Глава 1
КОМБИНАЦИОННЫЕ ЛОГИЧЕСКИЕ ЦЕПИ

1.1. Основные положения алгебры логики

Анализ и синтез логических схем осуществляется на базе аппарата алгебры логики или булевой алгебры [9]. Излагать весь аппарат не имеет смысла, так как в инженерной практике используются два-три закона алгебры логики.

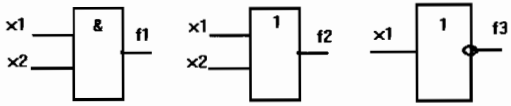
В алгебре логики переменные могут принимать только два значения: 0 или 1. Для двух аргументов существуют 16 логических функций (операций, логических действий). Над переменными в основном производятся три логических действия: сложение, умножение, отрицание (инверсия), что соответствует функциям ИЛИ, И, НЕ. Все функции в булевой алгебре могут быть описаны с помощью таблицы истинности. В нижеследующих таблицах описаны функции И(f_1), ИЛИ(f_2), НЕ(f_3).

Аргументы	Функции	
x_2 x_1	f_1	f_2
0 0	0	0
0 1	0	1
1 0	0	1
1 1	1	1

Аргумент	Функция
x	f_3
0	1
1	0

Вместо функции И часто используется термин «конъюнкция», вместо функции ИЛИ – термин «дизъюнкция». По ЕСКД логические

элементы, реализующие функции И(f_1), ИЛИ(f_2), НЕ(f_3), изображаются так, как представлено на рисунке.



При написании логических формул для функции И используются следующие знаки: &, $\wedge$, точка или ее отсутствие; для функции ИЛИ: $\vee$,+. Функция НЕ обозначается штрихом над аргументом. Мы для обозначения отрицания будем использовать апостроф. Таким образом, можно записать:

$f_1 = x_2 \& x_1 = x_2 \wedge x_1 = x_2 x_1$;

$f_2 = x_2 \vee x_1 = x_2 + x_1$;

$f_3 = x'$

Основные законы алгебры Буля

Как уже отмечалось, в булевой алгебре все операции осуществляются с логическими переменными и подчиняются законам алгебры логики. Опишем некоторые из них.

а) Переместительный закон

$a + в = в + a$; $ав = ва$

б) Сочетательный закон

$(a + в) + c = a + (в + c)$; $(ав)c = a(вс)$

в) Распределительный закон

$a(в + c) = ав + ac$; $a + вс = (a + в)(a + c)$

г) Закон поглощения

$a + ав = a(1 + в) = a$; $a(a + в) = a + ав = a$

д) Закон склеивания

$ав + ав' = a$; $(a + в)(a + в') = a$

е) Идемпотентный закон

$a + a = a$; $a \& a = a$

ё) Правила Де Моргана

Эти правила справедливы для любого числа аргументов.

$\overline{a + в + c + ... + z} = (a'в'c'...z')'$

$\overline{авс...} = (a' + в' + c' + ... + z')'$

Эти правила можно описать таким алгоритмом.

Для перехода от логической суммы к логическому произведению необходимо проделать следующие операции:

1) проинвертировать все слагаемые в отдельности;

2) заменить знаки дизъюнкции на знаки конъюнкции;

3) проинвертировать получившееся выражение.

Аналогично выполняется переход от логического произведения к логической сумме. В инженерной практике используются лишь правила Де Моргана и закон склеивания (в виде карт Карно).

Кроме основных функций И, ИЛИ, НЕ в алгебре логики часто используются функции равнозначности (эквивалентности) и неравнозначности (сумма по модулю 2).

Для обозначения этих функций используются следующие знаки: равнозначность - ~, сумма по модулю 2 - $\oplus$. Содержание этих функций отражено в таблице.

