, а одиннадцатый блок B_{115} может быть реализован на базе блока B_{33} модуля УЛМ3,3.

На рис. 11.6 приведен модуль УЛМ5,5, имеющий меньшую элементную сложность (при том же числе входов) по сравнению с модулем УЛМ5,5 с канонической структурой (рис. 11.4).

При этом на управляющие входы $z_1, \dots, z_8, z_{11}$ в парафазной форме должны подаваться только переменные x_1, x_2, x_3 , в то время как переменные x_4, x_5 могут быть однофазными. На входах z_9 и z_{10} любая из входных переменных (x_1, x_2, x_3, x_4, x_5) может потребоваться в парафазной форме.

Для определения настройки модуля заданная таблица истинности должна быть преобразована так, чтобы входные наборы в ней располагались в порядке их перечисления по блокам, описанным в разд. 11.3: $B_{15}, B_{25}, \dots, B_{115}$.

11.5. Модули УЛМ n, m первого типа

Будем использовать в этом случае обозначение УЛМ $n, m, 1$.

На рис.11.7 приведена схема модуля УЛМ $n, m, 1$, построенного из 2^{n-m} модулей УЛМ m, m и одного мультиплексора « 2^{n-m} в 1». На управляющие входы Z предлагаемого модуля могут подаваться константы 0, 1 и парафазные входные переменные $x_{n-m+1}, \dots, x_n$.

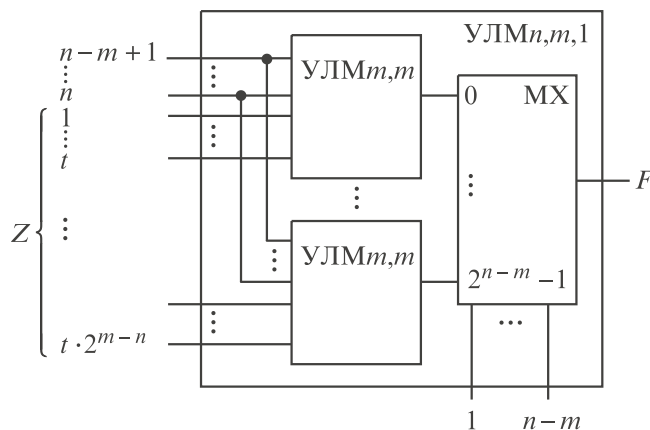


Рис. 11.7

Число входов в таком модуле при $m = 0, 3, 4, \dots, n$ определяется соотношением

$$M_4^m(n) = n + 2^{n-m} \left\lceil \frac{2^m}{[1 + \log(m+1)]} \right\rceil,$$

так как каждый из 2^{n-m} модулей УЛМ m, m , входящих в модуль УЛМ n, m первого типа, имеет по $t(n)$ управляющих входов.

При $m = 1$ предлагаемый модуль совпадает с единым модулем УЛМ $n, 1$, а при $m = 2$ — с композиционным модулем УЛМ $n, 1$.

Значения M_4 при различных значениях n и m приведены в табл. 11.1. На рис. 11.8 приведен модуль УЛМ $4,3,1$ с 10 входами, построенный из двух шестивходовых модулей УЛМ $3,3$ и одного мультиплексора «2 в 1».

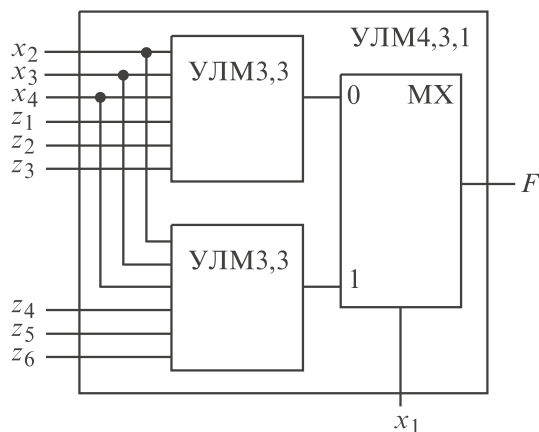


Рис. 11.8

При этом обратим внимание на тот факт, что модуль УЛМ $4,1$, построенный из двух шестивходовых модулей УЛМ $3,1$ и одного мультиплексора «2 в 1» (рис. 11.3), имеет 11 входов.

Из приведенного примера следует, что модули УЛМ $3,3$ и УЛМ $3,1$, имеющие одинаковое число входов и одинаковые функциональные возможности, обладают различной эффективностью при построении модуля УЛМ 4 на их основе. Меньшее число входов модуля УЛМ $4,3,1$ по сравнению с модулем УЛМ $4,1$ определяется большим числом парафазных переменных, подаваемых на его управляющие входы.

Из сравнения модулей УЛМ $4,3$ и УЛМ $4,4$ следует, что они имеют одинаковое число входов, а первый модуль имеет меньшую элементную сложность и требует использования однофазной переменной x_1 .

Из табл. 11.1 следует, что аналогичные выводы можно сделать и из сравнения модулей УЛМ6,5,1 и УЛМ6,6.

При других значениях n и m промежуточное число парафазных входных переменных приводит к промежуточному числу входов модулей:

$$M_3(n) < M_4^m(n) < M_2(n).$$

Пример 11.1. Определить настройку модуля УЛМ4,3,1 при реализации БФУ, заданной табл. 11.4.

Перестроим табл. 11.4 в соответствии с расположением конъюнкций в блоках модуля УЛМ3,3 для переменных x_2, x_3, x_4 (табл. 11.5).

Т а б л и ц а 11.4

x_1	x_2	x_3	x_4	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Т а б л и ц а 11.5

j	x_1	x_2	x_3	x_4	f	z_j
1	0	0	0	0	0	x_2
	0	1	0	1	1	
	0	0	1	1	0	
2	0	1	0	0	0	x_4
	0	0	0	1	1	
	0	1	1	1	1	
3	0	0	1	0	1	1
	0	1	1	0	1	
4	1	0	0	0	1	$\neg x_3$
	1	1	0	1	1	
	1	0	1	1	0	
5	1	1	0	0	1	$\neg x_4$
	1	0	0	1	0	
	1	1	1	1	0	
6	1	0	1	0	1	$\neg x_2$
	1	1	1	0	0	

Сопоставляя фрагменты «высоты» s столбца значений f заданной функции с соответствующими фрагментами той же «высоты» столбцов значений входных переменных x_2, x_3, x_4 , заполним столбец z_j таблицы. Из рассмотрения этого столбца следует, что $z_1 = x_2, z_2 = x_4, z_3 = 1, z_4 = \neg x_3, z_5 = \neg x_4, z_6 = \neg x_2$.

11.6. Модули УЛМ n, m второго типа

Существенного уменьшения элементной сложности при сохранении числа входов можно добиться при переходе от рассмотренного в предыдущем разделе класса модулей к универсальным логическим модулям n переменных с m парафазными входными переменными второго типа (рис. 11.9), которые будем обозначать УЛМ $n, m, 2$.

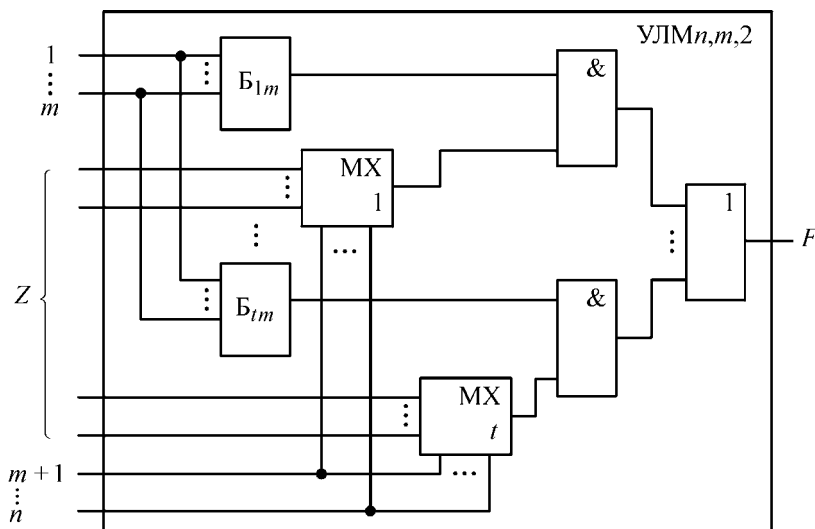


Рис. 11.9

Упрощение модулей достигается за счет использования в них не модулей УЛМ m, m , как в предыдущем случае, а только блоков, входящих в состав этих модулей.

Число входов в таких модулях при $m \geq 3$ определяется соотношением $M_4^m(n)$, приведенным в предыдущем разделе. В этих модулях число мультиплексоров равно $t(n)$, а каждый из мультиплексоров имеет 2^{n-m} управляющих входов.

На рис. 11.10 приведен модуль УЛМ $4, 3, 2$, построенный из блоков модуля УЛМ $3, 3$.

Пример 11.2. Определить настройку модуля УЛМ $4, 3, 2$ при реализации БФУ (табл. 11.4).

Перестроим табл. 11.4. Для этого каждую совокупность наборов значений входных переменных x_1, x_2, x_3 , соответствующую каждому блоку модуля УЛМ $3, 3$, повторим дважды и дополним четвертую позицию (для переменной x_4) первый раз нулями, а второй — единицами (табл. 11.6). Из рассмотрения табл. 11.6 следует, что $z_1 = x_3$, $z_2 = !x_1$, $z_3 = !x_2$, $z_4 = 0$, $z_5 = x_1$, $z_6 = 1$.

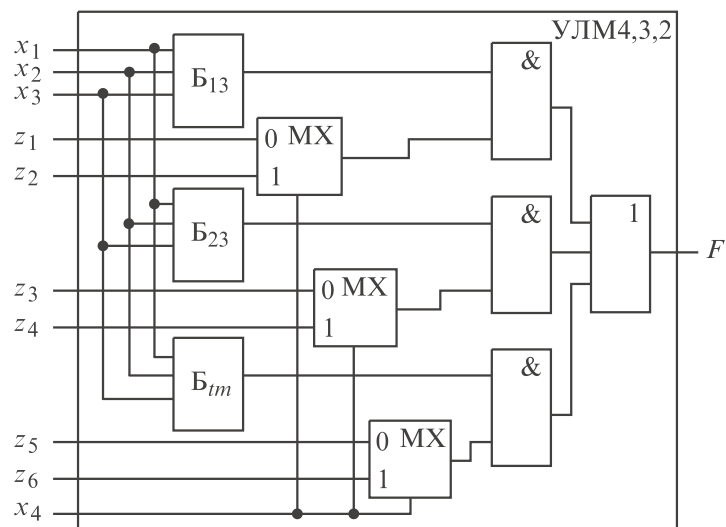


Рис.11.10

Т а б л и ц а 11.6

j	x_1	x_2	x_3	x_4	f	z_j
1	0	0	0	0	0	x_3
	1	0	1	0	1	
	0	1	1	0	1	
2	0	0	0	1	1	$\neg x_1$
	1	0	1	1	0	
	0	1	1	1	1	
3	1	0	0	0	1	$\neg x_2$
	0	0	1	0	1	
	1	1	1	0	0	
4	1	0	0	1	0	0
	0	0	1	1	0	
	1	1	1	1	0	
5	0	1	0	0	0	x_1
	1	1	0	0	1	
6	0	1	0	1	1	1
	1	1	0	1	1	

Из сравнения блоков, сформированных в табл. 11.6, с блоками модулей УЛМ4,4, приведенными в разд. 11.3, следует их полное совпадение. При этом $2t(m) = t(n)$. Это обеспечивает равное число входов модулей УЛМ4,3,2 и УЛМ4,4. Однако элементная сложность модуля УЛМ4,3,2 существенно ниже сложности как модуля УЛМ4,4, так и модуля УЛМ4,3,1. При этом для модуля УЛМ4,3,2 переменная x_4 может быть однофазной. Та же ситуация имеет место при $m = 5$ и $n = 6$.

В случаях, когда

$$2^{n-m} t(m) > t(n),$$

модули УЛМ $n,m,2$ имеют большее число входов по сравнению с модулями УЛМ n , n при меньшей элементной сложности и меньшем числе парафазных входных переменных.

При применении предлагаемых модулей в общем случае при перестроении таблицы истинности заданной БФУ наборы значений входных переменных $x_1, x_2, \dots, x_m$, соответствующие блокам модуля УЛМ m,m , повторяются 2^{n-m} раз, а остальные позиции на наборах, соответствующих блоку, заполняются $n - m$ разрядными двоичными числами, располагаемыми в лексикографическом порядке.

В заключение раздела отметим, что впервые использовать блоки, входящие в состав модулей УЛМ m,m , для построения модулей УЛМ n,m было предложено в [16] при построении модуля УЛМ4,3.

Частный случай модулей рассматриваемого класса — модули УЛМ $n,3,2$ описаны в [17]. В этой же работе приведен пример реализации БФУ пяти переменных модулем УЛМ5,3,2.

11.7. Модули УЛМ n без разделения информационных и настроечных входов

Определим нижнюю оценку числа входов (M) однокомпонентных модулей, универсальных в классе всех БФУ n переменных.

Для этого откажемся от разделения входов модулей на информационные и управляющие. Тогда при наличии всех входных переменных в парафазной форме справедливо, что

$$\begin{aligned} (2n+2)^M &\geq 2^{2^n}; \\ M(1 + \log(n+1)) &\geq 2^n; \\ M &\geq \left\lceil \frac{2^n}{1 + \log(n+1)} \right\rceil = t(n). \end{aligned}$$

Полученное соотношение показывает, что в модулях УЛМ n число входов не может быть меньше числа управляющих входов в модулях УЛМ n , n .

При $n = 3$ $t(3) = 3$. Попытка построения УЛМ3 с $M = 4$ не может привести к успеху, так как в [3] доказано, что не существует ни одной БФУ четырех переменных, которая одновременно реализует следующие три функции трех переменных: $f_1 = x_1 \oplus x_2 \oplus x_3$, $f_2 = (x_1 \vee x_2)x_3 \vee x_2x_3$, $f_3 = x_1x_2x_3$.

По этой причине невозможно построение модуля УЛМ3 с четырьмя входами даже при переходе к PN - или NPN -классификации.

Построение модуля УЛМ3 с пятью входами, реализующего все 256 функций трех переменных, видимо, невозможно, так как существует всего лишь 260 неэквивалентных вариантов настройки одной функции пяти переменных на функции трех переменных [18]. Поэтому можно утверждать, что модули УЛМ3,1 и УЛМ3,3 с шестью входами содержат минимальное число входов, которые могут иметь модули, универсальные в классе всех булевых функций трех переменных.

Переход к PN -классификации позволил решить эту задачу [19] для всех представителей 16 типов функций, существенно зависящих от трех переменных, с помощью порождающей функции пяти переменных.

Из рассмотрения таблицы настройки этого модуля, приведенной в разд. 8.4, следует, что он имеет один информационный и четыре управляющих входа. При этом на вход z_1 могут подаваться символы 0, x_2 , на вход z_2 — символы 0, 1, x_3 , $!x_3$, на вход z_3 — символы 0, 1, x_3 , $!x_3$, а на вход z_4 — символы 0, 1, x_2 , x_3 . Таким образом, рассмотренный модуль требует всего лишь одной парафазной переменной x_3 .

Отметим, что предложенный модуль УЛМ3 с пятью входами не является единственным [3—5].

При $n = 4$ $t(4) = 6$. Несмотря на то что существуют 560 неэквивалентных вариантов настройки одной функции шести переменных на функции четырех переменных [18], их оказалось недостаточно для реализации представителей всех 208 NPN -типов функций четырех переменных.

В [18] показано, что существуют 8400 неэквивалентных вариантов настройки одной функции семи переменных на функции четырех переменных. В [3, 4] построены функции семи переменных, порождающие представителей всех NPN -типов функций четырех переменных.

Материал настоящего раздела впервые опубликован в [20].

Выводы

1. Показано, что модули УЛМ t, t , обладающие тем же числом входов и теми же функциональными возможностями, что и модули УЛМ $t, 1$, более эффективны при построении на их основе модулей УЛМ n ($n > t$).
2. Показано, что использование блоков, входящих в состав модулей УЛМ t, t , позволяет упростить модули УЛМ n, n .
3. Предложен класс модулей УЛМ n, t , являющийся обобщением известных классов модулей УЛМ $n, 0$, УЛМ $n, 1$ и УЛМ n, n .
4. Показано, что модули УЛМ n, t первого типа, построенные на базе модулей УЛМ t, t , при некоторых значениях n проще модулей УЛМ n, n при том же числе входов.
5. Показано, что модули УЛМ n, t второго типа, построенные на базе блоков модулей УЛМ t, t , проще модулей УЛМ n, t первого типа, а при некоторых значениях n и модулей УЛМ n, n при том же числе входов.
6. Предложены методы настройки модулей УЛМ n, t первого и второго типа на реализацию произвольных булевых функций n переменных.
7. Получена оценка числа входов модулей УЛМ n, t .

Л и т е р а т у р а

1. Yau S. S., Tang C. K. Universal logic modules and their applications // IEEE Trans. on Computers. 1970. N 2.
2. Preparata F. P., Muller D. E. Generation of near-optimal universal boolean functions // J. Comp. and System Sciences. 1970. N 2.
3. Stone H. E. Universal logic modules // Recent developments in switching theory. Ed. by A. Mukhopadhyay. New York: Academic Press, 1971.
4. Вариавский В. И., Песчанский В. А., Мараховский В. Б. Многофункциональные модули, реализующие все функции трех и четырех переменных // II Всесоюз. совещ. по теории релейных устройств и конечных автоматов. Тез. докл. Рига : Зинатне, 1971.
5. Бочков П. Е., Попов Ю. А., Сладков В. А. Проектирование многофункциональных модулей с использованием инвариантов булевых функций // Автоматика и вычисл. техника. 1973. № 3.
6. Якубайтис Э. А. Универсальные логические элементы // Автоматика и вычисл. техника. 1973. № 5.
7. Friedman A. D., Menon P. R. Theory and design of switching circuits. New York: Computer Science Press, 1975.
8. Якубайтис Э. А. Логические автоматы и микромодули. Рига : Зинатне, 1975.
9. Якубайтис Э. А. Теория автоматов. Многофункциональные логические модули // Итоги науки и техники. Теория вероятностей. Математическая статистика. Теоретическая кибернетика. М.: ВИНТИ, 1976. № 13.
10. Майоров С. А., Скорубский В. И., Кравцов Л. Я. Универсальные логические модули и их применение // Управляющие системы и машины. 1977. № 1.
11. Малев В. А. Структурная избыточность в логических устройствах. М.: Связь, 1978.
12. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Судовые управляющие логические системы. Унифицированные логические схемы. Л.: ИПК СП, 1981.

13. Мищенко В. А., Козюминский В. Д., Семашко А. Н. Многофункциональные автоматы и элементная база цифровых ЭВМ. М.: Радио и связь, 1981.
14. Пупырев Е. И. Перестраиваемые автоматы и микропроцессорные системы. М.: Наука, 1984.
15. Викентьев Л. Ф., Аляев Ю. А., Шалыто А. А. Универсальный логический модуль. А. с. СССР № 1335974 // Бюл. изобр. 1987. № 33.
16. Артюхов В. Л., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР. № 924697 // Бюл. изобр. 1982. № 16.
17. Артюхов В. Л., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР. № 1073768 // Бюл. изобр. 1984. № 6.
18. Стародубцев Н. А. Соотношения для числа настроек многофункциональных логических модулей // Изв. АН СССР. Техническая кибернетика. 1972. № 4.
19. Артюхов В. Л., Шалыто А. А., Фишман Л. М. Многофункциональный логический модуль. А. с. СССР. № 760451 // Бюл. изобр. 1980. № 32.
20. Шалыто А. А. Модули, универсальные в классе всех булевых функций, с парафазными входными переменными // Изв. РАН. Теория и системы управления. 1997. № 5.

Глава 12

Оценка функциональных возможностей программируемых логических матриц

Имеется большое число работ, посвященных методам рационального применения программируемых логических матриц (ПЛМ), например [1—4].

Однако вопрос об оценке функциональных возможностей таких матриц при реализации произвольных булевых формул в базисе И, ИЛИ, НЕ в литературе не рассматривался.

Решению этого вопроса посвящена настоящая глава. При этом используется следующее ограничение: предполагается, что у ПЛМ имеются только прямые входы и выходы.

12.1. Непосредственная реализация произвольной скобочной булевой формулы

Реализация произвольной скобочной формулы в рассматриваемом базисе с помощью ПЛМ сводится к реализации системы из $c/2 + 1$ дизъюнктивных нормальных форм (ДНФ), являющихся подформулами заданной формулы, где c — число скобок в формуле.

При этом каждая ДНФ реализуется на соответствующем выходе ПЛМ, а каждый из выходов, кроме последнего, на котором реализуется формула в целом, соединяется с соответствующим входом ПЛМ.

Наихудшие параметры (v — число входов, t — число термов, b — число выходов) будет иметь ПЛМ, реализующая нормальную неповторную формулу в базисе И, ИЛИ, которая содержит максимальное число скобок и в которой h одиночных букв объединены между собой непосредственно только операцией ИЛИ. Например, при $h = 16$ соответствующая формула имеет следующую структуру:

$$f = ((1+1)(1+1) + (1+1)(1+1))((1+1)(1+1) + (1+1)(1+1)).$$

Для оценки параметров ПЛМ, реализующих формулы этого класса, определим число скобок в них.

На первом уровне каждая пара скобок объединяет две буквы. При этом одна скобка соответствует одной букве. Таким образом, число скобок первого уровня $c_1 = h = h/4^0$.

На втором уровне каждая пара скобок объединяет восемь букв: одна скобка на четыре буквы. Следовательно, число скобок второго уровня $c_2 = h/4 = h/4^1$.

На третьем уровне каждая пара скобок объединяет 32 буквы: одна скобка на 16 букв. Следовательно, $c_3 = h/16 = h/4^2$.

Максимальное число уровней скобок для БФ из h букв определяется соотношением $y = \log_4 h$. При этом число скобок максимального уровня $c_y = h/4^{\log_4 h - 1}$.

Таким образом, общее число скобок определяется соотношением

$$\begin{aligned} c &= \sum_{i=1}^y c_i = \frac{h}{4^0} + \frac{h}{4^1} + \frac{h}{4^2} + \dots + \frac{h}{4^{(\log_4 h) - 1}} = \\ &= \frac{h}{4^0} + \frac{h}{4^1} + \frac{h}{4^2} + \dots + \frac{h}{4^{(\log_4 h) - 1}} + \frac{h}{4^{\log_4 h}} - 1 = \\ &= \underbrace{(1 + 4 + 16 + \dots + h)}_{1 + \log_4 h} - 1 = \frac{4^{1 + \log_4 h} - 1}{4 - 1} - 1 = \frac{4h - 1}{3} - 1 = \frac{4(h - 1)}{3}. \end{aligned}$$

Ввиду того что число скобок четно, то

$$c = \left\langle \frac{4(h - 1)}{3} \right\rangle,$$

где $\langle \rangle$ — символ округления до ближайшего меньшего четного. Это соотношение определяет максимальное число скобок в базисе И, ИЛИ из h букв.

При этом

$$\begin{aligned} v &= h + \frac{c}{2} \leq h + \frac{\langle (4/3)(h - 1) \rangle}{2} < h + \frac{2(h - 1)}{3} = \frac{5h - 2}{3} < [1.67h]; \\ b &= 1 + \frac{c}{2} \leq 1 + \frac{\langle (4/3)(h - 1) \rangle}{2} < 1 + \frac{2(h - 1)}{3} = \frac{2h + 1}{3} < 1 + [0.67h]. \end{aligned}$$

Определим число термов для формул рассматриваемого вида. Искомая величина определяется соотношением $t_1 = h + u$, где u —

число двухвходовых элементов И в древовидной схеме, реализующей формулу рассматриваемого класса.

Число элементов И в дереве, последним элементом в котором является элемент И, определяется соотношением

$$u = \underbrace{\frac{h}{4} + \frac{h}{16} + \frac{h}{64} + \frac{h}{256} + \dots}_{\log_4 h} = \frac{h}{4} \left(1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots \right) =$$

$$= \frac{h}{4} \frac{(1/4)^{\log_4 h} - 1}{1/4 - 1} = \frac{h}{4} \frac{1 - (1/4)^{\log_4 h}}{3/4} = \frac{h}{3} (1 - 4^{-\log_4 h}) = \frac{h-1}{3}.$$

Если последним элементом в схеме является элемент ИЛИ, то

$$u = 2 \frac{h/2 - 1}{3} = \frac{h-2}{3}.$$

Таким образом,

$$u = \begin{cases} \frac{h-1}{3} & \text{при } h = 2^2, 2^4, 2^6, \dots \\ \frac{h-2}{3} & \text{при } h = 2^3, 2^5, 2^7, \dots \end{cases}$$

Следовательно,

$$u = \left\lfloor \frac{h-1}{3} \right\rfloor.$$

При этом

$$t_1 = h + \left\lfloor \frac{h-1}{3} \right\rfloor = [1.33 h].$$

Следовательно, площадь программируемой логической матрицы, реализующей произвольную БФ в базисе И, ИЛИ, определяется соотношением:

$$s_1 = (v + b) t_1 = \left(\frac{5h-2}{3} + \frac{2h+1}{3} \right) \frac{4h}{3} < 3.11 h^2.$$

Число точек, оставшихся в ПЛМ после программирования, определяется соотношением

$$T_1 \leq h + 2(h-1) + 1 = 3h - 1,$$

так как каждой букве и каждому выходу соответствует одна точка, а каждому элементу древовидной схемы — две.

12.2. Реализация дизъюнктивной нормальной формы, соответствующей произвольной скобочной булевой формуле

Оценим число термов и число букв в ДНФ, получающейся после раскрытия скобок в заданной формуле в рассматриваемом базисе. Автор предполагает, что эти величины максимальны, если заданная формула имеет вид:

$$f = (x_1 \vee x_2 \vee \dots \vee x_k)(x_{k+1} \vee x_{k+2} \vee \dots \vee x_{2k}) \dots \quad (12.1)$$

Определим средние значения величины k , при которых указанные характеристики принимают максимальные значения.

Если исходная формула состоит из h букв, то число сомножителей в ней равно h/k . Тогда число термов в соответствующей ДНФ определяется соотношением

$$t_2 = k^{h/k},$$

а число букв — соотношением

$$h_2 = (h/k)t_2 = (h/k)k^{h/k} = hk^{h/k-1}.$$

Определим производную

$$\begin{aligned} (t_2)'_k &= (k^{h/k})' = (e^{(h/k)\ln k})' = e^{(h/k)\ln k} ((h/k)\ln k)' = \\ &= k^{(h/k)} \left(-(h/k^2)\ln k + (h/k)(1/k) \right) = \frac{h}{k^2} (1 - \ln k) k^{h/k} = \\ &= h(1 - \ln k) k^{h/k-2}. \end{aligned}$$

Найдем значение k , при котором

$$h(1 - \ln k) k^{h/k-2} = 0.$$

Следовательно, $k = e$.

Определим теперь

$$\begin{aligned} (k^{h/k-1})'_k &= (e^{(h/k-1)\ln k})' = e^{(h/k-1)\ln k} ((h/k)\ln k - \ln k)' = \\ &= k^{h/k-1} \left(-(h/k^2)\ln k + (h/k)(1/k) - (1/k) \right) = \\ &= k^{h/k-1} \left((h/k^2)(1 - \ln k) - (1/k) \right) = \\ &= k^{h/k-2} \left((h/k)(1 - \ln k) - 1 \right). \end{aligned}$$

Определим соотношение, при выполнении которого

$$k^{(h/k)-2}((h/k)(1 - \ln k) - 1) = 0.$$

Из этого соотношения следует, что

$$k + h(\ln k) = h. \quad (12.2)$$

Таким образом, для любого h величина k , обеспечивающая максимум числа термов, постоянна и равна e , в то время как значение k , гарантирующее максимум числа букв в ДНФ, зависит от числа букв в заданной формуле.

Выполним анализ соотношения (12.2).

При $h \rightarrow \infty$ это соотношение приобретает вид:

$$h(\ln k) = h.$$

При этом $k = e$. Предположим, что $k = e$, тогда

$$e + h > h,$$

и поэтому $k < e$.

При $h = 0$ справедливо соотношение $k = 0$. Таким образом,

$$0 < k = f(h) < e.$$

Например, $k = 2.22$ при $h = 10$ и $k = 2.45$ при $h = 20$.

Предположим, что $h = 20$, и рассмотрим три формулы:

1) $f = (x_1 \vee x_2)(x_3 \vee x_4) \dots (x_{19} \vee x_{20})$, для которой $k = 2$, $t_2 = 2^{20/2} = 1024$, $h_2 = 20 \cdot 2^{20/2-1} = 10240$;

2) $f = (x_1 \vee x_2 \vee x_3) \dots (x_{16} \vee x_{17} \vee x_{18})(x_{19} \vee x_{20})$, для которой $k = 20/7 = 2.86$, $t_2 = 3^6 \cdot 2 = 1458$, $h_2 = 1458 \cdot 7 = 10206$;

3) $f = (x_1 \vee x_2 \vee x_3) \dots (x_{10} \vee x_{11} \vee x_{12})(x_{13} \vee x_{14}) \dots (x_{19} \vee x_{20})$, для которой $k = 20/8 = 2.5$, $t_2 = 3^4 \cdot 2^4 = 1296$, $h_2 = 1296 \cdot 8 = 10368$.

Из соотношения (12.1) и того, что величина t достигает максимума при $k = e$, следует, что

$$f = \begin{cases} (1+1+1)(1+1+1) \dots (1+1+1), & \text{если } \text{mod}_3 h = 0; \\ (1+1+1) \dots (1+1+1)(1+1)(1+1), & \text{если } \text{mod}_3 h = 1; \\ (1+1+1) \dots (1+1+1)(1+1), & \text{если } \text{mod}_3 h = 2. \end{cases}$$

При этом соотношение для подсчета максимального числа термов в ДНФ имеет вид:

$$t_2 = \begin{cases} 3^{h/3}, & \text{если } \text{mod}_3 h = 0; \\ 4 \cdot 3^{(h-4)/3}, & \text{если } \text{mod}_3 h = 1; \\ 2 \cdot 3^{(h-2)/3}, & \text{если } \text{mod}_3 h = 2. \end{cases} \quad (12.3)$$

Это соотношение полностью совпадает с соотношением для определения числа клик (максимальных полных подграфов) в графах Муна—Мозера [6], которые имеют наибольшие возможные клики для графов с h вершинами. При этом показано, что произвольный граф с h вершинами может иметь не более t_2 клик [7].

Таким образом, максимальное число термов в ДНФ, получаемых за счет раскрытия скобок в произвольных БФ в базисе И, ИЛИ из h букв, совпадает с максимальным числом клик в произвольном графе с h вершинами.

Из соотношения (12.3) следует, что

$$t_2 \leq 3^{h/3}. \quad (12.4)$$

Площадь программируемой логической матрицы, реализующей ДНФ, определяется соотношением

$$S_2 = (h+1)t_2.$$

Следовательно, площадь ПЛМ, требующаяся для реализации формул, порождающих ДНФ с максимальным числом термов, определяется соотношением:

$$S_2 = \begin{cases} (h+1) \cdot 3^{h/3}, & \text{если } \text{mod}_3 h = 0; \\ 4 \cdot (h+1) \cdot 3^{(h-4)/3}, & \text{если } \text{mod}_3 h = 1; \\ 2 \cdot (h+1) \cdot 3^{(h-2)/3}, & \text{если } \text{mod}_3 h = 2. \end{cases}$$

Из соотношения (12.4) следует, что

$$S_2 \leq (h+1) \cdot 3^{h/3}.$$

Число букв в формулах, порождающих ДНФ с максимальным числом термов, определяется соотношением:

$$h_2 = \begin{cases} (h/3) \cdot 3^{h/3}, & \text{если } \text{mod}_3 h = 0; \\ 4 \cdot]h/3[\cdot 3^{(h-4)/3}, & \text{если } \text{mod}_3 h = 1; \\ 2 \cdot]h/3[\cdot 3^{(h-2)/3}, & \text{если } \text{mod}_3 h = 2. \end{cases}$$

Таким образом,

$$h_2 \leq]h/3[\cdot 3^{h/3}.$$

Число точек программирования в этом случае определяется соотношением

$$T_2 = h_2 + t_2.$$

Следовательно, для формул, порождающих ДНФ с максимальным числом термов:

$$T_2 = \begin{cases} ((h/3) + 1) \cdot 3^{h/3}, & \text{если } \text{mod}_3 h = 0; \\ 4 \cdot (]h/3[+ 1) \cdot 3^{(h-4)/3}, & \text{если } \text{mod}_3 h = 1; \\ 2 \cdot (]h/3[+ 1) \cdot 3^{(h-2)/3}, & \text{если } \text{mod}_3 h = 2. \end{cases}$$

Таким образом,

$$T_2 \leq (]h/3[+ 1) \cdot 3^{h/3}.$$

Из приведенных соотношений следует, что метод реализации БФ, рассмотренный в предыдущем разделе, имеет более низкие оценки сложности.

Однако, несмотря на это, можно показать, что существуют такие постановки задачи, при которых оба рассмотренных метода позволяют использовать ПЛМ с одинаковой эффективностью.

Пусть выбрана ПЛМ 556 РТ1 с параметрами $v = 16, t = 48, b = 8$ и требуется определить максимальную величину h , при которой произвольная формула из этого числа букв будет реализована с помощью одной такой ПЛМ.

Другими словами, требуется определить, для какой величины h выбранная ПЛМ является модулем, универсальным в классе формул в базе И, ИЛИ, НЕ.

При использовании первого метода при $h = 10$ $v_1 = 16, t_1 = 13, b_1 = 7$; при $h = 11$ $v_1 = 17, t_1 = 14, b_1 = 7$. Таким образом, в этом случае $h = 10$.

При применении второго метода при $h = 10$ $v_2 = 10, t_2 = 36, b_2 = 1$; при $h = 11$ $v_2 = 11, t_2 = 54, b_2 = 1$. Таким образом, и в этом случае $h = 10$.

Следовательно, выбранная ПЛМ является при использовании любого из рассмотренных методов модулем, универсальным в рассматриваемом классе формул из $q = 10$ букв.

Это позволяет оценить [5] число программируемых логических матриц, требующихся для реализации произвольной формулы в рассматриваемом базисе из h ($h > 10$) букв

$$L \leq \left\lceil \frac{2(h-1)}{q} \right\rceil = \left\lceil \frac{h-1}{5} \right\rceil,$$

и для реализации системы из N произвольных формул в рассматриваемом базисе, правые части которых содержат H букв:

$$L \leq \left\lceil \frac{2(H-N)}{q} \right\rceil + N - 1 = \left\lceil \frac{H-N}{5} \right\rceil + N - 1.$$

Еще один подход к применению ПЛМ в качестве многофункциональных логических модулей состоит в программировании их по порождающим функциям таких модулей [8]. Это позволяет, в частности, использовать однократно программируемые ПЛМ (как, впрочем, и однократно программируемые ПЗУ) в режиме устройств, многократно программируемых с помощью настроечных входов.

Предложенный в настоящем разделе подход базируется на материале, изложенном в [9].

Выводы

1. Предложены два метода вложения произвольных булевых формул в базисе И, ИЛИ из h букв в программируемые логические матрицы и получены соответствующие оценки сложности последних.
2. Показано, что максимальное число термов в дизъюнктивных нормальных формах, получаемых за счет раскрытия скобок в произвольных булевых формулах в базисе И, ИЛИ из h букв, совпадает с числом клик (максимальных полных подграфов) в графах Муна—Мозера, имеющих наибольшие возможные клики для графов с h вершинами.
3. Предложены подходы к применению ПЛМ в качестве многофункциональных логических модулей, универсальных в рассматриваемом классе формул из q букв, что позволяет оценить число программируемых логических матриц, требующихся для реализации произвольных формул этого класса.

Литература

1. Баранов С. И., Синев В. Н. Программируемые логические матрицы в цифровых системах // Зарубежная радиоэлектроника. 1979. № 1.
2. Баранов С. И., Баркалов А. А. Применение программируемых логических матриц в цифровой технике // Зарубежная радиоэлектроника. 1982. № 6.
3. Баранов С. И., Скляр В. А. Цифровые устройства на программируемых БИС с матричной структурой. М.: Радио и связь, 1986.
4. Ачасова С. М. Алгоритмы синтеза автоматов на программируемых матрицах. М.: Радио и связь, 1987.
5. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Настраиваемые модули для управляющих логических устройств. Л.: Энергоиздат, 1981.
6. Рейнгольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. М.: Мир, 1980.
7. Moon J., Moser L. On cliques in graphs // Israel J. Math. 1965. N 3.
8. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Судовые управляющие логические системы. Унифицированные логические схемы. Л.: ИПК СП, 1981.
9. Артюхов В. Л., Шалыто А. А., Кузнецова О. С. Оценка функциональных возможностей программируемых логических матриц // Автоматика и вычисл. техника. 1985. № 2.

Глава 13

Однородные модули из элементов с двусторонней проводимостью и реализация комбинационных схем

13.1. Модули, универсальные в классе произвольных булевых формул в базисе И, ИЛИ, НЕ

К элементам с двусторонней проводимостью относятся, в частности, контактные элементы. Сравнительно большие габариты рележно-контактных элементов приводят к нецелесообразности построения на них порождающих функций, используемых при проектировании модулей из функциональных элементов (элементов с односторонней проводимостью). Это связано с тем, что такие модули обладают избыточностью как по числу элементов, так и по числу внешних выводов.

Поэтому в настоящем разделе решается задача создания модулей из элементов с двусторонней проводимостью, которые до определенного значения числа букв q в реализуемой БФ не обладают элементной избыточностью. При этих значениях q многофункциональность таких модулей обеспечивается только за счет избыточности по внешним выводам. При больших значениях q , как и в модулях из элементов с односторонней проводимостью, имеются оба вида избыточности. Эти модули в отличие от модулей из элементов с односторонней проводимостью настраиваются в основном за счет отождествления выходов.

В предположении о равной доступности прямых и инверсных входных переменных предлагаемые модули, построенные в качестве примера из однотипных одноконтактных реле, имеют однородную структуру, представленную на рис. 13.1. На этом рисунке каждое реле 3, состоящее из нормально разомкнутого контакта 1 и обмотки 2, выделено прямоугольником.

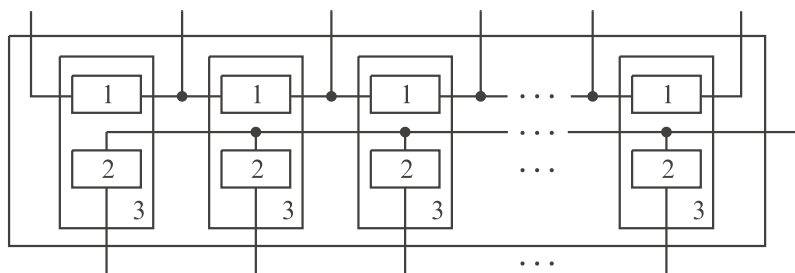


Рис. 13.1

Этот класс модулей с $2(q + 1)$ внешними выводами впервые описан в [1]. Такие модули из q элементов (при $q \leq 6$) являются универсальными в классе БФ в базисе $\{\&, \vee, !\}$ из q букв при равной доступности прямых и инверсных входных переменных. При этих значениях q модули обладают избыточностью только по внешним выводам.

Предложенные модули существенно более просты по сравнению с модулями из элементов с односторонней проводимостью с теми же значениями q . Так, например, при $q = 5$ модуль [2] имеет 14 внешних выводов (два служат для подачи питания) и состоит из 79 МДП-транзисторов, объединенных в функциональные элементы, в то время как предлагаемый модуль при этом значении q имеет 12 внешних выводов и содержит лишь пять элементов.

При $q = 7$ модуль из семи элементов реализует путем настройки 179 из 180 параллельно-последовательных контактных схем. Метод подсчета числа таких схем из q контактов предложен в [3], а их табулирование при $q \leq 8$ выполнено автором настоящей работы в [4].

С ростом величины q доля параллельно-последовательных контактных схем, реализуемых предлагаемым модулем из q элементов, уменьшается. Так, например, при $q = 8$ модуль из восьми элементов путем настройки реализует 518 из 522 возможных схем.

Причина невозможности построения безызбыточных по числу элементов модулей при $q \geq 7$ рассмотрена в следующем разделе, в котором показано также, что для произвольного значения q может быть построен модуль предлагаемого типа, число элементов в котором превышает q .

Модификации рассмотренных модулей, осуществленные за счет введения диодов, приведены в [6—10].

Если доступны лишь только прямые входные переменные, то в качестве q -универсального модуля при $q = 2, 3$ может быть использована схема из q переключательных контактов [5]. На рис. 13.2 в качестве примера приведен такой модуль при $q = 3$. Этот модуль состоит из реле 3, каждое из которых содержит обмотку 2 и переключательный контакт, условно представленный нормально разомкнутым 1 и нормально замкнутым 4 контактами.

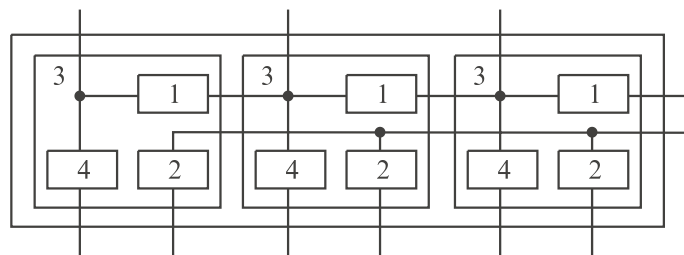


Рис. 13.2

В [11] приведена настройка этого модуля на реализацию представителя каждого из 20 P -типов бесповторных БФ в рассматриваемом базисе.

13.2. Топологический метод реализации многовыходных комбинационных схем

Предположим, что прямые и инверсные входные переменные равнодоступны и заданы модули, представленные на рис. 13.1.

Применяя формульный метод при $q \leq 6$, может быть построена схема, реализующая произвольную БФ в рассматриваемом базисе из h букв ($h \geq q$), число модулей в которой определяется соотношением (2.1). Из этого соотношения следует, что при таком методе синтеза избыточность по числу элементов может достигать двух, несмотря на то что при $h \leq 6$, как отмечено в предыдущем разделе, формулы заданного класса могут быть реализованы на таких модулях без избыточности по числу элементов.

Покажем, что свойство двусторонней проводимости позволяет предложить метод синтеза схем из рассматриваемых модулей, при котором имеет место существенно меньшая избыточность, чем при формульном методе.

Этот метод, базирующийся на свойстве эйлеровых графов, которые могут быть «пройдены» не отрывая руки от бумаги [9], может быть назван топологическим.

Граф является эйлеровым, если он содержит ноль или две вершины нечетной степени [12]. Вершина называется нечетной, если она образована нечетным числом ребер. Из теории графов известно, что при отсутствии нечетных вершин граф «проходим» при его обходе, начиная из любой четной вершины, а при двух нечетных вершинах его обход должен начинаться из одной из этих вершин.

Известен [12] алгоритм Хоанг Туи, который строит эйлерову цепь в случае, если граф содержит две нечетные вершины.

В произвольном графе число нечетных вершин четно, и при наличии более двух таких вершин он «непроходим».

Эти результаты теории графов могут быть использованы при реализации произвольных многополюсных схем, построенных из нормально разомкнутых контактов, с помощью модуля, который приведен на рис. 13.1.

Если реализуемая схема является эйлеровой, то в ней должен быть найден эйлеров маршрут (эйлерова линия), который определяет настройку модуля — обозначения контактов в «цепочке» и переключки между ее внешними выводами. Так как в цепочке обеспечен доступ к любой «точке», то произвольная эйлерова схема из q контактов может быть безызбыточно реализована такой цепочкой из q контактов.

На рис. 13.3 приведена контактная схема, реализующая систему булевых формул $z_1 = x_1 x_2 x_4$, $z_2 = x_3$. Эта схема из четырех контактов содержит два узла нечетной степени и поэтому является эйлеровой. Находя эйлеров маршрут в этой схеме, мы тем самым определяем настройку модуля из четырех контактов (рис. 13.4).

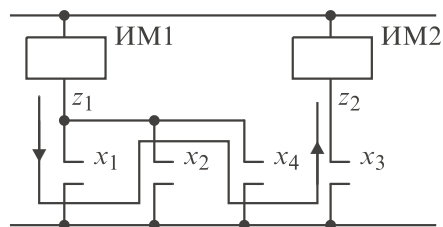


Рис. 13.3

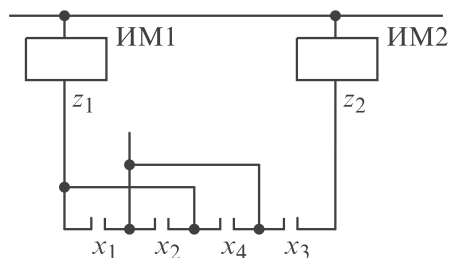


Рис. 13.4

Из изложенного следует, что особенность предлагаемого топологического метода состоит в том, что он в отличие от методов синтеза схем из функциональных элементов позволяет эффективно реализовать не только одну формулу, но и систему формул.

Применим предложенный метод синтеза для построения двухполюсных контактных схем, реализующих одиночные БФ в рассматриваемом базисе. Простейшая двухполюсная схема, содержащая четыре нечетных (n) узла, приведена на рис. 13.5. Из-за нали-

чия таких четырех узлов в этой схеме она «непроходима» и, следовательно, не обеспечивает 4-универсальности модуля в классе произвольных параллельно-последовательных контактных схем из четырех контактов. Однако это свойство обеспечивается в классе *PN*-типов таких схем. Так, например, вместо схемы на рис. 13.5 всегда может быть использована функционально эквивалентная ей схема с двумя нечетными узлами (рис. 13.6), являющаяся эйлеровой.

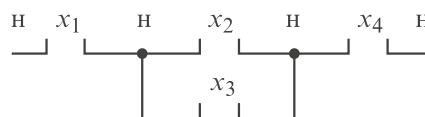


Рис. 13.5

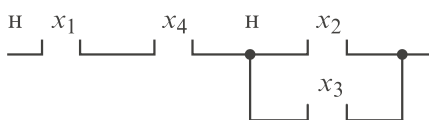


Рис. 13.6

Анализ, выполненный автором в [5], показал, что для любого *PN*-типа БФ в рассматриваемом базисе из шести и менее букв может быть выбран такой представитель, который реализуется эйлеровой схемой, что и обеспечивает безызбыточность *q*-универсальных модулей этого типа при $q \leq 6$.

При $q = 7$ существует один *PN*-тип БФ, для которого не может быть выбран представитель, который реализуется эйлеровой схемой, так как все схемы, функционально эквивалентные схеме на рис. 13.7, содержат четыре нечетных узла.

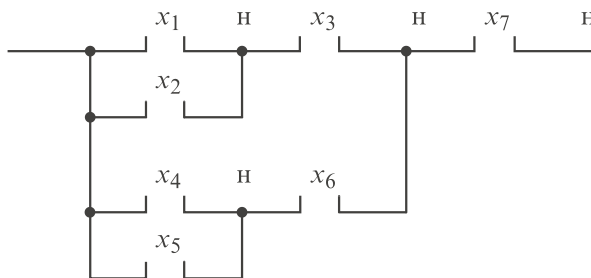


Рис. 13.7

Реализация этой схемы с помощью рассматриваемых модулей обеспечивается за счет введения избыточного (и) контакта (рис. 13.8) по аналогии с введением «моста» в задаче Эйлера о «кёнигсбергских мостах», породившей теорию графов [13]. При этом отметим, что на обмотку реле, соответствующую «избыточному»

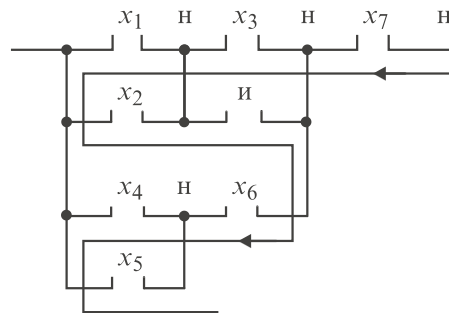


Рис. 13.8

контакту, подается логический ноль, что делает этот контакт всегда разомкнутым.

На рис. 13.9 приведен предлагаемый модуль из восьми контактов, реализующий схему на рис. 13.8.

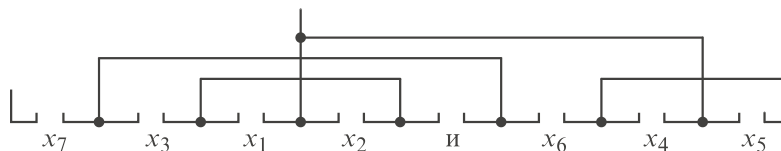


Рис. 13.9

В [5] показано, что длина «цепочки», требующейся для реализации произвольной формулы в базисе И, ИЛИ из h букв, удовлетворяет соотношению

$$h \leq d \leq [(4/3)h - 1/3].$$

Таким образом, в данном случае избыточность не превосходит 33 % в отличие от более чем 100 % избыточности, имеющей место при применении модулей из функциональных элементов.

Предложенные модули были использованы в НПО «Аврора» при создании унифицированных кассет логики на релейно-контактных элементах. Эти модули позволили согласовать большое число внешних выводов реле с относительно небольшим числом внешних выводов этих кассет.

Кроме [1, 5] имеется еще несколько работ [14—18], в которых рассмотрены предложенные модули.

Существенный интерес для практики представляет вопрос о реализации предложенных модулей на бесконтактных элементах с двусторонней проводимостью, например на оптронах, который в настоящей работе не рассматривается.

Весьма полезным было приложение подхода, изложенного в настоящей главе, к исследованию функциональных возможностей «цепочек» из одинаковых резисторов с промежуточными выводами

между ними. При этом в [19], например, показано, что цепочка из пяти одинаковых резисторов, каждый из которых имеет сопротивление R , может реализовать путем наложения перемычек 35 различных номиналов сопротивлений, лежащих в диапазоне от $R/5$ до $5R$.

Выводы

1. Предложен принципиально новый класс многофункциональных модулей из элементов с двусторонней проводимостью, универсальных в классе БФ в базисе И, ИЛИ из h букв, которые за счет настройки на выходах при $h \leq 6$ не обладают элементной избыточностью.

2. Показано, что для произвольных значений числа букв h в указанных булевых формулах элементная избыточность таких модулей не может превышать 33 %, что существенно меньше, чем у известных модулей из элементов с односторонней проводимостью.

3. Подход, использованный при построении рассмотренных модулей, был применен для исследования функциональных возможностей однородных «цепочек» из резисторов с промежуточными выходами между ними. При этом, в частности, показано, что такая «цепочка» из пяти резисторов порождает 35 различных номиналов сопротивлений.

4. На основе теории эйлеровых графов предложен топологический метод реализации многовыходных комбинационных схем, позволяющий эффективно применять свойство двусторонней проводимости элементов, из которых построены модули.

5. Предложены однородные многофункциональные модули из элементов с двусторонней проводимостью, универсальные в классе БФ в базисе И, ИЛИ, НЕ при $h \leq 3$.

Литература

1. Прангивили И. В., Ускач М. А., Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 427336 // Бюл. изобр. 1974. № 17.
2. Прангивили И. В., Ускач М. А., Копейкин Г. А. Комплекс логических МДП-интегральных схем для систем автоматики и телемеханики // Приборы и системы управления. 1970. № 4.
3. Шеннон К. Э. Работы по теории информации и кибернетике. М.: Изд-во иностр. литер., 1963.
4. Артюхов В. Л. Логические методы синтеза дискретных систем. Л.: ИПК СП, 1974.
5. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Настраиваемые модули для управляющих логических устройств. Л.: Энергоиздат, 1981.
6. Артюхов В. Л., Шалыто А. А., Суханов Ю. И. Многофункциональный логический модуль. А. с. СССР № 819809 // Бюл. изобр. 1981. № 13.

7. Артюхов В. Л., Шалыто А. А., Суханов Ю. И. Многофункциональный логический модуль. А. с. СССР № 840883 // Бюл. изобр. 1981. № 23.
8. Артюхов В. Л., Кузнецова О. С., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1629909 // Бюл. изобр. 1991. № 7.
9. Артюхов В. Л., Кузнецова О. С., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1629984 // Бюл. изобр. 1991. № 7.
10. Артюхов В. Л., Кузнецова О. С., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 1631713 // Бюл. изобр. 1991. № 8.
11. Артюхов В. Л., Фишман Л. М., Боброва И. Л., Шалыто А. А. Многофункциональный логический модуль. А. с. СССР № 841518 // Бюл. изобр. 1981. № 23.
12. Зыков А. А. Теория конечных графов. Новосибирск: Наука, 1969.
13. Оре О. Графы и их применение. М.: Мир, 1965.
14. Артюхов В. Л., Шалыто А. А. Функциональные возможности однородной одномерной реберной структуры из элементов с двусторонней проводимостью // Логический синтез в дискретных однородных средах. Третье Всесоюз. совещ. М.: Ин-т проблем передачи информации, 1974.
15. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Однородные одномерные реберные структуры из элементов с двусторонней проводимостью // Однородные вычислительные структуры и малые ЭВМ. Всесоюз. семинар. М.: Ин-т проблем управления, 1979.
16. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Многофункциональные логические модули из элементов с двусторонней проводимостью // Изв. вузов. Приборостроение. 1981. № 4.
17. Артюхов В. Л., Шалыто А. А. Применение эйлеровых и гамильтоновых графов при реализации переключательных схем // Методы и программы решения оптимизационных задач на графах и сетях. Тез. докл. III Всесоюз. совещ. Ч. 1. Новосибирск: ВЦ СО АН СССР, 1984.
18. Артюхов В. Л., Шалыто А. А. Однородные структуры для реализации булевых формул // Логические методы построения однородных и систолических структур. Труды I Всесоюз. совещ. М.: Ин-т проблем передачи информации, 1988.
19. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Функциональные возможности микроэлектронных резистивных наборов // Автометрия. 1979. № 3.

Глава 14

Реализация булевых формул и булевых функций однородными структурами

14.1. Реализация булевых формул в базисе И, ИЛИ, НЕ каскадами Макхопадхая и пороговыми элементами

Одномерная однонаправленная структура из элементов И и ИЛИ называется каскадом Макхопадхая [1].

Исследование функциональных возможностей таких каскадов выполнено в [2]. При этом, в частности, показано, что они реализуют только неповторные пороговые функции (формулы). Следовательно, каждый каскад Макхопадхая является пороговым элементом.

В этой работе, а также в [3] рассмотрены обобщенные каскады Макхопадхая, являющиеся древовидными схемами из каскадов Макхопадхая.

При этом показано, что такой обобщенный каскад с h входами содержит не более $\lceil h/2 \rceil$ каскадов Макхопадхая.

Так как произвольная БФ в базисе И, ИЛИ, НЕ из h букв при равной доступности прямых и инверсных входных переменных реализуется обобщенным каскадом с h входами, то тем самым доказывается, что формулы этого класса реализуются схемами из пороговых элементов, число которых определяется соотношением

$$1 \leq \Pi \leq \lceil h/2 \rceil.$$

Этот результат является чрезвычайно существенным для теории синтеза схем из пороговых элементов, так как позволяет более точно предсказать сложность таких схем по сравнению с известными подходами [4].

Простейшим однородным настраиваемым каскадом Макхопадхая является каскад из трехвходовых мажоритарных элементов [3] (рис. 14.1).

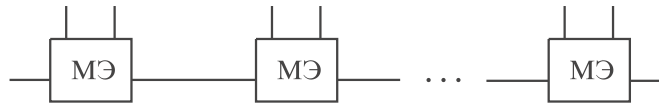


Рис. 14.1

Эти каскады из $n - 1$ элементов при равной доступности прямых и инверсных входных переменных реализуют неповторные пороговые функции n переменных.

При таких входных переменных второй тип однородных каскадов из $h - 1$ трехвходовых мажоритарных элементов (рис.14.2), которые не являются каскадами Макхопадхая, позволяет реализовать произвольные БФ в рассматриваемом базисе из h букв [5, 6].

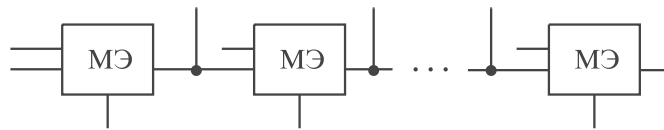


Рис. 14.2

Метод реализации указанного класса формул этим типом каскадов описан в [7].

14.2. Реализация булевых функций каскадами Майтра

Одномерная однонаправленная структура из двухвходовых элементов И, ИЛИ, НЕРАВНОЗНАЧНОСТЬ называется каскадом Майтра [8].

Функциональные возможности таких каскадов рассмотрены в [9]. В [10] были введены обобщенные каскады Майтра, являющиеся древовидными схемами из каскадов Майтра, и исследованы их функциональные возможности при $n \leq 6$. Дальнейшее исследование этих каскадов выполнено в [2].

Известны также настраиваемые однородные каскады Майтра, каждая ячейка которых может быть настроена на реализацию функций указанных выше элементов [3, 11]. Существенно более простым каскадом этого типа является однородный каскад из мультиплексоров «2 в 1», в котором выход предыдущего элемента соединяется с управляющим входом последующего элемента (рис. 14.3).

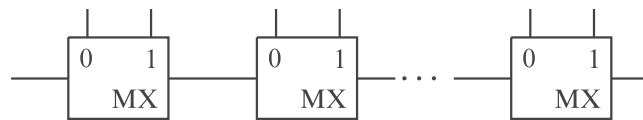


Рис. 14.3

Отметим, что каскады этого типа имеют принципиально другую структуру по сравнению со вторым типом однородных каскадов, которые также построены из мультиплексоров «2 в 1», но не являются каскадами Майтра (рис. 14.4).

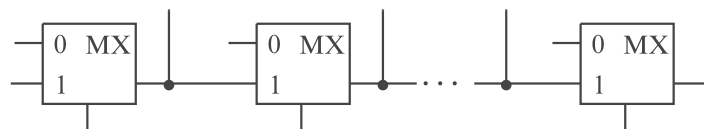


Рис. 14.4

Модификация структуры каскадов приводит к изменению их функциональных возможностей. При этом каскад второго типа из $h - 1$ мультиплексоров «2 в 1» позволяет реализовать произвольные БФ в рассматриваемом базисе из h букв (разд. 15.2) [6].

Каскады и настраиваемые каскады Майтра могут быть разделены на два класса — повторные и неповторные. Если в повторных каскадах одна и та же переменная может подаваться на несколько элементов каскада, то в неповторных каскадах это запрещено. При этом и в тех и в других каскадах для реализации функции НЕРАВНОЗНАЧНОСТЬ на входы одного элемента могут одновременно подаваться переменная и ее инверсия.

Будем называть настраиваемый каскад, реализующий заданную БФУ (БФ), настроенным.

В [12] были установлены следующие пять свойств:

если $f(x_i = 0) = 0$, то $f = x_i f(x_i = 1)$;

если $f(x_i = 1) = 0$, то $f = !x_i f(x_i = 0)$;

если $f(x_i = 0) = 1$, то $f = !x_i \vee f(x_i = 1)$;

если $f(x_i = 1) = 1$, то $f = x_i \vee f(x_i = 0)$;

если $f(x_i = 0) = !f(x_i = 0)$, то $f = x_i \oplus f(x_i = 0) = !x_i \oplus f(x_i = 1)$,

которые определяют возможность разложения заданной БФУ по переменной x_i или ее инверсии. Эти соотношения могут использоваться при построении неповторных каскадов по заданной БФ.

При задании БФУ в виде таблицы истинности применять эти соотношения удобно, проводя разложение БФУ и всех ее остаточных функций по крайней левой входной переменной.

В [6] было установлено свойство столбца значений БФУ, реализуемой неповторным настроенным каскадом Майтра, состоящее в том, что этот столбец должен состоять только из двух типов фрагментов длиной t , где $t = 2, 4, \dots, 2^{n-1}$. Однако метод построения рассматриваемых каскадов по таблицам истинности, обладающим указанным свойством, был не известен.

Ниже предлагается такой метод, который может быть назван мультиплексорным методом с настраиваемым образом декомпозиции. При его применении в БФУ и всех ее остаточных функциях, заданных таблицами истинности, «выделяется» крайняя правая вход-

ная переменная и на каждом шаге составляется и решается относительно подфункции Φ уравнение

$$f = \text{MX}(Q_0, Q_1; \Phi),$$

где f — заданная функция или подфункция; Q_0 и Q_1 — фрагменты длиной два, зависящие от выделенной переменной, из которых состоит столбец значений этой функции или подфункции.

Метод строит для БФУ n переменных, обладающей указанным свойством, настроенный каскад Майтра, число элементов в котором $n - 1$.

Пример 14.1. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0111 \ 0101 \ 0101 \ 0111|^T.$$

Заданная ТИ обладает указанным свойством.

Полагая, что $Q_{01} = x_4$, $Q_{11} = 1$, составим и решим уравнение:

$$f = \text{MX}(x_4, 1; \Phi_1);$$

$$\text{MX}(x_4, 1, x_4, x_4, x_4, x_4, x_4, 1; x_1, x_2, x_3) = \text{MX}(x_4, 1; \Phi_1);$$

$$\Phi_1 = \text{MX}(0, 1, 0, 0, 0, 0, 0, 1; x_1, x_2, x_3).$$

Полагая, что $Q_{02} = x_3$, $Q_{12} = 0$, составим и решим уравнение:

$$\Phi_1 = \text{MX}(x_3, 0; \Phi_2);$$

$$\text{MX}(x_3, 0, 0, 0, x_3; x_1, x_2) = \text{MX}(x_3, 0; \Phi_2);$$

$$\Phi_2 = \text{MX}(0, 1, 1, 0; x_1, x_2).$$

Полагая, что $Q_{03} = x_2$, $Q_{13} = !x_2$, составим и решим уравнение:

$$\Phi_2 = \text{MX}(x_2, !x_2; \Phi_3);$$

$$\text{MX}(x_2, !x_2, x_1) = \text{MX}(x_2, !x_2; \Phi_3);$$

$$\Phi_3 = \text{MX}(0, 1; x_1) = x_1.$$

На основе приведенных соотношений построим настроенный каскад Майтра (рис. 14.5), реализующий заданную БФУ.

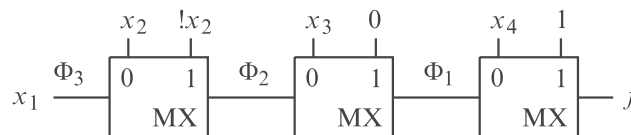


Рис. 14.5

Если столбец функции указанным свойством не обладает, то это не значит, что эта БФУ не будет обладать этим свойством при другом порядке входных переменных в таблице истинности. Вместо

проведения таких перестановок, «направление» которых не известно, автором предлагается по булевой функции построить булеву формулу в базисе $\{\&, \vee, \oplus, !\}$.

Если эта БФ бесповторна, то она реализуется формульным методом бесповторным каскадом Майтра, который, в свою очередь, может быть реализован бесповторным настраиваемым каскадом Майтра.

Пример 14.2. Реализовать БФУ

$$f(x_1, x_2, x_3, x_4) = |0111 \ 1101 \ 0101 \ 0111|^T.$$

Эта таблица истинности указанным свойством не обладает, так как содержит четыре типа фрагментов длиной четыре. Используя карту Карно, построим БФ в базисе $\{\&, \vee, !\}$:

$$f = !x_1 x_3 !x_2 \vee (x_1 \vee !x_3) x_2 \vee x_4.$$

Эта формула может быть преобразована следующим образом:

$$f = (!x_1 x_3 \oplus x_2) \vee x_4.$$

Полученная БФ реализуется формульным методом каскадом Майтра (рис. 14.6).

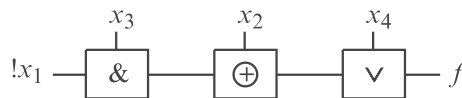


Рис. 14.6

Эта схема применяется для настройки бесповторного настраиваемого каскада Майтра (рис. 14.7).

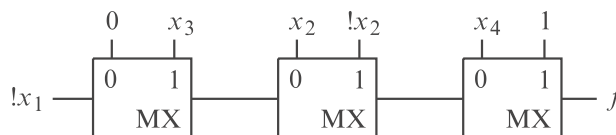


Рис. 14.7

Для построения повторных настроенных каскадов Майтра в [12] было предложено использовать разложения Рида (разд. 3.1.1). При этом шаг разложения может быть выполнен, если $f(x_i = 0)$ или $f(x_i = 1)$ является суммой по модулю два переменных, от которых она зависит, или является одной из переменных или ее инверсией.

В настоящей работе для указанной цели предлагается также применять новые преобразования булевых функций, приведенные в разд. 1.7.

Пример 14.3. Реализовать БФУ, заданную булевой формулой $f = x_1!x_3 \vee !x_1x_3 \vee x_2x_3$ [11].

Так как $f(x_2=0) = x_1 \oplus x_3$, $f(x_2=1) = x_1 \vee x_3$, $f(x_2=0) \oplus f(x_2=1) = x_1x_3$, то, используя первое разложение Рида, получим:

$$f = x_1 \oplus x_3 \oplus x_1x_3x_2.$$

Эта БФ может быть реализована формульным методом повторным каскадом Майтра, по которому, в свою очередь, может быть построен повторный настроенный каскад Майтра, состоящий из четырех мультиплексоров «2 в 1». Так как эта булева формула может быть упрощена:

$$f = x_1(1 \oplus x_2x_3) \oplus x_3 = (!x_2 \vee !x_3)x_1 \oplus x_3,$$

то она может быть реализована формульным методом указанным каскадом Майтра, состоящим из трех мультиплексоров «2 в 1» (рис. 14.8).

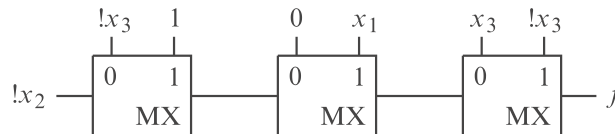


Рис. 14.8

Последняя формула может быть непосредственно получена с помощью первого преобразования Артюхова—Шалыто: $f(x_3=0) = x_1$, $f(x_3=1) = !x_1 \vee x_2$, $!f(x_3=1) = x_1!x_2$, $f = (x_1!x_3 \vee x_1!x_2x_3) \oplus \oplus x_3 = (!x_2 \vee !x_3)x_1 \oplus x_3$.

Пример 14.4. Реализовать булеву функцию, заданную формулой $f = !x_1x_2x_3 \vee x_1!x_3$.

В этом случае $f(x_1=0) = x_2x_3$, $f(x_1=1) = !x_3$, $f(x_1=0) \oplus f(x_1=1) = x_2 \vee !x_3$. При этом на основании второго разложения Рида получим:

$$f = (x_2 \vee !x_3)!x_1 \oplus !x_3.$$

Эта формула реализуется каскадом, представленным на рис. 14.9.

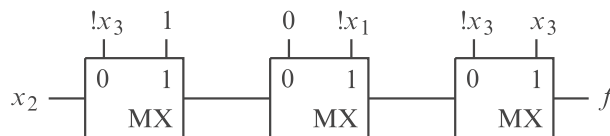


Рис. 14.9

Эта схема может быть построена также и на основе второго преобразования Артюхова—Шалыто: $f(x_3=0) = x_1$, $!f(x_3=0) = !x_1$, $f(x_3=1) = !x_1x_2$, $f = (!x_1!x_3 \vee !x_1x_2x_3) \oplus !x_3 = (x_2 \vee !x_3)!x_1 \oplus !x_3$.

Пример 14.5. Реализовать булеву функцию, заданную формулой

$$f = !x_1 x_2 x_3 \vee x_1 (!x_2 x_3 \vee x_2 !x_3).$$

В этом случае $f(x_3=0) = x_1 x_2$, $f(x_3=1) = x_1 \oplus x_2$, $f(x_3=0) \oplus f(x_3=1) = x_1 \vee x_2$. Используя второе разложение Ридда, получим:

$$f = (x_1 \vee x_2) !x_3 \oplus x_2 \oplus x_1.$$

Эта БФ реализуется каскадом, представленным на рис. 14.10.

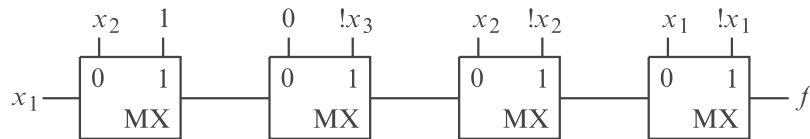


Рис. 14.10

Эта схема может быть построена также и за счет двухкратного применения первого преобразования Артюхова—Шалыто: $f(x_1=0) = x_2 x_3$, $f(x_1=1) = x_2 \oplus x_3$, $!f(x_1=1) = !x_2 \oplus x_3$, $f_1 = !x_1 x_2 x_3 \vee x_1 (!x_2 \oplus x_3)$, $f = f_1 \oplus x_1$, $f_1(x_2=0) = x_1 !x_3$, $f_1(x_2=1) = x_3$, $!f_1(x_2=1) = !x_3$, $f_1 = (x_1 !x_2 !x_3 \vee x_2 !x_3) \oplus x_2 = (x_1 \vee x_2) !x_3 \oplus x_2$.

Таким образом, $f = (x_1 \vee x_2) !x_3 \oplus x_2 \oplus x_1$.

Из приведенных примеров следует, что для класса БФУ, реализуемых повторными каскадами Майтра, разложения Ридда и новые преобразования приводят к одинаковым результатам.

Формулы, которые не могут быть реализованы каскадами Майтра, реализуются схемами, построенными на основе этих каскадов.

Пример 14.6. Реализовать булеву формулу $f = x_1 x_2 \oplus x_3 x_4$.

На рис. 14.11 приведена схема из двухвходовых элементов, являющаяся обобщенным каскадом Майтра, состоящим из двух каскадов, а на рис. 14.12 приведена схема из настроенных каскадов Май-

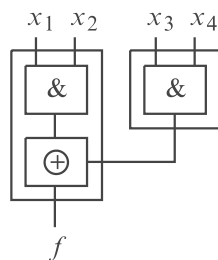


Рис. 14.11

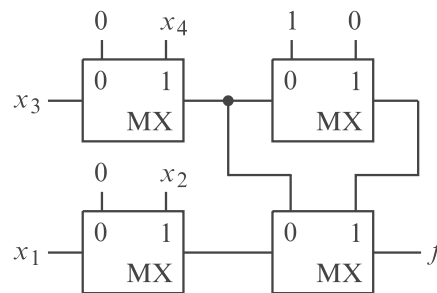


Рис. 14.12

тра, в первом из которых используется дополнительная ячейка, предназначенная для настройки ячейки второго каскада, реализующей элемент НЕРАВНОЗНАЧНОСТЬ, не связанный со входами схемы.

14.3. Реализация булевых формул многоканальными однородными структурами

Известны двухканальные однородные структуры из комбинационных элементов [14], называемые каскадами Шорта [12, 15]. В этих структурах каждая ячейка путем подачи констант на настроечные входы может реализовать произвольную пару БФУ трех переменных, подаваемых на информационный и боковые входы ячейки.

В этой структуре и ее модификациях [12, 15] БФ могут реализовываться в дизъюнктивной нормальной форме, в виде полиномов Жегалкина, а также на основе разложений Рида.

В [12, 15] была предложена трехканальная однородная структура из комбинационных элементов, каждая ячейка которой содержит только один внешний вход, который может использоваться в одних случаях как информационный, а в других — как настроечный вход.

Реализация БФ в этой структуре выполняется при ее представлении в виде полинома Жегалкина.

Для реализации скобочных формул ограниченной глубины в [15] были предложены двух- и трехканальные однородные структуры.

Известны также однородные структуры, состоящие из более простых ячеек [16—20]. Эти структуры, названные ленточными, обладают функциональной полнотой.

Общим недостатком всех перечисленных выше структур и методов вложения БФ в них является отсутствие линейной зависимости числа ячеек в структуре от числа букв в реализуемой БФ.

В настоящем разделе предлагаются однородные структуры, число ячеек в которых равно $h - 1$, где h — число букв в реализуемой формуле, записанной в базисе, для двухместных операций которого выполняется сочетательный закон [3, 21]. При этом число каналов в структуре равно числу уровней каскадов в древовидной схеме максимальной глубины, построенной из двухвходовых элементов этого базиса.

Таким образом, в отличие от известных подходов в однородную структуру вкладываются не «составляющие» реализуемой формулы, а элементы эквивалентной ей древовидной схемы указанного выше типа, реализующей заданную формулу.

В случае, когда доступны только прямые входные переменные, булева формула представляется в виде полинома Жегалкина или скобочной формы, построенной из этого полинома.

Предполагая, что прямые и инверсные входные переменные равнодоступны, будем записывать БФ в одном из двух базисов: $\{\&, \vee\}$ и $\{\&, \vee, \oplus\}$.

Общее правило построения предлагаемых структур состоит в том, что для каждой пары каналов, например $(i-1)$ - и i -го, должны выполняться соотношения

$$\begin{cases} f_{i-1} = x; \\ f_i = y_i * y_{i-1}; \end{cases} \quad \begin{cases} f_{i-1} = y_{i-1} * x; \\ f_i = y_i, \end{cases}$$

а для всех остальных каналов — соотношение

$$f_j = y_j,$$

где $*$ — операции применяемого базиса; $j \neq i-1, i$.

При реализации в каждую ячейку структуры вкладывается один элемент схемы. При этом все элементы каскадов уровня p реализуются в одном канале с номером p . Уровни каскадов нумеруются от входов к выходу, а каналы — сверху вниз.

Это позволяет в двухканальной структуре [22] реализовать двухуровневые каскадные схемы, которым, в частности, соответствуют дизъюнктивные нормальные формы и полиномы Жегалкина, что обеспечивает универсальность предлагаемых структур.

На рис. 14.13, 14.14 в качестве примеров приведены конфигурации, реализуемые путем настройки каждой ячейкой двухканальной и трехканальной структур соответственно.

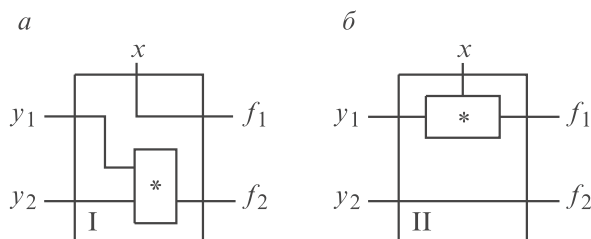


Рис. 14.13

Пример 14.7. Реализовать булеву функцию «точно два из трех», заданную полиномом Жегалкина вида:

$$f = x_1 x_2 \oplus x_1 x_3 \oplus x_2 x_3 \oplus x_1 x_2 x_3.$$

Построим древовидную схему максимальной глубины из двухвходовых элементов $\{\&, \oplus\}$ (рис. 14.15).

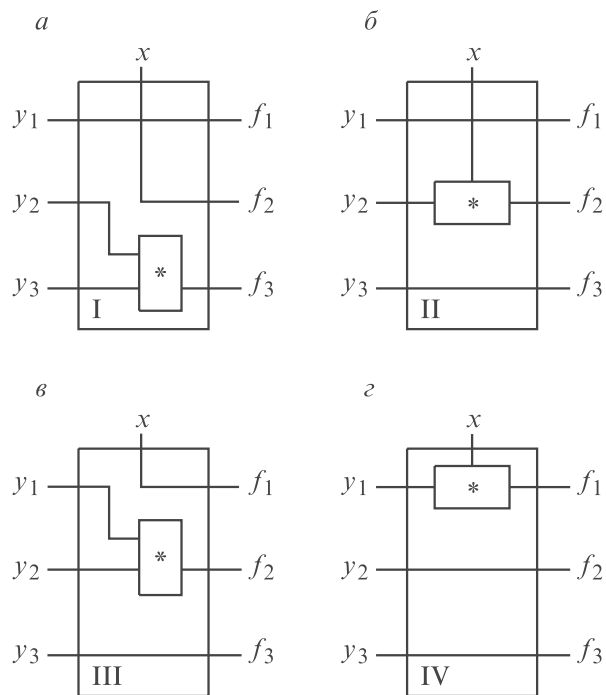


Рис. 14.14

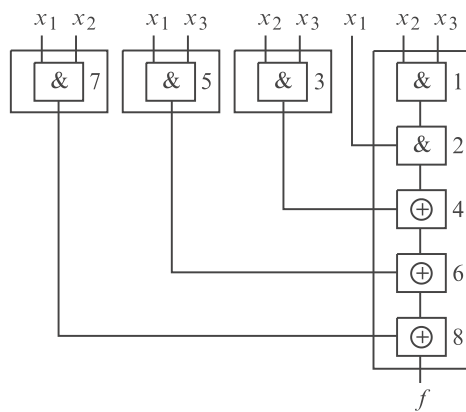


Рис. 14.15

Эта схема из восьми элементов содержит каскады двух уровней (три — первого уровня и один — второго) и реализуется двухканальной однородной структурой из восьми ячеек (рис. 14.16).

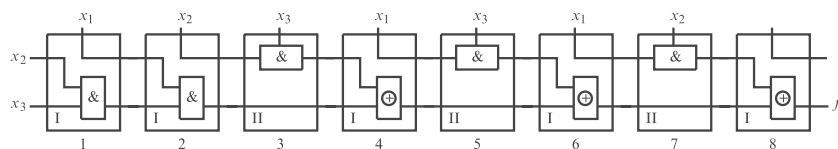


Рис. 14.16

Пример 14.8. Реализовать булеву формулу

$$f = (x_1 \vee x_2)(x_3 \vee x_4) \vee (x_5 \vee x_6)(x_7 \vee x_8).$$

Древовидная схема, реализующая эту БФ, приведена на рис. 14.17.

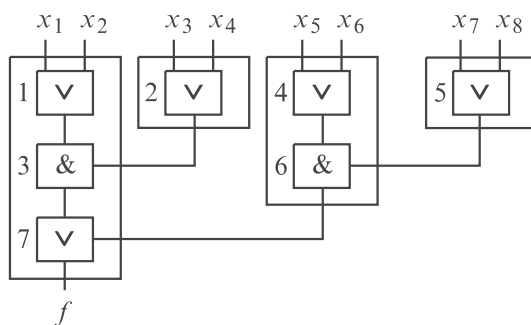


Рис. 14.17

Эта схема из семи элементов содержит каскады трех уровней (по одному — первого и третьего уровней и два каскада второго уровня) и реализуется трехканальной однородной структурой из семи ячеек (рис. 14.18).

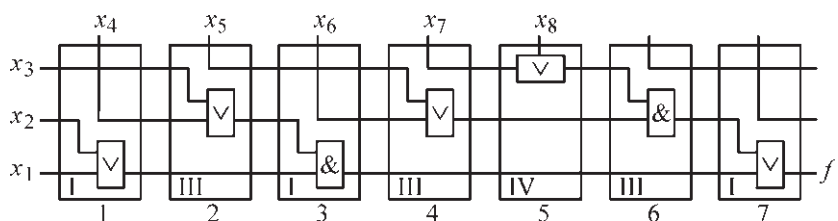


Рис. 14.18

Если в предлагаемых структурах разрешить повороты назад [3], то в k -канальной структуре может быть реализована древовидная схема, содержащая не более $k - 1$ каскадов первого уровня.

Пример 14.9. Реализовать булеву формулу, рассмотренную в предыдущем примере, в предлагаемой двухканальной однородной структуре.

Структура, реализующая эту БФ, приведена на рис. 14.19.

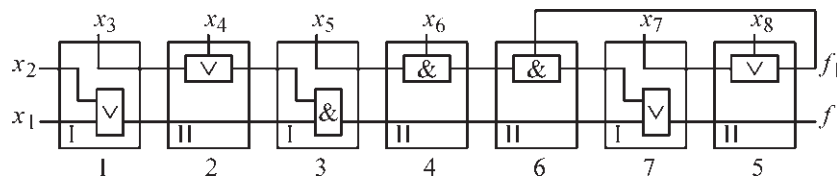


Рис. 14.19

Предложенные структуры позволяют реализовать также и системы булевых формул.

Пример 14.10. Реализовать однородной структурой с минимальным числом каналов систему булевых формул вида:

$$f_1 = (x_1 \vee x_2) x_3, \quad f_2 = x_1 \oplus x_2 \oplus x_3.$$

Так как в данном случае древовидная схема состоит из пяти элементов и содержит три каскада (рис. 14.20), то она может быть реализована трехканальной однородной структурой из пяти ячеек (рис. 14.21).

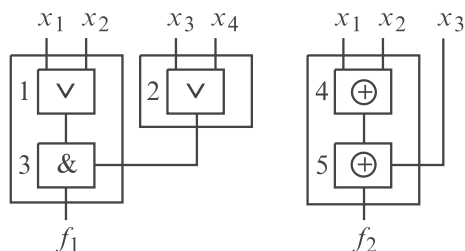


Рис. 14.20

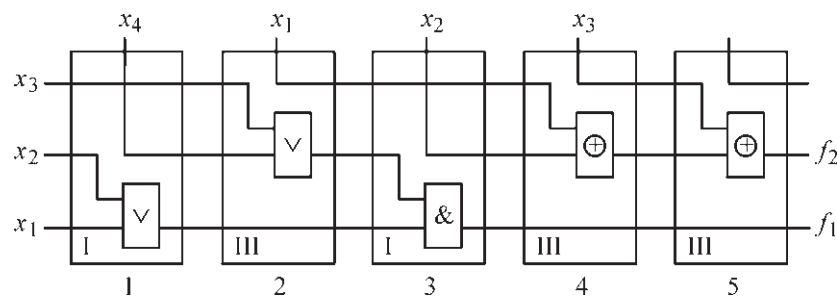


Рис. 14.21

14.4. Реализация булевых формул плоскостными однородными структурами

Известны [23—26] различные плоскостные однородные структуры. Однако такие структуры для реализации произвольных скобочных формул в выбранном базисе в литературе не рассматривались.

В [27] предложена ячейка однородной структуры, которая путем настройки может быть настроена на реализацию одной из четырех конфигураций, представленных на рис. 14.22.

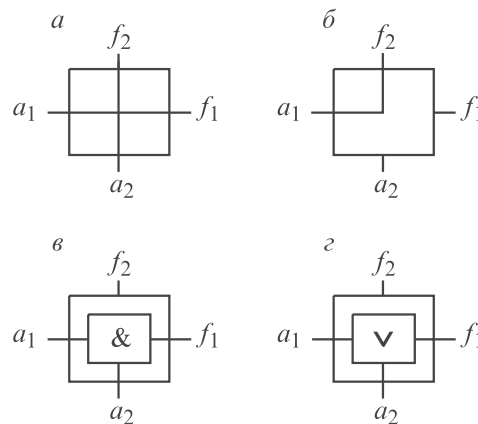


Рис. 14.22

Эта ячейка описывается следующей системой булевых формул:

$$\begin{aligned} f_1 &= a_1!z_1!z_2 \vee 0 \& !z_1z_2 \vee a_1a_2z_1!z_2 \vee (a_1 \vee a_2)z_1z_2; \\ f_2 &= a_2!z_1!z_2 \vee a_1!z_1z_2 \vee 0 \& z_1!z_2 \vee 0 \& z_1z_2. \end{aligned}$$

$$\begin{aligned} \text{При } z_1=0, z_2=0 &\text{ — } f_1=a_1, & f_2=a_2; \\ \text{при } z_1=0, z_2=1 &\text{ — } f_1=0, & f_2=a_1; \\ \text{при } z_1=1, z_2=0 &\text{ — } f_1=a_1a_2, & f_2=0; \\ \text{при } z_1=1, z_2=1 &\text{ — } f_1=a_1 \vee a_2, & f_2=0. \end{aligned}$$

На основе этой ячейки может быть построена однородная плоскостная структура. Булева формула в базисе $\{\&, \vee, !\}$ при равной доступности прямых и инверсных входных переменных реализуется в этой структуре следующим образом.

Заданная булева формула из h букв реализуется древовидной схемой из $h - 1$ двухвходовых элементов И и ИЛИ. В этой схе-

ме выделяются каскады, число которых определяется соотношением [3]

$$1 \leq K \leq [h/2].$$

Построенная схема преобразуется следующим образом:

— каждый выделенный каскад в схеме размещается в одной строке;

— каждый элемент схемы размещается в одном столбце.

Преобразованная схема «вкладывается» в рассматриваемую однородную структуру.

Таким образом, заданная булева формула реализуется плоско-стной однородной структурой, число ячеек в которой определяется соотношением

$$h - 1 \leq L \leq (h - 1) [h/2].$$

Пример 14.11. Реализовать в предлагаемой однородной структуре булеву формулу $f = x_1 x_2 \vee x_3 x_4$.

На рис. 14.23 приведена древовидная схема, на рис. 14.24 — преобразованная схема, а на рис. 14.25 эта схема «вложена» в однородную структуру.

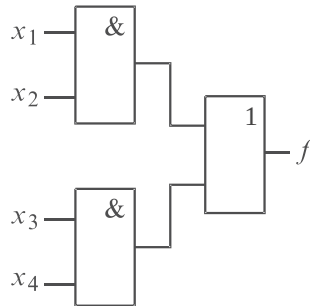


Рис. 14.23

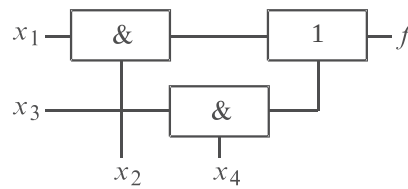


Рис. 14.24

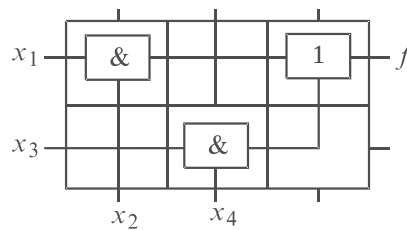


Рис. 14.25

Аналогичный подход может быть применен и для реализации булевых формул, использующих другие двухместные операции. При этом простейшей по числу настроек является ячейка для построения плоскостных однородных структур, реализующих булевы формулы в базисе И—НЕ (ИЛИ—НЕ). Такая ячейка должна настраиваться на реализацию лишь трех конфигураций (рис. 14.26).

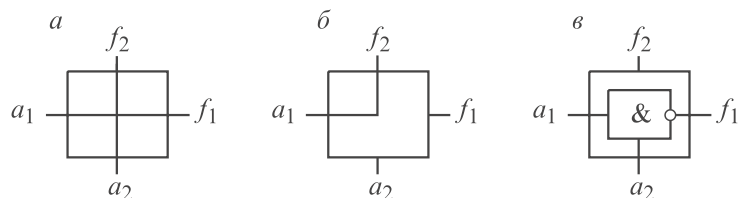


Рис. 14.26

В плоскостной однородной структуре из таких ячеек булева формула $f = x_1 x_2 \vee x_3 x_4 = \neg(\neg(x_1 x_2) \neg(x_3 x_4))$ реализуется, как и в предыдущем случае, с помощью шести ячеек, в то время как булева формула $f = x_1 x_2 x_3 x_4$ требует шести ячеек против трех ячеек в предыдущей структуре.

Рассмотренный подход за счет незначительного усложнения ячеек может быть применен и для совместной реализации систем булевых формул. В [28] предложена ячейка, которая может быть настроена на одну из четырех конфигураций, приведенных на рис. 14.27.

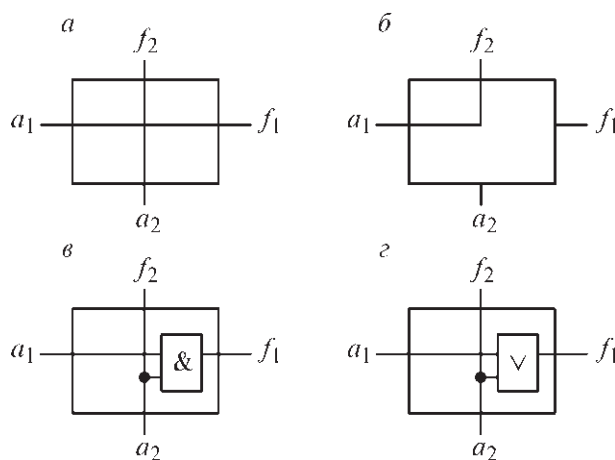


Рис. 14.27

На рис. 14.28 приведена однородная структура из девяти таких ячеек, реализующая систему БФ: $y_1 = x_1 x_2 \vee x_3 \vee x_4$, $y_2 = x_1 x_2 \vee x_3 x_4$.

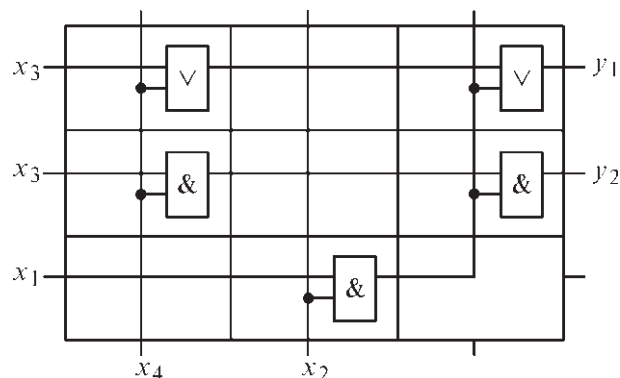


Рис. 14.28

Естественно, что, как и в случае реализации одной БФ, и для системы БФ могут быть предложены ячейки, в которых вместо двухвходовых элементов И и ИЛИ применяются двухвходовые элементы И—НЕ (ИЛИ—НЕ).

Выводы

1. Предложены неповторные однородные каскады Макхопад-хая из $h - 1$ трехвходовых мажоритарных элементов, позволяющие реализовать путем настройки неповторные пороговые формулы из h букв.

2. Показано, что число пороговых элементов, требующихся для реализации произвольной булевой формулы в базисе И, ИЛИ из h букв, удовлетворяет соотношению

$$1 \leq \Pi \leq [h/2].$$

3. Установлено, что для того, чтобы булева функция n переменных могла быть реализована неповторным однородным каскадом Майтра, ее столбец значений должен состоять только из двух типов фрагментов каждой из длин t , где $t = 2, 4, \dots, 2^{n-1}$.

4. Предложены однородные каскады Майтра из мультиплексоров «2 в 1», которые являются наиболее простыми каскадами этого типа из известных.

5. Предложен мультиплексорный метод с настраиваемым образом декомпозиции для реализации неповторными однородными каскадами Майтра булевых функций, столбцы значений которых обладают свойством, указанным в п. 3.

6. Предложено использовать преобразования Артюхова—Шалыто для поиска эффективных реализаций булевых функций повторными однородными каскадами Майтра.

7. Предложены многоканальные линейные однородные структуры из $h - 1$ ячеек для реализации булевых формул в рассматриваемых базисах из h букв.

8. Предложены плоскостные однородные структуры для реализации указанных классов булевых формул с квадратичной (от числа букв) верхней оценкой числа ячеек в них.

Л и т е р а т у р а

1. *Mahapadhyay A.* Unate cell logic // IEEE Trans. on Computers. 1969. N 2.
2. *Артюхов В. Л., Розенблюм Л. Я., Шалыто А. А.* Логические возможности некоторых типов каскадных структур // Сети связи и дискретные устройства управления. М.: Наука, 1976.
3. *Артюхов В. Л., Копейкин Г. А., Шалыто А. А.* Настраиваемые модули для управляющих логических устройств. Л.: Энергоиздат, 1981.
4. *Бутаков Е. А.* Методы синтеза релейных устройств из пороговых элементов. М.: Энергия, 1970.
5. *Артюхов В. Л., Шалыто А. А.* Однородные структуры для реализации булевых формул // Логические методы построения однородных и систолических структур. Труды I Всесоюз. семинара. М.: Ин-т проблем передачи информации, 1988.
6. *Артюхов В. Л., Шалыто А. А.* Реализация булевых формул однородными мультиплексорами и мажоритарными каскадами // Изв. РАН. Теория и системы управления. 1996. № 5.
7. *Артюхов В. Л., Шалыто А. А.* Однородная структура. А. с. СССР № 900279 // Бюл. изобр. 1982. № 3.
8. *Maitra K. K.* Cascaded switching networks of two-input flexible cells // IRE Trans. Elect. Comp. 1964. N 2.
9. *Sklansky I., Korenjak A. I., Stone H. S.* Canonical tributary networks // IEEE Trans. on Computers. 1965. N 6.
10. *Plish D. C., Scidmore A. K.* The number of equivalence classes of functions realizable by tributary networks // IEEE Trans. on Computers. 1966. N 2.
11. *Поспелов Д. А.* Логические методы анализа и синтеза схем. М.: Энергия, 1974.
12. *Варшавский В. И., Мараховский В. Б., Песчанский В. А., Розенблюм Л. Я.* Однородные структуры. Анализ. Синтез. Поведение. М.: Энергия, 1973.
13. *Артюхов В. Л., Шалыто А. А.* Судовые управляющие логические системы. Л.: ИПК СП, 1983.
14. *Шорт Р.* Каскады с матричной структурой из двухканальных элементов // Микроэлектроника и большие системы. М.: Мир, 1967.
15. *Варшавский В. И., Мараховский В. Б., Розенблюм Л. Я.* и др. Синтез схем в однородных структурах // Обзоры по корабельной автоматике. Л.: Судостроение, 1973. Вып. 6.
16. *Битюцкий В. П., Чистов В. П.* Функциональная полнота ленточных структур // Изв. АН СССР. Техн. кибернетика. 1971. № 3.
17. *Битюцкий В. П., Чистов В. П.* Простейшие ленточные структуры // Изв. АН СССР. Техн. кибернетика. 1971. № 6.
18. *Голунков Ю. В.* Несколько замечаний об однородных ленточных структурах // Изв. АН СССР. Техн. кибернетика. 1972. № 6.
19. *Голунков Ю. В.* Функциональная полнота ленточных структур из простейших односторонних каскадов // Изв. АН СССР. Техн. кибернетика. 1973. № 2.
20. *Битюцкий В. П., Ковалин Л. В., Чистов В. П.* О функциональной полноте полосы простейшей однородной структуры // Изв. АН СССР. Техн. кибернетика. 1974. № 5.

21. *Артюхов В. Л., Копейкин Г. А., Шалыто А. А.* О реализации скобочных формул произвольной глубины в линейных однородных структурах из комбинационных элементов // Однородные вычислительные системы и среды. Ч.1. Материалы IV Всесоюз. конф. Киев: Ин-т кибернетики, 1975.
22. *Артюхов В. Л., Шалыто А. А.* Ячейка однородной среды. А. с. СССР № 798804 // Бюл. изобр. 1981. № 3.
23. *Евреинов Э. В., Косарев Ю. Г.* Однородные универсальные вычислительные системы высокой производительности. Новосибирск: Наука, 1966.
24. *Прангизвили И. В., Абрамова Н. А., Бабичева Е. В., Игнатуценко В. В.* Микроэлектроника и однородные структуры для построения логических и вычислительных устройств. М.: Наука, 1967.
25. *Евреинов Э. В., Прангизвили И. В.* Цифровые автоматы с настраиваемой структурой. М.: Энергия, 1974.
26. *Каляев А. В.* Многопроцессорные системы с программируемой архитектурой. М.: Радио и связь, 1984.
27. *Артюхов В. Л., Шалыто А. А.* Ячейка однородной структуры. А. с. СССР № 1092492 // Бюл. изобр. 1984. № 18.
28. *Шалыто А. А.* Ячейка однородной структуры. А. с. СССР № 1264162 // Бюл. изобр. 1986. № 38.

Глава 15

Реализация булевых формул в базисе И, ИЛИ, НЕ линейными бинарными графами

15.1. Формульный метод

В [1] показано, что произвольная нормальная булева формула (БФ) в базисе $\{\&, \vee, !\}$ из h букв может быть реализована бинарным графом (БГ), содержащим $h + 2$ вершины (h — условных и две — операторных: $f = 0$ и $f = 1$).

Из того факта, что число вершин в бинарном графе определяется числом букв в булевой формуле, следует, что для уменьшения числа вершин в графе необходимо минимизировать число букв в формуле, например с помощью классических методов минимизации. Таким образом, эти методы, предложенные в 50-х годах для минимизации релейно-контактных схем, могут применяться и для упрощения программ, реализующих бинарные графы, названных в [1] бинарными программами (БП).

При использовании подхода, изложенного в [1], реализация повторной в указанном базисе БФ сводится к построению БГ для PN -однотипной с ней бесповторной БФ (ББФ) с последующей заменой обозначений переменных. Поэтому в дальнейшем будем рассматривать построение бинарного графа только для ББФ.

В [2] для ББФ был предложен метод построения плоскостных бинарных графов, для реализации которых бинарными программами с одноадресными условными переходами эти графы приходится линеаризовать, что весьма трудоемко [3].

В [1, 4] предложен метод непосредственного построения линейного бинарного графа (ЛБГ) по ББФ. Этот метод назван в [1] формульным. Однако он не обеспечивал планарности ЛБГ для произвольной ББФ. Изложим метод, который строит планарные ЛБГ [5, 6]. Он состоит из следующих этапов.

1. Заданная ББФ, если она содержит групповые инверсии, преобразуется к нормальному виду.

2. Преобразованная ББФ заменяется PN -однотипной с ней положительно монотонной ББФ.

3. n условных вершин располагаются горизонтально и соединяются между собой последовательно. К последней из них подсоединяется операторная вершина с пометкой « $f = 1$ ». При этом образуется остов линейного бинарного графа.

4. Расположим операторную вершину с пометкой « $f = 0$ » после операторной вершины с пометкой « $f = 1$ ».

5. Условные вершины помечим символами букв в соответствии с порядком их следования в ББФ. Это завершает построение остова графа.

6. Осуществим пометку ребер остова нулями и единицами таким образом, чтобы создались условия для прохождения всего остова от его начала до операторной вершины с пометкой « $f = 1$ ». При этом после выполнения условия, соответствующего пометке некоторого ребра остова, должен выполняться переход к следующей вершине. Тем самым оказываются помеченными половина ребер ЛБГ.

7. Если ребро, исходящее из условной вершины, помечено нулем (единицей), то второе исходящее из нее ребро ориентируется вверх (вниз) и помечается единицей (нулем).

8. Каждое вновь введенное исходящее ребро соединим с соответствующей условной или операторной вершиной в зависимости от того, какие буквы ББФ маскируются значением переменной, указанным на рассматриваемом исходящем ребре, считая, что фрагмент ББФ до этой переменной отсутствует.

9. Если в исходную нормальную форму некоторая буква входит с инверсией, то в случае равной доступности прямых и инверсных переменных либо пометка условной вершины, соответствующей этой букве, заменяется на инверсную, либо пометка условной вершины не изменяется, а инвертируется пометка ребер, исходящих из этой вершины. При доступности только прямых переменных имеется лишь вторая из указанных возможностей.

Пример 15.1. Реализовать неповторную булеву формулу $f = x_1 x_2 \vee x_3$.

На рис. 15.1 приведен линейный бинарный граф, построенный изложенным методом.

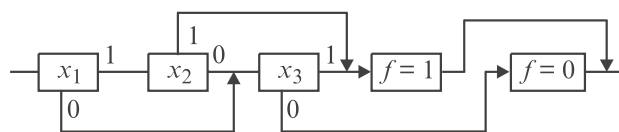


Рис. 15.1

Обратим внимание, что предложенный метод обеспечивает планарность ЛБГ, который может рассматриваться в качестве блока с одним входом и одним выходом.

Изложенный метод может использоваться для реализации не только неповторных, но и повторных БФ из h букв, зависящих от n переменных ($h > n$). При этом с помощью предлагаемого метода реализуется ББФ из h букв, однотипная с заданной, а затем выполняются переобозначения переменных, помечающих условные вершины.

Этот метод может применяться и для реализации БФ, полученных для не полностью определенных БФУ, заданных на q_0 нулевых и q_1 единичных наборах ($q_0 + q_1 = q$). При этом из [7, 8] следует, что

$$h \leq \min(2q_1 + q_0 - 3; \lceil 3q/2 - 2 \rceil).$$

Изложенный метод позволяет также строить формализованным образом логические схемы алгоритмов (ЛСА) [4] для БФ в рассматриваемом базисе.

При этом ЛСА строится в три этапа:

- построение линейного бинарного графа;
- преобразование построенного ЛБГ за счет изменения пометок условных вершин таким образом, чтобы все ребра остова были помечены единицами;
- построение ЛСА по преобразованному ЛБГ.

Для булевой формулы, рассмотренной в примере 15.1, на первом этапе строится ЛБГ (рис. 15.1), который преобразуется в ЛБГ с указанным свойством (рис. 15.2).

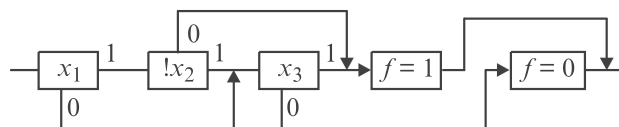


Рис. 15.2

Этому ЛБГ соответствует ЛСА вида:

$$\begin{matrix} 1 & 2 & 1 & 3 & 2 & 4 & 3 & 4 \\ x_1 \uparrow & !x_2 \uparrow & x_3 \uparrow & f \uparrow & w \uparrow & !f \downarrow & f \downarrow, \end{matrix}$$

где w — обозначение безусловного перехода.

В заключение раздела отметим, что при замене в ЛБГ каждой условной вершины переключательным контактом получается планарная контактная схема, реализующая одновременно функцию и ее инверсию. Эта схема может быть упрощена, если требуется реализовать только заданную функцию.

15.2. Проверка правильности построения линейного бинарного графа

Правильность построения ЛБГ может быть проверена несколькими методами.

Первый из них состоит в построении по ББФ таблицы истинности и использовании ее в качестве теста для проверки ЛБГ. Этот метод при больших значениях h весьма трудоемок, так как число тестов при этом равно 2^h .

Второй метод, названный ретрансляцией [9], состоит в свертке ЛБГ с целью получения неповторной булевой формулы, которую он реализует, и сравнения ее с исходной. Свертка выполняется за h шагов, причем на каждом шаге число условных вершин в ЛБГ уменьшается на одну.

Третий, наиболее интересный, метод состоит в замене каждой условной вершины в ЛБГ мультиплексором «2 в 1» с целью получения изоморфной схемы, в которой на выходе каждого элемента может быть записана формула. Ошибка в ЛБГ может быть обнаружена, как только фрагмент формулы, получающийся при «чтении» схемы от входов к выходу, будет отличаться от соответствующего фрагмента заданной ББФ.

Пример 15.2. Проверить правильность построения ЛБГ на рис. 15.1 с помощью третьего из указанных методов.

На рис. 15.3 приведена схема из мультиплексоров «2 в 1».

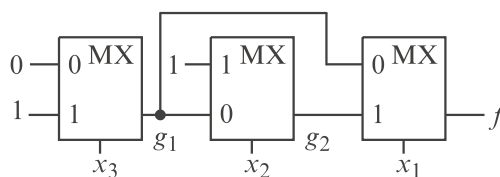


Рис. 15.3

Эта схема описывается следующими соотношениями:

$$g_1 = 0 \& !x_3 \vee 1 \& x_3 = x_3, \quad g_2 = g_1 !x_2 \vee 1 \& x_2 = x_2 \vee x_3, \\ f = g_1 !x_1 \vee g_2 x_1 = x_1 x_2 \vee x_3.$$

Таким образом, ЛБГ построен правильно.

При использовании этого метода восстановление ББФ осуществляется не по подформулам, а по фрагментам формулы, несмотря на наличие в ней в общем случае скобок произвольной глубины. При этом отметим, что фрагмент g_2 формулы f не является ее подформулой. Это свойство было впервые установлено в [10].

Четвертый метод состоит в применении в качестве тестов для проверки программы путей в ее управляющем графе [11]. В

этой работе поиск путей выполняется непосредственно по графу (возможно, ошибочному), который собственно и предстоит проверить.

Для устранения этого недостатка предлагается определять пути не по проверяемому графу, а по спецификации, по которой граф строился. Для ЛБГ спецификацией является реализуемая формула. Покажем, что задача перечисления путей в правильном графе может быть решена на основе преобразований заданной ББФ [12].

15.3. Перечисление путей в линейном бинарном графе

Рассмотрим два метода перечисления путей, которые должны быть в правильно построенном ЛБГ [13].

Первый из них состоит в преобразовании бесповторной булевой формы (ее инверсии) в ортогональную дизъюнктивную нормальную форму (ОДНФ) [12]. ОДНФ [14, 15] формулы перечисляет единичные пути, а ОДНФ инверсии — нулевые.

Конъюнкция не требует ортогонализации, а ортогонализация дизъюнкции выполняется по правилу:

$$g_1 \vee g_2 \vee \dots \vee g_n = g_1 \vee !g_1 g_2 \vee \dots \vee !g_1 !g_2 \dots g_n.$$

Пример 15.3. Перечислить пути в линейном бинарном графе по формуле из примера 15.1.

Построим ОДНФ этой формулы и ее инверсии:

$$\begin{aligned} f &= x_1 x_2 \vee x_3 = x_1 x_2 \vee (x_1 x_2) x_3 = x_1 x_2 \vee (!x_1 \vee !x_2) x_3 = \\ &= x_1 x_2 \vee (!x_1 \vee x_1 !x_2) x_3 = x_1 x_2 \vee !x_1 x_3 \vee x_1 !x_2 x_3; \\ !f &= (!x_1 \vee !x_2) !x_3 = (!x_1 \vee x_1 !x_2) !x_3 = !x_1 !x_3 \vee x_1 !x_2 !x_3. \end{aligned}$$

Каждая конъюнкция в этих ОДНФ соответствует одному пути в ЛБГ, а ранг рассматриваемой конъюнкции (число букв в ней) — длине этого пути. Таким образом, в данном случае

$$S = S_1 + S_0 = 3 + 2 = 5; \quad L = L_1 + L_0 = 7 + 5 = 12,$$

где $S(L)$, $S_1(L_1)$, $S_0(L_0)$ — число (длина) всех, единичных и нулевых путей.

Изложенный метод весьма трудоемок.

Рассмотрим второй метод, который позволяет одновременно ортогонализировать формулу и ее инверсию, что эквивалентно одновременному перечислению всех путей в правильно построенном ЛБГ.

Разложим по Шеннону формулу и все ее остаточные слева направо, последовательно по всем переменным.

Основная особенность этого метода состоит в том, что подформулы, получаемые в результате разложения, которые умножаются на ноль, не уничтожаются. В получаемой в результате полного разложения формуле раскрываются скобки. Пометка конъюнкции нулем (единицей) свидетельствует о принадлежности ее ОДНФ инверсной (прямой) формулы или о том, что этой конъюнкции соответствует нулевой (единичный) путь в ЛБГ.

Пример 15.4. Перечислить пути в линейном бинарном графе по формуле из примера 15.1.

Построим одновременно ОДНФ этой формулы и ее инверсии:

$$\begin{aligned} f &= x_1 x_2 \vee x_3 = !x_1(x_3) \vee x_1(x_2 \vee x_3) = \\ &= !x_1(!x_3 \& 0 \vee x_3 \& 1) \vee x_1(!x_2 x_3 \vee x_2 \& 1) = \\ &= !x_1(!x_3 \& 0 \vee x_3 \& 1) \vee x_1(!x_2(!x_3 \& 0 \vee x_3 \& 1) \vee x_2 \& 1) = \\ &= !x_1!x_3 \& 0 \vee !x_1 x_3 \& 1 \vee x_1!x_2!x_3 \& 0 \vee x_1!x_2 x_3 \& 1 \vee x_1 x_2 \& 1. \end{aligned}$$

Изложенный метод может быть применен также и для перечисления путей в ЛБГ по повторной формуле.

15.4. Использование всех путей в качестве полного теста

Покажем, что только перечисление всех путей в ЛБГ позволяет восстановить всю таблицу истинности (ТИ), а меньшее их количество эту проблему не решает [6].

Этот факт иллюстрируется табл. 15.1, в которой показано, какие минтермы восстанавливает каждый путь. Так как пути ортогональны, то они покрывают минтермы без пересечения. Из этой таблицы следует, что исключение хотя бы одного пути из теста не позволяет восстановить всю таблицу истинности.

Т а б л и ц а 15.1

Путь	Выход ЛБГ	Минтермы
1. $x_1 x_2$	1	1. $x_1 x_2 !x_3$ 2. $x_1 x_2 x_3$
2. $!x_1 x_3$	1	3. $!x_1 !x_2 x_3$ 4. $!x_1 x_2 x_3$
3. $x_1 !x_2 x_3$	1	5. $x_1 !x_2 x_3$
4. $!x_1 !x_3$	0	6. $!x_1 !x_2 !x_3$ 7. $!x_1 x_2 !x_3$
5. $x_1 !x_2 !x_3$	0	8. $x_1 !x_2 !x_3$

Из изложенного следует, что использование в качестве тестов путей, покрывающих все вершины или все дуги управляющего графа [16], в качестве полного теста недостаточно. Только все пути позволяют восстановить всю таблицу истинности.

15.5. Подсчет числа путей в бинарном графе

Подсчет числа путей в БГ можно осуществить с помощью метода Флойда [17], который весьма трудоемок, так как основан на возведении матрицы смежности в соответствующую степень.

Так как в данном случае интерес представляют только пути от крайней левой условной вершины до операторных вершин, то можно применить более эффективный метод, предложенный Дейкстрой [17]. Однако и этот метод весьма трудоемок, так как не учитывает специфику БГ.

Предложим два существенно менее трудоемких метода, первый из которых осуществляет подсчет числа путей от выходов к входу, а второй — в обратном порядке. Первый метод позволяет определять величину S , а второй — величины S_1 и S_0 [18].

Первый метод, при котором БГ рассматривается от выходов к входу, состоит из следующих этапов.

1. Выходы операторных вершин помечаются символом 1.
2. Вход условной вершины помечается числом, которым отмечен выход вершины, связанной с этим входом, а выход условной вершины помечается числом, равным сумме пометок ее входов.
3. Пометка выхода первой условной вершины равна числу путей в БГ.

Пример 15.5. Подсчитать число путей в ЛБГ (рис. 15.1).

Линейный бинарный граф на рис. 15.4 иллюстрирует изложенный метод. Из этого рисунка следует, что $S = 5$.

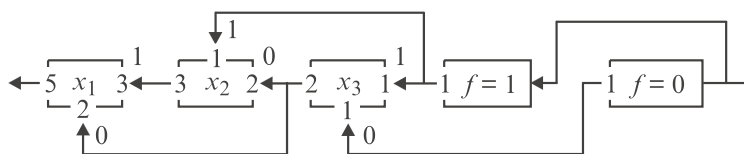


Рис. 15.4

Второй метод, идея которого предложена Акерсом [19], основан на рассмотрении БГ от входа к выходам и состоит в следующем.

1. Вход первой условной вершины помечается символом 1.
2. Пометка входа каждой из остальных вершин графа совпадает с пометкой связанной с ним компоненты графа — выхода условной вершины или точки объединения дуг.

3. Пометка каждого выхода условной вершины совпадает с пометкой ее входа.

4. В каждой точке объединения дуг используется аналог закона Кирхгофа: пометка точки равна сумме пометок выходов условных вершин, связанных с этой точкой.

5. Пометка на входе операторной вершины $f=1$ ($f=0$) равна числу единичных (нулевых) путей в БГ.

Пример 15.6. Подсчитать число путей в ЛБГ на рис.15.1.

Линейный бинарный граф на рис. 15.5 иллюстрирует изложенный метод. Из этого рисунка следует, что $S_1=3$, $S_0=2$.

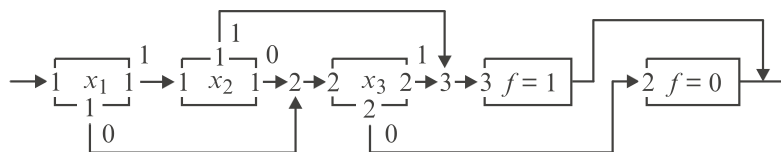


Рис. 15.5

Отметим интересный факт, состоящий в том, что если для положительно монотонной бесповторной пороговой формулы, записанной в порядке неубывания весов, построить ЛБГ и провести разметку с помощью второго метода, то числа, помечающие входы условных вершин, определяют веса соответствующих переменных, а пометка входа операторной вершины $f=0$ определяет порог функции. При этом отметим, что если для БФУ $f(x_1, \dots, x_i, \dots, x_n)$ найдена пороговая реализация $(p_1, \dots, p_i, \dots, p_n; T)$, то для БФУ $f(x_1, \dots, !x_i, \dots, x_n)$ пороговая реализация имеет вид: $(p_1, \dots, -p_i, \dots, p_n; T - p_i)$, а для функции, двойственной f , эта реализация записывается в виде: $\left(p_1, \dots, p_n; \sum_{i=1}^n p_i - T + 1 \right)$ [20].

Пример 15.7. Определить пороговую реализацию для пороговой функции $f = x_3 \vee x_2 !x_1$.

Рассмотрим PN -однотипную БПФ $f = x_3 \vee x_2 x_1$ и запишем ее в порядке неубывания весов: $f = x_1 x_2 \vee x_3$. Из ЛБГ на рис. 15.5 следует, что $p_1=1$, $p_2=1$, $p_3=2$; $T=2$. Таким образом, для заданной БПФ пороговая реализация имеет вид: $(-1, 1, 2; 1)$.

15.6. Подсчет числа путей в блочном бинарном графе

Первый метод осуществляет подсчет путей по блочному ЛБГ, в котором известно число единичных и нулевых путей в каждом блоке, а второй и третий — по реализуемой формуле. Рассмотрим эти методы на примерах.

Пример 15.8. Определить число путей в блочном ЛБГ, реализующем формулу $f = z_1 z_2 \vee z_3$, где $z_1 = x_1 x_2$ ($S_{11}=3, S_{01}=2$), $z_2 = x_3 x_4 \vee x_5 x_6$ ($S_{12}=3, S_{02}=4$), $z_3 = x_1 x_2 x_3$ ($S_{13}=1, S_{03}=3$).

Построим блочный ЛБГ (рис. 15.6) и подсчитаем от входа к выходу число путей в нем. При этом $S_1=9, S_0=18$.

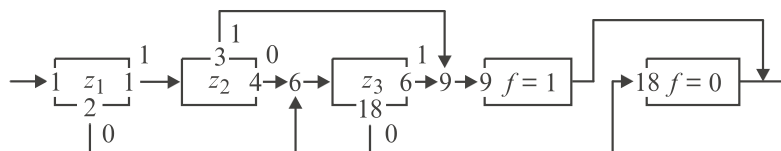


Рис. 15.6

Пример 15.9. Определить число путей в блочном ЛБГ по формуле, приведенной в предыдущем примере.

Построим ортогональные скобочные формы для заданной формулы и ее инверсии: $f = z_1 z_2 \vee (!z_1 \vee z_1!z_2)z_3, !f = (!z_1 \vee z_1!z_2)!z_3$.

Запишем по этим формам соотношения для подсчета числа путей:

$$S_1 = S_{11} S_{12} + (S_{01} + S_{11} S_{02}) S_{13} = 1 \cdot 3 + (2 + 1 \cdot 4) \cdot 1 = 9;$$

$$S_0 = (S_{01} + S_{11} S_{02}) S_{03} = (2 + 1 \cdot 4) \cdot 3 = 18.$$

Пример 15.10. Определить число путей в блочном ЛБГ по формуле, приведенной в примере 15.8.

Выполним разложение Шеннона ББФ и всех ее остаточных. Оно получается при $x_i = z_i$ из соотношения, приведенного в примере 15.4. Исключая в полученном соотношении все нули и единицы, запишем выражение для подсчета числа путей:

$$S = S_{01} (S_{03} + S_{13}) + S_{11} (S_{02} (S_{03} + S_{13}) + S_{12}) = \\ = 2(1 + 3) + 1(4(1 + 3) + 3) = 27.$$

Изложенные выше подходы рассмотрены в [21].

15.7. Оценки числа и суммарной длины путей в линейных бинарных графах

В [22, 23] доказано, что

$$h + 1 \leq S(h) \leq F_{h+2};$$

$$(1/2)h(h + 3) \leq L(h) \leq (1/5)(4hF_{h+1} + 3(h + 1)F_h),$$

где $S(h)$, $L(h)$ — число и суммарная длина путей в ЛБГ, содержащем h условных вершин; F_i — i -е число ряда Фибоначчи, в котором $F_1 = F_2 = 1$ [24].

В табл. 15.2 приведены значения F_h , S_h , S_b , L_h , L_b при $h = 1, \dots, 14$.

Т а б л и ц а 15.2

h	1	2	3	4	5	6	7	8	9	10	11	12	13	14
F_h	1	1	2	3	5	8	13	21	34	55	89	144	233	377
S_h	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_b	2	3	5	8	13	21	34	55	89	144	233	377	610	987
L_h	2	5	9	14	20	27	35	44	54	65	77	90	104	119
L_b	2	5	12	25	50	96	180	331	600	1075	1908	3360	5878	10255

Известно [24], что

$$F_h = \frac{a^h - b^h}{\sqrt{5}}; \quad a = \frac{1 + \sqrt{5}}{2} \approx 1.618; \quad b = \frac{1 - \sqrt{5}}{2}; \quad F_h \approx \left\lceil \frac{a^h}{\sqrt{5}} \right\rceil.$$

Таким образом, при $h > 1$ справедливо неравенство: $S_b(h) < 2^h$.

При $h \rightarrow \infty$ $F_{h+1} \approx aF_h$, и поэтому $L_b(h) \lesssim 2hF_h$.

При $h > 10$ $L_b < 2^h$.

Оценим среднюю длину путей:

$$\frac{1}{2} \left(h + \frac{2h}{h+1} \right) \leq L_{cp}(h) \lesssim \frac{2hF_h}{F_{h+2}} \approx 0.744h.$$

15.8. Анализ оценок числа путей в линейных бинарных графах

Верхние оценки числа и длины путей в ЛБГ получены на основе анализа таких знакопеременных положительно монотонных БПФ (разд. 1.4.3), у которых [25, 26]

$$p_i = F_i, \quad \sum_{i=1}^h p_i = F_{h+2} - 1.$$

Для этого класса булевых формул, записанных в порядке неубывания весов переменных, справедливо соотношение [27]: $T = S_0$.

Для формул этого класса с последней операцией ИЛИ:

$$S_0 = F_h, \quad S_1 = F_{h+1}, \quad S = F_{h+2}, \quad T + \sum_{i=1}^h p_i = F_{h+2} + F_h - 1.$$

Для формул этого класса с последней операцией И

$$S_0 = F_{h+1}, \quad S_1 = F_h, \quad S = F_{h+2}, \quad T + \sum_{i=1}^h p_i = F_{h+3} - 1.$$

Формулы этого класса обладают удивительным свойством: при одном порядке переменных число путей определяется верхней оценкой, а при противоположном порядке переменных — нижней. Таким образом, каждая формула этого класса одновременно является и самой сложной и самой простой по рассматриваемому критерию среди всех формул в указанном базисе из h букв [21]. Другие объекты в дискретной математике, обладающие таким свойством, автору не известны.

Пример 15.11. Определить число путей в линейных бинарных графах, реализующих булевы формулы

$$f = (x_1 x_2 \vee x_3) x_4 \vee x_5 \quad \text{и} \quad f = x_5 \vee x_4 (x_3 \vee x_2 x_1)$$

Так как в этом случае $h = 5$, то $6 \leq S(5) \leq 13$. На рис. 15.7 приведен ЛБГ, построенный по первой формуле, содержащий 13 путей, а на рис. 15.8 приведен ЛБГ, содержащий шесть путей, который построен по второй формуле.

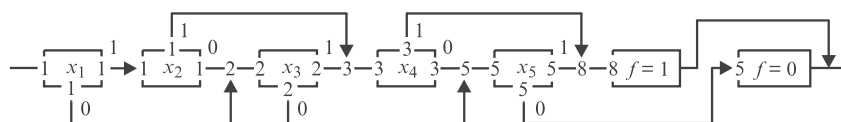


Рис. 15.7

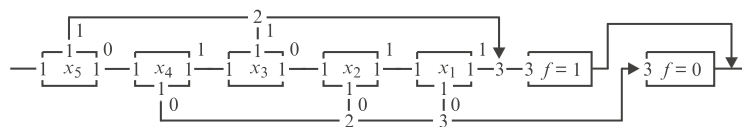


Рис. 15.8

Из рассмотрения ЛБГ (рис. 15.7) можно записать:

$$f = \text{sign}(x_1 + x_2 + 2x_3 + 3x_4 + 5x_5 - 5).$$

Из изложенного следует, что так как верхняя оценка достигается на тех формулах, которые могут быть реализованы также и по нижней оценке, то вопрос о верхней оценке числа путей для «оптимально» записанных формул остается открытым.

15.9. Верхняя оценка числа путей в линейных бинарных графах, реализующих инвариантные формулы

Назовем *инвариантными* булевы формулы, в которых изменение порядка их записи не изменяет числа путей в ЛБГ.

Автор предполагает [23], что среди инвариантных БФ наиболее сложными по рассматриваемому критерию являются дизъюнктив-

ные (конъюнктивные) нормальные формы. В классе ДНФ наиболее сложными являются формулы, которые образованы конъюнкциями, средняя длина которых равна « e ». Эти ДНФ состоят из конъюнкций, содержащих две и (или) три буквы. В [23] доказано, что

$$S_{\text{в}}(h) = \begin{cases} 2^{(h/2)+1} - 1, & \text{если } h = 2, 4, 6, 8, 10, 12, 14, 16; \\ (1/2)(3^{(h/3)+1} - 1), & \text{если } \text{mod}_3 h = 0 \text{ и } h \neq 6, 12; \\ 2 \cdot 3^{(h-1)/3} + 1, & \text{если } \text{mod}_3 h = 1 \text{ и } h \neq 4, 10, 16; \\ 3^{(h+1)/3}, & \text{если } \text{mod}_3 h = 2 \text{ и } h \neq 2, 8, 14. \end{cases}$$

Отметим, что из этой оценки следует, что число путей в ЛБГ, реализующих формулы указанного класса, превышает число клик (максимальных подграфов) в графах Муна—Мозера, являющихся наиболее сложными по числу клик (К) графами. Для графов Муна—Мозера справедливо [29] соотношение:

$$K = \begin{cases} 3^{h/3}, & \text{если } \text{mod}_3 h = 0; \\ 4 \cdot 3^{(h-4)/3}, & \text{если } \text{mod}_3 h = 1; \\ 2 \cdot 3^{(h-2)/3}, & \text{если } \text{mod}_3 h = 2. \end{cases}$$

Отметим, что именно эти оценки получены в разд. 12.2 [30] для определения максимального числа конъюнкций в ДНФ, получающихся в результате раскрытия скобок в бесповторных формулах в базисе $\{\&, \vee, !\}$.

Тот факт, что полученная оценка числа путей превышает оценку числа конъюнкций в указанных ДНФ, объясняется тем, что, как было показано в разд. 15.3, пути соответствуют конъюнкциям в ортогональных ДНФ формулы и ее инверсии.

15.10. Оценки числа путей в линейных бинарных графах для оптимально записанных формул

В [23] было отмечено, что знакопеременные положительно монотонные бесповторные пороговые формулы, для которых достигается верхняя оценка числа путей в ЛБГ при четных $h < 18$, могут быть получены расстановкой скобок в инвариантных ДНФ, состоящих из конъюнкций длиной два. Это приводит к весьма близкой структуре ЛБГ для формул этих классов.

Поэтому можно предположить, что приведенная выше верхняя оценка числа путей, возможно, является и верхней оценкой числа путей в ЛБГ для оптимально построенных формул в базисе $\{\&, \vee, !\}$ при $h > 3$.

Таким образом, можно предположить, что

$$h + 1 \leq S(h) \leq S_{\text{в}}(h) \leq \max(2^{(h/2)+1} - 1, 3^{(h+1)/3}).$$

В заключение раздела отметим, что верхняя оценка числа путей в бинарном графе, реализующем произвольную БФУ h переменных, достигается для булевой функции $f = x_1 \oplus x_2 \oplus \dots \oplus x_h$ и определяется соотношением

$$S_{\text{в}} = 2^h.$$

Таким образом, из оценок, приведенных в настоящей главе, следует, что, например, для произвольной БФ в базисе $\{\&, \vee, !\}$ из 12 букв верхнюю оценку числа путей в ЛБГ, оптимально реализующем эту формулу, можно снизить с $2^{12} = 4096$ до $F_{14} = 377$, а возможно, и до $2^7 - 1 = 127$.

15.11. Минимизация числа и суммарной длины путей в линейных бинарных графах

Пусть задана некоторая неповторная БФ. Обозначим в ней инвариантные подформулы новыми буквами. В преобразованной ББФ в общем случае любая перестановка может изменить число и суммарную длину путей в соответствующем ЛБГ. Например, ББФ $f = (x_1 x_2 x_3 \vee (x_4 \vee x_5) x_6) (x_7 x_8 \vee x_9 x_{10}) \vee x_{11} x_{12}$ может быть записана в виде: $f = (z_1 \vee z_2 z_3) z_4 \vee z_5$.

Для полученной ББФ существует 16 вариантов записи, каждому из которых соответствует свой ЛБГ с числом путей, равным 132, 123, 117, 114, 111, 108, 102, 93, 89, 87, 87, 87, 78, 78, 69, 69 соответственно. При этом 132 пути имеет ЛБГ, построенный непосредственно по исходной формуле. Уже тот факт, что эта величина больше верхней оценки для оптимально записанных формул (при $h = 12$ она равна 127), свидетельствует о том, что заданная ББФ может быть записана более оптимально.

Одной из формул, для которой ЛБГ имеет минимальное число путей, равное 69, является ББФ вида:

$$f = x_{11} x_{12} \vee (x_6 (x_4 \vee x_5) \vee x_1 x_2 x_3) (x_7 x_8 \vee x_9 x_{10}).$$

В [31] предложен метод «взвешивания подформул», обеспечивающий без перебора перестроение заданной ББФ с целью построения ЛБГ с минимальным числом путей. Именно с помощью этого метода и была построена приведенная выше ББФ, реализуемая ЛБГ с минимальным числом путей.

Этот метод минимизирует также и число единичных и нулевых путей в отдельности, а также суммарную длину путей и число точек

объединения дуг в ЛБГ. Оптимальность этого метода связана с бесповторностью реализуемых формул.

Метод основан на том, что если выполняется неравенство

$$\frac{S_{11}-1}{S_{01}} \leq \frac{S_{12}-1}{S_{02}} \leq \dots \leq \frac{S_{1h}-1}{S_{0h}}, \quad (15.1)$$

где S_{1i} и S_{0i} — число единичных и нулевых путей в ЛБГ подформулы f_i ($i = 1, \dots, h$), то минимальное число путей в ЛБГ, реализующем конъюнкцию, достигается при ее записи в виде: $f = f_1 f_2 \dots f_h$, а если выполняется неравенство

$$\frac{S_{01}-1}{S_{11}} \leq \frac{S_{02}-1}{S_{12}} \leq \dots \leq \frac{S_{0h}-1}{S_{1h}}, \quad (15.2)$$

то минимальное число путей в ЛБГ, реализующем дизъюнкцию, достигается при ее записи в виде: $f = f_1 \vee f_2 \vee \dots \vee f_h$.

Одно из следствий этого метода состоит в том, что для минимизации числа путей в ЛБГ бесповторная пороговая формула, соответствующая ему, должна быть записана в порядке невозрастания весов переменных.

Другое следствие относится к классу дизъюнктивных нормальных форм, для которых указанная минимизация обеспечивается, если в каждой формуле конъюнкции записаны в порядке неубывания их длин (рангов) [31].

Пример 15.12. Построить формулу, функционально эквивалентную непороговой ББФ $f = (x_1 \vee x_2)x_3 \vee x_4x_5$, такую чтобы ЛБГ, ей соответствующий, содержал минимальное число путей.

Подформула $f_1 = x_1 \vee x_2$ реализуется ЛБГ с $S_{11} = 2$, $S_{01} = 1$, а подформула $f_2 = x_2$ реализуется ЛБГ с $S_{12} = 1$, $S_{02} = 1$. Так как для подформулы $f_3 = f_1 f_2$ неравенство (15.1) не выполняется, то подформула f_3 должна быть перестроена: $f_3 = f_2 f_1$. Для этой подформулы может быть построен ЛБГ с $S_{13} = 2$, $S_{03} = 2$.

Для подформулы $f_4 = x_4x_5$ может быть построен ЛБГ с $S_{14} = 1$, $S_{04} = 2$.

Так как для формулы $f = f_3 \vee f_4$ неравенство (15.2) выполняется, то формула не перестраивается.

Таким образом, оптимально записанная ББФ имеет вид:

$$f = x_3(x_1 \vee x_2) \vee x_4x_5.$$

Для ЛБГ, построенного по этой формуле, $S = 8$ ($S_1 = S_0 = 4$), в то время как для исходной формулы $S = 11$ ($S_1 = 5$, $S_0 = 6$), а для формулы $f = x_4x_5 \vee x_3(x_1 \vee x_2)$ эти величины имеют следующие значения: $S = 9$, $S_1 = 5$, $S_0 = 4$.

При оптимизации ББФ по рассматриваемому критерию вместо соотношений 15.1 и 15.2 может использоваться также более простой эвристический критерий для определения целесообразности перестановки фрагментов формулы, предложенный автором и А. Ю. Еремеевым.

Этот подход основан на подсчете вероятностей нулевых (Q) и единичных (P) значений в столбце значений таблицы истинности, соответствующей каждой рассматриваемой подформуле, на сравнении этих показателей для подформул, объединенных конъюнкцией или дизъюнкцией, и на перестановке этих подформул по результатам сравнения.

При этом если $Q_1 > Q_2$, то ЛБГ строится по формуле $f = f_1 f_2$, в противном случае — по формуле $f = f_2 f_1$; если $P_1 > P_2$, то ЛБГ строится по формуле $f = f_1 \vee f_2$, в противном случае — по формуле $f = f_2 \vee f_1$.

Применение этого подхода бывает целесообразным, так как для подформулы f подсчет вероятностей P_i и Q_i с помощью первого метода, изложенного в разд. 1.4.2, менее трудоемок по сравнению с подсчетом числа путей S_{1i} и S_{0i} . Однако этот подход является эвристическим, так как в этом критерии используются характеристики, отличные от оптимизируемой.

Пример 15.13. Построить формулу, функционально эквивалентную непороговой ББФ $f = (x_1 \vee x_2)x_3 \vee x_4 x_5$, такую чтобы ЛБГ, ей соответствующий, содержал минимальное число путей.

В таблице истинности $p_i = q_i = 1/2$. При этом для подформулы $f_1 = x_1 \vee x_2$ справедливы соотношения: $Q_1 = q_1 q_2 = 1/4$, $P_1 = 1 - Q_1 = 3/4$, а для подформулы $f_2 = x_3$ — соотношения $Q_2 = q_3 = 1/2$, $P_2 = p_3 = 1/2$.

Так как $Q_2 > Q_1$, то $f_3 = f_2 f_1$. При этом $P_3 = P_2 P_1 = 3/8$, $Q_3 = 1 - P_3 = 5/8$.

Для подформулы $f_4 = x_4 x_5$ справедливы соотношения: $P_4 = p_4 p_5 = 1/4$, $Q_4 = 1 - P_4 = 3/4$.

Так как $P_3 > P_4$, то $f = f_3 \vee f_4$. Таким образом, искомая формула совпадает с найденной в предыдущем примере с помощью точного, но более трудоемкого метода.

Автором и С. Ф. Королюком был предложен еще более простой эвристический критерий для решения поставленной задачи: в конъюнкции на первое место ставится подформула с меньшим значением вероятности числа единиц, а для дизъюнкции — с большим значением этого показателя.

Подсчет этого показателя целесообразно проводить с помощью второго метода, изложенного в разд. 1.4.2.

Пример 15.14. Построить формулу, функционально эквивалентную непороговой ББФ $f = (x_1 \vee x_2)x_3 \vee x_4 x_5$, такую чтобы ЛБГ, ей соответствующий, содержал минимальное число путей.

В таблице истинности $p_i = q_i = 1/2$. При этом для подформулы $f_1 = x_1 \vee x_2$ справедливы соотношения: $Q_1 = q_1 q_2 = 1/4$, $P_1 = 1 - Q_1 = 3/4$, а для подформулы $f_2 = x_3$ справедливо соотношение $P_2 = p_3 = 1/2$.

Так как подформула f_3 — конъюнкция, а $P_2 < P_1$, то $f_3 = f_2 f_1$. При этом $P_3 = P_2 P_1 = 3/8$.

Для подформулы $f_4 = x_4 x_5$ справедливо соотношение $P_4 = p_4 p_5 = 1/4$.

Так как формула f_3 — дизъюнкция, а $P_3 > P_4$, то $f = f_3 f_4$. Таким образом, искомая формула совпадает с найденной в двух предыдущих примерах с помощью более трудоемких методов.

15.12. Реализация повторных формул

Реализация повторных формул с помощью ЛБГ сопряжена для отдельных типов контроллеров (без запоминания значений входных переменных на программный цикл) с проблемой риска — изменениями значений одной и той же входной переменной за время вычисления ЛБГ [31].

Для устранения риска бинарный граф может строиться по булевой формуле с помощью метода каскадов, основанного на разложении Шеннона формулы и всех ее остаточных на каждом шаге по одной из входных переменных [1, 32]. При этом в общем случае вопрос об оптимизации числа путей в БГ остается открытым.

Для БФ с малой повторностью переменных совместное использование формульного метода и метода каскадов может обеспечить построение БГ с минимальным числом путей и без «риска».

Пример 15.15. Реализовать булеву формулу $f = (x_1 x_2 \vee x_3)! x_4 \vee x_4 (x_5 \vee x_6) x_7$.

Выполним разложение заданной формулы по переменной x_4 :

$$f = f_1! x_4 \vee f_2 x_4.$$

При этом если остаточные записать в виде: $f_1 = x_3 \vee x_1 x_2$, $f_2 = x_7 (x_5 \vee x_6)$, и построить для каждой из них ЛБГ, то БГ в целом будет содержать минимальное число путей, равное восьми, а риск при вычислениях по нему отсутствует.

В заключение главы покажем, что существуют БФ, для которых предлагаемый метод построения ЛБГ существенно более эффективен по сравнению с известным методом построения БГ на основе многократного применения разложения Шеннона по одной переменной. В [33] показано, что булева формула $f_1 = x_1 x_2 \vee \dots \vee x_{2n-1} x_{2n}$ при порядке переменных $x_1, x_2, \dots, x_{2n}$ реализуется известным методом БГ, состоящим из $2n + 2$ вершин, в то время как булева формула $f_2 = x_1 x_{n+1} \vee \dots \vee x_n x_{2n}$ при том же порядке переменных реализует-

ся тем же методом БГ, содержащим 2^{n+1} вершин. При этом отметим, что формульный метод для каждой из этих формул строит планарный ЛБГ из $2n + 2$ вершин. Из изложенного также следует, что для многих булевых функций построение БФ позволяет «увидеть» порядок переменных, при котором строится более простой БГ, что не удастся сделать по таблице истинности.

Так как в [34] показано, что число букв в БФ в базисе $\{\&, \vee, !\}$ может зависеть от числа переменных экспоненциально, то в следующей главе рассматриваются методы построения бинарного графа для булевых функций, которые формульным методом реализуются неэффективно. При этом отметим, что в [33] показано, что среди $2n$ функций, описывающих работу параллельного умножителя двух n -разрядных двоичных чисел, существует такая функция, которая вне зависимости от порядка переменных при больших n реализуется БГ, содержащим по крайней мере $2^{n/8}$ вершин.

Выводы

1. Предложен метод реализации произвольной булевой формулы в базисе $\{\&, \vee, !\}$ из h букв планарным линейным бинарным графом (ЛБГ) из $h + 2$ вершин.
2. Предложен метод верификации ЛБГ, состоящий в восстановлении по нему реализуемой булевой формулы. При этом восстановление формулы осуществляется не по подформулам, а по ее фрагментам, которые могут не являться подформулами.
3. Предложены два метода перечисления путей в правильно построенном ЛБГ по заданной булевой формуле.
4. Показано, что только перечисление всех путей в бинарных графах позволяет восстановить всю таблицу истинности булевой функции.
5. Предложены простые методы подсчета числа путей в бинарных графах непосредственно по этим графам.
6. Предложен метод построения пороговой реализации для произвольной положительно монотонной неповторной пороговой формулы с помощью ЛБГ, содержащего максимальное число путей.
7. Получены оценки числа $S(h)$ и суммарной длины $L(h)$ путей в ЛБГ от числа букв h в реализуемой БФ в базисе $\{\&, \vee, !\}$, в которых используются числа Фибоначчи.
8. Определен класс формул, таких что они при одном порядке записи реализуются ЛБГ с минимальным числом путей, а при противоположном порядке записи — ЛБГ с максимальным числом путей.
9. Определена верхняя оценка числа $S(h)$ путей в линейных бинарных графах, реализующих инвариантные (относительно порядка записи) формулы в базисе $\{\&, \vee, !\}$ из h букв. Из этой оценки следу-

ет, что число путей в ЛБГ, реализующих формулы указанного класса, превышает число клик в графах Муна—Мозера, содержащих максимальное число клик среди графов с h вершинами.

10. Высказано предположение, что верхняя оценка числа путей в ЛБГ, реализующих инвариантные формулы, является также и верхней оценкой числа путей в ЛБГ для оптимально построенных произвольных булевых формул в рассматриваемом базисе.

11. Предложен точный метод, обеспечивающий минимизацию числа путей в ЛБГ, реализующем булевы формулы, неповторные в базисе $\{\&, \vee, !\}$. Предложены также два эвристических менее трудоемких метода для решения этой задачи.

12. Изложенные методы могут рассматриваться также как новый подход к построению, минимизации, подсчету и перечислению числа конъюнкций в ОДНФ, широко применяемых в логико-вероятностных методах расчета надежности [14]. При этом число единичных путей в ЛБГ для заданной булевой формулы, неповторной в рассматриваемом базисе, равно числу конъюнкций в однозначно построенной по ней ОДНФ, а для повторной в этом базисе формулы — верхней оценке числа указанных конъюнкций. Число конъюнкций в ОДНФ, построенной по произвольной булевой формуле в указанном базисе из h букв, не превышает числа Фибоначчи с номером $h + 1$.

Л и т е р а т у р а

1. Кузнецов О. П. О программной реализации логических функций и автоматов // Автоматика и телемеханика. 1977. № 7, 9.
2. Еришов А. П. Введение в теоретическое программирование. М.: Наука, 1977.
3. Кириллов А. П., Кузнецов Б. П., Шалыто А. А. Минимизация числа безусловных переходов в бинарных программах // Судостроит. пром-сть. Сер. Автоматика и телемеханика. 1986. Вып. 1.
4. Камынин С. С., Любимский Э. З., Шура-Бура М. Ю. Об автоматизации программирования при помощи программирующей программы // Проблемы кибернетики. М.: Физматгиз, 1958. Вып. 1.
5. Артюхов В. Л., Кузнецов Б. П., Шалыто А. А. Линейные бинарные графы: свойства, характеристики, алгоритмы // Методы и программы решения оптимизационных задач на графах и сетях. Ч. 2. Тез. докл. III Всесоюз. совещ., Ташкент, 1984.
6. Артюхов В. Л., Кузнецов Б. П., Шалыто А. А. Настраиваемые устройства для судовых управляющих систем. Л.: ИПК СП, 1986.
7. Кукинов А. М. Простой метод синтеза скобочных формул для недоопределенных булевых функций // Синтез дискретных автоматов и управляющих устройств. М.: Наука, 1968.
8. Редькин Н. П. О сложности реализации недоопределенных булевых функций // Автоматика и телемеханика. 1969. № 9.
9. Кириллов А. П., Кузнецов Б. П., Шалыто А. А. Метод ретрансляции булевых формул // Вопросы судостроения. Сер. Судовая автоматика. 1984. Вып. 30.
10. Артюхов В. Л., Шалыто А. А. Судовые управляющие логические системы. Л.: ИПК СП, 1984.
11. Липаев В. В. Качество программного обеспечения. М.: Финансы и статистика, 1983.
12. Кузнецов Б. П., Шалыто А. А. Система преобразований некоторых форм представления булевых функций // Автоматика и телемеханика. 1985. № 11.

13. Артюхов В.Л., Кузнецов Б.П., Шалыто А.А. Реализация булевых формул бинарными графами // Методы и программы решения оптимизационных задач на графах и сетях. Ч.2. Тез. докл. IV Всесоюз. совещ. Новосибирск, 1989.
14. Рябинин И. А. Основы теории и расчета надежности судовых электроэнергетических систем. Л.: Судостроение, 1971.
15. Смирнов А. С. Карточный метод ортогонализации функций алгебры логики // Вопросы судостроения. Сер. Судовая автоматика. 1977. Вып.16.
16. Бичевский Я. Я., Борзов Ю. В. Тестирование программ ЭВМ. Рига: Изд-во ЛатвГУ, 1985.
17. Свами М., Тхуласирами К. Графы, сети и алгоритмы. М.: Мир, 1984.
18. Кузнецов Б. П., Шалыто А. А. Тестирование граф-схем бинарных программ // Методы и системы технической диагностики. Математические методы кибернетики и их приложение. Межвуз. науч. сборник. Вып.14, ч. 2. Саратов: СГУ, 1990.
19. Akers S. B. Binary decision diagrams // IEEE Trans. on Computers. 1978. N 6.
20. Боголюбов И. Н., Овсевич Б. Л., Розенблюм Л. Я. Синтез схем из пороговых и мажоритарных элементов // Сети передачи информации и их автоматизация. М.: Наука, 1965.
21. Кузнецов Б. П., Шалыто А. А. Реализация булевых формул линейными бинарными графами. I. Синтез и анализ // Изв. РАН. Техн. кибернетика. 1994. № 5.
22. Артюхов В. Л., Кузнецов Б. П., Шалыто А. А. Оценки сложности реализации булевых формул бинарными графами // Методы и программы решения оптимизационных задач на графах и сетях. Ч.2. Тез. докл. III Всесоюз. совещ. Ташкент, 1984.
23. Кузнецов Б. П., Шалыто А. А. Реализация булевых формул линейными бинарными графами. II. Оценки числа и суммарной длины путей // Изв. РАН. Теория и системы управления. 1995. № 3.
24. Воробьев Н. Н. Числа Фибоначчи. М.: Наука, 1978.
25. Артюхов В. Л., Розенблюм Л. Я., Шалыто А. А. Логические возможности некоторых типов каскадных структур // Сети связи и дискретные устройства управления. М.: Наука, 1976.
26. Фридман А., Менон П. Теория переключательных схем. М.: Мир, 1978.
27. Кондратьев В. Н., Шалыто А. А. Реализация булевых функций одним линейным арифметическим полиномом с маскированием // Автоматика и телемеханика. 1996. № 1.
28. Артюхов В. Л., Кузнецов Б. П., Шалыто А. А., Плошко Н. С. Особенности записи булевых формул при их программной реализации // Вопросы судостроения. Сер. Судовая автоматика. 1984. Вып. 31.
29. Горбатов В. А. Основы дискретной математики. М.: Высш. школа, 1986.
30. Артюхов В. Л., Кузнецова О. С., Шалыто А. А. Оценка функциональных возможностей программируемых логических матриц // Автоматика и вычисл. техника. 1985. № 2.
31. Кузнецов Б. П., Шалыто А. А. Реализация булевых формул линейными бинарными графами. III. Оптимизация числа и суммарной длины путей // Изв. РАН. Теория и системы управления. 1995. № 5.
32. Поваров Г. Н. Математическая теория синтеза контактных $(1, k)$ -полусни-ков // Доклады АН СССР. 1955. № 5.
33. Brayant R. E. Graph-based algorithms for boolean function manipulation // IEEE Trans. on Computers. 1986. N 8.
34. Лупанов О. Б. Асимптотические оценки сложности управляющих систем. М.: Изд-во МГУ, 1984.

Глава 16

Методы построения бинарных графов для автоматов без памяти

16.1. Структурирование линейных бинарных графов

В [1, 2] рассмотрены три подхода к структурированию неструктурированных граф-схем (ГС): дублирование вершин, введение состояний и введение булевых признаков.

Рассмотрим применение этих методов для структурирования линейных бинарных графов (ЛБГ).

16.1.1. Дублирование вершин

Если ЛБГ по всем S путям развернуть в дерево, то получится структурированный БГ (СБГ), сложность которого определяется числом и длиной путей в ЛБГ.

При этом число операторных вершин (ОВ) в СБГ определяется соотношением

$$B_{\text{оп}} = S_0 + S_1 = S,$$

а число условных вершин (УВ) — соотношением

$$B_y = S - 1.$$

Таким образом, общее число вершин в СБГ

$$B = 2S - 1.$$

В предыдущей главе приведено следующее соотношение

$$h + 1 \leq S \leq F_{h+2},$$

поэтому

$$2h+1 \leq B \leq 2F_{h+2}-1,$$

где F_{h+2} — $(h+2)$ -е число Фибоначчи; $F_1 = F_2 = 1$.

Если ЛБГ минимизирована по числу путей [3], то верхняя оценка их числа может быть снижена (разд. 15.9). Соответственно уменьшается и верхняя оценка числа вершин в СБГ.

Пример 16.1. Построить структурированный бинарный граф для формулы $f = x_1 x_2 \vee x_3$.

Для этой булевой формулы $h = 3$, $F_5 = 5$, и поэтому $7 \leq B \leq 9$.

На рис. 16.1 приведен ЛБГ для заданной булевой формулы, а на рис. 16.2 — соответствующий СБГ, в котором число вершин определяется верхней оценкой.

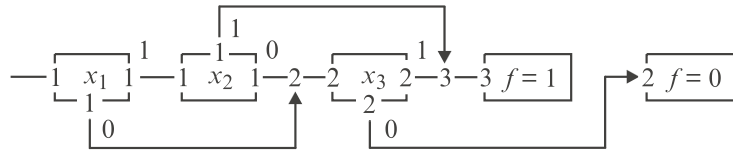


Рис. 16.1

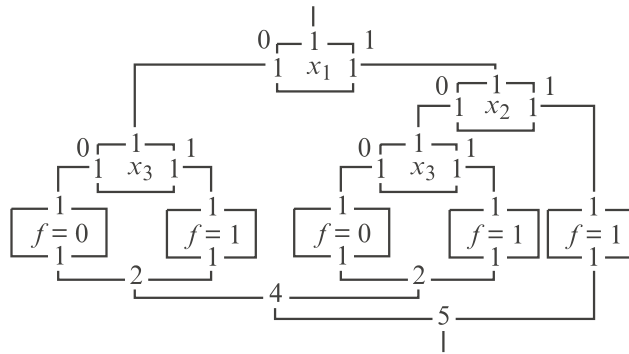


Рис. 16.2

Обратим внимание на тот факт, что в СБГ число вершин каждого типа совпадает с пометкой соответствующих вершин в ЛБГ, осуществляемой при подсчете числа путей от входа к выходу.

СБГ легко программируется без использования операторов `goto`, например с помощью условных выражений:

$$f = x_1 ? (x_2 ? 1 : (x_3 ? 1 : 0)) : (x_3 ? 1 : 0).$$

Пример 16.2. Построить структурированный бинарный граф для формулы $f = x_3 \vee x_2 x_1$.

На рис. 16.3 приведен ЛБГ для заданной булевой формулы, а на рис. 16.4 — соответствующий СБГ, в котором число вершин определяется нижней оценкой.

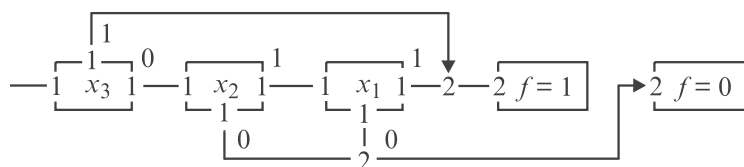


Рис. 16.3

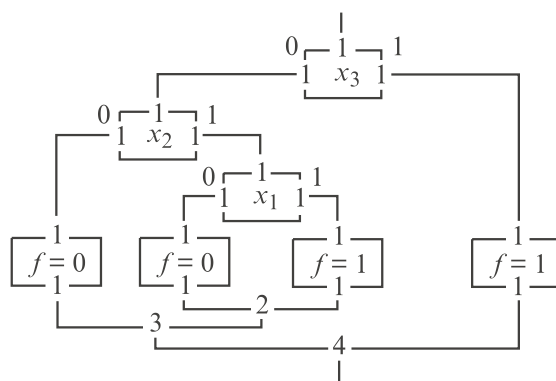


Рис. 16.4

Если после построения СБГ выполнить минимизацию числа вершин за счет подъема вверх ОВ, то число вершин в таких СБГ может быть оценено следующим образом:

$$B_{\text{оп}} = \min(S_0, S_1) + 1; \quad B_y = S - 1; \quad B_m = S + \min(S_0, S_1).$$

Так как $1 \leq \min(S_0, S_1) \leq F_{h+1}$, то $h + 2 \leq B_m \leq F_{h+3}$.

Как и в случае СБГ без минимизации числа вершин, эта верхняя оценка является завышенной.

Изложенный метод [4] не является эффективным, так как приведенные верхние оценки являются экспоненциальными от числа букв.

16.1.2. Введение состояний

При применении этого метода каждой точке объединения дуг в ЛБГ сопоставляется номер. Если при этом построить граф-схему, содержащую дешифратор введенных номеров, к каждому вы-

ходу которого подключить соответствующий фрагмент ЛБГ, в котором ОВ помечены номерами и (или) значениями выходной переменной, а после операторных вершин, помеченных значениями выходной переменной, осуществить сброс номера в других ОВ и это проверить, то, введя обратные связи, получится структурированная ГС (СГС), которая реализует заданную ЛБГ.

Эта СГС является арифметической [5] и эффективно реализуется, например, оператором switch языка СИ. Такая СГС является автоматной, а введенные номера соответствуют состояниям эквивалентного автомата.

Пример 16.3. Построить СГС для формулы $f = (x_1 \vee x_2)x_3 \vee x_4$.

На рис. 16.5 приведен ЛБГ для заданной БФ, содержащий три точки объединения дуг, а на рис. 16.6 — соответствующая СГС.

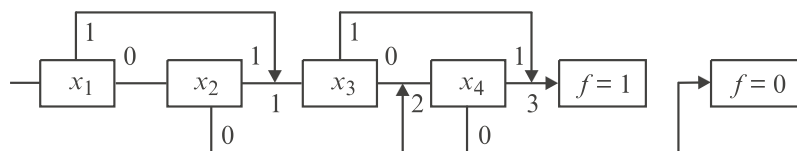


Рис. 16.5

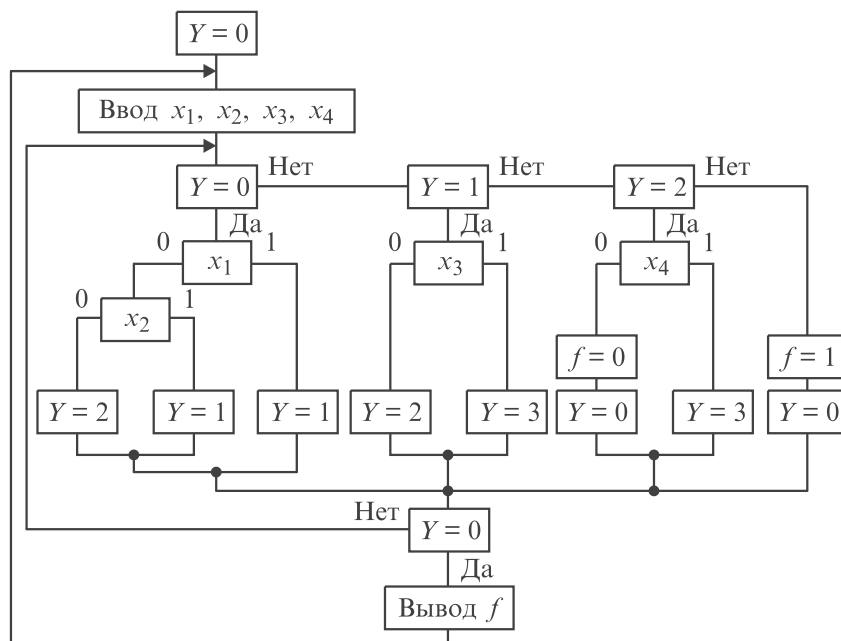


Рис. 16.6

Оценим число вершин в СГС этого типа: число бинарных УВ равно h , число бинарных ОВ равно двум, число арифметических УВ не превышает h , число арифметических ОВ не превышает $2h + 1$.

Таким образом,

$$B \leq 4h + 3.$$

Эта оценка, полученная автором и В. И. Рубиновым в [6], является линейной относительно числа букв в реализуемой БФ, правда при использовании арифметических УВ и ОВ.

16.1.3. Введение булевых признаков

При применении этого метода для каждой точки объединения дуг остова ЛБГ (за исключением точки, связанной с ОВ, если такая имеется) вводится булев признак (промежуточная переменная). После этого ЛБГ разбивается на фрагменты, которые структурируются методом дублирования с введением операторных вершин, в которых осуществляется присвоение булевым признакам значения, равного единице. Фрагменты объединяются с помощью условных вершин, предназначенных для проверки значений введенных признаков. На входе СБГ размещаются операторные вершины, в которых выполняется исходное обнуление этих признаков.

Оценим число вершин в таком СБГ, полагая, что остов ЛБГ имеет m точек объединения дуг, не связанных с операторными вершинами. При этом ЛБГ разбивается на $m + 1$ фрагментов. Так как i -й фрагмент уже не содержит точек объединения, то

$$B_i = 2h_i + 1.$$

При этом число вершин во всех фрагментах определяется соотношением

$$B_{\Phi} = \sum_{i=1}^{m+1} (2h_i + 1) = 2h + m + 1.$$

Число дополнительных условных и операторных вершин определяется соотношением: $B_y = B_{оп} = m$. Таким образом, суммарное число вершин

$$B = 2h + 3m + 1.$$

Так как $0 \leq m \leq h - 2$, то

$$2h + 1 \leq B \leq 5(h - 1).$$

При этом отметим, что уже при $h > 4$ выполняется неравенство

$$5(h - 1) < 2F_{h+2} - 1.$$

Если ЛБГ минимизируется по числу путей, то из [6] следует, что

$$m = \begin{cases} (h-2)/2 & \text{при } h=2k; \\ (h-2)/3 & \text{при } h=2k+1. \end{cases}$$

Следовательно, $m = \lceil (h-2)/2 \rceil$. Таким образом,

$$B_{\text{макс}} = 2h + 3 \lceil (h-2)/2 \rceil + 1;$$

$$B_{\text{макс}} = \begin{cases} 3.5h - 2 & \text{при } h=2k; \\ 3.5(h-1) \leq [3.5h - 2] & \text{при } h=2k+1. \end{cases}$$

Поэтому

$$2h + 1 \leq B_{\text{макс}} \leq [3.5h - 2].$$

Рассмотренный метод структурирования [6] по оценкам сложности более эффективен, чем методы, рассмотренные выше. Он обеспечивает получение линейной верхней оценки числа вершин, которые имеют только двоичные пометки.

Пример 16.4. Структурировать ЛБГ (рис. 16.1) с помощью рассмотренного метода.

В этом случае $h = 3$, $m = 1$, а ЛБГ построен без минимизации числа путей. Поэтому $7 \leq B \leq 10$. На рис. 16.7 приведен СБГ с десятью вершинами.

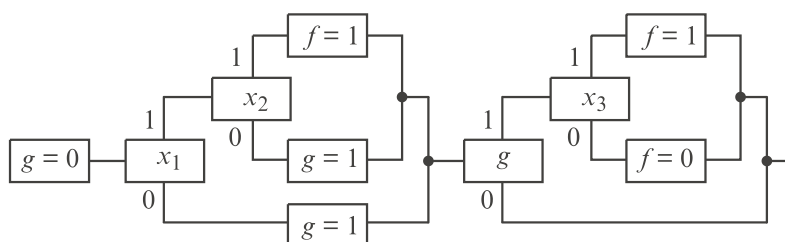


Рис. 16.7

16.1.4. Использование выходной переменной в качестве булева признака

1. Реализация неповторных пороговых формул. Исключить в методе введения булевых признаков проверку промежуточных переменных, соответствующих точкам остова, заменив ее проверкой значений выходной переменной, удастся в тех случаях, когда при каждом разложении Шеннона по одной переменной заданной БФ и всех ее остаточных по крайней мере одна из остаточных равна кон-

станте. Этот класс формул образуют бесповторные пороговые формулы (БПФ), которые обладают указанным свойством при их разложении в порядке невозрастания весов переменных.

Построение СБГ в этом случае выполняется следующим образом. Для заданной бесповторной пороговой формулы строится линейный бинарный граф с максимальным числом путей. Этот ЛБГ читается от выходов к входу и по нему формируются h фрагментов, первый из которых является блоком «полный выбор», а остальные — блоками «неполный выбор». При этом в каждом блоке используется только одна из условных вершин ЛБГ. Эти блоки формируются следующим образом.

Если в ЛБГ дуга, исходящая из условной вершины и не входящая в остов, соединена с нулевой (единичной) операторной вершиной или точкой остова, то соответствующая дуга условной вершины, расположенной в блоке, соединяется с операторной вершиной $f=0$ ($f=1$).

Блоки СБГ соединяются между собой последовательно до тех пор, пока в остова ЛБГ не встретится точка, которая заменяется условной вершиной, проверяющей значение выходной переменной и одна из дуг которой помечается символом, соответствующим символу остова, связанного с точкой, а другая соединяется с выходом.

После построения СБГ за счет выноса «вверх» ОВ (рис. 16.8) по нему в случае, изображенном на рис. 16.8, а, может быть построен линейаризованный СБГ (ЛСБГ), содержащий тоже число вершин.

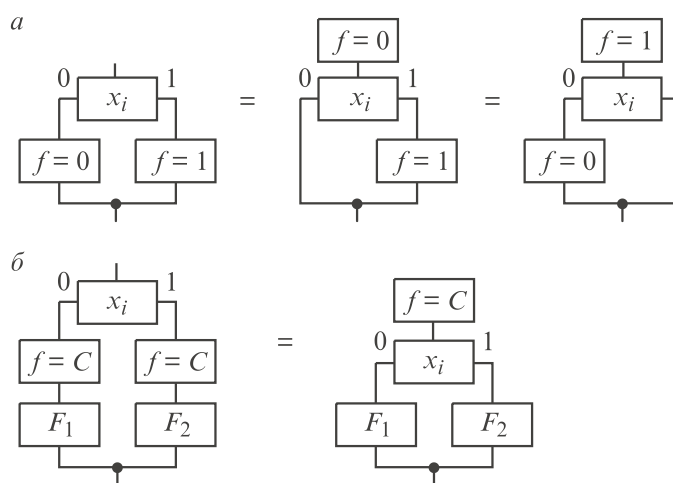


Рис. 16.8

Построенный ЛСБГ обладает длинными переходами, охватывающими несколько последовательно соединенных блоков.

Оценим число вершин в таких графах. В них число УВ с пометками входными переменными равно h , число ОВ равно $h + 1$, а число УВ с пометками выходной переменной — m . Таким образом, общее число вершин

$$B = 2h + m + 1.$$

Так как $0 \leq m \leq h - 2$, то

$$2h + 1 \leq B \leq 3h - 1.$$

Следовательно, предлагаемый метод позволяет для рассмотренного класса формул снизить верхнюю оценку сложности.

Пример 16.5. Построить ЛСБГ для неповторной пороговой формулы, рассмотренной в примере 16.1.

Для этой формулы $h = 3$. Поэтому $7 \leq B \leq 8$.

Построим по заданной формуле ЛБГ с максимальным числом путей (рис. 16.1). Структурированный бинарный граф, построенный по этому ЛБГ, приведен на рис. 16.9, а линейаризованный СБГ — на рис. 16.10. Эти графы содержат число вершин, определяемое верхней оценкой.

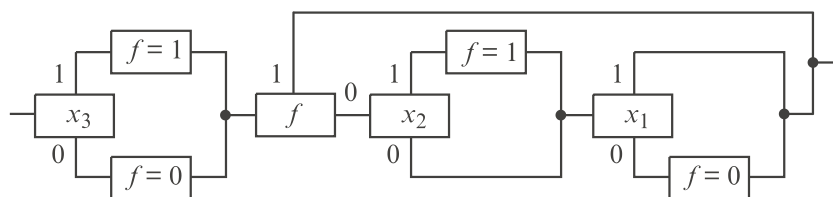


Рис. 16.9

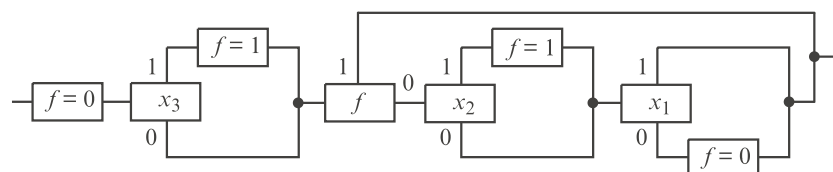


Рис. 16.10

2. Реализация каскадных булевых функций. Рассмотрим класс функций, реализуемых каскадами Майтра (разд. 14.2). В [8] эти функции были названы каскадными. Каждой каскадной функции соответствует каскадная формула, реализуемая ненастраиваемым каскадом Майтра. Функции этого класса могут быть также реализованы СБГ, в которых выходная переменная используется в качестве булева признака, если в построенном настраиваемом каскаде

Майтра каждый из $h - 1$ мультиплексоров «2 в 1» заменить моделирующим его фрагментом СБГ, в котором применяется управляющая конструкция «полный выбор». При таком методе построения каждая промежуточная переменная в каскаде может быть заменена в СБГ выходной переменной, используемой в качестве булева признака.

По сравнению с подходом, изложенным выше, данный подход расширяет класс функций, реализуемых СБГ с указанным свойством, так как БПФ являются лишь подклассом каскадных функций.

Пример 16.6. Построить СБГ для БПФ, рассмотренной в примере 16.1.

Настраиваемый каскад Майтра, реализующий эту формулу, приведен на рис. 16.11, а построенный по нему СБГ — на рис. 16.12.

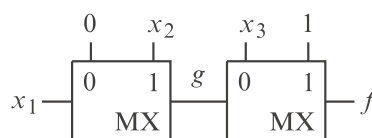


Рис. 16.11

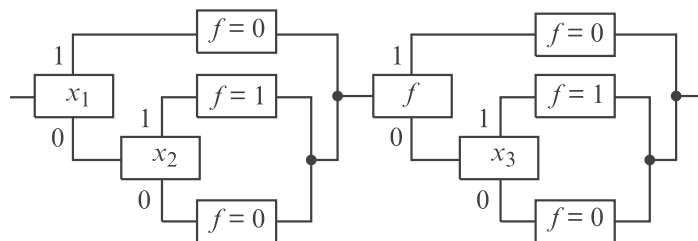


Рис. 16.12

Построенный СБГ содержит девять вершин против восьми у ЛСБГ (рис. 16.10).

Пример 16.7. Построить СБГ для булевой функции, реализуемой формулой

$$f = !x_1 x_3 \vee x_1 !x_3 \vee x_2 x_3.$$

Эта формула не является каскадной. Не является также каскадной и формула

$$f = (x_1 \oplus x_3) \vee x_2 x_3.$$

Однако каскадной является формула $f = (!x_1 \vee !x_2)x_3 \oplus x_1$. На рис. 16.13 представлен настраиваемый каскад Майтра, реализующий

щий эту формулу, а на рис. 16.14 — СБГ, соответствующий этому каскаду.

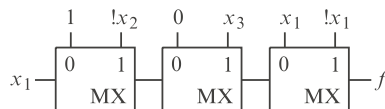


Рис. 16.13

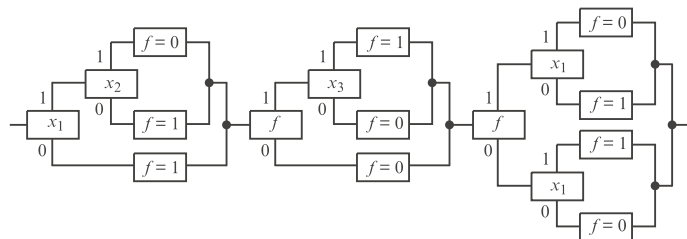


Рис. 16.14

16.2. Метод зависимых фрагментов

16.2.1. Реализация неповторных пороговых формул

Покажем, что для БПФ можно построить линейризованный СБГ, который не будет содержать булевых признаков и будет иметь только короткие переходы [7]. При этом возможны два подхода: структурирование ЛБГ и построение ЛСБГ непосредственно по формуле.

Первый подход состоит в построении линейного бинарного графа с минимальным числом путей и применении метода, изложенного в первом пункте предыдущего раздела.

Так как в таких ЛБГ для БПФ $m = 0$, то для ЛСБГ в этом случае $B = 2h + 1$.

Пример 16.8. Построить ЛСБГ для БПФ $f = x_1 x_2 \vee x_3$.

Линейный бинарный граф с минимальным числом путей приведен на рис. 16.3, структурированный БГ — на рис. 16.15, а линейри-

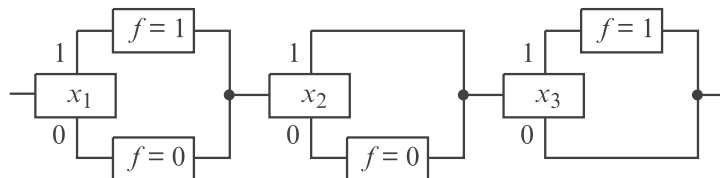


Рис. 16.15

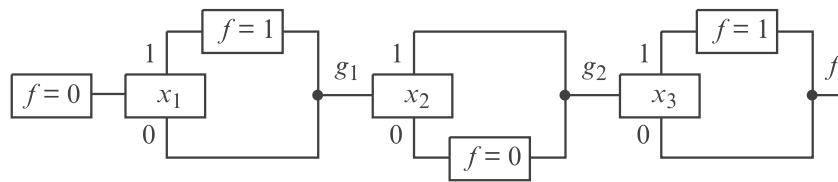


Рис. 16.16

зованный СБГ с короткими переходами, содержащий всего семь вершин, — на рис. 16.16.

В т о р о й п о д х о д состоит в следующем. Запишем заданную БПФ в порядке неубывания весов переменных. После этого заменим подформулы в ней фрагментами ЛСБГ, используя следующие правила:

1. $f = 0 \& !f_2 \vee f_1 f_2 = f_1 f_2$ (рис. 16.17);
2. $f = f_1 !f_2 \vee 1 \& f_2 = f_1 \vee f_2$ (рис. 16.18);
3. $f = f_1 !f_2 \vee 0 \& f_2 = f_1 !f_2$ (рис. 16.19);
4. $f = 1 \& !f_2 \vee f_1 f_2 = !f_1 \vee f_2$ (рис. 16.20).

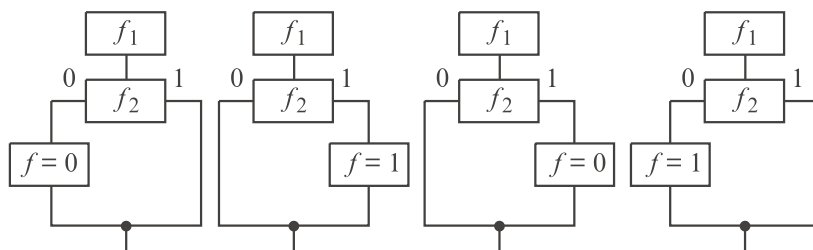


Рис. 16.17

Рис. 16.18

Рис. 16.19

Рис. 16.20

При этом отметим, что в отличие от метода независимых фрагментов (разд. 16.3), в котором правила определяют соответствующие операторы, приведенные соотношения соответствуют правилам алгебры логики.

Из этих правил следует, что решение для первой буквы БПФ неоднозначно: если она безынверсна, то могут применяться правила 1 и 2, если инверсна — правила 3 и 4.

Этот метод строит ЛСБГ с короткими переходами, содержащий $2h + 1$ вершин, в котором булевы признаки отсутствуют, а все блоки соединены последовательно. Он назван методом зависимых фрагментов (МЗФ), так как в построенном с его помощью ЛСБГ блоки, содержащие условные вершины, нельзя менять местами.

Пример 16.9. Построить ЛСБГ для БПФ $f = x_1 x_2 \vee x_3$.

Рассмотрим подформулу $g_1 = x_1$. Полагая $f_1 = (f = 0)$, $f_2 = x_1$, воспользуемся вторым правилом. В результате строится первый фрагмент ЛСБГ (рис. 16.21).

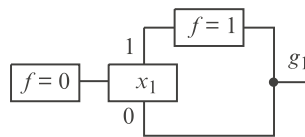


Рис. 16.21

Рассмотрим подформулу $g_2 = g_1 x_2$. Полагая $f_1 = g_1$, $f_2 = x_2$, воспользуемся первым правилом. В результате строится второй фрагмент ЛСБГ (рис. 16.22).

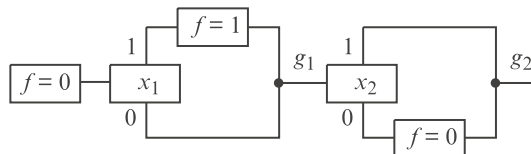


Рис. 16.22

Рассмотрим подформулу $f = g_2 \vee x_3$. Полагая $f_1 = g_2$, $f_2 = x_3$, воспользуемся вторым правилом. В результате строится искомый ЛСБГ (рис. 16.16).

16.2.2. Минимизация числа вершин

Расширим число типов фрагментов, которые можно применять при построении СБГ. Первый тип образуют фрагменты, реализующие конъюнкции, а второй — фрагменты, реализующие дизъюнкции. В этих фрагментах, в отличие от рассмотренных выше, блоки, содержащие условные вершины, являются вложенными.

На рис. 16.23 приведен ЛСБГ, реализующий конъюнкцию $f = x_1 x_2 x_3$, а на рис. 16.24 — дизъюнкцию $f = x_1 \vee x_2 \vee x_3$.

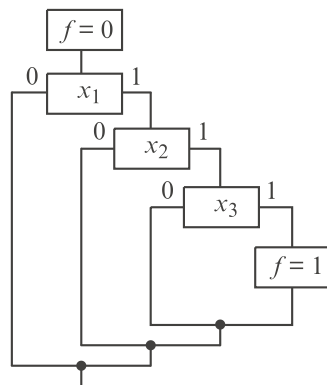


Рис. 16.23

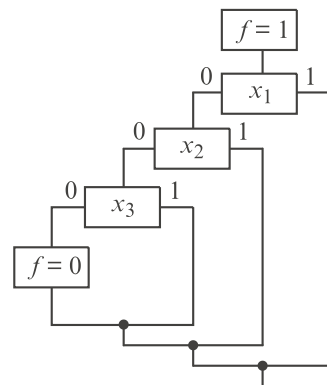


Рис. 16.24

Будем считать, что эти ЛСБГ с $h + 2$ вершинами также содержат только короткие переходы, так как в них отсутствуют «охватываемые» блоки, соединенные последовательно. При этом отметим, что для указанных формул метод, рассмотренный в предыдущем разделе, строит ЛСБГ с более короткими переходами, которые, однако, содержат $2h + 1$ вершин.

Для упрощения построения ЛСБГ может применяться соотношение

$$f = !f'',$$

где f'' — формула, инверсная заданной.

При этом построение ЛСБГ для формулы f выполняется по ЛСБГ для формулы f'' за счет изменения пометок всех операторных вершин на противоположные: $f = 0$ на $f = 1$ и $f = 1$ на $f = 0$.

Пример 16.10. Построить ЛСБГ для БПФ $f = !x_3 (!x_2 \vee !x_1)$.

Так как $f = !(x_3 \vee x_2 x_1)$, то преобразуем указанным образом ЛСБГ на рис. 16.16. Искомая ЛСБГ приведена на рис. 16.25.

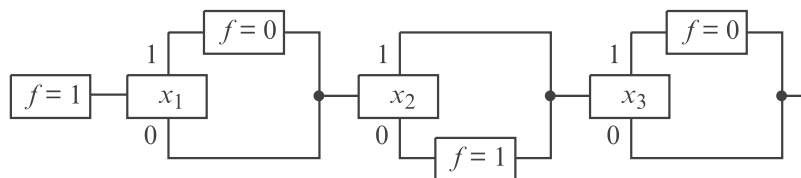


Рис. 16.25

Использование предложенных реализаций дизъюнкций и конъюнкций, а также последнего соотношения позволяет при $h > 1$ уменьшить оценку числа вершин в ЛСБГ, реализующих БПФ:

$$h + 2 \leq B \leq 2h.$$

При этом нижняя оценка достигается для дизъюнкций и конъюнкций, а верхняя оценка учитывает тот факт, что любая БПФ при ее записи в порядке неубывания весов переменных в однотипной с ней положительно монотонной формуле начинается либо с дизъюнкции, либо с конъюнкции, состоящих по крайней мере из двух букв и требующих для своей реализации четырех вершин, а не пяти, как в случае применения метода зависимых фрагментов.

Из изложенного следует, что рассмотренный подход может быть назван методом зависимых фрагментов с оптимизацией (МЗФО).

Пример 16.11. Реализовать БПФ $f = x_1 x_2 (x_3 \vee x_4 \vee x_5)$.

Метод зависимых фрагментов строит ЛСБГ с короткими переходами, содержащий 11 вершин (рис. 16.26).

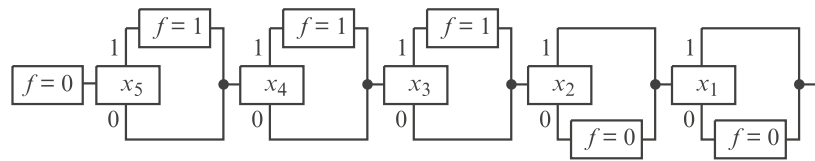


Рис. 16.26

Используем МЗФО. Для этого запишем заданную неповторную пороговую формулу в виде:

$$f = x_1 x_2 (!x_3 !x_4 !x_5) = g_1 !g_2.$$

Для реализации подформулы g_1 применим второе соотношение для МЗФ, а для формулы в целом — третье (рис. 16.27).

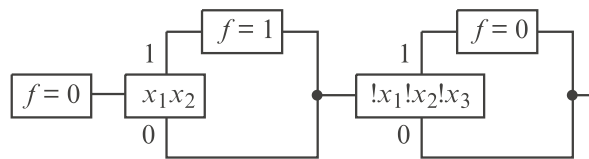


Рис. 16.27

Реализуя конъюнкции, как изложено выше, получим искомый ЛСБГ с короткими переходами, содержащий восемь вершин (рис. 16.28).

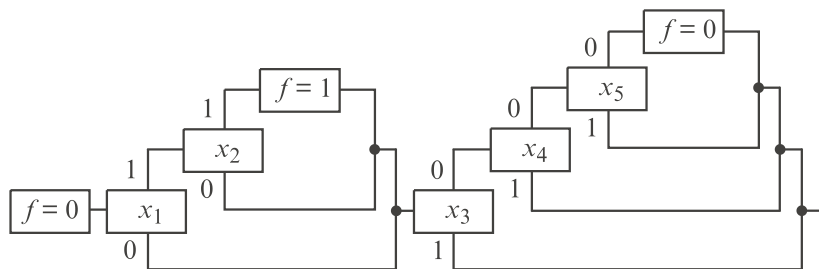


Рис. 16.28

Можно показать, что с помощью МЗФО могут быть реализованы непороговые неповторные БФ (ББФ), если они имеют неповторный образ, т.е. сводятся к БПФ либо после замены одиночными буквами отдельных дизъюнкций или конъюнкций в заданной ББФ или ее инверсии, либо после, по крайней мере, частичного раскрытия скобок в ББФ или ее инверсии. В первом случае строится неповторный ЛСБГ, а во втором — повторный.

Пример 16.12. Реализовать непороговую ББФ

$$f = (x_1 x_2 \vee x_3 x_4) (x_5 \vee x_6).$$

Построим для этой ББФ пороговый образ $f = (z_1 \vee z_2) z_3$ и реализуем его ЛСБГ (рис. 16.29). Реализуя подформулы z_1, z_2, z_3 , построим искомый бесповторный ЛСБГ (рис. 16.30).

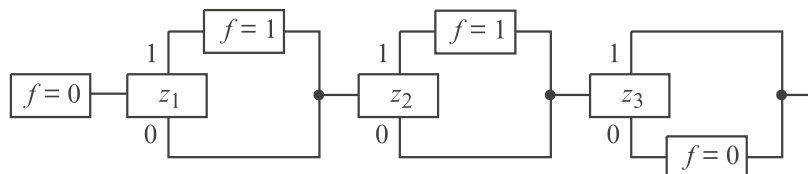


Рис. 16.29

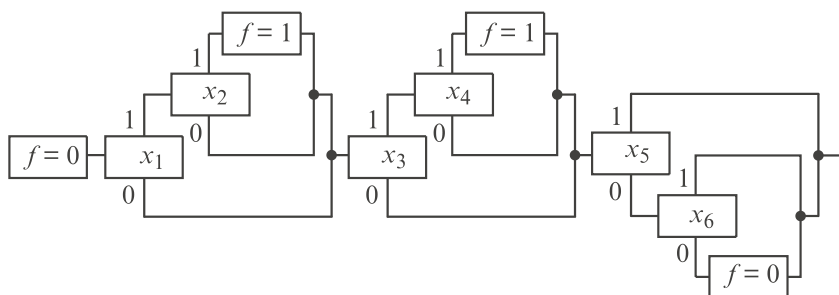


Рис. 16.30

Пример 16.13. Реализовать непороговую ББФ

$$f = (x_1 x_2 \vee x_3) (x_4 x_5 \vee x_6).$$

В данном случае пороговый образ имеет инверсия этой формулы при ее записи в виде: $!f = !((x_1 x_2)!x_3 \vee !x_4!x_6 \vee !x_5!x_6)$. Таким образом, в данном случае имеет место повторная реализация ББФ (рис. 16.31).

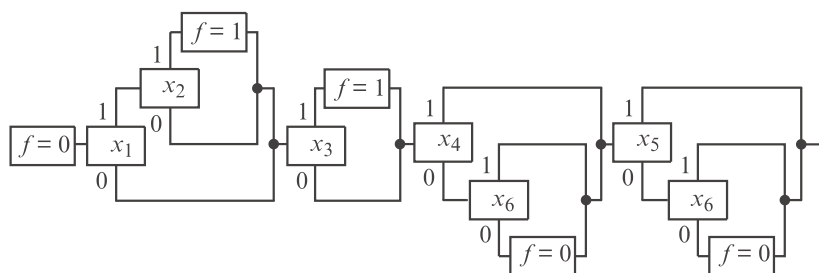


Рис. 16.31

Исследования, выполненные автором, показали, что:
при $h \leq 6$ произвольная ББФ при применении указанных методов в отдельности реализуется бесповторным ЛСБГ;

при $h = 7$ существуют два из 180 представителей PN -типов ББФ, которые этими методами, используемыми в отдельности, не реализуются бесповторными ЛСБГ. Эти представители типов имеют вид:

$$f = (x_1 x_2 \vee x_3 x_4) (x_5 x_6 \vee x_7) \text{ и}$$

$$f = (x_1 \vee x_2) (x_3 \vee x_4) \vee (x_5 \vee x_6) x_7.$$

16.2.3. Смешанный метод

Для уменьшения сложности реализации некоторых формул, и в частности для обеспечения бесповторности реализации при $h = 7$ в классе ЛСБГ с длинными переходами, используем совместно МЗФО и метод независимых фрагментов (МНФ), который будет рассмотрен в разд. 16.3.

Пример 16.14. Реализовать ББФ

$$f = (x_1 x_2 \vee x_3 x_4) (x_5 x_6 \vee x_7).$$

Представим эту формулу в виде:

$$f = (x_1 x_2 \vee x_3 x_4) (!x_7 (!x_5 \vee !x_6)) = (z_1 \vee z_2) z_3.$$

Реализуя пороговый образ с помощью МЗФО, а подформулы z_1 , z_2 , z_3 с помощью МНФ, построим бесповторный ЛСБГ (рис. 16.32).

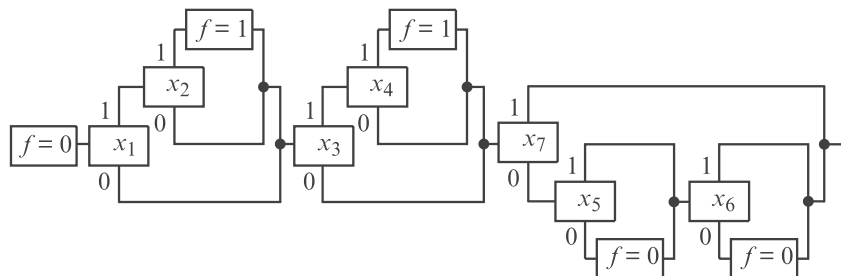


Рис. 16.32

Можно показать, что ББФ $f = (x_1 \vee x_2) (x_3 \vee x_4) \vee (x_5 \vee x_6) x_7$ с помощью смешанного метода также может быть реализована бесповторным ЛСБГ.

Таким образом, из изложенного следует, что при $h \leq 7$ произвольная ББФ реализуется бесповторным ЛСБГ. При этом можно показать, что:

при $h \leq 5$ произвольная ББФ реализуется бесповторным ЛСБГ с короткими переходами, число вершин в которых не превышает $2h$;

В заключение раздела отметим, что на основе правил, рассмотренных в разд. 16.3, каждый из рассмотренных ЛСБГ может быть описан оператором структуры. Эти операторы будут применены в разд. 16.5.5 при реализации систем формул.

Применение этого разложения позволяет упростить ЛСБГ для некоторых формул. Это разложение целесообразно использовать для ББФ вида $f = x_i \vee F$ или $f = x_i F$, если начальная операторная вершина при реализации фрагмента F получена в виде $f = 1$ или $f = 0$ соответственно. При этом к заданной ББФ применяется разложение Шеннона по переменной x_i ; строится ЛСБГ для фрагмента F ; осуществляется подъем вверх одинаковых операторных вершин, связанных с условной вершиной, реализующей указанное разложение.

$$f = ((x_1 \vee x_2 \vee x_3)x_4x_5 \vee x_6)x_7.$$
$$f = 0 \& !x_7 \vee (x_4 x_5 (!x_1 !x_2 !x_3) \vee x_6) x_7.$$

The diagram shows a logic circuit for a 7-variable function f . It consists of several blocks labeled x_1 through x_7 , each with two inputs (0 and 1) and a function block f . The inputs are connected in a hierarchical manner, with x_7 being the leftmost and x_1 being the rightmost. The function blocks f are labeled with $f=0$ or $f=1$.

Этот ЛСБГ содержит 11 вершин, в то время как МЗФО и МНФ в отдельности строят в этом случае граф-схемы с 12 вершинами.

16.2.5. Использование булевых признаков

Снижение трудоемкости построения ЛСБГ при $h \geq 8$ достигается за счет применения булевых признаков. При этом предлагается два подхода: использование «пороговых элементов» и использование «многофункциональных логических модулей».

При применении первого из них заданная ББФ представляется суперпозицией неповторных пороговых формул, каждая из которых реализуется с помощью МЗФО. Построенные фрагменты соединяются последовательно. Можно показать, что в этом случае число вершин в ЛСБГ с короткими переходами определяется соотношением

$$h + 2 \leq B \leq 2(h + [h/2] - 1).$$

Пример 16.16. Реализовать ББФ $f = (x_1 \vee x_2)(x_3 \vee x_4)$.

Представим эту формулу суперпозицией вида:

$$f_1 = x_1 \vee x_2 = !(x_1!x_2), \quad f = f_1(x_3 \vee x_4) = !(x_3!x_4)f_1.$$

Реализуем эту суперпозицию с помощью предлагаемого подхода (рис. 16.34).

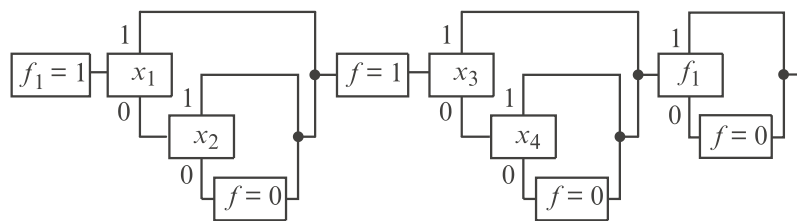


Рис. 16.34

При использовании второго подхода, применяя формульный метод построения схем (гл. 2), представим заданную ББФ в виде суперпозиции неповторных булевых формул, каждая из которых содержит не более q букв. Реализуем каждую из этих формул с помощью ЛСБГ. Соединяя эти ЛСБГ последовательно, получим искомый ЛСБГ. Можно показать, что этот метод строит:

ЛСБГ с короткими переходами, число вершин в которых удовлетворяет соотношению

$$h + 2 \leq B \leq]2.8(h - 1)[;$$

ЛСБГ с длинными переходами, число вершин в которых удовлетворяет соотношению

$$h + 2 \leq B \leq]2.28(h - 1)[.$$

Пример 16.17. Реализовать ББФ

$$f = (x_1 \vee x_2) (x_3 \vee x_4) \vee (x_5 \vee x_6) (x_7 \vee x_8).$$

При $q = 5$ представим заданную ББФ в виде суперпозиции формул:

$$f_1 = (x_1 \vee x_2) (x_3 \vee x_4) = (!x_1!x_2) (!x_3!x_4),$$

$$f = f_1 \vee (x_5 \vee x_6) (x_7 \vee x_8) = f_1 \vee (!x_5!x_6) (!x_7!x_8).$$

При этом заданная ББФ может быть реализована ЛСБГ с короткими переходами, представленным на рис. 16.35.

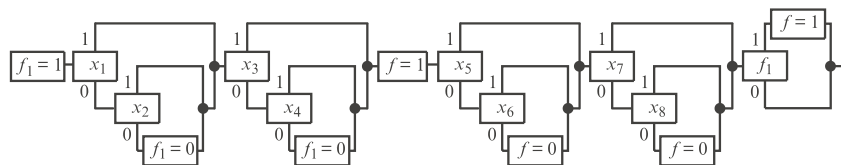


Рис. 16.35

16.2.6. Использование выходной переменной в качестве булева признака

Этот метод строит неповторные ЛСБГ, используя приведенные ниже соотношения.

На рис. 16.36 приведена структура, реализующая соотношение

$$f = f_1!f_1 \vee f_2 f_1 = f_1 f_2.$$

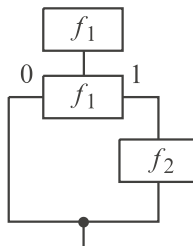


Рис. 16.36

Эта структура упрощается при замене в условной вершине пометки f_1 на f (рис. 16.37).

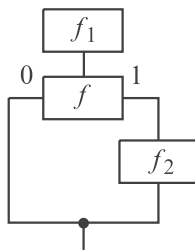


Рис. 16.37

На рис. 16.38 приведена структура, реализующая соотношение:

$$f = f_2!f_1 \vee f_1f_1 = f_1 \vee f_2.$$

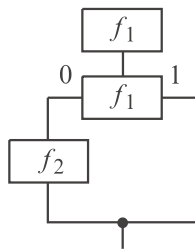


Рис. 16.38

Эта структура упрощается при замене в условной вершине пометки f_1 на f (рис. 16.39).

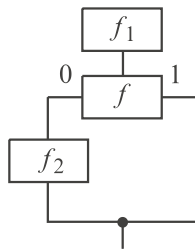


Рис. 16.39

Эти структуры позволяют за счет применения условной вершины с пометкой f присоединять подформулу f_2 к вычисленной подформуле f_1 .

Можно показать, что изложенный метод строит ЛСБГ с длинными переходами, число вершин в которых определяется соотношением

$$B \leqslant]2.25h - 1[.$$

Пример 16.18. Реализовать ББФ, приведенную в предыдущем примере.

Выделим в заданной ББФ две подформулы:

$$f_1 = (x_1 \vee x_2) (x_3 \vee x_4) = (!x_1!x_2) (!x_3!x_4),$$

$$f_2 = (x_5 \vee x_6) (x_7 \vee x_8) = (!x_5!x_6) (!x_7!x_8).$$

Так как в данном случае $f = f_1 \vee f_2$, то используем четвертую структуру (рис. 16.39) и заменим в ней операторные вершины с пометками f_1 и f_2 их реализациями (рис. 16.40).

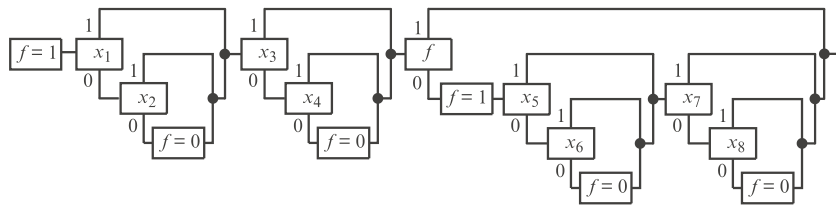


Рис. 16.40

Этот ЛСБГ содержит меньшее число вершин по сравнению с графом, приведенным на рис. 16.35.

16.3. Метод независимых фрагментов

Этот метод позволяет по заданной БФ в базисе $\{\&, \vee, !\}$ определять оператор структуры (ОС), по которому изоморфно строится структурированный бинарный граф (СБГ).

Этот метод в некотором смысле эквивалентен методу построения параллельно-последовательной контактной схемы по БФ в указанном базисе, так как при минимизации ОС по числу букв минимизируется также и число вершин в СБГ.

При этом если оператор структуры содержит в правой части h_2 букв, то и ЛСБГ содержит h_2 вершин.

Если по ОС может быть построен не только ЛСБГ, но и СБГ, то последний содержит число вершин меньшее чем h_2 .

Метод состоит в следующем.

1. Заданная БФ из h букв представляется ДНФ из h_1 букв, состоящей из t конъюнкций ($h_1 \geq h$).

2. В ДНФ конъюнкция $k_i (i = 1, \dots, t)$ помечается символом f .

3. Составляется ОС вида $F = !f | k_1 f | \dots | k_t f$.

4. Оператор структуры изоморфно реализуется с помощью ЛСБГ с короткими переходами, число вершин в котором определяется соотношением

$$h_2 = h_1 + t + 1.$$

Так как в [9] показано, что

$$t \leq 3^{h/3},$$

то

$$h_2 \leq h_1 + 3^{h/3} + 1.$$

Для булевых формул с максимальным числом термов

$$h_1 \leq \lceil h/3 \rceil \cdot 3^{h/3}.$$

Таким образом, для этого класса формул

$$h_2 \leq (\lceil h/3 \rceil + 1) \cdot 3^{h/3} + 1.$$

5. И наконец, если оператор структуры проминимизировать за счет выноса влево за скобки повторяющихся символов входных переменных и (или) их инверсий, то минимизированный ОС (МОС) из h_3 букв ($h_3 \leq h_2$) может быть изоморфно реализован ЛСБГ с длинными переходами, состоящим из h_3 вершин, либо СБГ, число вершин в котором меньше чем h_3 .

При этом отметим, что в МОС будем использовать только круглые скобки, которые могут быть произвольной глубины.

Переход от оператора структуры или минимизированного ОС к ЛСБГ осуществляется с помощью следующих замен фрагментов операторов на управляющие конструкции (УК): фрагмент $F_1 | F_2$ — на УК на рис. 16.41; фрагмент $x_i F$ — на УК на рис. 16.42; фрагмент $!x_i F$ — на УК на рис. 16.43.

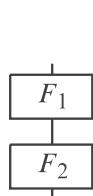


Рис. 16.41

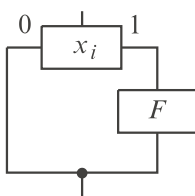


Рис. 16.42

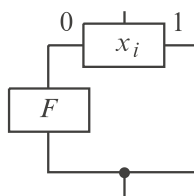


Рис. 16.43

При этом отметим, что если ОС (или МОС) содержит фрагмент $!x_i F_1 | x_i F_2$, то он реализуется структурой, представленной на рис. 16.44.

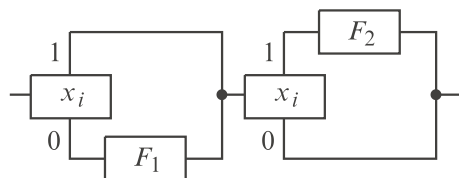


Рис. 16.44

Если в операторе фрагмент $!x_i F_1 | x_i F_2$ заменить на фрагмент $[!x_i F_1 \vee x_i F_2]$, то получим оператор с квадратными скобками. Со- поставим фрагменту с квадратными скобками структуру, представ- ленную на рис. 16.45.

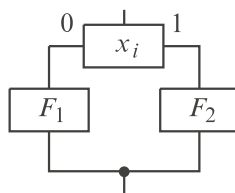


Рис. 16.45

Пример 16.19. Реализовать булеву формулу $f = x_1(x_2 \vee x_3)$.

Оператор структуры в этом случае имеет вид: $F = !f | x_1 x_2 f | x_1 x_3 f$. Этот оператор реализуется ЛСБГ с короткими переходами, содержащим семь вершин (рис. 16.46).

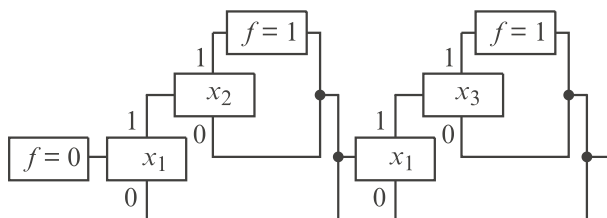


Рис. 16.46

Вынося переменную x_1 влево за скобки, построим МОС: $F = !f | x_1(x_2 f | x_3 f)$, который реализуется ЛСБГ с длинным переходом, содержащим шесть вершин (рис. 16.47).

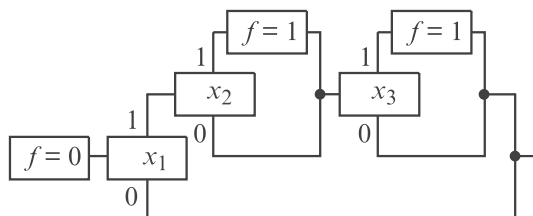


Рис. 16.47

Пример 16.20. Реализовать булеву формулу $f = !x_1!x_2!x_3 \vee x_1 x_2 x_3$.

Оператор структуры в этом случае имеет вид $F = !f | !x_1!x_2!x_3 f | x_1 x_2 x_3 f$ и реализуется ЛСБГ с короткими переходами, содержащим девять вершин.

Если построить оператор структуры с квадратными скобками

$$F = !f \mid [!x_1!x_2!x_3f \vee x_1x_2x_3f],$$

то он может быть реализован СБГ с восемью вершинами (рис. 16.48).

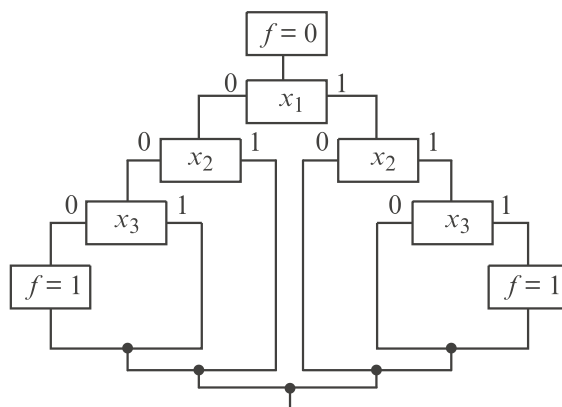


Рис. 16.48

Класс СБГ, который строится рассматриваемым методом, может быть расширен, если этот метод применять не только к ДНФ заданной функции, но и к ДНФ ее инверсии, содержащей t_1 конъюнкций. При этом оператор структуры ДНФ инверсии записывается в виде:

$$F_{\text{и}} = f \mid k_1^n !f \mid \dots \mid k_{t_1}^n !f.$$

Пример 16.21. Реализовать булеву формулу $f = x_1 x_2 x_3$.

В этом случае оператор структуры имеет вид $F = !f \mid x_1 x_2 x_3 f$ и реализуется ЛСБГ с короткими переходами, представленным на рис. 16.23. Тело этого графа состоит из вложенных управляющих конструкций «неполный выбор».

Если построить оператор структуры ДНФ инверсии ($F_{\text{и}} = f \mid [!x_1!f \mid !x_2!f \mid !x_3!f]$), то этот оператор реализуется ЛСБГ с еще более короткими переходами, так как в нем все блоки соединены только последовательно (рис. 16.49).

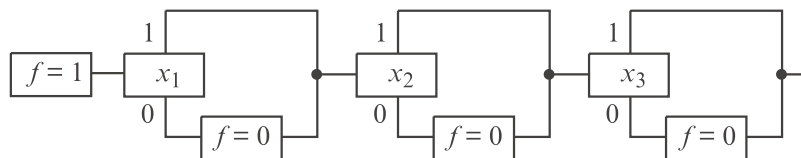


Рис. 16.49

В заключение отметим, что выше было показано как по оператору структуры, в частности, построить ЛСБГ. Естественно, может быть выполнен и обратный переход: по ЛСБГ, построенному любым методом, всегда может быть построен оператор структуры.

Анализ реализаций БПФ и их инверсий, построенных с помощью метода независимых фрагментов, показал, что для каждой пары таких формул суммарное число вершин в ЛСБГ, их реализующих, удовлетворяет соотношению

$$B = 3(h + 1).$$

Изложенный метод назван методом независимых фрагментов, так как правильность построения с его помощью бинарных графов не зависит от порядка расположения последовательно соединенных блоков, содержащих условные вершины.

Идея метода и его название были предложены в [10], а используемый математический аппарат и возможность его применения к ДНФ инверсии заданной функции — в [7, 11].

16.4. Методы построения бинарных графов для одной булевой функции

16.4.1. Распределительный метод

Предложим метод, который позволяет строить бинарный граф (БГ), в котором уровни условных вершин, рассматриваемые от корня графа, соответствуют входным переменным в таблице истинности (ТИ), читаемым в ней справа налево.

Метод состоит в распределении в условной вершине x_i двух частей столбца значений или его остатка. При этом часть, соответствующую $x_i=0$, образуют значения функции или ее остатка с четными номерами, а часть, соответствующую $x_i=1$, образуют значения функции или ее остатка с нечетными номерами.

Метод строит бинарные графы, в которых на каждом пути от входа к выходу каждая входная переменная встречается в условной вершине не более одного раза, а каждый путь завершается в операторной вершине.

Пример 16.22. Реализовать булеву функцию $f(x_3, x_2, x_1) = |0011\ 1101|^T$.

Построение искомого бинарного графа с семью вершинами выполнено на рис. 16.50.

Изменим порядок входных переменных в таблице истинности на противоположный.

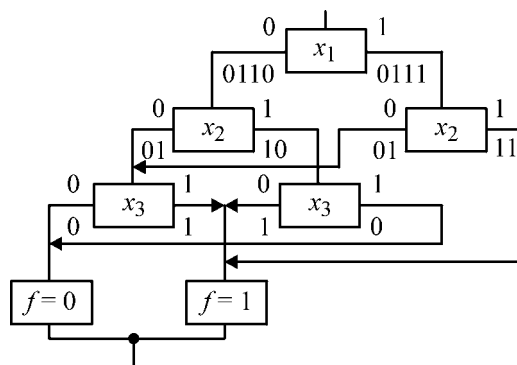


Рис. 16.50

Пример 16.23. Реализовать булеву функцию $f(x_1, x_2, x_3) = |0110\ 0111|^T$.

Построение искомого бинарного графа с шестью вершинами выполнено на рис. 16.51.

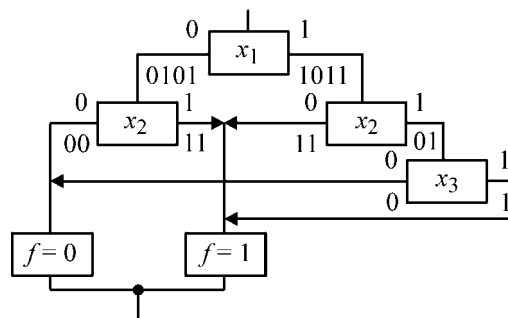


Рис. 16.51

Таким образом, перестановка входных переменных в таблице истинности позволила упростить бинарный граф.

16.4.2. Канонический метод Блоха

Рассмотрим метод [12], который позволяет строить БГ, в котором уровни УВ, рассматриваемые от корня графа, соответствуют входным переменным в ТИ, читаемым в ней слева направо.

Этот метод состоит из четырех этапов:

— построение канонической структуры (КС) по таблице истинности;

— кодирование кустов КС цифрами. При этом кусты первого уровня кодируются следующим образом: $|00|^T \rightarrow 0$; $|01|^T \rightarrow 2$; $|10|^T \rightarrow 3$; $|11|^T \rightarrow 1$. После этого если корни куста помечены одина-

ковыми цифрами, то и его вершина помечается той же цифрой. В противном случае для пометки вершины применяется следующая по значению цифра, которая при кодировании предыдущих кустов еще не была использована;

- объединение кустов, помеченных одинаковыми цифрами;
- построение по минимизированной КС изоморфного БГ.

При этом отметим, что если максимальный номер в КС равен N , то число вершин в построенном по ней бинарном графе определяется соотношением

$$B = N + 1.$$

Пример 16.24. Реализовать булеву функцию $f(x_1, x_2, x_3) = |0110\ 0111|^T$.

На рис. 16.52 приведена КС с $N = 6$, а на рис. 16.50 — искомый бинарный граф, содержащий семь вершин.

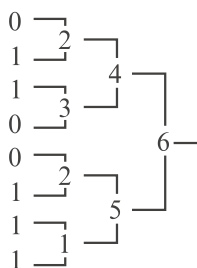


Рис. 16.52

Пример 16.25. Реализовать булеву функцию $f(x_3, x_2, x_1) = |0011\ 1101|^T$.

На рис. 16.53 приведена КС с $N = 5$, а на рис. 16.51 — искомый бинарный граф, содержащий шесть вершин.

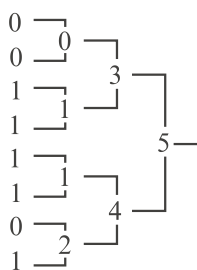


Рис. 16.53

Из изложенного следует, что применение канонического и распределительного методов к одной и той же таблице истинности позволяет без изменения порядка переменных в ней строить два бинарных графа и выбирать простейший из них.

Канонический метод позволяет строить также структурированные БГ, число вершин в которых определяется соотношением

$$2n + 1 \leq B \leq 3 \cdot 2^{n-1}.$$

После построения СБГ он может быть упрощен за счет подъема вверх тех ОВ, которых в построенном графе больше.

Пример 16.26. Реализовать БФУ

$$f(x_1, x_2, x_3) = |0111 \ 1111|^T = x_1 \vee x_2 \vee x_3.$$

Построение СБГ по КС выполнено на рис. 16.54. После выноса ОВ с пометкой $f=1$ вверх получается СБГ с минимальным числом вершин, равным пяти (рис. 16.24).

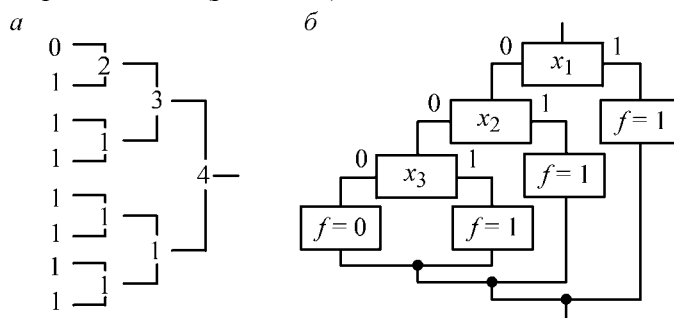


Рис. 16.54

В заключение раздела отметим, что если БФУ $f(x_1, x_2, x_3) = |0000 \ 0111|^T$ реализуется рассматриваемым методом бесповторным БГ, состоящим из пяти вершин, который изоморфен планарному ЛБГ, построенному по формуле $f = x_1(x_2 \vee x_3)$, то эта БФУ, записанная как $f(x_3, x_1, x_2) = |0001 \ 0011|^T$, реализуется этим методом непланарным повторным БГ, состоящим из шести вершин.

16.4.3. Реализация булевых функций линейными бинарными графами

В разд. 15.1 показано, что произвольная ББФ в базисе $\{\&, \vee, !\}$ может быть реализована как бесповторным планарным ЛБГ, так и бесповторной параллельно-последовательной контактной схемой.

Известно [13] также, что некоторые повторные БФ могут быть реализованы бесповторными контактными схемами, которые являются мостиковыми.

Покажем, что существуют повторные БФ, реализующиеся бесповторными ЛБГ, которые не являются планарными при рассмотрении их в качестве блоков с одним входом и одним выходом [3].

Пример 16.27. Реализовать повторную булеву формулу $f = !x_1x_2 \vee (x_1 \vee x_3)x_4$ бесповторным ЛБГ.
Эта БФ соответствует булевой функции

$$f(x_1, x_2, x_3, x_4) = |0001\ 1111\ 0101\ 0101|^T.$$

Построим по этой БФУ каноническую схему (рис. 16.55) и преобразуем ее в бинарный граф (рис. 16.56).

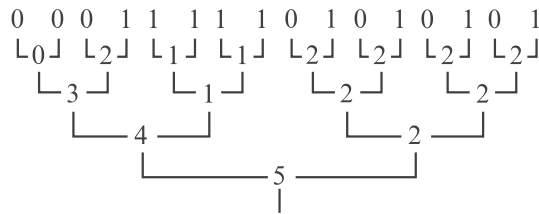


Рис. 16.55

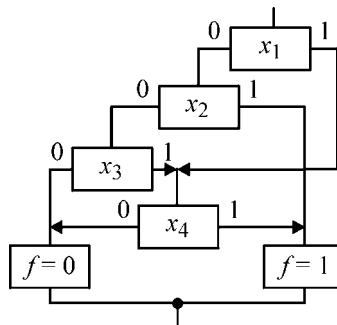


Рис. 16.56

Из рис. 16.56 следует, что заданная повторная БФ реализуется непланарным бесповторным бинарным графом. Линеаризуем этот граф [14] (рис. 16.57).

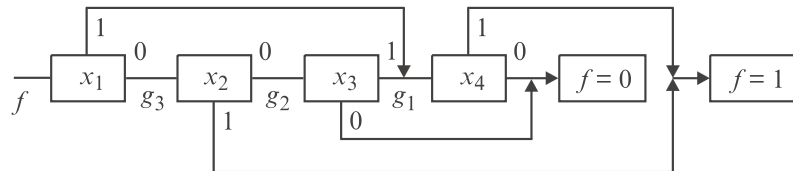


Рис. 16.57

Полученный граф является линейным и планарным, но после объединения выходов операторных вершин становится непланарным. В этом графе безынверсной переменной x_2 соответствует ребро, помеченное единицей и направленное вниз, а ОВ с пометкой $f = 0$ расположена перед ОВ с пометкой $f = 1$. Это отличает построенный граф от планарного повторного ЛБГ, построенного формульным методом (рис. 16.58).

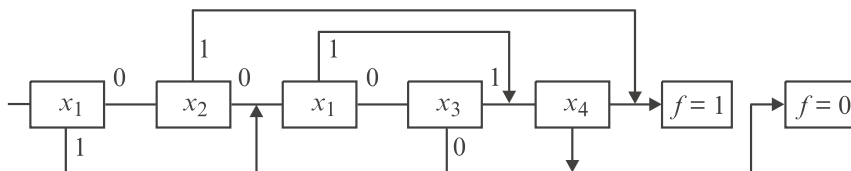


Рис. 16.58

Выполним верификацию построенного непланарного ЛБГ (рис. 16.57):

$$\begin{aligned} g_1 &= 0 \& !x_4 \vee 1 \& x_4 = x_4; \\ g_2 &= 0 \& !x_3 \vee g_1 x_4 = x_3 x_4; \\ g_3 &= g_2 !x_2 \vee 1 \& x_2 = x_2 \vee x_3 x_4; \\ f &= g_3 !x_1 \vee g_1 x_1 = !x_1 x_2 \vee (x_1 \vee x_3) x_4. \end{aligned}$$

Отметим, что формула g_3 не является ни подформулой, ни фрагментом заданной булевой формулы, что отличает рассмотренный метод от формульного.

Известно [15], что ББФ в базисе $\{\&, \vee, !\}$, существенно зависящие от всех своих переменных, содержат нечетное число единиц в столбцах значений таблиц истинности.

Из приведенного примера следует, что нечетность числа единиц является необходимым, но недостаточным условием реализуемости БФУ формулой, неповторной в базисе $\{\&, \vee, !\}$.

Автор предполагает также, что нечетность числа единиц является необходимым условием для реализации БФУ с помощью ЛБГ. Покажем, что это условие не является достаточным и в этом случае.

Пример 16.28. Реализовать булеву функцию

$$\begin{aligned} f(x_1, x_2, x_3) &= |0001 \ 0110|^T = \\ &= !x_1 x_2 x_3 \vee x_1 (x_2 \oplus x_3) = x_1 (x_2 \vee x_3) \oplus x_2 x_3, \end{aligned}$$

которая существенно зависит от всех переменных и содержит нечетное число единиц.

Построение бинарного графа выполнено на рис. 16.59.

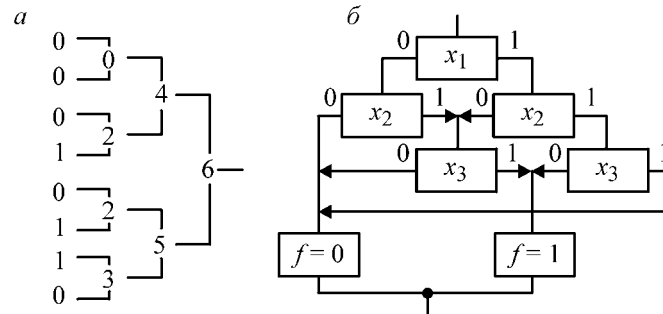


Рис. 16.59

Сформулируем необходимое и достаточное условие реализуемости булевой функции n переменных с помощью неповторного ЛБГ [3]. Это условие состоит из трех составляющих:

- БФУ существенно зависит от всех переменных;
- число единиц в столбце значений ТИ нечетно;
- максимальный номер в канонической структуре равен n , что обеспечивается тем, что каждый ее столбец содержит не более одного нового номера.