а в	f_4	f_5
0 0	1	0
0 1	0	1
1 0	0	1
1 1	1	0

Из таблицы получаем:

$f_4 = a \sim в = a'в' + ав$;

$f_5 = a \oplus в = a'в + ав'$

Из таблицы видно, что

$f_4 = f_5'$ или $f_5 = f_4'$

Таким образом,

$a'в' + ав = (ав' + a'в)'$, или $a \sim в = (a \oplus в)'$, $a \oplus в = (a \sim в)'$

1.2. Разновидности логических интегральных схем (ИС)

Логические ИС, выпускаемые промышленностью, являются функционально полными системами логических элементов и образуют базис построения логических схем.

Функционально полная система логических элементов - это такой набор элементов, используя который, можно реализовать любую, сколь угодно сложную логическую функцию. К числу функционально полных систем относятся, например, системы, реализованные на элементах «И-НЕ» либо на элементах «ИЛИ-НЕ».

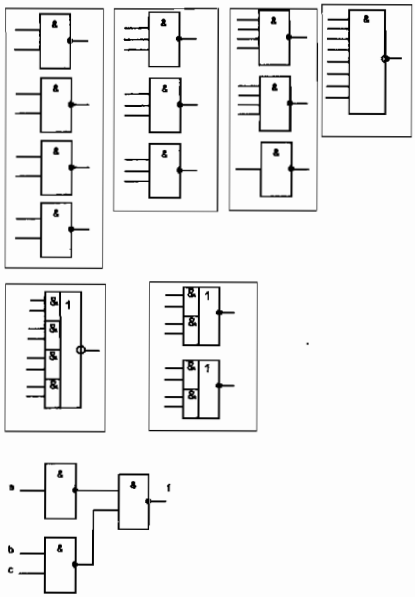
Наиболее распространены в настоящее время серии 1530, 1533, 555, 533, 561, 1561, 564, 1564, 176. Различаются эти серии по

быстродействию. Наиболее быстродействующей серией является серия 1530, самой инерционной – серия 176. Задержки в элементах серии 1530 порядка 3 нс, задержки в таких же элементах серии 176 – порядка 200нс. Максимальным потреблением обладает самая быстродействующая серия 1530, минимальным – серия 1564.

В состав перечисленных серий входят, например, ИС, представленные на рисунке. На этом рисунке контурными линиями ограничен состав одного корпуса ИС.

Пусть необходимо реализовать функцию f = a + вс в базисе И-НЕ.

Используя правило Де Моргана, получим: f = (a'(вс)')' .Её реализация представлена на рисунке.



1.3. Синтез комбинационных схем

Синтез комбинационных схем можно проиллюстрировать решением простой задачи.

Задача 1

Приёмная комиссия в составе трех членов комиссии и одного председателя решает судьбу абитуриента большинством голосов. В случае равного распределения голосов большинство определяется той группой, в которой оказался председатель приемной комиссии. Построить автомат, обеспечивающий определение большинства голосов.

Решение

Пусть f – функция большинства голосов. f = 1, если большинство членов комиссии проголосовало за приём абитуриента, и f = 0 в противном случае.

Обозначим через x_4 голос председателя комиссии. x_4 = 1, если председатель комиссии проголосовал за приём абитуриента. x_3, x_2, x_1 – голоса членов приёмной комиссии.

С учётом вышеуказанных допущений условие задачи можно однозначно представить в виде таблицы истинности.

Заполнение таблицы осуществляем с учётом того, что функция f является полностью определённой, т.е. она определена на всех возможных наборах переменных x_1 – x_4. Для n входных переменных существует $N = 2^n$ наборов переменных. В нашем примере $N = 2^4 = 16$ наборов. Записывать эти наборы можно в любом порядке, но лучше в порядке возрастания двоичного кода.

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

1 1 1 0	1
1 1 1 1	1

Примечание. Здесь и далее под набором будем понимать конъюнкцию всех входных переменных. Существует множество научных определений для набора (конституента, терм, импликанта, минтерм и т.д.), но они только вносят путаницу.

Все наборы, на которых функция принимает значение 1, будем называть единичными, или рабочими. Наборы, на которых функция принимает значение 0, будем называть нулевыми, или запрещенными.

Для того чтобы по таблице истинности найти функцию f, достаточно выписать все единичные наборы и соединить их знаком дизъюнкции.