Это условие выполняется для булевой функции из примера 16.27 и не выполняется для функции из примера 16.28.

16.4.4. Оценки числа условных вершин в бинарных графах

Известно [12, 16, 17], что для булевой функции n переменных может быть построен бинарный граф, число условных вершин в котором при $n \rightarrow \infty$ определяется соотношением

$$B_y \sim 2^n / n$$

Однако метод, который строит такие бинарные графы, весьма сложен и неэффективен при малых значениях n .

Канонический, а как следует из изложенного выше, и распределительный методы обладают тем свойством, что они строят эффективные бинарные графы при малых значениях n , а при $n \rightarrow \infty$ для них справедливо соотношение

$$B_y \sim 2^{n+1} / n$$

Докажем эту оценку несколько иначе, чем это сделано в [12], с целью получения расчетных соотношений не только для случая $n \rightarrow \infty$ [6].

Будем рассматривать правильные канонические структуры [12], в которых все вершины одного яруса помечены одной переменной.

Наиболее сложно реализуемой БФУ является та, в которой каждый столбец КС имеет максимально возможное число новых номеров. Каноническая структура для БФУ n переменных состоит из n столбцов. Разобьем эту схему на две части. При этом первая будет содержать m столбцов, а вторая — $(n - m)$ столбцов, где $m = 0, 1, \dots, n$. Для столбца значений таблицы истинности $m = 0$.

В первой половине КС номера повторяются, а во второй — для самых сложных БФУ все номера различны. Максимальное число номеров в столбце с номером m в КС определяется соотношением

$$N_m^1 = 2^{2^m}.$$

Максимальное число новых номеров в этом столбце:

$$N_m = N_m^1 - N_{m-1}^1 = 2^{2^m} - 2^{2^{m-1}}.$$

Максимальное число различных номеров в первых m столбцах канонической структуры определяется соотношением

$$N_{r1} = \sum_{i=1}^m (2^{2^i} - 2^{2^{i-1}}) = 2^{2^m} - 2.$$

Рассмотрим столбец канонической структуры с номером $n - m$. Максимальное число различных номеров в нем:

$$N_{n-m} = 2^{n-m-1}.$$

Максимальное число различных номеров в столбцах с номерами от $n - m$ до n определяется соотношением

$$N_{r2} = 2^{n-m} - 1.$$

Максимальное число различных номеров в КС:

$$N = N_{r1} + N_{r2} = 2^{2^m} + 2^{n-m} - 3.$$

Максимальное число различных номеров в канонической структуре равно максимальному числу условных вершин в БГ:

$$B_{ym} = N.$$

Определим величину m , при которой значение B_{ym} минимально. Предположим, что $m_1 = \log n$, тогда

$$B_{y1} = 2^n + \frac{2^n}{n} - 3.$$

Эта величина превышает число вершин в полном дереве, для которого

$$B_y = 2^n - 1.$$

Уменьшим величину m . Пусть

$$m_2 = \lceil \log(n - \lfloor \log n \rfloor) \rceil.$$

При этом

$$B_{y2} \approx \frac{2^n}{n} + \frac{2^n}{n - \log n} - 3 = \frac{2^n}{n} \left(1 + \frac{n}{n - \log n} \right) - 3.$$

При $n \rightarrow \infty$

$$B_{y2} \sim \frac{2^{n+1}}{n}.$$

Отметим, что $B_{y1} > B_{y2}$. Для того чтобы m_2 обеспечивало минимум, уменьшим m по сравнению с m_2 . Пусть

$$m_3 = \log(n - 2 \log n).$$

При этом

$$B_{y3} = \frac{2^n}{n^2} + \frac{2^n}{n - 2 \log n} - 3.$$

Уже при малых значениях n справедливо, что $B_{y3} > B_{y2}$. Таким образом, минимум числа условных вершин для самых сложных БФУ достигается при

$$m = \lceil \log(n - \lfloor \log n \rfloor) \rceil; \quad 2^m + m \leq n \leq 2^{m+1} + m.$$

Построим на основе последнего соотношения табл. 16.1.

Т а б л и ц а 16.1

m	0	1	2	3	4	5	6	7
$2^m + m$	1	3	6	11	20	37	70	135
$2^{m+1} + m$	2	5	10	19	36	69	134	263

Таким образом, при $n = 1, 2$ $B_{ym} = 2^n - 1$; при $n = 3 \div 5$ $B_{ym} = 2^{n-1} + 1$; при $n = 6 \div 10$ $B_{ym} = 2^{n-2} + 13$; при $n = 11 \div 19$ $B_{ym} = 2^{n-3} + 253$; при $n = 20 \div 36$ $B_{ym} = 2^{n-4} + 65\,533$.

Используя эти соотношения, построим табл. 16.2.

Т а б л и ц а 16.2

n	1	2	3	4	5	6	7	8	9	10
B_{ym}	1	3	5	9	17	29	45	77	141	269

Эти значения получены также в [12]. В этой работе показано, что данные оценки достижимы и не могут быть уменьшены за счет перестановки входных переменных. Они достигаются для следующих типов функций, заданных конституентами единицы: при $n = 1$ $f = \{1\}$; при $n = 2$ $f = \{1, 2\}$; при $n = 3$ $f = \{1, 2, 4, 7\}$; при $n = 4$ $f = \{0, 1, 2, 9, 10, 11, 15\}$; $f = \{0, 1, 2, 5, 11, 12, 14, 15\}$.

Как отмечено в разд. 15.10, максимальное число путей в бинарных графах $S = 2^n$ и достигается для функций типа $f = x_1 \oplus x_2 \oplus \dots \oplus x_n$, а число условных вершин определяется соотношением $B_y = 2n - 1$.

При $n = 4$ для линейной функции $B_y = 7$, в то время как из табл. 16.2 следует, что при $n = 4$ $B_{ym} = 9$.

Таким образом, самые сложные по числу вершин бинарные графы не являются таковыми по числу путей, и наоборот.

Из изложенного следует, что

$$n \leq B_y(n) \leq B_{ym}; \quad n + 1 \leq S(n) \leq 2^n.$$

16.4.5. Сокращение перебора при поиске минимальной реализации заданной булевой функции

Построение для заданной БФУ бинарного графа с минимальным числом вершин и (или) минимальным числом путей связано с перебором по порядку расположения входных переменных в ТИ.

1. Предложим подход, который при построении БГ с минимальным числом путей позволяет либо устранить или уменьшить перебор, либо оценить, насколько найденное без перебора решение по числу путей отличается от нижней оценки их числа, которая, возможно, недостижима [6].

Для заданной БФУ в качестве нижней оценки числа путей предлагается выбрать число максимальных ортогональных областей, которые можно выделить в карте Карно, построенной для этой функции. Это объясняется тем, что, во-первых, это число не зависит от порядка входных переменных в таблице истинности, а, во-вторых, пути в бинарном графе соответствуют ортогональным областям в карте Карно, которые не обязательно являются максимальными. При этом число максимальных единичных (нулевых) ортогональных областей является нижней оценкой числа единичных (нулевых) путей в бинарном графе.

В качестве верхней оценки числа путей для заданной БФУ может быть выбрано минимальное число путей в бинарных графах, построенных при заданном порядке переменных (без перебора) каноническим и распределительным методами.

При этом если в построенном БГ число путей совпадает с нижней оценкой, то она является минимальной по числу путей.

Если число путей отличается от нижней оценки, то становится ясно, на какую величину может быть возможно сократить число путей в результате перебора. Если при переборе достигается нижняя оценка, то перебор может быть завершен.

Пример 16.29. Использовать предложенный подход для БФУ

$$f(x_1, x_2, x_3) = |0101 \ 0111|^T.$$

Так как в этом случае $n = 3$, то $4 \leq S(3) \leq 8$. Построим карту Карно для этой функции и выделим в ней максимальные ортогональные области (табл. 16.3).

Т а б л и ц а 16.3

x_i	$x_2 x_3$			
	00	01	11	10
0	0	1	1	0
1	0	1	1	1

Из рассмотрения этой карты следует, что она содержит четыре максимальные ортогональные области, и поэтому в данном случае приведенная выше нижняя оценка не может быть повышена.

Канонический метод строит в данном случае бинарный граф с пятью путями, которые соответствуют ортогональным областям, выделенным в табл. 16.4.

Т а б л и ц а 16.4

x_1	$x_2 x_3$			
	00	01	11	10
0	0	1	1	0
1	0	1	1	1

Распределительный метод в данном случае строит бинарный граф с четырьмя путями. Таким образом, минимальное решение найдено в данном случае без перебора.

Пример 16.30. Использовать предложенный подход для БФУ

$$f(x_1, x_2, x_3) = |0110 \ 1001|^T.$$

Так как в этом случае $n = 3$, то $4 \leq S(3) \leq 8$. Карта Карно для этой БФУ содержит восемь максимальных ортогональных областей, и поэтому $S = 8$. Бинарный граф с этим числом путей строит как канонический, так и распределительный методы.

Пример 16.31. Использовать предложенный подход для БФУ

$$f(x_1, x_2, x_3, x_4) = |0001 \ 1011 \ 1000 \ 0111|^T.$$

В этом случае $5 \leq S(4) \leq 16$. Карта Карно для этой булевой функции содержит 10 максимальных ортогональных областей. Поэтому $10 \leq S(f) \leq 16$.

Канонический и распределительный методы строят в данном случае БГ с 12 путями. Поэтому $10 \leq S(f) \leq 12$. Таким образом, если перебор проводить, то нельзя построить бинарный граф, который будет содержать на три пути меньше, чем бинарные графы, построенные без перебора.

2. Сокращение перебора при поиске БГ, с минимальным числом вершин может обеспечить построение по ТИ булевой формулы в базисе $\{\&, \vee, !\}$ с минимизированным числом букв.

Если эта БФ содержит h букв, то, как было показано в разд. 15.1, формульный метод строит ЛБГ с h условными вершинами. Если при этом $h < B_{ym}$, то это позволяет снизить верхнюю оценку сложности реализации заданной БФУ. Таким образом, $n \leq B_y(f) \leq h$. Если $h = n$, то формульный метод строит ЛБГ, минимальный по числу вершин, а точный метод, изложенный в разд. 15.11, позволяет построить без перебора ЛБГ, который минимален также и по числу путей.

16.4.6. Модификация канонического метода

Эта модификация состоит в том, что предлагается [7] после построения канонической структуры с целью сокращения числа номеров (разнотипных кустов) вынести на нее числа, уменьшая значения, указанные в столбце значений ТИ, и отмечая точками неизменяющиеся значения. Это, в свою очередь, позволяет сократить число вершин в граф-схеме (ГС) по сравнению с их числом в бинарном графе, построенном каноническим методом. При этом ГС в от-

личие БГ содержит операторные вершины, помеченные арифметическими операциями.

Если в канонической структуре вынесенной оказывается константа C и если она размещается в корне дерева или является нулем, то в ГС ей соответствует ОВ с пометкой $f = C$. Если вынесенная константа размещается не в корне КС и не является нулем, то в ГС ей соответствует ОВ с пометкой $f = f + C$. При этом отметим, что константа C может быть и отрицательным числом.

Двигаясь по построенной ГС сверху вниз, пометки некоторых ОВ могут быть упрощены. Метод строит граф-схемы, в которых на каждом пути от входа к выходу каждая входная переменная, помечающая условные вершины, встречается не более одного раза, а операторные вершины (в том числе и однотипные) могут встречаться на одном пути многократно.

Пример 16.32. Построить СГС для булевой функции

$$f(x_1, x_2, x_3) = |0000 \ 0111|^T = x_1(x_2 \vee x_3).$$

Используя формульный метод, для заданной БФ может быть построен ЛБГ с пятью вершинами. Применяя канонический метод, можно построить СБГ с семью вершинами. На рис. 16.60 показано построение с помощью предлагаемой модификации канонического метода линейаризованной структурированной граф-схемы (ЛСГС) с шестью вершинами.

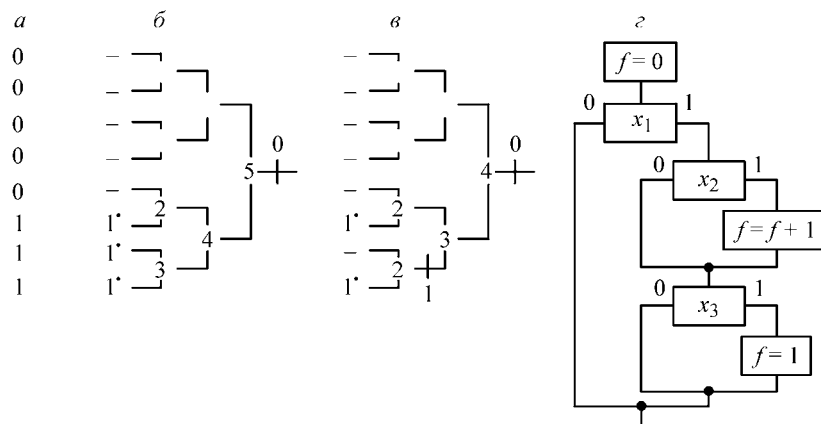


Рис. 16.60

В построенной граф-схеме пометка операторной вершины $f = f + 1$ может быть заменена на пометку $f = 1$.

Построенная ЛСГС может быть изоморфно запрограммирована с применением одноадресных условных переходов. Однако, так как

ЛСГС имеет длинный переход, она не может быть изоморфно запрограммирована на языке инструкций некоторых программируемых логических контроллеров, например [18], который позволяет изоморфно реализовывать только ЛСГС с короткими переходами.

Пример 16.33. Построить ЛСГС с короткими переходами для БФУ, рассмотренной в предыдущем примере.

На рис. 16.61 показано построение с помощью модификации канонического метода ЛСГС с короткими переходами, также содержащей шесть вершин.

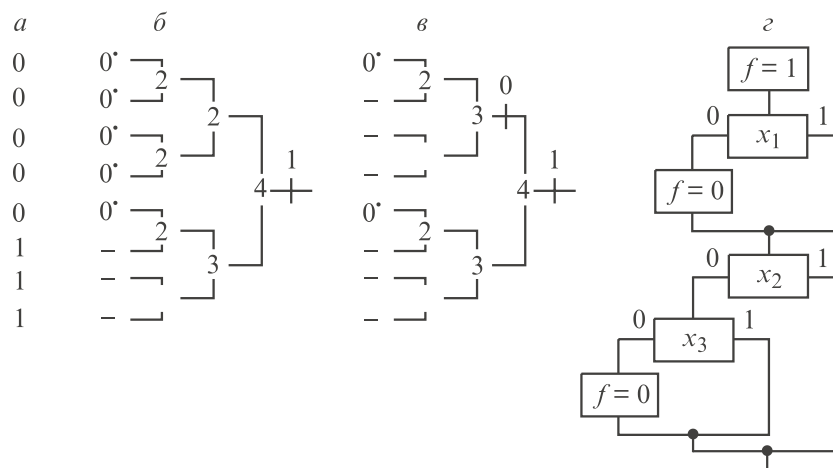


Рис. 16.61

Реализуем построенную ЛСГС программой на языке инструкций [18], записанной в два столбца:

```

EQ SO f    IF NI x2
IF NI x1   IF NI x3
EQ RO f    EQ RO f
CONT       CONT
           END

```

Из приведенного примера следует, что число команд в этой программе равно суммарному числу компонент (УВ, ОВ и точки объединения дуг) в ЛСГС с короткими переходами.

Пример 16.34. Построить ЛСГС с короткими переходами для БФУ, рассмотренной в примере 16.32, в которой изменен порядок входных переменных в таблице истинности:

$$f(x_3, x_2, x_1) = |0001 \ 0101|^T = (x_3 \vee x_2)x_1.$$

На рис. 16.62 показано построение с помощью модификации канонического метода ЛСГС с короткими переходами, которая преобразована в ЛСБГ (рис. 16.63).

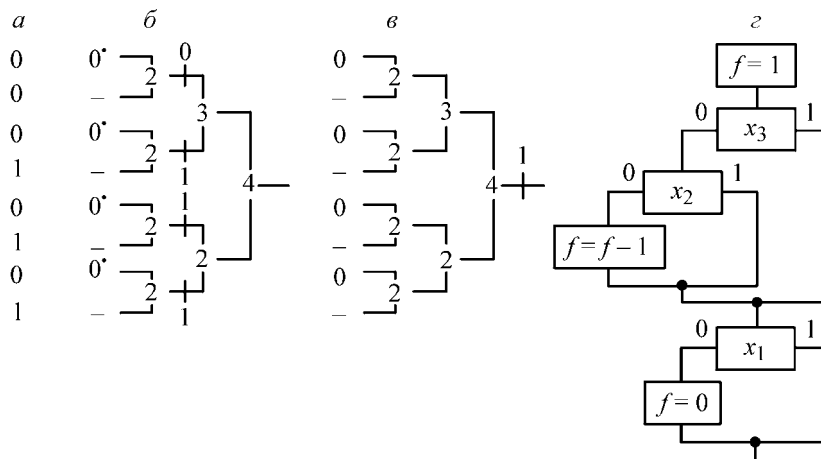


Рис. 16.62

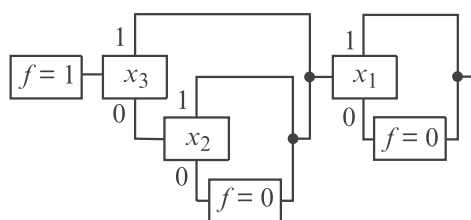


Рис. 16.63

16.5. Методы построения бинарных графов для систем булевых функций

Ветвящиеся граф-схемы разделим на два класса:

- без арифметических операций;
- с арифметическими операциями.

Ветвящиеся ГС могут быть классифицированы и по другому признаку — по расположению операторных вершин в них. Если эти вершины располагаются только после условных вершин, то такие ГС называются простыми, в противном случае — обобщенными [16].

В настоящем разделе рассматриваются ГС без арифметических операций, а в разд. 16.6 — с таковыми.

16.5.1. Метод построения простых бинарных графов

Изложим еще одну модификацию канонического метода, предназначенную для построения бинарных графов для систем БФУ (СБФУ). Эта модификация состоит в следующем:

- по таблице истинности СБФУ строится таблица истинности одной арифметической функции (АФ), заменяя каждый набор значений N заданных функций их десятичным эквивалентом;
- арифметическая функция реализуется каноническим методом;
- в построенной ГС каждая ОВ заменяется N операторными вершинами с двоичными операторами присваивания;
- в построенном БГ выполняется минимизация числа ОВ.

Пример 16.35. Реализовать параллельный одноразрядный двоичный сумматор, заданный табл. 16.7.

Т а б л и ц а 16.7

x_1	x_2	x_3	f_1	f_2	F
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	2
1	0	0	0	1	1
1	0	1	1	0	2
1	1	0	1	0	2
1	1	1	1	1	3

Реализуем АФ каноническим методом (рис. 16.64).

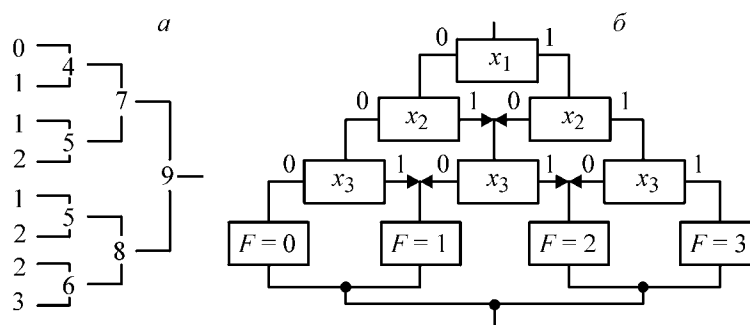


Рис. 16.64

Заменяя в этой ГС операторные вершины, получим искомый бинарный граф с 14 вершинами (рис. 16.65). Сокращение числа вершин до 12 достигается за счет объединения и изменения порядка расположения одинаковых операторных вершин (рис. 16.66). Изложенный метод впервые описан [19].

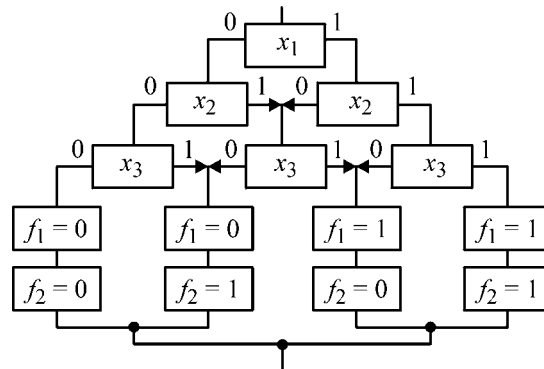


Рис. 16.65

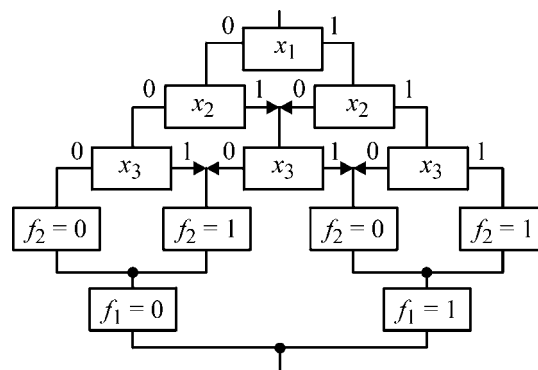


Рис. 16.66

16.5.2. Метод последовательного соединения простых бинарных графов

Этот метод состоит в построении простых бинарных графов для каждой БФУ системы и в последовательном их соединении. Получающийся в результате БГ является обобщенным.

Пример 16.36. Реализовать параллельный одноразрядный двоичный сумматор предлагаемым методом.

Обобщенный бинарный граф, построенный этим методом и содержащий 13 вершин, приведен на рис. 16.67.

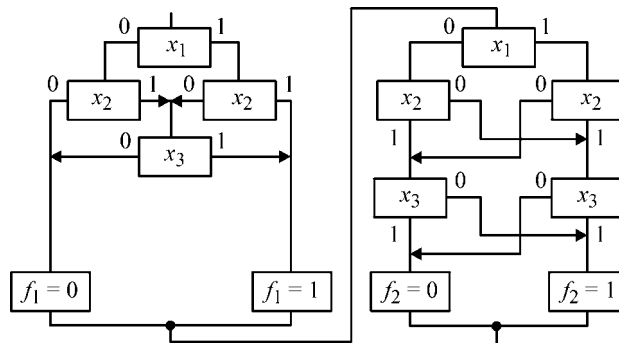


Рис. 16.67

Этот метод впервые описан в [20].

16.5.3. Метод построения обобщенных бинарных графов

Метод состоит в следующем. Построим каноническую структуру и вынесем на нее для отдельных функций рядом расположенные одинаковые значения, входящие в кусты любого уровня, с целью повышения однотипности кустов. По такой структуре изоморфно строится бинарный граф, который после этого минимизируется за счет объединения соответствующих ОВ нижнего уровня. Дальнейшее сокращение числа вершин может обеспечиваться за счет подъема вверх некоторых операторных вершин.

Пример 16.37. Реализовать параллельный одноразрядный двоичный сумматор предлагаемым методом.

Каноническая структура с вынесенными на нее операторными вершинами приведена на рис. 16.68.

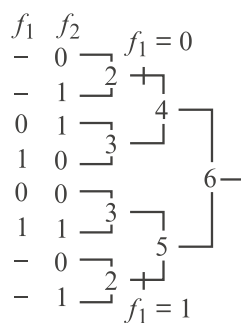


Рис. 16.68

Обобщенный БГ, построенный по этой структуре и содержащий 11 вершин, приведен на рис. 16.69. Дальнейшее сокращение числа вершин в БГ достигается за счет подъема вверх ОВ с пометкой $f_1 = 0$ (рис. 16.70). Этот граф содержит 10 вершин.

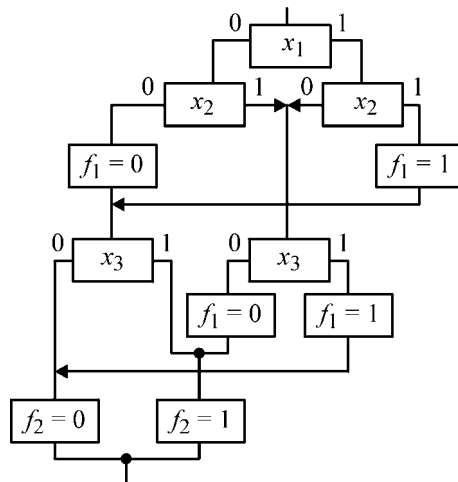


Рис. 16.69

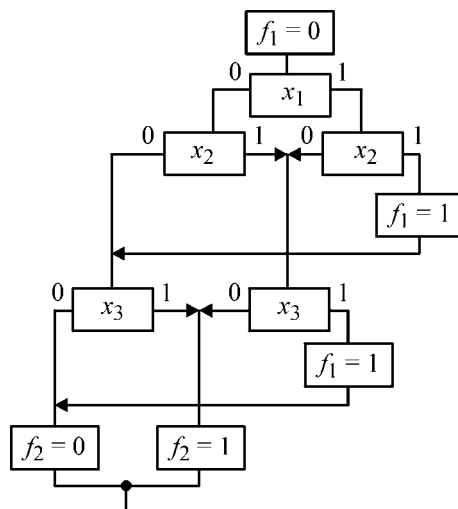


Рис. 16.70

Изложенный метод впервые описан в [21].

16.5.4. Метод независимых фрагментов

Этот метод применительно к реализации одной булевой формулы изложен в разд. 16.3. Расширим этот метод применительно к системам булевых формул (СБФ) [10, 11]:

— каждая из N булевых формул системы записывается в дизъюнктивной нормальной форме (ДНФ);

— каждая конъюнкция, входящая в i -ю БФ, помечается символом выходной переменной f_i , где $i = 1, \dots, N$. Если конъюнкция входит в m формул, то она помечается m указанными символами;

— все символы $!f_i$ и все помеченные конъюнкции объединяются в оператор структуры (ОС) с помощью символа $|$;

— ОС минимизируется по числу букв за счет выноса влево за скобки произвольной глубины одинаковых входных переменных x_j , входящих в разные конъюнкции, если такая возможность имеется;

— по минимизированному ОС (МОС) изоморфно строится ЛСБГ или СБГ, используя правила, изложенные в разд. 16.3.

Так же как и для одной БФ, в данном случае допускается, чтобы для некоторых БФ системы, например f_i и f_k , в ОС наряду со всеми конъюнкциями ДНФ остальных формул системы объединялись также все конъюнкции ДНФ инверсий формул f_i и f_k , помеченных символами $!f_i$ и $!f_k$ соответственно. В качестве символов начальной установки для этих БФ применяются символы f_i и f_k .

Если ОС содержит h_1 букв, то по нему может быть построен ЛСБГ с короткими переходами, содержащий h_1 вершин.

Если МОС содержит h_1 букв, то по нему может быть построен ЛСБГ с длинными переходами, содержащий h_1 вершин.

Если оператор содержит квадратные скобки и h_1 букв, то по нему может быть построен СБГ, содержащий менее чем h_1 вершин.

Пример 16.38. Реализовать СБФ $f_1 = x_1 \vee x_2 \vee x_3$, $f_2 = (!x_1!x_2 \vee \vee x_3)x_4$.

Построим оператор структуры:

$$F = !f_1!f_2|x_1f_1|x_2f_1|x_3f_1|!x_1!x_2x_4f_2|x_3x_4f_2.$$

Этот оператор состоит из 15 букв и может быть реализован ЛСБГ с короткими переходами, содержащим 15 вершин. В данном случае существуют два МОС, содержащих по 14 букв:

$$F = !f_1!f_2|x_1f_1|x_2f_1|x_3(f_1|x_4f_2)|!x_1!x_2x_4f_2;$$

$$F = !f_1!f_2|x_1f_1|x_2f_1|x_3f_1|x_4(!x_1!x_2x_4f_2|x_3f_2).$$

Каждый из этих операторов реализуется ЛСБГ с длинными переходами, содержащим 14 вершин. Если в первый МОС ввести квадратные скобки:

$$F = !f_1!f_2|[!x_1!x_2x_4f_2 \vee x_1f_1]|x_2f_1|x_3(f_1|x_4f_2),$$

то по нему может быть построен СБГ с 13 вершинами (рис. 16.71).

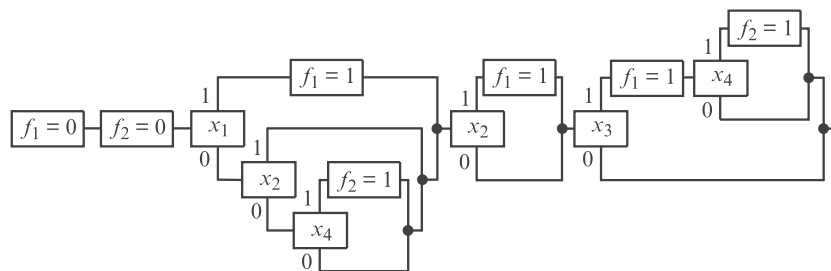


Рис. 16.71

Если первую формулу заданной системы заменить на инверсию, то

$$F = f_1 | !f_2 | !x_1 !x_2 !x_3 !f_1 | !x_1 !x_2 x_4 f_2 | x_3 x_4 f_2.$$

Этот оператор структуры, содержащий 13 букв, реализуется ЛСБГ с короткими переходами, состоящим из 13 вершин (рис. 16.72).

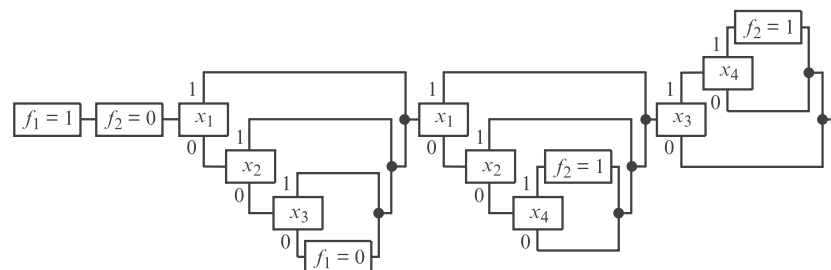


Рис. 16.72

Минимизируя полученный оператор по числу букв, получим МОС, содержащий 11 букв:

$$F = f_1 | !f_2 | !x_1 !x_2 (!x_3 !f_1 | x_4 f_2) | x_3 x_4 f_2.$$

Этот оператор реализуется ЛСБГ с длинными переходами, содержащим 11 вершин (рис. 16.73).

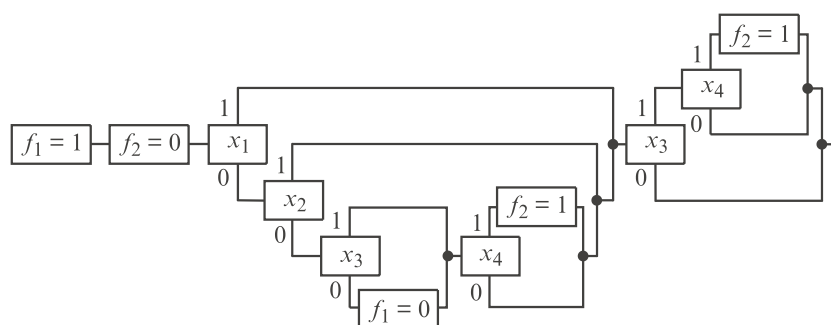


Рис. 16.73

16.5.5. Метод зависимых фрагментов

Для каждой булевой формулы, входящей в систему, с помощью метода зависимых фрагментов, изложенного в разд. 16.2, строится ЛСБГ. Для каждого ЛСБГ строится изоморфный оператор структуры (ОС) по правилам, используемым в методе независимых фрагментов (разд. 16.3).

Объединение построенных операторов осуществляется с учетом следующих особенностей:

- блоки с одним входом и одним выходом, соответствующие одной выходной переменной, не влияют на вычисление значений других выходных переменных;
- в объединенном ОС должен быть сохранен порядок относительного расположения фрагментов в каждом операторе;
- объединенный ОС может быть минимизирован за счет вынесения одинаковых (если они имеются) входных переменных влево за скобки.

Пример 16.39. Определить число вершин в ЛСБГ, реализующем СБФ:

$$f_1 = (x_1 \vee x_2 \vee x_3)x_4x_5; \quad f_2 = x_4x_5x_6 \vee x_7x_8.$$

Используя метод зависимых фрагментов, для каждой заданной БФ построим ЛСБГ и запишем по нему оператор структуры:

$$F_1 = !f_1|x_4x_5f_1|!x_1!x_2!x_3!f_1; \quad F_2 = !f_2|x_4x_5x_6f_2|x_7x_8f_2.$$

Построим объединенный ОС и проминимизируем его:

$$\begin{aligned} F &= !f_1|x_4x_5f_1|!x_1!x_2!x_3!f_1|!f_2|x_4x_5x_6f_2|x_7x_8f_2 = \\ &= !f_1|!f_2|x_4x_5(f_1|x_6f_2)|!x_1!x_2!x_3!f_1|x_7x_8f_2. \end{aligned}$$

Этот оператор содержит 14 букв и изоморфно реализуется ЛСБГ, состоящим из 14 вершин.

Отметим, что метод независимых фрагментов строит в данном случае оператор структуры

$$F = !f_1|!f_2|x_4x_5(x_1f_1|x_2f_1|x_3f_1|x_6f_2)|x_7x_8f_2$$

из 15 букв, который реализуется ЛСБГ, содержащим 15 вершин.

В заключение разд. 16.5 покажем, что существуют системы, состоящие только из двух БФУ, зависящих от одних и тех же переменных, совместная реализация которых с помощью модификации канонического метода вне зависимости от порядка переменных в ТИ приводит к построению бинарных графов с экспоненциальной (относительно числа переменных) оценкой числа вершин в них.

При этом отметим, что при применении этого метода сложность бинарного графа, реализующего две БФУ, не может быть меньше

сложности бинарного графа с наибольшим числом вершин, реализующего одну из этих функций, так как в противном случае эта функция могла бы быть реализована за счет исключения из бинарного графа, построенного для системы, операторных вершин, соответствующих второй функции.

Системы указанного типа формируются, если при фиксированном n в качестве первой выбрать функцию из первого класса функций, приведенных в конце разд. 15, а в качестве второй — функцию из второго класса приведенных там же функций [35]. Для этой системы функций изменение порядка переменных в их таблице истинности в общем случае не позволяет упростить совместную реализацию.

При этом отметим, что метод независимых фрагментов строит для таких систем структурированные бинарные графы с линейной оценкой сложности.

В общем случае вопрос о выборе метода эффективной реализации систем БФУ остается открытым.

16.6. Методы построения арифметических граф-схем для систем булевых функций

16.6.1. Реализация одной булевой функции

Известно [22, 23], что произвольная булева функция n переменных может быть реализована арифметическим полиномом (АП), содержащим не более 2^n ненулевых коэффициентов, который, в свою очередь, всегда может быть реализован линеаризованной СГС (ЛСГС) с короткими переходами, в которой в операторных вершинах выполняются арифметические операции вида: $f = f + a$; $f = f - a$, где a — ненулевой коэффициент АП.

Пример 16.40. Реализовать булеву формулу $f = x_1(x_2 \vee x_3)$.

Используя результаты [23], построим АП для заданной булевой формулы:

$$\begin{aligned} f &= x_1(x_2 \vee \neg x_2 x_3) = x_1(x_2 + (1 - x_2)x_3) = \\ &= x_1 x_2 + x_1 x_3 - x_1 x_2 x_3. \end{aligned}$$

Этот АП реализуется ЛСГС, представленной на рис. 16.74.

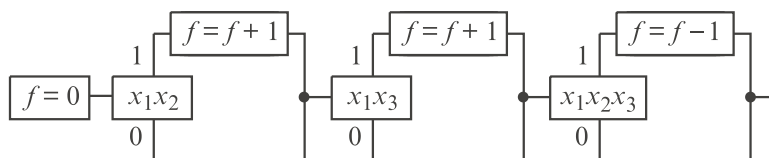


Рис. 16.74

Если АП проминимизировать, то по полученному арифметическому выражению

$$f = x_1(x_2 + x_3 - x_2x_3)$$

может быть построена упрощенная ЛСГС (рис. 16.75).

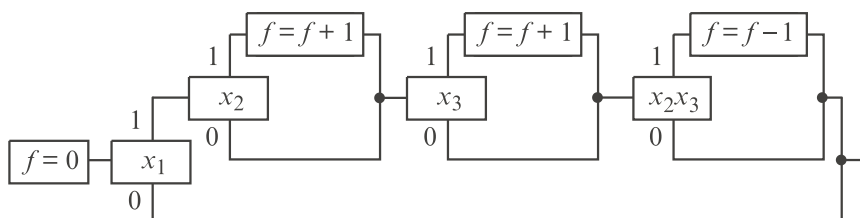


Рис. 16.75

Рассмотренный метод является своеобразным методом независимых фрагментов, так как в построенной с его помощью ЛСГС последовательно соединенные блоки, содержащие условные вершины, могут переставляться.

16.6.2. Метод вынесения операторных вершин

Заменяем заданную систему булевых функций n переменных одной арифметической функцией $F(x_1, \dots, x_n)$.

Применим к этой функции модификацию канонического метода, рассмотренного в разд. 16.4.6 [7]. Для получения по вычисленному значению кортежа битов в конце ГС должна размещаться операторная вершина, выполняющая операцию «десятичное число — двоичное число», которая обозначается символом «bin F». В построенных ниже граф-схемах эта вершина умалчивается.

Пример 16.41. Реализовать арифметическую функцию (АФ)

$$F(x_1, x_2, x_3) = |2 \ 3 \ 2 \ 4 \ 1 \ 3 \ 2 \ 3|^T.$$

Вынесем на каноническую структуру (КС) числа, на которые могут быть уменьшены пары значений АФ, объединенные кустами нижнего уровня, а затем вынесем число 2 в корень структуры, исключив и заменив соответствующие пометки в первоначально построенной канонической структуре (рис. 16.76).

Построим по полученной канонической структуре изоморфную ей граф-схему (рис. 16.77) и упростим в последней пометки операторных вершин, просматривая ее сверху вниз (рис. 16.78).

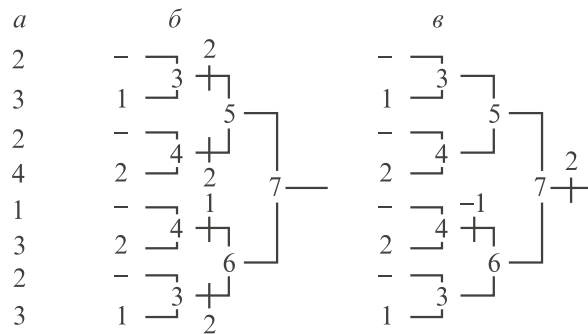


Рис. 16.76

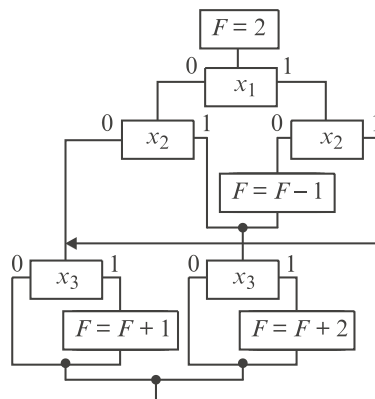


Рис. 16.77

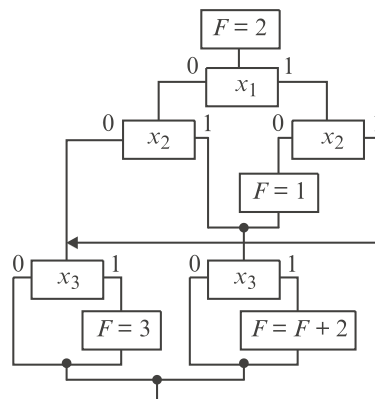


Рис. 16.78

Пример 16.42. Реализовать арифметическую функцию

$$F(x_1, x_2, x_3) = |0 \ 2 \ 0 \ 2 \ 2 \ 4 \ 1 \ 3|^T.$$

Уменьшим все значения АФ (кроме нулевых) на число два, вынесем его в корень канонической структуры, синтезированной по полученной функции, и построим по ней СГС (рис. 16.79).

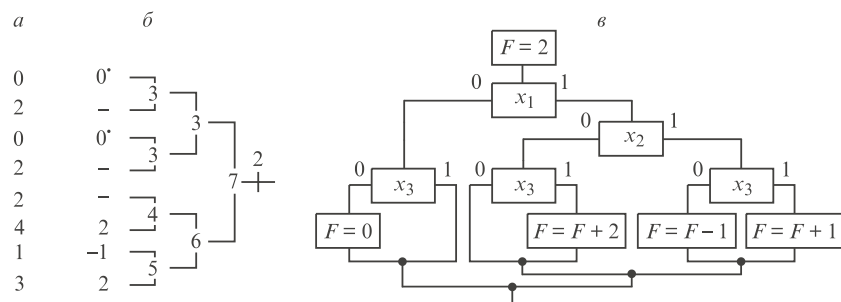


Рис. 16.79

В этой ГС могут быть выполнены следующие замены пометок
ОВ: $F + 2$ на $F = 4$, $F - 1$ на $F = 1$, $F + 1$ на $F = 3$.

Покажем, что при другом выносе чисел на КС изоморфная ей ГС по сравнению с предыдущей упрощается (рис. 16.80).

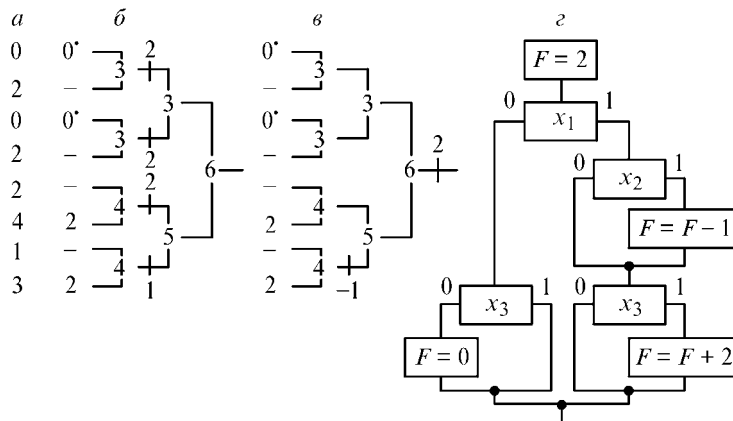


Рис. 16.80

В этой ГС может быть выполнена следующая замена пометки операторной вершины: $F - 1$ на $F = 1$.

Пример 16.43. Реализовать параллельный двоичный одноразрядный сумматор

$$F(x_1, x_2, x_3) = |0 \ 1 \ 1 \ 2 \ 1 \ 2 \ 2 \ 3|^T.$$

На рис. 16.81 показан процесс оптимизации КС, а на рис. 16.82 приведена ЛСГС, соответствующая оптимизированной структуре.

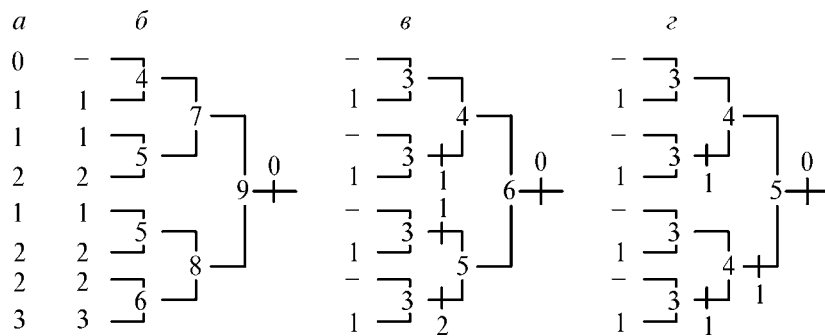


Рис. 16.81

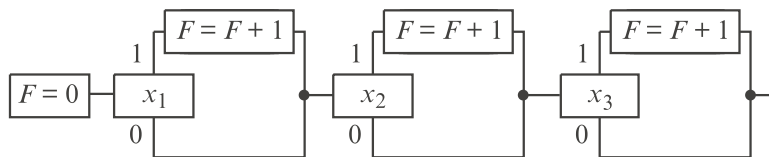


Рис. 16.82

В построенной ГС в крайней левой операторной вершине пометка $F = F + 1$ может быть заменена на пометку $F = 1$.

Сформулируем утверждение: если ЛСГС состоит только из последовательного соединения операторной вершины с пометкой « $F = C_0$ » и n блоков «неполный выбор», операторные вершины в которых имеют пометки « $F = F + C_1$ », ..., « $F = F + C_n$ » соответственно, то СБФ, соответствующая этой граф-схеме, может быть реализована линейным арифметическим полиномом (ЛАП) вида: $F = C_0 + C_1 x_1 + \dots + C_n x_n$.

Для рассмотренного примера $F = x_1 + x_2 + x_3$.

Пример 16.44. Реализовать арифметическую функцию

$$F(x_1, x_2, x_3) = |0 \ 2 \ 4 \ 6 \ 3 \ 5 \ 7 \ 9|^T.$$

Процесс построения канонической структуры изображен на рис. 16.83, а на рис. 16.84 приведена ЛСГС, построенная по этой структуре.

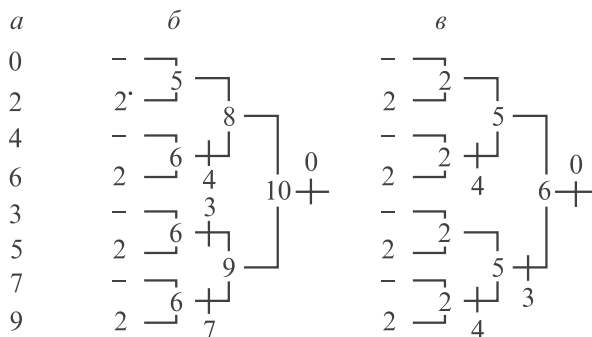


Рис. 16.83

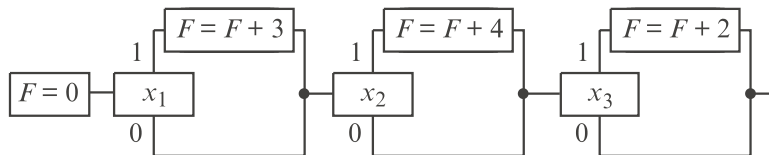


Рис. 16.84

В построенной ГС пометка операторной вершины « $F = F + 3$ » может быть заменена на пометку « $F = 3$ ». Из изложенного следует, что $F = 3x_1 + 4x_2 + 2x_3$.

Таким образом, можно утверждать, что изложена новая трактовка условия линейности арифметического полинома, предложенного в [23].

Если построенная СГС имеет структуру, отличную от описанной выше, то система БФУ реализуется нелинейным АП.

Пример 16.45. Реализовать арифметическую функцию

$$F(x_1, x_2, x_3) = |2 \ 2 \ 2 \ 2 \ 1 \ 2 \ 2 \ 3|^T.$$

Процесс построения канонической структуры изображен на рис. 16.85, а на рис. 16.86 приведена ЛСГС, построенная по этой КС.

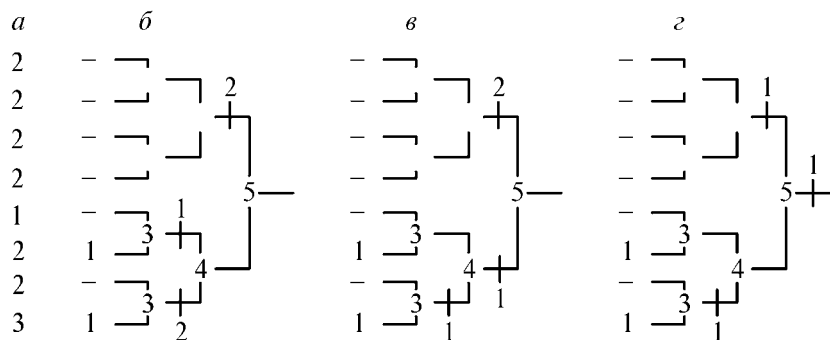


Рис. 16.85

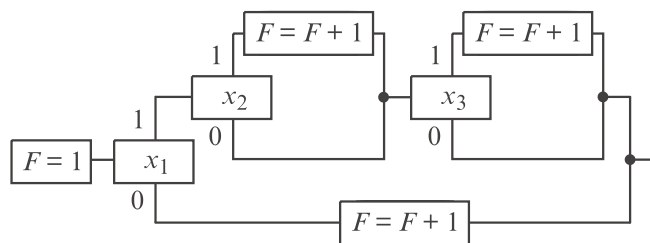


Рис. 16.86

В построенной ГС в двух левых операторных вершинах пометка « $F = F + 1$ » может быть заменена на пометку « $F = 1$ ». Построим арифметическое выражение по этой СГС:

$$F = 1 + 1(1 - x_1) + x_1(1x_2 + 1x_3) = 2 + x_1(1x_2 + 1x_3 - 1).$$

Построим СГС по этому выражению (рис. 16.87).

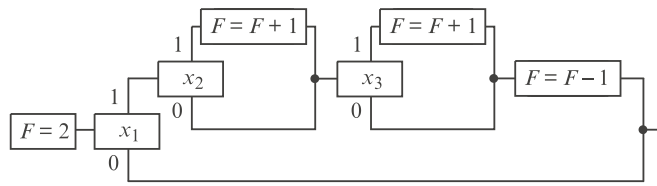


Рис. 16.87

В построенной ГС в левой операторной вершине пометка « $F = F + 1$ » может быть заменена на пометку « $F = 1$ ». Преобразуем полученное выражение в АП:

$$F = 2 - x_1 + x_1 x_2 + x_1 x_3.$$

Этому полиному соответствует ЛСГС, представленная на рис. 16.88.

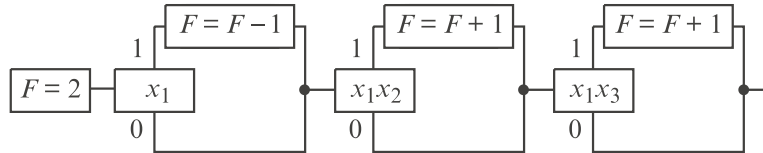


Рис. 16.88

В построенной ГС пометка операторной вершины « $F = F - 1$ » может быть заменена на пометку « $F = 1$ ».

В заключение раздела отметим, что предложенный подход [7] может рассматриваться в качестве графического метода построения арифметических полиномов, что отличает его от подходов, описанных в гл. 20 и [22, 23].

16.6.3. Метод взвешивания функций

Каждой из N функций системы может быть присвоен вес от 2^{N-1} до 2^0 .

После этого система БФУ может быть реализована ЛСГС с короткими переходами, представленной на рис. 16.89.

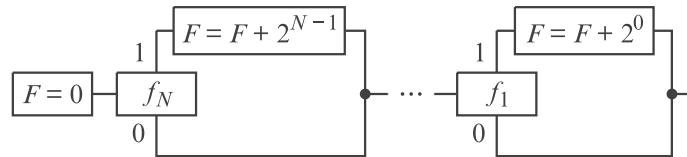


Рис. 16.89

Для одноразрядного сумматора $N=2$, $f_2 = x_1 \# x_2 \# x_3$, $f_1 = x_1 \oplus x_2 \oplus x_3$.

16.6.4. Стандартная реализация строк таблицы истинности

Предположим, что F_i — значение арифметической функции F на входном наборе i . При этом, например, для $n = 2$ функция F может быть реализована стандартными схемами на рис. 16.90—16.92.

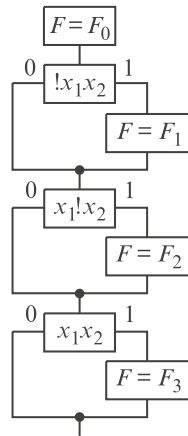


Рис. 16.90

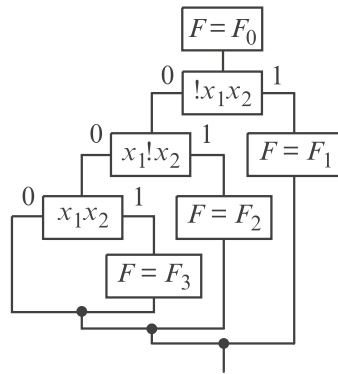


Рис. 16.91

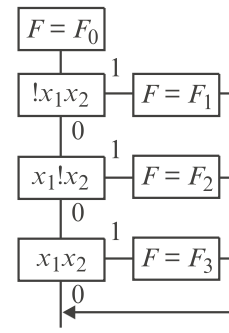


Рис. 16.92

Можно показать, что стандартной также является СГС на рис. 16.93, которая проще СГС, построенной по арифметическому полиному (рис. 16.94).

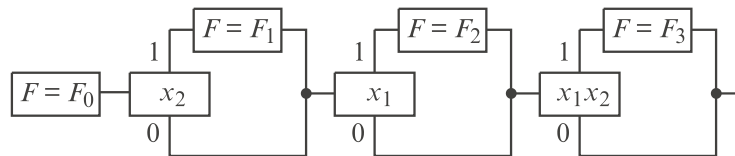


Рис. 16.93

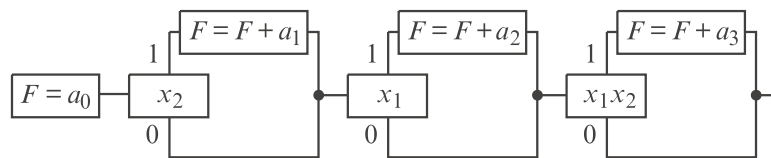


Рис. 16.94

16.6.5. Реализация типов значений арифметических функций

Предлагаемый метод строит ЛСГС (с короткими переходами), которые состоят из операторных вершин с пометкой « $F = F_j$ » и $R - 1$ блоков «неполный выбор», в каждом из которых в УВ записана булева формула g_i , осуществляющая выбор значения F_i , а ОВ в этом блоке имеет пометку « $F = F_i$ ». Здесь R — число типов значений АФ. Так как булевы формулы в ОВ ортогональны, то блоки в построенной таким образом ГС могут переставляться.

Пример 16.46. Реализовать параллельный двоичный одноразрядный сумматор $F(x_1, x_2, x_3) = |0 \ 1 \ 1 \ 2 \ 1 \ 2 \ 2 \ 3|^T$.

Формула, выбирающая значение $F = 1$, имеет вид:

$$g_1(x_1, x_2, x_3) = |0110 \ 1000|^T = !x_1(!x_2 \vee !x_3) \oplus !x_2!x_3.$$

Формула, выбирающая значение $F = 2$, имеет вид:

$$g_2(x_1, x_2, x_3) = |0001 \ 0110|^T = x_1(x_2 \vee x_3) \oplus x_2x_3.$$

Формула, выбирающая значение $F = 3$, имеет вид:

$$g_3(x_1, x_2, x_3) = |0000 \ 0001|^T = x_1x_2x_3.$$

На рис. 16.95 приведена ЛСГС, построенная предложенным методом.

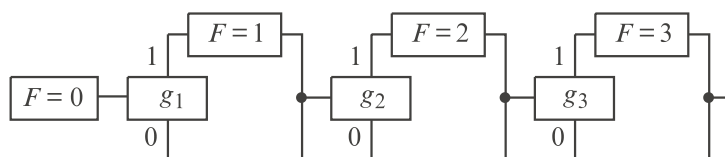


Рис. 16.95

Приведенная реализация неэффективна, так как суммарная сложность булевых формул g_1, g_2, g_3 выше суммарной сложности булевых формул f_2, f_1 , непосредственно описывающих сумматор.

Приведем пример реализации СБФ, для которой этот метод весьма эффективен.

Пример 16.47. Реализовать арифметическую функцию

$$F(x_1, x_2, x_3) = |0 \ 7 \ 12 \ 0 \ 21 \ 0 \ 0 \ 0|^T.$$

В этом случае для $F = 7$ $g_1 = !x_1!x_2x_3$, для $F = 12$ $g_2 = !x_1x_2!x_3$, для $F = 21$ $g_3 = x_1!x_2!x_3$, что приводит к эффективной реализации (рис. 16.96), так как арифметическая функция в данном случае задает весьма сложную СБФ:

$$f_5 = x_1!x_2!x_3, f_4 = !x_1x_2!x_3, f_3 = !x_1(x_2 \oplus x_3) \vee x_1!x_2!x_3, \\ f_2 = !x_1!x_2x_3, f_1 = (x_1 \oplus x_3)!x_2.$$

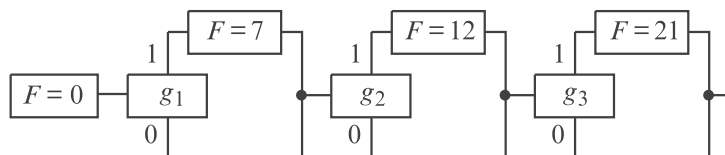


Рис. 16.96

16.6.6. Упрощение булевых формул, осуществляющих выбор типов значений арифметических функций

Ниже излагается метод построения СГС с $R - 1$ условными и R операторными вершинами, в которых сложность булевых формул, помечающих условные вершины, не превышает сложности соответствующих булевых формул, которые строятся методом, изложенным в предыдущем разделе.

Метод основан на том, что после формирования каждой булевой формулы, осуществляющей выбор однотипных значений арифметической функции, эти значения из таблицы истинности исключаются, повышая тем самым степень недоопределенности заданной функции. Это позволяет при формировании следующих булевых формул так доопределять столбец значений таблицы истинности, чтобы булевы формулы, осуществляющие выбор однотипных значений, получались максимально простыми. При использовании этого метода каждый тип значений арифметической функции вычисляется однократно.

Этот метод позволяет строить СГС как от входа к выходу, так и в обратном порядке. Получающиеся при этом СГС имеют разнотипные структуры.

Пример 16.48. Реализовать параллельный двоичный одноразрядный сумматор, строя СГС от входа к выходу.

Выберем значение $F = 3$ с помощью БФ $g_1 = x_1x_2x_3$. При этом $F = |0 \ 1 \ 1 \ 2 \ 1 \ 2 \ 2 \ -|^T$. Доопределим эту функцию следующим образом: $F = |0 \ 1 \ 1 \ 2 \ 1 \ 2 \ 2 \ 2|^T$ и выберем значение $F = 2$ с помо-

щью БФ $g_2 = x_1 \# x_2 \# x_3$. После этого АФ приобретает вид $F = |0 \ 1 \ 1 \ - \ 1 \ - \ - \ -|^T$. Доопределим АФ следующим образом: $F = |0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1|^T$. Выбирая $F = 1$ с помощью формулы $g_3 = x_1 \vee x_2 \vee x_3$, завершаем построение СГС (рис. 16.97).

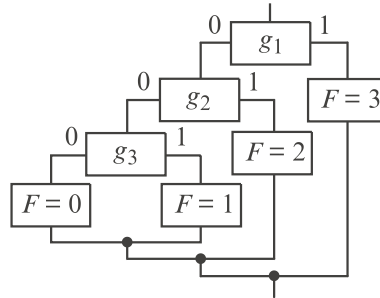


Рис. 16.97

Таким образом, две булевы формулы — f_2 и f_1 , описывающие одноразрядный сумматор, реализуются с применением трех булевых формул, причем $f_2 = g_2$.

Пример 16.49. Реализовать параллельный двоичный одноразрядный сумматор, строя СГС от выхода к входу.

Выберем значение $F = 3$ с помощью БФ $g_{31} = x_1 x_2 x_3$. При этом $F = |0 \ 1 \ 1 \ 2 \ 1 \ 2 \ 2 \ 2|^T$. Доопределим эту функцию следующим образом: $F = |0 \ 1 \ 1 \ 2 \ 1 \ 2 \ 2 \ 2|^T$. Выберем значение $F = 1$ с помощью формулы $g_{21} = x_1 \oplus x_2 \oplus x_3$. После этого АФ приобретает вид $F = |0 \ - \ - \ 2 \ - \ 2 \ 2 \ -|^T$. Доопределим АФ следующим образом: $F = |0 \ 2 \ 2 \ 2 \ 0 \ 2 \ 2 \ 2|^T$. Выбирая $F = 2$ с помощью $g_{11} = x_2 \vee x_3$, завершаем построение СГС (рис. 16.98).

Таким образом, две булевы формулы — f_2 и f_1 , описывающие одноразрядный сумматор, реализуются с использованием трех булевых формул, причем $f_1 = g_{21}$.

Изложенный метод впервые описан в [7].

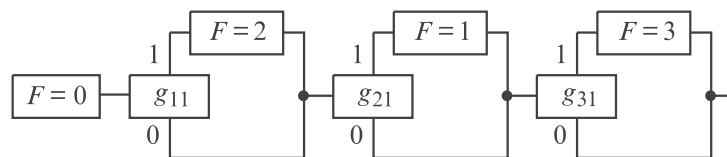


Рис. 16.98

ба», так как в нем сначала расположены только нули, а затем только единицы. Например, БПФ

$$f = x_1 \vee x_2 \left(x_3 \vee \left(x_4 \left(x_5 \vee x_6 (x_7 \vee x_8) \right) \right) \right)$$

на первых 85 входных наборах (с десятичными эквивалентами 0, ..., 84) принимает нулевые значения, а на остальных 171 входных наборах (с десятичными эквивалентами 85, ..., 255) — единичные.

Это позволяет при порядке переменных $x_1, \dots, x_8$ реализовать эту БПФ векторной ГС, содержащей только одну условную вершину:

$$\text{if } (X \geq 85) \text{ f} = 1 \text{ else f} = 0,$$

где X — десятичный эквивалент значений входного набора (вектора). При этом $X = 128x_1 + 64x_2 + 32x_3 + 16x_4 + 8x_5 + 4x_6 + 2x_7 + 1x_8$.

Такой ГС соответствует пороговая реализация вида:

$$f = \text{sign} (128x_1 + 64x_2 + 32x_3 + 16x_4 + 8x_5 + 4x_6 + 2x_7 + 1x_8 - 85).$$

Используя метод, изложенный в разд. 15.5, можно построить минимальную пороговую реализацию:

$$f = \text{sign} (21x_1 + 13x_2 + 8x_3 + 5x_4 + 3x_5 + 2x_6 + x_7 + x_8 - 21).$$

Несмотря на то что эта реализация имеет существенно меньшие значения весов и порога, она не может быть столь эффективно программно реализована, как это предложено выше.

Если порядок переменных во входном наборе не совпадает с порядком переменных, при котором БПФ имеет одну точку перегиба, то сложность ВГС увеличивается.

Пример 16.51. Реализовать БПФ $f = x_1 x_2 \vee x_3$.

При порядке переменных x_1, x_2, x_3 эта БПФ реализуется по диапазонам значений наборов входных переменных, так как

$$f(x_1, x_2, x_3) = |0101 \ 0111|^T.$$

При этом строится ВГС, представленная на рис. 16.100.

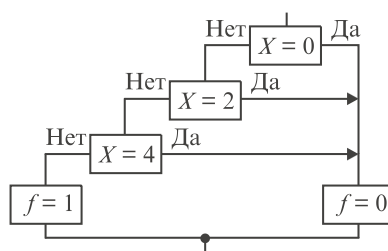


Рис. 16.100

При порядке переменных x_3, x_2, x_1 эта функция реализуется существенно проще:

$$\text{if } (X \geq 3) \text{ f} = 1 \text{ else f} = 0,$$

так как

$$f(x_3, x_2, x_1) = |0001 \ 1111|^T.$$

По диапазонам входных переменных может быть реализована и произвольная булева функция n переменных.

Пример 16.52. Реализовать булеву функцию

$$f(x_1, x_2, x_3, x_4) = |0001 \ 0001 \ 0001 \ 1111|^T = \\ = x_1 x_2 \vee x_3 x_4.$$

При порядке переменных x_1, x_2, x_3, x_4 эта БФУ реализуется ВГС, приведенной на рис. 16.101.

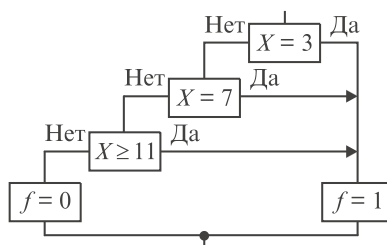


Рис. 16.101

Изложенный подход может быть использован и для реализации арифметических функций. Так, например, на рис. 16.102 приведена ВГС, реализующая параллельный двоичный одноразрядный сумматор

$$F(x_1, x_2, x_3) = |0 \ 1 \ 1 \ 2 \ 1 \ 2 \ 2 \ 3|^T$$

при порядке входных переменных x_1, x_2, x_3 .

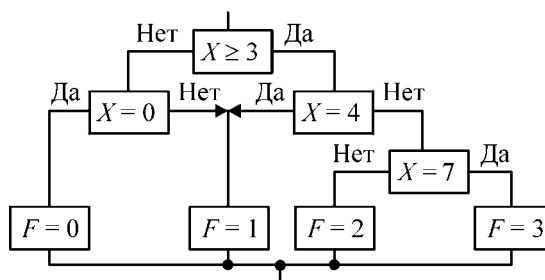


Рис. 16.102

Кроме работ, на которые в настоящей и предыдущей главах имеются ссылки, существует также и много других работ, посвященных бинарным графам, например [24—48].

16.8. Построение комбинационных схем из демультиплексоров

В последнее время бинарные графы все чаще применяются как при программной, так и при аппаратной реализации (в том числе и при проектировании сверхбольших интегральных схем на основе программируемой логики) в качестве средства представления отдельных БФУ (БФ) и систем БФУ (БФ) [30—43].

В разд. 5.1 изложены методы реализации бинарных графов (БГ), построенных для отдельных БФУ (БФ), изоморфными схемами из мультиплексоров «2 в 1». Эти методы основаны на рассмотрении каждого бинарного графа от его выходов к входу и поэтому не позволяют реализовать системы БФУ (БФ) совместно.

Ниже предлагается метод совместной схемной реализации указанных систем, основанный на рассмотрении бинарного графа, построенного для заданной системы с помощью методов, изложенных в разд. 16.5, от входа к выходам.

Метод базируется на применении демультиплексоров (DX) «1 в 2» (рис. 16.103), каждый из которых имеет два входа — b и x_i и два выхода — u_1 и u_0 , причем $u_1 = bx_i$, $u_0 = b!x_i$ [32, 49].

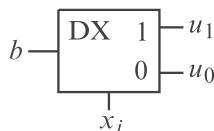


Рис. 16.103

Рассмотрим первоначально метод реализации N заданных функций системы и их инверсий по бинарному графу, который не содержит операторных вершин начальной установки. Этот метод состоит в следующем:

- каждая условная вершина бинарного графа заменяется демультиплексором «1 в 2» таким образом, что ее входу соответствует вход демультиплексора, а ее нулевому (единичному) выходу — нулевой (единичный) выход демультиплексора;
- каждая точка объединения дуг заменяется двухвходовым элементом ИЛИ;
- на b -вход первого демультиплексора подается константа «1», а на нижний вход i -го демультиплексора — входная переменная, указанная в i -й условной вершине графа;

— все одноименные операторные вершины $f_j = 1$ заменяются линиями, которые подключаются к элементу ИЛИ, на выходе которого реализуется соответствующая функция f_j ;

— все одноименные операторные вершины $f_j = 0$ заменяются линиями, которые подключаются к элементу ИЛИ, на выходе которого реализуется соответствующая функция $!f_j$.

В результате строится схема с $2N + 1$ выходами, на N из которых реализуются заданные функции, на N других выходах — их инверсии, а на последнем выходе — функция $f_k = 1$.

Число элементов в построенной таким образом схеме, изоморфной бинарному графу с точностью до реализации операторных вершин в нем, определяется соотношением:

$$n_3 = B_y + n_T + 2N,$$

где B_y — число условных вершин, n_T — число точек объединения дуг.

При реализации одной функции и ее инверсии справедливо соотношение

$$n_3 = B_y + n_T.$$

Построенная указанным образом схема может быть упрощена за счет исключения элемента ИЛИ, реализующего функцию f_k .

На рис. 16.104 приведена схема из шести элементов ($B_y = 3$, $n_T = 3$), реализующая формулы $f = x_1 x_2 \vee x_3$ и $!f = (!x_1 \vee !x_2)!x_3$, которая изоморфна ЛБГ, представленному на рис. 15.1.

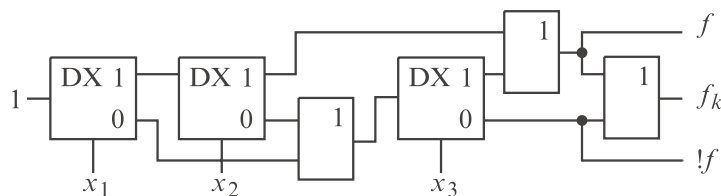


Рис. 16.104

После исключения последнего элемента ИЛИ полученная схема будет состоять из пяти элементов.

Так как в ЛБГ с одним входом и одним выходом при реализации формулы в базисе $\{\&, \vee, !\}$ из h букв $B_y = h$, $n_T = h$, то число элементов в рассматриваемой схеме, одновременно реализующей формулу и ее инверсию, удовлетворяет соотношению $n_3 = 2h$. При этом число элементов в схеме не зависит от порядка записи формулы.

На рис. 16.105 приведена схема, также содержащая шесть элементов, реализующая формулу $f = x_3 \vee x_2 x_1$ и ее инверсию.

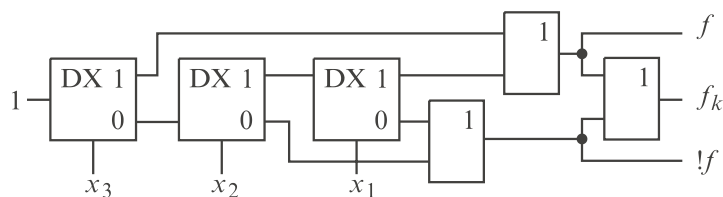


Рис. 16.105

После исключения последнего элемента ИЛИ полученная схема будет состоять из пяти элементов.

Приведенные схемы построены по ЛБГ. На рис. 16.106 приведены две схемы, построенные по бинарному графу (рис. 16.67), первая из которых реализует мажоритарную функцию трех переменных и ее инверсию, а вторая — линейную функцию трех переменных и ее инверсию.

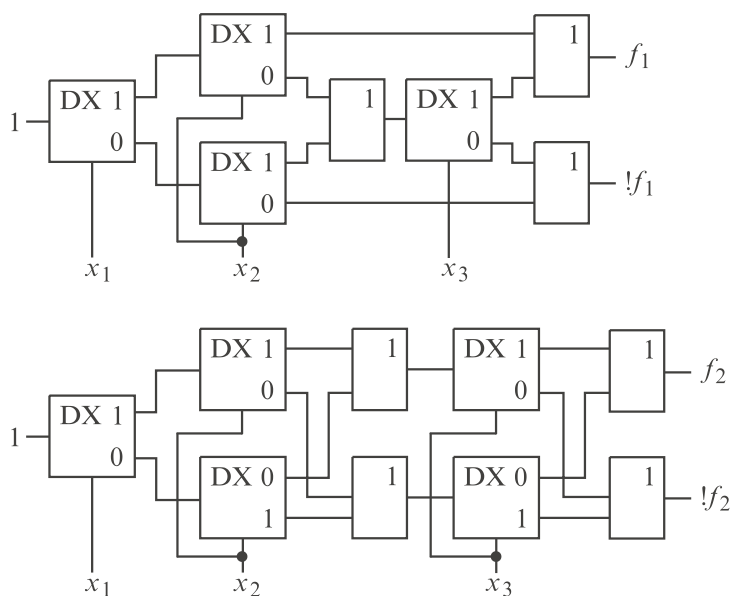


Рис. 16.106

Эти схемы реализованы независимо, несмотря на то что в БГ соответствующие им блоки соединены последовательно. Это объясняется тем, что если в БГ блоки с одним входом и одним выходом соединены последовательно и не «охватываются» другими блоками, то в схеме, изоморфной этому БГ, на выходе f_k каждой подсхемы, соответствующей блоку, формируется тождественная единица, что позволяет исключить элементы ИЛИ, связывающие эти блоки, подав единицы на входы каждой из образовавшихся при этом схем.

Эффективность предлагаемого метода повышается в том случае, когда схема, реализующая систему функций и их инверсии, строится по совместно минимизированному бинарному графу. На рис. 16.107 приведена схема из 11 элементов, построенная по бинарному графу (рис. 16.69), в котором исключен элемент ИЛИ, реализующий функцию f_k .

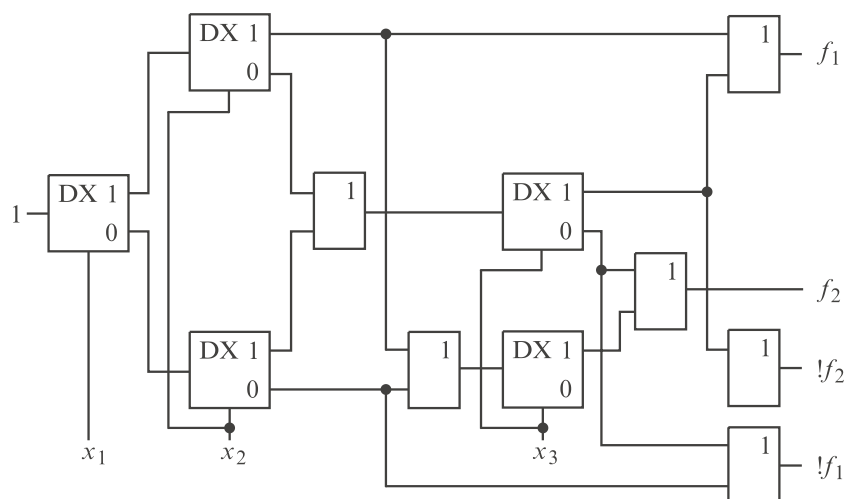


Рис. 16.107

Для реализации функций системы (без их инверсий) построенная схема должна быть упрощена за счет исключения и упрощения элементов, связанных с реализацией инверсных функций и функции f_k .

Для непосредственной реализации функций системы изложенный метод должен быть модифицирован. Эта модификация применима к бинарным графам, которые либо не содержат операторных вершин начальной установки, либо эти вершины должны быть типа $f_j = 0$.

Предлагаемая модификация связана с изменением пятого этапа изложенного выше метода, который в данном случае формулируется следующим образом: все операторные вершины типа $f_j = 0$ заменяются линиями.

Число элементов в схеме в этом случае определяется соотношением:

$$n_{э1} \leq B_y + n_{т1} + N,$$

где $n_{т1}$ — число точек объединения дуг, связанных с вычислением функций f .

Построенная указанным образом схема может быть упрощена за счет исключения элементов ИЛИ, реализующих функцию f_k , а также каждого демультиплексора совместно с элементом ИЛИ, подсоединенным к его выходам. В последнем случае функция, формируемая на выходе элемента ИЛИ, переносится в точку подключения входа исключаемого демультиплексора. Дальнейшее упрощение схемы достигается за счет замены некоторых из оставшихся демультиплексоров двухвходовыми элементами И, ЗАПРЕТ, переменными и (или) их инверсиями.

Так как в линейном бинарном графе $B_y = h$, $0 \leq n_{\tau 1} \leq h - 1$, то справедливо соотношение

$$h \leq n_{\tau 1} \leq 2h - 1.$$

При этом нижняя оценка достигается для функции И, а верхняя — для функции ИЛИ.

Сложность схемы, построенной по ЛБГ, в этом случае зависит от порядка ее записи, так как при этом может измениться значение переменной $n_{\tau 1}$.

На рис. 16.108 приведена схема, реализующая формулу $f = x_1 x_2 \vee x_3$, а на рис. 16.109 — формулу $f = x_3 \vee x_2 x_1$.

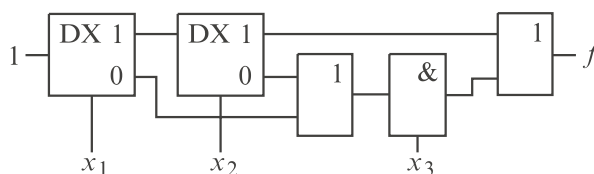


Рис. 16.108

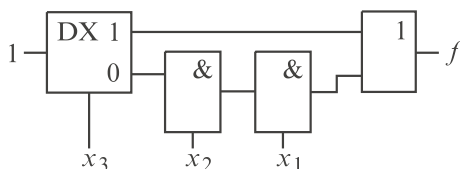


Рис. 16.109

Из приведенных примеров следует, что при реализации булевых формул, неповторных в базисе $\{\&, \vee, !\}$, предлагаемый метод менее эффективен по сравнению с формульным методом построения схем из двухвходовых элементов И и ИЛИ, для которого $n_{\tau 1}^{\phi} = h - 1$.

Предлагаемый метод при реализации повторных формул в указанном базисе обычно более эффективен, чем указанный формульный метод.

Так, например, схема, реализующая предлагаемым методом линейную функцию n переменных, состоит из $2(n - 1)$ демультиплек-

соров, $2n - 3$ двухвходовых элементов ИЛИ и одного двухвходового элемента И ($n_{\exists 1} = 4(n - 1)$). На рис. 16.110 приведена схема восьми двухвходовых элементов, реализующая линейную функцию трех переменных, которая может быть также построена в результате упрощения второй схемы, представленной на рис. 16.106.

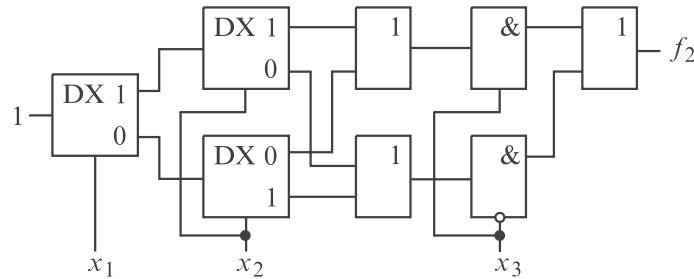


Рис. 16.110

При этом отметим, что для функций этого класса при их записи в базисе $\{\&, \vee, !\}$ справедливо соотношение

$$h = 3 \cdot 2^{n-1} - 2.$$

Эта схема может быть упрощена, если три последних элемента в ней заменить мультиплексором «2 в 1» (рис. 16.111).

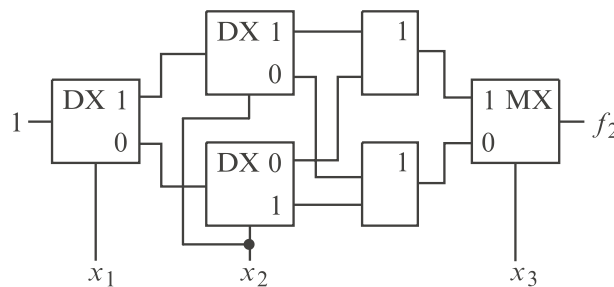


Рис. 16.111

Отметим, что линейные функции при их записи с использованием операций «сумма по модулю два» могут быть реализованы более эффективно схемами из мультиплексоров «2 в 1» с помощью формульного метода, изложенного в разд. 5.1.6. При этом $n_{\exists 1}^{\Phi} = n - 1$.

Изложенный метод позволяет реализовать мажоритарную функцию трех переменных, являющуюся симметрической, схемой из шести двухвходовых элементов (рис. 16.112), которая может быть также получена в результате упрощения первой схемы (рис. 16.106).

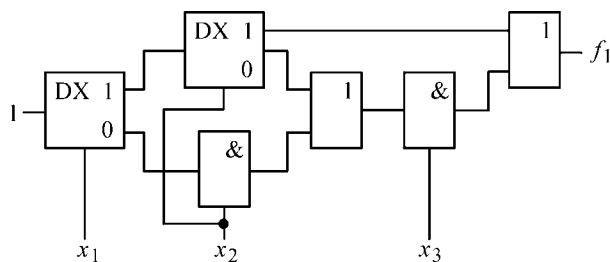


Рис. 16.112

Симметрические функции n переменных могут быть также реализованы и на основе схемы, изоморфной симметрической структуре из переключаемых контактов [50]. Эта схема состоит из $(n+1)n/2$ демультиплексов и $n(n-1)/2$ двухвходовых элементов ИЛИ ($n_3 = n^2$). На рис. 16.113 приведена схема этого класса при $n = 3$.

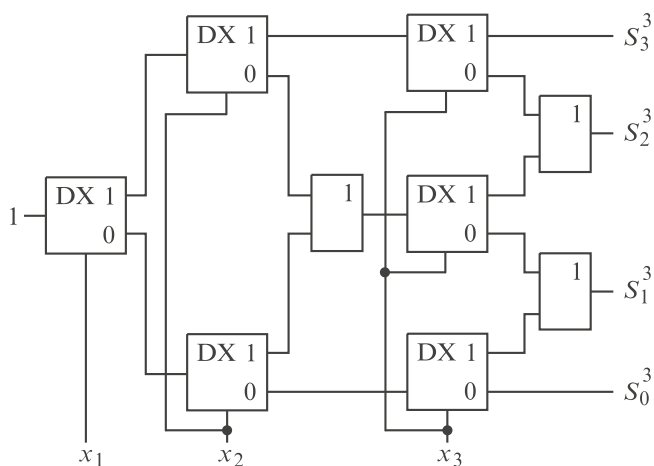


Рис. 16.113

Применение этой структуры в качестве универсальной для реализации одиночных симметрических функций неэффективно, так как для этих целей существует схема, построенная из $(n+1)n/2$ мультиплексов «2 в 1». Такая схема для $n = 3$ приведена на рис. 9.9.

Однако на основе предложенной схемы в отличие от других методов можно достаточно просто (по трудоемкости) реализовать системы симметрических функций n переменных. На рис. 16.114 приведена схема, реализующая параллельный одноразрядный двоичный сумматор и состоящая из 11 двухвходовых логических эле-

ментов. При этом отметим, что раздельная реализация функций сумматора, выполненная с помощью предлагаемого метода, строит схему из 12 логических элементов (рис. 16.111, 16.112).

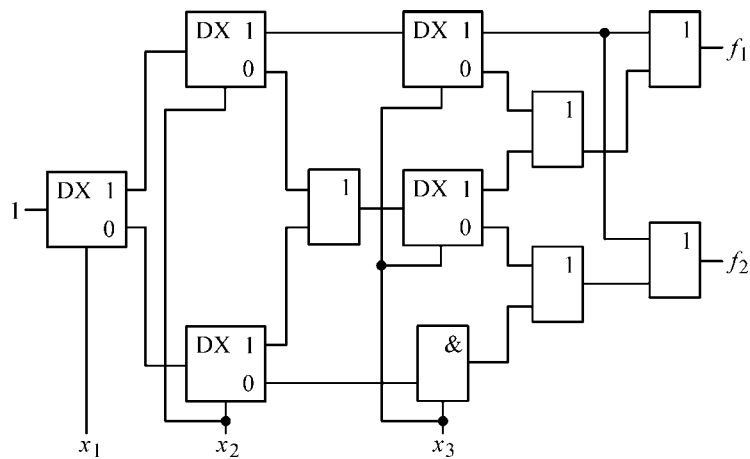


Рис. 16.114

Эта схема может быть также построена и с помощью предлагаемого метода по простому БГ (рис. 16.66).

Если этот метод применить к обобщенному БГ (рис. 16.70), то рассматриваемый сумматор реализуется схемой, содержащей девять двухвходовых элементов (рис. 16.115).

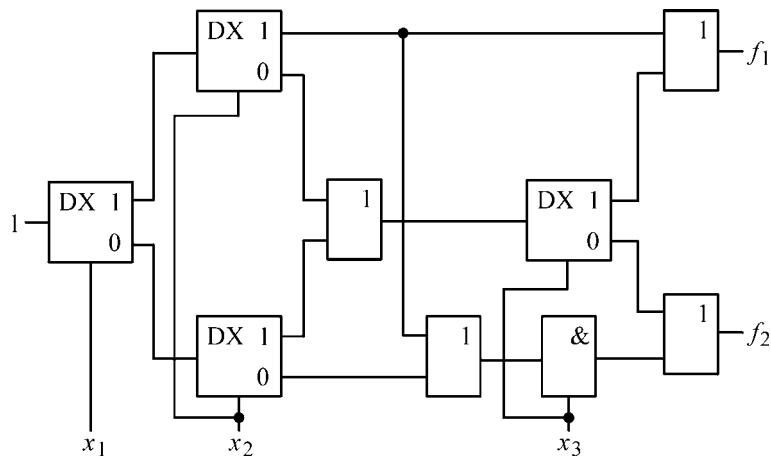


Рис. 16.115

Несмотря на то что известна схема рассматриваемого сумматора, состоящая всего лишь из пяти двухвходовых элементов

(рис. 10.19), это не уменьшает значения предлагаемого универсального метода.

Рассмотрим систему формул, отличных от рассмотренных выше, которая приведена в примере 16.38. После преобразования эта система приобретает вид:

$$f_1 = x_1 \vee x_2 \vee x_3, \quad f_2 = (! (x_1 \vee x_2) \vee x_3) x_4.$$

Выполним факторизацию этих формул:

$$\Phi = x_1 \vee x_2, \quad f_1 = \Phi \vee x_3, \quad f_2 = (!\Phi \vee x_3) x_4.$$

Полученная система реализуется схемой, использующей демультиплексор и содержащей пять элементов (рис. 16.116).

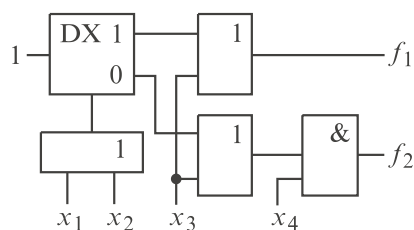


Рис. 16.116

Предлагаемый метод строит по структурированному БГ (рис. 16.71) изоморфную ему схему менее трудоемко (рис. 16.117).

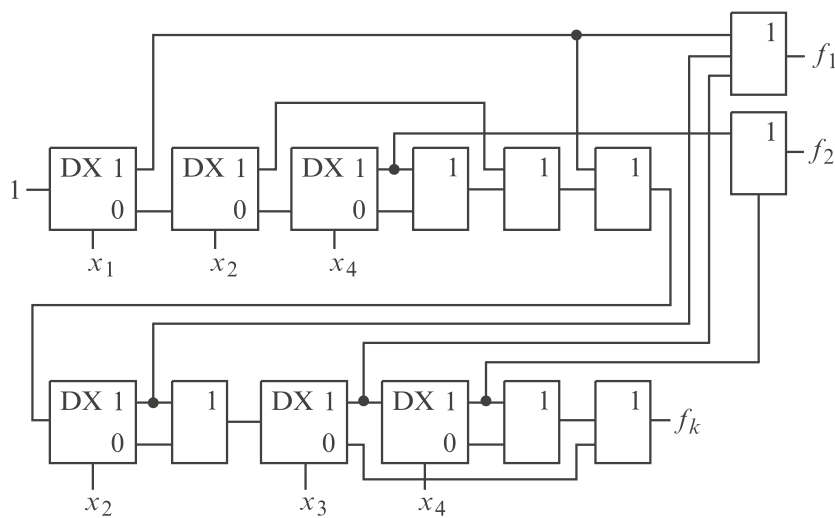


Рис. 16.117

Эта схема состоит из 14 элементов, так как в этом случае $B_y = 6$, $n_{\tau 1} = 6$, $N = 2$. На рис. 16.118 приведена схема из шести элементов, построенная в результате упрощения предыдущей схемы.

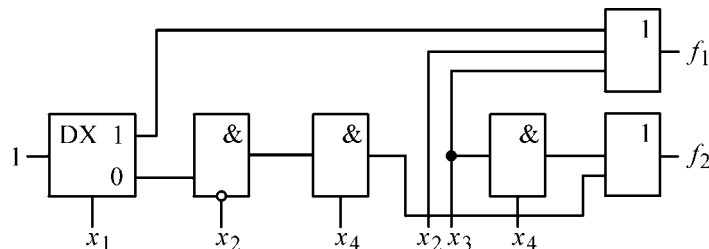


Рис. 16.118

В заключение раздела отметим, что предложенный метод дополняет изложенные в разд. 1.6 методы совместной реализации булевых функций.

Выводы

1. Рассмотрены три метода структурирования линейных бинарных графов и получены оценки числа вершин в структурированных бинарных графах (СБГ). Эти оценки являются в том числе и линейными от числа букв в реализуемой булевой формуле в базисе $\{\&, \vee, !\}$.

2. Предложен метод зависимых фрагментов, позволяющий по булевой формуле в указанном базисе строить линейаризованные СБГ (ЛСБГ). Показано, что этот метод реализует неповторные пороговые формулы (БПФ) с помощью ЛСБГ с числом вершин $2h + 1$.

3. Предложен метод зависимых фрагментов с оптимизацией, который обеспечивает реализацию неповторных пороговых формул с помощью ЛСБГ с числом вершин

$$h + 2 \leq B \leq 2h.$$

4. Показано, как с помощью метода зависимых фрагментов с оптимизацией реализовать булевы формулы, имеющие пороговый образ.

5. Усовершенствован метод независимых фрагментов, позволяющий по булевой формуле и системе булевых формул строить оператор структуры, число букв в котором равно числу вершин в ЛСБГ. Изложен метод минимизации числа букв в этом операторе. Показано, что в отдельных случаях этот метод позволяет строить СБГ, число вершин в которых меньше числа букв в указанном операторе.

6. Предложен распределительный метод, позволяющий реализовывать булевы функции, заданные таблицами истинности, при порядке переменных, противоположном таковому в каноническом методе Блоха. Это позволяет по одной таблице истинности (без перестановки столбцов входных переменных в ней) строить два бинарных графа и выбирать простейший из них.

7. Показано, что существуют повторные булевы формулы, реализуемые непланарными неповторными линейными бинарными графами. Определено необходимое и достаточное условие, обеспечивающее получение такой реализации.

8. Показано, что самые сложные по числу путей бинарные графы не являются самыми сложными по числу вершин, и наоборот.

9. Показано, что число максимальных ортогональных областей в карте Карно является нижней оценкой сложности числа путей в бинарном графе.

10. Предложена модификация канонического метода Блоха для построения простых бинарных графов, реализующих системы булевых функций.

11. Предложена модификация канонического метода Блоха для построения обобщенных бинарных графов, реализующих системы булевых функций, основанная на вынесении значений функций на остои канонической структуры.

12. Предложены методы построения для систем булевых функций, арифметических граф-схем, в операторных вершинах которых применяются операторы вида $F = F * C$, где $*$ = $\{+, -\}$; C — константа.

13. Для булевых функций, и систем булевых функций, предложены методы построения векторных бинарных графов, в условных вершинах которых выполняется сравнение значений входного вектора с соответствующими масками.

14. Предложен метод построения по бинарным графам изоморфных им схем из демультиплексоров «1 в 2» и двухвходовых элементов ИЛИ, совместно реализующих булевы функции.

Л и т е р а т у р а

1. Иодан Э. Структурное проектирование и конструирование программ. М.: Мир, 1979.
2. Лингер Р., Миллс Х., Уитт Б. Теория и практика структурного программирования. М.: Мир, 1982.
3. Артюхов В. Л., Кузнецов Б. П., Шалыто А. А. Настраиваемые логические устройства для судовых управляющих систем. Л.: ИПК СП, 1986.
4. Кузнецов Б. П., Шалыто А. А. Структурный подход к программной реализации булевых формул // Автоматика и вычисл. техника. 1985. № 5.
5. Aschcroft E., Manna Z. The translation of «goto» programs into «while» programs // Proceedings of 1971 IFIP Congress.
6. Шалыто А. А. Реализация алгоритмов судовых управляющих логических систем при использовании микропроцессорной техники. Л.: ИПК СП, 1988.

7. Шалыто А. А. Switch-технология. Алгоритмизация и программирование задач логического управления. СПб: Наука, 1998.
8. Артюхов В. Л., Шалыто А. А. Реализация булевых формул однородными мультиплексорными и мажоритарными каскадами // Изв. РАН. Теория и системы управления. 1996. № 5.
9. Артюхов В. Л., Кузнецова О. С., Шалыто А. А. Оценка функциональных возможностей программируемых логических матриц // Автоматика и вычисл. техника 1985. № 2.
10. Кузнецов Б. П. Структурирование бинарных программ // Вопросы судостроения. Сер. Судовая автоматика. 1983. Вып.29.
11. Кузнецов Б. П., Шалыто А. А. Метод независимых фрагментов для построения линеаризованных структурированных граф-схем алгоритмов, реализующих системы булевых формул // Автоматика и телемеханика. 1998. № 9.
12. Блох А. Ш. Граф-схемы и их применение. Минск: Вышэйшая школа, 1975.
13. Лофгрэн Л. Теория однородных переключающих сетей // Синтез релейных структур. М.: Наука, 1965.
14. Кириллов А. П., Кузнецов Б. П., Шалыто А. А. Минимизация числа безусловных переходов в бинарных программах // Судостроит. пром-сть. Сер. Автоматика и телемеханика. 1986. Вып.1.
15. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Настраиваемые модули для управляющих логических устройств. Л.: Энергоиздат, 1981.
16. Кузнецов О. П. О программной реализации логических функций и автоматов // Автоматика и телемеханика. 1977. № 7, 9.
17. Кузьмин В. А. Оценки сложности реализации ФАЛ простейшими видами бинарных программ // Методы дискретного анализа и теория кодов и схем. Новосибирск, 1976. Вып.29.
18. Autolog 32. Руководство пользователя. FF-Automation OY.
19. Рубинов В. И., Шалыто А. А. Метод построения граф-схем простых бинарных программ для систем булевых функций // Автоматика и вычисл. техника. 1986. № 4.
20. Артюхов В. Л., Рубинов В. И., Шалыто А. А. Метод построения обобщенных бинарных программ для систем булевых функций // Вопросы судостроения. Сер. Судовая автоматика. 1982. Вып. 27.
21. Рубинов В. И., Шалыто А. А. Построение граф-схем бинарных программ для систем булевых функций, заданных таблицами истинности // Автоматика и вычисл. техника. 1988. № 1.
22. Малюгин В. Д. Реализация булевых функций арифметическими полиномами // Автоматика и телемеханика. 1982. № 4.
23. Артюхов В. Л., Кондратьев В. Н., Шалыто А. А. Реализация булевых функций арифметическими полиномами // Автоматика и телемеханика. 1988. № 4.
24. Lee C. Y. Representation of switching circuits by binary decision programs // Bell Syst. Tech. J. 1959. N 7.
25. Arden B. W., Caller B. A., Graham R. M. An algorithm for translating boolean expressions // J. Assoc. Comput. Mach. 1962. N 4.
26. Clare C. R. Designing logic systems using state machines. New York: McGraw-Hill, 1973.
27. Boute R. T. The binary decision machine as programmable controllers // Euromicro News Lett. 1976. N 2.
28. Payne H. J., Meisel W. S. An algorithm for constructing optimal binary decision trees // IEEE Trans. on Comput. 1977. N 9.
29. Akers S. B. Binary decision diagrams // IEEE Trans. on Comput. 1978. N 6.
30. Akers S. B. Functional testing with binary decision diagrams // Proc. 8th Ann. IEEE Conf. Fault-Tolerant Comput. 1978.
31. Prather R. E., Casstevens H. T. Realization of boolean expressions by atomic digraphs // IEEE Trans. on Comput. 1978. N 8.
32. Cerny E., Mange D., Sanchez E. Synthesis of minimal binary decision trees // IEEE Trans. on Comput. 1979. N 9.

33. *Moret B. M.* Decision trees and diagrams // *Ass. Comput. Math., Comput. Surv.* 1982. N 12.
34. *Bryant R. E.* et al. Logic minimization algorithms for VLSI synthesis. Hingham, MA : Kluwer, 1984.
35. *Bryant R. E.* Graph-based algorithms for boolean function manipulation // *IEEE Trans. on Comput.* 1986. N 8.
36. *Brace K. S., Rudell R. L., Bryant R. E.* Efficient implementation of a BDD package // *Proceeding of the 27th Design Automation Conf. Orlando, FL.* 1990.
37. *Friedman S. J., Supowit K. J.* Finding the optimal variable ordering for binary decision diagrams // *IEEE Trans. on Comput.* 1990. N 5.
38. *Ishura N., Sawada H., Yajima S.* Minimization of binary decision diagrams based on exchanges of variables // *Proceeding of the Intern. Conf. on Computer-Aided Design.* Santa Clara, CA. 1991.
39. *Fujita M., Matsunaga Y., Kakuda T.* On variable ordering of binary decision diagrams for the application of multilevel synthesis // *Proceeding of the European Conf. on Design Automation.* Amsterdam, 1991.
40. *Brayton R. K., Sentovich E. M., Singh K. J.* et al. Sequential circuit design using synthesis and optimization // *Proceeding of the Intern. Conf. on Computer Design.* Cambridge, MA. 1992.
41. *Brayton R. K. et al.* SIS : A System for Sequential Circuit Synthesis // *Technical report UCB/ERL M92/41.* Electronics Research Laboratory. Univ. of California. Berkley, 1992.
42. *Jeong S. W., Kim T. S., Somenzi F.* An efficient method for optimal BDD ordering computation // *Intern. Conf. on VLSI and CAD (ICVC'93).* Taejon. Korea, 1993.
43. *Rudell R.* Dynamic variable ordering for ordered binary decision diagrams // *Proceeding of the Intern. Conf. on Computer-Aided Design.* Santa Clara, CA. 1993.
44. *Bahar R. I., Frofm E. A., Gaona C. M.* et al. Algebraic decision diagrams and their application // *Proceeding of the Intern. Conf. on Computer-Aided Design.* Santa Clara, CA. 1993.
45. *Panda S., Somenzi F., Plessior B. F.* Symmetry detection and dynamic variable ordering of decision diagrams // *Proceeding of the Intern. Conf. on Computer-Aided Design.* San Jose, CA. 1994.
46. *Panda S., Somenzi F.* Who are the variables in your neighborhood // *Proceeding of the Intern. Conf. on Computer-Aided Design.* San Jose, CA. 1995.
47. *Brayton R. K. et al.* VIS: A System for verification and synthesis // *Technical report UCB/ERL M95/104.* Electronics Research Laboratory. Univ. of California. Berkley, 1995.
48. *Карпов Ю. Г.* Теория алгоритмов и автоматов. Курс лекций. СПб.: СПбГТУ, Нестор, 1998.
49. *Артюхов В. Л., Шалыто А. А.* Судовые управляющие логические системы. Л.: ИПК СП, 1984.
50. *Шеннон К.* Работы по теории информации и кибернетике. М.: Изд-во иностр. литер., 1962.

Глава 17

Программная реализация автоматов с памятью

Известен [1, 2] метод аппаратной реализации автоматов с памятью, заданных граф-схемами (ГС) алгоритмов, которые для двоичных переменных могут быть названы бинарными графами (БГ).

При этом методы построения ГС для автоматов с памятью, реализуемых программно, не рассматривались, несмотря на то что в названии работы [1] присутствует термин «микропрограммный».

Прежде чем перейти к изложению методов программной реализации автоматов с памятью, перечислим основные отличия между граф-схемами, реализуемыми аппаратно и программно:

- ввод входных переменных в ГС первого типа осуществляется в каждой условной вершине (УВ), в то время как в ГС второго типа эти переменные обычно вводятся одним оператором «Ввод», являющимся первым оператором в теле ГС;

- в ГС первого типа в каждой операторной вершине (ОВ) указываются обозначения только тех выходных переменных, которые в ней принимают единичные значения, а для остальных, не указанных в этой вершине, по умолчанию считают, что они принимают в этой ОВ нулевые значения. В ГС второго типа в операторных вершинах указываются как единичные, так и нулевые значения выходных переменных, а для умалчиваемых выходных переменных полагают, что они сохраняют предыдущее значение;

- ГС первого типа могут содержать внутренние обратные связи, в то время как в ГС второго типа такие связи обычно не используются, а применяется одна внешняя обратная связь, соединяющая оператор «Вывод» с оператором «Ввод», что поддерживается язы-

ками инструкций большинства программируемых логических контроллеров;

— в ГС первого типа значения всех выходных переменных (в том числе и умалчиваемых) выдаются на выход в каждой ОВ, а в ГС второго типа на выход выдаются значения всех выходных переменных (в том числе и умалчиваемых) из тех ОВ, которые связаны с оператором «Вывод» непосредственно или через условные вершины. Кроме того, предыдущие значения выходных переменных могут сохраняться и выдаваться на выход при прохождении таких контуров, образованных внешней обратной связью, которые не содержат операторных вершин и называются поэтому «пустыми».

17.1. Метод независимых фрагментов

Предположим, что в общем случае автомат задан системой булевых формул (СБФ), состоящей из $2N$ формул, первые N из которых имеют вид: $z_{11} = f_1(z_1, \dots, y_m), \dots, z_{k1} = f_k(z_1, \dots, y_m), y_{11} = f_{k+1}(z_1, \dots, y_m), \dots, y_{m1} = f_N(z_1, \dots, y_m)$, где $k + m = N$, а оставшиеся N формул являются переобозначениями переменных: $z_1 = z_{11}, \dots, z_k = z_{k1}, y_1 = y_{11}, \dots, y_m = y_{m1}$.

Предлагаемый метод построения структурированного бинарного графа (СБГ) по этой СБФ состоит в следующем:

— с помощью метода независимых фрагментов для автоматов без памяти, изложенного в разд. 16.3, по первым N формулам строится ядро тела СБГ;

— к ядру последовательно подсоединяется одна или несколько операторных вершин, в которых записываются оставшиеся N формул системы;

— к построенной структуре последовательно подсоединяется одна или несколько ОВ, в которых выполняется обнуление переменных, введенных лишь с целью переобозначения и имеющих в СБФ двойные индексы.

Пример 17.1. Построить СБГ для R -триггера, описываемого СБФ

$$z_1 = !R(S \vee z), \quad z = z_1.$$

Построим на основе первой БФ оператор структуры

$$F = !RSz_1|!Rzz_1$$

и выполним его минимизацию:

$$F = !R(Sz_1|zz_1).$$

Искомый СБГ представлен на рис. 17.1.

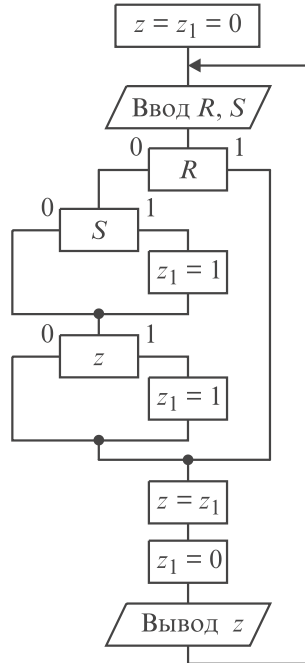


Рис. 17.1

Пример 17.2. Построить СБГ для счетного триггера, описываемого системой булевых формул:

$$z_1 = !xz \vee x!y, \quad y_1 = !xz \vee xy, \quad z = z_1, \quad y = y_1.$$

Построим на основе первых двух БФ оператор структуры

$$F = !xzz_1 | x!yz_1 | !xzy_1 | xyy_1$$

и выполним его минимизацию:

$$F = !xz(z_1 | y_1) | x(!yz_1 | yy_1).$$

Введем в этот оператор квадратные скобки, отображающие использование управляющих конструкций «полный выбор»:

$$F = [!xz(z_1 | y_1) \vee x[!yz_1 \vee yy_1]].$$

Искомый СБГ представлен на рис. 17.2.

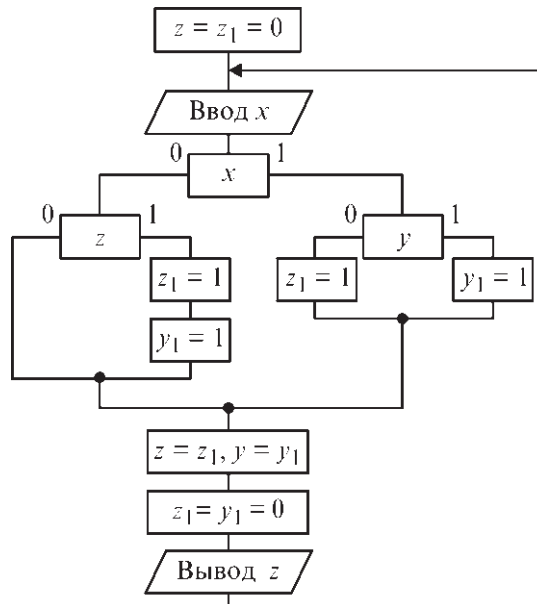


Рис. 17.2

Недостаток этого метода, изложенного в [3], состоит в том, что он строит СБГ, которые достаточно трудно понять, так как они не изоморфны эквивалентным им графам переходов (ГП).

17.2. Канонический и распределительный методы

Особенность применения этих методов для автоматов с памятью состоит в том, что после построения с их помощью тела бинарного графа и замыкания внешней обратной связи может появиться возможность исключения некоторых условных и операторных вершин, связанных с сохранением предыдущих значений выходных и (или) внутренних переменных.

Предположим, что требуется построить бинарный граф, реализующий R -триггер, по кодированной таблице переходов (КТП). Структура БГ зависит от порядка переменных в этой таблице. Построим БГ с помощью рассматриваемых методов для всех (шести) возможных порядков переменных, используя только три таблицы. Отметим, что в отличие от метода независимых фрагментов в данном случае отсутствует необходимость применять переобозначения переменных.

Пример 17.3. Построить бинарный граф по КТП

$$z(R, z, S) = |0111 \ 0000|^T.$$

1. Используя канонический метод, построим СБГ1 (рис. 17.3).

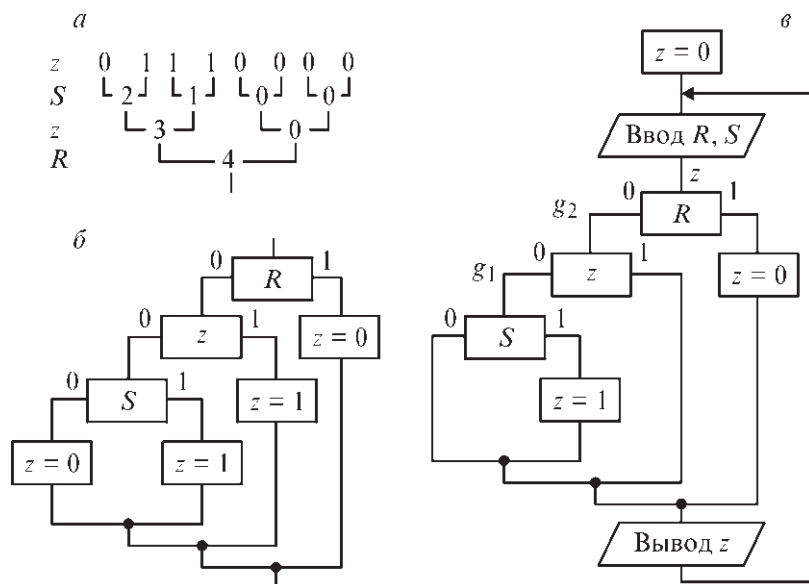


Рис. 17.3

Тело построенного СБГ1 состоит из пяти вершин, что характеризует его сложность. Так как СБГ1 имеет древовидную структуру, выполним его верификацию, как и для автоматов без памяти, от выхода к входу:

$$g_1 = z!S \vee 1 \& S = S \vee z; \quad g_2 = g_1!z \vee zz = S \vee z; \\ z = g_2!R \vee 0 \& R = !R(S \vee z).$$

СБГ1 может быть запрограммирован, например, следующим образом:

$$z = R ? (0 : (z ? z : (S ? 1 : z))).$$

2. Применяя распределительный метод, построим структуру, приведенную на рис. 17.4.

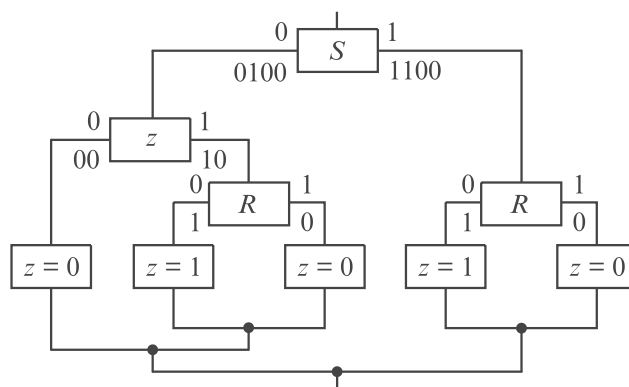


Рис. 17.4

На основе этой структуры могут быть построены БГ1 (рис. 17.5) и СБГ2 (рис. 17.6).

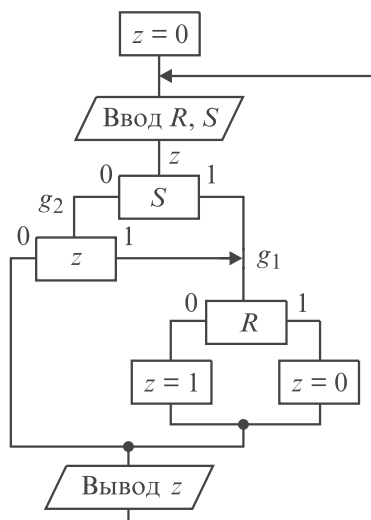


Рис. 17.5

Сложность этих бинарных графов составляет пять и семь соответственно.

Так как БГ1 не содержит последовательно соединенных блоков, то выполним его верификацию от выхода к входу:

$$g_1 = 1 \& !R \vee 0 \& R = !R; \quad g_2 = z!z \vee g_1 z = !Rz;$$

$$z = g_2!S \vee g_1 R = !R(S \vee z).$$

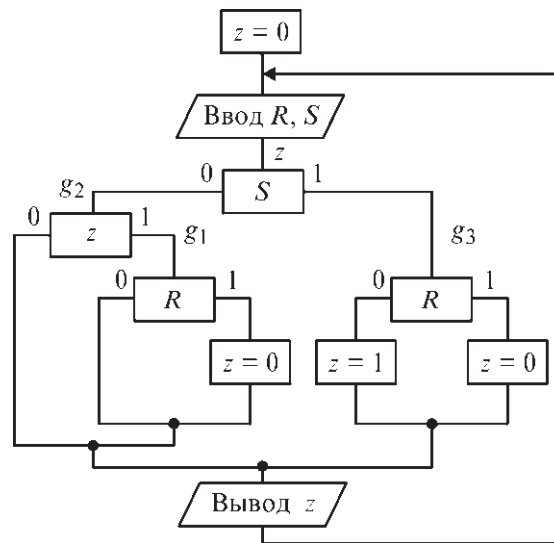


Рис. 17.6

При программировании тела БГ1 придется использовать метку:

```

    if (S) goto g1;
    if (z)
g1: { if (R) z=0;
      else z=1; }

```

Выполним верификацию СБГ2:

$$g_1 = z!R \vee 0 \& R = z!R; \quad g_2 = !zz \vee g_1 z = !Rz;$$

$$g_3 = 1 \& !R \vee 0 \& R = !R; \quad z = g_2!S \vee g_3S = !R(S \vee z).$$

Пример 17.4. Построить бинарный граф по КТП

$$z(R, S, z) = | 0111 \ 0000 |^T.$$

1. Применяя канонический метод, построим СБГ3 (рис. 17.7).

Сложность этого графа равна четырем. Выполним его верификацию:

$$g = z!S \vee 1 \& S = z \vee S, \quad z = g!R \vee 0 \& R = !R(S \vee z).$$

Обратим внимание на то, что при одном и том же столбце значений в этом и предыдущем примерах при использовании одного и того же метода для рассматриваемого автомата с памятью постро-

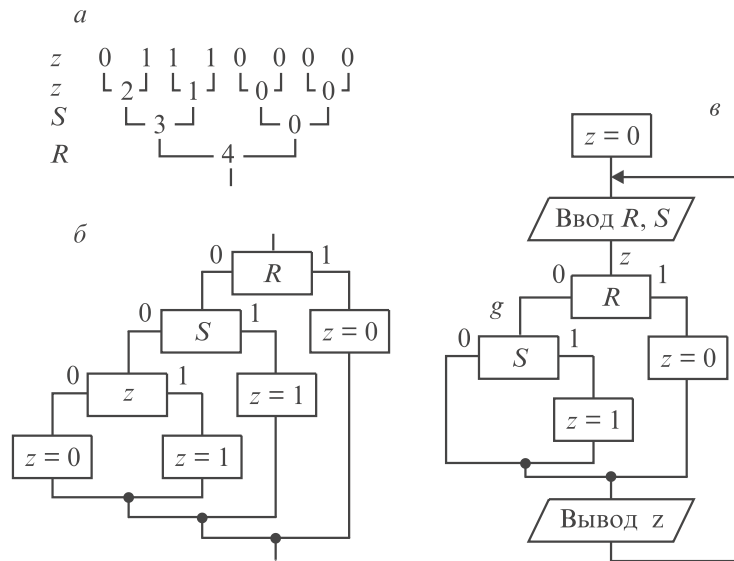


Рис. 17.7

ены СБГ разной сложности, что невозможно для автоматов без памяти.

Тело построенного СБГ программируется следующим образом:

```

if (R)    z=0;
else
if (S)    z=1;

```

2. Применяя распределительный метод, построим структуру, приведенную на рис. 17.8.

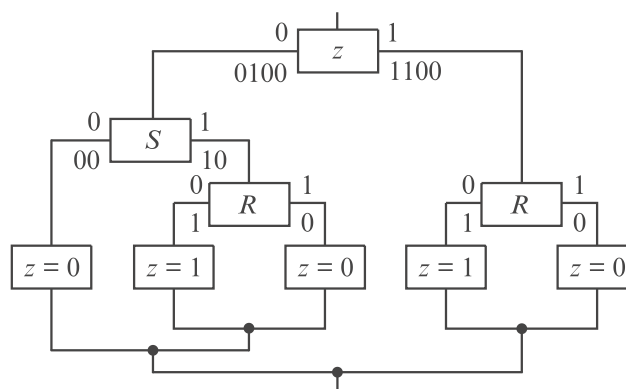


Рис. 17.8

На основе этой структуры могут быть построены БГ2 (рис. 17.9), БГ3 (рис. 17.10) и СБГ4 (рис. 17.11). Сложность этих графов составляет пять, пять и шесть соответственно.

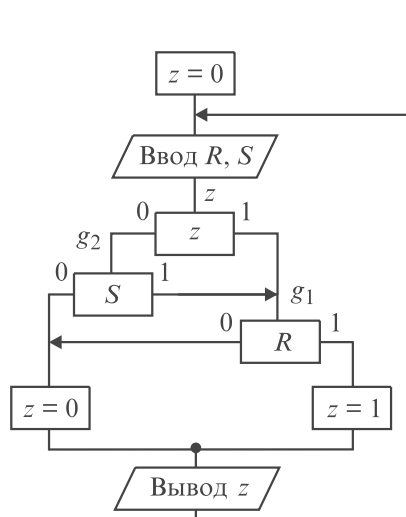


Рис. 17.9

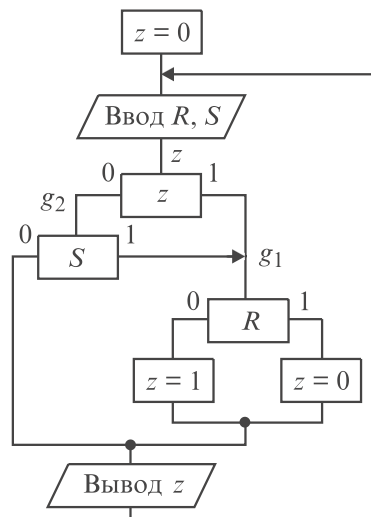


Рис. 17.10

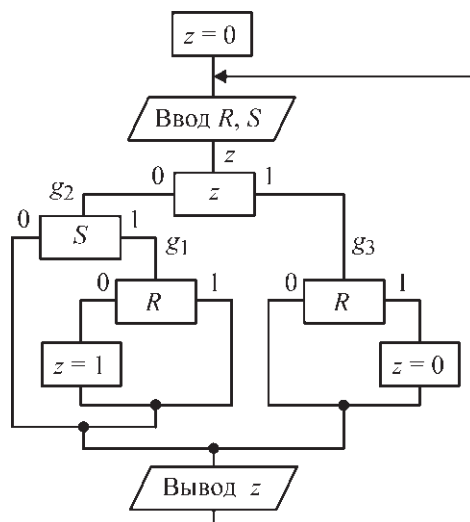


Рис. 17.11

Выполним верификацию БГ2 (рис. 17.9):

$$g_1 = 1 \& !R \vee 0 \& R = !R; \quad g_2 = 0 \& !S \vee g_1 S = S!R;$$

$$z = g_2!z \vee g_1 z = (z \vee S)!R.$$

Выполним верификацию БГЗ (рис.17.10)

$$g_1 = 1 \ \& \ !R \vee 0 \ \& \ R = !R; \ g_2 = 7 \\ z = g_2!z \vee g_1z$$

Выполним верификацию СБГ

$$g_1 = 1 \ \& \ !R \vee zR = !z$$

СБГ4 соответствует
фу переходов

Эти СБГ и ГП могут
мощью конструкции s

са

са

Пример 17.5. Построить структурированный бинарный граф по КТП

$$z(S, R, z) = |0100 \ 1100|^T.$$

1. Используя канонический метод и подъем одной из операторных вершин вверх, построим СБГ6 (рис. 17.14).

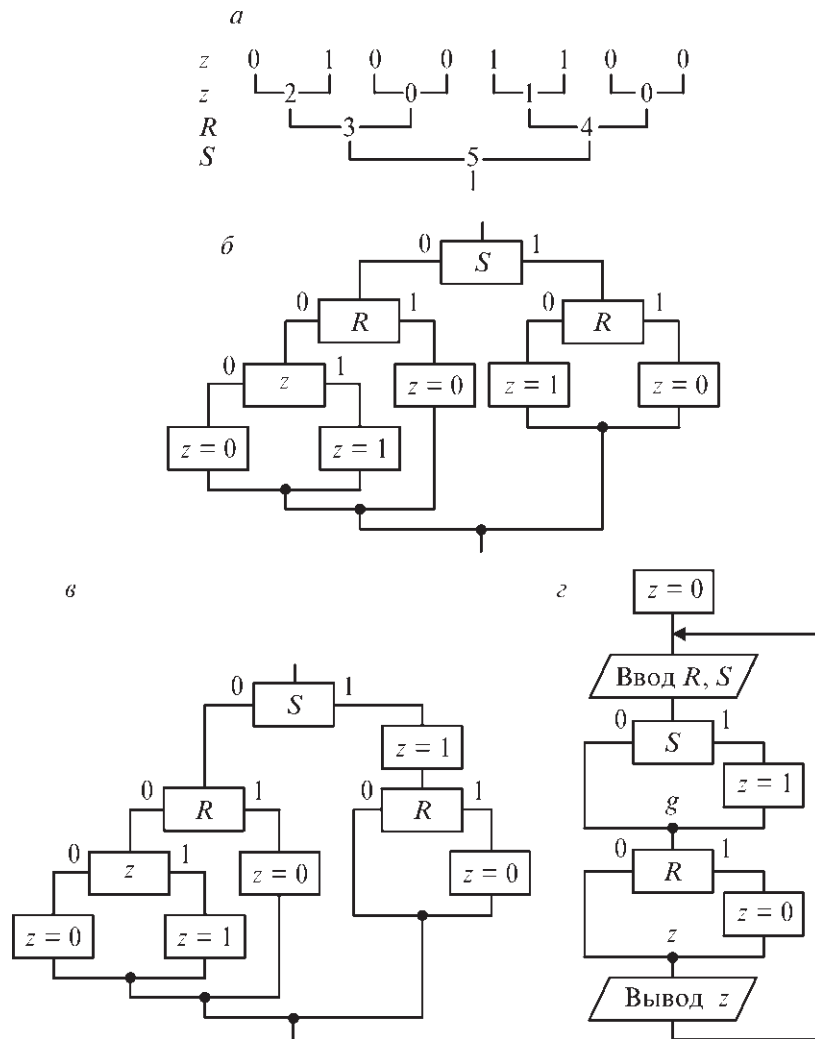


Рис. 17.14

Сложность этого графа равна четырем. Так как тело этого СБГ состоит только из последовательно соединенных блоков с одним

входом и одним выходом, то выполним его верификацию от входа к выходу:

$$g = z!S \vee 1 \& S = z \vee S, \quad z = g!R \vee 0 \& R = !R(S \vee z).$$

Тело СБГ в этом случае реализуется простейшей программой:

```
if (S)  z=1;
if (R)  z=0;
```

которая, однако, понимается хуже предыдущей программы, так как она в явном виде не содержит предшествующих значений переменной z .

2. Применяя распределительный метод, построим структуру, приведенную на рис. 17.15.

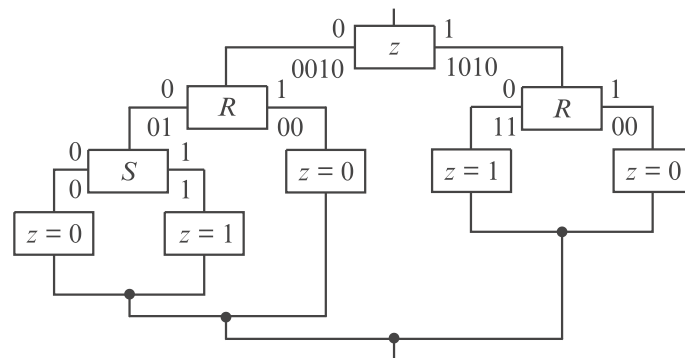


Рис. 17.15

На основе этой структуры построим СБГ7 (рис. 17.16).

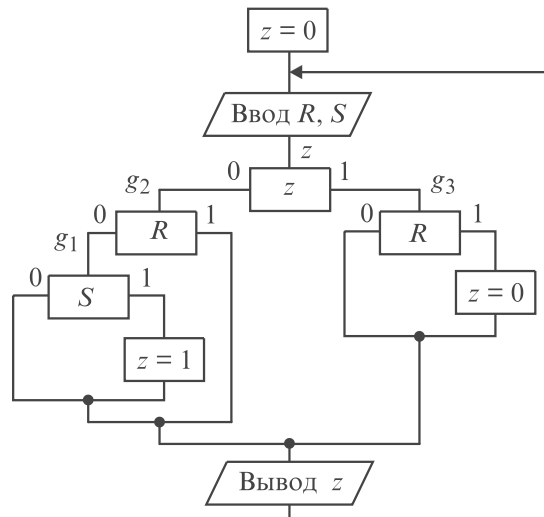


Рис. 17.16

Выполним верификацию этого графа:

$$g_1 = z!S \vee zS = S \vee z; \quad g_2 = g_1!R \vee zR = z \vee S!R;$$

$$g_3 = z!R \vee 0 \& R = !Rz; \quad z = g_2!z \vee g_3z = !R(S \vee z).$$

СБГ7 соответствует СБГ8 (рис. 17.17), который изоморфен графу переходов (рис. 17.18).

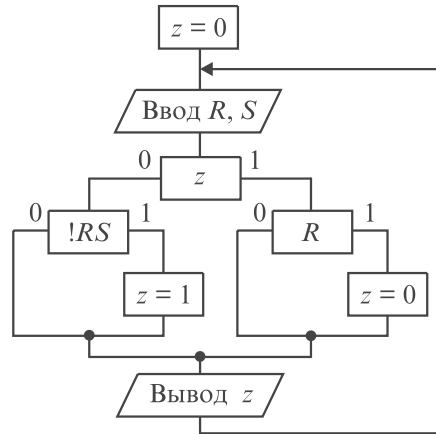


Рис. 17.17

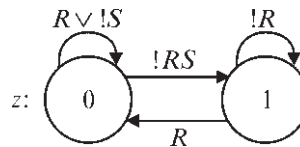


Рис. 17.18

Эти СБГ и ГП могут быть изоморфно запрограммированы с помощью конструкции switch:

```
switch(z) {
case 0:  if (!R&S)  z=1;
         break;
case 1:  if (R)     z=0;
         break;
}
```

После завершения построения БГ с помощью рассматриваемых методов отметим, что пять из них (БГ1, БГ2, БГ3, СБГ1 и СБГ3) могут быть построены также и при использовании формульного метода. Например, тело БГ3 (рис. 17.10) изоморфно БГ4 (рис. 17.19), построенному по булевой формуле $z = (z \vee S)!R$.

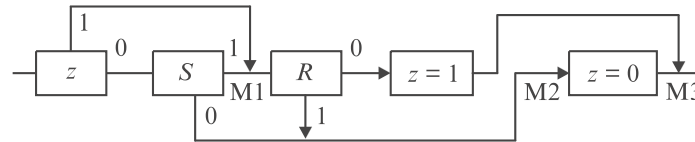


Рис. 17.19

Этот граф реализуется наиболее неструктурированной программой из рассмотренных:

```
        if (z)   goto M1;
        if (!S)  goto M2;
M1:     if (R)   goto M2;
        z=1;
        goto M3;
M2:     z=0;
M3:
```

Граф БГ2 (рис. 17.9) обладает уникальным свойством: из всех рассмотренных графов он в минимизированном виде не имеет «пустых» (от операторных вершин) путей, т.е. на каждом из них устанавливается одно из значений выходной переменной z , являющейся одновременно внутренней (промежуточной). Это свойство, характерное для бинарных графов, реализующих автоматы без памяти, сохраняется в данном случае и для автомата с памятью, так как этот граф содержит условную вершину, помеченную переменной z . При отсутствии таких вершин наличие в графе «пустых» путей свидетельствует о том, что он реализует автомат с памятью. В этом случае весьма простой по структуре бинарный граф может иметь весьма сложное поведение, полностью понять которое в общем случае можно лишь построив эквивалентный граф переходов.

Сравнение построенных бинарных графов может быть выполнено по следующим критериям, применяемым традиционно: быстродействие, число вершин, структурированность и сложность реализующей программы.

По первому критерию наилучшей является СБГЗ, по второму — СБГЗ и СБГ6. Структурированными являются все графы, за исключением БГ1—БГ3. Наиболее простой (из рассматриваемых ветвящихся программ) является программа, реализующая СБГ6.

Таким образом, наилучшими по традиционным критериям являются графы СБГЗ и СБГ6.

Оба эти графа обладают тем свойством, что условные вершины в них помечены только символами входных переменных, в то время как такие вершины, помеченные символом выходной переменной z , в этих графах отсутствуют.

Граф СБГЗ обладает тем преимуществом, что на каждом пути от входа к выходу выходной переменной z присваивается значение не более одного раза, а структура СБГ6 обеспечивает его изоморфную реализацию наиболее простой ветвящейся программой.

Однако оба эти графа обладают, по мнению автора, существенным недостатком, который может свести на нет указанные достоинства: они недостаточно понятны в том смысле, что при $R = S = 0$

сохраняют предыдущее значение переменной z без указания ее конкретного значения и поэтому прочесть это значение непосредственно по графу (без его «прокрутки») невозможно. В этих графах для определения значения переменной z на «пустых» путях необходимо использовать внемодельные средства — память ЭВМ или человека, в которой фиксируется предшествующее значение этой переменной.

Эта проблема имеет место и в общем случае. Поэтому не случайно Н. Вирт назвал свою книгу «Программы = алгоритмы + данные» [4], отмечая в ней, что при реализации алгоритмов граф-схемами последние в общем случае в явном виде указывают лишь связи по управлению, в то время как явные связи по данным в них отсутствуют.

Однако если ЭВМ обладает большой оперативной памятью, содержание которой весьма легко прочесть, то такая память у человека крайне ограничена [5], и поэтому при использовании БГ не только для программирования, но и в качестве языка общения между Специалистами различных областей знаний, по мнению автора, должен применяться новый критерий качества бинарных графов — **п о н я т н о с т ь**.

Так как понятность является субъективным фактором, введем формальный критерий понятности бинарного графа, реализующего автомат с памятью, — его изоморфизм с функционально эквивалентным графом переходов без флагов и умолчаний [3, 6]. При этом наиболее понятными, по мнению автора, являются бинарные графы, изоморфные таким графам переходов, которые соответствуют автоматам без выходного преобразователя или автоматам Мура, так как для каждого автомата, принадлежащего этим классам, граф переходов совпадает с графом достижимых маркировок [3].

По этому критерию наилучшими из рассмотренных графов являются СБГ5 и СБГ8. Несмотря на наличие «пустых» путей в них, они обладают такой структурой, что по этим графам, поднимаясь от оператора «Вывод» вверх, можно непосредственно определить предшествующее значение переменной z на этих путях, так как указанные графы начинаются с дешифратора значений этой переменной. Последний можно считать дешифратором внутренних состояний (в дальнейшем — **д е ш и ф р а т о р с о с т о я н и й**) изоморфного автомата, граф переходов которого, в свою очередь, обладает изоморфизмом с конструкцией `switch` языка СИ, изоморфной также и СБГ с указанными свойствами.

В графах СБГ3 и СБГ6 переменная, характеризующая состояния, в условных вершинах не применяется, а в графах БГ1, СБГ1 и СБГ2 эта переменная используется в качестве некоторой внутренней переменной, делая их весьма непонятными.

Если в общем случае построение СБГ начинать с построения дешифратора состояний, а не событий, как это обычно принято, то можно утверждать, что СБГ с такой структурой содержат в явном виде связи не только по управлению, но и по данным. Они могут быть названы «автоматные схемы алгоритмов» (АСА).

Из изложенного следует, что имеет место взаимно однозначное соответствие (изоморфизм) между тремя видами описаний алгоритмов — АСА, ГП и конструкцией *switch*, по каждому из которых два других могут быть восстановлены непосредственно (без дополнительных вычислений) (рис. 17.20).

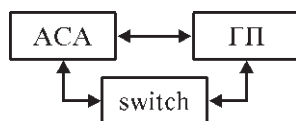


Рис. 17.20

17.3. Модификация канонического метода

Используем метод, изложенный в разд. 16.4.6, для реализации автоматов с памятью.

Пример 17.6. Реализовать последовательный одноразрядный двоичный сумматор.

Рис. 17.21 и 17.22 иллюстрируют применение рассматриваемого метода.

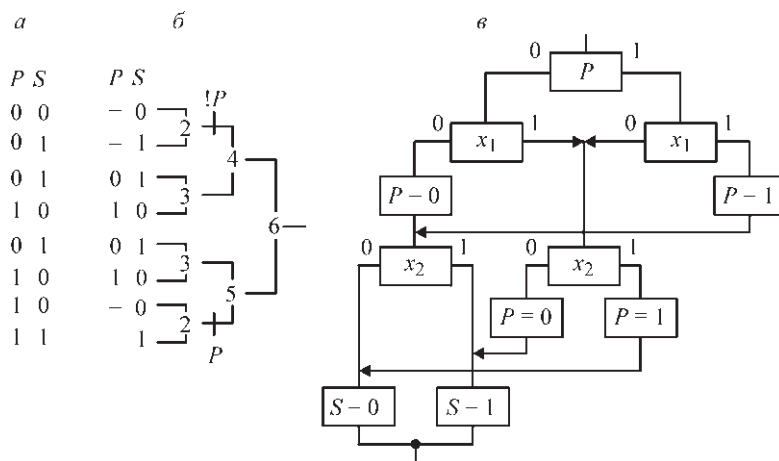


Рис. 17.21

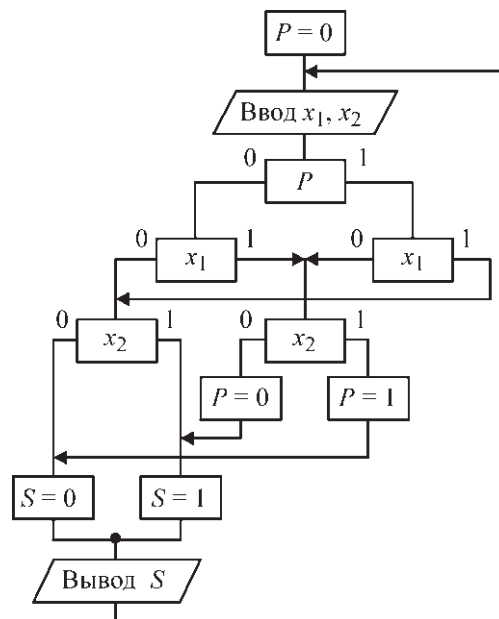


Рис. 17.22

17.4. Верификация бинарных графов

Если в бинарном графе с одной обратной связью (внешней), условные вершины которого помечены только символами входных переменных, на каждом пути от входа к выходу в операторных вершинах указаны значения всех выходных переменных, то такой граф реализует автомат без памяти.

Как показано в разд. 17.2, если в БГ указанного типа условные вершины могут быть помечены также и значениями выходных переменных, то такой граф реализует автомат с памятью.

Если в БГ с одной обратной связью указанного типа для каждой переменной существует хотя бы один путь, на котором в операторных вершинах не записано ее значение, то такой граф реализует автомат с памятью, являющийся автоматом без выходного преобразователя.

Для автоматов с памятью под верификацией бинарного графа будем понимать построение по нему эквивалентного ГП.

Пример 17.7. Верифицировать СБГ (рис. 17.23).

Этот БГ является комбинационным по переменной z_3 и автоматным по переменным z_1 и z_2 .

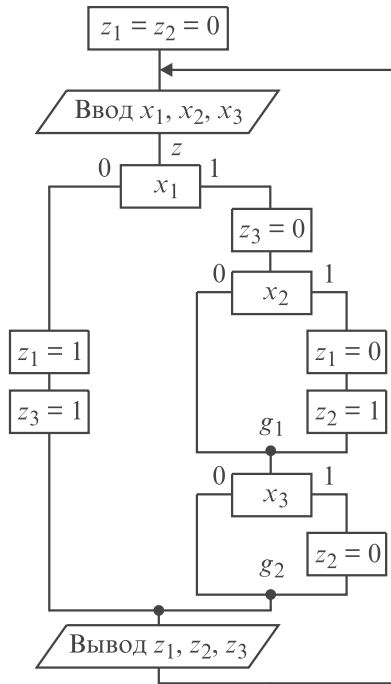


Рис. 17.23

Верификацию переменной z_3 будем проводить от выхода к входу:

$$z_3 = 1 \& !x_1 \vee 0 \& x_1 = !x_1.$$

Перейдем к верификации переменных z_1 и z_2 . Так как по этим переменным заданный СБГ состоит из двух последовательных блоков, охваченных третьим, то для первых двух блоков верификацию будем проводить сверху вниз, а для третьего (после верификации последовательно соединенных блоков) — снизу вверх:

$$\begin{aligned}
 g_{11} &= z_1 !x_2 \vee 0 \& x_2 = !x_2 z_1; \\
 g_{12} &= z_2 !x_2 \vee 1 \& x_2 = x_2 \vee z_2; \\
 g_{21} &= g_{11} !x_3 \vee g_{11} x_3 = !x_2 z_1; \\
 g_{22} &= g_{12} !x_3 \vee 0 \& x_3 = !x_3 (x_2 \vee z_2); \\
 z_1 &= 1 \& !x_1 \vee g_{21} x_1 = !x_1 \vee !x_2 z_1; \\
 z_2 &= z_2 !x_1 \vee g_{22} x_1 = !x_1 z_2 \vee !x_3 (x_1 x_2 \vee z_2).
 \end{aligned}$$

Каждой из полученных двух формул соответствует граф переходов с двумя вершинами. Однако, так как эти графы должны вычисляться параллельно и иметь общие входные переменные, такое описание рассматриваемой части алгоритма проблему понятности не решает. Она решается, если по найденной СБФ построить один граф переходов автомата без выходного преобразователя (рис. 17.24).

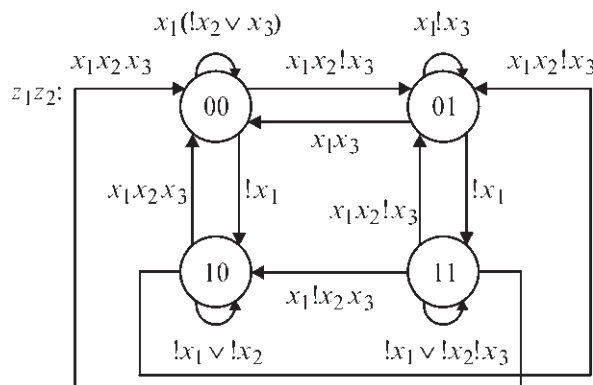


Рис. 17.24

Вводя переменную z_3 в построенный граф, получим ГП смешанного автомата (рис. 17.25), который эквивалентен заданному СБГ.

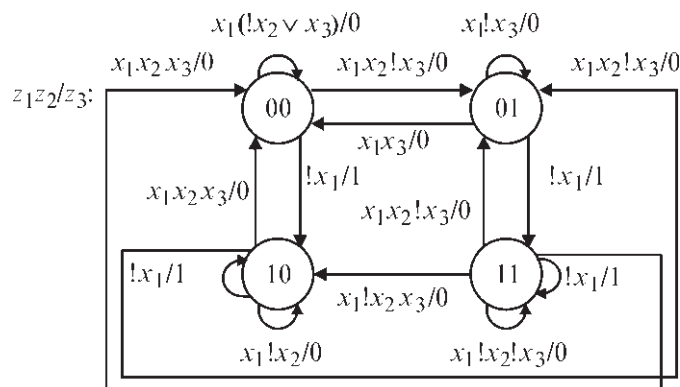


Рис. 17.25

Из приведенного примера следует, что сравнительно простой по структуре СБГ имеет весьма сложное поведение, описываемое построенным графом переходов.

Пример 17.8. Верифицировать бинарный граф (рис. 17.22).

Перечислим пути в бинарном графе с учетом умалчиваемых значений переменной P :

1. $!P!x_1!x_2!S!P$; 2. $!P!x_1x_2S!P$; 3. $!Px_1!x_2!PS$; 4. $!Px_1x_2P!S$;
5. $P!x_1!x_2!PS$; 6. $P!x_1x_2P!S$; 7. $Px_1!x_2!SP$; 8. Px_1x_2SP .

Запишем эти выражения так, чтобы в каждом из них на двух последних позициях сначала располагалась переменная P или ее инверсия, а затем переменная S или ее инверсия:

1. $!P!x_1!x_2!S!P$; 2. $!P!x_1x_2S!P$; 3. $!Px_1!x_2S!P$; 4. $!Px_1x_2!SP$;
5. $P!x_1!x_2S!P$; 6. $P!x_1x_2!SP$; 7. $Px_1!x_2!SP$; 8. Px_1x_2SP .

Объединяя второе и третье выражения, получим $!P(x_1 \oplus x_2)S!P$, а шестое и седьмое — $P(x_1 \oplus x_2)!SP$. Исходя из изложенного, можно построить ГП автомата Мили (рис. 17.26).

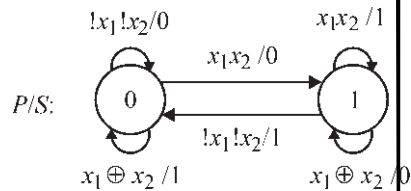
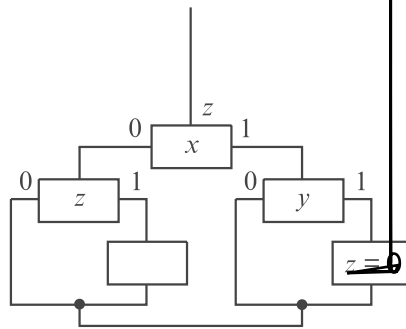


Рис. 17.26 Рис. 17.327

Построенный граф переходов существенно более компактен, обозрим и понятен по сравнению с исходным бинарным графом.

Пример 17.9. Верифицировать структурированный БГ (рис. 17.27).



Структура этого СБГ позволяет проводить его верификацию от входа к выходу:

$$\begin{aligned} g_{11} &= z!z \vee zz = z; & g_{21} &= 1 \& !y \vee 0 \& y = !y; \\ g_{12} &= 0 \& !z \vee 1 \& z = z; & g_{22} &= y!y \vee yy = y; \\ z_1 &= g_{11}!x \vee g_{21}x = !xz \vee x!y; \\ y_1 &= g_{12}!x \vee g_{22}x = !xz \vee xy. \end{aligned}$$

По каждой из последних двух формул может быть построен ГП. Но так как эти ГП связаны между собой по всем переменным (входной, внутренней и выходной) и должны вычисляться параллельно, то понять алгоритм по этому описанию весьма сложно. Поэтому построим по полученной СБФ граф переходов автомата без выходного преобразователя (рис. 17.28).

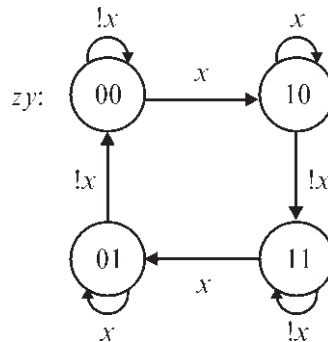


Рис. 17.28

Заданный алгоритм становится еще более понятным, если построенный граф преобразовать в граф переходов автомата Мура (рис. 17.29), так как при этом внутренняя и выходная переменные находятся на разных уровнях, что более соответствует физическому смыслу этих типов переменных.

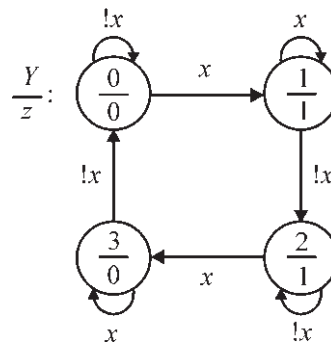


Рис. 17.29

Из рассмотрения последнего графа переходов следует, что заданный СБГ описывает работу счетного триггера, что весьма трудно понять по исходному заданию.

Из этого примера следует, что описание алгоритма с помощью графа переходов, в котором вершины пронумерованы десятичными числами, существенно более понятно по сравнению с описанием, в котором применяется БГ или СБГ без дешифратора состояний. Это объясняется тем, что в таком графе переходов используются понятия «состояние» и «многозначное кодирование состояний», а в указанном БГ или СБГ применяются двоичные переменные, влияющие друг на друга и не позволяющие непосредственно по этой модели определить пространство состояний автомата и переходы в этом пространстве.

Пример верификации бинарного графа с внутренними обратными связями приведен в [3, 6].

17.5. Построение бинарных графов по графам переходов

Возможность использования методов построения бинарных графов, рассмотренных выше, ограничена, так как метод независимых фрагментов требует знания формул, описывающих автомат, а остальные методы, связанные с табличным заданием автомата, предназначены для решения задач небольшой размерности.

Покажем, что бинарные графы весьма просто могут быть построены по графам переходов. На первый взгляд кажется, что при наличии графа переходов нет необходимости в построении бинарного графа, однако она может возникнуть ввиду требований ГОСТа [7], который определяет применение бинарных графов для описания алгоритмов.

Для кодирования вершин графа переходов в общем случае могут применяться различные виды кодирования, что влияет на структуру бинарного графа, которая зависит также и от класса автомата с памятью, которому соответствует граф переходов [3, 6].

17.5.1. Автоматы без выходного преобразователя

Принудительное кодирование состояний. Если состояния автомата могут быть различимы только за счет использования значений выходных переменных, то будем называть такой вид кодирования принудительным [3].

Пусть задан ГП (рис. 17.30).

Так как все вершины этого графа помечены различными кортежами значений выходных переменных, то в данном случае может

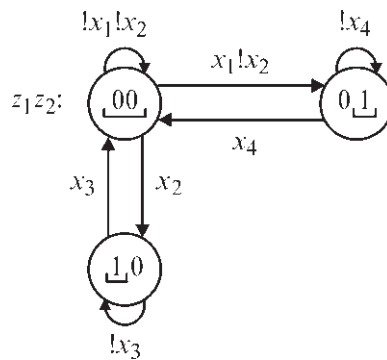


Рис. 17.30

применяться принудительное кодирование вершин, которое не требует введения дополнительных (внутренних) переменных для их различения.

Структурированный БГ, построенный по этому графу, приведен на рис. 17.31.

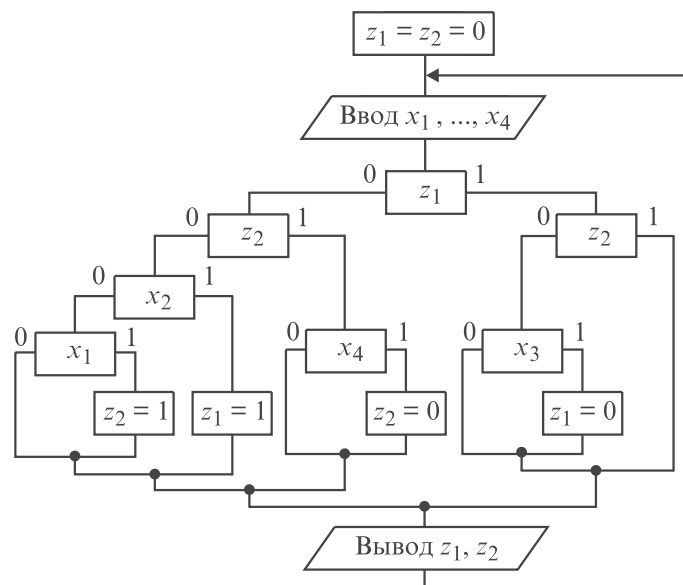


Рис. 17.31

Анализ заданного графа переходов позволяет упростить построенный СБГ за счет исключения в нем правой условной вершины с пометкой z_2 и подключения входа такой вершины с пометкой x_3 к единичному выходу условной вершины с пометкой z_1 (рис. 17.32).

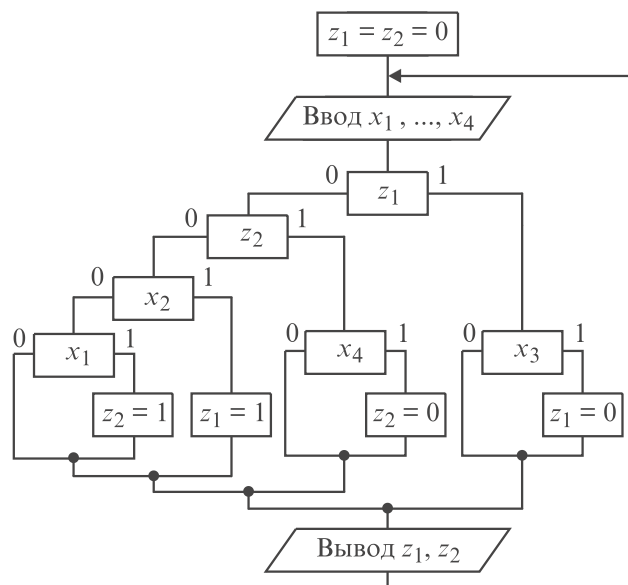


Рис. 17.32

Для упрощенного СБГ не может быть построен изоморфный ему граф переходов. Потеря изоморфизма приводит к тому, что этот СБГ становится плохо понимаемым, так как, на первый взгляд, кажется, что при $z_1 = 1$ значение z_2 может быть произвольным. Однако это не так, если учесть предысторию.

В силу того что в данном случае цель построения СБГ состоит прежде всего в том, чтобы весьма просто понимать его, а не в том, чтобы в нем «разбираться», то применение упрощенного СБГ по этому критерию нецелесообразно.

Обратим внимание, что в рассматриваемом примере для кодирования вершин могут использоваться неполные кортежи значений выходных переменных (неполные коды) [3]. Эти коды в графе переходов на рис. 17.30 отмечены скобками. Применение таких кодов целесообразно при программной реализации на основе СБФ, которая в данном случае имеет вид:

$$z_{11} = x_2!z_1!z_2 \vee !x_3z_1; \quad z_2 = x_1!x_2!z_1!z_2 \vee !x_4z_2; \quad z_1 = z_{11}.$$

При этом отметим, что построить ветвящийся бинарный граф, изоморфный ГП с неполными кодами, не удастся.

Принудительно-свободное кодирование состояний. Если состояния автомата для их различения кодируются значениями выходных и дополнительно вводимых внутренних переменных, то такой вид кодирования будем называть принудительно-сво-

бодным [3], так как в этом случае первая часть каждого кода определяется принудительно значениями выходных переменных в этом состоянии, а вторая ввиду программной реализации автомата — свободно.

Граф переходов на рис. 17.33, описывающий алгоритм работы счетного триггера, не может быть реализован непосредственно, так как в нем соответствующие пары вершин неразличимы.

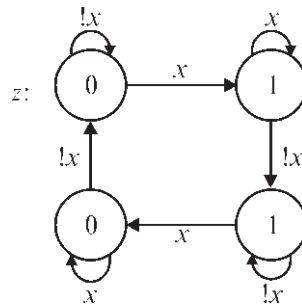


Рис. 17.33

Вводя дополнительную (промежуточную) переменную, используем принудительно-свободное кодирование, которое в данном случае является также и противогоночным (рис. 17.28). Этот граф переходов изоморфен СБГ на рис. 17.34. В теле этого графа первые два яруса образуют дешифратор состояний.

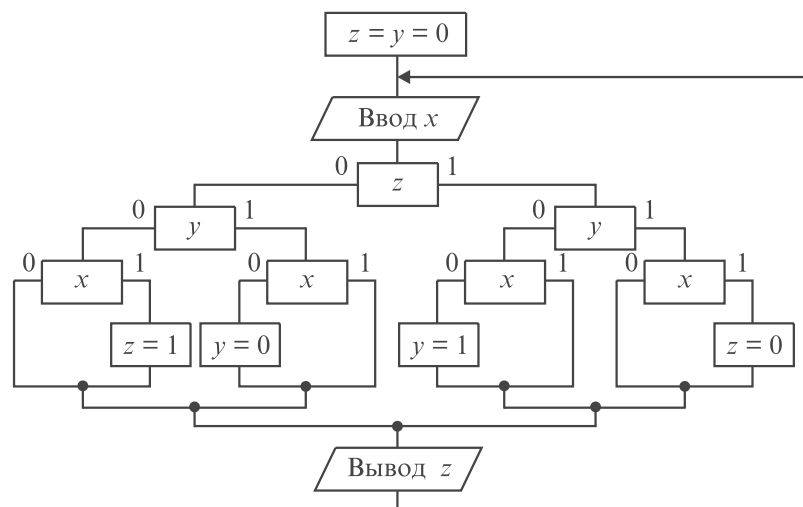


Рис. 17.34

В [3] показано, что этот СБГ может быть существенно упрощен за счет потери изоморфизма с графом переходов, если учесть,

что в данном случае для обеспечения корректной реализации в каждой вершине графа (рис. 17.28) может быть выбрано значение лишь одной переменной, которое совместно со значением входной переменной, помечающей петлю этой вершины, обеспечит сохранение состояния (рис. 17.35).

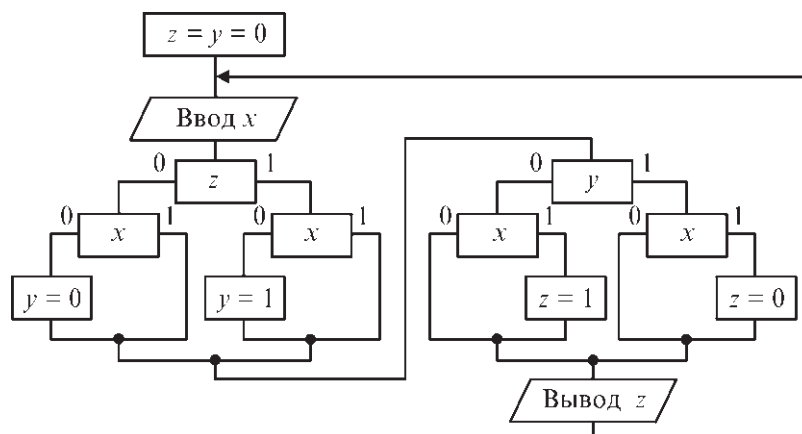


Рис. 17.35

Этот СБГ в отличие от предыдущего существенно менее понятен, так как в нем «разрушен» дешифратор состояний за счет применения в каждом блоке входной переменной x между переменными z и y , характеризующими состояния автомата.

На рис. 17.27 представлен еще более простой СБГ, реализующий счетный триггер [3]. Однако этот граф еще менее понятен, так как в нем от дешифратора состояний не осталось и следа.

Последний СБГ характерен для событийного программирования [8], так как он начинается с дешифратора значений входной переменной, называемого также дешифратором событий, каждое из которых совместно со значением выходной (внутренней) переменной приводит к формированию определенного значения внутренней (выходной) переменной.

Сформулируем несколько общих положений, характеризующих предлагаемый подход.

В [3] были введены понятия «автоматное программирование» и «автоматное проектирование программ», которые базируются на совместном использовании таких понятий, как «состояние» и «событие». В рамках предложенного в [3] подхода основным является понятие «состояние», так как при этом в любой момент времени можно в явном виде определить, в какой точке «пространства состояний» находится программа. Под состоянием при этом (в отличие от [9]) понимается не совокупность значений внутренних переменных, а некоторая абстракция, которая

лишь после кодирования приобретает связь с переменными. Например, для такого объекта, как вода, основным является то, что она может находиться в трех состояниях (жидкое, твердое и газообразное), в то время как значения переменной, характеризующие эти состояния, являются вторичными. Отметим, что это, в частности, позволяет применять ГП и при «нечетком управлении».

Если программирование проводить, как это делается традиционно, на основе использования внутренних переменных, а не состояний, то обычно приходится применять большое число таких переменных, являющихся, как правило, *д в о и ч н ы м и*, при непосредственном рассмотрении которых весьма трудно определить число состояний, кодируемых ими.

Если проектирование программы начинать с выделения состояний, то априорное знание их числа позволяет применять *м н о г о з н а ч н о е* кодирование, при котором для каждой компоненты программы, соответствующей одному автомату, достаточно использовать только *о д н у* внутреннюю переменную вне зависимости от числа выделенных состояний, значность которой равна числу этих состояний. При этом код состояния совпадает с номером соответствующей вершины в графе переходов. Это позволяет ввести в программирование понятие «наблюдаемость» (по одной внутренней переменной для каждой компоненты программы).

При логическом управлении состояния управляющего автомата могут быть выделены «естественно», если первоначально каждое из них однозначно сопоставить с физическим состоянием управляемого объекта. Например, для исправного клапана, который может находиться в четырех состояниях («Закрыт», «Открывается», «Открыт», «Закрывается»), управляющий им автомат Мура первоначально должен строиться также с четырьмя состояниями, каждое из которых поддерживает объект в требуемом состоянии. При этом переход в автомате, выполняемый под воздействием внешнего события (например, нажатие соответствующей кнопки), обеспечивает аналогичный переход в объекте, выполняемый под воздействием значений выходных переменных, формируемых автоматом в новом состоянии. В дальнейшем при необходимости за счет потери указанного однозначного соответствия может ставиться вопрос о минимизации числа состояний автомата, которая в рамках предлагаемого подхода может уменьшить лишь значность, а не число внутренних переменных. Например, для клапана с памятью вместо автомата с четырьмя состояниями может использоваться автомат с тремя состояниями, в котором состояния, соответствующие состояниям объекта «Закрыт» и «Открыт», объединены.

Отметим также, что в рамках предлагаемого подхода для системы, содержащей N автоматов (при ограничении, когда в каждом программном цикле в каждом автомате выполняется не более одно-

го перехода), N многозначных переменных позволяют не только закодировать состояния автоматов, но и осуществить их взаимодействие за счет вложенности или вызываемости. При этом у вызываемого ГП на соответствующей дуге выполняется проверка номера состояния, в котором должен находиться вызывающий граф, инициирующий переход по указанной дуге.

Из изложенного следует, что в рамках предлагаемого подхода при необходимости применения бинарных графов их построение должно начинаться с дешифратора состояний, после каждого выхода которого должен размещаться дешифратор входных событий. Построенные таким образом бинарные графы, несмотря на громоздкость, весьма просто понимаются, так как они изоморфны графам переходов, что резко упрощает проверку их функциональной эквивалентности. В то время как для бинарных графов, построенных иначе, проверка эквивалентности с графами переходов (верификация), как было показано в разд. 17.4, весьма трудоемка.

Обратим внимание также на то, что при использовании ЭВМ в контуре управления на дисплее отражаются состояния процесса, а события формируются, например, с помощью клавиатуры. Поэтому для обеспечения хорошей понимаемости управляющих программ принцип их построения должен совпадать с принципом управления, в соответствии с которым Оператор сначала определяет состояние процесса, а затем формирует событие, обеспечивающее изменение этого состояния. Странной является ситуация, в которой управление производится по одному принципу (начиная с анализа состояния), а написание программы, обеспечивающей управление, выполняется на основе прямо противоположного принципа (начиная с анализа события). Это иллюстрируется примером, приведенным в «Приложении 11» в [3].

В заключение раздела отметим, что недостатком рассмотренных в нем принудительного и принудительно-свободного видов кодирования является сложность дешифратора состояний, так как они основаны на применении двоичных переменных.

Принудительно-свободное кодирование обладает и еще одним недостатком: дополнительно вводимые внутренние переменные изображаются в графе переходов на одном уровне с выходными переменными, что отвлекает внимание Заказчика, которого внутренние переменные обычно не интересуют.

17.5.2. Автоматы Мура

Применение модели автомата Мура [3] позволяет внутренние и выходные переменные сделать «разноуровневыми».

Двоичное логарифмическое кодирование. Предположим, что автомат имеет S состояний. Присвоим каждому из них код дли-

На рис. 17.36 приведен граф переходов счетного триггера, в котором использован один из двоично-логарифмических кодов. При этом отметим, что при программной реализации этот код может не быть противогоночным.

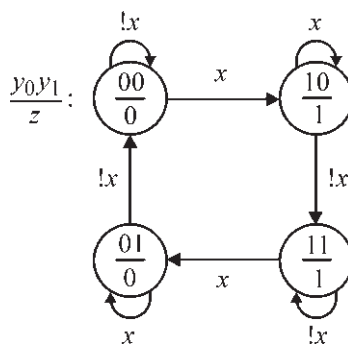


Рис. 17.36

Этому ГП изоморфен СБГ, приведенный на рис. 17.37.

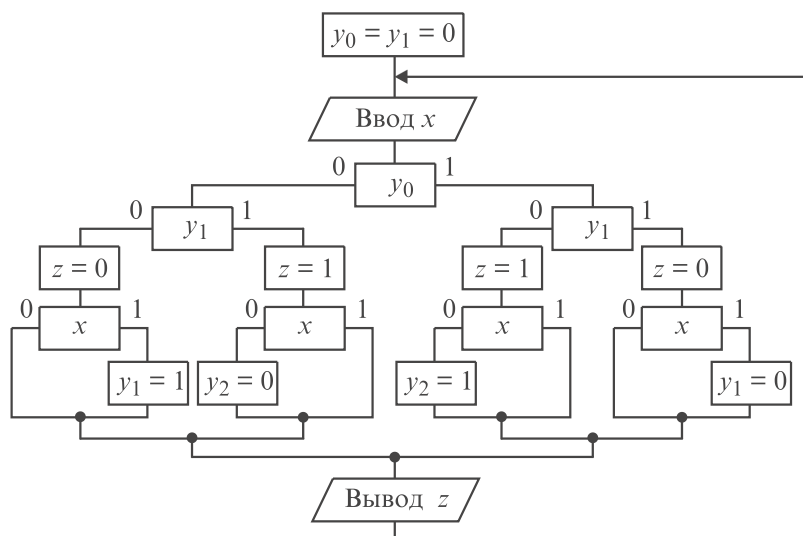


Рис. 17.37

Двоичное кодирование состояний. При таком кодировании каждому состоянию присваивается двоичная переменная, которая принимает единичное значение в этом состоянии и ноль во всех

остальных. На рис. 17.38 приведен граф переходов счетного триггера, в котором применен этот вид кодирования.

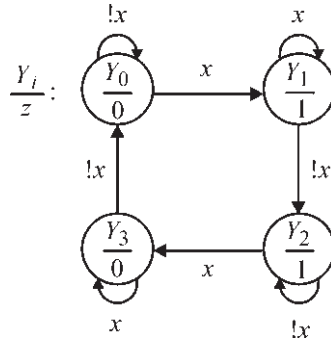


Рис. 17.38

Двоичное кодирование порождает наиболее простые булевы формулы, описывающие поведение автомата, каждая из которых всегда является бесповторной по переменным, кодирующим состояния:

$$\begin{aligned}
 Y_{01} &= !x(Y_3 \vee Y_0); & Y_{11} &= x(Y_0 \vee Y_1); \\
 Y_{21} &= !x(Y_1 \vee Y_2); & Y_3 &= x(Y_2 \vee Y_3); \\
 z &= Y_1 \vee Y_2; & Y_0 &= Y_{01}; & Y_1 &= Y_{11}; & Y_2 &= Y_{21}.
 \end{aligned}$$

Построенному графу переходов изоморфен СБГ на рис. 17.39.

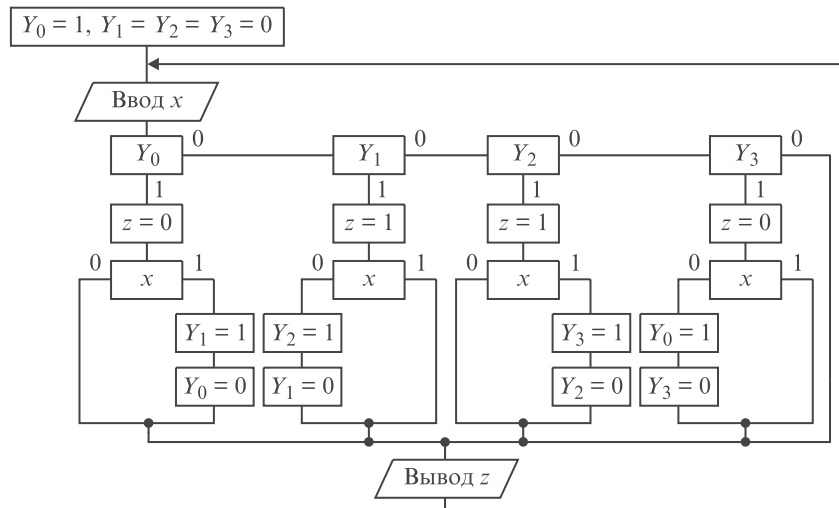


Рис. 17.39

Недостаток этого вида кодирования состоит в необходимости использования S двоичных переменных, каждая из которых требует принудительного сброса.

Многозначное кодирование состояний. При таком кодировании каждому состоянию присваивается одно из значений одной S -значной переменной, которую нет необходимости принудительно сбрасывать.

На рис. 17.29 приведен граф переходов счетного триггера, в котором применен этот вид кодирования. Построенному графу переходов изоморфен СБГ на рис. 17.40.

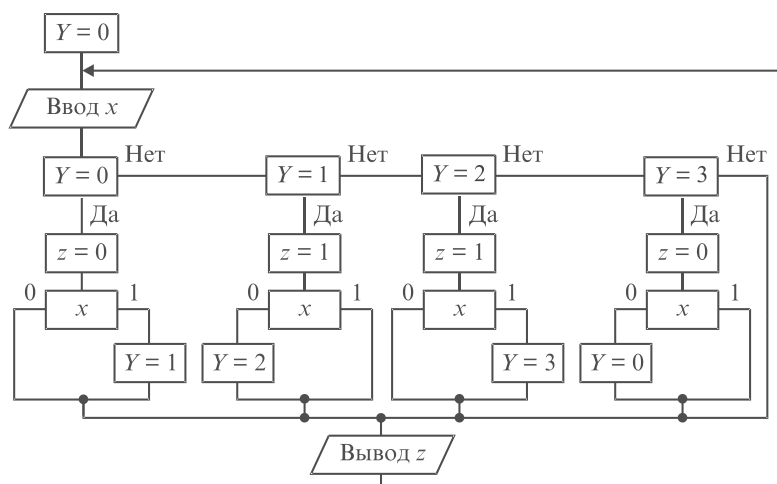


Рис. 17.40

Тело этого СБГ может быть изоморфно реализовано конструкцией switch языка СИ:

```
switch(Y) {
case0: z=0;
      if(x) Y=1;
      break;
case1: z=1;
      if(!x) Y=2;
      break;
case2: /* z=1; */
      if(x) Y=3;
      break;
case3: z=0;
      if(!x) Y=0;
      break; }
```

В этой программе неизменяющееся при переходе значение выходной переменной в состоянии 2 сохранено, но закомментировано,

что позволяет, не снижая понятности программы, сократить ее объектный код.

Отметим, что в этой программе в каждой метке `case` происходит присвоение нового значения переменной Y , входящей собственно в конструкцию `switch`, что характерно для автоматов с памятью. При этом программа в целом изоморфна графу переходов, что обеспечивает ее хорошую «понимаемость».

В событийно-управляемых программах по значению переменной, соответствующей событию, в общем случае осуществляется вызов процедуры [8], которая вовсе не обязательно должна изменять значения этой переменной (пример 2 Приложения). Однако и в этом случае программы (ввиду наличия внешней обратной связи) могут реализовать автоматы с памятью, поведение которых по тексту или СБГ понять весьма трудно из-за отсутствия изоморфизма с эквивалентным ГП.

Ниже приводится событийно-управляемая программа, реализующая счетный триггер и построенная по СБГ (рис. 17.27):

```
switch(x) {
case0: if(z) y=1;
      else y=0;
      break;
case1: if(y) z=0;
      else z=1;
      break; }
```

Сравнение этой программы с предыдущей показывает, что если во второй программе переключатель по переменной x является комбинационным (не изменяет значения этой переменной), то в первой из них переключатель по переменной Y является последовательностным (изменяет значения этой переменной). По этой причине понять по второй программе, что она реализует автомат с памятью и какой граф переходов ей эквивалентен, существенно труднее, чем по первой программе.

Таким образом, применение конструкции `switch` позволяет, структурируя программу, упростить ее понимание только при предлагаемом подходе к ее построению, что не было отмечено в теории структурного программирования [10, 11]. При этом отметим, что метод Ашкрофта и Манни в этом смысле весьма близок к предлагаемому, но обладает рядом недостатков, отмеченных в [3].

Изложенное, видимо, явилось одной из причин наблюдающегося в настоящее время массового перехода от процедурного к объектно-ориентированному подходу к проектированию сложных программ, при котором поведение классов (объектов) предлагается описывать не с помощью бинарных графов, характерных для процедурного подхода, а с помощью графов переходов, называемых в [9] диаграммами состояний и переходов. Предлагаемый в настоящем разделе метод построения би-

нарных графов может при необходимости сохранить их применение и при объектно-ориентированном подходе.

Структурированный бинарный граф (рис. 17.40) и изоморфная ему программа обладают тем недостатком, что в конце программного цикла выходная переменная z всегда сохраняет старое значение, а переменная Y , соответствующая состояниям, может принять новое значение. Этого недостатка лишен СБГ, которому изоморфна следующая программа:

```
switch(Y) {
case0: if(x)    Y=1;
       break;
case1: if(!x)   Y=2;
       break;
case2: if(x)    Y=3;
       break;
case3: if(!x)   Y=0;
       break; }
switch(Y) {
case0:    z=0;    break;
case1:    z=1;    break;
case2: /* z=1 */; break;
case3:    z=0;    break; }
```

Программе, построенной по СГБ на рис. 17.40, соответствует структурная модель автомата Мура первого рода, а последней программе — структурная модель автомата Мура второго рода [3, 12].

17.5.3. Автоматы Мили

Для автоматов этого класса [3] могут использоваться те же виды кодирования, что и для автоматов Мура. На рис. 17.41 приведен бинарный граф, изоморфный графу переходов на рис. 17.26.

Тело этого бинарного графа реализуется следующей программой:

```
switch(P) {
case0: if(!x1& !x2) S=0;
       if( x1⊕ x2) S=1;
       if( x1& x2) {S=0; P=1; } break;
case1: if(!x1& !x2) {S=1; P=0; }
       if( x1⊕ x2) S=0;
       if( x1& x2) S=1; break; }
```

В рассмотренном бинарном графе (за счет уменьшения изоморфизма с графом переходов) после перестановки последовательно

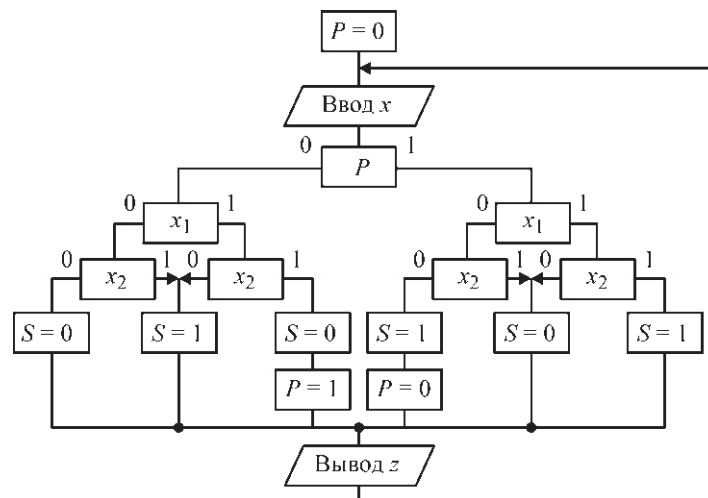


Рис. 17.41

соединенных операторных вершин с пометками S и P число таких вершин может быть сокращено до четырех.

Аналогично можно показать, как по графам переходов строятся бинарные графы для смешанных автоматов и управляющих автоматов, содержащих функциональные элементы задержки [3].

В заключение разд. 17.5 приведем предложенный автором и Б. П. Кузнецовым (НПО «Аврора») структурированный бинарный граф (рис. 17.42), который обеспечивает реализацию взаимо-

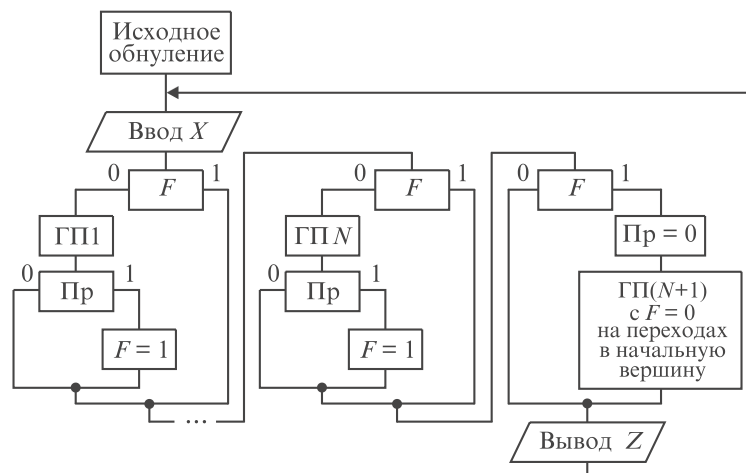


Рис. 17.42

связанной системы из N графов переходов и переход к выполнению $(N + 1)$ -го графа переходов при появлении сигнала прерывания ($Pr = 1$). В последнем графе на переходах в начальную вершину установленный ($F = 1$) ранее флаг должен сбрасываться ($F = 0$).

17.6. Программная реализация графов переходов изоморфными «схемами» из функциональных блоков

Одним из языков программирования программируемых логических контроллеров (ПЛК) является язык функциональных блоков [13]. При использовании этого языка программирование ПЛК и промышленных компьютеров [14, 21] сводится к построению функциональной схемы (ФС) в базисе заданных функциональных блоков.

Если по ГП построить систему булевых формул и реализовать ее функциональной схемой, то она не будет изоморфна ГП.

Ниже предлагаются два метода построения функциональных схем непосредственно по графам переходов, позволяющих устранить этот недостаток. Первый метод применим к ГП с принудительным и принудительно-свободным кодированием состояний, а второй — к ГП с многозначным кодированием состояний [23].

17.6.1. Первый метод

Этот метод, предложенный в [3], строит функциональные схемы, которые содержат три составляющие:

- дешифратор состояний;
- формирователь условий переходов;
- триггеры, фиксирующие коды состояний.

Пример 17.10. Построить ФС для программной реализации графа переходов, в котором используются полные коды (рис. 17.30).

Эта схема приведена на рис. 17.43.

Программная реализация построенной схемы выполняется в порядке, указанном цифрами, расположенными у выходов элементов.

17.6.2. Второй метод

В соответствии со стандартом [13], в библиотеке функциональных блоков при программной реализации функциональных схем рекомендуется наряду с другими блоками иметь логические и цифровые мультиплексоры.

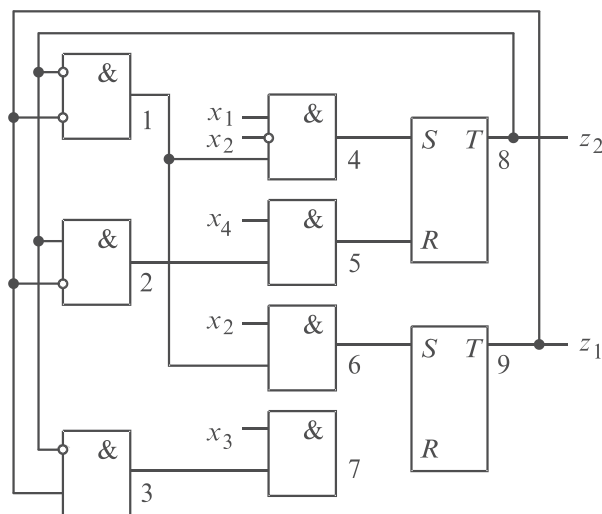


Рис. 17.43

Такие мультиплексоры в отличие от мультиплексоров, реализуемых аппаратно, предназначены для передачи с каждого информационного входа на выход десятичного числа. Выбор информационного входа, как и при аппаратной реализации, выполняется с помощью управляющих входов.

При этом логические мультиплексоры имеют один или несколько управляющих входов, на которые подаются логические переменные, а цифровые мультиплексоры имеют S информационных входов и один управляющий вход, на который подается многозначная переменная, значение которой обеспечивает коммутацию соответствующего информационного входа с выходом.

Если наряду с этими блоками использовать также преобразователь «десятичное число — двоичное число» (Dec — bin), обозначаемый символом «D/b», то появляется возможность непосредственной реализации графов переходов автоматов Мура с многозначным кодированием состояний с помощью изоморфной функциональной схемы, состоящей только из указанных типов блоков [15].

Схемы, которые строятся предлагаемым методом, не содержат триггеров и, по мнению автора, являются при наличии реализуемых ГП наиболее понятными по сравнению с другими разновидностями функциональных схем, реализующих автоматы с памятью.

Пример 17.11. Построить ФС, программно реализующую граф переходов на рис. 17.44.

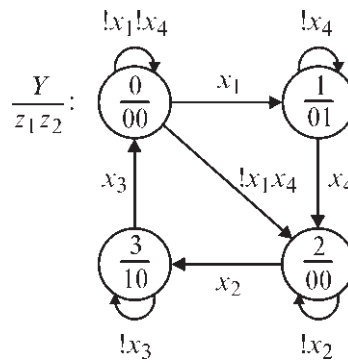


Рис. 17.44

Схема, изоморфная этому графу переходов, приведена на рис. 17.45.

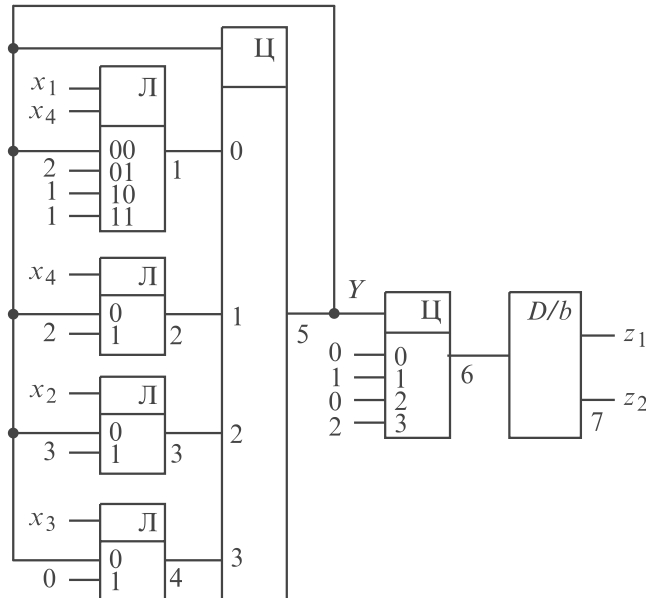


Рис. 17.45

Если число информационных входов логических мультиплексов ограничено, то в построенной схеме сложный логический мультиплексор может быть заменен схемой из более простых мультиплексов. На рис. 17.46 приведена схема, в которой может быть заменен элемент с номером 1 в схеме на рис. 17.45.

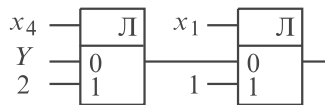


Рис. 17.46

В общем случае функциональные схемы рассматриваемого типа могут быть упрощены (при реализации будут занимать меньший объем памяти), если при их построении наряду с рассмотренными типами блоков применять также логические элементы, реализующие булевы формулы, помечающие дуги, исходящие из вершин ГП. Это целесообразно при выполнении соотношения

$$2^{n_i} > 2^{\lceil \log m_i \rceil},$$

где n_i — суммарное число переменных, помечающих все m_i дуг, исходящих из i -й вершины графа переходов.

Для рассмотренного примера применение логических элементов нецелесообразно, так как в этом случае для всех вершин $n_i = \lceil \log m_i \rceil$.

Если в ГП на рис. 17.44 пометку дуги между вершинами 0 и 2 заменить на $!x_1x_4x_5$, то вместо логического мультиплексора «8 в 1», соответствующего нулевой вершине графа переходов, может быть использована одна из двух схем на рис. 17.47.

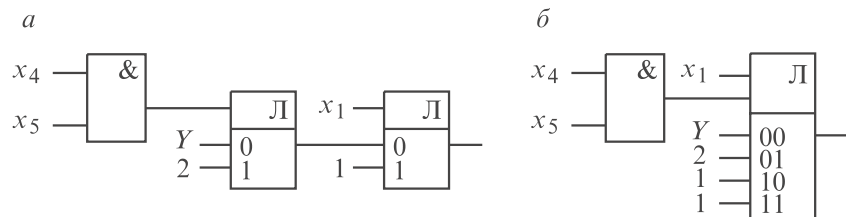


Рис. 17.47

Более подробно (включая реализацию систем графов переходов) рассмотренный метод изложен в [3].

Этот метод был применен в НПО «Аврора» при создании системы логического управления дизель-генератором ДГР-2А 500*500 судна проекта 15640 на базе аппаратуры «Selma-2» фирмы «ABB Stromberg» [16, 17].

Выводы

1. Предложен метод независимых фрагментов, позволяющий строить операторы структуры структурированных бинарных графов (СБГ) для автоматов с памятью. Этот оператор описывает структу-

ру СБГ так же, как булева формула в базисе $\{\&, \vee, !\}$ описывает структуру параллельно-последовательной контактной схемы, но в отличие от булевой формулы, описывающей и функционирование схемы, оператор не описывает функционирование СБГ.

2. Рассмотрен вопрос об использовании канонического и предложенного автором распределительного методов, а также модификации канонического метода для построения бинарных графов для автоматов с памятью.

3. Предложен новый критерий качества бинарных графов («понятность») и введен формальный показатель для его применения.

4. Рассмотрен вопрос о верификации бинарных графов, реализующих автоматы с памятью.

5. Для различных видов кодирования состояний и типов автоматов предложены стандартные структуры СБГ, которые изоморфны графам переходов и конструкции *switch* языка СИ. Такие СБГ в явном виде содержат связи не только по управлению, но и по данным. Они начинаются с дешифратора состояний, а не событий, как это принято в событийном программировании, что делает предлагаемые СБГ понятными. Такой подход связывает структурное программирование с объектно-ориентированным.

6. Предложены методы программной реализации графов переходов изоморфными «схемами» из функциональных блоков, в которые при необходимости легко могут быть внесены изменения.

7. Методы реализации автоматов с памятью в базисах булевых формул, языков инструкций и ассемблеров, лестничных схем, а также алгоритмических языков высокого уровня подробно рассмотрены в монографии автора [3]. В ней также рассмотрены новые подходы к реализации диаграмм «Графсет», используемых при реализации алгоритмов логического управления, содержащих параллельно протекающие процессы. Показаны преимущества систем взаимосвязанных графов переходов с многозначным кодированием вершин в каждом из них по сравнению с этими диаграммами при реализации таких алгоритмов.

8. Вопрос о противогоночном кодировании состояний асинхронных автоматов, которые реализуют простые бинарные графы, вычисляющие системы булевых функций, рассмотрен в [18]. При этом показано, что существуют такие системы функций, для которых противогоночное кодирование состояний является безызбыточным.

9. Исследование переходных процессов в одноконтурных схемах («четные контуры») выполнено В. В. Киселевым (НПО «Аврора») [19] при участии автора [20]. При этом был предложен метод построения функциональных элементов задержки, длительность запаздывания которых может превышать сумму запаздываний элементов, их составляющих. Этот метод описан в [3, 22].

Л и т е р а т у р а

1. Баранов С. И. Синтез микропрограммных автоматов (граф-схемы и автоматы). Л.: Энергия, 1979.
2. Baranov S. Logic synthesis for control automata. Boston: Kluwer academic publishers, 1994.
3. Шалыто А. А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
4. Вирт Н. Программы = алгоритмы + данные. М.: Мир, 1988.
5. Миллер Г. Магическое число семь плюс или минус два: о некоторых пределах нашей способности перерабатывать информацию // Инженерная психология. М.: Прогресс, 1964.
6. Шалыто А. А. Использование граф-схем алгоритмов и графов переходов при программной реализации алгоритмов логического управления // Автоматика и телемеханика. 1996. № 6, 7.
7. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. ГОСТ 19.701-90 (ИСО 5807-85).
8. Матчо Д., Фолкнер Д. DELPHI. М.: Бином, 1995.
9. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. М.: Бином; СПб.: Невский диалект, 1998.
10. Иодан Э. Структурное проектирование и конструирование программ. М.: Мир, 1979.
11. Лингер Р., Миллс Х., Уитт С. Теория и практика структурного программирования. М.: Мир, 1982.
12. Шалыто А. А., Антипов В. В. Алгоритмизация и программирование задач логического управления техническими средствами. СПб.: Моринтех, 1996.
13. International standart IEC 1131-3. Programmable controllers. Part 3. Programming languages // International Electrotechnical Commision. 1993.
14. ISA GRAF. Computer aided software engineering workbench for open PLCs and industrial computers. Version 2.10. User's guide. France: CJ International, 1994.
15. Кондратьев В. Н., Шалыто А. А. Использование функциональных схем при программной реализации автоматов // Судостроит. пром-сть. Сер. Автоматика и телемеханика. 1991. Вып.13.
16. «Селма-2». Описание функциональных блоков. АББ Стромберг, 1989.
17. Project 15640. AS21. DG1. Control. АМИЕ. 95564. 12М. St. Petersburg. ASS «Аврора», 1991.
18. Сагалович Ю. Л., Шалыто А. А. Бинарные программы и их реализация асинхронными автоматами // Проблемы передачи информации. 1987. Вып. 1.
19. Киселев В. В. Функциональные возможности четных контуров при реализации временных программ // Вопросы судостроения. Сер. Судовая автоматика. 1982. Вып. 27.
20. Киселев В. В., Шалыто А. А. Исследование переключательных процессов в «четных» контурах // Проблемы комплексной автоматизации судовых технических средств. Тез. докл. к VII Всесоюз. научно-техн. конф. Л.: ВНТО им. акад. А.Н. Крылова, 1989.
21. Шакиров С., Биосов Р., Якубович Б., Журавлев В. ULTRALOGIC — система подготовки программ для промышленных контроллеров // Современные технологии автоматизации. 1997. № 3.
22. Киселев В. В., Шалыто А. А. Исследование переходных процессов в одно-контурных логических схемах // Известия РАН. Теория и системы управления. 1999. № 5.
23. Шалыто А. А. Реализация алгоритмов логического управления программой на языке функциональных блоков // Промышленные АСУ и контроллеры. 2000. № 4.

Глава 18

Логические устройства для последовательного вычисления булевых функций

Последовательные устройства (ПУ), предназначенные для логического управления, называемые логическими машинами в работе [1], программными логическими устройствами в [2], программируемыми логическими контроллерами в [3] или программируемыми контроллерами в [4, 5], могут строиться на базе микропроцессоров [6], микроконтроллеров [7] или быть специализированными [8, 9].

Одной из задач, решаемых устройствами этого класса, является последовательное вычисление булевых функций (БФУ).

Несмотря на наличие большого числа работ [10—44], связанных с вычислением БФУ, в литературе отсутствуют работы, посвященные анализу и сравнению различных принципов организации специализированных устройств для таких вычислений.

В настоящей главе сделана попытка хотя бы частично устранить этот пробел.

Принципы обработки информации, используемые в специализированных устройствах, могут применяться при программировании микропроцессоров и микроконтроллеров, что кратко изложено в разд. 18.5.

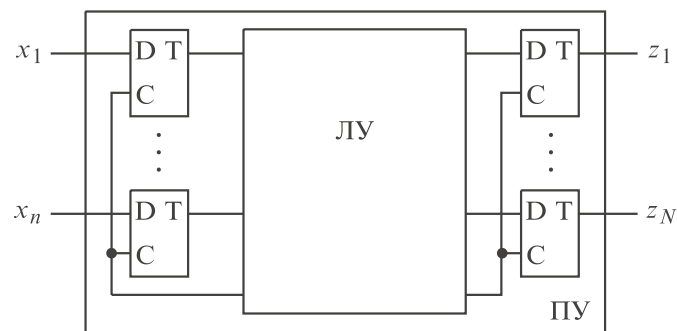


Рис. 18.1

Будем рассматривать специализированные ПУ, которые имеют обобщенную структурную схему, приведенную на рис. 18.1. В этой схеме логическое устройство (ЛУ) может быть построено на основе различных принципов, которые и излагаются в настоящей главе.

Входные *D*-триггеры в этой схеме не позволяют в течение программного цикла изменяться значениям переменных на входах ЛУ, что решает проблему риска для устройств последовательностного действия [45]. При этом ниже будет показано, что существуют последовательностные устройства, в которых входные *D*-триггеры не используются, а проблема риска не возникает.

Выходные *D*-триггеры применяются в этой схеме с целью записи окончательных результатов вычислений в конце программного цикла, что является необходимым при реализации систем булевых функций.

Запись информации в *D*-триггеры, которые в дальнейшем рассматриваться не будут, выполняется по сигналам, формируемым в постоянном запоминающем устройстве (ПЗУ) логического устройства.

В настоящей главе с целью упрощения получения оценок сложности ЛУ основное внимание уделяется вопросу реализации одной булевой функции.

18.1. Операторные устройства

Операторным будем называть устройство (ОУ), последовательно вычисляющее булеву функцию, используя в качестве команд только логические операции.

При этом если в программе применяются команды пересылки, то они реализуются как логические операции повторения.

Будем называть программы, реализуемые такими устройствами, операторными (ОП). Такие программы имеют структуру, соответствующую последовательному выполнению операторов, так как последовательность их выполнения не зависит от результатов промежуточных вычислений (значений промежуточных или внутренних переменных).

Принципиальная особенность устройств этого класса состоит в необходимости использования одноразрядных ячеек (Я) памяти для запоминания промежуточных результатов. Совокупность этих ячеек образует оперативное запоминающее устройство (ОЗУ).

Другая особенность операторных устройств состоит в применении в общем случае процессорного элемента (ПЭ), который выполняет логические операции. Однако, как будет показано ниже, если ячейки памяти используются не только для запоминания, но и для выполнения логических операций, то при подаче на вход логиче-

ского устройства прямых и инверсных переменных процессорный элемент может быть исключен.

При наличии ПЭ в зависимости от числа его информационных входов он может параллельно обрабатывать с помощью одной команды k переменных (входных и внутренних), где $2 \leq k < n$.

18.1.1. $(k + 1)$ -Адресные операторные устройства

Наиболее простым $(k + 1)$ -адресным операторным устройством является устройство, схема которого представлена на рис. 18.2.

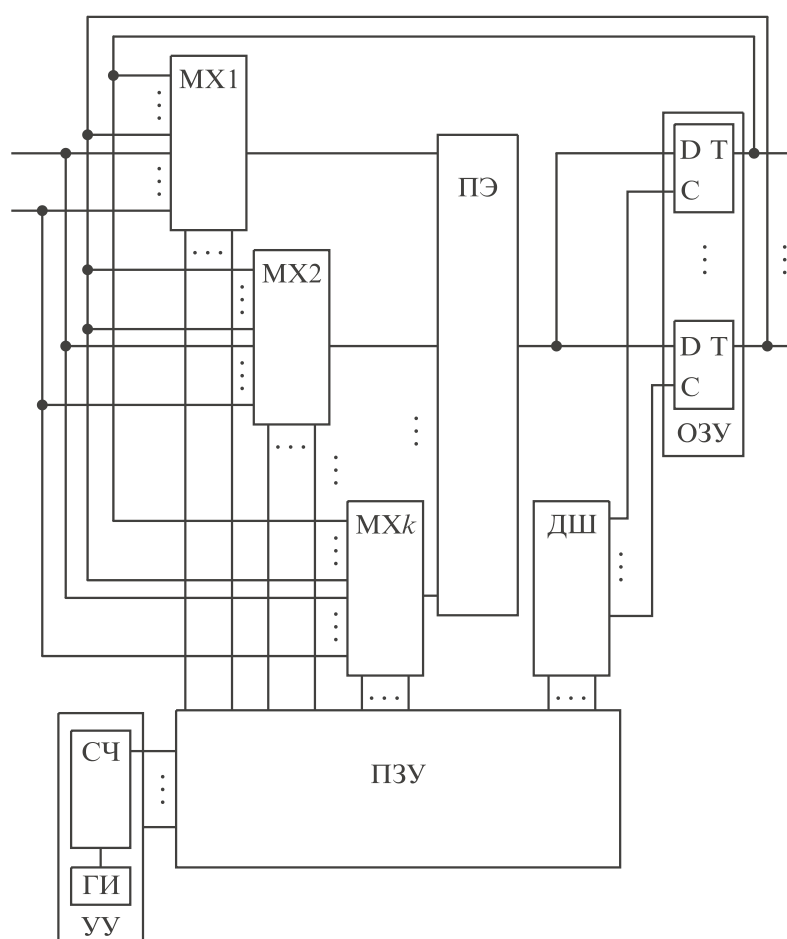


Рис. 18.2

Это устройство названо $(k + 1)$ -адресным, так как в формате его команд указываются k адресов операндов и один адрес результата операции. Будем обозначать символом i число адресов в формате команд операторного устройства.

При этом i -адресное операторное устройство будем обозначать символом ОУ (i), i -адресную программу — символом ОП (i), а число команд — символом K_i .

В рассмотренном устройстве используются:

- k мультиплексоров (МХ), применяемых в режиме коммутаторов каналов;
- процессорный элемент (ПЭ) с k информационными входами;
- D -триггеры для запоминания промежуточных и окончательных результатов;
- дешифратор (ДШ), выбирающий D -триггер, в который записывается результат операции;
- постоянное запоминающее устройство (ПЗУ) команд;
- устройство управления (УУ), состоящее из двоичного счетчика (СЧ) и генератора импульсов (ГИ).

Обратим внимание на то, что в этом устройстве используется дешифратор, который позволяет записывать полученный результат только в одну ячейку ОЗУ, что, однако, не снижает степени универсальности устройства.

В этом устройстве обычно в качестве ПЭ применяется многофункциональный логический модуль (МЛМ), настраиваемый только константами. Этот МЛМ должен быть универсален либо в классе произвольных булевых функций k переменных, либо в классе произвольных булевых формул в базисах $\{\&, \vee, !\}$ или $\{\&, \vee, !, \oplus\}$ из k букв. Естественно, что МЛМ, универсальный в классе произвольных булевых функций k переменных, универсален также и в указанных классах формул. Поэтому обычно в качестве ПЭ используется мультиплексор « 2^k в 1».

Так, например, в [46] в операторном устройстве в качестве процессорного элемента применяется мультиплексор «16 в 1», универсальный в классе произвольных булевых функций четырех и менее переменных.

Программирование операторных устройств наиболее удобно проводить по функциональной схеме, построенной из логических элементов, число входов которых не превышает величины k . При этом число команд в операторной программе совпадает с числом элементов в схеме. Эта схема строится по заданной булевой функции одним из известных или предлагаемых в настоящей работе методов, например мультиплексорным.

Естественно, что операторная программа может строиться также и по схеме, построенной по БФ, или непосредственно по БФ.

Так, например, при применении формульного метода булева формула из h букв, заданная в одном из указанных базисов, реали-

зуется операторной программой из k -универсальных «модулей» с числом команд, определяемым соотношением

$$\left\lceil \frac{h-1}{k-1} \right\rceil \leq K_{k+1} \leq \left\lceil \frac{2(h-1)}{k} \right\rceil.$$

При $k = 2$

$$K_3 = h - 1.$$

Эта идея, в частности, была использована при $k \geq 3$ при написании операторных программ для микропроцессоров [47].

Пример 18.1. При $k = 3$ построить программу типа ОП4, реализующую булеву функцию, которой соответствует формула

$$z = x_1 (x_2 \vee x_3 \vee x_4) \vee x_2 x_3 x_4.$$

При применении формульного метода и так как $h = 7$, то

$$3 \leq K_4 \leq 4.$$

При этом программа имеет следующий вид:

$$x_2 \vee x_3 \vee x_4 \rightarrow Y_1; x_2 x_3 x_4 \rightarrow Y_2; x_1 Y_1 \vee Y_2 \rightarrow Y_1.$$

С целью упрощения программы используем мультиплексорный метод и реализуем заданную БФУ формулой вида:

$$z = x_1 \# (x_1 \# x_2 \# x_3) \# x_4,$$

где $\#$ — символ операции мажорирования для функции трех переменных.

При этом программа упрощается и приобретает следующий вид:

$$x_1 \# x_2 \# x_3 \rightarrow Y_1; x_2 \# Y_1 \# x_4 \rightarrow Y_1.$$

Пример 18.2. При $k = 3$ построить программу типа ОП4, реализующую параллельный двоичный одноразрядный сумматор.

Одно из описаний сумматора имеет вид:

$$P = x_1 \# x_2 \# x_3; S = !P(x_1 \vee x_2 \vee x_3) \vee x_1 x_2 x_3.$$

Из предыдущего примера следует, что

$$P = x_1 \# x_2 \# x_3; S = !P \# (!P \# x_1 \# x_2) \# x_3.$$

Эта система реализуется следующей программой:

$$\begin{aligned} 1. x_1 \# x_2 \# x_3 \rightarrow Y_1; \quad 3. Y_2 \# x_1 \# x_2 \rightarrow Y_3; \\ 2. !Y_1 \rightarrow Y_2; \quad 4. Y_2 \# Y_3 \# x_3 \rightarrow Y_2. \end{aligned}$$

Сумматор может быть реализован также программой из двух команд, если для ее написания применить следующую систему булевых формул:

$$P = x_1 \# x_2 \# x_3; \quad S = x_1 \oplus x_2 \oplus x_3.$$

Из изложенного следует, что для рассматриваемого класса устройств число команд и время вычисления БФУ зависят от значения k . От этого значения зависит и длина команд этих устройств:

$$L_{k+1} = k \lceil \log(n + Y) \rceil + \lceil \log Y \rceil + 2^k,$$

где Y — требуемое число ячеек ОЗУ.

С целью упрощения устройства и ограничения длины команд на практике в большинстве случаев используются операторные устройства с $k = 2$. Перейдем к рассмотрению таких устройств.

18.1.2. Трехадресное операторное устройство

Пусть заданная булева функция n переменных представлена булевой формулой из h букв в одном из рассматриваемых базисов. Эта формула может быть реализована формульным методом схемой из $h - 1$ двухходовых элементов.

Определим оценки сложности устройства ОУЗ, моделирующего эту схему. Если при написании программы число ячеек ОЗУ минимизируется, то

$$2 \leq Y_3 \leq \lceil \log h \rceil.$$

Выберем для этого устройства

$$Y_3 = \lceil \log h \rceil,$$

при этом число обрабатываемых переменных

$$P_3 = n + Y_3 = n + \lceil \log h \rceil.$$

Число команд в устройстве

$$K_3 = h - 1.$$

Эти команды имеют следующий формат: адрес первого операнда, адрес второго операнда, код логической операции, адрес результата.

Применяя в качестве процессорного элемента мультиплексор «4 в 1» в режиме модуля, универсального в классе произвольных БФУ двух переменных, получим соотношение для определения длины команд рассматриваемого устройства:

$$L_3 = 2] \log (n + [\log h])[+] \log [\log h][+4.$$

Будем использовать в качестве характеристики, по которой сравниваются логические устройства, такой показатель, как площадь ПЗУ команд.

Для рассматриваемого устройства

$$S_3 = K_3 L_3.$$

При $h = n \rightarrow \infty$

$$K_3 \rightarrow h; \quad L_3 \rightarrow 2 \log h; \quad S_3 \rightarrow 2h \log h.$$

В [48] показано, что для булевой функции n переменных может быть построена нормальная булева формула в базисе $\{\&, \vee, !\}$, число букв h в которой при $n \rightarrow \infty$ определяется соотношением

$$h \rightarrow \frac{2^n}{\log n}.$$

При этом

$$K_3 \rightarrow \frac{2^n}{\log n}; \quad L_3 \rightarrow 3 \log n; \quad S_3 \rightarrow 3 \cdot 2^n.$$

Структурная схема устройства ОУЗ получается из схемы, приведенной на рис. 18.2, при $k = 2$.

Пример 18.3. Реализовать программой типа ОПЗ булеву формулу

$$z = (x_1 \vee x_2) (x_3 \vee x_4) \vee (x_5 \vee x_6) (x_7 \vee x_8).$$

В этом случае $K_3 = 7$:

1. $x_1 \vee x_2 \rightarrow Y_1$;	5. $x_7 \vee x_8 \rightarrow Y_3$;
2. $x_3 \vee x_4 \rightarrow Y_2$;	6. $Y_2 \& Y_3 \rightarrow Y_2$;
3. $Y_1 \& Y_2 \rightarrow Y_1$;	7. $Y_1 \vee Y_2 \rightarrow Y_1$.
4. $x_5 \vee x_6 \rightarrow Y_2$;	

18.1.3. 2.5-Адресное операторное устройство

Устройство ОУ2.5 получается из устройства ОУ3 за счет исключения связей входных или внутренних переменных со входами второго мультиплексора.

Так как в общем случае число входных переменных больше, то исключим связи этого мультиплексора с такими переменными (рис. 18.3).

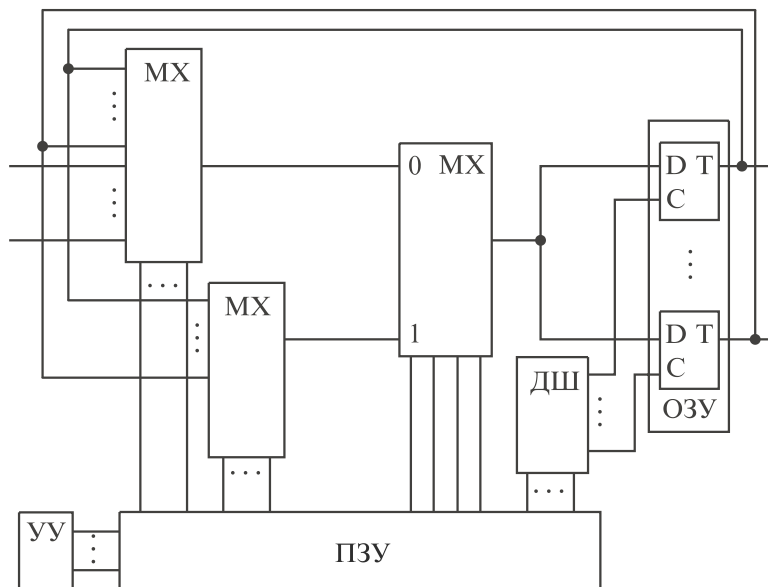


Рис. 18.3

При этом длина команд

$$L_{2.5} = 2] \log (n + [\log h]) [+2] \log [\log h] [+4$$

уменьшается за счет увеличения их числа

$$h \leq K_{2.5} \leq h + [h/2] - 1 = [(3h/2) - 1].$$

Составляющими этой верхней оценки являются: $[h/2]$ — число команд ввода входных переменных; $h - 1$ — число команд логической обработки.

Таким образом, в отличие от устройства ОУ3 уже в устройстве ОУ2.5 появляются команды ввода, «пустые» от логической обработки.

При $h = n \rightarrow \infty$

$$h \lesssim K_{2.5} \lesssim 3h/2; L_{2.5} \rightarrow \log h; h \log h \lesssim S_{2.5} \lesssim (3h/2) \log h.$$

При $n \rightarrow \infty$

$$\frac{2^n}{\log n} \lesssim K_{2.5} \lesssim 3 \frac{2^{n-1}}{\log n}; \quad L_{2.5} \rightarrow 3 \log n; \quad 3 \cdot 2^n \lesssim S_{2.5} \lesssim 4.5 \cdot 2^n.$$

Пример 18.4. Реализовать программой типа ОП 2.5 булеву формулу, рассмотренную в предыдущем примере.

В этом случае $8 \leq K_{2.5} \leq 11$:

1. $x_1 \rightarrow Y_1$;	5. $Y_1 \& Y_2 \rightarrow Y_1$;	9. $x_8 \vee Y_3 \rightarrow Y_3$;
2. $x_2 \vee Y_1 \rightarrow Y_1$;	6. $x_5 \rightarrow Y_2$;	10. $Y_2 \vee Y_3 \rightarrow Y_2$;
3. $x_3 \rightarrow Y_2$;	7. $x_6 \vee Y_2 \rightarrow Y_2$;	11. $Y_1 \vee Y_2 \rightarrow Y_1$.
4. $x_4 \vee Y_2 \rightarrow Y_2$;	8. $x_7 \rightarrow Y_3$;	

18.1.4. Двухадресное операторное устройство

В схеме на рис. 18.3 исключим второй входной мультиплексор, а освободившийся вход процессорного мультиплексора соединим с выходом одного из D -триггеров, который выделим из ОЗУ и назовем аккумулятором (АК). При этом получается устройство ОУ2 (рис. 18.4).

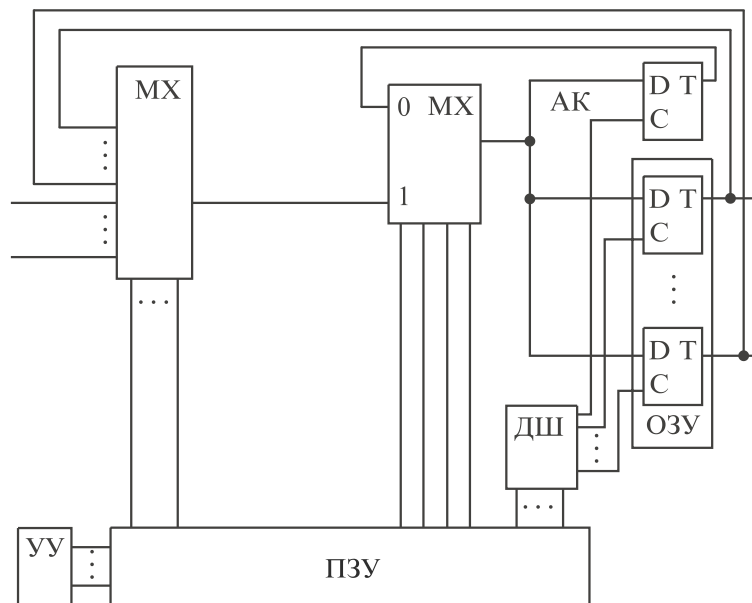


Рис. 18.4

В этом устройстве число команд по сравнению с устройством ОУ2.5 не изменяется:

$$h \leq K_2 \leq h + [h/2] - 1 = [(3h/2) - 1].$$

Длина команд в этом случае определяется соотношением

$$L_2 =]\log(n + [\log(h-1)])[+]\log[\log h][+ 4.$$

При $h = n \rightarrow \infty$

$$h \leq K_2 \leq 3h/2; L_2 \rightarrow \log h; h \log h \leq S_2 \leq (3h/2) \log h.$$

При $n \rightarrow \infty$

$$\frac{2^n}{\log n} \leq K_2 \leq \frac{3}{2} \frac{2^n}{\log n}; L_2 \rightarrow 2 \log n; 2^{n+1} \leq S_2 \leq 3 \cdot 2^n.$$

Пример 18.5. Реализовать программой типа ОП2 булеву формулу, рассмотренную в примере 18.3.

В этом случае $8 \leq K_2 \leq 11$:

1. $x_1 \rightarrow A$;	5. $Y_1 \& A \rightarrow Y_1$;	9. $x_8 \vee A \rightarrow A$;
2. $x_2 \vee A \rightarrow Y_1$;	6. $x_5 \rightarrow A$;	10. $Y_2 \& A \rightarrow A$;
3. $x_3 \rightarrow A$;	7. $x_6 \vee A \rightarrow Y_2$;	11. $Y_1 \vee A \rightarrow Y_1$.
4. $x_4 \vee A \rightarrow A$;	8. $x_7 \rightarrow A$;	

18.1.5. Одноадресное операторное устройство

Совместим в устройстве ОУ2 поля адресов. Введем в ПЗУ дополнительный разряд, обеспечивающий различие аккумулятора и ОЗУ, а также двухвходовые элементы И, соединенные со входами дешифратора. В результате получается устройство ОУ1 (рис. 18.5).

Число команд в этом случае определяется соотношением

$$h + 1 \leq K_1 \leq 2h - 1.$$

Составляющими верхней оценки являются: h — число команд пересылки; $h - 1$ — число команд логической обработки.

Длина команд в устройстве этого типа определяется соотношением

$$L_1 =]\log(n + [\log(h-1)])[+ 5.$$

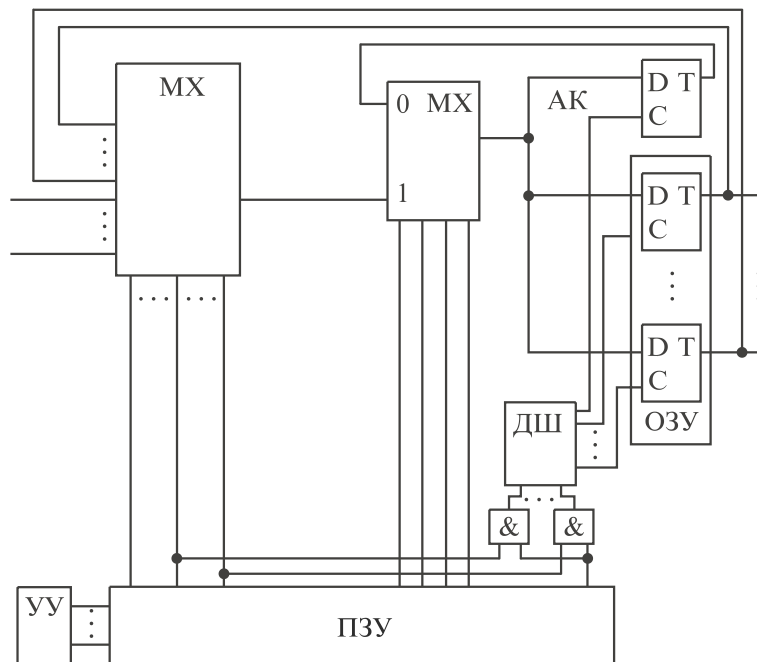


Рис. 18.5

При $h = n \rightarrow \infty$

$$h \leq K_1 \leq 2h; \quad L_1 \rightarrow \log h; \quad h \log h \leq S_1 \leq 2h \log h.$$

При $n \rightarrow \infty$

$$\frac{2^n}{\log n} \leq K_1 \leq \frac{2^{n+1}}{\log n}; \quad L_1 \rightarrow \log n; \quad 2^n \leq S_1 \leq 2^{n+1}.$$

Пример 18.6. Реализовать программой типа ОП1 булеву формулу, рассмотренную в примере 18.3.

В этом случае $9 \leq K_1 \leq 15$:

1. $x_1 \rightarrow A$;	6. $Y_1 \& A \rightarrow A$;	11. $x_7 \rightarrow A$;
2. $x_2 \vee A \rightarrow A$;	7. $A \rightarrow Y_1$;	12. $x_8 \vee A \rightarrow A$;
3. $A \rightarrow Y_1$;	8. $x_5 \rightarrow A$;	13. $Y_2 \& A \rightarrow A$;
4. $x_3 \rightarrow A$;	9. $x_6 \vee A \rightarrow A$;	14. $Y_1 \vee A \rightarrow A$;
5. $x_4 \vee A \rightarrow A$;	10. $A \rightarrow Y_2$;	15. $A \rightarrow Y_1$.

Многие ПЛК могут реализовывать программы типа ОП1. Так, для ПЛК «Autolog» фирмы «FF-Automation OY» [49] рассмотренная

программа может быть реализована на языке инструкций и записана для экономии места в два столбца:

STR	I	x1	OR	I	x6
OR	I	x2	EQ	M	Я2
EQ	M	Я1	STR	I	x7
STR	I	x3	OR	I	x8
OR	I	x4	AND	M	Я2
AND	M	Я1	OR	M	Я1
EQ	M	Я1	EQ	O	z
STR	I	x2	STOP		

18.1.6. Операторные устройства, универсальные в классе формул в базисе И, ИЛИ, НЕ

В рассмотренных выше устройствах при $k = 2$ в качестве процессорного элемента использовался мультиплексор «4 в 1» с четырьмя настроечными входами. Если ограничить класс реализуемых формул базисом $\{\&, \vee, !\}$, то число настроечных входов может быть уменьшено до трех.

Для устройств ОУ3 и ОУ2.5 процессорный элемент приведен на рис. 18.6, а для устройств ОУ2 и ОУ1 — на рис. 18.7.

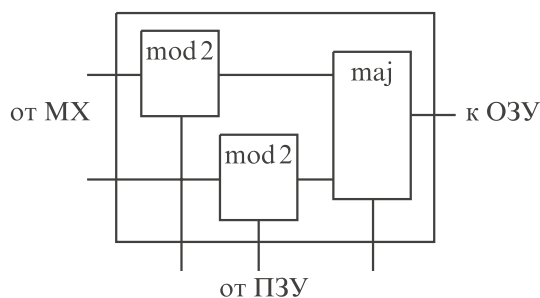


Рис. 18.6

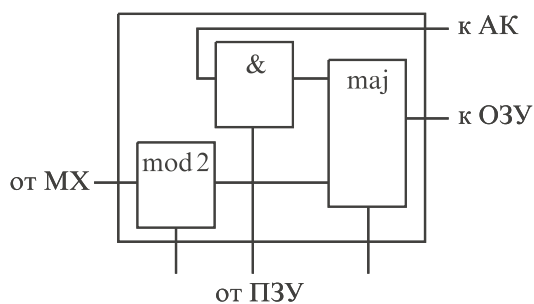


Рис. 18.7

В схеме на рис. 18.7 элемент И установлен для обеспечения возможности применения мажоритарного элемента в режиме повторителя переменной, поступающей с выхода мультиплексора.

При использовании таких процессорных элементов для рассматриваемого класса формул могут применяться оценки сложности устройств, приведенные выше, с учетом того факта, что число настроечных входов процессорного элемента уменьшается на один.

18.1.7. Операторные устройства с микропрограммной реализацией операторов

В рассмотренных устройствах процессорный элемент реализовывался аппаратно. Если его реализовать в виде последовательностной схемы, то получится двухуровневое устройство, верхний уровень которого называется программным, а нижний — микропрограммным [39].

При этом оператор в каждой команде программы интерпретируется микропрограммой.

На каждом из указанных уровней может использоваться свой принцип логической обработки информации, например на верхнем уровне — операторная обработка, а на нижнем — бинарная.

18.1.8. Операторные устройства без кода операций

В качестве процессорного элемента в устройствах ОУ3 и ОУ2.5 можно применять двухвходовой элемент, обладающий свойством функциональной полноты, например элемент И—НЕ. При этом в формате команд исключается поле «код операции». При программировании заданная булева формула реализуется схемой в базисе этих элементов, а затем моделируется рассматриваемым устройством. Отметим, что для моделирования инверторов на соответствующие входы мультиплексоров должна быть подана константа «единица».

Устройство ОУ2.5 этого типа приведено на рис.18.8.

18.1.9. Операторные устройства с RS-триггерами

Если в рассмотренных устройствах *D*-триггеры заменить на триггерные ячейки, каждая из которых состоит из *RS*-триггера и двухвходовых элементов И и ЗАПРЕТ, то при использовании этих

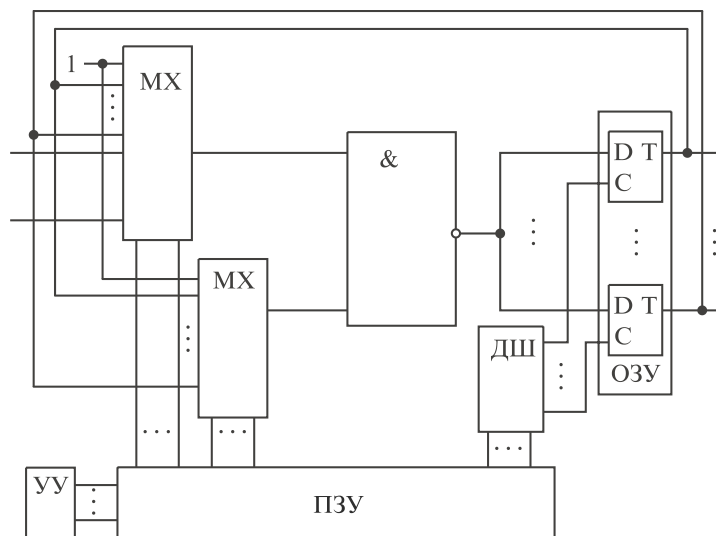


Рис. 18.8

ячеек только для запоминания информации все приведенные выше результаты сохраняются (рис. 18.9).

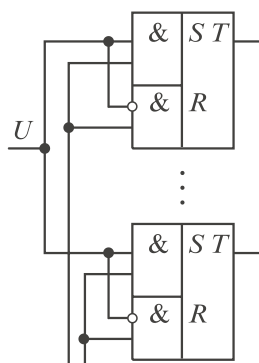


Рис. 18.9

При этом отметим, что в этих ячейках элемент ЗАПРЕТ может быть заменен на элемент И, а для всех ячеек может применяться один инвертор. Однако эти ячейки кроме функции запоминания могут выполнять также и логические операции. Это позволяет сократить число команд в программе при увеличении их длины, так как в этом случае приходится отказаться от дешифратора, ввиду того что в ряде случаев активными могут быть оба входа ячейки, обозначенные символами $\&$ и $\vee$ (рис. 18.10).

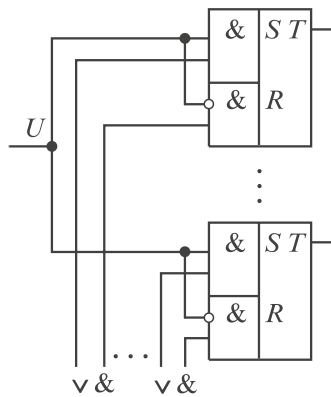


Рис. 18.10

Функционирование одной ячейки в этом случае описывается табл. 18.1.

Т а б л и ц а 18.1

v	&	UQ			
		00	01	10	11
0	0	0	1	0	1
0	1	0	0	0	1
1	0	0	1	1	1
1	1	0	0	1	1

Эта таблица может быть записана и иначе (табл. 18.2).

Т а б л и ц а 18.2

v	&	Q
0	0	Q
0	1	U & Q
1	0	U v Q
1	1	U

Из этой таблицы следует, что описанная триггерная ячейка позволяет реализовать функции & и v.

1. Рассмотрим трехадресное операторное устройство этого типа, которое обозначим символом ОУЗ1.

Пример 18.7. Реализовать БФ из примера 18.3 с помощью рассмотренного устройства.

При $\vee = 1, \& = 1$ $x_1 \vee x_2 \rightarrow Y_1$;
 при $\vee = 0, \& = 1$ $(x_3 \vee x_4) \& Y_1 \rightarrow Y_1$;
 при $\vee = 1, \& = 1$ $x_5 \vee x_6 \rightarrow Y_2$;
 при $\vee = 0, \& = 1$ $(x_7 \vee x_8) \& Y_2 \rightarrow Y_2$;
 при $\vee = 1, \& = 1$ $Y_1 \vee Y_2 \rightarrow z$.

Таким образом, получили наиболее компактную программу из рассмотренных.

Оценим число команд в этом устройстве. Из приведенного примера следует, что в данном случае используются двух- и трехместные команды. При $h = 2k$ существуют булевы формулы, при реализации которых первая команда является двухместной, а остальные — трехместными. Поэтому

$$K_{31\text{ н}} = \frac{(h-1)-1}{3-1} + 1 = \frac{h}{2}.$$

При $h = 2k + 1$ существуют булевы формулы, при реализации которых только первая и последняя команды являются двухместными, а остальные — трехместными. Поэтому

$$K_{31\text{ н}} = \frac{(h-2)-1}{3-1} + 2 = \frac{h+1}{2}.$$

Таким образом,

$$K_{31\text{ н}} = \lceil h/2 \rceil.$$

При любых h существуют булевы формулы, при реализации которых все команды являются двухместными, как например это имеет место для формулы

$$z = (x_1 \vee x_2)x_3 \vee (x_4 \vee x_5)x_6.$$

При этом

$$K_{31\text{ в}} = h - 1.$$

Следовательно,

$$\lceil h/2 \rceil \leq K_{31} \leq h - 1.$$

Оценим число ячеек в ОЗУ.

При $h = 3, \dots, 5$ требуется одна ячейка; при $h = 6, \dots, 11$ — две; при $h = 12, \dots, 23$ — три; при $h = 24, \dots, 47$ — четыре; при $h = 48, \dots, 95$ — пять ячеек и т. д.

При $h_{\text{н}} = 3, 6, 12, 24, 48, \dots = 3 \cdot 2^{Y-1}$; $Y_{\text{н}} = 1 + \log(h_{\text{н}}/3) = \log(2h_{\text{н}}/3)$.

При $h_{\text{н}} = 5, 11, 23, 47, 95, \dots = 2h_{\text{н}} - 1 = 3 \cdot 2^{Y-1} - 1$; $Y_{\text{в}} = \log(h_{\text{в}} + 1)/3$. Следовательно,

$$Y_{\text{мах}} = \lceil \log(2h/3) \rceil = \lceil \log((h+1)/3) \rceil; 3 \cdot 2^{Y-1} \leq h \leq 3 \cdot 2^Y - 1.$$

При этом

$$L_{31} = 2 \lceil \log(n + \lceil \log(2h/3) \rceil) \rceil + 2(\lceil \log(2h/3) \rceil + 1) + 4.$$

При $h = n \rightarrow \infty$

$$h/2 \leq K_{31} \leq h; \quad L_{31} \rightarrow 2(\log h + \log(2h/3));$$

$$h(\log h + \log(2h/3)) \leq S_{31} \leq 2h(\log h + \log(2h/3)).$$

При $n \rightarrow \infty$

$$\frac{2^{n-1}}{\log n} \leq K_{31} \leq \frac{2^n}{\log n}; \quad L_{31} \rightarrow 2n; \quad n \frac{2^n}{\log n} \leq S_{31} \leq n \frac{2^{n+1}}{\log n}.$$

2. Рассмотрим двухадресное операторное устройство этого типа, которое обозначим символом ОУ21.

Так как приведенные выше триггерные ячейки реализуют операции $\&$ и $\vee$, то для реализации инверсий будем использовать двухвходовой элемент НЕРАВНОЗНАЧНОСТЬ, работающий в режиме «повторитель — инвертор» (рис. 18.11).

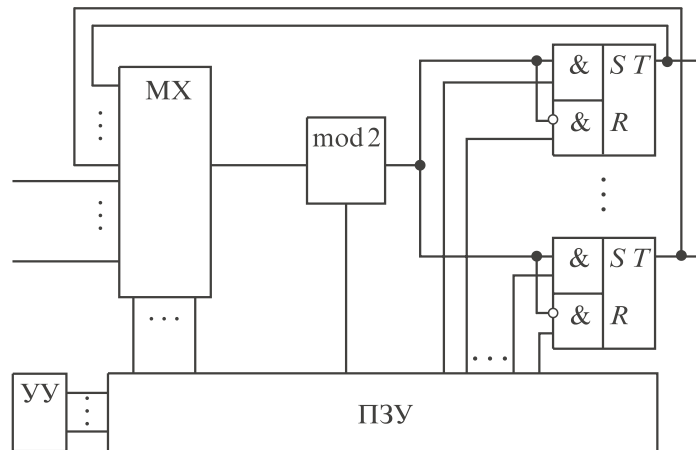


Рис. 18.11

Пример 18.8. Реализовать БФ из примера 18.3 с помощью рассмотренного устройства.

Программа в данном случае имеет следующий вид:

1. $x_1 \rightarrow Y_1;$	5. $Y_1 \& Y_2 \rightarrow Y_1;$	9. $x_8 \vee Y_3 \rightarrow Y_3;$
2. $x_2 \vee Y_1 \rightarrow Y_1;$	6. $x_5 \rightarrow Y_2;$	10. $Y_2 \& Y_3 \rightarrow Y_2;$
3. $x_3 \rightarrow Y_2;$	7. $x_6 \vee Y_2 \rightarrow Y_2;$	11. $Y_1 \vee Y_2 \rightarrow Y_1.$
4. $x_4 \vee Y_2 \rightarrow Y_2;$	8. $x_7 \rightarrow Y_3;$	

Для рассмотренного устройства

$$h + 1 \leq K_{21} \leq h + [h/2] = [3h/2]; \quad Я_{1 \max} = [\log h];$$

$$L_{21} = [\log(n + [\log h])][+2[\log h] + 1].$$

При $h = n \rightarrow \infty$

$$h \leq K_{21} \leq 3h/2; \quad L_{21} \rightarrow 3 \log h; \quad 3h \log h \leq S_{21} \leq 4.5 \log h.$$

При $n \rightarrow \infty$

$$\frac{2^n}{\log n} \leq K_{21} \leq 3 \frac{2^{n-1}}{\log n}; \quad L_{21} \rightarrow 2n; \quad n \frac{2^{n+1}}{\log n} \leq S_{21} \leq 3n \frac{2^n}{\log n}.$$

Из изложенного следует, что если на входы мультиплексора подать прямые и инверсные переменные, то элемент НЕРАВНОЗНАЧНОСТЬ может быть исключен, а получающееся операторное устройство не будет содержать процессорного элемента.

18.1.10. Стековое операторное устройство

Еще одним подклассом операторных устройств являются устройства, в которых триггеры, образующие ОЗУ, заменены стеком. Будем называть такие устройства **стековыми операторными устройствами (СОУ)**.

Одноадресное стековое операторное устройство (СОУ1) приведено на рис. 18.12.

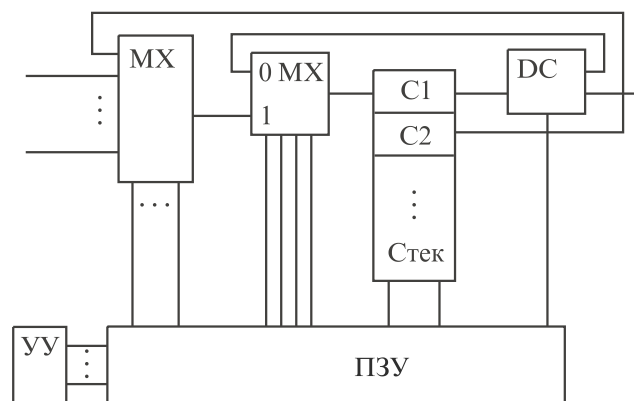


Рис. 18.12

Это устройство может реализовывать стековые операторные программы, написанные либо на основе обратной польской записи, либо непосредственно по формуле или лестничной схеме [50].

1. При реализации на основе обратной польской записи справедливы следующие соотношения:

$$K_{\text{сп}} = 2h; \quad L_{\text{сп}} = \lceil \log(n+1) \rceil + 7; \quad Я_{\text{сп}} = \lceil 1 + \log h \rceil.$$

При $h = n \rightarrow \infty$

$$K_{\text{сп}} = 2h; \quad L_{\text{сп}} \rightarrow \log h; \quad S_{\text{сп}} \rightarrow 2h \log h.$$

При $n \rightarrow \infty$

$$K_{\text{сп}} \rightarrow \frac{2^n}{\log n}; \quad L_{\text{сп}} \rightarrow \log n; \quad S_{\text{сп}} \rightarrow 2^{n+1}.$$

Пример 18.9. Реализовать формулу из примера 18.3 с помощью рассмотренного устройства.

Эта формула в обратной польской записи имеет вид:

$$x_1 x_2 \vee x_3 x_4 \vee \& x_5 x_6 \vee x_7 x_8 \vee \& \vee.$$

Поэтому в данном случае стековая программа может быть записана следующим образом:

1. $x_1 \rightarrow c_1;$	9. $x_6 \rightarrow c_1 c_2;$
2. $x_2 \rightarrow c_1 c_2;$	10. $c_1 \vee c_2 \rightarrow c_1 c_2;$
3. $c_1 \vee c_2 \rightarrow c_1;$	11. $x_7 \rightarrow c_1 c_2 c_3;$
4. $x_3 \rightarrow c_1 c_2;$	12. $x_8 \rightarrow c_1 c_2 c_3 c_4;$
5. $x_4 \rightarrow c_1 c_2 c_3;$	13. $c_1 \vee c_2 \rightarrow c_1 c_2 c_3;$
6. $c_1 \vee c_2 \rightarrow c_1 c_2;$	14. $c_1 \& c_2 \rightarrow c_1 c_2;$
7. $c_1 \& c_2 \rightarrow c_1;$	15. $c_1 \vee c_2 \rightarrow c_1;$
8. $x_5 \rightarrow c_1 c_2;$	16. $c_1 \rightarrow z;$

Используя язык инструкций ПЛК фирмы «Omron» (Япония) [50], эта программа, записанная в два столбца, приобретает следующий вид:

LD	x1	LD	x6
LD	x2	OR	LD
OR	LD	LD	x7
LD	x3	LD	x8
LD	x4	OR	LD
OR	LD	AND	LD
AND	LD	OR	LD
LD	x5	OUT	

2. При реализации непосредственно по формуле (лестничной схеме) справедливы следующие соотношения:

$$h + 1 \leq K_{\text{сф}} \leq [3h/2]; \quad L_{\text{сф}} = \lceil \log(n+1) \rceil + 7; \quad Я_{\text{сф}} = \lceil \log h \rceil.$$

При $h = n \rightarrow \infty$

$$h + 1 \leq K_{\text{сф}} \leq [3h/2]; \quad L_{\text{сф}} \rightarrow \log h; \quad h \log h \leq S_{\text{сф}} \leq 1.5 h \log h.$$

При $n \rightarrow \infty$

$$\frac{2^n}{\log n} \leq K_{\text{сф}} \leq 1.5 \frac{2^n}{\log n}; \quad L_{\text{сф}} \rightarrow \log n; \quad 2^n \leq S_{\text{сф}} \leq 1.5 \cdot 2^n.$$

Именно эти оценки и будем применять в качестве оценок одно-адресного стекового операторного устройства.

Пример 18.10. Реализовать формулу из примера 18.3.

Программа в этом случае имеет следующий вид:

1. $x_1 \rightarrow c_1$;	7. $x_6 \vee c_1 \rightarrow c_1 c_2$;
2. $x_2 \vee c_1 \rightarrow c_1$;	8. $x_7 \rightarrow c_1 c_2 c_3$;
3. $x_3 \rightarrow c_1 c_2$;	9. $x_8 \vee c_1 \rightarrow c_1 c_2 c_3$;
4. $x_4 \vee c_1 \rightarrow c_1 c_2$;	10. $c_1 \& c_2 \rightarrow c_1 c_2$;
5. $c_1 \& c_2 \rightarrow c_1$;	11. $c_1 \vee c_2 \rightarrow c_1$;
6. $x_5 \rightarrow c_1 c_2$;	12. $c_1 \rightarrow z$.

Используя язык инструкций ПЛК фирмы «Omron», эта программа, записанная в два столбца, приобретает следующий вид:

LD	x1	OR	x6
OR	x2	LD	x7
LD	x3	OR	x8
OR	x4	AND	LD
AND	LD	OR	LD
LD	x5	OUT	

18.1.11. Буквенное операторное устройство

Все рассмотренные выше операторные устройства обладали тем недостатком, что они не позволяли с помощью каждой команды обрабатывать одну букву заданной булевой формулы.

Ниже предлагается принципиально новое операторное устройство, которое позволяет побуквенно, справа налево вычислять с помощью h команд без предварительного преобразования произвольную (в том числе скобочную произвольной глубины) нормальную булеву формулу в базисе $\{\&, \vee, !\}$ из h букв. Таким обра-

зом, каждая команда рассматриваемого устройства обрабатывает одну букву заданной формулы.

При этом после выполнения i -й команды вычисленным является фрагмент булевой формулы из i букв, который может не быть подформулой вычисляемой формулы.

Рассматриваемое устройство является наиболее «последовательным», а применяемый принцип обработки логической информации является более «красивым» по сравнению с операторными устройствами, описанными выше. Назовем это устройство буквенным операторным устройством (БОУ).

Предлагаемое устройство программируется совершенно необычным для операторных устройств способом — начиная с построения линейного бинарного графа (ЛБГ), используемого традиционно для написания программ бинарных устройств.

Пример 18.11. Реализовать булеву формулу $z = (x_1 \vee x_2)x_3 \vee x_4$ с помощью предлагаемого подхода.

Построим ЛБГ (рис. 18.13).

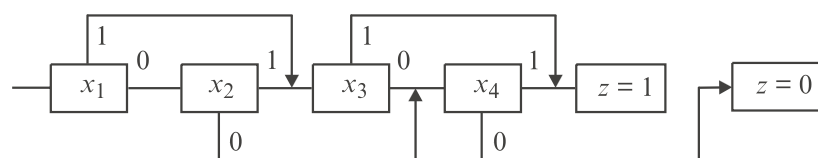


Рис. 18.13

Изменим в ЛБГ пометки так, чтобы все дуги остова были помечены единицами (рис. 18.14).

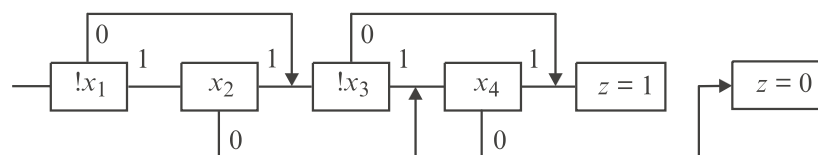


Рис. 18.14

Построим по этой ЛБГ мультиплексорный каскад (рис. 18.15).

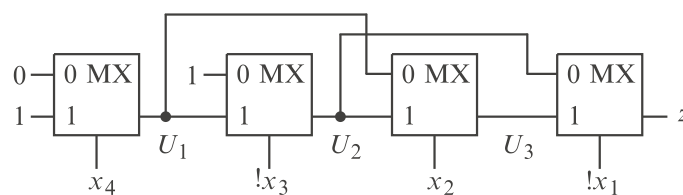


Рис. 18.15

Выполним верификацию построенного каскада:

$$U_1 = 0 \& !x_4 \vee 1 \& x_4 = x_4;$$

$$U_2 = 1 \& (!x_3) \vee U_1!x_3 = x_3 \vee x_4;$$

$$U_3 = U_1!x_2 \vee U_2x_2 = x_2x_3 \vee x_4;$$

$$U_4 = U_2!(x_1) \vee U_3!x_1 = (x_1 \vee x_2)x_3 \vee x_4.$$

Обратим внимание на тот факт, что в заданной булевой формуле подформула U_3 , совпадающая с ее фрагментом, отсутствует.

Для моделирования этого каскада, содержащего две длинные связи, достаточно иметь предлагаемое устройство с тремя D -триггерами для запоминания промежуточных результатов (рис. 18.16).

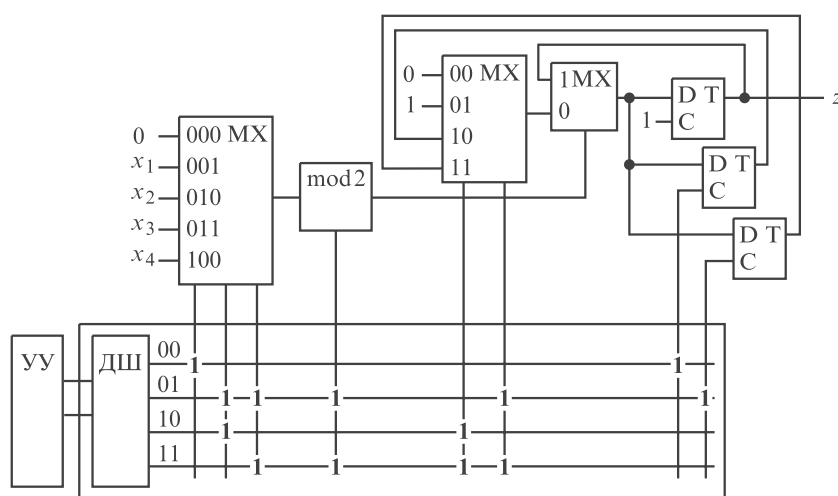


Рис. 18.16

Для обеспечения правильной работы устройства в начальном состоянии в первом D -триггере ОЗУ должна находиться единица, которая записывается в этот триггер последней (служебной) командой программы.

При этом программа может быть записана следующим образом:

$$MX(0, 1; x_4) \rightarrow Y_1 Y_3; \quad /* U_1 = x_4$$

$$MX(1, Y_1; !x_3) \rightarrow Y_1 Y_3; \quad /* U_2 = x_3 \vee x_4$$

$$MX(Y_2, Y_1; x_2) \rightarrow Y_1; \quad /* U_3 = x_2x_3 \vee x_4$$

$$MX(Y_3, Y_1; !x_1) \rightarrow Y_1; \quad /* z = (x_1 \vee x_2)x_3 \vee x_4.$$

Так как в этом случае

$$1 \leq Y_6 \leq h - 1,$$

то устройство должно содержать число D -триггеров в ОЗУ, определяемое выражением

$$Y_6 = h - 1.$$

Длина команд в таком устройстве определяется соотношением

$$L_6 =]\log(n+1)[+]\log(h-1)[+ 3.$$

При $h = n \rightarrow \infty$

$$K_6 = h; \quad L_6 \rightarrow 2 \log h; \quad S_6 \rightarrow 2h \log h.$$

При $n \rightarrow \infty$

$$K_6 \rightarrow \frac{2^n}{\log n}; \quad L_6 \rightarrow 2n; \quad S_6 \rightarrow n \frac{2^{n+1}}{\log n}.$$

Описанное устройство предложено автором в [51], а класс операторных устройств рассмотрен в [52].

18.2. 2^k -нарные устройства

2^k -нарным будем называть устройство, последовательно вычисляющее булеву функцию и использующее при этом только условные переходы по значениям входных переменных с 2^k исходами и присваивания выходным переменным констант.

Принципиальная особенность этого класса устройств состоит в том, что в них нет необходимости применять оперативное запоминающее устройство для запоминания промежуточных результатов и процессорный элемент для выполнения логических операций. При этом отметим, что ОЗУ для запоминания окончательных результатов вычислений и в этом случае является необходимым.

Устройства рассматриваемого класса описаны в [53]. Программирование таких устройств будем проводить по граф-схеме без обратных связей, состоящей только из двух типов вершин: условных с 2^k исходами и операторных вершин, в которых выходной переменной присваиваются константы.

Такие граф-схемы могут быть разбиты на два подкласса [54]:

— простые, в которых операторные вершины располагаются только в конце;

— обобщенные, в которых операторные вершины могут располагаться в любом месте.

Рассмотрим простые граф-схемы с двумя операторными вершинами: $z = 0$ и $z = 1$.

Будем называть 2^k -графами простые граф-схемы с двумя операторными вершинами. Такие графы строятся по БФУ и БФ с помо-

щью разложения Шеннона по k переменным и имеют в общем случае «плоскостную» структуру.

Среди таких графов важное значение имеют бинарные графы и их подкласс, представители которого строятся формульным методом [23, 55] и обладают линейной структурой. Эти графы называются линейными бинарными графами (ЛБГ). Ниже рассматриваются два типа устройств — тетрадные и бинарные.

18.2.1. Тетрадные устройства

Рассмотрим шестиадресное устройство, программируемое по тетрадным графам, которые строятся с помощью разложения Шеннона одновременно по двум входным переменным — x_i и x_j .

Каждая команда в этом устройстве содержит восемь полей:

- 1) адрес переменной x_i ;
- 2) адрес переменной x_j ;
- 3) адрес перехода при $x_i=0, x_j=0$;
- 4) адрес перехода при $x_i=0, x_j=1$;
- 5) адрес перехода при $x_i=1, x_j=0$;
- 6) адрес перехода при $x_i=1, x_j=1$;
- 7) управление (сохранение — запись) D -триггером;
- 8) запись информации в D -триггер.

Отметим, что для удобства третье, четвертое, пятое и шестое поля команды изображаются в виде $\log B$ [полей, каждое из которых образовано четырьмя битами, где B — число вершин в графе. При этом каждое такое поле соответствует одному и тому же разряду всех четырех адресов переходов.

Пример 18.12. Реализовать тетрадный граф (рис. 18.17), построенный по булевой формуле

$$z = x_1(x_2 \vee x_3 \vee x_4) \vee x_2x_3x_4.$$

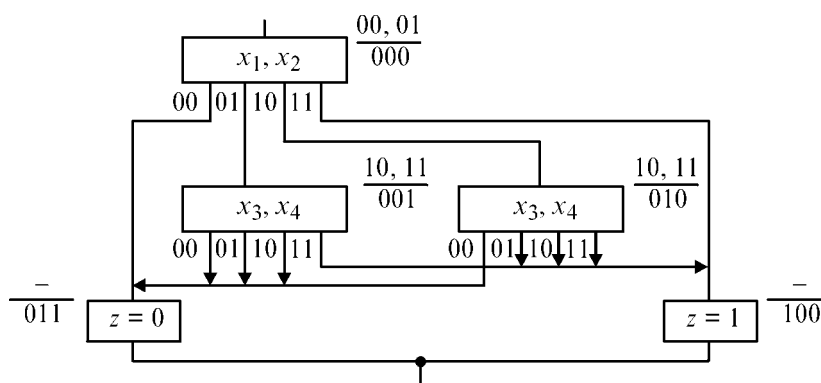


Рис. 18.17

На этом рисунке рядом с каждой вершиной расположена дробь, в числителе которой указаны адреса вводимых в этой вершине входных переменных, а в знаменателе — адрес этой вершины.

Этот тетрадный граф реализуется устройством на рис. 18.18, где РГ — параллельный двоичный регистр.

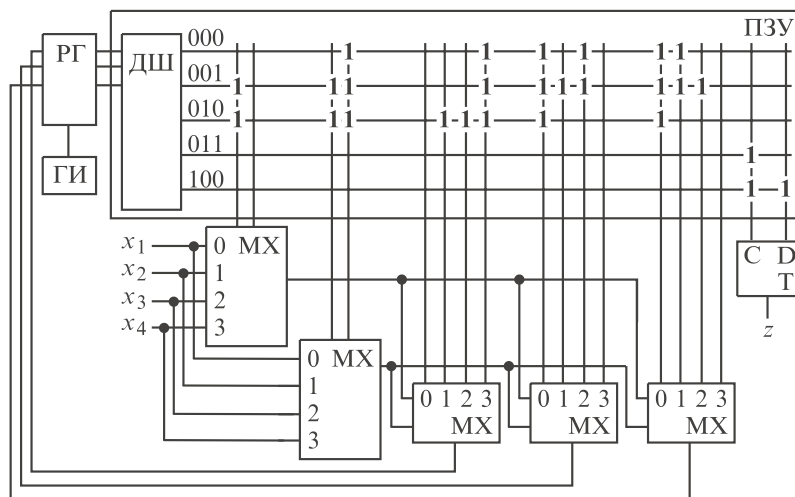


Рис. 18.18

Число разрядов в командах этого устройства определяется соотношением

$$L_T = 2 \lceil \log n \rceil + 4 \lceil \log B \rceil + 2,$$

а число команд в программе, построенной указанным образом, равно B .

18.2.2. Бинарные устройства

Устройства этого класса давно разработаны [2] и используются в течение многих лет при автоматизации, в частности судовых технических средств [2, 8].

Рассмотрим два подкласса устройств этого класса — трехадресные бинарные устройства (БУЗ) и двухадресные бинарные устройства (БУ2).

Первый подкласс устройств программируется на основе бинарных графов, а второй — на основе линейных бинарных графов.

1. Рассмотрим первоначально трехадресные бинарные устройства. В них каждая команда должна содержать пять полей:

- 1) адрес входной переменной x_i ;
- 2) адрес перехода при $x_i=0$;
- 3) адрес перехода при $x_i=1$;
- 4) управление D -триггером;
- 5) запись информации в D -триггер.

При этом отметим, что для удобства изображения второе и третье поля совмещены: четные разряды соответствуют второму адресу, а нечетные — третьему.

Пример 18.13. Реализовать бинарный граф (рис. 18.19), построенный методом каскадов по булевой формуле $z = !x_1 x_2 \vee x_1 !x_3$.

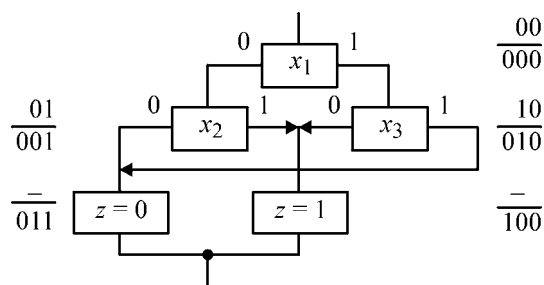


Рис. 18.19

Трехадресное устройство, реализующее этот бинарный граф, приведено на рис. 18.20.

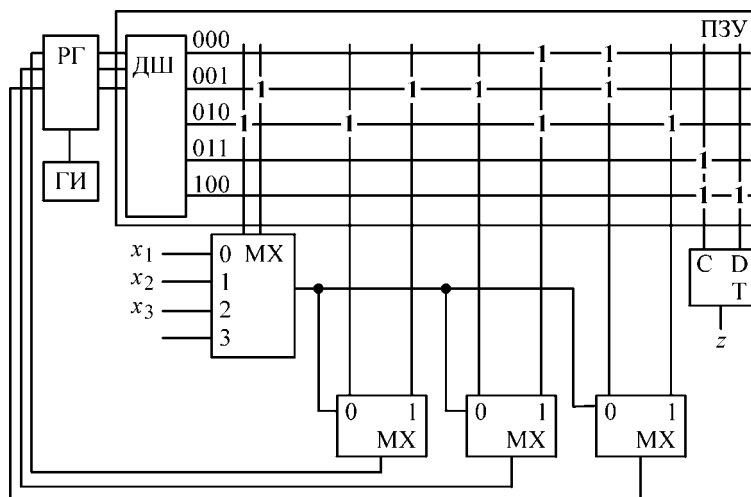


Рис. 18.20

Число разрядов в командах этого устройства определяется соотношением

$$L_{36} = \lceil \log n \rceil + 2 \lceil \log B \rceil + 2,$$

а число команд в программе, построенной указанным образом, равно B .

В [22] показано, что для бинарных графов при $n \rightarrow \infty$ справедливо соотношение

$$B \rightarrow \frac{2^n}{n}.$$

При этом

$$K_{36} \rightarrow \frac{2^n}{n}; \quad L_{36} \rightarrow 2n; \quad S_{36} \rightarrow 2^{n+1}.$$

При применении этого устройства в схеме на рис. 18.1 проблема риска не возникает даже при отсутствии входных D -триггеров, так как в построенном указанным образом бинарном графе на каждом пути от входа к выходу каждая переменная встречается только один раз.

Предложенное устройство является модификацией микропрограммного автомата Уилкса [56, 57].

2. Перейдем к рассмотрению двухадресных бинарных устройств. В них каждая команда также должна содержать пять полей:

- 1) адрес входной переменной x_i ;
- 2) сравнение значения входной переменной x_i с константой;
- 3) адрес «далекого» перехода, осуществляемого в случае, когда имеет место равенство при сравнении;
- 4) управление D -триггером;
- 5) запись информации в D -триггер.

Пример 18.14. Реализовать ЛБГ (рис. 18.21), который построен формульным методом по булевой формуле, рассмотренной в предыдущем примере.

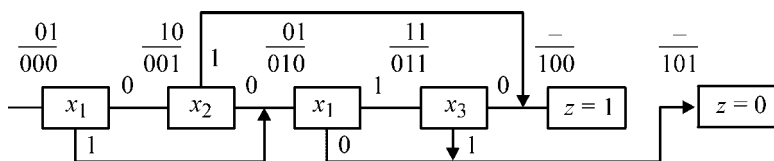


Рис. 18.21

Двухадресное бинарное устройство, реализующее этот ЛБГ, приведено на рис. 18.22.

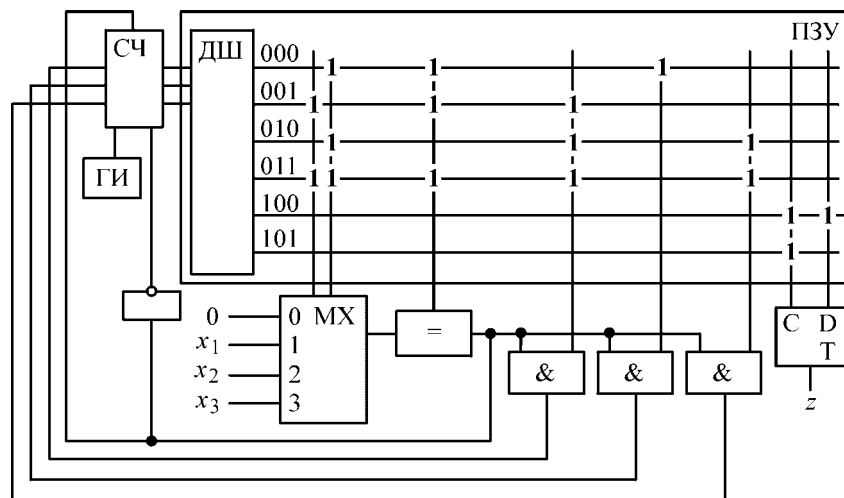


Рис. 18.22

В этой схеме используется счетчик, работающий в двух режимах — прибавление единицы и параллельная запись. Число команд, требующихся в данном случае для реализации произвольной булевой формулы в базисе $\{\&, \vee, !\}$ из h букв, удовлетворяет соотношению

$$K_{26} = h + 2,$$

а число разрядов в командах — соотношению

$$L_{26} = \lceil \log(n+1) \rceil + \lceil \log(h+2) \rceil + 3.$$

При $h = n \rightarrow \infty$

$$K_{26} = h; \quad L_{26} \rightarrow 2 \log h; \quad S_{26} \rightarrow 2h \log h.$$

При $n \rightarrow \infty$

$$K_{26} \rightarrow \frac{2^n}{\log n}; \quad L_{26} \rightarrow n; \quad S_{26} \rightarrow n \frac{2^n}{\log n}.$$

При использовании этого устройства в схеме на рис. 18.1 входные D -триггеры можно не применять при вычислении бесповторных формул в базисе $\{\&, \vee, !\}$, а также повторных формул в этом же базисе, в которых значения повторных переменных за время программного цикла не изменяются.

Это устройство рассмотрено в [51], а аналогичное устройство для реализации систем булевых формул, использующее вместо счетчика сумматор, — в [58].

18.3. Операторно-бинарные устройства

В силу того что операторно- 2^k -нарные устройства являются весьма громоздкими, а метод построения бинарных графов для их программирования не известен, то рассмотрим только операторно-бинарные устройства.

Среди устройств этого класса рассмотрим два подкласса, для которых отсутствует необходимость в применении ОЗУ для запоминания промежуточных результатов.

18.3.1. Четырехадресное операторно-бинарное устройство

Если булева формула реализуется операторно-бинарным графом (ОБГ) на основе разложения Шеннона по подформулам, то этот граф, в свою очередь, может быть реализован четырехадресным операторно-бинарным устройством (ОБУ4).

Пример 18.15. Реализовать ОБГ (рис. 18.23), который построен на основе разложения Шеннона по подформулам из двух букв булевой формулы из примера 18.13.

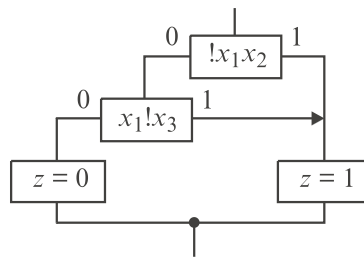


Рис. 18.23

На рис. 18.24 приведено устройство, позволяющее реализовать этот операторно-бинарный граф, а также другие ОБГ, в условных вершинах которых могут размещаться произвольные БФУ двух переменных.

Длина команд в этом устройстве определяется соотношением

$$L_{4об} = 2 \lceil \log n \rceil + 2 \lceil \log B \rceil + 6,$$

а число команд — соотношением

$$K_{4об} = B.$$

В [21] показано, что для ОБГ при $n \rightarrow \infty$

$$B \rightarrow \frac{2^{n-1}}{n},$$

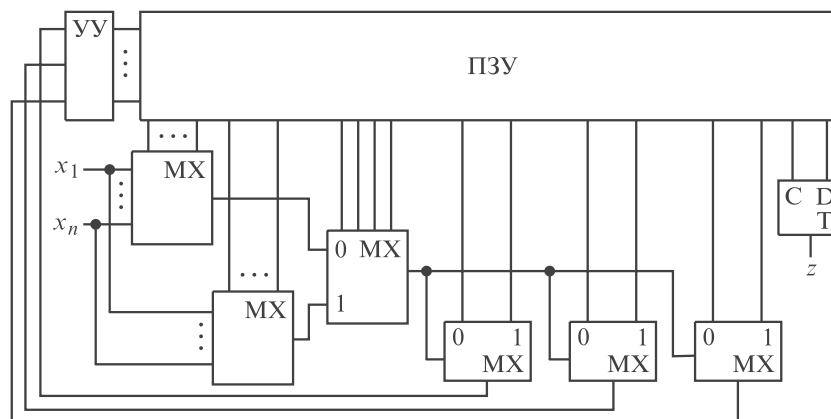


Рис. 18.24

и поэтому

$$K_{4об} \rightarrow \frac{2^{n-1}}{n}; L_{4об} \rightarrow 2n; S_{4об} \rightarrow 2^n.$$

Таким образом, построено последовательностное устройство, площадь ПЗУ которого совпадает с площадью ПЗУ, предназначенного для однократной реализации произвольной булевой функции n переменных.

Усложнение предлагаемого устройства по сравнению с ПЗУ той же площади позволяет упростить наращивание устройства по входам.

18.3.2. Трехадресное операторно-бинарное устройство

Если булева формула реализуется линейным ОБГ (ЛОБГ), то этот граф, в свою очередь, может быть реализован трехадресным операторно-бинарным устройством (ОБУЗ).

Пример 18.16. Реализовать ЛОБГ (рис. 18.25), который построен с помощью обобщенного формульного метода [59] по булевой формуле, рассмотренной в примере 18.13.

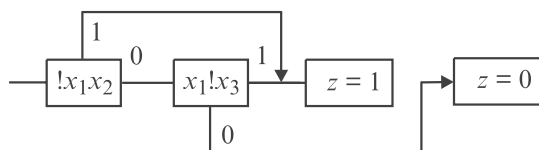


Рис. 18.25

На рис. 18.26 приведено устройство, позволяющее реализовать этот линейный ОБГ, а также другие ЛОБГ, в условных вершинах которых могут размещаться произвольные БФУ двух переменных.

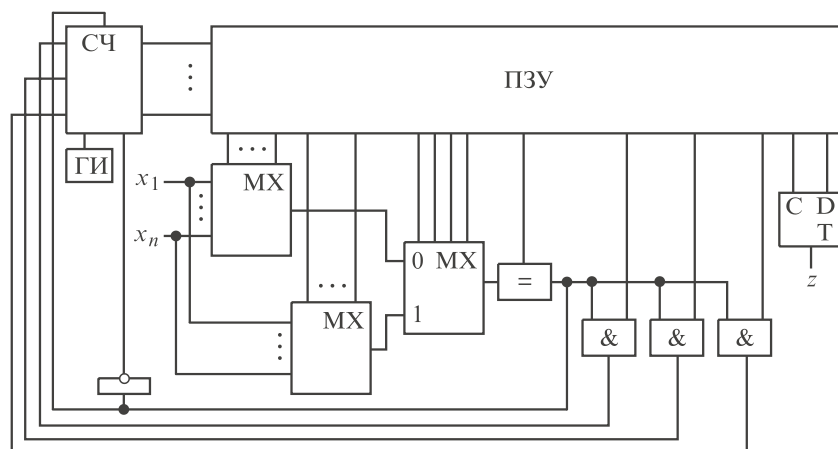


Рис. 18.26

Число команд в данном случае определяется соотношением

$$K_{3\text{об}} = \lceil h/2 \rceil + 2,$$

а их длина — соотношением

$$L_{3\text{об}} = 2 \lceil \log n \rceil + 2 \lceil \log(\lceil h/2 \rceil + 2) \rceil + 7.$$

При $h = n \rightarrow \infty$

$$K_{3\text{об}} = h/2; L_{3\text{об}} \rightarrow 3 \log h; S_{3\text{об}} \rightarrow 1.5 h \log h.$$

При $n \rightarrow \infty$

$$K_{3\text{об}} \rightarrow \frac{2^{n-1}}{\log n}; L_{3\text{об}} \rightarrow 2n; S_{3\text{об}} \rightarrow n \frac{2^{n-1}}{\log n}.$$

18.4. Ассоциативные устройства

18.4.1. Последовательностный аналог постоянного запоминающего устройства

Известно, что постоянные запоминающие устройства (ПЗУ) хорошо наращиваются по выходам и плохо — по входам [60]. Поэтому если при реализации булевой функции n переменных их подавать на входы ПЗУ, то такое устройство будет «гибким» до тех пор, пока $n \leq b$, где b — число входов ПЗУ.

Предложим устройство, построенное на базе ПЗУ, которое за счет подачи входы переменных на его выходы и перехода к многотактной обработке информации позволяет снять указанное ограничение.

Программирование таких устройств выполняется по совершенной дизъюнктивной нормальной форме (СДНФ). Такая форма используется при аппаратной (однотактной) реализации булевых функций в ПЗУ, и поэтому рассматриваемое устройство можно считать последовательностным аналогом ПЗУ.

В конъюнкцию СДНФ переменная x_i может входить как в прямой, так и в инверсной форме. Занесем для каждой конъюнкции СДНФ в ПЗУ маску M_i , i -й разряд которой (M_i) задается табл. 18.3.

Т а б л и ц а 18.3

Переменная	M_i
$\neg x_i$	0
x_i	1

Определим по этой таблице функцию $\Phi_i = f(M_i, x_i)$ (табл. 18.4).

Т а б л и ц а 18.4

M_i	x_i	Φ_i
0	0	1
0	1	0
1	0	0
1	1	1

Из табл. 18.4 следует, что

$$\Phi_i = (x_i = M_i).$$

Сформируем для каждой конъюнкции СДНФ из констант M_i маску M . В этой маске единицы соответствуют прямым переменным, а нули — инверсным.

Пример 18.17. Построить устройство рассматриваемого типа (рис. 18.27), вычисляющее СДНФ $z = \neg x_1 x_2 \vee x_1 \neg x_2$.

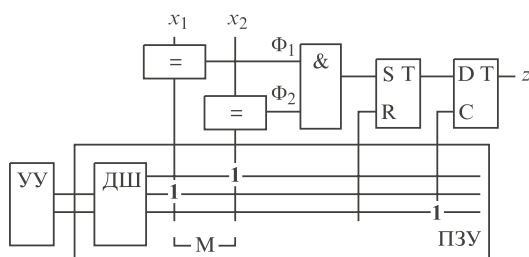


Рис. 18.27

Для корректного функционирования устройства RS -триггер должен иметь нулевую начальную установку.

Число команд в этом случае определяется соотношением

$$K_{A1} = q_1 + 1 \leq 2^{n-1} + 1,$$

где q_1 — число конъюнкций в СДНФ.

При этом длина команд (с учетом выходных D -триггеров) определяется соотношением

$$L_{A1} = n + 2.$$

Таким образом,

$$S_{A1} \leq (n + 2) (2^{n-1} + 1).$$

Компенсацией за большую площадь ПЗУ являются:

- простота наращивания устройства по входам;
- возможность одновременной реализации одинаковых конъюнкций, принадлежащих различным СДНФ.

Приведем пример реализации системы булевых формул с помощью рассматриваемого устройства.

Пример 18.18. Построить устройство рассматриваемого типа (рис. 18.28), реализующее систему булевых формул вида:
 $z_1 = x_1 \oplus x_2$; $z_2 = x_1 \vee x_2$.

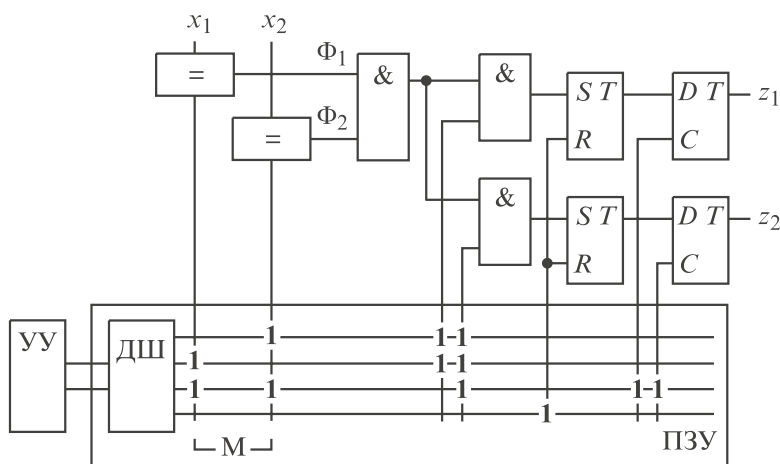


Рис. 18.28

Представим эти булевы формулы в СДНФ:

$$z_1 = !x_1 x_2 \vee x_1 !x_2; \quad z_2 = !x_1 x_2 \vee x_1 !x_2 \vee x_1 x_2.$$

Модификация рассмотренного устройства описана в [25].

18.4.2. Последовательностный аналог программируемой логической матрицы

Известно, что программируемые логические матрицы (ПЛМ) хорошо наращиваются по термам и выходам, но плохо — по входам [60].

Поэтому расширим подход, рассмотренный в предыдущем разделе, применительно к булевым функциям, заданным в дизъюнктивной нормальной форме (ДНФ).

Дизъюнктивные нормальные формы используются при аппаратной реализации булевых функций в ПЛМ, и поэтому рассматриваемое устройство можно считать последовательностным аналогом ПЛМ.

В конъюнкции ДНФ переменная x_i может не входить или входить в прямой или инверсной форме. Занесем для каждой конъюнкции ДНФ в постоянное запоминающее устройство (ПЗУ) две маски $M1$ и $M2$, i -е разряды которых ($M1_i$ и $M2_i$) задаются табл. 18.5.

Т а б л и ц а 18.5

Переменная	$M1_i$	$M2_i$
—	0	0
$!x_i$	1	0
x_i	1	1

Используя эту таблицу, определим функцию $\Phi_i = f(M1_i, M2_i, x_i)$, описанную табл. 18.6.

Т а б л и ц а 18.6

$M1_i$	$M2_i$	x_i	Φ_i
0	0	0	1
0	0	1	1
0	1	0	—
0	1	1	—
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Доопределим функцию Φ_i так, чтобы она могла быть реализована с помощью минимального числа элементов. При доопределении ее нулями получим:

$$\Phi_i = (x_i \& M1_i) = M2_i.$$

Число команд в этом устройстве определяется соотношением

где q_2 — число конъюнкций в ДНФ.

$$L_{A2} = 2(n + 1).$$
$$S_{A_2} = 2(n+1)(q_2+1).$$
$$q_2 \leq 3^{h/3}.$$
$$S_{A2} \leq 2(3^{h/3} + 1)(n + 1).$$

606

Для корректного функционирования устройства RS -триггер должен иметь нулевую начальную установку.

Приведем пример реализации системы ДНФ с помощью рассматриваемого устройства.

Пример 18.20. Построить устройство рассматриваемого типа (рис. 18.30), вычисляющее систему ДНФ $z_1 = !x_1!x_2 \vee x_1x_2 \vee x_3$, $z_2 = !x_1!x_2 \vee !x_4$.

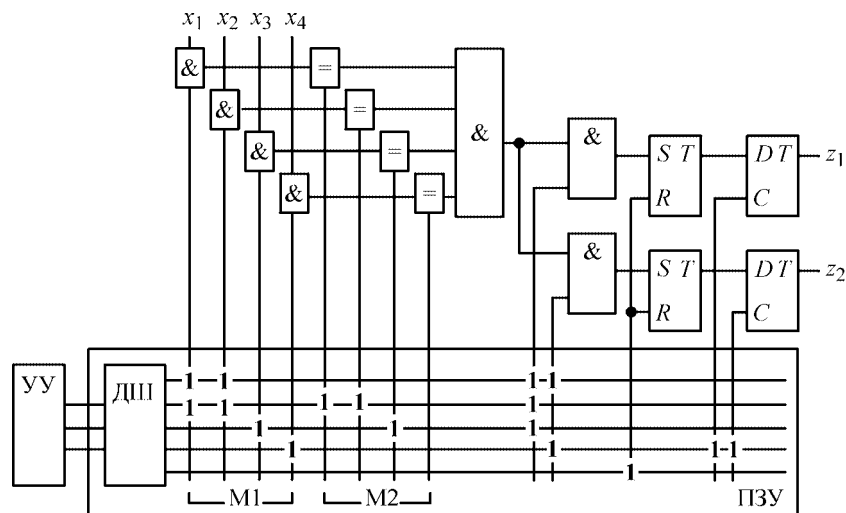


Рис. 18.30

Идея построения устройств рассмотренного типа изложена в [62, 63].

Рассмотренные выше устройства названы ассоциативными (АУ), так как в них поиск информации осуществляется не по адресу (входам ПЗУ), а по содержимому ПЗУ за счет сравнения на его выходах информации, записанной в нем, со значениями входных переменных. Такой подход назван в [64] поиском по ассоциации.

Устройства, рассмотренные в предыдущем разделе, обозначим символом АУ1, а устройства, описанные в настоящем разделе, — символом АУ2.

Менее удачным, по мнению автора, является другое название аналогичных устройств — «параллельные логические контроллеры» [42, 43].

18.5. Сравнительные характеристики рассмотренных устройств

Оценки числа команд и площади ПЗУ для рассмотренных устройств приведены в табл. 18.7.

Т а б л и ц а 18.7

Обозначение устройства	Число команд	Площадь ПЗУ ($h = n \rightarrow \infty$)	Площадь ПЗУ ($n \rightarrow \infty$)
ОУ3	$h - 1$	$2h \log h$	$3 \cdot 2^n$
ОУ2.5	$h \leq k \leq [(3h/2) - 1]$	$h \log h \leq S \leq (3h/2) \log h$	$3 \cdot 2^n \leq S \leq 4.5 \cdot 2^n$
ОУ2	$h \leq k \leq [(3h/2) - 1]$	$h \log h \leq S \leq (3h/2) \log h$	$2^{n+1} \leq S \leq 3 \cdot 2^n$
ОУ1	$h + 1 \leq k \leq 2h - 1$	$h \log h \leq S \leq 2h \log h$	$2^n \leq S \leq 2^{n+1}$
ОУ31	$h/2 \leq k \leq h - 1$	$h \log(h + f(h)) \leq S \leq \leq 4h \log(h + f(h))$	$n \cdot 2^n / \log n \leq S \leq \leq n \cdot 2^{n+1} / \log n$
ОУ21	$h + 1 \leq k \leq [3h/2]$	$3h \log h \leq S \leq 4.5h \log h$	$n \cdot 2^{n+1} / \log n \leq S \leq \leq 3n \cdot 2^n / \log n$
СОУ1	$h + 1 \leq k \leq [3h/2]$	$h \log h \leq S \leq 1.5h \log h$	$2^n \leq S \leq 1.5 \cdot 2^n$
БОУ	h	$2h \log h$	$n \cdot 2^{n+1} / \log n$
БУ3	$2^n / n$	—	2^{n+1}
БУ2	$h + 2$	$2h \log h$	$n \cdot 2^n / \log n$
ОБУ4	$2^{n-1} / n$	—	2^n
ОБУ3	$]h/2[+ 2$	$1.5h \log h$	$n \cdot 2^{n-1} / \log n$
АУ1	$\leq 2^{n-1} + 1$	—	$n \cdot 2^{n-1}$
АУ2	$\leq 3^{h/3} + 1$	$2h \cdot 3^{h/3}$	—

В этих устройствах не ограничивалась длина команд, так как предполагалось, что каждое устройство реализовано в виде одной компоненты. В случае, если оно реализуется в виде двух компонент (например, микросхем), одна из которых ПЗУ, указанное ограничение может накладываться [9]. При этом команды, записанные в ПЗУ, кодируются «плотно», а во вторую компоненту вводится дешифратор, преобразующий «узкую» команду в ту, которая требуется в соответствующем из рассмотренных выше устройств [65].

18.6. Микропроцессорные устройства

Структура приведенных выше логических устройств существенно отличается от структуры универсальных микропроцессоров, одной из особенностей которых является многоадресность.

Несмотря на то что некоторые из рассмотренных устройств могут одновременно обрабатывать несколько входных переменных, для них более характерна битовая обработка, так как логические переменные по своей природе являются битовыми (одноразрядными): один бит отражает всю требуемую информацию о каждом объекте или процессе (например, открыт или не открыт клапан). Поэтому такие устройства, может быть и не всегда точно, называют однобитовыми [20], или одноразрядными.

При необходимости выполнения в одном устройстве как логической, так и арифметической обработки информации возможны различные подходы к его построению и использованию:

— в устройстве применяются два процессорных элемента — одноканальный и многоканальный [4];

— в устройстве используется один многоканальный процессорный элемент, а логические функции заменяются арифметическими, например арифметическими полиномами [66];

— в устройстве применяется один многоканальный процессорный элемент, в котором логические операции выполняются с помощью логических (битовых) команд и (или) условных переходов, что позволяет в этом случае моделировать работу одноканальных процессоров;

— в устройстве используется один многоканальный процессорный элемент, в котором логические команды выполняются с помощью многоканальных битовых команд, что позволяет, в частности, моделировать работу ассоциативных логических устройств.

Еще одна возможность имеется в микроконтроллерах, в которых применяется один многоканальный процессорный элемент: логические операции могут выполняться в том числе и с помощью логических команд над битами одного слова, что также позволяет моделировать работу одноканальных процессоров [7].

Пример 18.21. Реализовать булеву формулу $z = (x_1 x_2 \vee x_3!x_4)x_5$ с помощью бинарной программы, написанной на языке макроассемблера ЭВМ PDP-11 или VAX, при условии, что переменные x_1, x_2, x_3 соответственно размещены в 14-, 6- и 3-м разрядах первого поля, а переменные x_4, x_5 — соответственно в 5- и 2-м разрядах второго поля.

Построим по заданной формуле линейный бинарный граф (рис. 18.31), введя в него операторную вершину, помеченную символом w , которая соответствует команде безусловного перехода в программе.

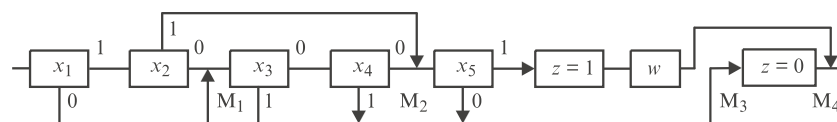


Рис. 18.31

Этому ЛБГ соответствует следующая программа:

```

        BIT #040000, POLE1
        BEQ M1
        BIT #000100, POLE1
        BNE M2
M1:     BIT #000010, POLE1
        BEQ M3
        BIT #000040, POLE2
        BNE M3
M2:     BIT #000004, POLE2
        BEQ M3
        MOV #1, z
        BR  M4
M3:     CLR z
M4:

```

Константы в этой программе записаны в восьмеричной системе счисления и являются масками для выделения соответствующих упакованных входных переменных.

В работах [52, 67] показано, что вне зависимости от расположения входных переменных в одном или разных полях число команд K и число слов V в программе, реализующей произвольную формулу в базисе $\{\&, \vee, !\}$ из h букв, определяются соотношениями

$$K = 2h + 3, \quad V = 2K.$$

В [45] рассмотрено более 60 программ на языке СИ, построенных различными методами, и каждая из них реализует логическую функцию «голосование два и более из трех» при размещении каждой входной переменной в крайнем правом разряде соответствующего слова. Выполнено сравнение этих программ по объему памяти (без учета оформления в виде функций) и быстродействию при их реализации на процессоре 80286. Показано, что наилучшей по этим показателям является программа, построенная по арифметическому полиному с маскированием [68]:

```

laps (int x1, int x2, int x3)
{int z;
 z = (x1 + x2 + x3) >> 1;
 return (z);}

```

Объем этой программы 14 байт, в то время как программа, использующая битовые операции и компилируемая как операторная,

```

sf2 (int x1, int x2, int x3)
{int z;
 z = ((x1 | x2) & x3) | (x1 & x2);
 return (z);}

```

требует 20 байт памяти, а программа, в которой применяются логические операции, компилируемая как бинарная,

```
sfb (int x1, int x2, int x3)
{int z;
  z = (x3 && (x1 || x2)) || (x1 && x2);
  return (z);}
```

требует 40 байт памяти.

В заключение главы отметим, что вопрос о реализации булевых формул на машинах Тьюринга рассмотрен в [71].

Выводы

1. Рассмотрены четыре класса специализированных логических устройств для последовательного вычисления булевых функций: операторные, 2^k -нарные, операторно-бинарные и ассоциативные. Эти классы устройств, использующие традиционные принципы логической обработки информации, не исчерпывают всех классов логических устройств, которые, в частности, могут быть построены на основе вычисления арифметических полиномов.

2. В каждом классе предложены одно или несколько устройств, каждое из которых обладает особенностями, позволяющими выделить его в отдельный подкласс.

3. Предложен принципиально новый подкласс операторных устройств, обеспечивающих побуквенное вычисление произвольных нормальных булевых формул в базисе $\{\&, \vee, !\}$ из h букв, несмотря на наличие в них скобок произвольной глубины.

4. Получены оценки числа команд и площади ПЗУ для рассмотренных устройств.

5. Предложено операторно-бинарное устройство, у которого площадь ПЗУ асимптотически равна 2^n , что совпадает с площадью ПЗУ при одноктактной реализации булевой функции n переменных. Преимущество такого устройства по сравнению с ПЗУ состоит в упрощении расширения по входам.

6. Предложены ассоциативные логические устройства, обеспечивающие наиболее простое расширение по входам.

7. Рассмотрен вопрос о применении различных методов последовательного вычисления булевых функций при программировании универсальных микропроцессоров.

8. Все устройства, рассмотренные в настоящей главе, являются синхронными. Реализация простых бинарных графов асинхронными автоматами и метод противоголочного кодирования их состояний описаны в работах [65, 69].

Л и т е р а т у р а

1. Павлов В. В. Управляющие логические машины — новое средство автоматического управления производственными процессами // Приборы и средства автоматизации. 1969. № 3.
2. Бутин Ю. Н., Маковеев О. Л., Харинский С. В. и др. Программные логические устройства. Л.: Судостроение, ЦНИИ «Румб», 1979.
3. Амбарцумян А. А., Потехин А. И., Запольский Е. Н. Программируемые логические контроллеры и их применение // Измерения. Контроль. Автоматизация. 1979. Вып. 4.
4. Мишель Ж. Программируемые контроллеры. Архитектура и применение. М.: Машиностроение, 1992.
5. International Standard IEC 1131. Programming languages // International Electrotechnical Commission. 1993.
6. Новик Г. Х., Першеев В. Г., Шамров М. И. Реализация комбинационных схем микропроцессорным автоматом // Приборы и системы управления. 1980. № 3.
7. Сташин В. В., Урусов А. В., Мологонцева О. Ф. Проектирование цифровых устройств на однокристалльных микроконтроллерах. М.: Энергоатомиздат. 1990.
8. Бутин Ю. Н., Золотаревская М. Я., Кириллов А. П., Юнг В. Н. О реализации алгоритмов логического управления в специализированных программируемых логических устройствах // Автоматика и телемеханика. 1983. № 6.
9. MC 14500B. Industrial control unit. Handbook Motorola. S. Products. 1977.
10. Ли С. Представление переключаемых схем с помощью программ двоичного решения // Вопросы теории математических машин. М.: Машиностроение, 1964.
11. Хазацкий В. Е. Некоторые вопросы построения управляющих логических устройств с общим функциональным логическим блоком // Труды ЦНИИКА. 1966. Вып. 14.
12. Альтигуль С. Д., Гильман Г. И., Рог Г. В. и др. Логическое устройство. А. с. СССР № 189630 // Бюл. изобр. 1966. № 24.
13. Гильман Г. И. Исследование и разработка универсальной логической машины для автоматизации судовых систем // Труды ЦНИИ им. А. Н. Крылова. 1966. Вып. 231.
14. Альтигуль С. Д., Буянов Б. Б., Волков А. Ф. и др. Универсальная логическая машина для управления производственными процессами // Приборы и системы управления. 1967. № 6.
15. Нефедов Г. С. О последовательном вычислении логических функций // Автоматика и телемеханика. 1973. № 2.
16. Савченко Ю. Г., Хмелева А. В. Некоторые задачи синтеза дискретных устройств управления медленными процессами // Дискретные системы. Т. 1. Симпозиум IFAC. Рига: Зинатне, 1974.
17. Балашихов Е. П., Пузанков Д. В. Логические процессоры для реализации разветвленных алгоритмов // Управляющие системы и машины. 1974. № 6.
18. Блох А. Ш. Граф-схемы и их применение. Минск: Вышэйшая школа, 1975.
19. Загарий Г. И. Принципы построения устройств управления последовательными операциями (обзор) // Управляющие системы и машины. 1975. № 1.
20. Свит А. Новый подход к программируемой логике на основе использования однокристалльного процессора // Экспресс-информация. Вычислительная техника. 1976. № 8.
21. Волков А. Ф., Лукашенко Г. А., Луненко К. М. и др. Управляющая логическая машина. А. с. СССР № 539301 // Бюл. изобр. 1976. № 46.
22. Кузьмин В. А. Оценка сложности реализации функций алгебры логики простейшими видами бинарных программ // Методы дискретного анализа в теории кодов и схем. Новосибирск, 1976. Вып. 29.
23. Кузнецов О. П. О программной реализации логических функций и автоматов // Автоматика и телемеханика. 1977. № 7, 9.
24. Глушков В. М., Деркач В. П., Корсунский В. М. и др. Микропроцессоры на ПЗУ // Управляющие системы и машины. 1977. № 5.

25. Румянцев И. А. Введение в судовую последовательную автоматику и принципы агрегатирования // Теоретические основы автоматизации процессов управления судном. Труды Ленингр. высш. инженер. морск. училища. 1975. Вып. 3.
26. Гильман Г. И., Лазарев В. Г., Альтигуль С. Д. Устройство для решения логических задач. А. с. СССР № 631160 // Бюл. изобр. 1977. № 12.
27. Камынин Ю. Н., Матвиенко Н. П., Жигульцев А. Ю. Управляющие логические устройства с перестраиваемой структурой // Приборы и системы управления. 1978. № 11.
28. Pokoski J. L. Software analyses for combinatorial logic // Compt. Design. 1978. № 6.
29. Plavic V., Danielsson P. Sequential evaluation of Boolean functions // IEEE Trans. Computers. 1979. № 12.
30. Абугов Ю. О., Диденко К. И., Загарий Г. И. и др. Микроэлектронные устройства программного и логического управления. М.: Машиностроение, 1979.
31. Гаранин Н. А. Программирование релейно-контактных схем управления в системе команд УЛП // Труды Всесоюз. науч.-исслед., проектно-конструктор. и технол. ин-та релестроения. Чебоксары, 1979. Вып. 10.
32. Капитонова Ю. В., Алексеенко В. Г., Миценко А. Т. Реализация устройств управления на основе больших стандартных схем // Кибернетика. 1979. № 3, 4.
33. Балаиов Е. П., Негода В. Н., Пузанков Д. В. Проектирование логических процессоров на ПЗУ // Управляющие системы и машины. 1980. № 5.
34. Золотаревская М. Я. О реализации систем булевых функций в программном логическом устройстве // Проблемы управления в технике, экономике, биологии. М.: Наука, 1981.
35. Срибнер Л. А. Программируемые устройства автоматики. Киев: Техника, 1982.
36. Иванов А. В. Минимизация длины стека при вычислении выражений // Программирование. 1982. № 1.
37. Бойет Г., Кац Р. Программная реализация комбинационных логических схем // Электроника. 1983. № 6.
38. Лапкин Л. Я. О векторной программной реализации логических функций // Автоматика и телемеханика. 1983. № 3.
39. Лазарев В. Г., Пийль Е. И., Турута Е. Н. Построение программируемых управляющих устройств. М.: Энергоатомиздат, 1984.
40. Пупырев Е. И. Перестраиваемые автоматы и микропроцессорные системы. М.: Наука, 1984.
41. Девятков В. В. Стековая программная реализация алгоритмов логического управления // Проектирование устройств логического управления. М.: Наука, 1984.
42. Вышинский В. А., Фурман И. А. Об использовании булевых матриц и операции над ними для построения быстродействующих программируемых устройств логического управления // Управляющие системы и машины. 1986. № 2.
43. Фурман И. А., Никонов А. И. Математическая модель и принципы структурной организации параллельных логических контроллеров // Управляющие системы и машины. 1986. № 4.
44. Лапкин Л. Я. Оценки сложности программной реализации при различных формах задания булевых функций // Автоматика и телемеханика. 1987. № 1.
45. Шальто А. А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
46. Евтодьева А. И., Клунт Б. Я., Шальто А. А. и др. Устройство для реализации логических функций. А. с. СССР № 1001080 // Бюл. изобр. 1983. № 8.
47. Артюхов В. Л., Кузнецов Б. П., Шальто А. А. Настраиваемые бинарные процедуры и циклические программы // Автоматика и телемеханика. 1984. № 11.
48. Лупанов О. Б. Асимптотические оценки сложности управляющих систем. М.: Изд-во МГУ, 1984.
49. Autolog 32. Руководство пользователя. FF-Automation OY.
50. SYSMAC. Programmable controllers. CV500/CV100. Operation Manual. Omron. 1993.

51. Артюхов В. Л., Кузнецов Б. П., Шалыто А. А. Настраиваемые логические устройства для судовых управляющих систем. Л.: ИПК СП, 1986.
52. Шалыто А. А. Программная реализация алгоритмов логического управления судовыми системами. Л.: ИПК СП, 1989.
53. Артюхов В. Л., Шалыто А. А. Судовые управляющие логические системы. Л.: ИПК СП, 1983.
54. Рубинов В. И., Шалыто А. А. Метод построения граф-схем простых бинарных программ для систем булевых функций // Автоматика и вычисл. техника. 1986. № 4.
55. Артюхов В. Л., Кузнецов Б. П., Шалыто А. А. Линейные бинарные графы: свойства, характеристики, алгоритмы // Методы и программы решения оптимизационных задач на графах и сетях. Ч. 2. Тезисы докл. III Всесоюз. совещ. Новосибирск, 1984.
56. Wilkes M. V., Stringer J. B. Microprogramming and the design of the control circuits in an electronic digital computer // Proc. Cambridge Philos. Soc. 1953. № 4.
57. Лазарев В. Г., Пийль Е. И. Синтез управляющих автоматов. М.: Энергия, 1978.
58. Вавилов В. Н., Вальшинок Е. С., Шалыто А. А. и др. Устройство для вычисления булевых функций. А. с. СССР № 1501033 // Бюл. изобр. 1989. № 8.
59. Артюхов В. Л., Кузнецов Б. П., Шалыто А. А. и др. Особенности записи булевых формул при их программной реализации // Вопросы судостроения. Сер. Судовая автоматика. 1984. Вып. 31.
60. Баранов С. И., Скляр В. А. Цифровые устройства на программируемых БИС с матричной структурой. М.: Радио и связь, 1986.
61. Артюхов В. Л., Кузнецова О. С., Шалыто А. А. Оценка функциональных возможностей программируемых логических матриц // Автоматика и вычисл. техника. 1985. № 2.
62. Артюхов В. Л., Шалыто А. А. Программируемое логическое устройство. А. с. СССР № 1587487 // Бюл. изобр. 1990. № 31.
63. Артюхов В. Л., Шалыто А. А. Программируемое логическое устройство. А. с. СССР № 1587488 // Бюл. изобр. 1990. № 31.
64. Коханен А. Ассоциативные устройства. М.: Мир, 1986.
65. Артюхов В. Л., Сагалович Ю. Л., Шалыто А. А. и др. Исследование целесообразности и возможности создания малогабаритных перепрограммируемых логических устройств для систем с децентрализованной структурой. Отчет по НИР. Л.: НПО «Аврора», 1982.
66. Артюхов В. Л., Кондратьев В. Н., Шалыто А. А. Реализация булевых функций арифметическими полиномами // Автоматика и телемеханика. 1988. № 4.
67. Кузнецов Б. П., Шалыто А. А. Реализация булевых формул линейными бинарными графами. I. Синтез и анализ // Известия РАН. Техн. кибернетика. 1994. № 5.
68. Кондратьев В. Н., Шалыто А. А. Реализация булевых функций одним линейным арифметическим полиномом с маскированием // Автоматика и телемеханика. 1996. № 1.
69. Сагалович Ю. Л., Шалыто А. А. Бинарные программы и их реализация асинхронными автоматами // Проблемы передачи информации. 1987. № 1.
70. Проблемы выбора и перспективы систем автоматизации. Отдел управления и информации компании «Rockwell Automation» // Мир компьютерной автоматизации. 1998. № 2.
71. Сэвидж Д. Э. Сложность вычислений. М.: Факториал, 1998.

Глава 19

Нетрадиционные методы вычисления булевых функций

19.1. Метод логических шкал

Простейший метод вычисления произвольной булевой функции, заданной таблицей истинности, состоит в выборе ее значения по входному набору, по которому первоначально определяется его десятичный эквивалент, вычисляемый, например, при $n = 3$ таким образом:

$$xt = 4x_1 + 2x_2 + x_3.$$

При этом функция «голосование два и более из трех» может быть вычислена следующей программой:

```
ti(int xt)
{int y;
 int yt[8] = {0, 0, 0, 1, 0, 1, 1, 1};
 y = yt[xt];}
```

Аналогично может быть вычислена и произвольная система БФУ, заданная таблицей истинности. Например, одноразрядный сумматор реализуется следующей программой:

```
ti1(int xt)
{int y;
 int yt[8] = {0, 1, 1, 2, 1, 2, 2, 3};
 y = yt[xt];}
```

Однако выбор значения БФУ, заданной таблицей истинности, может быть выполнен существенно иначе, если столбец значений

задать в виде десятичного числа. При этом значению БФУ на наборе xt присваивается вес 2^{xt} .

При таком подходе столбцу значений функции «голосование два и более из трех» соответствует число

$$232 = 2^7 \cdot 1 + 2^6 \cdot 1 + 2^5 \cdot 1 + 2^4 \cdot 0 + 2^3 \cdot 1 + 2^2 \cdot 0 + 2^1 \cdot 0 + 2^0 \cdot 0.$$

Тогда заданная БФУ может быть реализована следующей программой:

```
lsh(int xt)
{int y;
 y = (232 >> xt) & 1;}
```

Здесь $>>$ — операция сдвига двоичного представления числа 232 на xt разрядов вправо, а $\&1$ — операция маскирования, сохраняющая значение крайнего правого разряда.

Используя этот подход, могут быть записаны следующие соотношения:

$$\begin{aligned}x_1 x_2 x_3 &= (128 >> xt) \& 1; \\ !x_1 !x_2 !x_3 \vee x_1 x_2 x_3 &= (129 >> xt) \& 1; \\ x_1 \oplus x_2 \oplus x_3 &= (150 >> xt) \& 1; \\ x_1 \vee x_2 \vee x_3 &= (255 >> xt) \& 1.\end{aligned}$$

Рассмотренный подход может быть применен и для систем БФУ. При этом значениям пары функций, описывающих одноразрядный сумматор, соответствует число:

$$\begin{aligned}59\,796 &= 2^{15} \cdot 1 + 2^{14} \cdot 1 + 2^{13} \cdot 1 + 2^{12} \cdot 0 + 2^{11} \cdot 1 + 2^{10} \cdot 0 + 2^9 \cdot 0 + \\ &+ 2^8 \cdot 1 + 2^7 \cdot 1 + 2^6 \cdot 0 + 2^5 \cdot 0 + 2^4 \cdot 1 + 2^3 \cdot 0 + 2^2 \cdot 1 + 2^1 \cdot 0 + 2^0 \cdot 0.\end{aligned}$$

Тогда заданная система БФУ может быть реализована следующей программой:

```
lsh1(int xt)
{int y, p, m = 59796L;
 p = xt << 1;
 y = (m >> p) & 3;}
```

В этой программе сдвиг вправо осуществляется на четное число разрядов и сохраняются два крайних правых разряда.

В. И. Варшавский сообщил автору, что изложенный подход называется методом логических шкал, но не указал, где этот метод был опубликован.

19.2. Реализация одной булевой функции с помощью линейных арифметических выражений

19.2.1. Реализация пороговых функций

Известно [1], что пороговые функции (ПФ) могут быть реализованы с использованием арифметических выражений, являющихся линейными арифметическими полиномами. При этом ПФ реализуется выражением вида:

$$f = \text{sign} \left(\sum_{i=1}^n p_i x_i - T \right),$$

где

$$\text{sign } A = \begin{cases} 1 & \text{при } A \geq 0; \\ 0 & \text{при } A < 0. \end{cases}$$

Это позволяет реализовать, например, функцию «голосование два и более из трех» следующей программой:

```
prg(int x1, int x2, int x3)
{int y = 0;
 if(x1 + x2 + x3 - 2 ≥ 0) y = 1;}
```

Определение весов переменных p_i и порога функции T в общем случае является весьма трудоемкой задачей, однако при $n \leq 6$ для нахождения пороговых реализаций можно применять каталоги [2].

19.2.2. Реализация неповторных пороговых формул

Для положительно монотонных формул этого класса из h букв автором предложен простой метод построения пороговых реализаций, состоящий в следующем:

- для заданной неповторной булевой формулы (ББФ) в базисе И, ИЛИ строится древовидная схема максимальной глубины из $h - 1$ двухвходовых элементов И и ИЛИ. Если в построенной схеме все элементы соединены последовательно, то рассматриваемая ББФ является пороговой;

- для пороговой ББФ строится линейный бинарный граф (ЛБГ), содержащий максимальное число путей;

- используя метод Акерса, в ЛБГ подсчитывается число путей;

- пометка входа i -й условной вершины ЛБГ равна весу переменной x_i , а пометка входа «нулевой» операторной вершины равна порогу функции.

Пример применения этого метода, описанного в [3], изложен в разд. 15.5.

Если в заданную ББФ переменная x_i входит с инверсией, то после нахождения пороговой реализации для положительно монотонной ББФ, однотипной с заданной, пороговая реализация корректируется на основе правила [1]:

$$\{p_1, \dots, -p_i, \dots, p_n, T - p_i\}.$$

19.2.3. Реализация произвольных булевых формул с помощью суперпозиции неповторных пороговых формул

Известны методы реализации произвольных БФУ с помощью суперпозиции пороговых функций [1]. Однако эти методы весьма трудоемки.

Автором в [4] предложен простой метод реализации формул в базисе И, ИЛИ, НЕ из h букв схемой из пороговых элементов, число которых определяется соотношением

$$1 \leq L \leq \lfloor h/2 \rfloor.$$

Метод состоит в построении для заданной БФ древовидной схемы из двухвходовых элементов И и ИЛИ, обладающей максимальной глубиной. В этой схеме выделяются каскады максимальной глубины, каждый из которых соответствует пороговому элементу.

На рис. 19.1 приведена схема, реализующая БФ

$$f = (x_1 x_2 \vee x_3 x_4)(x_5 \vee x_6),$$

на которой выделены три каскада (три пороговых элемента).

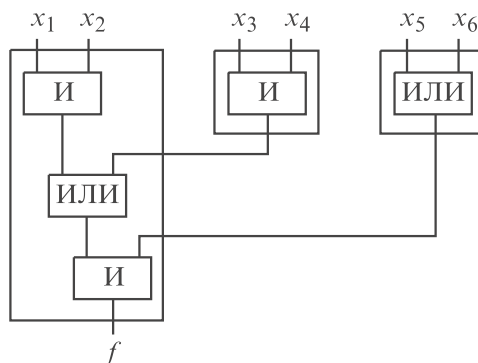


Рис. 19.1

**19.2.4. Реализация булевых формул
одним линейным арифметическим выражением,
использующим операции округления до ближайшего целого**

В [5] показано, что

$$x_1 \& x_2 \& \dots \& x_n = \left\lfloor \frac{x_1 + x_2 + \dots + x_n}{n} \right\rfloor = \left\lfloor \frac{\sum_{i=1}^n x_i}{n} \right\rfloor; \quad (19.1)$$

$$x_1 \vee x_2 \vee \dots \vee x_n = 1 - \left\lfloor \frac{n - x_1 - \dots - x_n}{n} \right\rfloor. \quad (19.2)$$

Применение этих соотношений и равенства $!x = 1 - x$ обеспечивает возможность реализации произвольной БФ в базисе И, ИЛИ, НЕ с использованием следующих базовых операций: сложение, вычитание, деление, округление до ближайшего меньшего целого.

Покажем, что могут быть построены арифметические выражения, в которых применяется меньшее число базовых операций.

Так как

$$\begin{aligned} [1 - A] &= 1 -]A[, \\ x_1 \vee x_2 \vee \dots \vee x_n &= 1 - \left\lfloor 1 - \frac{x_1 + \dots + x_n}{n} \right\rfloor = \\ &= 1 - 1 + \left\lfloor \frac{x_1 + \dots + x_n}{n} \right\rfloor = \left\lfloor \frac{\sum_{i=1}^n x_i}{n} \right\rfloor. \end{aligned} \quad (19.3)$$

Из соотношений (19.1) и (19.3) следует, что конъюнкция (дизъюнкция) является средним арифметическим, округленным в меньшую (большую) сторону.

При наличии инверсий справедливы следующие соотношения:

$$x_1^{g_1} \& x_2^{g_2} \& \dots \& x_n^{g_n} = \left\lfloor \frac{\sum_{i=1}^n g_i + \sum_{g_i=0}^n x_i^{g_i} - \sum_{g_i=1}^n x_i^{g_i}}{n} \right\rfloor; \quad (19.4)$$

$$x_1^{g_1} \vee x_2^{g_2} \vee \dots \vee x_n^{g_n} = \left[\frac{\sum_{i=1}^n g_i + \sum_{g_i=0}^n x_i^{g_i} - \sum_{g_i=1}^n x_i^{g_i}}{n} \right], \quad (19.5)$$

где $g_i = 0$ для $!x_i$ и $g_i = 1$ для x_i .

Из приведенных соотношений следует, что

$$\begin{aligned} x_1 x_2 \vee x_3 x_4 &= \left[\frac{\left[\frac{x_1 + x_2}{2} \right] + \left[\frac{x_3 + x_4}{2} \right]}{2} \right]; \\ (x_1 \vee x_2) x_3 &= \left[\frac{\left[\frac{x_1 + x_2}{2} \right] + x_3}{2} \right]; \\ !x_1 x_2 \vee x_1 !x_2 &= \left[\frac{\left[\frac{1 - x_1 + x_2}{2} \right] + \left[\frac{1 + x_1 - x_2}{2} \right]}{2} \right]. \end{aligned}$$

Приведем еще одно выражение для дизъюнкции:

$$x_1 \vee x_2 \vee \dots \vee x_n = \left[\frac{x_1 + x_2 + \dots + n - 1}{n} \right]. \quad (19.6)$$

При этом справедливо соотношение

$$x_1 x_2 \vee x_3 x_4 = \left[\frac{\left[\frac{x_1 + x_2}{2} \right] + \left[\frac{x_3 + x_4}{2} \right] + 1}{2} \right]. \quad (19.7)$$

Используя соотношения (19.1) и (19.6), введем два новых, позволяющих проводить деление не на произвольные целые числа, а всегда только на степени двух:

$$x_1 \vee x_2 \vee \dots \vee x_n = \left\lceil \frac{x_1 + x_2 + \dots + 2^{\lfloor \log n \rfloor} - 1}{2^{\lfloor \log n \rfloor}} \right\rceil; \quad (19.8)$$

$$x_1 \& x_2 \& \dots \& x_n = \left\lfloor \frac{x_1 + x_2 + \dots + 2^{\lfloor \log n \rfloor} - n}{2^{\lfloor \log n \rfloor}} \right\rfloor. \quad (19.9)$$

Исходя из изложенного, могут быть записаны следующие соотношения:

$$\begin{aligned} x_1 \vee x_2 \vee x_3 &= \left\lceil \frac{x_1 + x_2 + x_3}{3} \right\rceil = \left\lceil \frac{x_1 + x_2 + x_3 + 2}{3} \right\rceil = \\ &= \left\lceil \frac{x_1 + x_2 + x_3 + 3}{4} \right\rceil; \\ x_1 \& x_2 \& x_3 &= \left\lfloor \frac{x_1 + x_2 + x_3}{3} \right\rfloor = \left\lfloor \frac{x_1 + x_2 + x_3 + 1}{3} \right\rfloor. \end{aligned}$$

При наличии инверсий сначала применяются соотношения (19.8) и (19.9), которые в дальнейшем могут быть изменены на основе соотношений (19.4) и (19.5):

$$\begin{aligned} x_1 \vee x_2 \vee !x_3 &= \left\lceil \frac{x_1 + x_2 - x_3 + 1}{3} \right\rceil = \left\lceil \frac{x_1 + x_2 - x_3 + 3}{3} \right\rceil = \\ &= \left\lceil \frac{x_1 + x_2 - x_3 + 4}{4} \right\rceil. \\ x_1 \& x_2 \& !x_3 &= \left\lfloor \frac{x_1 + x_2 - x_3 + 1}{3} \right\rfloor = \left\lfloor \frac{x_1 + x_2 - x_3 + 2}{4} \right\rfloor. \end{aligned}$$

Деление на степени двух позволяет сократить число базовых операций до трех: сложение, вычитание, сдвиг вправо. Это объясняется тем, что деление на два в степени n двоичных чисел эквивалентно сдвигу (обозначается символом $\gg$) этого числа на n разрядов вправо. При этом округление в меньшую сторону происходит автоматически.

Исходя из изложенного, соотношение (19.7) может быть записано следующим образом:

$$x_1 x_2 \vee x_3 x_4 = (((x_1 + x_2) \gg 1) + ((x_3 + x_4) \gg 1) + 1) \gg 1.$$

19.3. Реализация логических функций интерполяционными полиномами

19.3.1. Реализация полностью определенных булевых функций

Пусть заданы $m + 1$ узловых точек $x_0 < x_1 < \dots < x_m$ и $m + 1$ действительных чисел $f_i (i = 0, \dots, m)$, являющихся значениями функции $f(X)$ в этих узловых точках.

При этом задача интерполяции ставится следующим образом: найти многочлен $I_m(X)$ степени, не большей m , такой что $I_m(x_i) = f_i$ при $i = \{0, \dots, m\}$.

Предположим, что значения аргумента отстоят на единицу и принимают значение от нуля до m , тогда можно построить интерполяционный полином (ИП) Ньютона по нисходящим разностям [6], который имеет вид:

$$I_m(X) = \sum_{k=0}^m \binom{X}{k} p^k Y = \binom{X}{0} p^0 Y + \binom{X}{1} p^1 Y + \binom{X}{2} p^2 Y + \dots,$$

$$\text{где } \binom{X}{0} = 1; \binom{X}{1} = \frac{X}{1!}; \binom{X}{2} = \frac{X(X-1)}{2!}; \dots$$

При реализации булевых функций $m = 2^n - 1$, а X и Y — десятичные эквиваленты входного и выходного наборов. Покажем на примерах, как определяются приращения $p^k Y$.

Пример 19.1. Определить ИП для булевой функции $f_1 = x_1 \oplus x_2$.

Построим интерполяционную таблицу (ИТ) для этого примера (табл. 19.1).

Т а б л и ц а 19.1

X	x_1	x_2	f_1	$Y = p^0 Y$	$p^1 Y$	$p^2 Y$	$p^3 Y$
0	0	0	0	0			
				→	1		
1	0	1	1	1	→	-1	
				→	0	→	0
2	1	0	1	1	→	-1	
				→	-1		
3	1	1	0	0			

Таким образом, в данном случае

$$f_1(X) = 1 \cdot 0 + \frac{X}{1!} \cdot 1 - \frac{X(X-1)}{2!} \cdot 1 + \frac{X(X-1)(X-2)}{3!} \cdot 0 = \\ = \frac{X}{2}(3-X),$$

где $X = 2x_1 + x_2$.

Пример 19.2. Определить ИП для булевой функции $f_2 = x_1 \vee x_2$. Построим ИТ для этого примера (табл. 19.2).

Т а б л и ц а 19.2

X	x_1	x_2	f_2	$Y = p^0 Y$	$p^1 Y$	$p^2 Y$	$p^3 Y$
0	0	0	0	0			
				→	1		
1	0	1	1	1	→	-1	
				→	0	→	1
2	1	0	1	1	→	0	
				→	0		
3	1	1	1	1			

Таким образом, в данном случае

$$f_2(X) = 1 \cdot 0 + \frac{X}{1!} \cdot 1 - \frac{X(X-1)}{2!} \cdot 1 + \frac{X(X-1)(X-2)}{3!} \cdot 1 = \\ = \frac{X}{6}(X^2 - 6X + 11).$$

Аналогично можно показать, что для формул: $f_3 = x_1 x_2$, $f_4 = !x_1 x_2$, $f_5 = x_1 !x_2$, $f_6 = !x_1 !x_2$, — соответствующие ИП имеют вид:

$$f_3(X) = \frac{X(X-1)(X-2)}{6},$$

$$f_4(X) = \frac{X(X-2)(X-3)}{2},$$

$$f_5(X) = -\frac{X(X-1)(X-3)}{2},$$

$$f_6(X) = -\frac{(X-1)(X-2)(X-3)}{6}.$$

На основе этих соотношений интерполяционный полином для булевой функции f_1 может быть получен иначе. Так как в совер-

шенной дизъюнктивной нормальной форме этой функции конъюнкции ортогональны, то можно записать выражение

$$f_1 = !x_1 x_2 \vee x_1 !x_2 = !x_1 x_2 + x_1 !x_2.$$

Поэтому

$$f_1(X) = \frac{X(X-2)(X-3)}{2} - \frac{X(X-1)(X-3)}{2} = \frac{X}{2}(3-X).$$

Пример 19.3. Определить ИП для системы булевых функций $f_3 = x_1 x_2$, $f_5 = x_1 !x_2$, $f_7 = 0$, $f_8 = x_2$.

Построим интерполяционную таблицу для этой системы (табл. 19.3).

Т а б л и ц а 19.3

X	x_1	x_2	f_3	f_5	f_7	f_8	$Y = p^0 Y$	$p^1 Y$	$p^2 Y$	$p^3 Y$
0	0	0	0	0	0	0	0	1		
1	0	1	0	0	0	1	1		2	
2	1	0	0	1	0	0	4		2	
3	1	1	1	0	0	1	9			0

Из этой таблицы следует, что

$$Y(X) = 1 \cdot 0 + \frac{X}{1!} \cdot 1 + \frac{X(X-1)}{2!} \cdot 2 + \frac{X(X-1)(X-2)}{3!} \cdot 0 = X^2.$$

Покажем, что изменение порядка расположения функций в системе может изменить сложность ИП. Например, если вместо кортежа функций $f_3 f_5 f_7 f_8$ реализовать кортеж $f_8 f_7 f_5 f_3$, то, построив интерполяционную таблицу, получим следующее выражение:

$$Y(X) = \frac{X}{6}(27X^2 - 123X + 144).$$

Покажем также, что для упрощения ИП в заданную систему БФУ могут вводиться дополнительные функции и использоваться операция маскирования.

Выше было показано, что функция $f_5(X)$ непосредственно реализуется ИП третьей степени, в то время как из рассмотрения табл. 19.3 следует, что

$$f_5(X) = r_4^3(\text{bin}(X^2)),$$

где r_j^i — выделение i -го разряда в j разрядном двоичном числе; $\text{bin } a$ — операция преобразования десятичного числа « a » в его двоичный эквивалент.

Таким образом, изложенный подход характеризуется тем, что:

- ИП строится как функция от входного вектора X в целом, а не от отдельных его компонент;
- имеется возможность параллельного вычисления систем БФУ;
- в ряде случаев сложность ИП для системы БФУ ниже сложности ИП отдельных функций, входящих в эту систему;
- имеется возможность изменения сложности ИП за счет изменения порядка расположения функций в системе;
- имеется возможность изменения сложности ИП за счет введения в систему дополнительных функций и применения операции маскирования. ИП в этом случае могут быть названы интерполяционными полиномами с маскированием.

19.3.2. Реализация не полностью определенных булевых функций

Теорема 19.1. Если система БФУ определена на g ($g \leq 2^n$) наборах входных переменных $x_1, x_2, \dots, x_n$, то ее всегда можно реализовать ИП с маскированием, содержащим не более g ненулевых коэффициентов $p^k Y$ (не более g членов в ИП).

Доказательство этой теоремы базируется на том факте, что каждое неопределенное значение системы БФУ можно выбрать таким, чтобы соответствующее приращение было равно нулю.

Пример 19.5. Построить ИП для БФУ $f(x_1, x_2) = |0 \ 1 \ z \ 1|^T$, где z — неопределенное значение функции.

Построим ИТ для рассматриваемой БФУ (табл. 19.4).

Т а б л и ц а 19.4

X	x_1	x_2	f	$Y = p^0 Y$	$p^1 Y$	$p^2 Y$	$p^3 Y$
0	0	0	0	0			
				→	1		
1	0	1	1	1	→	$z - 2$	
				→	$z - 1$	→	$4 - 3z$
2	1	0	z	z	→	$2 - 2z$	
				→	$1 - z$		
3	1	1	1	1			

Таким образом,

$$p^0 Y = 0, \quad p^1 Y = 1, \quad p^2 Y = z - 2, \quad p^3 Y = 4 - 3z.$$

При $z = 0$

$$p^0 Y = 0, \quad p^1 Y = 1, \quad p^2 Y = -2, \quad p^3 Y = 4;$$

$$f = x_2, \quad f(X) = \frac{X}{3}(2X^2 - 9X + 10).$$

При $z = 1$

$$p^0 Y = 0, \quad p^1 Y = 1, \quad p^2 Y = -1, \quad p^3 Y = 1;$$

$$f = x_1 \vee x_2, \quad f(X) = \frac{X}{6}(X^2 - 6X + 11).$$

При $z = 2$

$$p^0 Y = 0, \quad p^1 Y = 1, \quad p^2 Y = 0, \quad p^3 Y = -2;$$

$$f(X) = -\frac{X}{3}(X^2 - 3X - 1), \quad f = r_2^1(\text{bin } f(X)) = x_2.$$

19.3.3. Реализация многозначных логических функций

Пример 19.6. Построить ИП для троичной логической функции одной переменной, заданной табл. 19.5.

Т а б л и ц а 19.5

X	f	$Y = p^0 Y$	$p^1 Y$	$p^2 Y$
0	2	2		
		→	-1	
1	1	1	→	2
		→	1	
2	2	2		

Из этой таблицы следует, что

$$f(X) = 1 \cdot 2 - X \cdot 1 + \frac{X(X-1)}{2} \cdot 2 = X^2 - 2X + 2.$$

Таким образом, рассмотренный подход для некоторых функций более эффективен как по трудоемкости, так и по сложности полино-

ма по сравнению с быстрым преобразованием Фурье [7]. Для данного примера это преобразование строит выражение вида:

$$f(X) = \text{mod}_3(X^2 + X + 2).$$

Для некоторых других функций предложенный подход бывает крайне неэффективным.

Пример 19.7. Построить ИП для троичной логической функции двух переменных, заданной табл. 19.6.

Т а б л и ц а 19.6

X	0	1	2	3	4	5	6	7	8
x_1	0	0	0	1	1	1	2	2	2
x_2	0	1	2	0	1	2	0	1	2
f	0	0	0	0	0	0	0	1	2

В этом случае $X = 3x_1 + x_0$, $p^0 Y = \dots = p^6 Y = 0$, $p^7 Y = 1$, $p^8 Y = -6$. Таким образом,

$$f(X) = \frac{\prod_{i=0}^6 (X-i)}{7!} \cdot 1 - \frac{\prod_{i=0}^7 (X-i)}{8!} \cdot 6 = (50 - 6X) \frac{\prod_{i=0}^6 (X-i)}{8!}.$$

Эта функция может быть вычислена «по интервалам» существенно проще:

```

if (X ≥ 7)
    { if (X = 8)  f = 2;
      else       f = 1; }
else f = 0.

```

Использовать ИП для вычисления булевых функций было предложено автором и В. Н. Вавиловым в [8].

19.4. Реализация булевых функций с помощью спектрального метода

Рассмотрим функцию Радемахера i -го порядка [9] при применении n входных булевых переменных:

$$R_n^i = (-1)^{x_i}, \quad R_n^0 = 1, \quad 1 \leq i \leq n.$$

Преобразуем эти соотношения в табличную форму для $n = 1, 2, 3$ (табл. 19.7—19.9 соответственно). Две последние таблицы содержат также строки, соответствующие всем возможным произведениям функций Радемахера, отличных от R_n^0 .

Т а б л и ц а 19.7

x	0	1
R_1^0	1	1
R_1^1	1	-1

Т а б л и ц а 19.8

x_1	0	0	1	1
x_2	0	1	0	1
R_2^0	1	1	1	1
R_2^1	1	1	-1	-1
R_2^2	1	-1	1	-1
$R_2^1 R_2^2$	1	-1	-1	1

Т а б л и ц а 19.9

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
R_3^0	1	1	1	1	1	1	1	1
R_3^1	1	1	1	1	-1	-1	-1	-1
R_3^2	1	1	-1	-1	1	1	-1	-1
$R_3^1 R_3^2$	1	1	-1	-1	-1	-1	1	1
R_3^3	1	-1	1	-1	1	-1	1	-1
$R_3^1 R_3^3$	1	-1	1	-1	-1	1	-1	1
$R_3^2 R_3^3$	1	-1	-1	1	1	-1	-1	1
$R_3^1 R_3^2 R_3^3$	1	-1	-1	1	-1	1	1	-1

Содержимое этих таблиц задает матрицы Уолша при $n = 1, 2, 3$.

В [10, 11] показано, что прямое преобразование (построение спектра S по многозначной функции Y , которая зависит от многозначной переменной X , принимающей значения от 0 до 2^n) определяется соотношением

$$|S| = \frac{1}{2^n} |W_n| |Y|,$$

где $|S|$, $|Y|$ — вектор-столбцы спектра и функции соответственно, а $|W_n|$ — матрица Уолша с размерностью $2^n \times 2^n$.

Используя это соотношение для двоичного одноразрядного сумматора, для которого $Y(X) = |0 \ 2 \ 2 \ 1 \ 2 \ 1 \ 1 \ 3|^T$, можно получить следующий спектр:

$$|\mathbf{S}| = \frac{1}{8} |12, -2, -2, 0, -2, 0, 0, -6|^T.$$

Обратное преобразование (построение функции Y по спектру S) выполняется на основе соотношения [10, 11]

$$|\mathbf{Y}| = |\mathbf{W}_n| |\mathbf{S}|.$$

Обозначая i -ю строку матрицы Уолша символом W_n^i , называемой функцией Уолша, для двоичного одноразрядного сумматора получим:

$$Y = \frac{1}{8} (12W_3^0 - 2W_3^1 - 2W_3^2 - 2W_3^4 - 6W_3^7).$$

Заменяя каждую функцию Уолша ее представлением через функции Радемахера, получим:

$$Y = \frac{1}{4} (6 - R_3^1 - R_3^2 - R_3^3 - 3R_3^1 R_3^2 R_3^3).$$

Для получения расчетного соотношения заменим функции Радемахера их выражениями:

$$Y = \frac{1}{4} (6 - (-1)^{x_1} - (-1)^{x_2} - (-1)^{x_3} - 3(-1)^{x_1 + x_2 + x_3}).$$

Это выражение с линейными показателями степени может применяться при программной реализации указанного сумматора на основе спектрального метода.

Более изящный подход в рамках ортогональных преобразований может быть предложен, если вместо матриц Уолша использовать матрицы Адамара, являющиеся рекурсивными.

Рассмотрим матрицу Адамара [12]

$$|\mathbf{H}_n| = \begin{vmatrix} \mathbf{H}_{n-1} & \mathbf{H}_{n-1} \\ \mathbf{H}_{n-1} & -\mathbf{H}_{n-1} \end{vmatrix},$$

которая является n -й кронекеровской степенью матрицы

$$|\mathbf{H}_1| = \begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix}.$$

Построим матрицы Адамара для $n = 2, 3$ и выразим их через соответствующие функции Радемахера

$$|\mathbf{H}_2| = \begin{vmatrix} H_2^0 \\ H_2^1 \\ H_2^2 \\ H_2^3 \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{vmatrix} = \begin{vmatrix} R_2^0 \\ R_2^2 \\ R_2^1 \\ R_2^1 R_2^2 \end{vmatrix};$$

$$|\mathbf{H}_3| = \begin{vmatrix} H_3^0 \\ H_3^1 \\ H_3^2 \\ H_3^3 \\ H_3^4 \\ H_3^5 \\ H_3^6 \\ H_3^7 \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{vmatrix} = \begin{vmatrix} R_3^0 \\ R_3^3 \\ R_3^2 \\ R_3^2 R_3^3 \\ R_3^1 \\ R_3^1 R_3^3 \\ R_3^1 R_3^2 \\ R_3^1 R_3^2 R_3^3 \end{vmatrix}.$$

Рассмотрим соотношение

$$|\mathbf{Y}| = |\mathbf{H}_n| |\mathbf{S}|. \quad (19.10)$$

Тогда

$$|\mathbf{S}| = |\mathbf{H}_n|^{-1} |\mathbf{Y}|.$$

Для матриц Адамара справедливо равенство

$$|\mathbf{H}_n|^{-1} = \frac{1}{2^n} |\mathbf{H}_n|.$$

Таким образом,

$$|\mathbf{S}| = \frac{1}{2^n} |\mathbf{H}_n| |\mathbf{Y}|. \quad (19.11)$$

При этом S называется спектром функции Y по ортогональному базису H , соотношение (19.11) — прямым преобразованием Уолша—Адамара, а соотношение (19.10) — обратным преобразованием.

Пример 19.9. Реализовать двоичный одноразрядный сумматор

$$Y(x_1, x_2, x_3) = |0 \ 1 \ 1 \ 2 \ 1 \ 2 \ 2 \ 3|^T.$$

1. Используя соотношение (19.11), определим спектральные коэффициенты:

$$|\mathbf{S}| = \frac{1}{8} |\mathbf{H}_3| \quad |\mathbf{Y}| = \frac{1}{8} |12, -4, -4, 0, -4, 0, 0, 0|^T.$$

2. Применяя соотношение (19.10), определим выражение для функции Y через функции Адамара:

$$\begin{aligned} Y &= \frac{1}{8} (12H_3^0 - 4H_3^1 - 4H_3^2 - 4H_3^4) = \\ &= \frac{1}{2} (3H_3^0 - (H_3^1 + H_3^2 + H_3^3)). \end{aligned}$$

3. Заменяем функции Адамара на функции Радемахера:

$$Y = \frac{1}{2} (3R_3^0 - (R_3^3 + R_3^2 + R_3^1)).$$

4. Запишем функции Радемахера в явной форме:

$$Y = \frac{1}{2} (3 - ((-1)^{x_3} + (-1)^{x_2} + (-1)^{x_1})).$$

Обратим внимание, что это выражение существенно проще, чем при использовании матриц Уолша.

Пример 19.10. Реализовать функцию «голосование два и более из трех»

$$f_2(x_1, x_2, x_3) = |0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1|^T.$$

1. Вычислим значения спектральных коэффициентов этой функции:

$$|\mathbf{S}_2| = \frac{1}{8} |4, -2, -2, 0, -2, 0, 0, 2|^T.$$

2. Запишем выражение для функции f_2 через функции Адамара:

$$f_2 = \frac{1}{8} (4H_3^0 - 2H_3^1 - 2H_3^2 - 2H_3^4 + 2H_3^7).$$

3. Запишем выражение для функции f_2 через функции Радемахера:

$$f_2 = \frac{1}{4} (2 - (R_3^3 + R_3^2 + R_3^1 - R_3^1 R_3^2 R_3^3)).$$

4. При этом расчетное соотношение для вычисления функции f_2 имеет вид:

$$f_2 = \frac{1}{4} [2 - ((-1)^{x_3} + (-1)^{x_2} + (-1)^{x_1} - (-1)^{x_1+x_2+x_3})].$$

Из сравнения результатов примеров 19.9 и 19.10 следует, что реализация двоичного одноразрядного сумматора, образованного двумя булевыми функциями («голосование два и более из трех» и «нечетность трех переменных»), проще, чем реализация первой из них.

Поэтому может быть предложен следующий метод реализации БФУ:

— к заданной БФУ добавляются дополнительные булевы функции;

— полученная система БФУ реализуется арифметическим выражением с помощью спектрального метода;

— применяется операция маскирования для получения значений построенной системы БФУ.

Используя такой подход, функция «голосование два и более из трех» может быть реализована следующим образом:

$$f_2 = r_2^2 \left\{ \text{bin} \left[\frac{1}{2} (3 - ((-1)^{x_3} + (-1)^{x_2} + (-1)^{x_1})) \right] \right\}.$$

Пример 19.11. Реализовать функцию «нечетность трех переменных»

$$f_1(x_1, x_2, x_3) = |0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1|^T.$$

1. Вычислим значения спектральных коэффициентов этой функции:

$$|S_1| = \frac{1}{8} |4, 0, 0, 0, 0, 0, 0, -1|^T.$$

2. Запишем выражение для функции f_1 через функции Адамара:

$$f_1 = \frac{1}{8} (4H_3^0 - 2H_3^7).$$

3. Запишем выражение для функции f_1 через функции Радемахера:

$$f_1 = \frac{1}{2} (1 - R_3^1 R_3^2 R_3^3).$$

4. При этом расчетное соотношение для вычисления функции f_1 имеет вид:

$$f_1 = \frac{1}{2} \left(1 - (-1)^{x_1 + x_2 + x_3} \right).$$

Пример 19.12. Построить арифметическое выражение для двоичного одноразрядного сумматора, применяя найденные в предыдущих примерах арифметические выражения для функций f_2 и f_1 .

Это выражение строится следующим образом:

$$Y = 2f_2 + f_1 = \frac{1}{2} \left[3 - ((-1)^{x_3} + (-1)^{x_2} + (-1)^{x_1}) \right].$$

Из последнего примера следует, что сложность арифметического выражения зависит от порядка расположения функций в кортеже функций.

Приведем спектры для неповторных формул в базисе $\{\&, \vee, !\}$ из двух и трех букв:

$$\begin{aligned} f_3 = x_1 x_2 & \quad |S_3| = \frac{1}{4} |1, -1, -1, 1|^T; \\ f_4 = x_1 \vee x_2 & \quad |S_4| = \frac{1}{4} |3, -1, -1, -1|^T; \\ f_5 = x_1 x_2 x_3 & \quad |S_5| = \frac{1}{8} |1, -1, -1, 1, -1, 1, 1, -1|^T; \\ f_6 = x_1(x_2 \vee x_3) & \quad |S_6| = \frac{1}{8} |3, -1, -1, -1, -3, 1, 1, 1|^T; \\ f_7 = x_1 \vee x_2 x_3 & \quad |S_7| = \frac{1}{8} |5, -1, -1, 1, -3, -1, -1, 1|^T; \\ f_8 = x_1 \vee x_2 \vee x_3 & \quad |S_8| = \frac{1}{8} |7, -1, -1, -1, -1, -1, -1, -1|^T. \end{aligned}$$

Из рассмотренных примеров следует, что простые (бесповторные) формулы в булевом базисе реализуются арифметическими выражениями весьма сложно, в то время как сложные (повторные) формулы могут быть реализованы сравнительно просто.

Рассмотренный класс арифметических выражений характеризуется тем, что входные переменные входят в них линейно. При программировании возведение в степень величины « -1 » может быть заменено проверкой четности показателя степени, командой условного перехода и командами присвоения переменным значений « -1 » и « 1 ».

Изложенный подход, основанный на использовании матриц Адамара, был предложен в [8]. Другие методы построения арифметических выражений на основе ортогональных преобразований рассмотрены в [13, 14]. Этой тематике посвящены также работы [15—23].

Выводы

1. Рассмотрен метод вычисления булевых функций, называемый методом логических шкал.
2. Предложены методы построения пороговых реализаций для неповторных пороговых формул.
3. Предложен метод представления булевых формул линейными арифметическими выражениями, использующими операции округления до ближайшего целого.
4. Предложен метод представления булевых функций интерполяционными полиномами.
5. Предложен метод построения арифметических выражений с линейными показателями степени на основе матриц Адамара.

Л и т е р а т у р а

1. Бутаков Е. А. Методы синтеза релейных устройств из пороговых элементов. М.: Энергия, 1970.
2. Боголюбов И. Н., Овсевич Б. Л., Розенблум Л. Я. Пороговая и мажоритарная логика. Информ. листок. 1964. № 2 (29).
3. Кондратьев В. Н., Шалыто А. А. Реализация булевых функций одним линейным арифметическим полиномом с маскированием // Автоматика и телемеханика. 1996. № 1.
4. Артюхов В. Л., Розенблум Л. Я., Шалыто А. А. Логические возможности некоторых типов каскадных структур // Сети связи и дискретные устройства управления. М.: Наука, 1976.
5. Задорожный В. И. Реализация логических функций арифметическими выражениями // Автоматизация анализа и синтеза структур ЭВМ и вычислительных алгоритмов. Новосибирск: НИСИ, 1978.
6. Бронштейн И. Н., Семендяев К. А. Справочник по математике. М.: Наука, 1986.
7. Лабунец В. Г., Ситников О. П. Использование быстрого многомерного преобразования Фурье для полиномиального представления функций многозначной логики // Изв. АН СССР. Техн. кибернетика. 1978. № 5.
8. Шалыто А. А. Реализация алгоритмов судовых управляющих логических систем при использовании микропроцессорной техники. Л.: ИПК СП, 1988.
9. Ефремов В. Д., Кузьмин А. А., Степанов В. А. Вычисление логических функций с использованием преобразования Радемахера // Автоматика и телемеханика. 1984. № 2.
10. Карповский М. Г., Москалев Э. С. Спектральные методы анализа и синтеза дискретных устройств. Л.: Энергия, 1973.
11. Karpovsky M. G. Finite orthogonal series in the design of digital devices. New York: John Willey & Sons, 1976.
12. Трахман А. М., Трахман В. А. Основы дискретных сигналов на конечных интервалах. М.: Сов. радио, 1975.
13. Артюхов В. Л., Кондратьев В. Н., Шалыто А. А. Реализация булевых функций арифметическими полиномами // Автоматика и телемеханика. 1988. № 4.
14. Шмерко В. П. Синтез арифметических форм булевых функций посредством преобразования Фурье // Автоматика и телемеханика. 1989. № 5.
15. Muller D. E. Application of boolean algebra to switching circuits design and to error detection // IRE Trans. Electron. Comp. 1954. Vol. EC-3.

16. *Hurst S. L.* Logical processing of digital signals. London; Basel: Crane Russak and Edward Arnold, 1978.
17. *Hurst S. L., Miller D. M., Muzio J. C.* Spectral techniques in digital logic. Bristol: Academic Press, 1985.
18. *Muzio J. C., Wesselkamper T. C.* Multiple-valued switching theory. Bristol: Adam Hilger, 1986.
19. *Sasao T.* Logic design and optimization. MA: Kluwer Academic Publisher, 1993.
20. *Recent* developments in abstract harmonic analysis with applications in signal processing / Eds R. S. Stankovic, M. R. Stojic, M. S. Stankovic. Belgrad: Nauka, 1995.
21. *Sasao T.* Logic design: switching circuit theory. Kindai: Kadaku, 1995.
22. *Representation* of discrete functions / Eds T. Sasao, M. Fujita. MA: Kluwer Academic Publisher, 1996.
23. *Stankovic R. S.* Spectral transform decision diagrams. Belgrad: Nauka, 1998.

Глава 20

Реализация булевых функций с помощью арифметических полиномов

20.1. Реализация булевых функций арифметическими полиномами

20.1.1. Реализация одной булевой формулы

Булева формула (БФ) в базисе $\{\&, \vee, !\}$ преобразуется в арифметический полином (АП) следующим образом:

- в БФ ортогонализуются дизъюнкции;
- знаки $\vee$ заменяются на знаки $+$;
- символы $!x_i$ заменяются выражением $1 - x_i$;
- раскрываются скобки и приводятся подобные члены, если они имеются.

Пример 20.1. Построить АП для БФ $f = (x_1 \vee x_2)x_3 \vee x_1x_2$.

Для упрощения ортогонализированной формулы переставим подформулы в заданной БФ:

$$\begin{aligned} f &= x_1x_2 \vee (x_1x_2 \vee x_3) = x_1x_2 \vee !(x_1x_2)(x_1 \vee !x_1x_2)x_3 = \\ &= x_1x_2 \vee (!x_1 \vee x_1!x_2)(x_1 \vee !x_1x_2)x_3 = \\ &= x_1x_2 \vee (!x_1x_2 \vee x_1!x_2)x_3 = \\ &= x_1x_2 \vee ((1 - x_1)x_2 + x_1(1 - x_2))x_3 = \\ &= x_1x_2 + x_1x_3 + x_2x_3 + 2x_1x_2x_3. \end{aligned}$$

Совместное использование рассмотренного подхода и подхода, изложенного в разд. 19.2.4, позволяет получить для заданной БФ следующие арифметические выражения:

$$\begin{aligned}
f &= \left[\frac{x_1 + x_2}{2} \right] + \left[\frac{x_1 + x_3}{2} \right] + \left[\frac{x_2 + x_3}{2} \right] - 2 \left[\frac{x_1 + x_2 + x_3}{3} \right] = \\
&= \left[\frac{x_1 + x_2}{2} \right] + \left[\frac{x_1 + x_3}{2} \right] + \left[\frac{x_2 + x_3}{2} \right] - 2 \left[\frac{\left[\frac{x_1 + x_2}{2} \right] + x_3}{3} \right] = \\
&= \left[\frac{x_1 + x_2}{2} \right] + \left[\frac{x_1 + x_3}{2} \right] + \left[\frac{x_2 + x_3}{2} \right] - 2 \left[\frac{x_1 + x_2 + x_3 + 1}{4} \right].
\end{aligned}$$

Приведем без доказательства два утверждения [1].

1. Если заданная БФ бесповторна в указанном базисе, то коэффициенты АП равны -1, 0, 1. Обратное утверждение неверно.

2. Если коэффициенты АП равны -1, 0, 1, то соответствующий полином Жегалкина может быть найден по арифметическому полиному заменой знаков плюс и минус на знак $\oplus$.

20.1.2. Реализация одной симметрической функции

Метод, изложенный в предыдущем разделе, применим для произвольных булевых функций (БФУ), в том числе и симметрических функций (СФ), что, в частности, продемонстрировано в примере 20.1.

Однако специфика СФ позволяет предложить более простой метод для их реализации. Это объясняется тем, что если произвольная БФУ n переменных реализуется арифметическим полиномом, содержащим до 2^n коэффициентов a_i , то произвольная СФ может быть реализована АП, содержащим только $n + 1$ коэффициентов b_j . В этих полиномах a_i и b_j — произвольные целые числа.

Например, при $n = 4$ арифметический полином для произвольной БФУ имеет вид:

$$\begin{aligned}
f &= a_0 + a_1 x_4 + a_2 x_3 + a_3 x_3 x_4 + a_4 x_2 + a_5 x_2 x_4 + a_6 x_2 x_3 + \\
&+ a_7 x_2 x_3 x_4 + a_8 x_1 + a_9 x_1 x_4 + a_{10} x_1 x_3 + a_{11} x_1 x_3 x_4 + a_{12} x_1 x_2 + \\
&+ a_{13} x_1 x_2 x_4 + a_{14} x_1 x_2 x_3 + a_{15} x_1 x_2 x_3 x_4,
\end{aligned}$$

в то время как АП для произвольной СФ может быть представлен в виде:

$$\begin{aligned}
f &= b_0 + b_1(x_1 + x_2 + x_3 + x_4) + \\
&+ b_2(x_1 x_2 + x_1 x_3 + x_1 x_4 + x_2 x_3 + x_2 x_4 + x_3 x_4) + \\
&+ b_3(x_1 x_2 x_3 + x_1 x_2 x_4 + x_1 x_3 x_4 + x_2 x_3 x_4) + b_4 x_1 x_2 x_3 x_4.
\end{aligned}$$

Запишем последнее соотношение более компактно:

$$f = b_0 + b_1 X_1^4 + b_2 X_2^4 + b_3 X_3^4 + b_4 X_4^4.$$

Из приведенных соотношений следует, что

$$b_0 = a_0; b_1 = a_1 = a_2 = a_4 = a_8; b_2 = a_3 = a_5 = a_6 = a_9 = a_{10} = a_{12};$$

$$b_3 = a_7 = a_{11} = a_{13} = a_{15}; b_4 = a_{15};$$

$$X_1^4 = x_1 + x_2 + x_3 + x_4;$$

$$X_2^4 = x_1 x_2 + x_1 x_3 + x_1 x_4 + x_2 x_3 + x_2 x_4 + x_3 x_4;$$

$$X_3^4 = x_1 x_2 x_3 + x_1 x_2 x_4 + x_1 x_3 x_4 + x_2 x_3 x_4;$$

$$X_4^4 = x_1 x_2 x_3 x_4.$$

Сформулируем задачу нахождения арифметического полинома для симметрической функции следующим образом: для симметрической реализации $S_{j, \dots, l}^n$ построить АП, который обозначим символом $F_{j, \dots, l}^n$.

Обозначим символами $F_j^n, \dots, F_l^n$ арифметические полиномы для элементарных симметрических функций $S_j^n, \dots, S_l^n$ соответственно.

При этом для $n = 4$

$$S_0^4 = !x_1!x_2!x_3!x_4;$$

$$S_1^4 = x_1!x_2!x_3!x_4 \vee !x_1x_2!x_3!x_4 \vee !x_1!x_2x_3!x_4 \vee !x_1!x_2!x_3x_4;$$

$$S_2^4 = x_1x_2!x_3!x_4 \vee x_1!x_2x_3!x_4 \vee x_1!x_2!x_3x_4 \vee !x_1x_2x_3!x_4 \vee \\ \vee !x_1x_2!x_3x_4 \vee !x_1!x_2x_3x_4;$$

$$S_3^4 = x_1x_2x_3!x_4 \vee x_1x_2!x_3x_4 \vee x_1!x_2x_3x_4 \vee !x_1x_2x_3x_4;$$

$$S_4^4 = x_1x_2x_3x_4.$$

Так как

$$S_{j, \dots, l}^n = S_j^n \vee \dots \vee S_l^n,$$

то в силу ортогональности элементарных СФ между собой будем использовать соотношение

$$F_{j, \dots, l}^n = F_j^n + \dots + F_l^n.$$

Введем операторное представление для АП рассматриваемого класса.

Сопоставим полиному F_i^n оператор

$$O_i^n = C_n^i x^i (1 - x)^{n-i}, \quad (20.1)$$

где $i = 0, \dots, n$.

Сопоставим полиному $F_{j,\dots,l}^n$ оператор

$$O_{j,\dots,l}^n = O_j^n + \dots + O_l^n. \quad (20.2)$$

Введем правило для перехода от оператора к АП:

$$O(bx^i) \rightarrow F\left(\frac{b}{C_n^i} X_i^n\right). \quad (20.3)$$

На основе соотношения 20.1 перечислим в качестве примера все операторы O_i^n при $n = 4$:

$$\begin{aligned} O_0^4 &= C_4^0 x^0 (1-x)^4 = 1 - 4x + 6x^2 - 4x^3 + x^4; \\ O_1^4 &= C_4^1 x^1 (1-x)^3 = 4x - 12x^2 + 12x^3 - 4x^4; \\ O_2^4 &= C_4^2 x^2 (1-x)^2 = 6x^2 - 12x^3 + 6x^4; \\ O_3^4 &= C_4^3 x^3 (1-x)^1 = 4x^3 - 4x^4; \\ O_4^4 &= C_4^4 x^4 (1-x)^0 = x^4. \end{aligned} \quad (20.4)$$

Пример 20.2. Определить $F_{2,4}^4$ для $S_{2,4}^4$.

На основе соотношения (20.2) следует, что

$$O_{2,4}^4 = O_2^4 + O_4^4.$$

Поэтому из соотношений (20.4) следует, что

$$O_{2,4}^4 = 6x^2 - 12x^3 + 7x^4.$$

На основе соотношения (20.3)

$$F_{2,4}^4 = \frac{6}{C_4^2} X_2^4 - \frac{12}{C_4^3} X_3^4 + \frac{7}{C_4^4} X_4^4 = X_2^4 - 3X_3^4 + 7X_4^4.$$

Таким образом, в данном случае $b_0 = 0$; $b_1 = 0$; $b_2 = 1$; $b_3 = -3$; $b_4 = 7$.

Пример 20.3. Определить $F_{2,3}^3$ для $S_{2,3}^3$.

На основе соотношения (20.1) следует, что

$$\begin{aligned} O_2^3 &= C_3^2 x^2 (1-x)^1 = 3x^2 - 3x^3; \\ O_3^3 &= C_3^3 x^3 (1-x)^0 = x^3. \end{aligned} \quad (20.5)$$

На основе соотношения (20.2) получим

$$O_{2,3}^3 = O_2^3 + O_3^3.$$

Из соотношения (20.5) следует, что

$$O_{2,3}^3 = 3x^2 - 2x^3.$$

На основе соотношения (20.3) получим

$$F_{2,3}^3 = \frac{3}{C_3^2} X_2^3 - \frac{2}{C_3^3} X_3^3 = X_2^3 - 2X_3^3,$$

где $X_2^3 = x_1 x_2 + x_1 x_3 + x_2 x_3$; $X_3^3 = x_1 x_2 x_3$.

Метод предложен В. Л. Артюховым и автором в [2].

20.1.3. Реализация одной булевой формулы с использованием операции «sign»

Пусть задана нормальная БФ в базисе $\{\&, \vee, !\}$, которую обозначим символом f . Выполним в этой формуле следующие замены: $!x_i$ на $1 - x_i$; $\&$ на $*$; $\vee$ на $+$.

В результате получим арифметическое выражение, которое обозначим символом f_1 . При этом

$$f = \text{sign } f_1,$$

где

$$\text{sign } f_1 = \begin{cases} 1 & \text{при } f_1 > 0; \\ 0 & \text{при } f_1 = 0. \end{cases}$$

Пример 20.4. Реализовать с помощью предлагаемого подхода булеву формулу $f = x_1 \vee !x_2 x_3$.

Из изложенного следует, что

$$f = \text{sign} (x_1 + (1 - x_2)x_3) = \text{sign} (x_1 + x_3 - x_2 x_3).$$

Полученное выражение может быть реализовано следующим образом:

$$\begin{aligned} f &= 0; \\ \text{if } (x_1 + x_3 - x_2 x_3 > 0) \quad f &= 1. \end{aligned}$$

Отметим, что АП для данной функции существенно более сложен:

$$f = x_1 + x_3 - x_2 x_3 - x_1 x_3 + x_1 x_2 x_3.$$

Так как заданная БФУ является пороговой, то она может быть реализована также и следующим образом:

$$f = 0; \\ \text{if } (2x_1 - x_2 + x_3 \geq 0) \text{ then } f = 1.$$

Описанный в настоящем разделе подход был предложен М. Х. Сахартовым и автором [3].

20.1.4. Реализация одной булевой формулы с использованием операции «абсолютная величина»

Построим арифметический полином для БФ $f = x_1 \oplus x_2$. Исходя из изложенного в разд. 20.1.1, получим:

$$f = x_1 \oplus x_2 = !x_1 x_2 \vee x_1 !x_2 = \\ = (1 - x_1)x_2 + x_1(1 - x_2) = x_1 - 2x_1 x_2 + x_2.$$

Проведем дальнейшее преобразование этого выражения

$$f = x_1^2 - 2x_1 x_2 + x_2^2 = (x_1 - x_2)^2 = |x_1 - x_2|.$$

Из рассмотренного примера следует, что если формулы f_1 и f_2 являются конъюнкциями, то

$$f_1 \oplus f_2 = |f_1 - f_2|.$$

Если эти формулы конъюнкциями не являются, то

$$f_1 \oplus f_2 = |f'_1 - f'_2|,$$

где f'_1 и f'_2 — АП функций f_1 и f_2 соответственно.

Пример 20.5. Построить арифметическое выражение для булевой формулы $f = (x_1 \vee x_2) \oplus x_3$.

Из изложенного следует, что $f = |x_1 + x_2 - x_1 x_2 - x_3|$. Это выражение существенно проще, чем соответствующий АП.

Построим АП, используя модификацию разложения Шеннона по переменной x_1 и приведенные выше соотношения

$$f = (1 - x_1)|x_2 - x_3| + x_1|1 - x_3| = \\ = (1 - x_1)(x_2 + x_3 - 2x_2 x_3) + x_1(1 - x_3) = \\ = x_1 + x_2 + x_3 - x_1 x_2 - 2x_1 x_3 - 2x_2 x_3 + 2x_1 x_2 x_3.$$

Представление БФ с помощью операции «абсолютная величина» не является единственным. Покажем это на рассмотренном примере. Для этого ортогонализуем заданную БФ: $f = (x_1 \vee !x_1 x_2) \oplus x_3$. В этом выражении могут быть сделаны следующие замены: $!x$ на $1 - x$; $\vee$ на $\oplus$. При этом

$$f = (x_1 \oplus (1 \oplus x_1)x_2) \oplus x_3 = (x_1 \oplus x_2 \oplus x_1 x_2) \oplus x_3.$$

Из изложенного следует, что

$$f = ||x_1 - x_2| - x_1 x_2| - x_3|.$$

В данном случае можно получить и другое выражение:

$$f = |x_1 + x_2 - 3x_1 x_3| - x_3|.$$

Совместное применение подходов, изложенных в настоящем разделе и в разд. 19.2.4, позволяет получить для заданной БФ следующие арифметические выражения, которые не содержат нелинейных членов:

$$f = \left| |x_1 - x_2| - \left\lfloor \frac{x_1 + x_2}{2} \right\rfloor - x_3 \right| = \left| x_1 - x_2 - 3 \left\lfloor \frac{x_1 + x_2}{2} \right\rfloor - x_3 \right|.$$

В заключение раздела получим еще несколько соотношений, использующих операцию «абсолютная величина»:

$$\begin{aligned} x_1 \vee x_2 &= x_1 + x_2 - x_1 x_2 = x_1 + x_2 - 2x_1 x_2 + x_1 x_2 = |x_1 - x_2| + x_1 x_2; \\ x_1 \vee x_2 &= x_1 + x_2 - x_1 x_2 = \frac{x_1 + x_2 - 2x_1 x_2 + x_1 + x_2}{2} = \\ &= \frac{x_1 + x_2 + |x_1 - x_2|}{2}. \end{aligned}$$

Из соотношения

$$x_1 + x_2 - 2x_1 x_2 = |x_1 - x_2|$$

следует, что

$$x_1 x_2 = \frac{x_1 + x_2 - |x_1 - x_2|}{2}.$$

Изложенный подход описан в [1, 4].

20.1.5. Реализация системы булевых функций одним арифметическим полиномом

Введем понятие «обобщенная совершенная дизъюнктивная нормальная форма» (ОСДНФ) для системы булевых функций (СБФУ).

Под ОСДНФ будем понимать выражение вида:

$$Y = y_0!x_1!x_2 \dots!x_n \vee y_1!x_1!x_2 \dots!x_{n-1}x_n \vee \dots \vee y_{2^n-1}x_1x_2 \dots x_n, \quad (20.6)$$

где y_i — десятичный эквивалент значений СБФУ на наборе i .

Справедливость этого соотношения следует из ортогональности конъюнкций в ОСДНФ.

Арифметический полином для ОСДНФ будем искать в виде

$$Y = a_0 + a_1x_n + a_2x_{n-1} + a_3x_{n-1}x_n + \dots + a_{2^n-1}x_1x_2 \dots x_n. \quad (20.7)$$

Выражение (20.7) получается из выражения (20.6) заменой коэффициентов y_i на коэффициенты a_i и исключением переменных с инверсиями.

Подставляя 2^n значений переменных $x_1, \dots, x_n$ и приравнявая правые части соотношений (20.6) и (20.7), получим 2^n равенств, из которых могут быть найдены выражения для коэффициентов арифметического полинома (АП) через коэффициенты ОСДНФ.

Эти равенства могут быть представлены в матричной форме

$$|\mathbf{A}| = |\mathbf{P}_n| |\mathbf{Y}|, \quad (20.8)$$

где $|\mathbf{A}|$, $|\mathbf{Y}|$ — вектор-столбцы коэффициентов АП и ОСДНФ соответственно.

Это соотношение определяет переход от ОСДНФ (и тем самым от СБФУ) к АП. Для матрицы $|\mathbf{P}_n|$ выполняется следующее соотношение:

$$|\mathbf{P}_n| = \begin{vmatrix} P_{n-1} & 0 \\ -P_{n-1} & P_{n-1} \end{vmatrix}.$$

Эта матрица является n -й кронекеровской степенью матрицы

$$|\mathbf{P}_1| = \begin{vmatrix} 1 & 0 \\ -1 & 1 \end{vmatrix}.$$

При этом отметим, что матрица $|-P_{n-1}|$ получается из матрицы $|\mathbf{P}_{n-1}|$ путем инвертирования знаков единичных элементов.

При $n = 2$ справедливо соотношение

$$\begin{vmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 1 & -1 & -1 & 1 \end{vmatrix} \begin{vmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{vmatrix} = \begin{vmatrix} y_0 \\ y_1 - y_0 \\ y_2 - y_0 \\ y_0 - y_1 - y_2 + y_3 \end{vmatrix}. \quad (20.9)$$

Получающийся на основе соотношений (20.7) и (20.8) арифметический полином является единственным и безызбыточным: каждый столбец в двоичном представлении результатов вычислений по этому полиному реализует одну из заданных функций.

Пример 20.6. Построить АП для системы булевых формул $f_1 = x_1 x_2$, $f_2 = x_1 \oplus x_2$.

По этой системе может быть записано соотношение

$$Y = f_1 f_2 = |y_0 \ y_1 \ y_2 \ y_3|^T = |0 \ 1 \ 1 \ 2|^T.$$

Так как в этом случае $n = 2$, то

$$\begin{aligned} Y &= 0! x_1! x_2 \vee 1! x_1 x_2 \vee 1 x_1! x_2 \vee 2 x_1 x_2; \\ Y &= a_0 + a_1 x_2 + a_2 x_1 + a_3 x_1 x_2. \end{aligned}$$

Используя соотношение (20.9), определим, что $a_0 = a_3 = 0$, $a_1 = a_2 = 1$.

Таким образом, $Y = x_1 + x_2$, и поэтому $f_1 f_2 = \text{bin}(x_1 + x_2)$.

Приведем без доказательства теорему [1].

Т е о р е м а 20.1. Если система булевых функций определена на g входных наборах, то всегда можно построить АП, содержащий не более g ненулевых коэффициентов.

Пример 20.7. Построить АП для не полностью определенной системы булевых функций

$$Y(x_1, x_2) = |1 \ 2 \ 3 \ -|^T.$$

Из соотношения (20.9) следует, что, для того чтобы выполнялось соотношение $a_3 = y_0 - y_1 - y_2 + y_3 = 0$, должно быть выполнено соотношение $y_3 = 4$.

При этом $Y = 1 + x_2 + 2 x_1$, $f_1 = x_1 x_2$, $f_2 = x_1 \oplus x_2$, $f_3 = !x_2$.

Обратный переход (от АП к ОСДНФ и тем самым к СБФУ) осуществляется на основе соотношения:

$$|Y| = |Q_n| |A|, \quad (20.10)$$

где $|Q_n|$ — n -я кронекеровская степень матрицы

$$|\mathbf{Q}_1| = \begin{vmatrix} 1 & 0 \\ 1 & 1 \end{vmatrix}.$$

В заключение раздела отметим, что изложенный подход, естественно, применим и для отдельных БФУ, который, однако, не столь эффективен, как при его использовании для систем БФУ, некоторые из которых могут быть реализованы линейными АП (ЛАП), в то время как отдельные булевы функции при $n \geq 2$ линейными арифметическими полиномами не реализуются (разд. 20.3.1).

При этом отметим, что вычисление СБФУ по арифметическому полиному обеспечивает ее параллельную реализацию.

Алгебраический метод построения АП для системы БФ был предложен в [5]. Матричный метод построения АП для СБФУ предложен в [1] и независимо в [6, 7], что и было отмечено в [1].

20.1.6. Вычисление коэффициентов арифметического полинома на основе метода Гуда

Оценим количество действий в алгоритме, построенном на основе соотношения (20.8).

Матрица $|\mathbf{P}_n|$ содержит 3^n единиц. Поэтому матрица, получаемая в результате умножения матриц $|\mathbf{P}_n|$ и $|\mathbf{Y}|$, будет содержать 3^n элементов y_i . Количество действий, выполняемых в каждой из 2^n строк этой матрицы, на единицу меньше числа единиц в соответствующей строке матрицы $|\mathbf{P}_n|$. При этом количество действий в преобразовании

$$D = 3^n - 2^n.$$

Количество действий можно уменьшить, выполнив факторизацию матрицы $|\mathbf{P}_n|$ по методу Гуда [8]. Матрица $|\mathbf{P}_n|$ может быть записана в виде произведения n слабозаполненных матриц порядка 2^n , имеющих вид:

$$|\mathbf{L}_n| = \begin{vmatrix} 1 & 0 & 0 & 0 & - & - & - & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & - & - & - & 0 & 0 & 0 & 0 \\ - & - & - & - & - & - & - & - & - & - & - \\ 0 & 0 & 0 & 0 & - & - & - & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & - & - & - & 0 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 & - & - & - & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & - & - & - & 0 & 0 & 0 & 0 \\ - & - & - & - & - & - & - & - & - & - & - \\ 0 & 0 & 0 & 0 & - & - & - & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & - & - & - & 0 & 0 & -1 & 1 \end{vmatrix}.$$

При этом соотношение (20.8) приобретает вид:

$$|\mathbf{A}| = |\mathbf{L}_n| |\mathbf{L}_n| \dots |\mathbf{L}_n| |\mathbf{Y}|. \quad (20.11)$$

В матрицах Гуда в верхней половине каждая строка содержит по одной единице, а в нижней половине — две. Следовательно, для верхней половины не требуется действий, а для обработки каждой строки нижней половины требуется по одному действию. Так как нижняя половина каждой матрицы Гуда содержит 2^{n-1} строк, а вся процедура преобразования требует n матриц Гуда, то количество действий в этом случае

$$D_1 = n \cdot 2^{n-1}.$$

Так как $D_1 < D$, то преобразование (20.11) называется быстрым. Это преобразование было предложено в [7, 9].

Программы вычисления коэффициентов АП в соответствии с соотношениями (20.8) и (20.11), написанные на ассемблере MACRO-11 [10], приведены в [11].

20.1.7. Условие представимости системы булевых функций линейным арифметическим полиномом — условие линейности

Условие представимости СБФУ линейным АП, которое называется условием линейности, определяется из соотношения (20.8) приравниванием нулю коэффициентов в нелинейных членах соотношения (20.7) и представлением их через базовые значения y_i , где $i = 0, 1, 2, 4, \dots, 2^{n-1}$.

В [1] показано, что это условие имеет вид:

$$|\mathbf{Y}_k| = |\mathbf{D} \mathbf{C}| |\mathbf{Y}_b|, \quad (20.12)$$

где $|\mathbf{Y}_k|$ — вектор-столбец небазовых значений ОСДНФ; $|\mathbf{Y}_b|$ — вектор-столбец базовых значений ОСДНФ; $\mathbf{D}$ — подматрица, каждая строка которой равна двоичному представлению числа k , обозначаемому как $\text{bin } k$; $\mathbf{C}$ — вектор-столбец, каждое значение которого равно единице минус сумма единиц в $\text{bin } k$.

Для $n = 3$ условие линейности имеет вид:

$$\begin{pmatrix} y_3 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ 1 & 1 & 1 & -2 \end{pmatrix} \begin{pmatrix} y_4 \\ y_2 \\ y_1 \\ y_0 \end{pmatrix} = \begin{pmatrix} y_2 + y_1 - y_0 \\ y_4 + y_1 - y_0 \\ y_4 + y_2 - y_0 \\ y_4 + y_2 - y_1 - 2y_0 \end{pmatrix}. \quad (20.13)$$

Пример 20.8. Определить, представима ли система булевых формул $f_1 = x_1 x_2$; $f_2 = x_1 \vee x_2$; $f_3 = x_1 \oplus x_2$ с помощью линейного АП. Эта система формул может быть представлена в виде:

$$Y(x_1, x_2) = |0 \ 3 \ 3 \ 6|^T.$$

Условие линейности при $n = 2$ имеет вид:

$$|y_3| = |1 \ 1 \ -1| \begin{vmatrix} y_2 \\ y_1 \\ y_0 \end{vmatrix} = |y_2 + y_1 - y_0|.$$

Это условие для рассматриваемого примера выполняется. Из соотношения (20.9) следует, что

$$f_1 f_2 f_3 = \text{bin}(3x_1 + 3x_2).$$

20.1.8. Арифметические полиномы и полиномы Жегалкина

Возможность представления произвольной БФУ булевой формулой в базисе $\{0, 1, \oplus, \&\}$ была установлена Жегалкиным [12] и переоткрыта Ридом [13].

Полином Жегалкина (ПЖ) для произвольной БФУ n переменных, заданной СДНФ, может быть построен матричным методом, использующим матрицу $|Q_n|$ из соотношения (20.10). При этом коэффициенты ПЖ определяются с помощью соотношения

$$|G| = |Q_n| * |Y|. \quad (20.14)$$

Справедливо также и обратное преобразование (построение СДНФ по ПЖ), выполняемое на основе соотношения

$$|Y| = |Q_n| * |G|. \quad (20.15)$$

В этих выражениях применяются следующие обозначения: $|G|$, $|Y|$ — вектор-столбцы коэффициентов ПЖ и СДНФ соответственно; $*$ — символ операции умножения матриц, в которой арифметическая сумма заменяется суммой по модулю два [14].

Из соотношений (20.10) и (20.14) следует, что

$$|G| = |Q_n| * |Q_n| |A|,$$

а из соотношений (20.8) и (20.15) —

$$|A| = |P_n| |Q_n| * |G|.$$

Эти соотношения приведены в [1, 7, 9]. В [1] определено также условие линейности для ПЖ, которое имеет вид:

$$|\mathbf{Y}_k| = |\mathbf{D} \mathbf{B}| * |\mathbf{Y}_b|,$$

где $\mathbf{B}$ — вектор-столбец, каждое значение которого равно сумме по модулю два единицы и единиц в $\text{bin } k$.

При $n = 3$ это условие выражается соотношением вида:

$$\begin{vmatrix} y_3 \\ y_5 \\ y_6 \\ y_7 \end{vmatrix} = \begin{vmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{vmatrix} * \begin{vmatrix} y_4 \\ y_2 \\ y_1 \\ y_0 \end{vmatrix} = \begin{vmatrix} y_2 \oplus y_1 \oplus y_0 \\ y_4 \oplus y_1 \oplus y_0 \\ y_4 \oplus y_2 \oplus y_0 \\ y_4 \oplus y_2 \oplus y_1 \end{vmatrix}.$$

20.1.9. Арифметические полиномы и спектральные представления булевых функций

Переход от арифметических выражений, использующих функции Радемахера (разд. 19.4), к арифметическим полиномам может быть выполнен на основе соотношения

$$(-1)^{x_i} = 1 - 2x_i.$$

При этом арифметическое выражение для двоичного одноразрядного сумматора, полученное в указанном выше разделе на основе спектрального метода, преобразуется к виду:

$$\begin{aligned} Y &= \frac{1}{2} [3 - ((-1)^{x_3} + (-1)^{x_2} + (-1)^{x_1})] = \\ &= \frac{1}{2} [3 - (1 - 2x_3 + 1 - 2x_2 + 1 - 2x_1)] = x_1 + x_2 + x_3. \end{aligned}$$

В разд. 19.4 получены также арифметические выражения для функций, образующих сумматор.

Построим АП для функции «голосование два и более из трех» по соответствующему арифметическому выражению:

$$\begin{aligned} f_2 &= \frac{1}{4} (2 - ((-1)^{x_1} + (-1)^{x_2} + (-1)^{x_3} + (-1)^{x_1+x_2+x_3})) = \\ &= \frac{1}{4} [2 - ((-1)^{x_1} + (-1)^{x_2} + (-1)^{x_3} - (-1)^{x_1} (-1)^{x_2} (-1)^{x_3})] = \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{4} [2 - (1 - 2x_1 + 1 - 2x_2 + 1 - 2x_3 - \\
&\quad - (1 - 2x_1)(1 - 2x_2)(1 - 2x_3))] = \\
&= x_1x_2 + x_2x_3 + x_1x_3 - 2x_1x_2x_3.
\end{aligned}$$

Построим также АП для функции «нечетность трех переменных»:

$$\begin{aligned}
f_1 &= \frac{1}{2} (1 - (-1)^{x_1+x_2+x_3}) = \frac{1}{2} (1 - (-1)^{x_1} (-1)^{x_2} (-1)^{x_3}) = \\
&= \frac{1}{2} (1 - (1 - 2x_1)(1 - 2x_2)(1 - 2x_3)) = \\
&= x_1 + x_2 + x_3 - 2x_1x_2 - 2x_2x_3 - 2x_1x_3 + 4x_1x_2x_3.
\end{aligned}$$

Обратный переход от АП к арифметическим выражениям рассматриваемого вида может быть выполнен на основе соотношения

$$x_i = \frac{1}{2} (1 - (-1)^{x_i}).$$

При этом справедливы соотношения

$$\begin{aligned}
f_3 = x_1x_2 &= \frac{1}{4} (1 - (-1)^{x_1}) (1 - (-1)^{x_2}) = \\
&= \frac{1}{4} (1 - (-1)^{x_2} - (-1)^{x_1} + (-1)^{x_1+x_2}), \\
|S_3| &= \frac{1}{4} |1, -1, -1, 1|; \\
f_4 = x_1 \vee x_2 &= x_1 + x_2 - x_1x_2 = \\
&= \frac{1}{2} (1 - (-1)^{x_1}) + \frac{1}{2} (1 - (-1)^{x_2}) - \frac{1}{4} (1 - (-1)^{x_1}) (1 - (-1)^{x_2}) = \\
&= \frac{1}{4} (3 - (-1)^{x_2} - (-1)^{x_1} - (-1)^{x_1+x_2}), \\
|S_4| &= \frac{1}{4} |3, -1, -1, -1|.
\end{aligned}$$

20.2. Реализация системы булевых функций с использованием линейных арифметических полиномов

Если условие линейности выполняется, то система БФУ n переменных реализуется одним линейным арифметическим полиномом (ЛАП), который строится по значениям системы БФУ только на базовых входных наборах. Такой ЛАП будем называть базовым.

Коэффициенты этого ЛАП определяются матричным соотношением

$$\begin{pmatrix} a_{2^{n-1}} \\ a_{2^{n-2}} \\ \vdots \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & -1 \\ 0 & 1 & 0 & \cdots & 0 & 0 & -1 \\ & & & \ddots & & & \\ 0 & 0 & 0 & \cdots & 1 & 0 & -1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{2^{n-1}} \\ y_{2^{n-2}} \\ \vdots \\ y_2 \\ y_1 \\ y_0 \end{pmatrix}, \quad (20.16)$$

получаемым из соотношения (20.8) вычеркиванием строк и столбцов матрицы P_n и элементов из вектор-столбцов $\mathbf{A}$ и $\mathbf{Y}$, соответствующих нелинейным членам АП.

Эта матрица имеет размерность $(n+1) \times (n+1)$.

20.2.1. Реализация систем булевых функций двух и трех переменных

Пусть задана система булевых функций двух переменных, для которой из четырех входных наборов три являются базовыми, при этом она может быть реализована двумя ЛАП:

$$P_0 = a_0 + a_1 x_2 + a_2 x_1, \quad P_1 = y_3.$$

Коэффициенты полинома P_0 определяются из соотношения (20.16) при $n = 2$:

$$\begin{pmatrix} a_2^0 \\ a_1^0 \\ a_0^0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_2 \\ y_1 \\ y_0 \end{pmatrix}.$$

Например, система функций $Y(x_1, x_2) = [3 \ 4 \ 2 \ 5]^T$ реализуется с помощью двух ЛАП: $P_0 = 3 + x_2 - x_1, P_1 = 5$.

Объединим построенные ЛАП с помощью управляющей конструкции (УК), определяющей область действия каждого полинома. При этом используется тот факт, что каждый базовый входной набор содержит не более одной единицы (рис. 20.1).

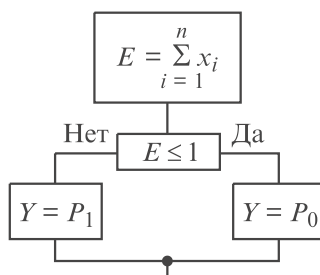


Рис. 20.1

Перейдем к рассмотрению случая $n = 3$. Для этого представим столбец значений Y заданной системы БФУ в виде двух не полностью определенных БФУ, первая из которых Y_0 образуется значениями системы на базовых наборах, а вторая (Y_1) — из всех остальных значений.

Коэффициенты нулевого ЛАП определяются из соотношения (20.16) при $n = 3$:

$$\begin{vmatrix} a_4^0 \\ a_2^0 \\ a_1^0 \\ a_0^0 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} y_4 \\ y_2 \\ y_1 \\ y_0 \end{vmatrix}. \quad (20.17)$$

При этом недоопределенные значения функции Y_0 приобретают значения, определяемые ЛАП.

Столбец функции Y_1 доопределим значениями системы булевых функций на базовых наборах таким образом, чтобы АП, реализующий этот столбец, стал линейным.

Для этого применим соотношение, вытекающее из условия линейности (20.12):

$$|Y_b| = |D C|^{-1} |Y_k|. \quad (20.18)$$

В [15, 16] показано, что при $n = 3$

$$|D C|^{-1} = |D C|,$$

и поэтому из соотношения (20.13) следует, что

$$\begin{vmatrix} y_4 \\ y_2 \\ y_1 \\ y_0 \end{vmatrix} = \begin{vmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ 1 & 1 & 1 & -2 \end{vmatrix} \begin{vmatrix} y_3 \\ y_5 \\ y_6 \\ y_7 \end{vmatrix}.$$

При $n = 3$, подставляя в соотношение (20.16) найденное матричное равенство, получим:

$$\begin{vmatrix} a_4^1 \\ a_2^1 \\ a_1^1 \\ a_0^1 \end{vmatrix} = \begin{vmatrix} -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 1 & 1 & 1 & -2 \end{vmatrix} \begin{vmatrix} y_3 \\ y_5 \\ y_6 \\ y_7 \end{vmatrix}. \quad (20.19)$$

ЛАП, которые могут быть построены с помощью полученных соотношений, объединяются управляющей конструкцией, как показано на рис. 20.1.

Пример 20.9. Реализовать СБФУ, представленную в виде:

$$Y(x_1, x_2, x_3) = |0 \ 3 \ 3 \ 2 \ 3 \ 2 \ 2 \ 7|^T.$$

Используя соотношения (20.17) и (20.19), получим:

$$P_0 = 3x_3 + 3x_2 + 3x_1, \quad P_1 = -8 + 5x_3 + 5x_2 + 5x_1.$$

Для компактного представления линейных АП введем матричную форму их записи:

$$\begin{vmatrix} P_0 \\ P_1 \end{vmatrix} = \begin{vmatrix} 0 & 3 & 3 & 3 \\ -8 & 5 & 5 & 5 \end{vmatrix} \begin{vmatrix} 1 \\ x_3 \\ x_2 \\ x_1 \end{vmatrix}.$$

Применяя результаты разд. 20.1.5, заданная СБФУ может быть реализована одним АП, который является нелинейным и весьма сложным:

$$Y = 3x_3 + 3x_2 - 4x_2x_3 + 3x_1 - 4x_1x_3 - 4x_1x_2 + 10x_1x_2x_3.$$

Предложим табличную форму для записи нелинейных АП (НАП) [16]. Для рассматриваемого примера НАП записан в виде табл. 20.1.

Т а б л и ц а 20.1

	1	x_3	x_2	x_2x_3
1	0	3	3	-4
x_3	3	-4	-4	10

20.2.2. Формирование групп значений системы булевых функций, реализуемых одним линейным полиномом

Значения системы БФУ на базовых наборах могут быть реализованы линейными арифметическими полиномами. Предложим подход к реализации системы булевых функций с помощью ЛАП на остальных наборах.

Разобьем эти наборы на минимальное число групп таким образом, чтобы каждая из них содержала не более $n + 1$ наборов, а значения системы БФУ, входящие в группу, реализовались бы одним ЛАП. Последнее условие выполняется, если для рассматриваемой группы наборов матрица $|\mathbf{D} \ \mathbf{C}|$ имеет обратную, что позволяет вычислять коэффициенты ЛАП на основе соотношений (20.16) и (20.18). Обратимость матрицы $|\mathbf{D} \ \mathbf{C}|$ для группы, содержащей $n + 1$ набор, обеспечивается линейной независимостью [17] системы уравнений, задаваемой соотношением (20.12) на этой группе.

В [15, 16] показано, что линейная независимость указанной системы уравнений достигается при формировании каждой группы входных наборов на основе одного из следующих соотношений:

$$X_{i+1} = X_i + F_i; \quad (20.20)$$

$$X_{i+1} = X_i - F_i, \quad (20.21)$$

где X_i — номер в таблице истинности i -го набора, входящего в группу ($i = 1, 2, \dots, n + 1$); F_i — обобщенное число Фибоначчи [18], определяемое соотношениями

$$F_i = F_{i-1} + F_j, \quad F_1 = F_2 = 1, \quad j = i - 1 \text{ или } i - 2.$$

При $j = i - 1$

$$F_i = 2F_{i-1}; \quad (20.22)$$

при $j = i - 2$

$$F_i = F_{i-1} + F_{i-2}. \quad (20.23)$$

Пример 20.10. Пусть задана система БФУ четырех переменных, реализованная на базовых наборах линейным АП. Из оставшихся наборов необходимо выделить группу из $n + 1$ элементов, значения системы БФУ на которых можно реализовать другим ЛАП.

Выбирая номера наборов, например в соответствии с соотношением (20.20), получим:

$$X_2 = X_1 + F_1, \quad X_3 = X_2 + F_2, \quad X_4 = X_3 + F_3, \quad X_5 = X_4 + F_4.$$

Используя соотношение (20.23), определим, что $F_3 = 2$, $F_4 = 3$. Тогда при $X_1 = 5$ получим: $X_2 = 6$, $X_3 = 7$, $X_4 = 9$, $X_5 = 12$.

Проверим обратимость матрицы $|D \ C|$ для выбранной группы наборов. Для этого сначала запишем условие линейности (20.12) для сформированной группы:

$$\begin{vmatrix} y_5 \\ y_6 \\ y_7 \\ y_9 \\ y_{12} \end{vmatrix} = \begin{vmatrix} 0 & 1 & 0 & 1 & -1 \\ 0 & 1 & 1 & 0 & -1 \\ 0 & 1 & 1 & 1 & -2 \\ 1 & 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 & -1 \end{vmatrix} \begin{vmatrix} y_8 \\ y_4 \\ y_2 \\ y_1 \\ y_0 \end{vmatrix}.$$

Решая эту систему уравнений, получим:

$$\begin{vmatrix} y_8 \\ y_4 \\ y_2 \\ y_1 \\ y_0 \end{vmatrix} = \begin{vmatrix} 0 & 1 & -1 & 1 & 0 \\ 1 & 1 & -1 & 0 & 0 \\ 0 & 2 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ 1 & 2 & -2 & 1 & -1 \end{vmatrix} \begin{vmatrix} y_5 \\ y_6 \\ y_7 \\ y_9 \\ y_{12} \end{vmatrix}.$$

Следовательно, в данном случае обратная матрица существует, и поэтому, применяя последнее соотношение и соотношение (20.16), можно построить ЛАП для найденной группы наборов.

Нахождение обратной матрицы в общем случае весьма трудоемко, однако в [15, 16] установлено, что при формировании группы из $n + 1$ наборов в соответствии с соотношениями (20.21) и (20.23) выполняется равенство

$$|D \ C|^{-1} = |D \ C|.$$

20.2.3. Реализация систем булевых функций четырех переменных

Нулевая группа состоит из значений СБФУ на базовых наборах. При этом из соотношения (20.16) следует, что

$$\begin{vmatrix} a_8^0 \\ a_4^0 \\ a_2^0 \\ a_1^0 \\ a_0^0 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} y_8 \\ y_4 \\ y_2 \\ y_1 \\ y_0 \end{vmatrix}.$$

Первую группу сформируем на основании соотношений (20.21) и (20.22). Полагая $X_1 = 15$, получим $X_2 = 14$, $X_3 = 13$, $X_4 = 11$, $X_5 = 7$. Для этих наборов условие линейности приобретает вид:

$$\begin{vmatrix} y_7 \\ y_{11} \\ y_{13} \\ y_{14} \\ y_{15} \end{vmatrix} = \begin{vmatrix} 0 & 1 & 1 & 1 & -2 \\ 1 & 0 & 1 & 1 & -2 \\ 1 & 1 & 0 & 1 & -2 \\ 1 & 1 & 1 & 0 & -2 \\ 1 & 1 & 1 & 1 & -3 \end{vmatrix} \begin{vmatrix} y_8 \\ y_4 \\ y_2 \\ y_1 \\ y_0 \end{vmatrix}.$$

В силу того что при этом $|\mathbf{D} \mathbf{C}|^{-1} = |\mathbf{D} \mathbf{C}|$, то

$$\begin{vmatrix} y_8 \\ y_4 \\ y_2 \\ y_1 \\ y_0 \end{vmatrix} = \begin{vmatrix} 0 & 1 & 1 & 1 & -2 \\ 1 & 0 & 1 & 1 & -2 \\ 1 & 1 & 0 & 1 & -2 \\ 1 & 1 & 1 & 0 & -2 \\ 1 & 1 & 1 & 1 & -3 \end{vmatrix} \begin{vmatrix} y_7 \\ y_{11} \\ y_{13} \\ y_{14} \\ y_{15} \end{vmatrix}.$$

Подставляя в соотношение (20.16) при $n = 4$ найденное равенство, получим:

$$\begin{vmatrix} a_8^1 \\ a_4^1 \\ a_2^1 \\ a_1^1 \\ a_0^1 \end{vmatrix} = \begin{vmatrix} -1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \\ 1 & 1 & 1 & 1 & -3 \end{vmatrix} \begin{vmatrix} y_7 \\ y_{11} \\ y_{13} \\ y_{14} \\ y_{15} \end{vmatrix}.$$

Разобьем оставшиеся шесть наборов на две группы и определим для каждой из них ЛАП. В силу того что в 3-, 5-, 6-м наборах старшие разряды ноль, а в 9-, 10-, 12-м наборах — единица, сформируем вторую и третью группы по этому признаку: (3-, 5-, 6-й) и (9-, 10-, 12-й). Запишем условие линейности для каждой из этих групп:

$$\begin{vmatrix} y_3 \\ y_5 \\ y_6 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 1 & 1 & -1 \\ 0 & 1 & 0 & 1 & -1 \\ 0 & 1 & 1 & 0 & -1 \end{vmatrix} \begin{vmatrix} y_8 \\ y_4 \\ y_2 \\ y_1 \\ y_0 \end{vmatrix}; \quad \begin{vmatrix} y_9 \\ y_{10} \\ y_{12} \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 1 & -1 \\ 1 & 0 & 1 & 0 & -1 \\ 1 & 1 & 0 & 0 & -1 \end{vmatrix} \begin{vmatrix} y_8 \\ y_4 \\ y_2 \\ y_1 \\ y_0 \end{vmatrix}.$$

Так как в рассматриваемых системах число переменных превышает количество уравнений, то эти системы могут иметь множество решений. В качестве одного из них получим:

$$\begin{vmatrix} y_8 \\ y_4 \\ y_2 \\ y_1 \\ y_0 \end{vmatrix} = \begin{vmatrix} 1 & 1 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & -1 \\ 1 & 1 & -1 \end{vmatrix} \begin{vmatrix} y_3 \\ y_5 \\ y_6 \end{vmatrix}; \quad \begin{vmatrix} y_8 \\ y_4 \\ y_2 \\ y_1 \\ y_0 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} \begin{vmatrix} y_9 \\ y_{10} \\ y_{12} \end{vmatrix}.$$

Подставляя эти выражения в соотношение (20.16) при $n = 4$, получим:

$$\begin{vmatrix} a_8^2 \\ a_4^2 \\ a_2^2 \\ a_1^2 \\ a_0^2 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & -1 \end{vmatrix} \begin{vmatrix} y_3 \\ y_5 \\ y_6 \end{vmatrix}; \quad \begin{vmatrix} a_8^3 \\ a_4^3 \\ a_2^3 \\ a_1^3 \\ a_0^3 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} \begin{vmatrix} y_9 \\ y_{10} \\ y_{12} \end{vmatrix}.$$

Полученные линейные АП объединяются с помощью управляющей конструкции (рис. 20.2), которая основана на следующих особенностях выделенных групп: наборы нулевой группы содержат не более одной единицы; наборы первой группы содержат не менее трех единиц; наборы, образующие вторую и третью группы, разделяются по значению старшей переменной x_1 .

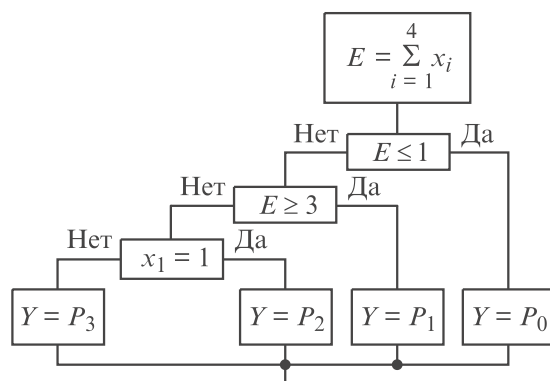


Рис. 20.2

Пример 20.11. Реализовать систему булевых функций, представленную в виде:

$$Y(x_1, x_2, x_3, x_4) = |1 \ 3 \ 2 \ 0 \ 0 \ 1 \ 1 \ 3 \ 1 \ 2 \ 2 \ 0 \ 1 \ 1 \ 1 \ 3|^T.$$

Используя найденные соотношения, получим:

$$\begin{vmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{vmatrix} = \begin{vmatrix} 1 & 2 & 1 & -1 \\ -4 & 2 & 2 & 3 \\ 0 & 0 & 0 & 1 \\ 0 & 2 & 2 & 1 \end{vmatrix} \begin{vmatrix} 1 \\ x_4 \\ x_3 \\ x_2 \end{vmatrix}.$$

Нелинейный арифметический полином, получаемый для этой системы БФУ, задается табл. 20.2.

Т а б л и ц а 20.2

	1	x_4	x_3	$x_3 x_4$
1	1	2	1	-4
x_2	-1	-1	0	5
x_1	0	-1	0	1
$x_1 x_2$	1	0	-1	0

В [16] приведены условия линейности второго и третьего порядков, при выполнении которых системы БФУ четырех переменных реализуются более простой граф-схемой по сравнению с приведенной на рис. 20.2.

20.2.4. Реализация систем булевых функций пяти и более переменных

Применяя подход, изложенный в разд. 20.2.2, удастся все входные наборы при $n = 5$ разбить на шесть групп: 0, 1, 2, 4, 8, 16; 15, 23, 27, 29, 30, 31; 10, 18, 22, 24, 25, 26; 5, 13, 17, 19, 20, 21; 3, 7, 9, 11; 6, 12, 14, 28.

Для каждой из этих групп в [15, 16] найдено матричное соотношение для построения ЛАП. Однако при этом управляющая конструкция, объединяющая ЛАП, получается весьма сложной.

Рассмотрим другой подход к формированию групп, при котором управляющая конструкция также будет состоять из ЛАП и условных вершин, в которых производится сравнение значений этих полиномов с константами.

При этом нулевая и симметричная ей (первая) группы выделяются на основе метода, изложенного в разд. 20.2.2. Выбор этих групп осуществляется с помощью ЛАП вида:

$$E = \sum_{i=1}^n x_i$$

и предикатов с условиями $E \leq 1$ и $E \geq n - 1$ соответственно.

Исходя из соотношения (20.16), получим соотношение для определения коэффициентов ЛАП на базовых наборах. Для первой группы воспользуемся подходом, изложенным в разд. 20.2.2. При этом

$$\begin{pmatrix} a_{16}^0 \\ a_8^0 \\ a_4^0 \\ a_2^0 \\ a_1^0 \\ a_0^0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{16} \\ y_8 \\ y_4 \\ y_2 \\ y_1 \\ y_0 \end{pmatrix};$$

$$\begin{pmatrix} a_{16}^1 \\ a_8^1 \\ a_4^1 \\ a_2^1 \\ a_1^1 \\ a_0^1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 & 1 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 & -4 \end{pmatrix} \begin{pmatrix} y_{15} \\ y_{23} \\ y_{27} \\ y_{29} \\ y_{30} \\ y_{31} \end{pmatrix}.$$

Формирование следующих групп будем выполнять с помощью построения карт Карно, определяющих функции выбора каждой группы. При этом в карте на месте наборов, входящих в выделенные ранее группы, ставятся прочерки. На остальных позициях карты будем пытаться разместить не более $n + 1$ единиц, соответствующих наборам выделяемой группы (оставшиеся позиции карты заполним нулями), таким образом, чтобы функция выбора была пороговой и обеспечивалась линейная независимость системы уравнений, задаваемой условием линейности для этой группы. Пороговость функции позволяет реализовать ее одним ЛАП и одной условной вершиной.

В случае если функция выбора, удовлетворяющая указанным выше условиям, найдена, то в карте Карно нули удаляются, а единицы заменяются прочерками. Тем самым повышается степень неопределенности функции выбора следующей группы, что увеличивает вероятность доопределения этой функции до пороговой. Приведенная процедура повторяется многократно, причем ввиду увеличения

степени неопределенности шансы получения пороговой функции увеличиваются с каждым шагом.

Разобьем оставшиеся двадцать наборов на четыре группы (по пять в каждой). Для определения функции выбора наборов второй группы проставим в карте Карно прочерки на позициях, соответствующих наборам нулевой и первой групп. На остальных позициях попытаемся разместить пять единиц и пятнадцать нулей таким образом, чтобы функция выбора наборов этой группы была пороговой, а соответствующая система уравнений обладала линейной независимостью.

Это решение, найденное эвристически в [15, 16], имеет вид:

$$f_2 = !x_1(!x_2 \vee x_4 x_5).$$

При этом

$$E_2 = -3x_1 - 2x_2 + x_4 + x_5, E_2 \geq 0,$$

а вторая группа состоит из наборов 3, 5, 6, 7, 11. Для этих наборов (в силу того что $x_1 = 0$ и, следовательно, $a_{16}^2 = 0$) соотношение (20.8) приобретает вид:

$$\begin{vmatrix} y_3 \\ y_5 \\ y_6 \\ y_7 \\ y_{11} \end{vmatrix} = \begin{vmatrix} 0 & 0 & 1 & 1 & -1 \\ 0 & 1 & 0 & 1 & -1 \\ 0 & 1 & 1 & 0 & -1 \\ 0 & 1 & 1 & 1 & -1 \\ 1 & 0 & 1 & 1 & -2 \end{vmatrix} \begin{vmatrix} y_8 \\ y_4 \\ y_2 \\ y_1 \\ y_0 \end{vmatrix}.$$

Решая эту систему, получим:

$$\begin{vmatrix} y_8 \\ y_4 \\ y_2 \\ y_1 \\ y_0 \end{vmatrix} = \begin{vmatrix} 0 & 1 & 1 & -2 & 1 \\ 0 & 1 & 1 & -1 & 0 \\ 1 & 0 & 1 & -1 & 0 \\ 1 & 1 & 0 & -1 & 0 \\ 1 & 1 & 1 & -2 & 0 \end{vmatrix} \begin{vmatrix} y_3 \\ y_5 \\ y_6 \\ y_7 \\ y_{11} \end{vmatrix}.$$

На основе этого соотношения и соотношения (20.16) с учетом того, что $a_{16}^2 = 0$, получим:

$$\begin{vmatrix} a_{16}^2 \\ a_8^2 \\ a_4^2 \\ a_2^2 \\ a_1^2 \\ a_0^2 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 1 & 1 & 1 & -2 & 0 \end{vmatrix} \begin{vmatrix} y_3 \\ y_5 \\ y_6 \\ y_7 \\ y_{11} \end{vmatrix}.$$

Далее составим карту Карно с прочерками на месте наборов нулевой, первой и второй групп и построим функцию выбора значений третьей группы. В [15, 16] показано, что

$f_3 = x_1(x_4 \vee !x_2!x_3)$, $E_3 = 3x_1 - x_2 - x_3 + 2x_4$, $E_3 \geq 3$,
а наборами третьей группы являются: 17-, 18-, 19-, 22-, 26-й.

Полагая, что $y_0 = y_{16}$, и выполняя преобразования, аналогичные проведенным для второй группы, получим:

$$\begin{vmatrix} a_{16}^3 \\ a_8^3 \\ a_4^3 \\ a_2^3 \\ a_1^3 \\ a_0^3 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 \end{vmatrix} \begin{vmatrix} y_{17} \\ y_{18} \\ y_{19} \\ y_{22} \\ y_{26} \end{vmatrix}.$$

Составим карту Карно с прочерками на месте наборов всех предшествующих групп. Выделим единицами наборы четвертой группы, а нулями — пятой. В [15, 16] показано, что функция выбора наборов четвертой группы имеет вид:

$$f_4 = x_2(x_5 \vee !x_1x_3), E_4 = -x_1 + 3x_2 + x_3 + 2x_5, E_4 \geq 4.$$

При этом в четвертую группу войдут 9-, 12-, 13-, 14-, 15-й наборы, а в пятую — 10-, 20-, 21-, 24-, 28-й наборы.

Полагая, что $y_0 = 0$, и выполняя преобразования, аналогичные приведенным для второй группы, получим:

$$\begin{vmatrix} a_{16}^4 \\ a_8^4 \\ a_4^4 \\ a_2^4 \\ a_1^4 \\ a_0^4 \end{vmatrix} = \begin{vmatrix} -1 & 0 & 0 & 0 & 1 \\ 1 & 1 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} \begin{vmatrix} y_9 \\ y_{12} \\ y_{13} \\ y_{14} \\ y_{25} \end{vmatrix};$$

$$\begin{vmatrix} a_{16}^5 \\ a_8^5 \\ a_4^5 \\ a_2^5 \\ a_1^5 \\ a_0^5 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \\ 1 & -1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} \begin{vmatrix} y_{10} \\ y_{20} \\ y_{21} \\ y_{24} \\ y_{28} \end{vmatrix}.$$

Таким образом, произвольная систем БФУ пяти переменных может быть реализована граф-схемой, приведенной на рис. 20.3.

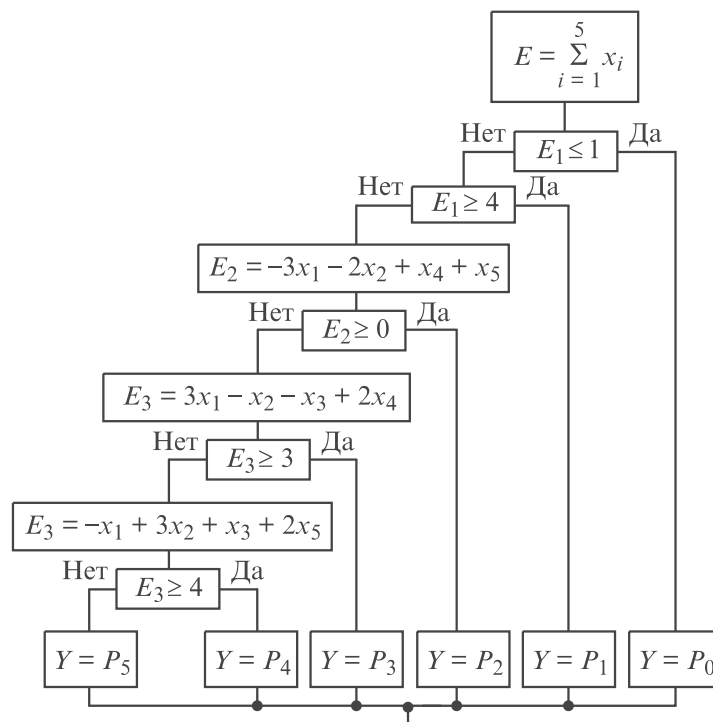


Рис. 20.3

Поэтому можно утверждать, что произвольная система БФУ пяти переменных может быть реализована с помощью десяти ЛАП и пяти предикатов, что по эквивалентной сложности меньше, чем десять пороговых элементов. Следовательно, предлагаемый подход позволил получить принципиально новый результат, относящийся к области реализации систем БФУ в базисе пороговых элементов.

Реализация систем БФУ при $n > 5$ может выполняться, как показано в [16], на основе разложения Шеннона, используя результаты, приведенные выше при $n \leq 5$.

Изложенный в настоящем разделе подход был впервые опубликован в [19].

20.2.5. Реализация алгоритма управления посадкой плавучей полупогружной буровой установки

При проектировании системы управления посадкой плавучей полупогружной буровой установки (ППБУ) возможно применение различных алгоритмов. В настоящей работе предлагается использовать ЛАП для реализации алгоритма управления, предложенного ЦКБ-проектантом ППБУ и основанного на применении факторов E_t .

Фактор E_t это функция, определяющая необходимость изменения количества балласта в t -й группе цистерн ($t = 1, \dots, 4$) ППБУ в зависимости от заданных оператором знаков изменения осадки (pT), крена (pQ) и дифферента ($p\Phi$). При необходимости откачки балласта $E_t = -1$, при отсутствии необходимости его изменения $E_t = 0$, при приеме балласта $E_t = 1$. Троичными (с теми же значениями $-1, 0, 1$) являются и значения аргументов $pT, pQ, p\Phi$.

При этом заданный алгоритм можно представить троичной таблицей истинности. Построенная ТИ преобразуется в таблицу с аргументом X и функцией E за счет представления каждого набора значений аргументов (функций) исходной таблицы в десятичной системе счисления. Для новой таблицы характерно, что

$$E(-X) = -E(X). \quad (20.24)$$

Поэтому будем рассматривать функцию E только при $X \geq 0$ (табл. 20.3).

Т а б л и ц а 20.3

X	0	1	2	3	4	5	6	7	8	9	10	11	12	13
E	0	-32	6	-20	-26	-1	-10	-9	-4	-40	-36	-3	-30	-27

Представим значения X в виде четырехразрядных двоичных чисел $x_1 x_2 x_3 x_4$ и реализуем приведенную таблицу нелинейным АП и выделим в нем линейную часть: $P_0 = -4x_1 - 26x_2 - 6x_3 - 32x_4$. Этот полином реализует значения функции E при $X = 0, 1, 2, 4, 8$. Оставшиеся девять наборов разобьем, используя условие линейности (20.12), на две группы так, чтобы значения функции E на наборах группы реализовывались ЛАП: (5, 9, 11, 12, 13) и (3, 6, 7, 10). При этом

$$P_1 = -17 - 26x_1 + 13x_2 + 37x_3 + 3x_4,$$

$$P_2 = -21 - 15x_1 + 11x_2 + x_4.$$

Сформируем управляющую конструкцию для выбора полиномов P_0, P_1, P_2 , применяя другие ЛАП и условные операторы. Базовый ЛАП выделяется с помощью линейного арифметического по-

линома $P_3 = x_1 + x_2 + x_3 + x_4$ и предиката с условием $P_3 \leq 1$. Для различения полиномов P_1 и P_2 формируется карта Карно с прочерками на базовых и неопределенных наборах, единицами на наборах второй группы и нулями на наборах третьей группы. Функция выбора в этом случае является пороговой и реализуется формулой $P_4 = x_1 x_4 \vee !x_3$, и поэтому $P_4 = x_1 - 2x_3 + x_4$ и $P_4 \geq 0$.

Таким образом, заданная таблица реализуется пятью ЛАП и двумя предикатами. Если $X < 0$, то используется соотношение (20.24).

Следовательно, заданный алгоритм реализуется за счет:

- преобразования троичных входных наборов в двоичные;
- вычисления десятичных значений фактора E с помощью ЛАП и предикатов;
- преобразования этих значений в троичную систему счислений.

При этом отметим, что в настоящей работе первый и третий этапы реализации алгоритма не рассматриваются.

Предложенная реализация [15, 20] является оптимальной по числу операторов для вычислительных устройств, в которых один ЛАП реализуется одной командой [21].

20.3. Реализация одной булевой функции одним линейным арифметическим полиномом с маскированием

20.3.1. Реализация булевой функции безыбыточным линейным арифметическим полиномом

Т е о р е м а 20.2. Линейным арифметическим полиномом безыбыточно можно реализовать лишь булеву функцию, существенно зависящую не более чем от одной переменной: 0, 1, ! x , x [22].

Д о к а з а т е л ь с т в о. Каждое значение булевой функции должно быть равно 0 или 1, а коэффициенты ЛАП (кроме $a_0 = 0, 1$) могут принимать только следующие значения: $-1, 0, 1$. Если $a_0 = 0$ ($a_0 = 1$), то не более одного из оставшихся коэффициентов может быть равно единице (минус единице), в то время как остальные коэффициенты должны быть равны нулю. В противном случае хотя бы на одном входном наборе значение полинома не будет равно ни нулю, ни единице, что и доказывает сформулированную теорему.

Из этой теоремы следует, что одна булева функция, существенно зависящая от двух и более переменных, не может быть реализована безыбыточным линейным арифметическим полиномом, и поэтому рассмотрим вопрос о реализации одной БФУ избыточным ЛАП, который наряду с заданной реализует еще и другие функции. Так как при этом значение заданной БФУ выделяется из двоичного представления избыточного ЛАП с помощью операции маскирования, то такие полиномы будем называть ЛАП с маскированием (ЛАПМ).

20.3.2. Реализация пороговых функций линейным арифметическим полиномом с маскированием

В настоящем разделе рассматриваются два метода реализации пороговых функций (ПФ), первый из которых строит ЛАПМ n переменных по пороговому представлению функции n переменных, а второй — по ЛАПМ пороговой функции, зависящей от $n - 1$ переменных.

Первый метод. Известно [23], что пороговая функция f может быть реализована ЛАП

$$p = p_1 x_1 + p_2 x_2 + \dots + p_n x_n$$

и условным оператором

$$\text{if } (p \geq T) \text{ then } f = 1 \text{ else } f = 0,$$

где p_i — вес переменной x_i , T — порог функции.

В настоящем разделе ставится задача нахождения для функции f другого ЛАП:

$$P = p_0 + p_1 x_1 + p_2 x_2 + \dots + p_n x_n,$$

такого чтобы вычисление соотношения

$$f = r_m^m(\text{bin } P) \quad (20.25)$$

было проще, чем пороговая реализация. В этом выражении нижний индекс равен числу разрядов в двоичном представлении максимального значения полинома P , а верхний — номеру выделяемого маскированием разряда.

Следовательно, задача реализации одной пороговой функции (ПФ) сводится к определению и добавлению к ней справа $m - 1$ булевых функций таким образом, чтобы выполнялось соотношение (20.25).

При этом отметим, что если $T = 2^a$ ($a = 0, 1, 2, \dots$), то наличие единицы хотя бы в одном разряде двоичного представления значения выражения

$$\sum_{i=1}^n p_i x_i,$$

начиная с разряда с номером $a + 1$ (при нумерации разрядов с единицы), свидетельствует о том, что вычисленное значение достигло или превысило величину T .

Если $T \neq 2^a$, то задача сводится к рассмотренной за счет увеличения порога до величины, равной ближайшей целой степени двух. Для этого необходимо добавить к левой и правой части неравенства

$$\sum_{i=1}^n p_i x_i \geq T,$$

входящего в определение ПФ, величину $(2^{\lceil \log T \rceil} - T)$:

$$2^{\lceil \log T \rceil} - T + \sum_{i=1}^n p_i x_i \geq 2^{\lceil \log T \rceil}. \quad (20.26)$$

При этом новым порогом будет величина $T_1 = 2^a = 2^{\lceil \log T \rceil}$.

Пусть b — число разрядов в двоичном представлении максимального значения величины P .

Рассмотрим случай, когда $b = a + 1$. В этом случае превышение порога происходит только в одном разряде двоичного представления значения величины P . Для этого величина b не должна превышать число разрядов двоичного представления величины порога T_1 .

Сформулированное условие можно записать в виде:

$$P_{\max} < 2T, \quad (20.27)$$

где $P_{\max}$ — максимальное значение полинома P .

Если для пороговой функции условие (20.27) не выполняется, то необходимо увеличить величину T_1 в два раза. Для этого требуется добавить к левой и правой части неравенства (20.26) величину T_1 . Если и при этом условие (20.27) не выполняется, указанная процедура повторяется до его выполнения.

Таким образом, можно добиться выполнения условия (20.27) для любой пороговой функции, так как при увеличении левой и правой части неравенства (20.26) правая часть неравенства (20.27) возрастает в два раза быстрее левой.

Следовательно, доказано, что произвольная пороговая функция может быть вычислена следующим образом:

$$f = r_b^b(\text{bin } P).$$

Программная реализация этого соотношения на языке высокого уровня, например СИ, чрезвычайно проста и состоит в вычислении на заданном входном наборе значения полинома P , в сдвиге полученного результата на $b - 1$ разрядов вправо и считывании полученного результата, равного значению ПФ на этом наборе.

Для того чтобы воспользоваться предложенным подходом для заданной пороговой функции, необходимо предварительно определить ее пороговую реализацию (ПР).

Процедура определения весов переменных и порога для пороговой функции весьма трудоемка [23], и поэтому укажем три более простых подхода к нахождению пороговой реализации. Первый подход строит ПР непосредственно по описанию ПФ, второй основан на использовании для произвольных пороговых функций шести и менее переменных каталогов оптимальных ПР (например, [24]), а третий, изложенный в разд. 15.2, применим для неповторных пороговых формул (БПФ), имеющих важное практическое и теоретическое значение [22, 25—27].

Пример 20.12. Реализовать ПФ «два и более из трех», которая описывается формулой $f = (x_1 \vee x_2)x_3 \vee x_1x_2$.

Из словесного описания функции следует, что $x_1 + x_2 + x_3 \geq 2$. Так как в этом случае $T = 2^a$, а соотношение (20.27) выполняется, то

$$f = r_2^2(\text{bin}(x_1 + x_2 + x_3)).$$

Проиллюстрируем полученное соотношение табл. 20.4.

Т а б л и ц а 20.4

x_1	x_2	x_3	$x_1 + x_2 + x_3$	$\text{bin}(x_1 + x_2 + x_3)$	
				r_2^2	r_2^1
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	2	1	0
1	0	0	1	0	1
1	0	1	2	1	0
1	1	0	2	1	0
1	1	1	3	1	1

Из этой таблицы следует, что введение избыточности (реализация полиномом двух функций вместо одной) и применение операции маскирования позволило построить ЛАПМ вместо безызыточного (по числу объединяемых функций), но более сложного нелинейного арифметического полинома:

$$f = x_1x_2 + x_1x_3 + x_2x_3 - 2x_1x_2x_3.$$

Таким образом, повторной булевой формуле соответствует неповторный арифметический полином с маскированием, что является основной предпосылкой использования последнего для эффективной программной реализации. Для неповторной булевой

формулы ее непосредственная программная реализация всегда более эффективна, чем реализация на основе ЛАПМ. Однако для систем пороговых формул, даже неповторных, применение ЛАПМ может обеспечить минимальную по объему памяти программную реализацию (разд. 20.4.1).

При использовании языка СИ найденный ЛАПМ реализуется следующей программой:

```
laps (int x1, int x2, int x3)
{int f;
 f = (x1 + x2 + x3) >> 1;
 return (f);},
```

где $\gg L$ — символ операции «сдвиг на L двоичных разрядов вправо».

Эта программа, как уже было отмечено в разд. 18.6, является наилучшей, как по объему памяти, так и по быстродействию, по сравнению с другими шестьюдесятью программами, приведенными в [3], вычисляющими эту функцию в предположении, что i -я входная переменная находится в крайнем правом разряде i -го слова.

Пример 20.13. Реализовать ПФ «семь и более из десяти». Из описания функции следует, что

$$\sum_{i=1}^{10} x_i \geq 7.$$

Увеличим порог функции до величины 2^a . При этом

$$1 + \sum_{i=1}^{10} x_i \geq 8.$$

Так как при этом неравенство (20.27) выполняется, то

$$f = r_4^4 \left(\text{bin} \left(1 + \sum_{i=1}^{10} x_i \right) \right).$$

Этот пример демонстрирует возможность высокой эффективности применения ЛАПМ, так как таблица истинности, булева формула или нелинейный АП в этом случае чрезвычайно громоздки и поэтому программируются с большими затратами памяти.

Пример 20.14. Реализовать пороговую функцию

$$f = (x_1 \vee x_2 \vee x_3) x_4 \vee x_1 x_2 x_3.$$

В этом случае из [24] следует, что $x_1 + x_2 + x_3 + 2x_4 \geq 3$.

В силу того что в этом случае $T \neq 2^a$, увеличим порог функции:

$$1 + x_1 + x_2 + x_3 + 2x_4 \geq 4.$$

Так как при этом соотношения (20.26) и (20.27) выполняются, то

$$f = r_3^3 (\text{bin} (1 + x_1 + x_2 + x_3 + 2x_4)).$$

Пример 20.15. Реализовать БПФ $f = x_1 x_2$.

Используя метод, изложенный выше, получим, что $x_1 + x_2 \geq 2$. Так как при этом $T = 2^a$, а соотношение (20.27) выполняется, то

$$f = r_2^2 (\text{bin} (x_1 + x_2)).$$

Пример 20.16. Реализовать БПФ $f = x_1 \vee x_2$.

Применяя метод, изложенный выше, получим, что $x_1 + x_2 \geq 1$.

Так как в этом случае $T = 2^a$, но условие (20.27) не выполняется, увеличим порог функции:

$$1 + x_1 + x_2 \geq 2.$$

Ввиду того что при этом оба условия выполняются, то

$$f = r_2^2 (\text{bin} (1 + x_1 + x_2)).$$

Пример 20.17. Реализовать БПФ $f = x_1 x_2 x_3$.

В этом случае $x_1 + x_2 + x_3 \geq 3$. Так как $T \neq 2^a$, увеличим порог функции до ближайшей степени двух:

$$1 + x_1 + x_2 + x_3 \geq 4.$$

Ввиду того что при этом условие (20.27) выполняется, то

$$f = r_3^3 (\text{bin} (1 + x_1 + x_2 + x_3)).$$

Пример 20.18. Реализовать БПФ $f = x_1 x_2 \vee x_3$.

В этом случае $x_1 + x_2 + 2x_3 \geq 2$ и $T = 2^a$, и так как соотношение (20.27) не выполняется ($P_{\max} = 2T$), то увеличим порог до следующей величины, равной степени двух: $2 + x_1 + x_2 + 2x_3 \geq 4$. Так как при этом соотношение (20.27) выполняется, то

$$f = r_3^3 (\text{bin} (2 + x_1 + x_2 + 2x_3)).$$

Второй метод. Метод основан на теореме, приведенной в [28].

Теорема 20.3. Если кортеж N булевых функций n переменных

$$f_N f_{N-1} \dots f_1$$

реализуется линейным арифметическим полиномом P_N , то кортеж из $(N+1)$ -й функций

$$(f_N \& x_{n+1})(f_N \oplus x_{n+1})f_{N-1} \dots f_1 \quad (20.28)$$

реализуется ЛАП вида $P_{N+1} = P_N + 2^{N-1} x_{n+1}$, а кортеж из того же числа функций

$$(f_N \vee x_{n+1})(f_N \oplus x_{n+1} \oplus 1)f_{N-1} \dots f_1 \quad (20.29)$$

реализуется ЛАП вида $P_{N+1} = P_N + 2^{N-1} x_{n+1} + 2^{N-1}$.

В силу установленного в [15, 16] факта, что ЛАП реализует лишь такие кортежи функций, в которых крайняя левая функция f_N является пороговой (в [23] показано, что если f_N — пороговая функция n переменных, то и функции $f_N \& x_{n+1}$ и $f_N \vee x_{n+1}$ также являются пороговыми), приведенная теорема определяет:

- рекурсивный метод построения ЛАПМ для БПФ;
- простой метод реализации ПФ вида $f_N \& x_n$ и $f_N \vee x_n$ для случая, когда ЛАПМ для функции f_N уже найден.

Пример 20.19. Реализовать пороговую функцию $f = x_1 x_2 x_3$.

Из примера 20.15 следует, что для $f = x_1 x_2$ $n = 2$, $N = 2$, и поэтому из соотношения (20.28) получим:

$$f = r_3^3 (\text{bin} (x_1 + x_2 + 2x_3)).$$

Из сопоставления этого соотношения с приведенным в примере 20.17 следует, что представление булевых формул с помощью ЛАПМ в отличие от их представления АП не является единственным.

Пример 20.20. Реализовать пороговую функцию

$$f = (x_1 \vee x_2)x_3 \vee x_1 x_2 \vee x_4.$$

Из примера 20.12 следует, что для $f_1 = (x_1 \vee x_2)x_3 \vee x_1 x_2$ $n = 2$, $N = 2$, и поэтому, исходя из соотношения (20.29), получим:

$$f = r_3^3 (\text{bin} (2 + x_1 + x_2 + x_3 + 2x_4)).$$

Методы, рассмотренные выше, применимы к пороговым функциям, представимым формулами без инверсий. При наличии в реализуемой формуле переменных типа $\neg x_i$ первоначально ЛАПМ

строится для PN -однотипной с ней ПФ без инверсий, а затем в ЛАПМ переменная x_i заменяется соотношением $1 - x_i$.

Пример 20.21. Реализовать пороговую функцию $f = x_1 \vee !x_2$.
Из изложенного и примера 20.16 следует, что

$$f = r_2^2(\text{bin}(2 + x_1 - x_2)).$$

Пример 20.22. Реализовать пороговую функцию

$$f = (x_1 \vee x_2)x_3 \vee x_1x_2 \vee !x_4.$$

Из изложенного и примера 20.20 следует, что

$$f = r_3^3(\text{bin}(4 + x_1 + x_2 + x_3 - 2x_4)).$$

20.3.3. Реализация линейных булевых функций линейным арифметическим полиномом с маскированием

Линейной называется булева формула, представимая формулой

$$f = c_0 \oplus c_1x_1 \oplus c_2x_2 \oplus \dots \oplus c_nx_n,$$

где $c_i = 0, 1$.

Линейная функция реализуется ЛАПМ вида

$$f = r_{1+\lceil \log n \rceil}^1(\text{bin}(c_0(1 - x_n) + \dots + c_kx_k)).$$

Поэтому

$$\begin{aligned} f_1 &= x_1 \oplus x_2 = r_2^1(\text{bin}(x_1 + x_2)); \\ f_2 &= x_1 \oplus x_2 \oplus x_3 = r_2^1(\text{bin}(x_1 + x_2 + x_3)). \end{aligned}$$

Вторая из этих функций реализуется на языке СИ следующей программой:

```
mod (int x1, int x2, int x3)
{int f2;
 f2 = (x1 + x2 + x3) & 1;
 return (f2);}
```

Рассмотренные функции реализуются теми же ЛАПМ, что и функции $f_{11} = x_1x_2$, $f_{21} = (x_1 \vee x_2)x_3 \vee x_1x_2$ соответственно, как и должно быть для двоичных одноразрядных полусумматора и сумматора.

20.3.4. Реализация порогово-линейных функций линейным арифметическим полиномом с маскированием

Порогово-линейной функцией (ПЛФ) будем называть булеву функцию, представимую формулой вида:

$$f = f_1(x_1, \dots, x_n) \oplus \tilde{x}_i \oplus \dots \oplus \tilde{x}_j,$$

где f_1 — пороговая функция; i, j — целые числа; $\tilde{x}_i = \{!x_i, x_i\}$.

Рассмотрим методы построения ЛАПМ для БФУ, заданных в порогово-линейной форме.

Использование теоремы 20.3 позволяет реализовать ПЛФ, для которых $i, j > n$, а ЛАПМ для функции f_1 найден с помощью методов, изложенных в разд. 20.3.2.

Пример 20.23. Реализовать ПЛФ $f = x_1 x_2 x_3 \oplus !x_4$.

Из примера 20.17 следует, что

$$f_1 = x_1 x_2 x_3 = r_3^3(\text{bin}(1 + x_1 + x_2 + x_3)).$$

Поэтому из соотношения (20.29) следует, что

$$\begin{aligned} f &= x_1 x_2 x_3 \oplus !x_4 = x_1 x_2 x_3 \oplus x_4 \oplus 1 = \\ &= r_4^3(\text{bin}(5 + x_1 + x_2 + x_3 + 4x_4)). \end{aligned}$$

Эта булева функция может быть реализована и другим ЛАПМ. Из соотношения (20.28) и разд. 20.3.2 следует, что

$$f = x_1 x_2 x_3 \oplus !x_4 = r_4^3(\text{bin}(5 + x_1 + x_2 + x_3 - 4x_4)).$$

Это соотношение реализуется на языке СИ следующей программой:

```
plf (int x1, int x2, int x3, int x4)
{int f;
  f = ((5 + x1 + x2 + x3 - 4*x4) >> 2) & 1;}
```

Недостаток теоремы 20.3 состоит в том, что ее нельзя использовать для случая, когда $i, j \leq n$. Этот недостаток устраняется применением следующей теоремы [29].

Т е о р е м а 20.4. Если $f_1 = r_m^k(\text{bin } P_N)$, то $f = f_1 \oplus x_i$ реализуется в k -м разряде полинома $P_N + 2^{k-1}x_i$, где $i = \{1, 2, \dots, n+1\}$.

Пример 20.24. Реализовать ПЛФ $f = x_1 \oplus x_2(!x_3 \vee x_4)$.

Из примера 20.21 следует, что $f_1 = !x_3 \vee x_4 = r_2^2(\text{bin}(2 - x_3 + x_4))$. Используя соотношение (20.28), получим:

$$f_2 = x_2(!x_3 \vee x_4) = r_3^3(\text{bin}(2 + 2x_2 - x_3 + x_4)).$$

Применяя теорему 20.4, находим:

$$f = r_4^3 (\text{bin} (2 + 4x_1 + 2x_2 - x_3 + x_4)).$$

При этом отметим, что в [5] эта формула реализуется существенно более сложным нелинейным АП:

$$\begin{aligned} f = & 1 - x_1 - x_2 - x_3 + 2x_1x_2 + 2x_1x_3 + 2x_2x_3 + x_3x_4 - \\ & - 4x_1x_2x_3 - 2x_1x_3x_4 - 2x_2x_3x_4 + 4x_1x_2x_3x_4. \end{aligned}$$

Пример 20.25. Реализовать ПЛФ $f = (x_1 \vee x_2 \vee x_3) \oplus x_3$.
В силу того что

$$f_1 = x_1 \vee x_2 \vee x_3 = r_3^3 (\text{bin} (3 + x_1 + x_2 + x_3)),$$

то из теоремы 20.4 следует, что

$$f = r_4^3 (\text{bin} (3 + x_1 + x_2 + 5x_3)).$$

С другой стороны, так как

$$f = (x_1 \vee x_2 \vee x_3) \oplus x_3 = (x_1 \vee x_2)x_3$$

является ПФ, то, используя результаты разд. 20.3.2, получим:

$$f = r_3^3 (\text{bin} (3 + x_1 + x_2 - 2x_3)).$$

Пример 20.26. Реализовать ПЛФ $f = x_1x_2 \oplus x_3 \oplus x_4 \oplus x_5$.
В данном случае

$$f_1 = x_1x_2 = r_2^2 (\text{bin} (x_1 + x_2)).$$

Двукратно применяя теорему 20.4, получим:

$$\begin{aligned} f_2 = x_1x_2 \oplus x_3 &= r_3^2 (\text{bin} (x_1 + x_2 + 2x_3)); \\ f_3 = x_1x_2 \oplus x_3 \oplus x_4 &= r_3^2 (\text{bin} (x_1 + x_2 + 2x_3 + 2x_4)). \end{aligned}$$

Таким образом,

$$f = r_4^2 (\text{bin} (x_1 + x_2 + 2x_3 + 2x_4 + 2x_5)).$$

Интересным является тот факт, что в этом примере в отличие от предыдущих реализуемая функция «сдвинулась» в середину двоичного представления.

Перейдем к рассмотрению случая, когда ПЛФ задана в форме, отличной от порогово-линейной. Поиск такой формы предлагается проводить на основе соотношений, приведенных в разд. 1.7:

$$f = (!x_i f(0) \vee x_i !f(1)) \oplus x_i; \quad (20.30)$$

$$f = (!x_i !f(0) \vee x_i f(1)) \oplus !x_i. \quad (20.31)$$

Пример 20.27. Реализовать ПЛФ $f = !x_1 x_2 \vee x_1 !x_2 x_3$.
На основе соотношения (20.30) получим:

$$f = f_1 \oplus x_1 = (x_1 !x_3 \vee x_2) \oplus x_1.$$

Для пороговой функции

$$f_1 = r_3^3 (\text{bin}(3 + x_1 + 2x_2 - x_3)).$$

Поэтому, используя теорему 20.4, получим:

$$f = r_4^3 (\text{bin}(3 + x_1 + 2x_2 - x_3)).$$

Эта функция реализована в [29] с помощью алгоритма Балаша более сложным ЛАПМ:

$$f = r_6^4 (\text{bin}(5 + 12x_1 + 6x_2 + 14x_3)).$$

Пример 20.28. Реализовать булеву формулу

$$f = (!x_1 \vee !x_2 \vee !x_3)(x_1 \vee x_2 \vee x_3).$$

Применяя соотношение (20.30), получим:

$$f = f_1 \oplus x_1 = (!x_1(x_2 \vee x_3) \vee x_2 x_3) \oplus x_1.$$

Для пороговой функции

$$f_1 = r_2^2 (\text{bin}(1 - x_1 + x_2 + x_3)).$$

Поэтому, используя теорему 20.4, получим:

$$f = r_3^2 (\text{bin}(1 + x_1 + x_2 + x_3)).$$

Пример 20.29. Реализовать функцию

$$f = x_1 !x_2 !x_3 x_4 \vee !x_1 x_2 x_3 x_4 \vee !x_1 !x_2 !x_4 \vee x_1 x_3 !x_4 \vee x_2 !x_3 !x_4.$$

Из соотношения (20.30) при $i = 4$ следует, что

$$f = f_1 \oplus x_4 = (!x_1 !x_2 \vee x_1 x_3 \vee x_2 !x_3) \oplus x_4.$$

Применяя соотношение (20.31) к функции f_1 , получим:

$$f = f_2 \oplus !x_1 \oplus x_4 = ((x_1 \vee x_2)x_3 \vee x_1x_2) \oplus !x_1 \oplus x_4.$$

Из примера 20.12 следует, что

$$f_2 = r_2^2 (\text{bin} (x_1 + x_2 + x_3)).$$

Поэтому, используя теорему 20.4, получим:

$$f_1 = r_3^2 (\text{bin} (2 - x_1 + x_2 + x_3)).$$

Повторно применяя теорему 20.4, окончательно получим:

$$f = r_3^2 (\text{bin} (2 - x_1 + x_2 + x_3 + 2x_4)).$$

Найденное соотношение в отличие от других форм представления этой функции является неповторным, что обеспечивает более простую программную реализацию.

Вид и сложность ЛАПМ, получаемого по ПЛФ, зависят от того, какое из этих соотношений применяется и какая переменная используется в качестве x_i .

Пример 20.30. Реализовать ПЛФ $f = !x_1!x_2x_3 \vee x_1x_2!x_3$.

Полагая в соотношении (20.30) $x_i = x_1$, получим:

$$\begin{aligned} f &= f_1 \oplus x_1 = (!x_1(!x_2x_3) \vee x_1!(x_2!x_3)) \oplus x_1 = \\ &= ((x_1 \vee x_3)!x_2 \vee x_1x_3) \oplus x_1. \end{aligned}$$

При этом, так как

$$\begin{aligned} f_2 &= (x_1 \vee x_3)x_2 \vee x_1x_3 = r_2^2 (\text{bin} (x_1 + x_2 + x_3)), \\ f_1 &= r_2^2 (\text{bin} (1 + x_1 - x_2 + x_3)). \end{aligned}$$

Используя теперь теорему 20.4, находим:

$$f = r_3^2 (\text{bin} (1 + 3x_1 - x_2 + x_3)).$$

Применяя в качестве x_i переменные $!x_1, x_2, !x_2, x_3, !x_3$, получим соответственно следующие соотношения

$$\begin{aligned} f &= r_3^2 (\text{bin} (4 - 3x_1 + x_2 - x_3)) = r_3^2 (\text{bin} (1 - x_1 + 3x_2 + x_3)) = \\ &= r_3^2 (\text{bin} (4 + x_1 - 3x_2 - x_3)) = r_3^2 (\text{bin} (x_1 + x_2 + 3x_3)) = \\ &= r_3^2 (\text{bin} (5 - x_1 - x_2 - 3x_3)). \end{aligned}$$

Таким образом, простейшая реализация обеспечивается при $x_i = x_3$. Вопросы принадлежности функции классу ПЛФ и выбора переменной x_i в соотношениях (20.29), (20.30) остаются открытыми. Изложенный подход описан в [15, 22].

20.3.5. Реализация симметрических функций линейным арифметическим полиномом с маскированием

Рассмотрим класс ЛАП вида:

$$P = \sum_{i=1}^n x_i.$$

Эти ЛАП описывают работу комбинационных счетчиков, для построения которых требуется определить систему из $1 + \lceil \log n \rceil$ булевых функций. Эти функции относятся к классу симметрических и реализуются следующим образом.

Запишем матрицу-столбец размерности $(n+1) \times 1$, все элементы которой принимают значения 0, 1, ..., n . Эта матрица преобразуется в двоичную матрицу размерности $(n+1) \times (1 + \lceil \log n \rceil)$, i -я ($i = 0, 1, \dots, n$) строка которой является двоичным представлением i -го элемента матрицы-столбца. При этом j -й столбец двоичной матрицы является характеристическим числом j -й функции, единица в i -м разряде которого указывает на наличие рабочего числа в этой функции.

Пример 20.31. Определить симметрические функции (СФ), реализуемые линейным арифметическим полиномом

$$P = x_1 + x_2 + x_3 + x_4.$$

Выполняя указанные построения и преобразования, получим

$$P = \begin{vmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{vmatrix} = S_4^4 S_{2,3}^4 S_{1,3}^4.$$

Рассмотрим более широкий класс ЛАП:

$$P_1 = c_1 + c_2 \sum_{i=1}^n x_i, \quad (20.32)$$

где c_1, c_2 — положительные целые числа.

Этот класс ЛАП реализует СФ так же, как рассмотрено выше.

Пример 20.32. Определить СФ, реализуемые ЛАП

$$P_1 = 3 + 5(x_1 + x_2 + x_3 + x_4).$$

Искомые функции определяются следующим образом:

$$P_1 = 3 + 5 \begin{vmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{vmatrix} = 13 \begin{vmatrix} 3 \\ 8 \\ 18 \\ 23 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \end{vmatrix} =$$

$$= S_{3,4}^4 S_{1,2}^4 S_{2,4}^4 S_{0,3,4}^4 S_{0,2,4}^4.$$

Из изложенного следует, например, что

$$f = S_{2,4}^4 = r_5^3 [\text{bin}(3 + 5(x_1 + x_2 + x_3 + x_4))].$$

Такая реализация существенно более эффективна, чем непосредственное построение АП для СФ, так как, используя результаты разд. 20.1.2, можно получить, что $b_0 = 0$, $b_1 = 0$, $b_2 = 1$, $b_3 = -3$, $b_4 = 7$, и поэтому

$$f = S_{2,4}^4 = x_1 x_2 + x_1 x_3 + x_1 x_4 + x_2 x_3 + x_2 x_4 + x_3 x_4 -$$

$$- 3(x_1 x_2 x_3 + x_1 x_2 x_4 + x_1 x_3 x_4 + x_2 x_3 x_4) + 7 x_1 x_2 x_3 x_4.$$

Из изложенного следует, что выражение (20.32) может рассматриваться в качестве эффективного генератора симметрических функций. Исследование показало, что все СФ двух переменных порождаются пятью линейным АП при $c_1 = 0, \dots, 4$, $c_2 = 1$, а все СФ трех переменных — одиннадцатью линейными АП, в которых $c_1 = 0, \dots, 5$, $c_2 = 1$; $c_1 = 7$, $c_2 = 1$; $c_1 = 0$, $c_2 = 3$; $c_1 = 2$, $c_2 = 3$; $c_1 = 4$, $c_2 = 3$; $c_1 = 5$, $c_2 = 5$.

Рассмотренный метод является переборным и не гарантирует реализацию с помощью ЛАПМ произвольной СФ n переменных. Конструктивные методы реализации СФ, являющихся пороговыми, порогово-линейными и линейными, рассмотрены в предыдущих разделах. Например, с помощью ЛАПМ представимы следующие СФ:

$$S_{0,2,3}^3 = (x_1 x_3 \vee !x_2) \oplus x_1 \oplus x_3 = r_4^3 (\text{bin}(4 + 5x_1 - 2x_2 + 5x_3));$$

$$S_1^3 = (!x_1 \vee !x_3) x_2 \oplus x_1 \oplus x_3 = r_4^3 (\text{bin}(3 + 3x_1 + 2x_2 + 3x_3)).$$

Вопрос о принадлежности произвольной СФ n переменных одному из перечисленных классов остается открытым. Изложенный подход описан в [22].

20.4. Реализация системы булевых функций линейными арифметическими полиномами с маскированиями

20.4.1. Реализация системы линейно представимых функций одним линейным полиномом с маскированиями

Булева функция, реализуемая одним ЛАПМ, называется линейно представимой. В предыдущих разделах показано, что пороговые, линейные и порогово-линейные функции линейно представимы.

Т е о р е м а 20.5. Произвольная система N линейно представимых функций n переменных также линейно представима полиномом, зависящим от n переменных, т. е. всегда может быть реализована одним линейным арифметическим полиномом n переменных с N маскированиями.

Эта теорема доказывается тем, что всегда можно построить один линейный арифметический полином, маскирование разрядов которого порождает N линейно представимых функций, если

$$P = 2^{g_1 + \dots + g_{N-1}} \cdot P_{g_N}^N + \dots + 2^{g_1 + g_2} \cdot P_{g_3}^3 + 2^{g_1} \cdot P_{g_2}^2 + 2^0 \cdot P_{g_1}^1, \quad (20.33)$$

где $P_{g_i}^i$ — ЛАПМ i -й булевой функции, двоичное представление результатов вычислений которой занимает g_i разрядов.

Этот результат чрезвычайно важен для теории пороговых функций, так как, в частности, утверждает, что любая система из N указанных функций реализуется одним линейным полиномом с N маскированиями, что по эквивалентной сложности существенно проще, чем N пороговых элементов.

Отметим, что при использовании соотношения (20.33) двоичное представление результатов вычислений по ЛАПМ занимает

$$g = g_1 + g_2 + \dots + g_N \quad (20.34)$$

разрядов.

При применении предлагаемого подхода полиномы, соответствующие заданным функциям, могут объединяться в порождающий полином в различном порядке. Это может привести к построению до $N!$ различных порождающих полиномов, которые, однако, по вводимому ниже критерию имеют одинаковую сложность.

Структуры этих полиномов отличаются друг от друга существенно меньше, чем структуры нелинейных АП, так как в первом случае они определяются $n + 1$ коэффициентами, а во втором — 2^n коэффициентами. В первом случае полином существенно зависит от всех n переменных, если он содержит по крайней мере n ненулевых коэффициентов при одиночных переменных, а во втором — он

может существенно зависеть от всех n переменных даже при наличии только одного ненулевого коэффициента с номером $2^n - 1$.

Будем характеризовать каждый полином P , существенно зависящий от n переменных, величиной вычисляемого им максимального десятичного значения, обозначаемого символом $|P|$. При этом считается, что тот полином, существенно зависящий от n переменных, проще, у которого величина

$$g = \lceil \log |P| \rceil$$

меньше.

Пример 20.33. Проверить линейную представимость функций:

$$f_1 = (x_1 \vee x_2) x_3 \vee x_1 x_2, \quad f_2 = !x_1 \vee !x_2 \vee !x_3, \\ f_3 = (!x_1 \vee !x_2) !x_3,$$

и реализовать эти функции одним ЛАПМ в случае положительных результатов проверки.

Так как каждая из заданных функций является пороговой и поэтому линейно представимой, то:

$$f_1 = r_2^2 (\text{bin} (x_1 + x_2 + x_3)); \quad f_2 = r_3^3 (\text{bin} (6 - x_1 - x_2 - x_3)); \\ f_3 = r_3^3 (\text{bin} (5 - x_1 - x_2 - 2x_3)).$$

Эти функции могут быть порождены полиномами, каждый из которых в результате вычислений формирует восьмиразрядное двоичное число

$$g = g_1 + g_2 + g_3 = 2 + 3 + 3 = 8.$$

Используя соотношение (20.33), получим:

$$P_1 = 2^6 (x_1 + x_2 + x_3) + 2^3 (6 - x_1 - x_2 - x_3) + \\ + 2^0 (5 - x_1 - x_2 - 2x_3) = 53 + 55x_1 + 55x_2 + 54x_3; \\ f_1 = r_8^8 (\text{bin } P_1); \quad f_2 = r_8^6 (\text{bin } P_1); \quad f_3 = r_8^3 (\text{bin } P_1).$$

При этом $|P_1| = 217$.

Приведем программу на языке СИ, реализующую полученные соотношения:

```
lapm (int l, int x2, int x3, int P1, int f1,
int f2, int f3)
{ P1 = 53 + 55*x1 + 55*x2 + 54*x3;
f1 = P1 >> 7;
f2 = (P1 >> 5) & 1;
f3 = (P1 >> 2) & 1}.
```

Проиллюстрируем найденное соотношение табл. 20.5.

Т а б л и ц а 20.5

x_1	x_2	x_3	P_1	$f_1 = r_8^8$	r_8^7	$f_2 = r_8^6$	r_8^5	r_8^4	$f_3 = r_8^3$	r_8^2	r_8^1
0	0	0	53	0	0	1	1	0	1	0	1
0	0	1	107	0	1	1	0	1	0	1	1
0	1	0	108	0	1	1	0	1	1	0	1
0	1	1	162	1	0	1	0	0	0	1	0
1	0	0	108	0	1	1	0	1	1	0	0
1	0	1	162	1	0	1	0	0	0	1	0
1	1	0	163	1	0	1	0	0	0	1	1
1	1	1	201	1	1	0	1	1	0	0	1

Запишем формулы для БФУ, приведенных в таблице:

$$f_1 = r_8^8 = (x_1 \vee x_2) x_3 \vee x_1 x_2, \quad r_8^7 = x_1 \oplus x_2 \oplus x_3,$$

$$f_2 = r_8^6 = !x_1 \vee !x_2 \vee !x_3,$$

$$r_8^5 = !x_1 !x_2 !x_3 \vee x_1 x_2 x_3 = ((x_1 \vee !x_3) !x_2 \vee x_1 !x_3) \oplus x_1,$$

$$r_8^4 = x_1 \oplus x_2 \oplus x_3, \quad f_3 = r_8^3 = (!x_1 \vee !x_2) !x_3,$$

$$r_8^2 = x_1 x_2 \oplus x_3, \quad r_8^1 = x_1 \oplus x_2.$$

При этом восьмая, шестая и третья функции являются пороговыми, пятая и вторая — порогово-линейными, а седьмая, четвертая и первая — линейными.

Аналогично рассмотренному могут быть построены ЛАПМ и для других кортежей функций f_1, f_2, f_3 . При этом простейшим по введенному критерию будет полином $P_2 = 166 - 25x_1 - 25x_2 - 57x_3$, полученный для кортежа $f_3 f_1 f_2$. При использовании этого полинома $f_3 = r_8^8 (\text{bin } P_2)$, $f_1 = r_8^5 (\text{bin } P_2)$, $f_2 = r_8^3 (\text{bin } P_2)$. В этом случае $|P_2| = 166$.

Пример 20.34. Реализовать систему линейно представимых функций:

$$f_1 = x_1 x_2 x_3 = r_3^3 (\text{bin } (1 + x_1 + x_2 + x_3));$$

$$f_2 = !x_1 !x_2 !x_3 = r_3^3 (\text{bin } (4 - x_1 - x_2 - x_3));$$

$$f_3 = x_1 \vee x_2 \vee x_3 = r_3^3 (\text{bin } (3 + x_1 + x_2 + x_3));$$

$$f_4 = !x_1 \vee !x_2 \vee !x_3 = r_3^3 (\text{bin } (6 - x_1 - x_2 - x_3)),$$

приведенную в примере 1 Приложения в [30].

Так как все функции этой системы линейно представимы, то

$$P = 798 + 455(x_1 + x_2 + x_3), \quad f_1 = r_{12}^{12}(\text{bin } P), \\ f_2 = r_{12}^9(\text{bin } P), \quad f_3 = r_{12}^6(\text{bin } P), \quad f_4 = r_{12}^3(\text{bin } P).$$

В [30] функции этой системы реализуются раздельно.

Пример 20.35. Реализовать систему линейно представимых функций

$$f_3 = x_1! x_2 x_3! x_4 = r_3^3(\text{bin}(2 + x_1 - x_2 + x_3 - x_4)); \\ f_2 = x_2 \vee x_3 x_4 = r_3^3(\text{bin}(2 + 2x_2 + x_3 + x_4)); \\ f_1 = x_1 \vee x_2 = r_2^2(\text{bin}(1 + x_1 + x_2)),$$

приведенную в примере 2 Приложения в [30].

Так как все функции системы линейно представимы, то, исходя из изложенного, $P = 73 + 33x_1 - 23x_2 + 36x_3 - 28x_4$, $f_3 = r_8^8(\text{bin } P)$, $f_2 = r_8^5(\text{bin } P)$, $f_1 = r_8^2(\text{bin } P)$. В [30] эта система реализуется существенно сложнее — двумя ЛАП с семью маскированиями.

Оптимизация полинома. Если полиномы при объединении «не портят» разрядов порождающего полинома, в которых формируются значения заданных функций, то последний может быть построен таким образом, что ему будет соответствовать число разрядов, меньшее, чем определяемое соотношениями (20.33), (20.34).

Пример 20.36. Реализовать простейшим ЛАПМ систему линейно представимых функций:

$$f_1 = x_1 x_2 \oplus x_3 = r_3^2(\text{bin}(x_1 + x_2 + 2x_3)); \\ f_2 = !x_1! x_2 x_3 \vee x_1 x_2! x_3 = r_3^2(\text{bin}(1 + 3x_1 - x_2 + x_3)).$$

Используя соотношение (20.33), можно получить два ЛАПМ с $g = 6$:

$$P_1 = 2^3(x_1 + x_2 + 2x_3) + 2^0(1 + 3x_1 - x_2 + x_3) = \\ = 1 + 11x_1 + 7x_2 + 17x_3, \\ f_1 = r_6^5(\text{bin } P_1), \quad f_2 = r_6^2(\text{bin } P_1), \quad |P_1| = 36; \\ P_2 = 2^3(1 + 3x_1 - x_2 + x_3) + 2^0(x_1 + x_2 + 2x_3) = \\ = 8 + 25x_1 - 7x_2 + 10x_3, \\ f_2 = r_6^5(\text{bin } P_2), \quad f_1 = r_6^2(\text{bin } P_2), \quad |P_2| = 43.$$

Последнее соотношение иллюстрируется табл. 20.6.

Т а б л и ц а 20.6

x_1	x_2	x_3	P_2	$2^3(1+3x_1-x_2+x_3)$			$2^0(x_1+x_2+2x_3)$		
				r_6^6	$f_2=r_6^5$	r_6^4	r_6^3	$f_1=r_6^2$	r_6^1
0	0	0	8	0	0	1	0	0	0
0	0	1	18	0	1	0	0	1	0
0	1	0	1	0	0	0	0	0	1
0	1	1	11	0	0	1	0	1	1
1	0	0	33	1	0	0	0	0	1
1	0	1	43	1	0	1	0	1	1
1	1	0	26	0	1	1	0	1	0
1	1	1	36	1	0	0	1	0	0

Запишем формулы для БФУ, приведенных в табл. 20.6:

$$r_6^6 = x_1(!x_2 \vee x_3),$$

$$r_6^5 = f_2 = !x_1!x_2x_3 \vee x_1x_2!x_3 = ((x_1 \vee x_2)x_3 \vee x_1x_2) \oplus x_3,$$

$$r_6^4 = 1 \oplus x_1 \oplus x_2 \oplus x_3, \quad r_6^3 = x_1x_2x_3,$$

$$r_6^2 = f_1 = x_1x_2 \oplus x_3, \quad r_6^1 = x_1 \oplus x_2.$$

При этом шестая и третья функции являются пороговыми, пятая и вторая — порогово-линейными, а четвертая и первая — линейными.

В силу того что арифметическое сложение значений «младшей» функции (r_6^4), порождаемой ЛАП для f_2 , и «старшей» функции (r_6^3), порождаемой ЛАП для f_1 , не образует переносов при всех входных наборах, полиномы для f_2 и f_1 могут быть объединены «плотнее», чем в соответствии с соотношением (20.33):

$$\begin{aligned} P_3 &= 2^2(1+3x_1-x_2+x_3) + 2^0(x_1+x_2+2x_3) = \\ &= 4 + 13x_1 - 3x_2 + 6x_3, \end{aligned}$$

$$f_2 = r_5^4(\text{bin } P_3), \quad f_1 = r_5^2(\text{bin } P_3), \quad |P_3| = 23.$$

Полученное соотношение иллюстрируется табл. 20.7.

Запишем формулы для БФУ, приведенных в табл. 20.7:

$$r_5^5 = x_1(!x_2 \vee x_3), \quad r_5^4 = f_2 = ((x_1 \vee x_2)x_3 \vee x_1x_2) \oplus x_3,$$

$$r_5^3 = S_{0,2,3}^3 = !x_1!x_2!x_3 \vee (x_1 \vee x_2)x_3 \vee x_1x_2 =$$

$$= (!x_1!x_2 \vee x_1x_2!x_3) \oplus x_3 = (x_1x_3 \vee !x_2) \oplus x_1 \oplus x_3,$$

$$r_5^2 = f_1 = x_1x_2 \oplus x_3, \quad r_5^1 = x_1 \oplus x_2.$$

Т а б л и ц а 20.7

x_1	x_2	x_3	P_3	r_5^5	$f_2 = r_5^4$	r_5^3	$f_1 = r_5^2$	r_5^1
0	0	0	4	0	0	1	0	0
0	0	1	10	0	1	0	1	0
0	1	0	1	0	0	0	0	1
0	1	1	7	0	0	1	1	1
1	0	0	17	1	0	0	0	1
1	0	1	23	1	0	1	1	1
1	1	0	14	0	1	1	1	0
1	1	1	20	1	0	1	0	0

При этом пятая функция является пороговой, четвертая, третья и вторая — порогово-линейными, а первая — линейной.

Пример 20.37. Реализовать простейшим ЛАПМ систему линейно представимых функций:

$$f_1 = x_1!x_3 \vee x_2 = r_3^3(\text{bin}(3 + x_1 + 2x_2 - x_3));$$

$$f_2 = x_1!x_3 \oplus x_2 = r_3^2(\text{bin}(1 + x_1 + 2x_2 - x_3)).$$

В силу того что и в этом случае отсутствуют переносы при «наложении» полиномов, объединим их более «плотно», чем в соответствии с соотношением (20.33):

$$P = 2^2(3 + x_1 + 2x_2 - x_3) + 2^0(1 + x_1 + 2x_2 - x_3) =$$

$$= 13 + 5x_1 + 10x_2 - 5x_3,$$

$$f_1 = r_5^5(\text{bin } P), \quad f_2 = r_5^2(\text{bin } P).$$

Приведенные примеры, а также ряд других, рассмотренных автором, позволяют сформулировать следующие гипотезы [31].

Г и п о т е з а 20.1. Линейный арифметический полином порождает только пороговые, порогово-линейные и линейные функции, причем крайняя левая функция в двоичном представлении результатов вычислений по этому полиному всегда пороговая, а крайняя правая — всегда линейная функция.

Г и п о т е з а 20.2. При $n \geq 3$, $g \geq 3$ в двоичном представлении результатов вычислений по ЛАПМ, построенному с помощью методов, изложенных в разд. 20.3, для реализации одной БФУ, принадлежащей одному из рассмотренных классов, в средних разрядах размещаются только порогово-линейные функции.

Вторую гипотезу иллюстрируют полиномы для функций f_2 и f_3 из примера 20.33 и полиномы для функций f_1 и f_2 из примера 20.36.

Еще один подход к оптимизации полиномов описан в [31].

Изложенный подход для рассматриваемых классов функций универсален, однако не всегда эффективен. Продемонстрируем это на примере.

Пример 20.38. Реализовать систему линейно представимых функций:

$$f_1 = x_1 \vee x_2 = r_2^2(\text{bin}(1 + x_1 + x_2)),$$

$$f_2 = x_1 \oplus x_2 = r_2^1(\text{bin}(x_1 + x_2)).$$

Используя предлагаемый метод, получим:

$$P = 2^2(1 + x_1 + x_2) + 2^0(x_1 + x_2) = 4 + 5x_1 + 5x_2,$$

$$f_1 = r_4^4(\text{bin } P), \quad f_2 = r_4^1(\text{bin } P).$$

С другой стороны, анализируя, какой кортеж функций реализует первый из заданных полиномов, можно установить, что он порождает и вторую функцию, и поэтому объединять полиномы нет необходимости.

Учитывая изложенное, предложенный подход может быть усовершенствован. При этом необходимо:

- реализовать линейным арифметическим полиномом с маскированием (ЛАПМ) каждую из заданных функций;
- для каждого ЛАПМ определить, какие функции он реализует;
- определить, не реализуются ли какие-либо другие функции системы полиномом, реализующим одну из функций системы, и в случае, если это имеет место, исключить из состава системы «поглощаемые» функции;
- объединить в один поглощающие ЛАПМ и ЛАПМ, соответствующие функциям, которые не поглощены.

Пример 20.39. Реализовать систему линейно представимых функций:

$$f_1 = x_1 x_2 = r_2^2(\text{bin}(x_1 + x_2)),$$

$$f_2 = x_1 \vee x_2 = r_2^2(\text{bin}(1 + x_1 + x_2)),$$

$$f_3 = x_1 \oplus x_2 = r_2^1(\text{bin}(x_1 + x_2)).$$

Используя соотношение (20.33), получим:

$$P_1 = 2^4(x_1 + x_2) + 2^2(1 + x_1 + x_2) + 2^0(x_1 + x_2) = 4 + 21(x_1 + x_2),$$

$$f_1 = r_6^6(\text{bin } P_1), \quad f_2 = r_6^4(\text{bin } P_1), \quad f_3 = r_6^1(\text{bin } P_1).$$

Применяя усовершенствованный метод оптимизации, получим:

$$P_2 = 2^2(x_1 + x_2) + 2^0(1 + x_1 + x_2) = 1 + 5x_1 + 5x_2,$$

$$f_1 = r_4^4(\text{bin } P_2), \quad f_2 = r_4^2(\text{bin } P_2), \quad f_3 = r_4^3(\text{bin } P_2).$$

При этом отметим, что, используя методы построения НАП, получим:

$$f_1 f_2 f_3 = \text{bin}(3x_1 + 3x_2).$$

Пример 20.40. Реализовать систему линейно представимых функций:

$$f_1 = (x_1 \vee x_2)x_3 \vee x_1 x_2 = r_2^2(\text{bin}(x_1 + x_2 + x_3));$$

$$f_2 = x_1 \oplus x_2 \oplus x_3 = r_2^1(\text{bin}(x_1 + x_2 + x_3)).$$

Из описания заданных функций следует, что

$$P = x_1 + x_2 + x_3, \quad f_1 = r_2^2(\text{bin } P), \quad f_2 = r_2^1(\text{bin } P).$$

Иногда вместо линейного АП с маскированиями целесообразно применить нелинейный АП, сложность которого, однако, априори не предсказуема. Так, например, кортеж $f_1 f_2 f_3$, где $f_1 = x_1(x_2 \vee x_3)$, $f_2 = x_1! x_3 \vee x_2$, $f_3 = x_1! x_3 \oplus x_2$, реализуется простым нелинейным арифметическим полиномом $P_1 = 3x_1 + 3x_2 + x_1 x_3$, в то время как простейший линейный АП с маскированиями, полученный «наложением» полиномов для функций f_2 и f_3 , существенно более сложен:

$$P_2 = 45 + 69x_1 + 42x_2 + 27x_3,$$

$$f_1 = r_8^8(\text{bin } P_2), \quad f_2 = r_8^5(\text{bin } P_2), \quad f_3 = r_8^2(\text{bin } P_2).$$

Несмотря на высказанные замечания, в целом можно утверждать, что в настоящем разделе для класса линейно представимых функций решена задача, поставленная в [5]: для заданной системы БФУ определить количество, вид и месторасположение дополнительных функций, упрощающих полином, реализующий эту систему.

Более того, для рассматриваемого класса функций решена и другая задача: каким образом для систем линейно представимых функций, если условие линейности не выполняется, без перебора определить количество, вид и месторасположение дополнительных функций, таких чтобы указанное условие выполнялось. При этом метод не требует введения этих функций при синтезе полинома в явном виде, а они могут быть определены в результате его анализа.

20.4.2. Реализация системы произвольных булевых функций системой линейных арифметических полиномов с маскированиями

Предлагаемый метод состоит в следующем.

В каждой функции системы, заданной формулой, выделяются пороговые, порогово-линейные и линейные фрагменты. Каждый новый фрагмент обозначается соответствующей буквой, и в исходной системе осуществляется замена выделенных фрагментов буквами. В результате формируется расширенная система булевых формул, состоящая из выделенных фрагментов и преобразованной исходной системы.

Все формулы новой системы БФ линейно представимы, и поэтому их группы могут быть реализованы соответствующими ЛАПМ, число которых не ограничено и зависит от способа выделения фрагментов.

Предлагаемый метод является универсальным, так как произвольная система БФУ всегда может быть заменена новой системой с требуемыми свойствами. Важная особенность предлагаемого подхода состоит в том, что сложность каждого из полиномов, зависящих от одних и тех же переменных (с точностью до величины коэффициентов) совпадает со сложностью объединяющего их полинома, и поэтому объединение, если оно возможно, всегда целесообразно.

Предлагаемый подход обеспечивает возможность параллельной, последовательной или смешанной реализации в зависимости от свойств заданной системы булевых формул.

Таким образом, метод, предложенный в [30], может рассматриваться как частный случай излагаемого ниже метода, во-первых, потому что фиксирует метод выделения фрагментов и ограничивает число ЛАПМ двумя, а во-вторых, так как позволяет выделять в качестве фрагментов формул, заданных в дизъюнктивных нормальных формах, лишь конъюнкции, являющиеся наряду с формируемыми дизъюнкциями только малой частью класса линейно представимых функций.

Перейдем к рассмотрению методов выделения фрагментов в системе булевых формул.

Рассмотрим вопрос о реализации системы произвольных функций двумя полиномами с маскированиями.

Теорема 20.6. Произвольная система булевых формул всегда может быть реализована не более чем двумя линейными арифметическими полиномами с маскированиями, первый из которых реализует все разнотипные линейно представимые фрагменты заданных формул, а второй — все дизъюнкции, формируемые в результате замены в исходной или преобразованной системе формул указанных фрагментов новыми буквами [31].

Нижняя оценка числа ЛАПМ (единица) относится к классу систем БФ, состоящих только из линейно представимых функций, а до-

казательство верхней оценки следует из результатов предыдущего раздела.

Метод, определяемый этой теоремой, строит либо меньшее число ЛАПМ по сравнению с методом, изложенным в [30], либо при одинаковом количестве ЛАПМ (два) строит более простые полиномы: первый из них имеет меньшие значения коэффициентов, а второй кроме меньших значений коэффициентов содержит еще и меньшее число переменных.

Пример 20.41. Реализовать формулу

$$f = (x_1 \vee x_2)(x_3 \vee x_4) \vee (x_5 \vee x_6)(x_7 \vee x_8).$$

Так как формула не является линейно представимой, то она не может быть записана одним ЛАПМ. Для реализации функции в виде двух ЛАПМ преобразуем формулу за счет частичного раскрытия скобок. При этом

$$\begin{aligned} f_1 &= (x_1 \vee x_2)x_3 = r_3^3(\text{bin}(1 + x_1 + x_2 + 2x_3)); \\ f_2 &= (x_1 \vee x_2)x_4 = r_3^3(\text{bin}(1 + x_1 + x_2 + 2x_4)); \\ f_3 &= (x_5 \vee x_6)x_7 = r_3^3(\text{bin}(1 + x_5 + x_6 + 2x_7)); \\ f_4 &= (x_5 \vee x_6)x_8 = r_3^3(\text{bin}(1 + x_5 + x_6 + 2x_8)); \\ f &= f_1 \vee f_2 \vee f_3 \vee f_4 = r_3^3(\text{bin}(3 + f_1 + f_2 + f_3 + f_4)). \end{aligned}$$

Используя метод, изложенный в предыдущем разделе, реализуем ЛАПМ формулы первого яруса. В результате получим:

$$\begin{aligned} P &= 585 + 576x_1 + 576x_2 + 1024x_3 + 128x_4 + \\ &\quad + 9x_5 + 9x_6 + 16x_7 + 2x_8, \\ f_1 &= r_{12}^{12}(\text{bin } P), \quad f_2 = r_{12}^9(\text{bin } P), \\ f_3 &= r_{12}^6(\text{bin } P), \quad f_4 = r_{12}^3(\text{bin } P), \\ f &= r_3^3(\text{bin}(3 + f_1 + f_2 + f_3 + f_4)). \end{aligned}$$

Пример 20.42. Реализовать систему булевых формул

$$f_1 = x_1x_2 \vee x_3x_4, \quad f_2 = x_1x_2 \vee !x_3!x_4.$$

Преобразуем заданную систему к требуемому виду:

$$\begin{aligned} f_3 &= x_1x_2 = r_2^2(\text{bin}(x_1 + x_2)), \quad f_4 = x_3x_4 = r_2^2(\text{bin}(x_3 + x_4)), \\ f_5 &= !x_3!x_4 = r_2^2(\text{bin}(2 - x_3 - x_4)), \\ f_1 &= f_3 \vee f_4 = r_2^2(\text{bin}(1 + f_3 + f_4)), \\ f_2 &= f_3 \vee f_5 = r_2^2(\text{bin}(1 + f_3 + f_5)). \end{aligned}$$

Реализуем формулы f_3, f_4, f_5 одним полиномом, а формулы f_1 и f_2 — другим:

$$\begin{aligned} P_1 &= 2 + 16x_1 + 16x_2 + 2x_3 + 2x_4, \\ f_3 &= r_6^6(\text{bin } P_1), \quad f_4 = r_6^4(\text{bin } P_1), \quad f_5 = r_6^2(\text{bin } P_1); \\ P_2 &= 5 + 5f_3 + 4f_4 + f_5, \\ f_1 &= r_4^4(\text{bin } P_2), \quad f_2 = r_4^2(\text{bin } P_2). \end{aligned}$$

«Ярусная» реализация систем булевых функций. В ряде случаев бывает целесообразно реализовать заданную систему булевых формул более чем двумя линейными арифметическими полиномами с маскированиями. Для этого реализуем заданную систему булевых формул M -ярусной схемой из элементов, описываемых линейно представимыми функциями. Тогда эта схема может быть реализована M линейными полиномами с маскированиями, причем в один полином могут быть объединены все элементы, отнесенные к одному ярусу схемы.

Пример 20.43. Реализовать формулу, рассмотренную в примере 20.41.

Эта формула реализуется трехъярусной схемой, для которой

$$\begin{aligned} f_1 &= x_1 \vee x_2, \quad f_2 = x_3 \vee x_4, \quad f_3 = x_5 \vee x_6, \quad f_4 = x_7 \vee x_8; \\ f_5 &= f_1 \& f_2, \quad f_6 = f_3 \& f_4; \\ f &= f_5 \vee f_6. \end{aligned}$$

Реализуем все формулы каждого яруса одним ЛАПМ. При этом

$$\begin{aligned} P_1 &= 85 + 64x_1 + 64x_2 + 16x_3 + 16x_4 + 4x_5 + 4x_6 + x_7 + x_8, \\ f_1 &= r_8^8(\text{bin } P_1), \quad f_2 = r_8^6(\text{bin } P_1), \\ f_3 &= r_8^4(\text{bin } P_1), \quad f_4 = r_8^2(\text{bin } P_1); \\ P_2 &= 4f_1 + 4f_2 + f_3 + f_4, \\ f_5 &= r_4^4(\text{bin } P_2), \quad f_6 = r_4^2(\text{bin } P_2); \\ f &= r_2^2(\text{bin } (1 + f_5 + f_6)). \end{aligned}$$

«Каскадная» реализация систем булевых функций. Во многих случаях бывает целесообразно применять не «ярусную», а «каскадную» реализацию.

К а с к а д о м будем называть фрагмент схемы (формулы), реализующий линейно представимую функцию. Частным случаем каскада является схема из двухвходовых элементов И и ИЛИ, соединенных последовательно, так как в [25] показано, что любой каскад

из таких элементов реализует неповторную пороговую формулу. Каскадом также является последовательная схема из элементов НЕ-РАВНОЗНАЧНОСТЬ. Таким образом, каждый каскад может быть реализован одним линейным арифметическим полиномом с маскированиями. Одним ЛАПМ могут быть также реализованы каскады, входящие в схемы для различных функций системы.

Пример 20.44. Реализовать формулу, рассмотренную в предыдущем примере.

Реализуем формулу четырехкаскадной схемой, каскады которой описываются следующим образом:

$$\begin{aligned} f_1 &= x_7 \vee x_8 = r_2^2(\text{bin}(1 + x_7 + x_8)); \\ f_2 &= (x_5 \vee x_6) f_1 = r_3^3(\text{bin}(1 + x_5 + x_6 + 2f_1)); \\ f_3 &= x_3 \vee x_4 = r_2^2(\text{bin}(1 + x_3 + x_4)); \\ f &= (x_1 \vee x_2) f_3 \vee f_2 = r_4^4(\text{bin}(5 + x_1 + x_2 + 3f_2 + 2f_3)). \end{aligned}$$

Сопоставляя результаты примеров 20.41, 20.43, 20.44, можно утверждать, что с увеличением числа полиномов при реализации одной и той же задачи значения коэффициентов уменьшаются. Полиномы из последнего примера могут рассматриваться как декомпозиция полиномов, найденных в двух предыдущих примерах.

Пример 20.45. Реализовать формулу

$$f = (!x_1 x_2 \vee x_1 !x_2 x_3) x_4 \oplus x_5.$$

Фрагмент в скобках является порогово-линейным. После замены его буквой заданная формула также становится порогово-линейной:

$$\begin{aligned} f_1 &= !x_1 x_2 \vee x_1 !x_2 x_3 = x_1(x_2 \vee x_3) \oplus x_2 = \\ &= r_4^3(\text{bin}(1 + 2x_1 + 5x_2 + x_3)); \\ f &= f_1 x_4 \oplus x_5 = r_3^2(\text{bin}(f_1 + x_4 + 2x_5)). \end{aligned}$$

Пример 20.46. Реализовать систему БФ из примера 20.42.

Эта система формул после преобразования к требуемому виду:

$$f_3 = x_1 x_2, \quad f_1 = f_2 \vee x_3 x_4, \quad f_2 = f_3 \vee !x_3 !x_4,$$

может быть реализована тремя линейными арифметическими полиномами с тремя маскированиями:

$$\begin{aligned} f_3 &= r_2^2(\text{bin}(x_1 + x_2)); \\ f_1 &= r_3^3(\text{bin}(2 + 2f_3 + x_3 + x_4)); \\ f_2 &= r_3^3(\text{bin}(4 + 2f_3 - x_3 - x_4)), \end{aligned}$$

или при объединении каскадов, соответствующих разным формулам, — двумя ЛАП также с тремя маскированиями:

$$\begin{aligned}f_3 &= r_2^2(\text{bin}(x_1 + x_2)), \\P &= 20 + 18f_3 + 7x_3 + 7x_4, \\f_1 &= r_6^6(\text{bin } P), \quad f_2 = r_6^3(\text{bin } P).\end{aligned}$$

Последняя реализация существенно более эффективна, чем рассмотренная в примере 20.42.

Пример 20.47. Реализовать систему булевых формул

$$f_1 = (!x_1x_2 \vee x_1!x_2x_3)x_4 \oplus x_5, \quad f_2 = (x_1 \oplus x_2 \oplus x_3)x_4.$$

Эта система формул после преобразования принимает вид:

$$\begin{aligned}f_3 &= !x_1x_2 \vee x_1!x_2x_3 = (x_1!x_3 \vee x_2) \oplus x_1, \\f_1 &= f_3x_4 \oplus x_5, \\f_4 &= x_1 \oplus x_2 \oplus x_3, \quad f_2 = f_4x_4,\end{aligned}$$

и поэтому может быть реализована четырьмя линейными арифметическими полиномами с четырьмя маскированиями:

$$\begin{aligned}f_3 &= r_4^3(\text{bin}(1 + 2x_1 + 5x_2 + x_3)), \\f_1 &= r_3^2(\text{bin}(f_3 + x_4 + 2x_3)), \\f_4 &= r_2^2(\text{bin}(x_1 + x_2 + x_3)), \\f_2 &= r_2^2(\text{bin}(f_4 + x_4)),\end{aligned}$$

или двумя ЛАП также с четырьмя маскированиями:

$$\begin{aligned}P_1 &= 4 + 9x_1 + 21x_2 + 5x_3, \quad f_3 = r_6^6(\text{bin } P_1), \quad f_4 = r_6^1(\text{bin } P_1), \\P_2 &= 4f_3 + f_4 + 5x_4 + 8x_5, \quad f_1 = r_5^4(\text{bin } P_2), \quad f_2 = r_5^2(\text{bin } P_2).\end{aligned}$$

«Ярусно-каскадная» реализация систем булевых функций. В ряде случаев бывает целесообразно использовать смешанную реализацию, при которой каскады одного яруса вычисляются параллельно.

Пример 20.48. Реализовать булеву формулу

$$f = (!x_1x_2 \vee x_1!x_2x_3)x_4 \oplus x_5.$$

Построим по этой формуле схему из двух пороговых (каскады первого яруса) и одного порогово-линейного (каскад второго уровня) «элементов»:

$$f_1 = !x_1 x_2, \quad f_2 = x_1 !x_2 x_3, \quad f = (f_1 \vee f_2) x_4 \oplus x_5.$$

Каскады первого яруса реализуются параллельно следующим ЛАПМ:

$$\begin{aligned} P &= 2^3(1 - x_1 + x_2) + 2^0(2 + x_1 - x_2 + x_3) = \\ &= 10 - 7x_1 + 7x_2 + x_3, \\ f_1 &= r_5^5(\text{bin } P), \quad f_2 = r_5^3(\text{bin } P), \end{aligned}$$

а каскад второго уровня — полиномом вида:

$$f = r_4^3(\text{bin}(1 + f_1 + f_2 + 2x_4 + 4x_5)).$$

Таким образом, формула в этом случае реализуется двумя линейными арифметическими полиномами с тремя маскированиями, в то время как учет того факта, что фрагмент

$$!x_1 x_2 \vee x_1 !x_2 x_3$$

является порогово-линейным, позволяет реализовать формулу существенно проще — двумя более простыми линейными арифметическими полиномами с двумя маскированиями (пример 20.45).

20.4.3. Реализация системы произвольных булевых функций одним линейным полиномом с условными маскированиями и операторами присваивания

В предыдущих разделах было показано, что только системы из линейно представимых булевых формул могут быть реализованы одним линейным полиномом с маскированиями.

Покажем, что, расширяя базис используемых операций, можно обеспечить линейную представимость и для системы произвольных булевых формул.

Будем называть выражение $\text{if}(r_m^k(\text{bin } P))$ условным маск и р о в а н и е м.

При этом справедлива теорема.

Т е о р е м а 20.7. Произвольная система N булевых функций, каждая из которых зависит не более чем от n переменных, всегда может быть реализована либо одним линейным арифме-

тическим полиномом n переменных с N масками, либо одним линейным арифметическим полиномом, N операторами $f_i = 0$, M_1 операторами $f_i = 1$ ($i = 1, \dots, N$) и M_2 условными масками, где M_1 — число линейно представимых фрагментов в системе; M_2 — число различных линейно представимых фрагментов в системе.

Доказательство первой части теоремы, справедливой для систем линейно представимых функций, следует из материала разд. 20.4.1, а второй — из материала, изложенного в разд. 20.4.2, и отличается от последнего заменой второго ЛАП с N масками на N операторов $f_i = 0$, M_1 операторов $f_i = 1$ и M_2 условных масок.

При этом фрагмент реализации, состоящий из условных масок и операторов присваивания, предлагается строить с помощью метода независимых фрагментов (разд. 16.3).

Пример 20.49. Реализовать СБФ, приведенную в примере 20.42.

Обозначая $f_3 = x_1 x_2$, $f_4 = x_3 x_4$, $f_5 = !x_3 !x_4$ и используя метод независимых фрагментов, построим обобщенный оператор структуры:

$$F = !f_1 | !f_2 | f_3 f_1 | f_4 f_1 | f_3 f_2 | f_5 f_2,$$

который минимизируется следующим образом:

$$F = !f_1 | !f_2 | f_3(f_1 | f_2) | f_4 f_1 | f_5 f_2.$$

Полученный оператор структуры однозначно задает структуру фрагмента с условными масками. Используя этот оператор и результаты примера 20.42, получим:

$$P = 2 + 16x_1 + 16x_2 + 2x_3 + 2x_4;$$

$$f_1 = 0; f_2 = 0;$$

$$\text{if } (r_6^6(\text{bin } P)) \{f_1 = 1; f_2 = 1;\}$$

$$\text{if } (r_6^4(\text{bin } P)) f_1 = 1;$$

$$\text{if } (r_6^2(\text{bin } P)) f_2 = 1.$$

Таким образом, заданную систему булевых формул вместо двух ЛАП и пяти масок удалось реализовать двумя операторами $f_i = 0$, одним ЛАП, тремя условными масками и четырьмя операторами $f_i = 1$.

Из изложенного следует, что реализация системы булевых формул, определяемая этой теоремой, является весьма элегантной и достаточно стандартной, а в некотором смысле, и неупрощаемой, так как определяет возможность реализации произвольной системы булевых формул как бы одним пороговым элементом с «добавкой». Для рассмот-

ренного примера эта «добавка» состоит из маскирования, операции bin , одного оператора $f_i = 0$, трех операторов $f_i = 1$ и двух условных маскирований. Изложенный подход впервые описан в [31].

В заключение отметим, что первыми из известных автору работ по рассматриваемой в настоящей главе тематике являются работы [32, 33], а последней — [34].

Выводы

1. Рассмотрен аналитический метод построения арифметических полиномов (АП) для произвольных одиночных булевых формул (БФ).
2. Предложен метод построения арифметических полиномов для одиночных симметрических функций.
3. Предложен метод представления булевых формул арифметическими выражениями, использующими операции «sign» и «абсолютная величина».
4. Предложены прямое и обратное матричные преобразования между системами булевых функций и арифметическими полиномами, являющиеся разновидностью дискретных ортогональных преобразований.
5. Установлено условие линейности — условие представимости системы булевых функций линейным арифметическим полиномом (ЛАП).
6. Установлено условие представимости булевой функции линейным полиномом Жегалкина.
7. Предложены прямое и обратное преобразования между арифметическими выражениями с линейными показателями степени и арифметическими полиномами. Таким образом, установлена связь спектральных и арифметических представлений булевых функций и систем булевых функций.
8. Предложены алгоритмические структуры при $n = 2, \dots, 5$, позволяющие реализовать произвольную систему булевых функций n переменных с помощью ЛАП. При этом, в частности, показано, что при $n = 5$ произвольная система булевых функций может быть реализована с помощью десяти ЛАП и пяти предикатов, которые по эквивалентной сложности проще, чем десять пороговых элементов. Это является принципиально новым результатом для теории пороговых функций.
9. Предложены методы реализации пороговых, линейных и порогово-линейных булевых функций линейными арифметическими полиномами с маскированием. Эти классы функций названы линейно представимыми.
10. Изложен метод реализации произвольной системы из N линейно представимых функций одним ЛАП с N маскированиями, что

существенно эффективнее, чем реализация таких систем схемами из пороговых элементов.

11. Высказана гипотеза, что линейный арифметический полином порождает только пороговые, порогово-линейные и линейные функции, причем крайняя левая функция в двоичном представлении результатов вычислений по этому полиному всегда пороговая, а крайняя правая — всегда линейная.

12. Предложен более эффективный, чем известный, метод реализации произвольной системы БФУ двумя ЛАП с маскированиями.

13. Предложен метод реализации произвольной системы БФУ одним ЛАП с условными маскированиями и операторами присваивания.

Л и т е р а т у р а

1. Артюхов В. Л., Кондратьев В. Н., Шалыто А. А. Реализация булевых функций арифметическими полиномами // Автоматика и телемеханика. 1988. № 4.
2. Артюхов В. Л., Беляев Н. С., Шалыто А. А. Методы программной реализации симметрических функций // Проблемы комплексной автоматизации судов. ВНТО им. акад. А. Н. Крылова. Вып. 19. Л.: Судостроение, 1989.
3. Шалыто А. А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
4. Шалыто А. А. Реализация алгоритмов судовых управляющих логических систем при использовании микропроцессорной техники. Л.: ИПК СП, 1988.
5. Малюгин В. Д. Реализация булевых функций арифметическими полиномами // Автоматика и телемеханика. 1982. № 4.
6. Малюгин В. Д. Применение алгебры кортежей логических функций // X Всесоюз. совещ. по проблемам управления: Тез. докл. Кн.1. М.: Ин-т проблем управления, 1986.
7. Шмерко В. П., Кухарев Г. А. Метод и алгоритмы оперативного решения систем булевых уравнений, заданных в полиномиальных формах // X Всесоюз. совещ. по проблемам управления: Тез. докл. Кн.1. М.: Ин-т проблем управления, 1986.
8. Good I. J. The relationship between two fast Fourier transforms // IEEE Trans. on Computers. 1971. N 3.
9. Малюгин В. Д., Кухарев Г. А., Шмерко В. П. Преобразования полиномиальных форм булевых функций: Препринт. М.: Ин-т проблем управления, 1986.
10. Сингер М. Мини-ЭВМ RDP-11. Программирование на языке ассемблера и организация машины. М.: Мир, 1984.
11. Осипов В. А., Шалыто А. А., Кондратьев В. Н. Программная реализация алгоритмов логического управления в судовых системах. Методические разработки. Л.: ИПК СП, 1988.
12. Жегалкин И. И. О технике вычисления предложений в символической логике // Математический сборник. Т. 34. 1927.
13. Reed I. S. A class of multiple-error-correcting codes at the decoding scheme // Trans. IRE. 1954. N 4.
14. Авсаркисян Г. С., Брайловский Г. С. Представление логических функций в виде полиномов Жегалкина // Автоматика и вычисл. техника. 1975. № 6.
15. Шалыто А. А., Кондратьев В. Н. Методы программной реализации алгоритмов логического управления для судовых микропроцессорных систем. Л.: ИПК СП, 1990.
16. Кондратьев В. Н., Шалыто А. А. Реализация систем булевых функций с использованием линейных арифметических полиномов // Автоматика и телемеханика. 1993. № 3.

17. Головина Л. И. Линейная алгебра и некоторые ее приложения. М.: Наука, 1985.
18. Воробьев Н. Н. Числа Фибоначчи. М.: Наука, 1978.
19. Артюхов В. Л., Кондратьев В. Н., Шалыто А. А. Использование линейных арифметических полиномов при реализации систем логического управления // XI Всесоюз. совещ. по проблемам управления: Тез. докл. М.: Наука, 1989.
20. Кондратьев В. Н., Шалыто А. А. Реализация алгоритма управления посадкой плавучей полупогружной буровой установки линейными арифметическими полиномами // VII Всесоюз. научно-техн. конф. «Проблемы комплексной автоматизации судовых технических средств»: Тез. докл. Л.: ВНТО им. акад. А. Н. Крылова, 1989.
21. Прангишвили И. В., Вепхвадзе А. Н., Малюгин В. Д. и др. Векторная вычислительная система ПС-320 // Микропроцессорные средства и системы. 1986. № 5.
22. Кондратьев В. Н., Шалыто А. А. Реализация булевых функций одним линейным арифметическим полиномом с маскированием // Автоматика и телемеханика. 1996. № 1.
23. Бутаков Е. А. Методы синтеза релейных устройств из пороговых элементов. М.: Энергия, 1970.
24. Боголюбов И. Н., Овсиевич Б. Л., Розенблюм Л. Я. Пороговая и мажоритарная логика: Информ. листок. 1964. № 2 (29).
25. Артюхов В. Л., Розенблюм Л. Я., Шалыто А. А. Логические возможности некоторых типов каскадных структур // Сети связи и дискретные устройства управления. М.: Наука, 1976.
26. Артюхов В. Л., Кузнецов Б. П., Шалыто А. А. Настраиваемые логические устройства для судовых управляющих систем. Л.: ИПК СП, 1986.
27. Akers S.B. Binary decision diagrams // IEEE Trans. on Computers. 1978. N 6.
28. Malyugin V. Realization of logical function sets by arithmetic polynoms // Computers and artificial intelligence. 1987. N 6.
29. Ефремов В. Д., Кузьмин А. А., Степанов В. А. Вычисление логических функций с использованием преобразований Радемахера // Автоматика и телемеханика. 1984. № 2.
30. Малюгин В. Д. Реализация кортежей булевых функций посредством линейных арифметических полиномов // Автоматика и телемеханика. 1984. № 2.
31. Кондратьев В. Н., Шалыто А. А. Реализация систем булевых функций линейными арифметическими полиномами с маскированиями // Автоматика и телемеханика. 1997. № 3.
32. Синтез электронных вычислительных и управляющих схем / Пер. с англ. под ред. В. И. Шестакова. М.: Иностран. лит., 1954.
33. Тошич Ж. Арифметические представления логических функций // Дискретные автоматы и сети связи. М.: Наука, 1970.
34. Малюгин В. Д. Параллельные логические вычисления посредством арифметических полиномов. М.: Наука, 1997.

Глава 21

SWITCH-технология.

Алгоритмизация и программирование для систем логического управления и «реактивных» систем

Современные системы логического управления состоят из двух составляющих: аппаратной и программной.

Особенность аппаратуры состоит в том, что она разрабатывается одними специалистами, а изготавливается (комплексируется) — другими. Это приводит к необходимости проводить разработку в форме *проектирования*, связанного с выпуском большого числа разнотипных схем и других конструкторских документов, соответствующих действующим стандартам и досконально отражающих все аспекты жизненного цикла аппаратуры. Поэтому даже через много лет эта часть систем при необходимости может быть сравнительно легко модифицирована, в том числе и другими специалистами.

Принципиально иная ситуация имеет место при создании программного обеспечения, так как оно разрабатывается и «изготавливается» одними и теми же специалистами. Поэтому обычно разработка программ не выполняется в форме *проекта* той же степени подробности, как это делается для аппаратуры, что часто приводит к значительным трудностям при необходимости их модификации.

В настоящей главе излагаются основы проектирования алгоритмического и программного обеспечения для систем рассматриваемых классов.

Известен международный стандарт [1], определяющий языки программирования для программируемых логических контроллеров (ПЛК) и программно реализованных ПЛК (РС-контроллеров) — промышленных (управляющих) компьютеров (обычно IBM PC совместимых) с программным обеспечением класса SoftPLC или Soft-Logic для создания прикладных программ.

Известны также языки программирования для микроконтроллеров и собственно промышленных (управляющих) компьютеров [2].

Однако ведущими в области автоматизации фирмами мира [1—7] до сих пор не был выбран (разработан) язык алгоритмизации для систем логического (основанного на истинности и ложности) управления, который позволил бы:

- специалистам различных областей знаний однозначно и просто понимать, что должно быть сделано, что делается и что сделано в программно реализуемом проекте;

- формально и изоморфно переходить от алгоритма к программам на различных языках программирования, используя минимальное число внутренних (управляющих) переменных в программах, так как эти переменные затрудняют понимание программ;

- легко и корректно вносить изменения в разработанные алгоритмы и в построенные по ним программы;

- корректно проводить сертификацию программ.

Отсутствие такого языка оставляло открытым вопрос о создании сквозной технологии алгоритмизации и программирования для систем логического управления, которая позволила бы повысить качество проектирования их программного обеспечения.

В [8] выполнен подробный анализ известных технологий алгоритмизации и программирования для систем этого класса, на основе которого в [8] разработана новая технология, названная «SWITCH-технология», позволяющая обеспечить выполнение указанных выше требований. Эта технология может быть названа также «STATE-технология» или, более точно, «AUTOMATON-технология». Изложим основные положения этой технологии.

1. В качестве основного в предлагаемой технологии используется понятие «внутреннее состояние» (в дальнейшем — «состояние»).

Состояния рассматриваются как некоторые абстракции, вводимые в начале процесса алгоритмизации, например путем однозначного сопоставления каждого из них с одним из физических состояний управляемого объекта, так как обычно «функционирование производственных систем проявляется через изменение их состояний» [9]. При этом каждое состояние в алгоритме поддерживает объект в соответствующем состоянии, а переход в новое состояние в алгоритме приводит к переходу объекта в новое соответствующее состояние, что и обеспечивает процесс логического управления объектом.

Например, объект «клапан» может находиться в одном из четырех рабочих состояний («закрыт», «открывается», «открыт», «закрывается»), каждое из которых может поддерживаться соответствующим состоянием в алгоритме управления. Для клапана «с памятью» алгоритм управления может иметь и три состояния, поддерживая закрытое и открытое состояния объекта одним своим состоянием (разд. 5.4.1 в [8]).

При необходимости в алгоритм управления могут быть введены также и другие состояния, связанные, например, с неисправностями объекта и неправильными действиями Оператора.

Связь состояний с внутренними (управляющими) переменными появляется в дальнейшем на этапе кодирования состояний, отсутствующем в традиционном программировании. При этом число вводимых управляющих переменных зависит от принятого вида кодирования.

Такой подход, известный из теории автоматов, принципиально отличается от подхода, обычно применяемого в программировании, при котором в ходе процесса программирования по необходимости вводятся внутренние (обычно двоичные) переменные, а затем каждый различный набор их значений объявляется состоянием программы [10]. Однако, так как понятие «состояние» в программировании прикладных задач обычно не используется, ответ на вопрос о количестве состояний в программе, содержащей, например, n двоичных внутренних переменных, остается в большинстве случаев открытым. Отметим, что в этом случае число состояний может находиться в диапазоне от n до 2^n . При этом остается неясным также, откуда «берутся» эти переменные, сколько их должно быть и для какой цели каждая из них применяется. Отметим также, что при циклическом выполнении программы (ввиду наличия обратной связи от выхода к входу) она может быть последовательностной даже и при отсутствии управляющих переменных (гл.13 в [8]).

2. Добавляя к понятию «состояние» понятие «входное воздействие», естественным образом вводится понятие «автомат без выхода» (автомат без выхода = состояния + входные воздействия), а после введения понятия «выходное воздействие», под которым понимается «действие» («action») и «деятельность» («activity»), эта формула приобретает вид: автомат = состояния + входные воздействия + выходные воздействия.

При этом соответствующая область программирования может быть названа «автоматным программированием», а процесс создания программ в этом случае — «автоматным проектированием программ».

Так же как и в [11, 12], предполагается, что действие является однократным, мгновенным и непрерываемым, а деятельность может длиться достаточно долго и возможно ее прерывание некоторым входным воздействием.

Входное воздействие в общем случае может изменить состояние; инициировать выходное воздействие без изменения состояния; инициировать выходное воздействие и изменить состояние.

3. В рассматриваемой технологии в качестве основной математической модели используется «конечный детерминированный автомат» (в дальнейшем — «автомат»), понимая под ним по класси-

фикации [13] автомат с внутренними состояниями, а не автомат с функциями поведения (приложение 13 в [8]).

4. В качестве структурных моделей (гл. 3 в [8]) применяются автоматы без выходного преобразователя, автоматы Мура, автоматы Мили и смешанные автоматы (С-автоматы или автоматы Мура—Мили) [14, 15].

В качестве основной структурной модели предлагается использовать автоматы Мура, в которых коды состояний и значения выходов принципиально разделены, а значения выходных переменных в каждом состоянии не зависят от входных воздействий, что упрощает внесение изменений в описания таких автоматов. Свойства алгоритмов и программ, обеспечивающие упрощение внесения корректных изменений в них, названы в [8] «управляемостью».

Первоначально число состояний в автомате Мура может быть выбрано равным числу состояний в управляемом им объекте (включая его неисправные состояния при необходимости). В дальнейшем в автомат могут вводиться дополнительные состояния, связанные, например, с неправильными действиями Оператора (приложение 12 в [8]), а после этого число состояний в автомате может минимизироваться за счет объединения эквивалентных состояний или перехода к другой структурной модели, что, правда, без крайней необходимости делать нецелесообразно.

В автомате Мура значения выходных переменных сохраняются в течение времени его пребывания в соответствующем состоянии за счет памяти, реализующей эти состояния. В автомате Мили значения указанных переменных формируются в том числе и на соответствующем переходе, и, следовательно, для их «длительного» хранения требуется память, отличная от реализующей состояния. Поэтому в указанном смысле эти автоматы не эквивалентны.

5. Применяются автоматы первого и второго рода [8, 16]. Приоритет отдается автоматам второго рода, у которых новое состояние и значения выходных переменных формируются без задержки на «такт» (в одном программном цикле).

6. Используются различные виды кодирования состояний автоматов (разд. 3.3 в [8]): принудительное, принудительно-свободное, двоичное, двоичное логарифмическое и многозначное. Приоритет отдается многозначному кодированию состояний. Возможность применения в автоматах (ввиду наличия только одного активного состояния в каждом из них) многозначного кодирования состояний позволяет отказаться от традиционной точки зрения на автоматы как на частный случай сетей Петри, так как в последних из-за принципиальной необходимости обеспечения в общем случае одновременной активности нескольких позиций одной сети отсутствует возможность использования одной переменной для их кодирования. При этом если для автомата вне зависимости от числа состояний для их отличия достаточно только одной многозначной

переменной, то, например, для частного случая сети Петри, в которой в каждой позиции не может находиться более одной метки (аналог диаграммы «Графсет» по стандарту NFC-03-190 Grafset), число двоичных переменных, требующихся для отличия позиций, равно числу последних. В разд.12.3 в [8] показано, что диаграмма «Графсет» с параллельными «участками» может быть заменена системой взаимосвязанных графов переходов. При этом если в первом случае число указанных двоичных переменных равно числу позиций, то во втором — число многозначных переменных равно числу графов переходов вне зависимости от числа вершин в них.

7. В качестве алгоритмической модели для автоматов с памятью при их программной реализации применяется такой непроедурный визуальный формализм, разработанный в теории автоматов, как графы переходов, которые в литературе называются также диаграммами состояний, диаграммами состояний и переходов, диаграммами переходов («State Diagram», «State Transition Diagram», «Transition Diagram» соответственно в англоязычной литературе).

При этом отметим, что в этих названиях, так же как и в предлагаемой технологии, отдается приоритет понятию «состояние», а не понятию «событие», являющемуся одним из наиболее распространенных в современном программировании. В излагаемом подходе событие вторично и наряду с входной переменной рассматривается в качестве разновидности входного воздействия, которое может изменить состояние.

В другом визуальном формализме (схемы (граф-схемы) алгоритмов и программ), предложенном в теоретическом программировании и являющемся процедурным, понятие «состояние» в явном виде не используется, что резко усложняет его понимаемость [17]. Это понятие не применяется и в языке регулярных выражений алгебры событий [18].

Отметим также, что графы переходов имеют плоскостное изображение, а не вертикальную направленность (как, например, схемы алгоритмов, SDL-диаграммы [8] и диаграммы «Графсет»), что существенно повышает их обозримость.

Они значительно более компактны по сравнению с эквивалентными схемами из функциональных блоков (традиционных логических элементов) и более обозримы, так как взаимодействие между графами переходов выполняется по д а н н ы м, а между этими схемами — по у п р а в л е н и ю.

8. Одно из достоинств графов переходов, которые должны быть «максимально» планарными, состоит в том, что на каждой его дуге указываются не все входные воздействия (например, в форме минтермов), а только те из них, которые обеспечивают переход по этой дуге. На каждой дуге входные воздействия могут объединяться в булевы формулы, в том числе и произвольной глубины, что резко повышает компактность описания алгоритма.

Введение в модель булевых формул, как и при аппаратной реализации (структурном синтезе последовательностных схем), расширяет классическую модель абстрактного конечного автомата, в которой дуги помечаются лишь отдельными буквами входного алфавита, и обеспечивает «параллельную» обработку входных воздействий. По сравнению с последовательной по входам программной реализацией такое описание алгоритмов в общем случае позволяет уменьшить число состояний в реализующих их автоматах [8].

Можно утверждать, что каждое состояние в графе переходов «выделяет» из множества всех входных воздействий только то их подмножество, которое обеспечивает переходы из этого состояния — декомпозирует указанное множество.

Изложенное позволяет применять графы переходов для решения задач логического управления при большом числе входных воздействий, значительно упрощая решение проблемы тестирования по всем путям.

9. Каждый граф переходов должен быть семантически и синтаксически корректен. Первое свойство определяет, *п р а в и л ь н а* ли построена модель, а второе — *п р а в и л ь н о* ли она построена [19]. При проверке синтаксической корректности граф переходов проверяется на достижимость, полноту, непротиворечивость, реализуемость и отсутствие генерирующих контуров. При проверке *д о с т и ж и м о с т и* определяется наличие по крайней мере одного пути в графе переходов из любой его вершины в любую другую. *П о л н о т а* [8] проверяется для каждой вершины графа переходов и позволяет, в частности для автоматов Мура, умалчивать пометку ее петли. При обеспечении *н е п р о т и в о р е ч и в о с т и* [8] каждой вершины графа переходов вместо ортогонализации могут расставляться приоритеты, что уменьшает сложность реализующей его программы. *Р е а л и з у е м о с т ь* обеспечивается за счет различения одинаково помеченных вершин в исходном графе переходов (кодирование состояний автомата). *Г е н е р и р у ю щ и е к о н т у р ы* [8] в графе переходов устраняются, например, за счет ортогонализации пометок дуг, образующих соответствующие контуры.

10. Дальнейшее расширение модели абстрактного конечного автомата и обеспечение «параллелизма» по выходам, как и при аппаратной реализации, достигается за счет указания в компонентах (в вершинах и/или на дугах) графа переходов не отдельных букв выходного алфавита, а значений выходных переменных (деятельностей), формируемых в этих компонентах. При этом непосредственное указание в каждой компоненте (в зависимости от выбранной структурной модели автомата) значений *в с е х* выходных переменных резко упрощает понимание алгоритма, описанного графом переходов, возможно за счет увеличения числа вершин в нем. По-

построенный таким образом граф переходов задает структурный конечный автомат без умолчаний значений выходных переменных. При этом отметим, что, например, для графа переходов автомата Мура при необходимости сокращения объектного кода в программе, его реализующей, неизменяющиеся при соседних переходах значения выходных переменных, указанные в вершине, в которую выполняются переходы, должны быть закоментированы.

Еще большее расширение модели конечного автомата достигается (при уменьшении ее понятности) за счет указания в компонентах графа переходов не только значений выходных переменных, но и булевых формул (возможно, автоматных, что соответствует вложенным автоматам) (разд. 12.4 в [8]).

11. «Параллелизм» по входам и выходам позволяет даже с помощью одного автомата, являющегося по определению последовательностным по состояниям (в соответствующем ему графе переходов в каждый момент времени активной является только одна вершина), реализовать параллельные процессы.

12. Поведение автоматов с памятью в отличие от автоматов без памяти зависит от предыстории, сводящейся к тому, что каждый переход в некоторое состояние зависит от одного, непосредственно связанного с ним состояния, а значения выходов, например для автоматов Мура, зависят только от состояния, в котором находится автомат.

В рамках предлагаемой технологии эти свойства автоматов с памятью должны быть сохранены, а для обеспечения независимости от глубокой предыстории по состояниям и выходам графы переходов не должны содержать флагов и умолчаний соответственно [8, 17]. При этом число вершин в графе переходов должно совпадать с числом состояний в автомате. Независимость от глубокой предыстории («будущее зависит от настоящего и не зависит от прошлого») существенно упрощает понимание [20] поведения автомата и внесение изменений как в граф переходов, так и в реализующую его программу.

Аналогичная ситуация имеет место и в марковских процессах, для исследования которых разработан весьма простой математический аппарат, который становится существенно более трудоемким для процессов, зависящих от глубокой предыстории.

13. При формальном и изоморфном переходе к программе от графа переходов без флагов и умолчаний последний является сертификационным тестом для ее проверки. При этом появляется возможность проводить сертификацию, начиная со сверки каждого графа переходов с изоморфным ему фрагментом программы. Вопросы тестирования и верификации программ, построенных с использованием конечных автоматов («Finite State Machine» или «State Machine» в англоязычной литературе), рассмотрены в [21—25, 170,

178, 179]. Кроме этих работ в пятом номере журнала «Computer» за 2000 г. описывается подход к практической верификации встроеного программного обеспечения, основанной на расширенной модели конечных автоматов, которые применяются в программном продукте «VisualState». В рамках этого подхода проверяется достижимость состояний верифицируемой системы, которая, однако, задается неявно.

14. При использовании многозначного кодирования состояний для каждого автомата с любым (конечным) их числом достаточно одной внутренней переменной, значность которой равна числу состояний.

При этом отметим, что так как (по крайней мере теоретически) любой алгоритм логического управления может быть реализован одним графом переходов, то вне зависимости от числа вершин в нем он может быть реализован программой, содержащей только одну внутреннюю (управляющую) переменную.

Отметим также, что при каждом переходе предыдущее значение многозначной переменной сбрасывается автоматически, а кроме того, ей «не с чем» состязаться.

Применять такое кодирование можно лишь в том случае, когда априори (в начале процесса алгоритмизации) известно, какое число состояний будет иметь автомат.

15. Работа с многозначными переменными поддерживается большинством языков программирования, в том числе и таким «экзотическим» языком, как язык функциональных блоков [8, 168].

Граф переходов с многозначным кодированием вершин, соответствующий выбранной структурной модели автомата, формально и изоморфно реализуется одной или двумя конструкциями `switch` языка СИ (гл. 5 в [8]) или их аналогами в других языках программирования, что и определило название предлагаемой технологии, а кроме того, слово `switch` (переключатель) ассоциируется с теорией переключательных схем, являющейся основой теории логического управления.

Возможны два подхода к реализации графов переходов, которые могут быть обозначены следующим образом: «состояния—события» и «события—состояния».

В первом случае при реализации графа переходов используются одна или две конструкции `switch` по «состояниям», а во втором — каждому типу «событий» сопоставляется функция, описываемая конструкцией `switch` по «состояниям», отражающая переходы, инициируемые этим «событием» [26].

Если событиям соответствуют значения многозначной переменной (функции, формирующей эти значения), то в первом случае первичной является конструкция `switch` по «состояниям», в которую вложены конструкции `switch` по «событиям», а во втором — конструкция `switch` по «событиям», в которую вложены конструкции `switch` по «состояниям» (приложение 11 в [8]). При этом отметим,

что в первом случае принцип построения программ совпадает с принципом логического управления (переход из состояния в состояние под воздействием события), что резко упрощает понимание таких программ Пользователем. Отметим также, что в «развитых» науках (например, физике) понятие «пространство состояний» является одним из основополагающих [175], в то время как понятие «поток событий» таковым в этих науках не является. Например, для воды определяющими являются состояния (жидкое, твердое, газообразное), а условия переходов и тем более выполняемые ею действия вторичны. Понятие «событие» широко применяется в истории.

16. Значение многозначной переменной полностью характеризует «положение» программы, реализующей один граф переходов, в пространстве ее состояний, что позволяет ввести в программирование понятие «наблюдаемость» (по одной внутренней переменной). При этом программа рассматривается как «белый ящик» с одним внутренним параметром. Сертификация программы, построенной формально и изоморфно, например по графу переходов автомата Мура без флагов и умолчаний, может проводиться в два этапа: 1) проверка реализации всех переходов с наблюдением при этом за значениями только одной внутренней переменной; 2) проверка значений всех выходных переменных в каждом состоянии.

Графы переходов автоматов других типов всегда могут быть сведены к графам переходов эквивалентных автоматов Мура. При этом, например, каждой вершине графа переходов автомата Мили без флагов и умолчаний сопоставляются вершины графа переходов эквивалентного автомата Мура, число которых равно числу дуг, входящих в рассматриваемую вершину и помеченных различными значениями выходных переменных (разд. 4.4.3 в [8]).

Анализ всех функциональных возможностей графа переходов может выполняться с помощью графа достижимых маркировок, под которым будем понимать граф достижимых состояний (ГДС), каждой вершине которого в дальнейшем могут быть однозначно приписаны выполняемые в ней деятельности. Поэтому граф переходов автомата Мура (или автомата без выходного преобразователя) без флагов и умолчаний может использоваться в качестве ГДС, а для других типов автоматов графы переходов и ГДС различаются между собой. Так, например, для графа переходов автомата Мили без флагов и умолчаний в качестве ГДС может применяться граф переходов эквивалентного автомата Мура.

17. Из изложенного в предыдущем пункте следует, что предлагаемый подход позволяет снять проблему, названную в [27] расшифровкой автоматов, в [28] — распознаванием автоматов, а в [29] — идентификацией автоматов, так как считается, что автомат распознан (расшифрован, идентифицирован), если для него построен граф переходов.

В [27] показано, что автомат нераспознаваем, если он является «абсолютно черным ящиком», для которого отсутствует какая-либо информация о его внутренних состояниях. Таким образом, для автоматов с памятью тестирование типа «вход—выход», наиболее часто используемое при создании систем логического управления, проблеме распознавания не решает и говорить при этом о гарантированном поведении системы управления не приходится, что, видимо, неизвестно авторам работы [167].

В [29] показано, что конечный детерминированный автомат распознаваем в результате анализа последовательности «вход—выход», если известно максимальное число состояний в минимальной (по числу состояний) форме этого автомата, а его граф переходов является сильносвязанным.

В [27] автомат, для которого известна указанная оценка числа состояний, назван «относительно черным ящиком».

18. Из изложенного в предыдущем пункте следует, что если понятие «состояние» априори не вводится, то в результате тестирования алгоритмов и программ с помощью входо-выходных последовательностей в общем случае не удастся распознать автоматы, которые они реализуют.

В предлагаемой технологии состояния вводятся априори, а графы переходов строятся как сильносвязанные, что снимает проблему распознавания.

В рамках SWITCH-технологии предлагается отказаться от использования «черных ящиков» и перейти к «белым ящикам». При этом, если автомат задан в любой форме, отличной от графа переходов без флагов и умолчаний, он может быть назван «относительно белым ящиком», а если он задан графом переходов без флагов и умолчаний, то — «абсолютно белым ящиком».

Распознавание «относительно белых ящиков» может проводиться с помощью математических преобразований [8], некоторые из которых изложены в разд. 17.4.

19. В качестве основной алгоритмической модели в рамках SWITCH-технологии предлагается применять систему взаимосвязанных графов переходов [30], что поддерживает возможность композиции и декомпозиции алгоритмов и обеспечивает практическое применение технологии при построении сложных систем логического управления.

При этом автоматы (графы переходов) могут образовывать централизованные, децентрализованные и иерархические структуры (рис. 5.41, 5.51, 5.55 в [8]).

20. Если в систему входят N графов переходов с любым числом состояний каждый, то при программной реализации даже с учетом взаимодействия графов могут использоваться только N внутренних многозначных переменных, введенных при кодировании состояний автоматов.

Для этого на программную реализацию накладывается ограничение, состоящее в том, что в каждом программном цикле в каждом графе переходов выполняется не более одного перехода: сохраняется предыдущее состояние автомата или выполняется только один переход в нем, даже в том случае, когда в рассматриваемом программном цикле в этом автомате последовательно могут быть выполнены несколько переходов.

Это обеспечивает доступность каждого значения каждой внутренней переменной для «всего окружения» — остальных $N - 1$ графов системы, что позволяет не увеличивать число внутренних переменных для осуществления взаимосвязи графов переходов и делает их взаимодействие с помощью значений многозначных внутренних переменных весьма наглядным. При этом предикаты, проверяющие значения этих переменных, используются в качестве еще одной из разновидностей входных воздействий.

Таким образом, при автоматном программировании, так же как и при объектно-ориентированном программировании, автоматы обмениваются сообщениями. Однако возможность взаимодействия автоматов, из которых состоит программа, за счет обмена номерами их внутренних состояний существенно отличает автоматное программирование от объектно-ориентированного, в котором объекты рассматриваются как «черные ящики», внутреннее содержание каждого из которых скрыто [31].

21. Графы переходов в системе могут взаимодействовать по принципам «запрос—ответ—сброс» или «запрос—ответ», обмениваясь при этом десятичными номерами состояний.

Частным случаем такого взаимодействия является взаимосвязь головного и остальных графов переходов. При этом в случае необходимости имеется возможность запускать из головного графа параллельные процессы, реализуемые указанными графами переходов, и возвращать управление головному графу после завершения их работы. Это расширяет область применения конечных автоматов, каждый из которых, как отмечалось выше, по определению является последовательностным по состояниям. В [8] показано, что при реализации алгоритма в виде системы взаимосвязанных графов переходов они имеют преимущества по сравнению с языком «Графсет», разработанным для описания последовательно-параллельных процессов фирмой «Телемеханик» (Франция), входящей в настоящее время в группу «Шнайдер Электрик».

22. Кроме обмена номерами состояний между автоматами, реализуемыми последовательным расположением в программе конструкций `switch`, графы переходов могут взаимодействовать и по принципу вложенности, что может быть реализовано вложенными конструкциями `switch` произвольной глубины или обращениями к функциям, построенным из этих конструкций, которые реализуют графы переходов указанной глубины вложенности.

23. Указанные принципы взаимодействия графов переходов поддерживают возможность иерархического построения алгоритмов.

Предлагаемая технология обеспечивает также проектирование алгоритмов как «сверху вниз», так и в обратном порядке. При этом первая стратегия упрощает построение корректных алгоритмов. Так, например, если алгоритм не обладает параллелизмом по состояниям, то в первоначально построенном графе переходов каждая вершина может быть заменена фрагментом, содержащим несколько вершин; в нее могут быть вложены другие графы переходов; из нее могут вызываться другие графы переходов. В полученной структуре каждая вершина, в свою очередь, может быть детализирована. Структура программы, реализующей построенный алгоритм, должна быть изоморфна его структуре.

24. При задании алгоритма в виде системы графов переходов, последняя, если это возможно, разбивается на не связанные между собой подсистемы, образованные взаимосвязанными графами переходов, и для каждой из них, если позволяет размерность, строится граф достижимых маркировок (состояний), описывающий все функциональные возможности подсистемы.

Таким образом, в рамках предлагаемой технологии, если позволяет размерность выделяемых подсистем графов переходов, тестирование систем графов переходов и реализующих их программ может быть заменено определением всех функциональных возможностей рассматриваемого описания, как это делается при верификации протоколов [32].

При использовании сильносвязанных систем графов переходов большой размерности проектирование на основе предлагаемой технологии обеспечивает возможность построения «контролепригодных» программ, для которых может быть выполнено автоматическое протоколирование их работы в терминах автоматов [169, 183].

25. Вводится понятие управляющего автомата, под которым понимается совокупность автомата и функциональных элементов задержки (ФЭЗ). При этом для автомата эти элементы рассматриваются наравне с объектом управления (его исполнительными механизмами и сигнализаторами): автомат кроме значений «объектных» выходных переменных формирует также значения «временных» выходных переменных, а кроме значений «объектных» входных переменных получает также значения «временных» входных переменных. «Временные» переменные отображаются на графах переходах так же, как и «объектные» переменные. В [8] рассмотрены различные подходы к программной реализации ФЭЗ, например с использованием функций (приложение 4 в [8]).

В общем случае кроме таких функций в компонентах графов переходов могут применяться и функции других типов, например реализующие регуляторы (приложение 10 в [8]).

26. Графы переходов могут использоваться также и при создании моделей объектов логического управления, что позволяет с единых позиций проводить описание и выполнять моделирование как разомкнутого, так и замкнутого комплекса «Управляющий алгоритм — модель объекта управления».

27. При применении предлагаемой технологии спецификация на разработку управляющей программы должна состоять из замкнутой (в общем случае) схемы связей «источники информации — система взаимосвязанных автоматов — функциональные элементы задержки — исполнительные механизмы объекта управления — средства представления информации», описывающей интерфейсы автоматов, и системы взаимосвязанных графов переходов, описывающих поведение автоматов, входящих в эту схему.

При этом схема связей (некоторый аналог диаграммы потоков данных в структурном системном анализе [33]) и графы переходов должны быть изображены так, чтобы их (по возможности) можно было охватить одним взглядом, что упрощает понимание спецификации (требование «гештальтности» описания [34]). Для этого, в частности, в графах переходов должны использоваться только символы переменных, а не их аббревиатуры, а смысл каждой переменной должен быть ясен при «переводе взгляда» на схему связей, которая может содержать в том числе и комментарии.

Для задач большой размерности указанная схема может строиться (с учетом всего «окружения») для каждого автомата в отдельности.

28. В рамках SWITCH-технологии предлагается иметь один язык спецификации алгоритмов (графы переходов) при применении различных языков программирования, что не было отмечено в [1].

Автором разработаны методы формальной и изоморфной реализации графов переходов программами, написанными на различных языках программирования, которые используются в управляющих вычислительных устройствах, в том числе и в ПЛК [8]. Для ПЛК наиболее целесообразно применять аналоги конструкции *switch*, позволяющие строить компактные и обозримые текстовые программы. Эту же конструкцию моделирует цифровой мультиплексор, обеспечивающий возможность построения для программирования ПЛК функциональных схем, формально и изоморфно реализующих графы переходов (разд. 4.2.3 в [8] и разд. 17.6.2).

Графы переходов, построенные, как изложено выше, и используемые в качестве спецификации для разработки программы (определяя ее функциональные возможности), а в дальнейшем и в качестве теста для ее сертификации, могут быть реализованы также и с помощью языка «Графсет». При этом появляется возможность при сертификации программы визуально наблюдать переходы в графе переходов. Однако этот язык программирования имеет ряд недостатков. Перечислим некоторые из них: возможность одновременного

отражения активности нескольких вершин в одной диаграмме приводит при реализации к двоичному их кодированию, при котором число двоичных внутренних переменных (используемых для кодирования вершин и имеющихся в ПЛК в ограниченном количестве) равно числу вершин в диаграмме; язык «Графсет» позволяет непосредственно реализовать только графы переходов автоматов Мура, требуя увеличения числа вершин для графов переходов, соответствующих другим типам автоматов; каждый граф переходов реализуется на нескольких экранах; графы переходов с переходами «назад» реализуются ненаглядно, так как для таких переходов линии, связывающие вершины, не изображаются или не могут быть изображены, и поэтому заменяются «соединителями».

При этом интересно отметить, что если в ПЛК, например при реализации одной совокупности графов переходов, израсходованы все двоичные внутренние переменные, выделенные для программ на языке «Графсет», то, изменив язык программирования при реализации другой совокупности графов переходов, может появиться возможность использовать другие типы внутренних переменных контроллера. Таким образом, даже при программировании одного ПЛК может возникнуть необходимость в программировании на разных языках при применении одного и того же языка спецификаций.

29. Предлагаемый подход позволяет Заказчику, Проектанту (Технологу), Разработчику, Программисту, Пользователю и Контролеру однозначно понимать, что должно быть сделано, что делается и что сделано в программно реализуемом проекте. Он позволяет разделять работу, а самое главное, ответственность между Специалистами различных областей знаний, а также между организациями, что особенно важно при работе с инофирмами из-за наличия языкового барьера и неоднозначности понимания естественных языков.

Подход обеспечивает возможность уже на ранних стадиях проектирования учесть все детали технического задания и продемонстрировать Заказчику в удобной для него форме, как оно понято.

30. Предлагаемая технология позволяет Участникам разработки проекта общаться не традиционным путем, в терминах технологического процесса (например, не «идет» режим экстренного пуска дизель-генератора), а на полностью формализованном, но понятном Специалистам различных областей знаний языке (на своего рода техническом эсперанто), на котором объясняться можно, например, следующим образом: «В третьем графе переходов, в пятой вершине, на четвертой позиции — изменить значение 0 на 1», что не вызывает разночтений, возникающих из-за неоднозначности понимания даже для одного естественного языка, а тем более для нескольких таких языков в случае, когда Участники разработки представляют разные страны, и не требует привлечения Специалистов, знающих

технологический процесс, для корректного внесения изменений в программы [35].

31. Излагаемый подход позволяет снять с Программиста необходимость знания технологического процесса, а с Разработчика — тонкостей программирования.

Он дает возможность Программисту прикладных задач ничего не додумывать за Заказчика, Технолога и Разработчика, а только однозначно реализовать формализованную спецификацию, что позволяет резко снизить требования к его квалификации, а в конечном счете и вовсе отказаться от его услуг и перейти к автопрограммированию Разработчиком.

Этот подход обеспечивает возможность оставлять понятные «следы» после завершения разработки, что упрощает сопровождение и модификацию программ даже новыми Специалистами.

Он позволяет контролировать разработку и тексты программ, а не только результаты их выполнения, как это имеет место в большинстве случаев в настоящее время, приближая приемку программ к приемке аппаратуры, которая проверяется не только на функционирование [8].

32. При реализации графов переходов значения «временных» выходных переменных заменяются, например, вызовами функций, реализующих ФЭЗ, а значения «объектных» выходных переменных могут заменяться вызовами других типов функций.

Поэтому рассматриваемый подход может использоваться также и при реализации управляющей части логико-вычислительных алгоритмов, как это, например, показано в Приложении 10 в [8], в котором приведена программа, обеспечивающая синхронизацию генератора с шинами главного распределительного щита.

33. Предлагаемая технология применяется в НПО «Аврора» при создании систем управления техническими средствами для судов различных проектов и других технологических объектов, в которых используются вычислительные средства разных фирм с различными языками программирования. При применении одного языка спецификаций в [35] программирование выполнялось на языке ПЛ/М, в [36] — на языке функциональных блоков, в [37] — на языке ассемблера однокристалльной микроЭВМ КР 1816 ВЕ51, а при создании комплексной системы управления техническими средствами для судна проекта 17310 — на языке инструкций ALPro [38,39].

Для упрощения программирования в последнем случае Б. П. Кузнецовым (НПО «Аврора») при участии автора был создан транслятор «Ядро языка СИ — язык ALPro» [40], который применяется к программам на языке СИ, построенным формально и изоморфно по графам переходов.

34. Опыт проектирования, испытаний и эксплуатации ряда систем логического управления подтвердил эффективность использования SWITCH-технологии для систем рассматриваемого

класса. При этом фирма «Norcontrol» (Норвегия) в [35] отметила, что применение излагаемого подхода позволило ей повысить качество системы логического управления дизель-генератором ДГР-2А 500*500 для судна проекта 15640.

Примеры использования этой технологии для других классов задач управления приведены в Приложениях 10 и 11 в [8].

35. Из изложенного выше следует, что наибольший эффект от применения предлагаемой технологии достигается в случае, когда алгоритмизация и программирование выполняются в разных организациях или странах [35].

36. Предлагаемая технология может характеризоваться совокупностью, состоящей из семи терминов: «состояние» — «автомат» — «независимость от глубокой предыстории» — «многозначное кодирование состояний» — «система взаимосвязанных графов переходов» — «формальное и изоморфное программирование» — «конструкция switch».

Ее применение обеспечивает наглядность, понятность, структурность, вызываемость, вложенность, иерархичность, управляемость и наблюдаемость программ, формально и изоморфно построенных по графам переходов.

Относительно небольшое число внутренних переменных, получаемое за счет применения многозначного кодирования состояний, резко упрощает и ускоряет их пересылку между вычислителями, которая может возникнуть, например, при построении отказоустойчивых систем, в которых при традиционном использовании, например, языка функциональных блоков, приходится пересылать содержимое сотен, а то и тысяч триггеров.

37. Для задач, отличных от логического управления, следует различать состояния управляющего автомата, описываемого графом переходов, от состояний остальной памяти. Если число состояний первого типа обычно не превышает нескольких десятков, то число состояний второго типа существенно превышает эту величину [18, 42], и поэтому они в явном виде не выделяются. Если указанное разделение состояний не выполнять, как это сделано, например, в [31], то состояния в дальнейшем и не используются, а поведение программной компоненты определяется как набор действий, выполняемых в ответ на события, без упоминания состояний первого типа, с которыми эти действия связаны. При этом следует отличать управляющие переменные от других внутренних переменных, соответствующих остальным атрибутам.

38. В рамках предлагаемой технологии управляющие автоматы могут строиться для:

- отдельных режимов (например, автомат открытия и автомат закрытия нескольких клапанов);
- объединенных режимов (например, автомат открытия и закрытия нескольких клапанов);

— индивидуальных объектов (например, клапанов), реализуя для каждого из них либо отдельные, либо объединенные режимы.

Построенные автоматы могут взаимодействовать между собой непосредственно путем обмена номерами состояний, вложения и вызова, а также через дополнительные автоматы. При этом отметим, что с увеличением числа автоматов в системе доказательство правильности их совместного функционирования усложняется.

Из изложенного следует, что предлагаемый подход может использоваться как при объектно-ориентированном проектировании (приложение 8 в [8]), так и при процедурном проектировании (приложение 5 в [8]).

39. Если исходный алгоритм реализуется граф-схемой с одним входом и одним выходом [41], то по ней с помощью методов, изложенных, например, в [42], может быть построен граф переходов соответствующего автомата (разд. 11.5 в [8] и [43]), который в общем случае реализуется конструкцией `switch`, являющейся телом в конструкции `do-while` с условием выхода из цикла по номеру исходной вершины. Изложенный метод является более эффективным по сравнению с методом структурирования программ Ашкрофта и Манн [8]. При этом отметим, что в [44] в качестве примера приведен граф переходов, реализующий один из алгоритмов сортировки чисел, который традиционно описывается циклической граф-схемой.

40. При непроцедурном задании автоматов в виде таблиц переходов и выходов их строкам и столбцам (или наоборот) соответствуют состояния и события, и поэтому при непроцедурной реализации автоматов интерпретатором порядок обработки (состояние—событие или событие—состояние) не имеет значения (приложение 3 в [8]). При процедурном задании автоматов в виде `с х е м а л г о р и т м о в ( п р о г р а м м )` [45] выполняется процедурная реализация, которая обычно более экономична по объему памяти и для которой порядок обработки состояний и событий существенен.

Будем называть эти схемы, начинающиеся с дешифратора событий (значений входных переменных), `с о б ы т и й н ы м и`, а схемы, начинающиеся с дешифратора состояний, — `а в т о м а т н ы м и`. При этом подходы к построению таких схем будем называть `событийными` и `автоматными` соответственно.

Автоматный подход позволяет отказаться от применения указанных схем, так как программирование в этом случае может выполняться непосредственно по наиболее «обозримой» форме задания автоматов — `г р а ф а м п е р е х о д о в` (с многозначным кодированием вершин), которые изоморфны таким схемам с дешифратором состояний (автоматные схемы алгоритмов) и конструкции `switch` языка СИ (разд. 17.5 и [17]).

В примерах 1 и 2 Приложения приведены программы, построенные с помощью событийного и автоматного подходов, которые реа-

лизируют *R*-триггер и счетный триггер соответственно. По мнению автора, автоматные схемы алгоритмов являются более «понятными».

В примере 3 Приложения приведены программы, реализующие некоторые функции управления элементом графического пользовательского интерфейса, построенные на основе событийного и автоматного подходов [46]. В первом случае, так же как и в объектном программировании, в котором «делается упор на создание автономных агентов, взаимодействующих между собой для достижения желаемого результата» [31], для каждого события реализован свой обработчик. При этом так же, как и в объектном программировании для обеспечения взаимодействия методов класса, эвристически вводятся флаговые (управляющие) переменные (в данном случае одна), оставляя вопрос о понятности и корректности последовательности логики такой программы открытым. Во втором случае обработчики событий вызывают функцию, формально и изоморфно реализующую граф переходов, передавая ей в качестве параметра номер произошедшего события. Благодаря тому что «логика», ранее рассредоточенная по обработчикам, теперь сконцентрирована внутри одной функции, резко упрощается понимание поведения программы [46, 47].

41. Предложенная технология является сквозной и прозрачной, так как она охватывает этапы алгоритмизации и программирования, для которых разработаны методы, обеспечивающие высокое качество проектирования не только в пределах одного этапа, но и между этапами.

42. Изложенный подход подробно описан в [8], а также в [17, 48—51]. Подходы, наиболее близкие к предлагаемому, содержатся в [30, 52—54].

Разработанный подход существенно дополняет международный стандарт ИЕС 1131-3 (ИЕС 61131-3) [1], который, описывая типовые языки программирования для ПЛК, не содержит (в отличие от [8]) методов проектирования программ на этих языках. Эта проблема не решена и в документации ведущих в области автоматизации фирм мира [166], в которой в лучшем случае содержатся лишь отдельные примеры (например, в [162]). Такая же ситуация имеет место и в литературе [55—63, 163].

Рассмотренный подход был предложен и внедрен автором в 1991 г. [36, 64].

В 1993 г. фирма «Модикон» (США), входящая в настоящее время в группу «Шнайдер Электрик» [4], а в 1996 г. и фирма «Сименс» (Германия) предложили использовать графы переходов в качестве одного из языков программирования для своих ПЛК, причем последняя в работе [2] утверждает, что описание на таком языке не только подходит для Программиста ПЛК, но также понятно Инженеру-механику, Инженеру по запуску оборудования и Инженеру по

обслуживанию. При этом отметим, что этот язык был введен указанными фирмами в практику проектирования, несмотря на то что он не входит в состав языков, рекомендуемых международным стандартом [1].

Подходы, предложенные этими фирмами, обладают по сравнению с технологией, излагаемой автором, тем преимуществом, что они разработали трансляторы непосредственно с этого языка, получив тем самым исполняемый язык спецификаций [65], а недостатки этих подходов состоят в следующем:

- эти трансляторы, естественно, не взаимозаменяемы;
- трансляторы ограничивают возможность применения моделей и подходов, отличных от принятых при их разработке;
- в документации не указаны особенности построения графов переходов, перечисленные выше, которые обеспечивают их понятность и возможность построения качественных программ на их основе;
- отсутствует указание на необходимость создания описанной выше схемы связей;
- не удастся охватить даже один граф переходов «одним взглядом», так как собственно граф, условия переходов и выходные переменные изображаются на разных экранах («негештальтное» описание);
- они используют графы переходов в качестве одного из языков программирования, не указывая при этом, что графы переходов целесообразно применять и в качестве языка алгоритмизации при использовании других языков программирования.

В настоящее время подход, основанный на применении графов переходов, названный «State Logic», начал использоваться также и фирмой «GE Fanuc Automation» (GE — General Electric) для программирования старших моделей своих ПЛК [66]. При этом был разработан язык программирования ECLiPS (English Control Language Program Software), позволяющий описывать графы переходов. При этом в материалах фирмы отмечено, что предложенный подход может применяться специалистами, которые не имеют опыта программирования. Этот подход рассматривается фирмой как практическая альтернатива лестничным схемам.

По мнению автора, недостатки указанного подхода состоят в следующем: графы переходов представляются как картинки, помеченные английскими словами; применяется специализированный язык программирования, использующий английский язык, а не математическую нотацию, позволяющую исключить умолчания; он требует применения специализированного процессора (State Logic Processor); не предложено использовать графы переходов в качестве языка спецификаций при программировании на языке лестничных схем младших моделей ПЛК.

Диаграммы состояний применяются в качестве одного из языков программирования промышленных контроллеров фирмы «Matsushita

Automation Controls» [164], а язык «список состояний» — для программирования контроллеров фирмы «Festo Cybernetic» [165].

Отметим также, что до последнего времени графы переходов применялись лишь в отдельных задачах теоретического программирования [67], а в практическом программировании они в основном использовались лишь для разработки компиляторов [68, 69], в то время как для аппаратных реализаций графы переходов давно и широко применяются [16]. В [70] было предложено использовать графы переходов в программировании, ввиду того что «мы можем рассматривать любую программу так, будто бы она полностью реализована аппаратными средствами».

Однако при создании программ для решения прикладных задач до последнего десятилетия главенствующим был процедурный подход (функциональная декомпозиция [184]), использующий не графы переходов, а такой визуальный формализм, как схемы алгоритмов.

Ограниченность (по мнению Г. Буча [71]) этого подхода при создании сложных программ привела к разработке и широкому распространению объектно-ориентированного подхода. Однако первое время новый подход поддерживали лишь языки программирования [72], а метод проектирования программ в рамках этой парадигмы отсутствовал.

Такой метод был разработан Г. Бучем в 1991 г. [71]. Он включает несколько визуальных формализмов (диаграмм) для отображения различных особенностей выделяемых классов и объектов. При этом в [71] было предложено (на одной странице) для описания поведения объектов использовать диаграммы переходов и был приведен пример применения этих диаграмм. Однако в рамках метода Буча в отличие от подхода, предлагаемого автором, эти диаграммы являлись лишь «картинками» [12], помеченными словами, а не были математическими моделями. Кроме того, Г. Бучем (в отличие от автора) не было отмечено, что при определенном подходе к построению схем алгоритмов они могут быть изоморфны графам переходов [8, 17 и разд. 17.5], и поэтому их противопоставление в этом случае некорректно. Этот факт не был отмечен также и в работах [43, 73].

Примерно та же картина, что и в книге Г. Буча, сохранялась в работе [74], авторы которой предложили моделировать «мир» в состояниях. При этом отметим, что этот «мир» в основном состоит из микроволновых печей, клапанов и насосов, относящихся к объектам, управляемым системами логического типа.

Существенно большее внимание диаграммам состояний и переходов, не изменив определения состояний через значения внутренних переменных, Г. Буч уделил в книге [11], разд. 5.3 которой написан на основе работ Д. Харела [75,76], предложившего, по мнению Г. Буча, «простой, но очень выразительный подход, который гораздо эффективнее традиционных автоматов с конечным числом состояний».

По мнению автора, указанное противопоставление некорректно, так как подход Д. Харела, как и подход, излагаемый в настоящей главе, базируется на таких автоматах.

Д. Харел предложил свой подход для несколько более широкого класса систем управления по сравнению с системами логического управления, которые называются «реактивными» или «реагирующими» («reactive») [77], «поведение которых лучше всего характеризуется их реакцией на события, произошедшие вне их контекста» [99]. Такие системы реагируют на поток событий изменением состояний и выполнением действий при переходах из состояния в состояние или действий и деятельности в состояниях [78—86]. При этом Д. Харелом был предложен визуальный формализм (модификация диаграммы состояний и переходов), названный им «картой состояний» («Statechart» в [75], или «State chart» в [78]), которая «математически эквивалентна диаграммам автоматов Мура—Мили» [77], но позволяет в ряде случаев описывать поведение этих автоматов более компактно. Карты состояний могут быть также названы как «диаграммы Харела». Отметим основные особенности этих диаграмм.

Наряду с состояниями могут использоваться «гиперсостояния (суперсостояния [12]), объединяющие несколько состояний, имеющих идентичную реакцию на одно и то же событие» [87]. При этом вместо изображения таких переходов в некоторое состояние из всех состояний, охватываемых гиперсостоянием, изображается только один переход из гиперсостояния в указанное состояние (о б о б щ е н и е п е р е х о д о в [87]).

Гиперсостояния теоретически могут иметь произвольную глубину вложения. Переходы из гиперсостояния связаны со всеми уровнями вложенности в него.

Гиперсостояния могут объединять ИЛИ-состояния (последовательные состояния) и И-состояния (параллельные состояния) [99]. В первом случае, перейдя в гиперсостояние, автомат может находиться только в одном из состояний (или в одном состоянии или в другом), а во втором случае в гиперсостоянии «возникает необходимость в параллельных состояниях» [99], изображаемых в «ортогональных регионах».

В вершинах диаграмм, соответствующих состояниям или гиперсостояниям, могут указываться: деятельности (помечаются словом do); действия, выполняемые однократно при входе в вершину (помечаются словом entry); действия, выполняемые однократно при выходе из вершины (помечаются словом exit).

Два последних случая могут быть названы о б о б щ е н и е м д е й с т в и й, так как при этом действия, однократно выполняемые при входе в вершину, заменяют одинаковые действия, которые в автоматах Мили или смешанных автоматах указываются на всех входящих в нее дугах, соответствующих переходам, а действия, одно-

кратно выполняемые при выходе из вершины, заменяют одинаковые действия, которые в автоматах Мили или смешанных автоматах указываются на всех выходящих из нее дугах, соответствующих переходам.

На каждой дуге диаграммы может быть указано событие, вызывающее переход, и логическое условие, «охраняющее» этот переход. На дуге также могут быть указаны действия, однократно выполняемые на переходе, и формируемые на переходе события.

В диаграммах могут применяться также псевдосостояния, например условное (С), терминальное (Т) и историческое (Н), которые не являются реальными состояниями.

Если гиперсостояние содержит историческое псевдосостояние, то при переходе в это гиперсостояние «управление передается тому состоянию, в котором система находилась в данном гиперсостоянии в последний раз» [87]. Использование обобщения переходов и исторического псевдосостояния позволяет, например, эффективно специфицировать прерывание, связанное с переходом системы из любого состояния, принадлежащего гиперсостоянию, в некоторое другое состояние, в котором выполняется обработка прерывания, и продолжение работы системы (после обработки прерывания) из того состояния, в котором наступило прерывание [87].

Некоторые из предложений Д. Харела, например обобщение переходов, могут применяться и в рамках SWITCH-технологии.

Несмотря на некоторую похожесть подхода Д. Харела и предлагаемого, они имеют принципиальные отличия, состоящие в том, что, во-первых, у Д. Харела могут использоваться как состояния, так и гиперсостояния, в то время как в SWITCH-технологии применяются только состояния, а во-вторых, при наличии гиперсостояний, по крайней мере с ИЛИ-состояниями, считается, что такая диаграмма Харела описывает поведение одного автомата, в то время как в предлагаемой технологии в этом случае применяется система взаимосвязанных графов переходов, описывающих функционирование системы взаимосвязанных автоматов.

При этом, во-первых, не ясно, как с помощью нотации Харела изобразить диаграмму при большом числе гиперсостояний с большим уровнем вложенности, а во-вторых, видимо, по этой причине в объектном моделировании считается, что диаграмма Харела (диаграмма состояний) описывает жизненный цикл отдельного объекта, а для моделирования поведения сообщества совместно работающих объектов предлагается использовать диаграммы другого типа (диаграммы взаимодействий) [99], что резко усложняет формальный переход к реализации.

В предлагаемой технологии вместо понятия «объект» применяются понятие «автомат» и динамические диаграммы одного типа (система взаимосвязанных графов переходов) даже при наличии большого числа автоматов с большим уровнем вложенности, кото-

рые весьма просто формально и изоморфно реализуются практически на любом языке программирования.

Кроме того, при объектном подходе остается неясной формальная связь между статическими и динамическими диаграммами [77, 99], в то время как при замене диаграммы объектов схемой взаимодействия автоматов, использовании схемы связей, включающей систему взаимосвязанных автоматов, и системы взаимосвязанных графов переходов вся концепция в целом оказывается выраженной в терминах автоматов.

Видимо, по этим причинам, а также из-за сложности и специфичности нотации диаграммы Харела в отличие от графов переходов не применяются до настоящего времени при программировании ПЛК.

Предлагаемая технология может рассматриваться в качестве методологии проектирования программного обеспечения по крайней мере для класса SoftLogic [62]. Этот подход позволяет «не говорить о том, что программное обеспечение работает, а объяснить, почему оно работает» [20]. Подход, в частности, позволяет ответить на три вопроса: откуда появляются внутренние (управляющие) переменные, сколько их должно быть и для какой цели каждая из них используется?

Автоматный подход применяется для спецификации протоколов сетей ЭВМ [32, 88, 89—91], а также в SDL-методологии [81—83], разработанной Международной комиссией по телефонии и телеграфии для создания программного обеспечения телекоммуникационных систем. Однако SDL-диаграммы, являющиеся некоторой разновидностью схем алгоритмов, в которые в явном виде могут вводиться состояния, весьма громоздки и соответствуют только одному классу автоматов — автоматам Мили. В SDL-методологии не оговаривается ряд вопросов, отмеченных выше, которые связаны, например, с умолчаниями и флагами. Ряд других недостатков SDL-диаграмм (в частности, их вертикальная направленность, существенно уменьшающая обзорность спецификации и не позволяющая эффективно использовать плоскость листа бумаги или дисплея) отмечен в [95].

Это привело к тому, что при разработке [95—98] технологии программирования встроенных систем реального времени, реализующих широкий класс алгоритмов, ее авторы, отметив недостатки модели Харела, например «отсутствие ориентации на генерацию конечного кода», при создании модели поведения объектов объединили достоинства SDL-диаграмм и диаграмм Харела, сохранив, правда, альтернативные нотации каждого из них. Та же ситуация сохранилась и при создании объектно-ориентированной методологии разработки программного обеспечения систем реального времени [153]: в поведенческой модели используются диаграммы состояний в модифицированной нотации Харела, а для детализации — SDL-диаграммы [154]. Это делает модели, предложенные в

[95, 154], весьма специфичными и ограничивает их применение для задач логического управления технологическими процессами, при решении которых SDL-диаграммы (в отличие от телефонии) ведущими в этой области фирмами мира не используются.

Еще более разнообразной является модель поведения, предложенная при разработке унифицированного языка моделирования (Unified Modeling Language (UML)) [12, 78, 99, 100], являющегося, по мнению его авторов, коллекцией наилучших инженерных нотаций, применяемых при объектном моделировании сложных систем общего назначения. Построение этого языка базируется на подходах, предложенных в [11, 71, 101—103].

В этом языке для описания поведения объектов используется «конечный автомат» («State Machine»), который в зависимости от приоритета состояний или деятельности в этом описании задается диаграммой состояний (State Diagram) или диаграммой деятельности (Activity Diagram) соответственно, что, по мнению автора, некорректно, так как в последних возможен параллелизм по переходам в одной диаграмме. При этом диаграмма состояний не является одной из классических, а считается эквивалентной диаграмме Харела [77], но почему-то несколько отличается от нее [78].

Наличие в языке диаграмм деятельности, «являющихся одной из самых больших неожиданностей UML» [12], которые, как отмечено выше, в рамках теории автоматов не могут рассматриваться как конечные автоматы, создает неопределенность при выборе нотации для описания поведения объектов. При этом отметим, что эти диаграммы сильно напоминают диаграммы «Графсет», которые уже более двух десятилетий используются при программировании систем автоматизации, но отличаются (в худшую сторону) от последних отсутствием «полочек» для указания в явном виде условий, при которых выполняются переходы, и наличием условных вершин, характерных для схем алгоритмов. Отметим также, что если в диаграммах «Графсет» вершины названы этапами, то в диаграммах деятельности аналогичные вершины некорректно названы состояниями.

Кроме того, в [15], в отличие от [12,78], показано, что каждая диаграмма деятельности (так же как и каждая диаграмма «Графсет» (п. 21)) может быть заменена системой взаимосвязанных графов переходов, которые, как отмечено в п. 6 для диаграмм «Графсет», могут быть реализованы с меньшим числом внутренних переменных.

Неопределенность с выбором нотации для описания поведения объектов в UML еще более увеличивается, в связи с тем что к диаграммам деятельности отнесены также и конструкции, аналогичные SDL-диаграммам, что свидетельствует о недостаточном научном обосновании формирования рассмотренной части «коллекции».

Для большего единообразия фирма «I-Logix», принимавшая участие в разработке указанного языка, при создании методологии проектирования объектно-ориентированных встроенных систем реального времени выбрала из этой части «коллекции» только диаграммы Харела в их первоначальной нотации [85], но, однако, не решила и даже не поставила вопрос о сертификации поведения совокупности таких диаграмм.

Отметим также, что диаграммы Харела используются в [148] при визуальной разработке программных компонент, описываемых конечными автоматами и предназначенных для различных целевых платформ. Только эти диаграммы применяются и для описания поведения объектов в [180].

Расширение нотации диаграмм Харела выполнено в [104]. Если в диаграммах Харела на переходах допустимо применение лишь отдельных условных вершин, характерных для схем алгоритмов (Flow Diagram), в которых понятие «состояние» не используется (stateless), то в [104] эти вершины могут образовывать более сложные конфигурации. Получающаяся при этом тесная связь между «state diagram» и «flow diagram» привела к появлению нового термина «stateflow». Возможность применения этих диаграмм в качестве исполняемых спецификаций на широко используемом (по крайней мере, в университетских кругах) языке «MATLAB» [105] позволяет также реализовать эти диаграммы на языках C и C++.

Однако, так как и эта нотация является весьма специфической, то для систем логического управления, принадлежащих к нижнему уровню управляющих систем (особенно при реализации их на ПЛК), автор считает более целесообразным при алгоритмизации применять системы взаимосвязанных графов переходов, содержащих минимальную номенклатуру компонент и обозначений («не надо размножать сущности без необходимости» (Оккам)) и ориентированных на классические модели теории автоматов, по которым, как отмечено выше, программирование может выполняться формально и изоморфно даже вручную.

Для этого класса систем предлагаемая технология позволяет строить математические модели, по которым, в частности, при любом типе двоичного кодирования состояний для программной реализации могут быть построены системы булевых формул, в то время как при других подходах такая задача даже не ставится.

Предлагаемая технология поддерживает проектирование и реализацию программного обеспечения для рассматриваемого класса систем и определяет состав и содержание выпускаемой документации, которая должна контролироваться Заказчиком.

Изложенное весьма актуально, так как обычно эта документация в лучшем случае содержит только тексты программ [106], их описа-

ния и руководство пользователя, а в худшем — тексты программ в документацию не включаются [107].

Кроме перечисленных выше работ конечные автоматы, рассматриваемые в [181] как разновидность процессов, в настоящее время применяются при управлении автоматическими выключателями корабельных электроэнергетических систем [108], при создании не процедурных языков программирования для автоматизированных систем управления технологическими процессами [109], при описании поведения «объектов» [15, 110] программно реализуемых сложных систем [87, 111—130], в том числе «многониточных» контроллеров [26], а также при решении других задач, связанных с параллельными процессами (например, задачи о пяти философях, собравшихся за круглым столом и имеющих «проблемы» с вилками [149], о функционировании мячиков внутри окна [155], о синхронизации цепи стрелков [156]).

Графы переходов являются также основной формой описания управляющих последовательностных процессов при создании программного обеспечения систем обработки информации на основе методов структурного системного анализа и проектирования [33]. Они используются и для проводимого по методологии IDEF (Integration Definition for Function Modeling) документирования трансформаций объектов, происходящих в ходе выполнения технологического процесса [161].

Роль и место состояний («состояния бытия»), конечных автоматов, графов переходов и конструкции `switch` при программировании игр описаны в [131].

Более того, графы переходов могут использоваться для описания поведения машин Тьюринга, применяемых при формальном определении понятия алгоритм [87, 132]. Поэтому заданная машина Тьюринга также может быть реализована на основе конструкции `switch`, в которой используются дополнительные действия, имитирующие сдвиг головки «чтение—запись» по рабочей ленте. Таким образом, предлагаемый подход в принципе может применяться и для более широкого по сравнению с рассматриваемым классом задач, как например это делается при использовании модели «расширенного конечного автомата», который рассматривается в виде композиции двух автоматов — конечного и контекстного, состояния первого из которых называются основными и кодируются одной переменной, а состояния второго отражают переменные, называемые контекстными [32, 88]. Этот же подход может применяться при описании «объектов» в объектно-ориентированном программировании, что позволяет в явном виде поддерживать одно из определений объекта как некоторого элемента, на который можно воздействовать, изменяя его состояния.

Из изложенного следует, что в настоящее время автоматный подход еще только начинает использоваться при программной реализа-

ции алгоритмов, и утверждение, высказанное в [133], о «смерти» теории автоматов является, по мнению автора, сильно преувеличенным. Более того, рецензент книги автора [8] определяет в [107] «конечные автоматы как инструмент борьбы с монополизмом фирм, разрабатывающих программное обеспечение», а в [43, 73] и вовсе приведен «гимн» применению автоматов в программировании.

В этой связи особо следует отметить, что в [150] создатель операционной системы Unix К. Томпсон на вопрос о текущей работе ответил: «Мы создали язык генерации машин с конечным числом состояний, так как реальный селекторный телефонный разговор — это группа взаимодействующих машин с конечным числом состояний. Этот язык применяется в „Bell Labs” по прямому назначению — для создания указанных машин, а кроме того, с его помощью стали разрабатывать д р а й в е р ы».

Подход, аналогичный с предложенным для программной реализации алгоритмов, широко используется в настоящее время для настройки с х е м программируемой логики [134], также называемой программированием [135]. При этом разработаны языки описания аппаратуры (Hardware Description Language (HDL) в англоязычной литературе), которые, в частности, позволяют конвертировать текст программы в функциональную схему и наоборот. Описание, выполненное на одном из таких языков, компилируется в схему [136]. Среди языков этого класса отметим языки VHDL [137] и AHDL [138].

Например, в языке AHDL для реализации условной логики (Conditional Logic) включены конструкции `if then` и `case`, последняя из которых является аналогом конструкции `switch` языка СИ. При этом одно из «золотых» правил, приведенных в [138], состоит в применении везде, где это возможно, конструкции `case` вместо вложенных конструкций `if then`.

Это правило используется для реализации последовательностной логики (Sequential Logic), описываемой в терминах состояний с помощью конечных автоматов (State Machine). При этом автоматы с многозначным кодированием состояний (этот термин в [138] не применяется) задаются в виде диаграмм состояний (State Diagram). В [138] приводятся примеры реализации автоматов с синхронными выходами (State Machines with Synchronous Outputs), соответствующих автоматам Мура, и автоматов с асинхронными выходами (State Machines with Asynchronous Outputs), соответствующих автоматам Мили.

Дальнейшее развитие автоматного подхода для настройки схем программируемой логики состоит в визуализации диаграмм состояний, которая осуществляется с помощью пакетов типа «State CAD», один из которых описан в [139].

Таким образом, предложенная автором технология создания программ дополняет известные подходы к настройке схем программируемой логики, что позволяет с единых позиций проводить аппа-

ратную [140] и программную реализации алгоритмов логического управления, так же как это имеет место в технологии «Co-Design», предназначенной для совместной разработки аппаратных и программных средств с помощью формализованных методов, в которой в качестве языка описания высшего уровня абстракции применяются конечные автоматы [141].

В заключение отметим, что после издания [8] автор познакомился с работами [142,143]. Некоторые из особенностей предложенного автором подхода, базирующегося на парадигме конечных автоматов, являющейся одной из парадигм программирования, рассмотренных Р. Флойдом [142], совпадают с предложениями, высказанными лауреатами премии Тьюринга и самим А. Тьюрингом в [182].

Так, А. Дж. Перлис в 1966 г. [142] предложил в описания языка, среды и правил вычислений включить состояния, которые могут подвергаться мониторингу во время исполнения, позволяя диагностировать программы, не нарушая их целостности. При этом под мониторингом он понимал распределенное управление, более известное в настоящее время под названием «объектно-ориентированное программирование».

В этом же году Э. Дейкстра [143] предложил ввести так называемые переменные состояния, с помощью которых можно описывать состояния системы в любой момент времени, и ввел для этих целей целочисленные переменные. При этом им были поставлены вопросы о том, какие состояния должны вводиться, как много значений должны иметь переменные состояния и что эти значения должны означать. Он предложил сначала определять набор подходящих состояний, а лишь затем строить программу. Он также предложил сопоставлять процессы с переменными состояниями и связывать процессы через эти переменные. По мнению Э. Дейкстры, диаграммы состояний могут оказаться мощным средством для проверки программ. Все это обеспечивает поддержку его идеи, состоящей в том, что программы должны быть с самого начала составлены правильно, а не отлаживаться до тех пор, пока они не станут правильными.

В 1977 г. Дж. Бэкус [142] отметил, что в языках программирования для компьютера фон Неймана семантика тесно сплетается с переходами между состояниями и их недостатком является то, что «всякая подробность вычислений изменяет состояние. Вследствие такой тесной связи семантики с состояниями всякая деталь каждого свойства должна быть встроена в состояние и в его правила переходов». При этом он указал на необходимость того, чтобы вычислительные системы обладали свойством «исторической чувствительности», и отметил, что «система не может быть исторически чувствительной (допускать влияние выполнения одной программы на поведение следующей программы), если она не обладает некоторым состоянием, которое первая программа может изменять, а вторая

воспринимать. Поэтому исторически чувствительная модель вычислительной системы должна обладать семантикой смены состояний». Он предложил рассматривать состояние в целом, а его изменение производить только после «большого» вычисления.

Отметим также, что использование в программировании формальных методов (языков формальных спецификаций в [65]) позволяет повысить качество программного продукта даже в тех случаях, когда их удастся применить лишь для части разрабатываемого проекта [65].

Автор надеется, что предлагаемый подход, использующий элементы бихевиоризма, когнитивной психологии [49] и гештальт-психологии [34], позволит повысить качество проектирования и реализации программ логического управления ответственными технологическими объектами, в том числе и за счет автопрограммирования, при котором человек, знающий объект, сам выполняет алгоритмизацию и программирование, так как для рассматриваемого класса задач предлагаемая система обозначений является «естественным продолжением образа мышления, а не чуждого ему формализма» (Н. Вирт в [142]). Графы переходов в качестве языка алгоритмизации, как и любой «язык, на котором записывается решение задачи, напрямую влияют на ход мыслей человека, заставляя его рассматривать задачу под определенным углом» [31], определяя «дисциплину мышления» [20]. В настоящее время считается [66], что графы переходов позволяют наиболее естественным образом описывать процессы, протекающие в реальном мире, дополняя объектный подход, являющийся «естественным способом размышления о мире» [144].

Если процедурное программирование ориентировано на выбор и действия, объектное программирование — на объекты [144], событийное программирование — на события, то предлагаемое автоматное программирование — на автоматы, включающие состояния и входные и выходные воздействия.

При этом отметим, что по «данным Киевского представительства фирмы „Шнайдер Электрик“ в настоящее время только 4 % украинских пользователей ее ПЛК используют язык „Графсет“, входящий в стандарт ИЕС 1131-3. Это, видимо, связано с инерцией мышления у программистов [185], которые в большинстве случаев автоматически переносят стиль программирования на языках для персональных компьютеров на программирование для ПЛК» [9].

Однако, по мнению автора, указанная ситуация связана не только с особенностями пользователей, но и с методическим обеспечением применения этого языка и его реализацией. Так, например, в [145] сформулирована одна из задач логического управления и для нее построена диаграмма «Графсет», которая, однако, является «картинкой», а не математической моделью. Это исключает возможность формального и изоморфного перехода от нее к программе и применение диаграммы в качестве сертификационного те-

ста. Удивительным является то, что эта диаграмма в [145] не используется для написания программы на языке «Графсет», а по ее «мотивам» эвристически строятся лестничная схема и программа на языке инструкций. Кроме того, как было отмечено в [8], графы переходов в системах логического управления обычно имеют многочисленные возвраты назад, что делает реализующие их диаграммы «Графсет» ненаглядными. Небольшое число двоичных переменных, выделяемых в ПЛК для кодирования вершин в этих диаграммах, резко ограничивает допустимое число вершин в реализуемых таким образом графах переходов.

По мнению автора, в рамках рассматриваемой технологии программирование ПЛК наиболее целесообразно выполнять на языке «структурированный текст» («Structured Text») [1], особенно для тех ПЛК, в которых этот язык содержит конструкцию, аналогичную конструкции `switch` языка СИ.

Предлагаемый подход является методическим обеспечением процесса алгоритмизации и программирования для систем логического управления при применении различных языков программирования. Этот подход в настоящее время, в частности, использован в НПО «Аврора» при создании:

- системы управления дизель-генератором ДГР-2А 500*500 для трех судов проекта 15640 на базе аппаратуры «Selma-2» фирмы «ABB Stromberg» (Финляндия). Программирование выполнялось на языке функциональных блоков [36];

- системы управления дизель-генератором того же типа для судна проекта 15967 на базе аппаратуры «Selma-2» фирмы «ABB Stromberg». Программирование выполнялось на языке функциональных блоков;

- системы управления дизель-генератором того же типа для судна проекта 15760 на базе аппаратуры фирмы «Norcontrol» (Норвегия). Программирование выполнялось фирмой на языке ПЛ/М [35];

- комплексной системы управления техническими средствами пяти судов проекта 17310 на ПЛК «Autolog» фирмы «FF-Automation OY» (Финляндия) [38]. Программирование для общесудовых систем выполнялось на языке инструкций ALPro [39], а для системы управления вспомогательными механизмами — с помощью транслятора «Ядро языка СИ — язык инструкций ALPro» [40];

- комплекта «Авролог» для управления техническими средствами судна на ПЛК «Autolog» фирмы «FF-Automation OY». Программирование выполнялось с помощью указанного выше транслятора;

- автоматизированной системы управления технологическими процессами (АСУ ТП) центральной подготовительной станции первичной обработки нефти «Авролог-НП1» (Северо-Ореховское месторождение). Программирование выполнялось на языке инструкций ALPro;

— АСУ ТП дожимной насосной станции «Авролог-НП2» (Северо-Покурское месторождение). Программирование выполнялось на языке инструкций ALPro;

— системы управления турбокомпрессорным агрегатом «Ларина», используемой при производстве полиэтилена в ПО «Полимир» (Новополоцк). Программирование выполнялось на языке ассемблера однокристалльной микроЭВМ КР 1816 BE51 [37].

Изложенная технология предназначена для создания алгоритмического и программного обеспечения систем логического управления, в которых ввод входных переменных выполняется только путем опроса [146] — используется «циклический исполнитель» [99]. Другая особенность этой технологии состоит в том, что при ее применении программы реализуются компилятивно: по графам переходов строятся конструкции switch или их аналоги, которые выполняются непосредственно и поэтому являются «активными».

А. А. Грабовский (АО «Каскод», Санкт-Петербург) применил предлагаемый автоматный подход для программной реализации логической части систем управления, проектируемых на микроконтроллерах фирмы «Сименс», для которых характерен развитый механизм обработки прерываний. При этом по таблично заданным графам переходов автоматически строится массив (напоминающий конструкции switch), который является «пассивным» и обрабатывается интерпретатором, учитывающим возможность появления запросов на обработку во время, когда микроконтроллер реализует соответствующий переход в одном из графов.

Предложенный подход использовал также и С. Б. Терентьев (НПО «Аврора») при реализации протоколов в распределенных управляющих системах, построенных на основе указанных микроконтроллеров. При этом написание программ выполнялось формально и изоморфно непосредственно по построенным графам переходов.

Н. И. Туккель (НПО «Аврора») и автор применили SWITCH-технологию при разработке системы управления дизель-генератором, реализуемой на промышленном компьютере с операционной системой QNX, в которой управляющая программа выполняется как один процесс, а программа, моделирующая объект управления, — как другой процесс. При этом был создан вариант SWITCH-технологии для «реактивных» систем [169, 183], ввиду того что управление на основе событий приводит к существенным отличиям при проектировании системы по сравнению со случаем применения в ней только «циклического исполнителя» [99]. Такие системы обычно реализуются на промышленных компьютерах, работающих под управлением операционных систем реального времени.

Предлагаемый вариант технологии характеризуется следующими особенностями:

— в качестве базового используется понятие «а в т о м а т», а не «класс», «объект», «алгоритм» или «агент», как это имеет место при других подходах. При этом соответствующая область программирования может быть названа «автоматно-ориентированное программирование»;

— в общем случае автоматы рассматриваются не изолированно, а как составные части взаимосвязанной системы — системы взаимосвязанных автоматов, поведение которой формализуется с помощью системы взаимосвязанных графов переходов;

— в качестве основной применяется модель смешанного автомата, для описания поведения которого используется соответствующий граф переходов только с «простыми» состояниями (гиперсостояния не используются);

— расширена (по сравнению с [8]) нотация, применяемая при построении графов переходов (например, в части перечисления вложенных автоматов);

— на этапе изучения предметной области на основе технического задания, которое при автоматизации технологических процессов обычно выдается Заказчикам в словесной форме в виде совокупности сценариев и случаев использования [99], строится структурная схема системы, позволяющая получить общее представление об организации управления, применяемой аппаратуре и интерфейсе объекта управления;

— на этапе анализа на основе технического задания выделяются сущности, каждая из которых называется автоматом (например, автомат управления насосом или автомат контроля температуры);

— состояния каждого автомата первоначально определяются по выделенным состояниям объекта управления или его части, а при большом их количестве — по алгоритму управления, построенному в другой нотации (например, в виде схемы алгоритма [44]);

— в автоматы также могут быть введены и другие состояния, связанные, например, с неправильными действиями Оператора;

— каждый автомат при необходимости может быть декомпозирован;

— итеративный процесс анализа может выполняться многократно и завершается созданием перечня автоматов и перечня состояний для каждого из них;

— на этапе проектирования в отличие от традиционного программирования вводится подэтап — кодирование состояний автомата. При этом в каждом автомате для различия состояний применяется многозначный код, в качестве комбинаций которого вводятся десятичные номера состояний;

— автоматы взаимодействуют за счет обмена номерами состояний, вложенности и вызываемости. Они также могут быть одновременно вложенными и вызываемыми;

— строится схема взаимодействия автоматов, отражающая указанные типы взаимодействий. Она формализует систему взаимодействующих автоматов. Эта схема заменяет в предлагаемой технологии диаграмму объектов и частично диаграмму взаимодействий (диаграмму кооперации), которые применяются в объектном моделировании [99];

— входные воздействия разделяются на события, действующие кратковременно, и входные переменные, вводимые путем опроса;

— входные воздействия целесообразно реализовывать в виде входных переменных, а применять события — для сокращения времени реакции системы. При этом одно и то же входное воздействие может быть одновременно представлено и событием и входной переменной;

— прерывания обрабатываются операционной системой и передаются программе в виде сообщений, а после этого обрабатываются как события с помощью соответствующих обработчиков;

— некоторые входные переменные могут формироваться в результате сравнения входных аналоговых сигналов с уставками;

— номера состояний других автоматов, с которыми автомат взаимодействует за счет обмена номерами состояний, также рассматриваются в качестве его входных воздействий;

— все выходные воздействия являются действиями, а не деятельностими;

— группы входных и выходных воздействий связываются с состояниями, выделенными для каждого автомата;

— связи каждого автомата с его «окружением» формализуются схемой связей автомата, предназначенной для полного описания интерфейса автомата. В этой схеме приводятся источники и приемники информации, полные названия всех воздействий и их обозначения, а также информация о том, в какой автомат он вложен и какие автоматы вложены в него;

— имя автомата начинается с символа *A*, имя события — с символа *e* (от английского слова *event* — событие), имя входной переменной — с символа *x*, имя переменной состояния автомата — с символа *y*, а имя выходного воздействия — с символа *z*. После каждого из указанных символов следует номер соответствующего автомата или воздействия;

— система взаимосвязанных автоматов образует системонезависимую (например, от операционной системы) часть программы, которая реализует алгоритм функционирования системы управления;

— реализация входных переменных, обработчиков событий, выходных воздействий, вспомогательных модулей и пользовательских интерфейсов образует системозависимую часть программы;

— обработчики событий содержат вызовы функций графов переходов (автоматов) с передачей им соответствующих событий. Функции входных и выходных воздействий вызываются из функций, реализующих автоматы. Функции, образующие вспомогательные модули, вызываются из функций входных и выходных воздействий. Таким образом, автоматы находятся в «центре» структуры программы, создаваемой на основе предлагаемого подхода;

— если составляющая системозависимой части программы спроектирована как автомат (например, автомат разбора файла рекомендаций Оператору), то он также может быть введен в схему взаимодействия автоматов;

— запуск автоматов может производиться как из системозависимой части программы (например, из обработчиков событий), так и из системонезависимой части;

— в л о ж е н н ы е а в т о м а т ы последовательно запускаются с передачей «текущего» события в соответствии с путем в схеме взаимодействия автоматов, определяемым их состояниями в момент запуска головного автомата. При этом последовательность запуска и завершения работы автоматов напоминает алгоритм поиска в глубину [160];

— в ы з ы в а е м ы е а в т о м а т ы запускаются из выходных воздействий с передачей соответствующих «внутренних» событий;

— автоматы могут запускаться однократно с передачей какого-либо события или многократно (в цикле) с передачей одного и того же события;

— при реализации системы учтено, что функции, реализующие автоматы, не реентерабельны (не допускают повторного запуска до их завершения);

— каждый автомат при запуске выполняет н е б о л е е о д н о - г о п е р е х о д а;

— после обработки очередного события автомат сохраняет свое состояние и «засыпает» до появления следующего события;

— д у г и и п е т л и графов переходов помечаются произвольными логическими формулами, которые могут содержать входные переменные и предикаты, проверяющие номера состояний других автоматов и номера событий;

— дуги и петли кроме условий переходов могут содержать список последовательно выполняемых выходных воздействий;

— в е р ш и н ы в графах переходов практически всегда являются устойчивыми и содержат петли. Если на петле не выполняются выходные воздействия, то она умалчивается. В противном слу-

чае, в явном виде изображаются одна или несколько петель, каждая из которых помечена по крайней мере выходными воздействиями;

- вершина графа переходов может содержать список последовательно запускаемых вложенных автоматов и список последовательно выполняемых выходных воздействий;

- для обобщения «одинаковых» исходящих дуг в каждом графе переходов допускается объединение вершин в группы. Также допускается слияние входящих в вершину дуг в одну линию;

- каждый граф переходов проверяется на достижимость, непротиворечивость, полноту и отсутствие генерирующих контуров;

- этап завершается построением графа переходов для каждого автомата, совокупность которых образует систему взаимосвязанных автоматов;

- на этапе реализации строится программа, в которой графы переходов, входные переменные, обработчики событий и выходные воздействия выполняются в виде функций. Кроме того, программа содержит вспомогательные модули, состоящие из наборов соответствующих функций (например, модуль управления таймерами);

- выделение функций декомпозирует задачу, а выделение состояний — ее последовательностную логику;

- для хранения номера состояния автомата (различения состояний) используется одна внутренняя переменная. Для различения изменения состояния применяется вторая переменная, носящая вспомогательный характер;

- разработан универсальный алгоритм программной реализации иерархии графов переходов с произвольным их количеством и произвольным уровнем вложенности;

- каждый граф переходов формально и изоморфно реализуется отдельной функцией (подпрограммой), создаваемой по шаблону, содержащему две конструкции `switch` и оператор `if`. Первая конструкция `switch` вызывает вложенные автоматы и реализует переходы и действия на дугах и петлях. Оператор `if` проверяет, изменилось ли состояние, и если оно изменилось, вторая конструкция `switch` активизирует вложенные автоматы и реализует действия в вершине;

- после реализации графа переходов текст подпрограммы должен корректироваться для обеспечения бесповторности опроса входных переменных, помечающих дуги, исходящие из одного состояния. Таким образом решается проблема «риска» [8];

- каждая входная переменная и каждое выходное воздействие также реализуется функцией, что позволяет применять SWITCH-технологию не только для решения задач логического управления;

— имена функций и переменных, используемых при реализации автоматов, совпадают с обозначениями, применяемыми в схемах связей автоматов и графах переходов. Например, переменная, в которой хранится номер произошедшего события, имеет имя «с»;

— все функции, реализующие входные переменные, записываются в порядке возрастания их номеров в один файл, а реализующие выходные воздействия — в другой;

— функции, реализующие автоматы, входные переменные и выходные воздействия, содержат вызовы функций для обеспечения протоколирования;

— этап завершается построением структурной схемы разработанного программного обеспечения, отражающей взаимодействие его частей. Эта схема может включать схему взаимодействия автоматов (рис. 21.1), которая при этом отдельно не выпускается;

— на этапе отладки обеспечена возможность одновременной индикации значений переменных состояний всех автоматов на одном экране;

— на этапе сертификации за счет введения в функции автоматов, входных и выходных воздействий вызовов функций протоколирования обеспечено автоматическое ведение протокола. В нем указываются события, запуск автоматов, их состояния в момент запуска, переходы в новые состояния, завершение работы автоматов, значения входных переменных, выходные воздействия и время начала выполнения каждого из них. Кроме «полного» протокола также автоматически строится «короткий» протокол, в котором фиксируются только события и инициируемые ими выходные воздействия, интересующие Заказчика;

— сообщения в «полном» протоколе о запуске и завершении реализации каждого автомата играют роль скобок, логически выделяющих разные уровни вложенности автоматов;

— на этапе документирования для точного оформления результатов проектирования и разработки программы предлагается создавать и сдавать в архив документацию (по крайней мере в электронном виде), имеющую как минимум следующую комплектность: структурная схема системы; схема разработанного программного обеспечения; распечатки экранов пользовательских интерфейсов; перечни событий, входных переменных и выходных воздействий; диаграмма взаимодействия автоматов; описание нотации, используемой в графах переходов; шаблон для реализации графов переходов смешанных автоматов произвольного уровня вложенности; для каждого автомата: словесное описание (фрагмент технического задания), рассматриваемое в качестве комментария, схема связей автомата, граф переходов и исходный текст функции, реализующей автомат; алгоритмы, например в виде

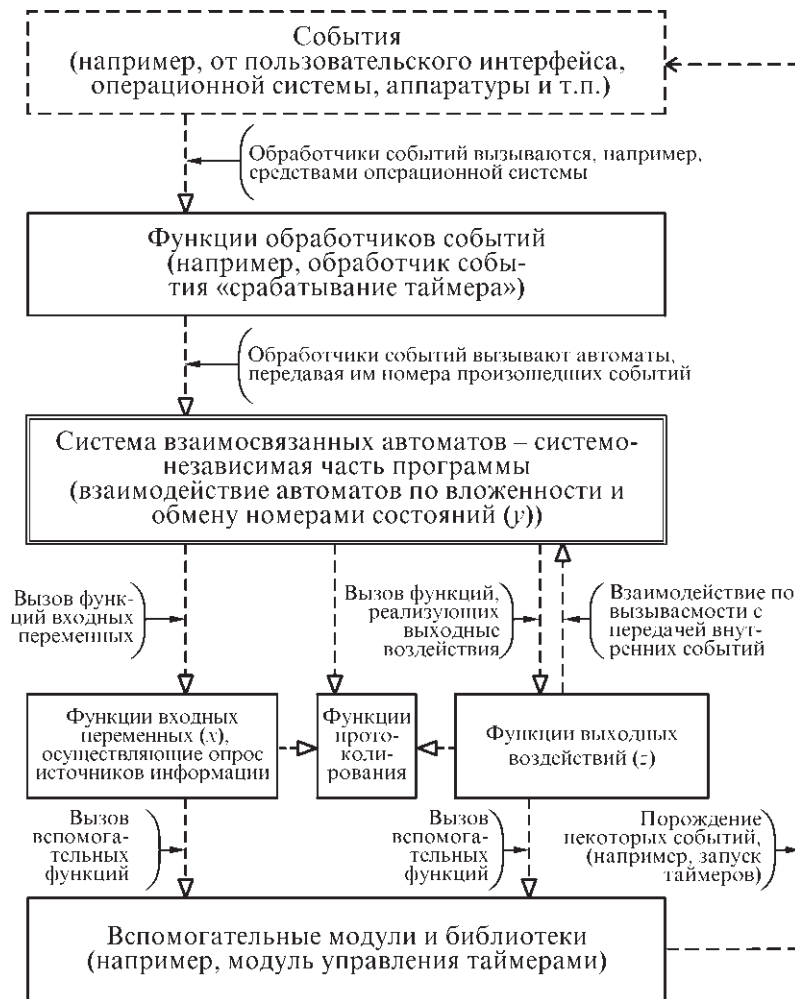


Рис. 21.1

графов переходов, и исходные тексты вспомогательных модулей и функций, реализующих входные переменные, обработчики событий и выходные воздействия; протоколы для сертификации программы, выполняющие роль контрольных примеров [151]; руководство программиста; руководство пользователя;

— изложенный вариант технологии может использоваться и при построении модели объекта управления;

— при наличии модели объекта для нее должен создаваться аналогичный комплект документации;

— после этого, при появлении любых изменений, возникающих в ходе дальнейших этапов жизненного цикла программы,

весь комплект документации (по завершении каждого этапа) должен корректироваться [159]. Для этого составляется перечень исполняемых модулей программы, в котором для каждого из них указывается значение циклической контрольной суммы (сигнатура), отражающей любое изменение в модуле. При этом Контролер по документации должен знать значение этой суммы для исходного файла, и после завершения каждого этапа при любом изменении указанного значения требовать представления извещения на выполненную модификацию, по которому документация должна быть полностью откорректирована и сдана в архив.

Предлагаемый вариант технологии обладает следующими достоинствами:

- в отличие от объектного моделирования [99, 153], во-первых, построение всех основных моделей основано на применении только автоматной терминологии, а во-вторых, используется динамическая модель только одного типа — система взаимосвязанных графов переходов;

- применение такой динамической модели позволяет эффективно описывать и реализовывать задачи рассматриваемого класса даже при большой их размерности. Применение графов переходов в качестве языка спецификаций алгоритмов делает обозримым даже весьма сложное поведение программы и позволяет легко вносить изменения как в спецификацию, так и в ее реализацию;

- совместное рассмотрение схемы связей автомата и его графа переходов позволяет понимать этот граф, а совместное рассмотрение этого графа и изоморфной ему подпрограммы позволяет понимать эту подпрограмму;

- подробное документирование проекта создания программного обеспечения позволяет при необходимости вносить изменения в него через длительный срок после его выпуска и даже другими специалистами;

- без использования объектно-ориентированного подхода программа четко разделяется на две части — системонезависимую и системозависимую;

- при проектировании системонезависимой части программы детали реализации входных и выходных воздействий скрыты. Они раскрываются только при реализации системозависимой части программы;

- этапы проектирования и реализации системонезависимой части программы полностью разделены;

- реализация входных переменных и выходных воздействий в виде функций обеспечивает: их протоколирование, простоту перехода от одних типов источников и приемников информации к другим, наличие действующего макета программы [151] в любой момент времени после начала реализации системозависимой части;

— упорядоченное хранение функций, реализующих входные переменные и выходные воздействия, упрощает внесение изменений;

— для кодирования любого числа состояний автомата используется только одна внутренняя переменная, что обеспечивает надежность поведения автомата за счет «слежения» за изменениями значений только этой переменной. Для системы из N автоматов «слежение» выполняется по N многозначным переменным, значения каждой из которых выводятся на «отладочный» экран, представленный в форме, определяемой схемой взаимодействия автоматов;

— каждый граф переходов формально и изоморфно реализуется по шаблону, что при необходимости позволяет решить обратную задачу — однозначно восстановить граф переходов по этой подпрограмме;

— системонезависимая часть программы имеет регулярную структуру и, следовательно, легко читается и корректируется;

— системонезависимая часть программы зависит только от наличия компилятора или интерпретатора выбранного языка программирования на используемой платформе. При смене аппаратуры или переносе программы под другую операционную систему необходимо изменить только системозависимую часть. Если платформа не изменяется, то вспомогательные модули могут быть использованы как образцы [99] и повторно;

— автоматическое ведение протокола в терминах спецификации обеспечивает возможность сертификации программы. При этом демонстрируется соответствие функционирования программы «поведению» системы взаимосвязанных графов переходов для рассматриваемых событий при выбранных значениях входных переменных. Это достигается за счет сопоставления «полного» протокола со спецификацией. Совокупность «полных» протоколов обеспечивает возможность сертификации программы в целом. Для сертификации в терминах, понятных Заказчику, могут применяться «короткие» протоколы, которые можно использовать также и в качестве фрагментов методики проверки функционирования системы. При этом отметим, что из двух групп понятий «объект—алгоритм» и «алгоритм — программа» сертификация обычно связывается только со второй группой понятий;

— «короткий» протокол позволяет определить наличие ошибки в выдаче выходных воздействий, а «полный» — определить автомат, который при этом необходимо откорректировать. Поэтому «короткие» протоколы могут быть названы «проверяющими», а «полные» — «диагностирующими»;

— возможность автоматического получения «полных» протоколов в терминах автоматов показывает, что система взаимосвязанных графов переходов, используемая для спецификации алгоритмов, является не «картинкой», а математической моделью;

— несмотря на достаточно высокую трудоемкость проведения «подробной» сертификации при применении предлагаемых прото-

колов, этот способ существенно более практичен, чем другие подходы к построению качественных программ, которые изложены, например, в [171, 172];

— порождаемый некоторыми событиями протокол или его часть являются соответствующим сценарием. Таким образом, сценарий [161] строится автоматически при анализе программы, а не вручную при ее синтезе, как это предлагается делать при других подходах [101, 153]. Ручное построение всей совокупности сценариев и формальный синтез системонезависимой части программы по ним для задач со сложной логикой практически не осуществимы;

— предлагаемый подход не исключает интерактивной отладки и сертификации;

— поведение систем взаимосвязанных автоматов является разновидностью коллективного поведения автоматов [173] и может использоваться при построении «многоагентных систем, состоящих из реактивных агентов» [174].

Разработка системы управления дизель-генератором, выполненная с помощью предлагаемого варианта технологии [183], подтвердила мнение Ф. Брукса [151] о том, что время, затрачиваемое на проектирование алгоритма, при котором строятся не картинки, а математические модели, значительно превышает время, затрачиваемое на написание системонезависимой части программы, изоморфной спроектированному алгоритму, а также мнение Д. Воаса [152] о CASE-средствах, которые сами по себе мало что убыстряют, а «практика их применения показала только то, что из некорректных картинок можно получить некорректный код».

Текст подпрограммы, построенной с помощью предложенного варианта технологии, и «полный» протокол ее работы, проверяющий все переходы автомата, реализующего алгоритм управления объектом «тулбар», приведен в п. 3.2 Приложения.

Таким образом, в настоящее время SWITCH-технология используется при построении систем управления применительно ко всем трем типам управляющих вычислительных устройств, перечисленных в начале главы, и показала свою эффективность.

Из изложенного следует, что для рассматриваемого класса задач предлагаемая технология обеспечивает построение «добротных» [147] программ, подтверждая тем самым высказывание: «то, что не специфицировано формально, не может быть проверено, а то, что не может быть проверено, не может быть безошибочным» [32].

«Сложность служит причиной трудности перечисления, а тем более понимания, всех возможных состояний программы, а отсюда возникает ее ненадежность. Сложность служит также источником невизуализируемых состояний, в которых нарушается система защит» [151].

Предлагаемая технология основана на априорном задании состояний и их визуализации, и поэтому хочется надеяться, что она, обладая минимализмом [176], по крайней мере для систем логического управления и «реактивных» систем является приближением к «серебряной пуле» [151] в части создания качественных программ, тем более что Ф. Брукс в этой работе отозвался благосклонно только о подходе Д. Харела, сравнение с которым выполнено выше.

Эта технология может рассматриваться как одно из направлений выхода из «кризиса линейного мышления» [177], другое из которых, например, связано с заменой текста гипертекстом.

Применение изложенной технологии может быть особенно важным для обеспечения требований международного стандарта IEC 880 «Программное обеспечение ЭВМ, использующихся в системах эксплуатационной надежности атомных электростанций (АЭС)», который регламентирует в том числе и процесс разработки и контроля за созданием программного обеспечения для систем управления ядерными энергетическими установками, а также порядок внесения в него изменений [158].

Применение в предлагаемой парадигме понятия «автомат» в качестве центрального понятия соответствует его месту в теории управления, что принципиально отличает ее от других парадигм программирования.

Л и т е р а т у р а

1. *International standard IEC 1131-3. Programmable controllers. Part 3. Programming languages.* International Electrotechnical Commission. 1993.
2. *SIMATIC. Simatic S7/M7/C7. Programmable controllers.* SIEMENS. Catalog ST 70. 1996.
3. *TSX T607. Programming terminal. User's manual.* Telemecanique. 1987.
4. *Modicon catalog & specifier's guide.* Modicon. AEG Schneider Automation. 1995.
5. *Programmable controller. MELSEC-A. Programming manual. Type ACPU. Common instructions.* Mitsubishi Electric.
6. *ABB Procontic T200. Mid-range automation systems using modern technology.* Asea Brown Boveri. 1994.
7. *ET-PDS. Software for programmable logic controllers.* Toshiba International (Europe) Ltd. 1995.
8. *Шалыто А. А. SWITCH-технология. Алгоритмизация и программирование задач логического управления.* СПб.: Наука, 1998.
9. *Гузик И. М. Стандарт МЭК 1131: язык GRAFCET — знакомство поближе // Schneider Automation Club.* 1999. № 6.
10. *Лавров С. С. Лекции по теории программирования. Учебное пособие.* СПб.: СПбГТУ, Нестор, 1999.
11. *Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. М.: Бином; СПб.: Невский диалект, 1998.*
12. *Фаулер М., Скотт К. UML в кратком изложении. Применение стандартного языка объектного моделирования.* М.: Мир, 1999.
13. *Клир Д. Абстрактное понятие системы как методологическое средство // Исследования по общей теории систем.* М.: Прогресс, 1969.

14. Кузнецов Б. П. Структура и сложность модулей циклических программ // Автоматика и телемеханика. 1999. № 2.
15. Odell J. J. Advanced object-oriented analysis & design using UML. NY: SIGS Books, 1998.
16. Таль А. А., Айзерман М. А., Розоноэр Л. И. и др. Логика. Автоматы. Алгоритмы. М.: Физматгиз, 1963.
17. Шалыто А. А. Использование граф-схем алгоритмов и графов переходов при программной реализации алгоритмов логического управления // Автоматика и телемеханика. 1996. № 6, 7.
18. Глушков В. М. Синтез цифровых автоматов. М.: Физматгиз, 1962.
19. Yourdon E., Argila C. Case studies in object-oriented analysis & design. NJ: Yourdon Press, 1997. (Йордон Э., Аргила К. Структурные модели в объектно-ориентированном анализе и проектировании. М.: Лори, 1999).
20. Росс Д. Структурный анализ: язык для передачи понимания // Требования и спецификации в разработке программ. М.: Мир, 1984.
21. Chow T. S. Testing software design modeled by finite state machines // IEEE Trans. Soft. Eng. 1978. N 3.
22. King D. et al. On object state testing // Proceeding The Eighteenth Annual Inter. Comp. Software & Applications Conf. Los Alamitos: IEEE Computer Society Press, 1993.
23. Turner C. D., Robson D. J. The state-based testing of object-oriented programs // Conf. on Software Maintenance. Los Alamitos: IEEE Computer Society Press, 1993.
24. Jorgenson P. C., Erickson C. Object-oriented integration testing // Communications of the ACM. 1994. N 9.
25. Bunder R. V. The FREE-flow graph: implementation-based testing of objects using state-determined flows // Proc. 8th Annual Software Quality Week. San Francisco: Software Research Inc., 1995.
26. Martin R. C. Designing object-oriented C++ applications using the Booch method. NJ: Prentice-Hall, 1995.
27. Трахтенброт Б. А., Бардзинь Я. М. Конечные автоматы. Поведение и синтез. М.: Наука, 1970.
28. Гилл А. Введение в теорию конечных автоматов. М.: Наука, 1966.
29. Фридман А., Менон П. Теория и проектирование переключательных схем. М.: Мир, 1978.
30. Руднев В. В. Система взаимосвязанных графов и моделирование дискретных процессов // Автоматика и телемеханика. 1984. № 9.
31. Бадд Т. Объектно-ориентированное программирование в действии. СПб.: Питер, 1997.
32. Зайцев С. С. Описание и реализация протоколов сетей ЭВМ. М.: Наука, 1989.
33. Калянов Г. Н. CASE. Структурный анализ (автоматизация и применение). М.: Лори, 1996.
34. Шульц Д., Шульц С. История современной психологии. СПб.: Евразия, 1998.
35. *Functional description*. Warm-up & prelubrication logic. Generator control unit. Severnaya hull N 431. Norcontrol. 1993.
36. *Project 15640*. AS 21. DG 1. Control. АМН. 95564. 12М. St. Petersburg. ASS «Аврора». 1991.
37. Система управления турбокомпрессорным агрегатом «Ларина». Техническое описание. Приложение 1. АМН. 421417. 010 ТО. 01. СПб.: НПО «Аврора», 1998.
38. *Autolog 32*. Руководство пользователя. FF-Automation OY.
39. Баглюк Ю. В., Шалыто А. А. Программируемые логические контроллеры «Autolog». 124 примера программ на языке «ALPro», реализующих алгоритмы логического управления. СПб.: FF-Automation, 1999. На русском и английском языках.
40. *Транслятор «CF-ALPro»* для программирования контроллеров типа «Autolog». Руководство пользователя. NZF. TR 1. 1ПП (на русском языке); NZF. TR 2. 1ПП (на английском языке). FF-Automation, 1999.

41. Литов Д. В. Автомат или машина Тьюринга // Мир ПК. 1999. № 3.
42. Баранов С. И. Синтез микропрограммных автоматов (граф-схемы и автоматы). Л.: Энергия, 1979.
43. Любченко В. С. Мы выбираем, нас выбирают... (к проблеме выбора алгоритмической модели) // Мир ПК. 1999. № 3.
44. Затуливетер Ю. С., Халатян Т. Г. Синтез общих алгоритмов по демонстрациям частных примеров (автоматная модель обобщения по примерам). М.: Ин-т проблем управления, 1997.
45. Матвеев В. И. Windows CE — новый этап в развитии PLC // Приборы и системы управления. 1999. № 3.
46. Туккель Н. И., Шалыто А. А. Сравнение событийного и автоматного подходов к программированию задач логического управления // Телематика 99. Тез. докл. Всерос. научно-метод. конф. СПб.: СПбГИТМО (ТУ), 1999.
47. Tukkel N. Programming with use of a SWITCH-technology // Euroxchange. Special student's issue. ISA. 1999. N 2.
48. Шалыто А. А. Технология программной реализации алгоритмов логического управления как средство повышения живучести // Проблемы обеспечения живучести кораблей и судов. Тез. докл. научно-техн. конф. СПб.: НТО им. акад. А. Н. Крылова, 1992.
49. Shalyto A. A. Cognitive properties of hierarchical representations of complex logical structures // Architectures for semiotic modeling and situation analysis in large complex systems. 10th IEEE Intern. symposium on intelligent control. Monterey, California, 1995.
50. Шалыто А. А., Антупов В. В. Алгоритмизация и программирование задач логического управления техническими средствами. СПб.: Моринтех, 1996. Книжное издание и лазерный диск.
51. Bagljuk Y. V., Shalyto A. A. SWITCH-technology. Algorithmic and programming methods in solution the logic control problems of shipping equipment // Intern. conference on informatics and control. ICI & c'97. Proceedings. Vol. 1. St. Petersburg, 1997.
52. Кузнецов О. П., Макаревский А. Я., Марковский А. В. и др. Ярус — язык описания работы сложных автоматов // Автоматика и телемеханика. 1972. № 6, 7.
53. Кузнецов О. П. Графы логических автоматов и их преобразования // Автоматика и телемеханика. 1975. № 9.
54. Кузнецов О. П., Шитилина Л. Б., Марковский А. В. и др. Проблемы разработки языков логического программирования и их реализация на микроЭВМ (на примере языка «Ярус-2») // Автоматика и телемеханика. 1985. № 6.
55. Programmable controllers fundamentals. Allen-Bradley Inc., 1985.
56. Micro Mentor: understanding and applying micro programmable controllers. Allen-Bradley Inc., 1995.
57. Jones C. T., Bryan L. A. Programmable controllers concepts and applications. International Programmable Controllers Inc., 1983.
58. Gilbert R. A., Llewellyn J. A. Programmable controllers — practices and concepts. Industrial Training Corporation, 1985.
59. Lloyd M. Grafset — graphical functional charts programming for programmable controllers // Measurement & Control Magazine. 1987. N 9.
60. Bryan L. A., Bryan E. A. Programmable controllers: theory and implementation. Industrial Text Corp., 1988.
61. Webb J. W., Reis R. A. Programmable logic controllers: principles and applications. NJ: Prentice-Hall, 1995.
62. Wisnosky D. E. SoftLogic: overcoming funnel vision. Wizdom Controls Inc., 1996.
63. Hughes T. A. Programmable controllers. Instrument Society of America (ISA) Publications, 1997.
64. Шалыто А. А. Программная реализация управляющих автоматов // Судостроит. пром-сть. Сер. Автоматика и телемеханика. 1991. Вып. 13.
65. Боуэн Д., Хинчи М. Десять заповедей формальных методов // Мир ПК. 1997. № 9, 10.
66. Series 90-70. State logic control system. User's manual. NA: GE Fanuc Automation. 1998.

67. Мартынюк В. В. Об анализе графа переходов для операторной схемы // Журн. вычисл. математики и мат. физики. 1965. № 2.
68. Карнов Ю. Г. Основы построения компиляторов. Учебное пособие. Л.: Изд-во ЛПИ, 1982.
69. Касьянов В. Н., Поттосин И. В. Методы построения трансляторов. Новосибирск: Наука, 1986.
70. Байцер Б. Архитектура вычислительных комплексов. Т. 1. М.: Мир, 1974.
71. Буч Г. Объектно-ориентированное проектирование с примерами применения. Киев: Диалектика; М.: ИВК, 1992.
72. Страуструп Б. Язык программирования Си++. М.: Радио и связь, 1991.
73. Любченко В. С. Новые песни о главном (римейк для программистов) // Мир ПК. 1998. № 6, 7.
74. Shlaer S., Mellor S. Object Lifecycles: Modeling the World in State. NJ: Prentice-Hall, 1992. (Шлеер С., Меллор С. Объектно-ориентированный анализ: моделирование мира в состояниях. Киев: Диалектика, 1993).
75. Harel D. Statecharts: A visual formalism for complex systems // Sci. Comput. Program. 1987. Vol. 8.
76. Harel D. et al. STATEMATE: A working environment for the development of complex reactive systems // IEEE Trans. Eng. 1990. N 4.
77. Douglass B. P. UML Statecharts. MA: I-Logix Inc., 1998.
78. Unified Modeling Language (UML). Version 1.0. CA: Rational Software Corp., 1997.
79. Harel D., Pnueli A. On the development of reactive systems / Logic and model of computer systems. Ed. K. R. Apt. NY: Springer-Verlag, 1985.
80. Harel D. et al. On formal semantics of software // Pros. 2nd IEEE Symp. Logic in Computer Science. NY: IEEE Press, 1987.
81. Coleman D., Hayes E., Bear S. Introducing objectcharts, or how to use statecharts in object oriented design // IEEE Trans. Soft. Eng. 1992. N 1.
82. Harel D., Naamad A. The STATEMATE semantics of statecharts // ACM Trans. Soft. Eng. Metodology. 1996. N 10.
83. Harel D., Gery E. Executable object modeling with statecharts // Computer. 1997. N 7.
84. Harel D., Politi M. Modeling reactive systems with statecharts. NY: McGraw-Hill, 1998.
85. Douglass B. P. Real-time UML: Developing efficient objects for embedded systems. MA: Addison-Wesley, 1998.
86. Douglass B. P. Doing hard time: Using object oriented programming and software patterns in real time applications. MA: Addison-Wesley, 1998.
87. Карнов Ю. Г. Теория алгоритмов и автоматов. Курс лекций. СПб.: СПбГТУ, Нестор, 1998.
88. Information processing systems. Open systems interaction. ESTELLE: a formal description technique based on an extended state transition model // International standard. ISO 9074, 1989.
89. Bochman G. V. Finite state description of communication protocols // Comput. Network Protocol Symp. Liege, 1978, v. 2.
90. Dantine A. Protocol representation with finite state models // IEEE Trans. on Commun. 1980. N 4.
91. Brand D., Zafiropulo P. On communicating finite state machines // Journal ACM. 1983. N 2.
92. CCITT Recommendation Z. 100: CCITT Specification and description language (SDL) // COM X-R 26, ITU General Secretariat. Geneva, 1992.
93. Braek R., Haugen F. Engineering real time systems. NJ: Prentice-Hall, 1993.
94. Гольштейн Б. С. Сигнализация в сетях связи. М.: Радио и связь, 1997.
95. Иванов А., Кознов Д., Мурашова Т. Поведенческая модель RTST++ // Записки семинара кафедры системного программирования. CASE-средства RTST++. Вып. 1. СПб.: Изд-во СПбГУ, 1998.

96. Парфенов В. В., Терехов А. Н. RTST-технология программирования встроенных систем реального времени // Системная информатика. Вып. 5. Новосибирск: Наука, 1997.
97. Терехов А. Н. RTST-технология программирования встроенных систем реального времени // Записки семинара кафедры системного программирования. CASE — средства RTST++. Вып. 1. СПб.: Изд-во СПбГУ, 1998.
98. Долгов П., Иванов А., Терехов А. и др. Объектно-ориентированное расширение технологии RTST // Записки семинара кафедры системного программирования. CASE — средства RTST++. Вып. 1. СПб.: Изд-во СПбГУ, 1998.
99. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language. User guide. MA: Addison-Wesley, 1998. (Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя. М.: ДМК, 2000).
100. Microsoft. Решения '99 // Microsoft Corp. 1999. Вып. 7.
101. Rumbaugh J., Blaha M., Premerlani W. et al. Object-oriented modeling and design. NJ: Prentice-Hall. 1991.
102. Jacobson I. Object-oriented software engineering: A use case driven approach. MA: Addison-Wesley, 1992.
103. Cook S., Daniels J. Designing object systems. Object-oriented modeling with syntropy. NJ: Prentice-Hall, 1994.
104. STATEFLOW for use with Simulink. User's guide. Version 1. MA: Math Works, Inc. 1998.
105. MATLAB. The language of technical computing. Version 5. 2. MA: Math Works, Inc. 1998.
106. Зюбин В. Е. К пятилетию стандарта IEC 1131-3. Итоги и прогнозы // Приборы и системы управления. 1999. № 1.
107. Герп Р. Новый поворот // PC Magazine / Russian Edition. 1998. № 10.
108. Губанов Ю. А., Залманов С. З., Кузнецов Б. П. Управление автоматическими выключателями корабельной электроэнергетической системы // Третья международная конф. по морским интеллектуальным технологиям «Моринтех-99». СПб.: Моринтех, 1999. Т. 3.
109. Менделевич В. А. Непроцедурные языки — новое поколение средств разработки АСУТП // Промышленные АСУ и контроллеры. 2000. № 1.
110. Odell J. J. Approaches to finite-state machine modeling // Journal of object-oriented programming. 1995. N 1.
111. Heizinger T., Manna Z., Pnueli A. Timed transition systems. Technical report TR 92-1263. Dept. of Computer Science. Cornell University. 1992.
112. Ward P., Mellor S. Structured techniques for real-time systems. NJ: Yourdon Press—Prentice-Hall, 1985.
113. Hatley D., Pirbhai I. Strategies for real-time system specification. NY: Dorset House, 1987.
114. Dryinsky D. Visual programming. Better state. Product overview. Cupertino. California. R-Active Concepts, 1993.
115. Selic B. An efficient object-oriented variation of statecharts formalism for distributed real-time systems // CHDL'93: IFIP Conf. on hardware description languages and their applications. Ottawa, 1993.
116. Selic B., Gullekson G., Ward P. Real-time object-oriented modeling. NY: John Wiley & Sons Inc., 1994.
117. Borshchev A. V., Karpov Y. G., Roudakov V. V. COVERS — A tool for the design of real-time concurrent systems // Parallel Computing Technologies. Proceedings of the 3rd Inter. Conf. PACT-95. Lecture Note in Computer Science. 1995. N 964.
118. Booch G., Rumbaugh J. Unified method for object-oriented development. Documentation Set. Version 0. 8. Rational Software Corp., 1996.
119. Сосонкин В. Л., Мартинов Г. М., Любимов А. Б. Интерпретация диалога в Windows-интерфейсе систем управления // Приборы и системы управления. 1998. № 12.
120. Ran A. S. Modelling states as classes / Proc. of the tools USA 94. Ed. M. Singh, B. Meyer. NJ: Prentice-Hall, 1994.

121. *Ran A. S.* Patterns of events / Pattern languages of programm design. Ed. J. O. Coplien, D. C. Schmidt. MA: Addison-Wesley, 1995.
122. *Ilgum K., Kemmerer R., Porras P.* State transition analysis: a rule-based intrusion detection approach // IEEE Trans. on Software Eng. 1995. N 3.
123. *Corbett J. C.* Evaluating deadlock detection methods for concurrent software // IEEE Trans. on Software Eng. 1996. N 3.
124. *Alur R., Henzinger T., Pei-Hsin Ho.* Automatic symbolic verification of embedded systems // IEEE Trans. on Software Eng. 1996. N3.
125. *Heimdahl Mats P. E., Leveson N. G.* Completeness and consistency in hierarchical state-based requirements // IEEE Trans. on Software Eng. 1996. N 6.
126. *Ardis M. A. et al.* A framework for evaluating specification methods for reactive systems experience report // IEEE Trans. on Software Eng. 1996. N 6.
127. *Corbett J. C.* Timing analysis of Ada tasking programs // IEEE Trans. on Software Eng. 1996. N 7.
128. *Zave P., Jackson M.* Where do operations come from? A multiparadigm specification technique // IEEE Trans. on Software Eng. 1996. N 7.
129. *Coen-Porisini A., Ghezzi C., Kemmerer R.* Specification of real time systems using ASTRAL // IEEE Trans. on Software Eng. 1997. N 9.
130. *Avrynin G. S., Corbett J. C., Dillon L. K.* Analyzing partially-implemented real-time systems // IEEE Trans. on Software Eng. 1998. N 8.
131. *Секреты программирования игр* / А. Ла Мот, Д. Ратклифф, М. Семинаторе и др. СПб.: Питер, 1995.
132. *Шоломов Л. А.* Основы теории дискретных логических и вычислительных устройств. М.: Наука, 1980.
133. *Кузнецов О. П.* Неклассические парадигмы в искусственном интеллекте // Изв. РАН. Теория и системы управления. 1995. № 3.
134. *Антонов А. П., Мелехин В. Ф., Филиппов А. С.* Обзор элементной базы фирмы ALTERA. СПб.: ЭФО, 1997.
135. *Wirth N.* Digital circuit design. NY: Springer-Verlag, 1995.
136. *Wirth N.* Hardware compilation: translating programs into circuits // Computer. 1998. June. (Открытые системы. 1998. № 4—5).
137. *Армстронг Д.* Моделирование цифровых систем на языке VHDL. М.: Мир, 1992.
138. *MAX + PLUS II.* AHDL. Version 6. 0. CA: Altera, 1995.
139. *Xilinx Foundation.* M. 1. 5. CA: Xilinx, 1997.
140. *Clare C. R.* Designing logic systems using state machines. NY: McGraw-Hill, 1973.
141. *A framework for hardware-software Co-Design of embedded systems* // Technical report. Comp. Science Department. Univ. of California. Berkley. 1995.
142. *Лекции лауреатов премии Тьюринга за первые двадцать лет 1966—1985.* М.: Мир, 1993.
143. *Дейкстра Э.* Взаимодействие последовательных процессов // Языки программирования. М.: Мир, 1972.
144. *Дейтел Х. М., Дейтел П. Д.* Как программировать на C++. М.: Бином, 1999.
145. *TSX Nano.* PL7-07 language self-instruction manual. Groupe Schneider. 1997.
146. *Шалыто А. А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления // Промышленные АСУ и контроллеры. 1999. № 9.
147. *Поттосин И. В.* О критериях добротности программ // Системная информатика. Вып. 6. Новосибирск: Наука, 1998.
148. *xjCharts.* Release 2. 0. User's Manual. Experimental Object Technologies. 1999.
149. *Питильер К.* Синхронный C++ для интерактивных приложений // Открытые системы. 1999. № 3.
150. *Кук Д., Урбан Д., Хамилтон С.* Unix и не только. Интервью с Кеном Томпсоном // Открытые системы. 1999. № 4.
151. *Брукс Ф.* Мифический человеко-месяц или как создаются программные системы. СПб.: Символ, 2000.
152. *Воас Д.* Качество ПО: восемь мифов // Открытые системы. 1999. № 9—10.

153. Терехов А. Н., Романовский К. Ю., Кознов Д. В. и др. REAL: Методология и CASE-средство разработки информационных систем и программного обеспечения систем реального времени // Программирование. 1999. № 5.
154. Кознов Д. В. Конечный автомат — основа для визуальных представлений поведения объектов // Объектно-ориентированное визуальное моделирование. СПб.: Изд-во СПбГУ, 1999.
155. Любченко В. С. О бильярде с Microsoft Visual C++ 5. 0 // Мир ПК. 1998. № 1.
156. Любченко В. С. Задача Майхилла для Microsoft Visual C++ 5. 0 (о синхронизации процессов в среде Windows) // Мир ПК. 2000. № 2.
157. Романовский К. Ю., Кузнецов С. В., Кознов Д. В. Объектно-ориентированный подход и диаграммы классов в UML // Объектно-ориентированное визуальное моделирование. СПб.: Изд-во СПбГУ, 1999.
158. Астров В. В., Василенко В. С., Тотъменинов Л. В. Вопросы создания интегрированных систем управления ядерными энергетическими установками // Системы управления и обработки информации. СПб.: НПО «Аврора», 2000.
159. Липаев В. В. Документирование и управление конфигурацией программных средств. Методы и стандарты. М.: Синтег, 1998.
160. Гудман С., Хидетниemi С. Введение в разработку и анализ алгоритмов. М.: Мир, 1981.
161. Верников Г. Основы обследования деятельности организаций. Стандарт IDEF3 // READ. ME. 2000. № 2.
162. ADAM-5510/P31 / Все необходимое для автоматизации на базе PC. Advantech. 1999. Т. 91.
163. Бабаджан Ф. Э. SCADA-пакет «Virgo» // Промышленные АСУ и контроллеры. 1999. № 10.
164. Промышленные контроллеры фирмы «Matsushita». Matsushita Automation Controls.
165. Программное обеспечение для программируемых контроллеров и визуализации. Festo Cybernetic.
166. Любашии А. Промышленные и встраиваемые системы. На «стандартном» пути // PC WEEK. 2000. № 15.
167. Канер С., Фолк Д., Енг Кек Нгуен. Тестирование программного обеспечения. Киев: DiaSoft, 2000.
168. Шалыто А. А. Реализация алгоритмов логического управления программой на языке функциональных блоков // Промышленные АСУ и контроллеры. 2000. № 4.
169. Шалыто А. А., Туккель Н. И. SWITCH-технология — автоматный поход к созданию программного обеспечения «реактивных» систем // Телематика 2000. Тез. докл. Международной научно-метод. конф. СПб.: СПбГИТМО (ТУ), 2000.
170. Бурдонов И. Б., Косачев А. С., Кулямин В. В. Использование конечных автоматов для тестирования программ // Программирование. 2000. № 2.
171. Непомнящий В. А., Рякин О. М. Прикладные методы верификации программ. М.: Радио и связь, 1988.
172. Woodcock J., Davies J. Using Z-specification. Refinement and proof. Oxford: Oxford University Press, 1995.
173. Варшавский В. И. Коллективное поведение автоматов. М.: Наука, 1973.
174. Круглый стол «Парадигмы искусственного интеллекта» // Новости искусственного интеллекта. 1998. № 3.
175. Пригожин И., Стенгерс И. Порядок из хаоса. М.: Эдиториал УПСС, 2000.
176. Герр Р. Отладка человечества // PC Magazine / Russian Edition. 2000. № 5.
177. Герняк Л. XML, взгляд со стороны // Открытые системы. 2000. № 4.
178. Kurshan R. P. Computer-aided verification of coordinated processes — the automata-theoretic approach. Princeton University Press, 1994.
179. Kurshan R. P. Program verification // Notices of the ACM. 2000. N 5.

180. *Боггс У., Боггс М.* UML и Rational Rose. М.: Лори, 2000.
181. *Тьюринг А.* Может ли машина мыслить? Саратов: Колледж, 1999.
182. *Романовский И. В.* Дискретный анализ. СПб.: Невский диалект, 2000.
183. *Шалыто А. А., Туккель Н. И.* SWITCH-технология — автоматный подход к созданию программного обеспечения «реактивных» систем // Промышленные АСУ и контроллеры. 2000. № 10.
184. *Дейл Н., Уимз Ч., Хедингтон М.* Программирование на C++. М.: Изд-во ДМК, 2000.
185. *Кузнецов Б. П.* Психология автоматного программирования // BYTE / Россия. 2000. № 11.

Заключение

Итак, эта книга закончена. В ней удалось изложить практически все результаты, которые были получены автором самостоятельно и в соавторстве за почти тридцать лет работы в области логического управления.

Эти исследования начинались в начале 70-х годов в период расцвета научных работ в этой области, когда только в СССР этими вопросами занимались сотни ученых, а заканчиваются в России, когда «одних уж нет, а те далече».

За эти годы, на первый взгляд, многое изменилось: элементная база, критерии, ограничения... Действительно, если раньше основные усилия ученых были направлены на минимизацию аппаратуры и повышение ее быстродействия, то теперь во многих случаях эти вопросы часто уходят на второй план и имеет место «дрейф» в область вычислительной техники, характеристики которой улучшаются с каждым «днем». Это приводит к общему снижению интереса к научным результатам в области логического управления и все более широкому практическому применению систем и устройств этого класса.

Однако, по мнению автора, снижение интереса к научным исследованиям и публикациям в этой области связано с недостаточным пониманием изменившейся ситуации с использованием систем логического управления и роли и красоты математического аппарата, применяемого для описания этого класса систем.

Выскажу несколько соображений по этому поводу.

1. Понятие «современная элементная база» является относительным, так как в каждой достаточно крупной организации, давно работающей в области автоматизации, параллельно ведутся работы по проектированию систем на различных типах элементной базы (от реле до микропроцессоров). При этом применение «старой» элементной базы объясняется тем, что объекты автоматизации имеют срок службы в несколько десятилетий, а системы управления имеют меньший срок службы, и поэтому совершенно естественно при замене

системы повторить ее, используя «старую» элементную базу. Такая же ситуация имеет место и при модернизации старых систем или разработке их модификаций. Более того, если на одном объекте, сданном в эксплуатацию много лет назад, установлена некоторая система управления, то при строительстве нового аналогичного объекта с целью сокращения эксплуатационных затрат (включая обучение обслуживающего персонала) Заказчику выгодно иметь и систему управления, максимально похожую на «старую».

2. Из изложенного в предыдущем пункте следует, что в каждый момент времени в организациях указанного типа должны быть Специалисты, обладающие знаниями в области построения систем управления на элементной базе разного типа, в то время как высшие учебные заведения в основном дают знания по направлениям, «модным» в настоящее время. При этом «связь времен» могут обеспечить лишь книги, старые и выпускаемые в настоящее время, которые не подвержены «увольнению, уходу на пенсию или смерти».

3. Естественно, что для систем на «старой» элементной базе «старые» критерии и ограничения сохраняются, а для систем на «новой» элементной базе, позволяющей снять многие из предыдущих ограничений, появляются новые критерии и ограничения.

Эти критерии в основном связаны с большой ответственностью объектов, в которых используются системы логического управления, так как аварии на таких объектах могут приводить к катастрофам различных типов.

Если до последнего времени в основном требовалось обеспечить работоспособность таких систем, что могло быть достигнуто на основе инженерного опыта, то в последнее время при проектировании систем этого класса на первый план выходит вопрос о доказательстве правильности (верификации) построенных алгоритмов, схем и программ, что не может быть обеспечено без применения математических методов анализа и синтеза, некоторые из которых изложены в настоящей работе.

4. На решение практических вопросов, возникающих при алгоритмизации и программировании систем логического управления и «реактивных» систем, направлена разработка SWITCH-технологии, описанной в гл. 21. При этом введены такие понятия, как «автоматное программирование», «автоматное проектирование программ» и «наблюдаемость программ».

Если эта глава имеет в основном методическую и практическую направленность, то большинство из остальных глав книги связаны с описанием полученных научных результатов, которые представляют не только теоретический интерес, но и применялись и могут применяться в дальнейшем при проектировании элементов, устройств и систем логического управления.

5. Многолетний теоретический интерес автора к рассматриваемой области связан с тем, что, по его мнению, булева алгебра явля-

ется фундаментом всех наук, входящих в информационные технологии. Более того, она, видимо, является наиболее «красивой» из наук вообще, ведь недаром физик-теоретик академик А. Б. Мигдал, которому было с чем сравнивать, в книге, посвященной красоте в науке, выделяет именно булеву алгебру, которая всего лишь при двух значениях переменных и нескольких простейших операциях стала основой всех информационных технологий.

«По свидетельству П. Лапласа, Г. Лейбниц (один из создателей дифференциального исчисления) был буквально одержим страстью выражать все числа посредством двух символов — единицы и нуля. В его представлении только такая система и имела право на существование, поскольку в ее основе лежала духовная концепция мироздания. Единица, по Лейбницу, соответствовала образу божественного начала, а ноль отображал собой небытие, пустоту. По его мнению, как Всевышний создал все из небытия и своего всемогущества, так из единицы и нуля возникли все остальные числа. Таким образом, Лейбниц пришел к компромиссу, который состоял в том, чтобы сочленить разум с верой. Результатом этого компромисса и стала двоичная система, способная „поверить гармонию алгебры”» (Бернатосян С. Г. Воровство и обман в науке. СПб.: Эрудит, 1998).

6. Особую важность булевой алгебры подтверждают и результаты, полученные в настоящей работе. Действительно, рассматривая простейший «объект» булевой алгебры, каким являются неповторные формулы в базисе И, ИЛИ, НЕ, удалось установить их связь с такими фундаментальными понятиями математики (в основном дискретной), как контактные схемы (разд. 1.1), деревья (разд. 1.1), эйлеровы графы (разд. 13.2), клики в графах (разд. 12.2), линейные бинарные графы — схемы алгоритмов и программ (гл. 15), разбиения (разд. 1.1), числа Фибоначчи (разд. 1.4.3 и 15.7) и число «е» (разд. 12.2 и 15.9).

При этом отметим, что исследованием числа неповторных параллельно-последовательных контактных схем занимался и «отец» теории информации К. Э. Шеннон. Да и с другими отцами в этой области был полный порядок: Дж. Буль был отцом Этель Лилиан Войнич, а лорд Дж. Байрон — отцом графини Ады Лавлейс, заложившей основы современного программирования и имя которой увековечено в названии одного из известнейших языков программирования (АДА).

На базе исследования неповторных формул автору удалось разработать метод, названный формульным, для реализации булевых формул схемами из модулей, при построении которых также использовались неповторные формулы. Этот метод обеспечивает получение линейных (от числа букв) оценок сложности схем (гл. 2).

Другой формульный метод, также основанный на свойствах неповторных формул, позволяет реализовать произвольные нормальные булевы формулы в указанном базисе из h букв линейными и

планарными бинарными графами, состоящими из $h + 2$ вершин (разд. 15.1).

Таким образом, неповторные формулы в информатике можно сопоставить с «мухой дрозофилой», сыгравшей важную роль в генетике. Кстати, ее геном полностью расшифровали только в 2000 г., а в 1996 г. результаты исследований свойств другой мухи (майской) были отмечены Нобелевской премией в области биологии, что в очередной раз свидетельствует о том, что в «жизни всегда есть место подвигам».

7. Особый интерес представляет указанная связь неповторных формул с числами Фибоначчи, которые, в свою очередь, связаны с «золотым сечением», так как современная наука считает, что если в каком-либо объекте исследований «появляются» эти числа, то это является несомненным признаком его красоты. Указанный факт, видимо, связан со спецификой представления о красоте Человеком, так как физиологами установлено, что нормальные параметры ритмов его мозга также связаны с этими числами.

8. Свойство красоты, которое, естественно, не всегда связано с числами Фибоначчи, присуще и более сложным классам булевых функций, например пороговым, которые могут задаваться весами переменных и порогом и являются основой такого важного направления искусственного интеллекта, как нейронные сети.

При этом именно неповторные пороговые формулы, в которых операции И и ИЛИ чередуются, имеют веса и пороги, соответствующие числам Фибоначчи. Эти числа определяют также количество путей в линейных бинарных графах, вычисляющих такие формулы при одном порядке переменных, изменив который на противоположный, указанное количество путей резко сокращается, так как верхняя экспоненциальная оценка количества путей заменяется на нижнюю линейную оценку.

К классу пороговых функций относятся и мажоритарные функции, описывающие работу мажоритарных элементов, синтез схем из которых рассмотрен в гл. 4. Значение этих элементов для построения отказоустойчивых систем было установлено одним из гениев XX века Дж. фон Нейманом, сфера научных интересов которого пролегла от химии и теории игр до основ вычислительной техники и самоорганизующихся систем.

9. Замечательными свойствами обладают линейные функции (разд. 1.7), роль которых в теории кодирования нельзя переоценить. Один код Хэмминга чего стоит!

При этом нельзя не отметить, что мажоритарная и линейная функции трех переменных описывают одноразрядный двоичный сумматор, являющийся основой всей вычислительной техники.

10. Красота симметрических функций, которые могут задаваться рабочими числами, и самодвойственных функций, которые двойственны сами себе, видна даже при беглом просмотре гл. 9 и 10.

11. Связь булевых функций с функциями Радемахера, матрицами Уолша и Адамара (разд. 19.4), лежащими в основе дискретного преобразования Фурье, вводит булевы функции в общее русло математики и цифровой обработки данных.

12. Разновидности дискретного преобразования Фурье (в том числе быстрого) связывают полиномы Жегалкина и совершенные нормальные формы, но что более удивительно, они связывают также (через кронекеровские степени двоичной матрицы стандартного вида) системы булевых функций с одним арифметическим полиномом, в общем случае нелинейным. Определены условия реализации одной булевой функции и систем булевых функций с помощью одного линейного арифметического полинома. Предложены алгоритмические структуры, использующие линейные полиномы (гл. 20).

13. Уникальность свойств такого элемента, как мультиплексор, который может использоваться в качестве коммутатора каналов и логического модуля, универсального в классе произвольных функций, позволило автору почти через 30 лет, прошедших с момента постановки задачи, во время написания этой книги неожиданно просто решить задачу о реализации произвольной булевой функции схемой из произвольных (априори известных) логических элементов (гл. 3—6) с помощью метода, названного мультиплексорным, связав решение этой задачи с разработкой метода построения многофункциональных логических модулей (гл. 7).

14. Свойства мультиплексоров позволили также: предложить распределительный метод реализации произвольных булевых функций схемами (разд. 5.1.2) и бинарными графами (разд. 16.4.1), дополняющий канонический метод Блоха; установить, что в линейных бинарных графах формулы вычисляются не по подформулам, а по фрагментам (разд. 15.2); предложить устройство последовательностного действия, обеспечивающее по-буквенное вычисление формул (разд. 18.1.11); изоморфно реализовать программами графы переходов с многозначным кодированием вершин (разд. 17.6).

15. Простой аналитический метод построения структурированных бинарных графов для систем булевых формул изложен в разд. 16.3. Этот метод напоминает метод построения параллельно-последовательной контактной схемы по формуле.

16. Установлены свойства булевых функций, позволяющие реализовать их неповторными однородными каскадами Майтра, и предложены однородные структуры различных типов для реализации булевых формул.

17. Совместное рассмотрение в одной работе методов аппаратной и программной реализации алгоритмов логического управления, и в частности методов построения схем из мультиплексоров, мажоритарных элементов и демультимплексоров непосредственно по схемам алгоритмов, применяемым в программировании, соответст-

вует предложенному Н. Виртом (*Wirth N. Hardware compilation: translating programs into circuits // Computer. 1998. June*) подходу к построению схем из массивов вентилях путем трансляции программ. Более того, как утверждает Н. Вирт, по программе, написанной на одном языке, некоторые ее части могут компилироваться в последовательность команд процессора, а другие — транслироваться в схемы из вентилях. Интересным является также и то, что один из классиков программирования выпустил книгу, посвященную проектированию цифровых схем (*Wirth N. Digital circuit design. New York: Springer-Verlag, 1995*).

18. Проходит время и вдруг выясняется (после разговоров с бывшими соотечественниками), что автор был одним из «пионеров» создания программируемых логических схем, так как три десятилетия назад большинство инженеров считало, что простые вентиля при построении нерегулярных логических схем решают все их проблемы, а до СБИС с программируемой структурой было еще очень далеко.

Думаю, что и в области применения автоматов при программировании задач логического управления я был в «пионерах».

19. И последнее. Многие приходят в этот мир, чтобы «перевернуть» его. Как отмечалось выше, булева алгебра с ее простейшими средствами с помощью усилий многих ученых и инженеров «перевернула» мир информационных технологий.

Как ни странно, но одна из простейших (по формулировке) задач булевой алгебры, возможно, сможет изменить и значительную часть научного мировоззрения.

Для этого необходимо доказать (или опровергнуть, что было бы существенно хуже для практики), что с полиномиальной оценкой времени вычислений может быть решена задача выполности: для произвольной булевой функции, зависящей от n переменных и заданной в конъюнктивной нормальной форме, определить, существует ли хотя бы один набор этих переменных, при котором функция равна единице.

Интересно отметить, что даже столь простая формулировка задачи может быть еще более упрощена, без уменьшения эффекта от решения задачи в новой постановке.

Такая задача носит название «3-выполнимость» и отличается от задачи выполнимости, сформулированной выше, тем, что в новой постановке каждая дизъюнкция в конъюнктивной нормальной форме содержит только три буквы.

На первый взгляд, кажется странным, что решение задачи в столь простой формулировке может «перевернуть» мир, однако про нее до к а з а н о, что она принадлежит к классу с а м ы х т р у д н ы х по времени вычисления задач, а также до к а з а н о то, что л ю б а я другая задача из этого класса, к которому, например, принадлежит и «задача о коммивояжере», может быть сведена к ней за полиномиальное время.

При этом отметим, что, в свою очередь, формулировка задачи 3-выполнимости не может быть упрощена, так как доказано, что задача 2-выполнимости имеет полиномиальный алгоритм решения.

Отметим также, что в литературе высказано предположение, что для решения задачи 3-выполнимости полиномиальным алгоритмом по крайней мере потребуется крупное научное открытие.

Так что, если Вы захотите с интересом провести время (и, возможно, немалое), возьмитесь за решение этой задачи, в чем Вам, может быть, поможет книга М. Гэри и Д. Джонсона «Вычислительные машины и труднорешаемые задачи» (М. : Мир, 1982) или книга Т. Кормена, Ч. Лейзерсона и Р. Ривеста «Алгоритмы. Построение и анализ» (М.: Московский центр непрерывного математ. образования (МЦНМО), 1999), особенно учитывая тот факт, что для произвольной конъюнктивной нормальной формы, не содержащей дизъюнкций длины более трех, установлена связь времени ее вычисления с «золотым сечением» (*Гури Э. А.* Теоретические оценки времени работы алгоритмов для задачи выполнимости булевых формул: Автореф. дис. на соиск. учен. степени канд. физ.-мат. наук. СПб.: СПбГУ, 1998).

Конечно, кроме этой Задачи существуют и другие задачи в рассматриваемой области и хочется надеяться, что их решение будет связано с получением новых и красивых результатов, применимых в логическом управлении и в теории автоматов, математический аппарат которой «является одной из наиболее трудных областей математики, так как мало соприкасается с непрерывностью, рассматриваемой в наиболее разработанной области математики — математическом анализе» (*Дж. фон Нейман.* Общая и логическая теория автоматов // *Тьюринг А.* Может ли машина мыслить? Саратов: Колледж, 1999).

ПРИЛОЖЕНИЕ

СРАВНЕНИЕ СОБЫТИЙНОГО И АВТОМАТНОГО ПОДХОДОВ К ПРОГРАММИРОВАНИЮ УПРАВЛЕНИЯ

Пример 1. Программная реализация R-триггера

1.1. Событийный подход

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
int R,S,Z=0;
void main()
{
    for(;;){
        clrscr();
        cout << "\n\n\r Задайте R,S";
        cout << "\n Z=" << Z;
        cout << "\n R="; cin >> R; if(R>1) break;
        cout << " S=";    cin >> S; if(S>1) break;
            if(S)        Z=1;
            if(R)        Z=0;
    }
}
```

1.2. Автоматный подход. Используется модель автомата без входного преобразователя

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
int R, S, Z=0;
void main()
{
    for(;;){
        clrscr();
        cout << "\n\n\r Задайте R,S";
        cout << "\n Z=" << Z;
        cout << "\n R="; cin >> R; if(R>1) break;
        cout << " S=";    cin >> S; if(S>1) break;
        switch(Z){
            case 0: if(!R&S) Z=1;
                    break;
            case 1: if(R)      Z=0;
                    break;
        }
    }
}
```

```

    }
}
}

```

Пример 2. Программная реализация счетного триггера

2.1. Событийный подход

```

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
int x,y=0,z=0;
void main()
{
for(;;){
clrscr();
cout << "\n\n\r Задайте x";
cout << "\n z=" << z;
cout << "\n y=" << y;
cout << "\n x="; cin >> x; if(x>1) break;
switch(x){
case 0: if(z) y=1;
else y=0;
break;
case 1: if(y) z=0;
else z=1;
break;
}
}
}

```

2.2. Автоматный подход. Используется модель автомата Мура второго рода

```

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
int x, Y=0, z=0;
void main()
{
for(;;){
clrscr();
cout << "\n\n\r Задайте x";
cout << "\n Y=" << Y;
cout << "\n z=" << z;
cout << "\n x="; cin >> x; if(x>1) break;
switch(Y){
case 0: if(x) {Y=1;z=1;}
break;
case 1: if(!x){Y=2;z=1;}
break;
}
}
}

```

```

        case 2: if(x) {Y=3;z=0;}
                break;
        case 3: if(!x){Y=0;z=0;}
                break;
    }
}

```

Пример 3. Фрагмент программы, реализующей некоторые функции управления элементом графического пользовательского интерфейса «тулбар»

Н.И.Туккель, А.А.Шалыто

3.1. Событийный подход

```

//===== toolbars.h =====
#ifndef _TOOLBARS_H_INCLUDED_
#define _TOOLBARS_H_INCLUDED_

#include <Pt.h> // Подключение библиотеки
               // Photon.
#define TBMAXCOUNT 10 // Максимальное количество
                       // тулбаров.

//---
// Кнопка тулбара
//
typedef struct
{
    int toggle ;
    int function_index ;
} toolbar_btn_t ;

//---
// Тулбар
//
typedef struct
{
    toolbar_btn_t buttons[30] ; // Массив кнопок тулбара
    int          btn_count ;    // Количество кнопок в
                                // тулбаре
    PtWidget_t    *wgt, *pane ; // wgt - окно тулбара,
                                // pane - виджет PtPane
                                // внутри окна
    PhPoint_t     pos ;         // Положение тулбара
    int           menu ;        // Переменная состояния
                                // тулбара
    PhPoint_t     drag_pos ;    // Положение курсора мыши
                                // в начале движения
    int           cfg_index ;    // Индекс тулбара
                                // в массиве конфигураций
} toolbar_t ;
#endif

```

```

//===== toolbars.c =====
// массив toolbars[] описан как
// toolbar_t toolbars[TBMAXCOUNT] ;
//=====
// Обработчик события "нажатие правой кнопки мыши
// при нахождении ее курсора на тулбаре"
//
int toolbar_btn_press( PtWidget_t *widget,
                      ApInfo_t *apinfo,
                      PtCallbackInfo_t *cbinfo )
{
    // Используется для получения номера тулбара
    PtArg_t      args[1] ;
    // Указатель на номер тулбара
    int          *number ;
    // Координаты курсора мыши
    PhRect_t     *rect ;
    // Произошедшее событие
    PhEvent_t     *event = cbinfo->event ;
    // Дополнительная информация о событии
    PhPointerEvent_t *edata = PhGetData(event) ;

    // Получить номер тулбара, для которого был вызван
    // обработчик события
    PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
    PtGetResources( widget, 1, args ) ;

    if( edata->buttons&Ph_BUTTON_MENU )
        // Была нажата правая кнопка мыши
        {
            rect = PhGetRects( event ) ;

            // Запомнить координаты курсора мыши
            toolbars[*number].drag_pos = rect->ul ;

            // Переместить окно тулбара выше всех остальных
            PtWindowToFront( toolbars[*number].wgt ) ;

            // Запомнить, что была нажата правая кнопка мыши
            toolbars[*number].menu = 1 ;
        } ;
    return( Pt_CONTINUE ) ;
} ;

```

```

//=====
// Обработчик события "отпускание правой кнопки мыши при
// нахождении ее курсора на тулбаре"
//
int toolbar_btn_release( PtWidget_t *widget,
                        ApInfo_t *apinfo,
                        PtCallbackInfo_t *cbinfo )
{
    // Используется для получения номера тулбара
    PtArg_t          args[1] ;
    // Указатель на номер тулбара
    int              *number ;
    // Произошедшее событие
    PhEvent_t        *event = cbinfo->event ;
    // Дополнительная информация о событии.
    PhPointerEvent_t *edata = PhGetData( event ) ;

    //Получить номер тулбара, для которого был вызван
    //обработчик события
    PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
    PtGetResources( widget, 1, args ) ;

    if( event->subtype==Ph_EV_RELEASE_REAL
        && edata->buttons&Ph_BUTTON_MENU )
        // Была отпущена правая кнопка мыши

        if( toolbars[*number].menu == 1 )
            // Перед этим была нажата правая кнопка мыши
            {
                // Отобразить меню
                ApCreateModule( ABM_toolbar_menu, NULL, NULL ) ;
            } ;

        toolbars[*number].menu = 0 ;

    return( Pt_CONTINUE ) ;
} ;

//=====
// Обработчик события "перемещение мыши с нажатой правой
// кнопкой при нахождении ее курсора на тулбаре"
//
int toolbar_btn_move( PtWidget_t *widget,
                    ApInfo_t *apinfo,
                    PtCallbackInfo_t *cbinfo )
{
    // Используется для получения номера тулбара
    PtArg_t          args[1] ;
    // Указатель на номер тулбара
    int              *number ;

```

```

// Координаты события
PhRect_t      *rect ;
// Произошедшее событие
PhEvent_t     *event = cbinfo->event ;
// Дополнительная информация о событии
PhPointerEvent_t *edata = PhGetData( event ) ;

// Если не нажата правая кнопка мыши - завершить
// выполнение функции
if(!edata->buttons&Ph_BUTTON_MENU) return Pt_CONTINUE;

//Получить номер тулбара, для которого был вызван
//обработчик события
PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
PtGetResources( widget, 1, args ) ;

// Переместить тулбар вслед за курсором мыши
rect = PhGetRects( event ) ;
toolbars[*number].menu = 0 ;
toolbar_move( &toolbars[*number],
              rect->ul.x - tb->drag_pos.x,
              rect->ul.y - tb->drag_pos.y ) ;
return( Pt_CONTINUE ) ;
} ;

//=====
// Обработчик события "пересечение курсором мыши
// границы тулбара"
//
int toolbar_boundary( PtWidget_t *widget,
                     ApInfo_t *apinfo,
                     PtCallbackInfo_t *cbinfo )
{
    // Используется для получения номера тулбара
    PtArg_t  args[1] ;
    // Указатель на номер тулбара
    int      *number ;

    //Получить номер тулбара, для которого был вызван
    //обработчик события
    PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
    PtGetResources( widget, 1, args ) ;

    toolbars[*number].menu = 0 ;

    return( Pt_CONTINUE ) ;
} ;

```

3.2. Автоматный подход. Используется модель смешанного автомата

```
//===== toolbars.h =====

#ifndef _TOOLBARS_H_INCLUDED_
#define _TOOLBARS_H_INCLUDED_

#include <Pt.h>

#define TBBTN_SIZE 24
#define TBMAXCOUNT 10

//=====
// Кнопка тулбара
//
typedef struct
{
    int toggle ;
    int function_index ;
} toolbar_btn_t ;

//=====
// Тулбар
//
typedef struct
{
    toolbar_btn_t buttons[30] ; // Массив кнопок тулбара
    int btn_count ; // Количество кнопок
                        // в тулбаре
    PtWidget_t *wgt, *pane ; // wgt - окно тулбара
                        // pane - виджет PtPane
                        // внутри окна
    PhPoint_t pos ; // Положение тулбара
    int y0 ; // Переменная состояния
                        // тулбара
    PhPoint_t drag_pos ; // Положение курсора мыши
                        // в начале движения
    int cfg_index ; // Индекс тулбара
                        // в массиве конфигураций
} toolbar_t ;

int toolbars_create() ;
void toolbars_config_read() ;
void toolbars_ontop() ;

#endif
```

```

//===== toolbars.c =====

//
// Модуль, реализующий управление тулбаром
//

#include "photon_stuff.h"

#include "toolbars.h"
#include "config.h"
#include "status.h"
#include "functions.h"

toolbar_t toolbars[TBMAXCOUNT] ;

//=====
// Перемещение указанного тулбара
// на заданное расстояние
//
void toolbar_move( toolbar_t *tb, int inc_x, int inc_y )
{
    PtArg_t    args[1] ;
    PhPoint_t  *pos = NULL, new_pos = {0, 0} ;
    short      abs_x = 0, abs_y = 0 ;

    PtSetArg( args, Pt_ARG_POS, &pos, 0 ) ;
    PtGetResources( tb->wgt, 1, args ) ;
    new_pos.x = pos->x + inc_x ;
    new_pos.y = pos->y + inc_y ;
    PtSetArg( args, Pt_ARG_POS, &new_pos, 0 ) ;
    PtSetResources( tb->wgt, 1, args ) ;

    PtGetAbsPosition( ABW_base, &abs_x, &abs_y ) ;
    cfg[tb->cfg_index].value.INT = new_pos.x - abs_x ;
    cfg[tb->cfg_index+1].value.INT = new_pos.y - abs_y ;
} ;

//=====
// Обработчик события "нажатие правой кнопки мыши при
// нахождении ее курсора на тулбаре"
//
int toolbar_bpress( PtWidget_t *widget,
                    ApInfo_t *apinfo,
                    PtCallbackInfo_t *cbinfo )
{
    PtArg_t      args[1] ;
    int          *number = NULL ;
    toolbar_t    *tb = NULL ;
    PhEvent_t     *event = cbinfo->event ;
    PhPointerEvent_t *edata = PhGetData( event ) ;

```



```

if( edata->buttons == Ph_BUTTON_MENU )
{
    PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
    PtGetResources( widget, 1, args ) ;
    tb = &toolbars[*number] ;

    // Вызвать управляющий автомат
    A0( 10, tb, event ) ;
} ;

return( Pt_CONTINUE ) ;
} ;

//=====
// Обработчик события "отпускание правой кнопки мыши при
// нахождении ее курсора на тулбаре"
//
int toolbar_brelease( PtWidget_t *widget,
                      ApInfo_t *apinfo,
                      PtCallbackInfo_t *cbinfo )
{
    PtArg_t          args[1] ;
    int               *number = NULL ;
    toolbar_t         *tb = NULL ;
    PhEvent_t         *event = cbinfo->event ;
    PhPointerEvent_t  *edata = PhGetData( event ) ;

    if( event->subtype == Ph_EV_RELEASE_REAL
        && edata->buttons == Ph_BUTTON_MENU )
    {
        PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
        PtGetResources( widget, 1, args ) ;
        tb = &toolbars[*number] ;

        // Вызвать управляющий автомат
        A0( 20, tb, event ) ;
    } ;

    return( Pt_CONTINUE ) ;
} ;

//=====
// Обработчик события "перемещение мыши
// с нажатой правой кнопкой при
// нахождении ее курсора на тулбаре"
//
int toolbar_bmove( PtWidget_t *widget, ApInfo_t *apinfo,
                  PtCallbackInfo_t *cbinfo )
{
    PtArg_t          args[1] ;
    int               *number = NULL ;

```

```

toolbar_t      *tb = NULL ;
PhEvent_t      *event = cbinfo->event ;
PhPointerEvent_t *edata = PhGetData( event ) ;

if( edata->buttons == Ph_BUTTON_MENU )
{
    PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
    PtGetResources( widget, 1, args ) ;
    tb = &toolbars[*number] ;

    // Вызвать управляющий автомат
    A0( 30, tb, event ) ;
} ;

return( Pt_CONTINUE ) ;
} ;

//=====
// Обработчик события "пересечение курсором мыши границы
// тулбара"
//
int toolbar_ptrleave( PtWidget_t *widget,
                     ApInfo_t *apinfo,
                     PtCallbackInfo_t *cbinfo )
{
    PtArg_t    args[1] ;
    int        *number = NULL ;
    toolbar_t  *tb = NULL ;
    PhEvent_t  *event = cbinfo->event ;

    if( event->subtype == Ph_EV_PTR_LEAVE )
    {
        PtSetArg( args, Pt_ARG_USER_DATA, &number, 0 ) ;
        PtGetResources( widget, 1, args ) ;
        tb = &toolbars[*number] ;

        // Вызвать управляющий автомат
        A0( 40, tb, event ) ;
    } ;

    return( Pt_CONTINUE ) ;
} ;

//=====
// Функция, реализующая граф переходов
// алгоритма управления тулбаром
//
void A0( int e, toolbar_t *tb, PhEvent_t *event )
{

```

```

// Запомнить текущее состояние
int y_old = tb->y0 ;

#ifdef GRAPH_EVENTS_LOGGING
    log_exec( "A0", y_old, e ) ;
#endif

// Проверить условия переходов, если необходимо выполнить
// переход и действия на дуге
//
switch( tb->y0 )
{
    case 0:
        if( e == 10 )                tb->y0 = 1 ;
        break ;

    case 1:
        if( e == 20 ) { z30(tb) ; tb->y0 = 0 ; }
        else
            if( e == 30 )                tb->y0 = 2 ;
            else
                if( e == 40 )                tb->y0 = 0 ;
        break ;

    case 2:
        if( e == 30 ) { z20(tb, event) ; }
        else
            if( e != 30 )                tb->y0 = 0 ;
        break ;

    default:
        #ifdef GRAPH_ERRORS_LOGGING
            log_write( LOG_GRAPH_ERROR,
                "ОШИБКА В A0: неизвестное состояние", 0 ) ;
        #endif
        break ;
} ;

// Если состояние не изменилось -
// завершить выполнение функции
if( y_old == tb->y0 ) goto A0_end ;

#ifdef GRAPH_TRANS_LOGGING
    log_trans( "A0", y_old, tb->y0 ) ;
#endif
// Выполнить действия в состоянии
switch( tb->y0 )
{
    case 1:
        z10(tb, event) ; z40(tb) ;
        break ;

```

```

        case 2:
            z20(tb, event) ;
            break;
    } ;

A0_end:
#ifdef GRAPH_ENDS_LOGGING
    log_end( "A0", tb->y0, e ) ;
#endif
;
} ;

//=====
// Запомнить координаты курсора мыши
//
void z10( toolbar_t *tb, PhEvent_t *event )
{
    PhRect_t      *rect = NULL ;

#ifdef ACTIONS_LOGGING
    log_write( LOG_ACTION,
               "z10. Запомнить координаты курсора
               мыши", 0 ) ;
#endif

    rect = PhGetRects( event ) ;
    tb->drag_pos = rect->ul ;
} ;

//=====
// Переместить тулбар
//
void z20( toolbar_t *tb, PhEvent_t *event )
{
    PhRect_t      *rect = NULL ;

#ifdef ACTIONS_LOGGING
    log_write( LOG_ACTION,
               "z20. Переместить тулбар", 0 ) ;
#endif

    rect = PhGetRects( event ) ;
    toolbar_move( tb, rect->ul.x - tb->drag_pos.x,
                  rect->ul.y - tb->drag_pos.y ) ;
} ;

```

```

//=====
// Вызвать меню тулбара
//
void z30( toolbar_t *tb )
{
    #ifdef ACTIONS_LOGGING
        log_write( LOG_ACTION,
                    "z30. Вызвать меню тулбара", 0 ) ;
    #endif

    ApCreateModule( ABM_toolbar_menu, NULL, NULL ) ;
} ;

//=====
// Переместить окно тулбара выше остальных
//
void z40( toolbar_t *tb )
{
    #ifdef ACTIONS_LOGGING
        log_write( LOG_ACTION, "z40. Переместить окно
                    тулбара выше остальных", 0 ) ;
    #endif

    PtWindowToFront( tb->wgt ) ;
} ;

```

**«Полный» протокол проверки всех переходов автомата,
реализующего алгоритм управления тулбаром**

Обработка события «нажатие правой кнопки мыши при нахождении ее курсора на тулбаре»

```

16:44:58.543{ A0: в состоянии 0 запущен с событием e10
16:44:58.543T A0: перешел из состояния 0 в состояние 1
16:44:58.543* z10. Запомнить координаты курсора мыши
16:44:58.543* z40. Переместить окно тулбара выше остальных
16:44:58.543} A0: завершил обработку события e10 в состоянии 1

```

Обработка события «отпускание правой кнопки мыши при нахождении ее курсора на тулбаре» — вызов меню тулбара

```

16:44:59.903{ A0: в состоянии 1 запущен с событием e20
16:44:59.903* z30. Вызвать меню тулбара
16:44:59.903T A0: перешел из состояния 1 в состояние 0
16:44:59.903} A0: завершил обработку события e20 в состоянии 0

```

Обработка события «выход курсора мыши за границу тулбара»

```

16:44:59.903{ A0: в состоянии 0 запущен с событием e40
16:44:59.903} A0: завершил обработку события e40 в состоянии 0

```

Обработка события «нажатие правой кнопки мыши при нахождении ее курсора на тулбаре»

```

16:45:03.963{ A0: в состоянии 0 запущен с событием e10

```

16:45:03.963T A0: перешел из состояния 0 в состояние 1
 16:45:03.963* z10. Запомнить координаты курсора мыши
 16:45:03.963* z40. Переместить окно тулбара выше остальных
 16:45:03.963} A0: завершил обработку события e10 в состоянии 1

Обработка события «выход курсора мыши за границу тулбара»

16:45:05.933{ A0: в состоянии 1 запущен с событием e40
 16:45:05.933T A0: перешел из состояния 1 в состояние 0
 16:45:05.933} A0: завершил обработку события e40 в состоянии 0

Обработка события «нажатие правой кнопки мыши при нахождении ее курсора на тулбаре»

16:45:10.482{ A0: в состоянии 0 запущен с событием e10
 16:45:10.482T A0: перешел из состояния 0 в состояние 1
 16:45:10.482* z10. Запомнить координаты курсора мыши
 16:45:10.482* z40. Переместить окно тулбара выше остальных
 16:45:10.482} A0: завершил обработку события e10 в состоянии 1

Обработка события «перемещение мыши с нажатой правой кнопкой при нахождении ее курсора на тулбаре»

16:45:12.812{ A0: в состоянии 1 запущен с событием e30
 16:45:12.812T A0: перешел из состояния 1 в состояние 2
 16:45:12.812* z20. Переместить тулбар
 16:45:12.812} A0: завершил обработку события e30 в состоянии 2
 16:45:12.852{ A0: в состоянии 2 запущен с событием e30
 16:45:12.852* z20. Переместить тулбар
 16:45:12.852} A0: завершил обработку события e30 в состоянии 2

Обработка события «отпускание правой кнопки мыши при нахождении ее курсора на тулбаре»

16:45:15.812{ A0: в состоянии 2 запущен с событием e20
 16:45:15.812T A0: перешел из состояния 2 в состояние 0
 16:45:15.812} A0: завершил обработку события e20 в состоянии 0

Обработка события «выход курсора мыши за границу тулбара»

16:45:16.742{ A0: в состоянии 0 запущен с событием e40
 16:45:16.742} A0: завершил обработку события e40 в состоянии 0

Обработка события «нажатие правой кнопки мыши при нахождении ее курсора на тулбаре»

16:45:18.992{ A0: в состоянии 0 запущен с событием e10
 16:45:18.992T A0: перешел из состояния 0 в состояние 1
 16:45:18.992* z10. Запомнить координаты курсора мыши
 16:45:18.992* z40. Переместить окно тулбара выше остальных
 16:45:18.992} A0: завершил обработку события e10 в состоянии 1

Обработка события «перемещение мыши с нажатой правой кнопкой при нахождении ее курсора на тулбаре»

16:45:20.472{ A0: в состоянии 1 запущен с событием e30
 16:45:20.472T A0: перешел из состояния 1 в состояние 2
 16:45:20.472* z20. Переместить тулбар

16:45:20.472} A0: завершил обработку события e30 в состоянии 2
16:45:21.192{ A0: в состоянии 2 запущен с событием e30
16:45:21.192* z20. Переместить тулбар
16:45:21.192} A0: завершил обработку события e30 в состоянии 2

Обработка события «выход курсора мыши за границу тулбара» — прекращение перемещения

16:45:21.232{ A0: в состоянии 2 запущен с событием e40
16:45:21.232T A0: перешел из состояния 2 в состояние 0
16:45:21.232} A0: завершил обработку события e40 в состоянии 0

Используемые сокращения

АК	— аккумулятор
АО	— акционерное общество
АП	— арифметический полином
АСУ ТП	— автоматизированная система управления технологическими процессами
АУ	— ассоциативное устройство
АФ	— арифметическая функция
ББФ	— неповторная булева формула
БГ	— бинарный граф
БОУ	— буквенное операторное устройство
БП	— бинарная программа
БПФ	— неповторная пороговая формула
БУ	— бинарное устройство
БФ	— булева формула
БФУ	— булева функция
ВГС	— векторная граф-схема
ГДМ	— граф достижимых маркировок
ГДС	— граф достижимых состояний
ГИ	— генератор импульсов
ГП	— граф переходов
ГС	— граф-схема
ДНФ	— дизъюнктивная нормальная форма
ДШ	— дешифратор
ИП	— интерполяционный полином
ИПК СП	— Институт повышения квалификации руководящих работников и специалистов судостроительной промышленности
ИТ	— интерполяционная таблица
КНФ	— конъюнктивная нормальная форма
КС	— каноническая структура
КСх	— комбинационная схема
КТП	— кодированная таблица переходов
ЛАП	— линейный арифметический полином
ЛАПМ	— линейный арифметический полином с маскированием
ЛБГ	— линейный бинарный граф
ЛОБГ	— линейный операторно-бинарный граф
ЛСА	— логическая схема алгоритма
ЛСБГ	— линеаризованный структурированный бинарный граф
ЛСГС	— линеаризованная структурированная граф-схема
ЛУ	— логическое устройство
МЗФ	— метод зависимых фрагментов

МЗФО	— метод зависимых фрагментов с оптимизацией
МКС	— модифицированная каноническая структура
МЛМ	— многофункциональный логический модуль
МНФ	— метод независимых фрагментов
МОС	— минимизированный оператор структуры
МФ	— мультиплексорная форма
МЭ	— мажоритарный элемент
НАП	— нелинейный арифметический полином
НПО	— научно-производственное объединение
ОБГ	— операторно-бинарный граф
ОБУ	— операторно-бинарное устройство
ОВ	— операторная вершина
ОДНФ	— ортогональная дизъюнктивная нормальная форма
ОЗУ	— оперативное запоминающее устройство
ОП	— операторная программа
ОС	— оператор структуры
ОСДНФ	— обобщенная совершенная дизъюнктивная нормальная форма
ОСФ	— ортогональная скобочная форма
ОУ	— операторное устройство
ПБПФ	— положительная неповторная пороговая формула
ПДС	— параллельный двоичный счетчик
ПЖ	— полином Жегалкина
ПЗУ	— постоянное запоминающее устройство
ПЛК	— программируемый логический контроллер
ПЛМ	— программируемая логическая матрица
ПЛФ	— порогово-линейная форма
ППБУ	— плавучая полупогружная буровая установка
ППФ	— положительная пороговая функция
ПР	— пороговая реализация
ПУ	— последовательностное устройство
ПФ	— пороговая функция
ПФО	— порождающая формула
ПЭ	— пороговый элемент
РГ	— регистр
СБГ	— структурированный бинарный граф
СБИС	— сверхбольшая интегральная схема
СБФ	— система булевых формул
СБФУ	— система булевых функций
СГС	— структурированная граф-схема
СДНФ	— совершенная дизъюнктивная нормальная форма
СМ	— симметрический модуль
СОУ	— стековое операторное устройство
СПФ	— самодвойственная пороговая функция
СР	— симметрическая реализация
СУМ	— сумматор
СФ	— симметрическая функция
СЧ	— счетчик
ТГ	— тетрадный граф
ТИ	— таблица истинности
ТН	— таблица настройки
УВ	— условная вершина

УК	— управляющая конструкция
УЛМ	— универсальный логический модуль
УУ	— устройство управления
ФС	— функциональная схема
ФЗЗ	— функциональный элемент задержки
ЭСФ	— элементарная симметрическая функция
Я	— ячейка

Предметный указатель

- Автомат 726
 - без выхода 696
 - — выходного преобразователя 553, 698
 - — памяти 459
 - вложенный 728
 - вызываемый 728
 - конечный 718
 - — абстрактный 700
 - — структурный 698
 - — детерминированный 697
 - Мили 564, 698
 - Мура 559, 698
 - — второго рода 698, 749
 - — первого рода 698
 - — внутренними состояниями 698
 - — памятью 23, 253, 532, 701
 - смешанный 565, 698
 - Уилкса 598
 - управляющий 566, 708, 710
- Агент 726
- Адрес 574
- Аккумулятор 580
- Алгебра
 - булева 742
 - логики 469
- Алгоритм 546, 732
 - полиномиальный 746
 - управления 696
 - — логического 23
 - — судовыми техническими средствами 24, 27
 - Хоанг Туи 416
- Алфавит
 - входной 332
 - настроечный 332
 - настройки 261, 268, 286
- База элементная 741
- Базис 26
 - логических операций 26, 52
 - произвольный 69
 - формулы 26
- Блок 566
 - функциональный 566
- Буква 23
- Верификация
 - графа бинарного 548
 - схемы 57, 198
- Вершина 440
 - операторная 441, 506, 532
 - условная 441
- Вес переменной 37, 299, 447
- Взаимодействие автоматов 705, 731
- Вложенность 727
- Воздействие
 - входное 697
 - выходное 697
- Возможности функциональные 54, 60, 98, 175, 326, 357, 706

- Вход 291
 — боковой 429
 — информационный 59, 242, 402, 567
 — информационно-настроечный 261
 — настроечный 59, 242, 402, 429
 — парафазный 391, 398
 — управляющий 390
 Вывод
 — внешний 31, 51
 — логический 51, 312
 — промежуточный 419
 Вызываемость 727
 Выражение арифметическое 617, 619, 649
 Выход
 — инверсный 31
 — прямой 31
 вычисление последовательностное 572
 Граф
 — бинарный 440, 532, 545, 745
 — — блочный 447
 — — линейный 37, 128, 199, 440, 592
 — — планарный 440
 — — простой 498
 — — обобщенный 500
 — — структурированный 459, 527, 542
 — — линеаризованный 471
 — достижимых маркировок 703
 — — состояний 703
 — Муна—Мозера 457
 — переходов 547, 699
 — — планарный 699
 — — вызываемый 705
 — — головной 705
 — схема 532
 — — алгоритма 699
 — — векторная 516
 — — ветвящаяся 497
 — — программы 699
 — — структурированная 533
 — тетрадный 595
 — эйлеров 23, 416
 Действие 697
 Декомпозиция 71, 211, 227
 — булевых функций 67
 — дизъюнктивная 72
 — мажоритарная 94, 129
 — — простая 94
 — мультиплексорная 72, 149, 278
 — — простая 72
 — однократная 71
 — разделительная 71
 — — функциональная 71, 167, 357
 — симметрическая 357
 — — разделительная 357
 — функции 88
 — эвристическая 252
 Демультимплексор 519
 Дерево 29, 30
 — дихотомическое 27
 — неизоморфное 29
 — покрывающее 30, 287
 Дешифратор 76, 328, 575
 — событий 557, 711
 — состояний 546, 711
 Деятельность 697
 Диаграмма
 — Графсет 570, 699
 — деятельностей 718
 — переходов 699
 — состояний и переходов 699
 — Харела 715
 — SDL 717
 Дизель-генератор 569, 708, 710, 725
 Дизъюнкция 27, 393
 Длина пути 448
 Достижимость 700, 729
 Дублирование вершин 459
 Дуга 728
 Задача
 — выполнимости 746
 — логического проектирования 41
 Закон
 — Кирхгофа 447
 — сочетательный 26, 48

Запись польская обратная 590
Запрос 705

Идентификация автомата 703
Избыточность 419
Изоморфизм 701
Инвариант 29, 36
Инверсия 37, 185
Инвертор 57
Источник информации 48

Карта
— декомпозиции 81
— Карно 44, 171, 202, 493, 658
Каскад 618
— линейный 30, 36, 435
— Майтра 287, 423
— — неповторный 424
— — повторный 424
— — настраиваемый 424
— — настроенный 424
— — однородный 423
— — обобщенный 423
— Макхопадхья 422
— — настраиваемый 422
— — — однородный 423
— — обобщенный 422
— Шорта 429
Клапан 696
Класс функций 236, 726
Классификация 27
— N 28
— NPN 29, 239, 262, 403
— P 28, 296, 416
— — N 27, 232, 239, 271, 275
— SD 368
— самодвойственная 368
Клика 23
Код
— настройки 327
— неполный 555
— полный 566
Кодирование состояний 697, 726
— двоичное 560, 698
— — логарифмическое 559, 698
— многозначное 562, 698

— принудительное 553, 698
— принудительно-свободное 555, 698
— противогоночное 570
Компьютер промышленный 696
Контакт
— замкнутый 415
— переключательный 185, 415, 442, 525
— разомкнутый 415
Конструкция
— do while 711
— switch 546, 702
— управляющая 651
Контроллер
— параллельный логический 607
— программируемый 572
— — логический 22, 572, 695
Контур
— генерирующий 700
— четный 570
Конъюнкция 27, 392
Кортеж функций 669
Коэффициент
— логической эффективности 58
— многофункциональности 58
Кристалл матричный 22

Логика программируемая 22, 250, 721

Мажоранта 231
Маршрут эйлеров 416
Маска 606
Математика дискретная 23
Матрица
— Адамара 629
— булева 112
— дополнительная 113
— программируемая логическая 406, 605
— псевдоединичная 112
— транспонированная 113
— Уолша 628
Машина
— логическая 572

- Тьюринга 611, 720
- Метод
 - Акерса 166
 - Ашкрофта и Манна 711
 - Блоха 186, 745
 - взвешивания функций 511
 - гарвардский 39
 - Гуда 645
 - графовый 126, 197
 - Дейкстры 446
 - декомпозиционный 67, 108
 - зависимых фрагментов 468, 504
 - — — с оптимизацией 471
 - канонический 68, 484, 535
 - — модифицированный 68, 185, 200, 494
 - каскадов 68, 455
 - Левенгейма 69
 - логических шкал 615
 - матричный 645
 - мультиплексорный 24, 68, 128, 191, 230
 - — с настраиваемым образом декомпозиции 191, 197, 424
 - — модифицированный 280
 - направленного поиска 69
 - независимых фрагментов 474, 479, 502, 533
 - проверки 443
 - распределительный 196, 483, 535
 - реализации
 - — аппаратной 21
 - — программой 21
 - решения логических уравнений 67, 69
 - синтеза 48
 - — комбинационных схем 48
 - — Миллера 108
 - смешанный 196, 474
 - совместной реализации булевых функций 39
 - спектральный 627
 - структурирования 464
 - Флойда 446
 - формульный 48, 105, 201, 440
 - — двухэтапный 55, 232, 233
 - — в мультиплексорной форме 203
 - — таблично-аналитический 33
 - — топологический 416
- Метрика 41
- Механизм исполнительный 706
- Микроконтроллер 572, 725
- Микропроцессор 572
- Микросборка 61
- Микросхема 22
- Минимизация 440
 - длины путей 452
 - — вершин 470
 - — модулей 52
 - — путей 452
- Многополюсник
 - симметрический 326
 - Шеннона 326
- Модель 700
 - автомата структурная 700
 - алгоритмическая 704
 - математическая 719
- Модуль 22
 - виртуальный 49, 60
 - многофункциональный 260
 - универсальный 231
 - многофункциональный 49
 - — логический 58
 - — монотонный 45, 64
 - — настраиваемый 22, 49, 320
 - — на функции генерации 322
 - — — — триггеров 320
 - одновыходной 294
 - однородный 414
 - Препараты—Мюллера 242, 333
 - симметрический 326
 - с параллельно-последовательной настройкой 323
 - универсальный 49, 247, 390
 - — в классе дизъюнктивных нормальных форм 291
 - — — — пороговых функций 225, 241
 - — — — произвольных булевых функций 239, 317, 390
 - — — — самодвойственных функций 368, 378
 - — — — симметрических функций 326

- — — — — формул в базисе И, ИЛИ, НЕ 31, 295, 414
- — — — — формул в базисе И, ИЛИ, НЕРАВНОЗНАЧНОСТЬ, НЕ 314
- — немонотонный 55, 231
- WOS 217
- Мост кенигсбергский 418
- Мультиплексор 41, 74, 185, 331, 369, 443
- логический 568
- цифровой 568

- Наблюдаемость поведения** 703, 731
- Набор микросхем 62
- Настройка 56, 244
- модуля 272
- Непротиворечивость 700, 729
- Неравенство 70

- Обеспечение программное** 734
- Образ
- декомпозиции 94, 359
- — настраиваемый 107
- заданный 94
- Объединение
- фрагментов 269
- функций 267
- Объект 21, 595, 726
- автоматизации 746
- Оператор 638
- структуры 479
- — минимизированный 480
- Операция
- абсолютная величина 641
- арифметическая 23
- двухместная 26
- мультиплексирования 203
- логическая 41
- маскирования 616
- Оптрон 419
- Отношение покрытия 37, 265
- Оценка
- верхняя 37
- нижняя 62
- полиномиальная 746
- сложности 50, 53, 136, 180
- числа путей 448
- — условных вершин 489
- длины путей 448
- эффективности 58

- Параллелизм** 701
- Переключатель 702
- Переменная 27
- внутренняя 696, 702, 704
- временная 706
- входная 56
- выходная 477
- многозначная 704
- управляющая 696
- Переход 465, 478, 495, 728
- изоморфный 701
- Перечисление путей 444
- Петля 728
- Плата 61
- Поведение автомата 701
- Подсчет числа путей 446
- Подформула 50, 127, 198, 443
- Подход
- автоматный 725, 748, 754
- алгоритмический 731
- объектно-ориентированный 731
- событийный 748, 750
- Полином
- арифметический 505, 636
- — линейный 646, 650
- — — безызбыточный 663
- — — нелинейный 652
- — — с маскированием 664, 683
- — — условными маскированиями 690
- Жегалкина 44, 429, 637, 647
- интерполяционный 622
- структурный 27, 50
- Полнота 700, 729
- функциональная 429
- Полусумматор 329
- Пометка Акерса 37

- Понятность 570
- Порог функции 37, 345, 447
- Порядок лексикографический 402
- Правило эвристическое 56
- Представление спектральное 648
- Предыстория 701
 - глубокая 701
- Преобразование
 - Артюхова—Шалыто 41, 136, 151, 202, 427
 - булевых функций 286
 - Кона—Линдемана 124
 - прямое 630
 - обратное 630
 - Уолша—Адамара 630
 - Фурье 745
- Преобразователь функциональный 350, 567
- Прерывание 565, 727
- Признак булев 463, 476
- Принцип
 - двойственности 287
 - локального кодирования 326
- Проблема риска 729
- Программа 701
 - бинарная 440
 - операторная 585
 - — стековая 590
- Программирование
 - автоматное 557, 697
 - объектно-ориентированное 570
 - структурное 570
 - событийное 557
- Проектирование 695
 - автоматное 557, 697
 - логическое 23, 89
 - процедурное 711
 - объектно-ориентированное 711, 714
- Пространство состояний 703
- Протокол
 - диагностирующий 732
 - короткий 730
 - полный 730, 732
 - проверяющий 732
- Процесс 725
 - алгоритмизации 696
 - марковский 701
 - переходный 570
- логического управления 696
- технологический 23
- Процессор однобитовый 609
- Пуля серебряная 733
- Путь
 - единичный 445
 - нулевой 445
 - ортогональный 445
- Разложение 91
 - Акерса 125, 156
 - мажоритарное 134, 162, 167
 - мультиплексорное 102
 - Рида 44, 68, 86, 153, 428
 - Розенблюма 125
 - симметрическое 255
 - универсальное 86
 - Шеннона 35, 67, 84, 125, 180, 188, 444, 475
- Ранг
 - нечетный 32
 - функции 32
 - четный 33
- Распознавание автомата 703
- Расшифровка автомата 703
- Реализация
 - аппаратная 21
 - каскадная 687
 - пороговая 345, 665
 - программная 22, 532, 748
 - симметрическая 313
 - — неповторная 328
 - — повторная 328, 348
 - совместная 41
 - стандартная 340, 512
 - ярусная 687
 - ярусно-каскадная 689
- Реализуемость 700
- Регулятор 706
- Реле
 - одноконтактное 414
 - трехмембранное пневматическое 233
- Риск 729
- Ряд Фибоначчи 32

Свидетельство авторское 261, 289, 324
 Сертификация 707, 730
 Сеть Петри 698
 Сечение золотое 744, 747
 Синтез
 — логический 48
 — схем 48
 — — комбинационных 63
 — формальный 733
 Система
 — автоматизации проектирования 63
 — булевых формул 41, 43, 329
 — — функций 643
 — взаимосвязанных графов переходов 704
 — отказоустойчивая 710
 — реактивная 715, 725, 742
 — управления 725
 — — комплексная 21
 — — логического 21
 — — судовыми техническими средствами 24
 — уравнений 70
 Системонезависимость 731
 Системозависимость 731
 Событие 557, 702
 Соотношение
 — матричное 628
 — рекуррентное 29
 Соппротивление 420
 Состояние 461, 557, 696, 702
 — внутреннее 696
 Спектр функции 629
 Спецификация 709, 731
 Стандарт международный 695, 712
 Степень кронекеровская 643
 Структура
 — блочная 311
 — децентрализованная 704
 — иерархическая 704
 — каноническая 185, 484, 490
 — — модифицированная 185
 — одномерная 422, 423
 — однонаправленная 422, 423
 — однородная 422
 — — многоканальная 429
 — — плоскостная 434
 — ленточная 429
 — симметрическая 525
 — централизованная 704
 Структурирование 459
 Сумматор
 — параллельный одноразрядный двоичный 40, 43, 80, 155, 174, 329, 378, 629
 — вычитатель параллельный одноразрядный двоичный 43
 Суперпозиция 313, 618
 Схема 263
 — алгоритма 442, 547, 711
 — — автоматная 547
 — — логическая 442
 — древовидная 125, 141, 202, 422, 429
 — изоморфная 566
 — интегральная 22
 — — большая 22
 — — сверхбольшая 22, 250
 — каскадная 125, 430
 — — неповторная 125, 140
 — — второго типа 199
 — — однородная 43
 — — первого типа 192
 — — третьего типа 205
 — комбинационная 23, 48, 263, 414, 519
 — — с обратной связью 298
 — контактная 27, 417
 — — параллельно-последовательная 27, 415, 745
 — лестничная 570, 713
 — недревовидная 140, 204
 — пирамидальная 137
 — связей 727
 — стандартная 71, 74
 — структурная 726
 — функциональная 566
 — — изоморфная 566
 — эйлерова 416
 Сценарий 732
 Счетчик
 — двоичный 575
 — параллельный двоичный 328

- Таблица
 - интерполяционная 622
 - истинности 26, 36, 73
 - настройки 403
 - переходов кодированная 535, 542
 - полная 339, 344
 - табулирования 28
- Теория
 - вероятностей 33
 - графов 418
 - пороговых функций 677
- Терм 412
- Тест 445
 - полный 445
- Технология 708
 - алгоритмизации 696
 - программирования 696
 - automaton 696
 - state 696
 - switch 24, 695
- Тип функции 262
- Транзистор 415
- Транслятор 709
- Транспонирование 42
- Триггер 253
 - D 575, 599
 - JK 254
 - R 256, 533, 748
 - S 254
 - T 254
 - счетный 524, 749
- Управление логическое 21, 572
- Управляемость 698
- Уравнение 40
 - алгебраическое 70
 - логическое 70, 63, 94, 157
- Уровень интеграции 22, 60
- Условие
 - каскадности 191, 201
 - линейности 646
- Устройство
 - ассоциативное 602
 - бинарное 596
 - — двухадресное 598
 - — трехадресное 597
 - — запоминающее оперативное 573
 - — постоянное 41, 74, 263, 575, 602
 - логическое 572
 - — программное 22, 572
 - микропроцессорное 608
 - операторное 573
 - — без кода операций 584
 - — буквенное 591
 - — двухадресное 580
 - — одноадресное 581
 - — с RS-триггерами 584
 - — — микропрограммной реализацией операторов 584
 - — стековое 589
 - — трехадресное 577
 - — универсальное в классе формул 583
 - операторно-бинарное 601
 - последовательностного действия 572
 - последовательностное 572
 - управления 575
- Факториал 87
- Форма
 - двоичная 113
 - нормальная дизъюнктивная 28, 38, 293, 406
 - — ортогональная 35, 444, 451
 - — — совершенная 603, 647
 - — — обобщенная 643
 - мультиплексорная 79, 84, 100, 251
 - параметрическая 79
- Формула
 - неповторная 26, 48, 302
 - — пороговая 36, 299, 447, 464, 617
 - булева 26
 - — неповторная 27
 - знакопеременная 449
 - инвариантная 450
 - монотонная 449
 - — положительная 449

- немонотонная 57
- непороговая 127, 312, 472
- ортогональная скобочная 39
- повторная 26, 27, 334, 455
- порождающая 261
- рекуррентная 28
- скобочная 429
- Фрагмент** 101, 132
- формулы 127, 198, 443
- функции 38
- Функция**
- Адамара 631
- антидвойственная 369, 371, 386
- арифметическая 498, 509, 513
- неповторная 361
- булева 44
- — каскадная 466
- — монотонная 137
- — произвольная 106
- двойственная 346, 369, 376
- декомпозируемая 251
- дуальная 346
- зеркальная 43, 202, 373
- инверсная 370
- каскадная 467
- линейная 23, 368, 682
- линейно-представимая 680
- логическая многозначная 626
- мажоритарная 42
- монотонная 98, 214, 223, 368
- — положительная 99
- — отрицательная 100
- многозначная 628
- мультиплексирования 38, 232
- однотипная 280
- определенная не полностью 53, 138, 644
- особенная 368
- остаточная 368
- пороговая 23, 225, 617
- — положительная 345
- — — неповторная 348
- порогово-линейная 671
- порождающая 94, 97, 261
- произвольная 75, 242, 368
- Радемахера 627
- самоантидвойственная 372

- самодвойственная 23, 29, 124, 274, 347, 368, 377
- симметрическая 23, 33, 74, 326, 525, 637, 675
- — элементарная 327
- Уолша 629

Цикл программный 455

Число

- букв 26, 261
- вершин 701
- входов 261
- выходов 261
- единиц 32, 33, 89, 454
- модулей 50
- переменных 261
- путей 199, 446, 451
- рабочее 327
- решений 95
- символов 261
- Фибоначчи 23, 37, 448
- формул неповторных 29
- шагов 50

Шаблон программы 729

Шаг

- декомпозиции 206
- реализации 50

Элемент

- двухвходовой 29
- — И 296
- — И—НЕ 43, 211, 436
- — ИЛИ 296
- — ИЛИ—НЕ 43, 211, 413
- — НЕРАВНОЗНАЧНОСТЬ 423
- задержки функциональный 570, 706
- логический 58
- — произвольный 62
- мажоритарный 31, 103, 123, 310, 422
- — пятивходовой 173, 181
- — трехвходовой 123, 181
- пороговый 422

- процессорный 573
- релейно-контактный 22, 419
- с проводимостью
- — — двусторонней 260, 414
- — — односторонней 260, 414
- УСЭППА 233, 285
- Этап
- анализа 726
- документирования 730
- изучения предметной области 726
- отладки 729
- проектирования 726
- реализации 729
- сертификации 730
- Эффективность логическая 61, 313
- Язык**
- АДА 743
- алгоритмизации 696
- алгоритмический 570
- ассемблер 570
- Графсет 708
- инструкций 570
- программирования 707, 713
- СИ 610, 702
- спецификаций 707, 713
- функциональных блоков 702
- MATLAB 719
- UML 718
- Ячейка 287, 430
- одноразрядная 573
- однородной структуры 434
- Ящик
- белый 704
- — абсолютно 704
- — относительно 704
- черный 704
- — абсолютно 704
- — относительно 704