Таким образом:

$$f = x_4'x_3x_2x_1 + x_4x_3'x_2'x_1 + x_4x_3'x_2x_1' + x_4x_3'x_2x_1 + x_4x_3x_2'x_1' + x_4x_3x_2'x_1 + x_4x_3x_2x_1' + x_4x_3x_2x_1.$$

Полученная форма функции называется совершенной дизъюнктивной нормальной формой (СДНФ), так как каждое логическое слагаемое представляет собой конъюнкцию всех аргументов.

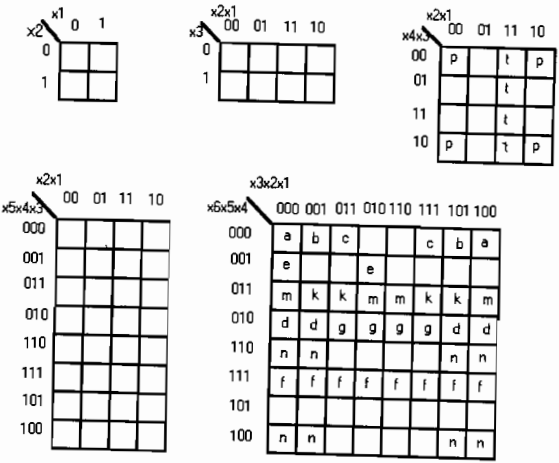
Очевидно, применяя основные законы булевой алгебры, мы могли бы аналитически уменьшить сложность полученного выражения. Но это наихудший способ минимизации булевых функций.

1.4. Минимизация полностью определённых булевых функций

Существует несколько способов минимизации булевых функций. Прежде всего, это метод Квайна-Мак-Класки, метод Блека-Порецкого и метод минимизации с помощью карт Карно или диаграмм Вейча [9,17]. Здесь будет подробно излагаться метод карт Карно, как самый удобный метод, позволяющий быстро решать задачи минимизации булевых функций от достаточно большого числа аргументов (6-12). При этом получается минимальная форма в бвзисе И, ИЛИ, НЕ.

Карты Карно и диаграммы Вейча хорошо описаны в [9], но лишь для небольшого числа переменных.

Существуют карты Карно на 2, 3, 4, 5 и 6 переменных. Причём последние стали использоваться достаточно недавно. На рисунке представлены карты Карно для 2, 3, 4, 5 и 6 аргументов.



Карты и прямоугольники Карно.

Метод Карно основан на законе склеивания. Склеиваются наборы, отличающиеся друг от друга значением одного разряда. Такие наборы называются соседними. Карно закодировал клетки своей карты так, что в соседних клетках оказались соседние, а значит, склеивающиеся наборы. Соседними могут быть не только отдельные клетки, которые мы назовем элементарными квадратами Карно, но и целые группы соседних клеток (назовем их прямоугольниками Карно). Под прямоугольником Карно будем понимать некоторую, зачастую разрозненную фигуру покрытия, все соседние клетки которой закодированы соседними наборами. Например, на вышеприведённом рисунке в поле карты для 4-х переменных изображён прямоугольник Карно P, состоящий из четырёх элементарных квадратов Карно, описываемых наборами $x_4'x_3'x_2'x_1'$, $x_4'x_3'x_2x_1'$, $x_4x_3'x_2'x_1'$, $x_4x_3'x_2x_1'$. Если над логической суммой этих четырёх наборов произвести последовательно операции склеивания, то мы аналитически получим результат в виде импликанты (под импликантой будем понимать неполный набор) $x_3'x_1'$. Карта

Карно позволяет получить этот результат графически значительно быстрее и проще. Для решения этой задачи используем алгоритм графической минимизации. Кстати говоря, сам Карно никакого алгоритма не предложил.

Алгоритм графической минимизации логических функций

1. Заполнить карту Карно нулями и единицами в соответствии с таблицей истинности.

2. Покрыть все единичные наборы минимальным количеством прямоугольников Карно, каждый из которых имеет максимальную площадь.

3. Каждому прямоугольнику Карно соответствует одна импликанта, причём, если в границах прямоугольника Карно какая-либо переменная принимает значения как 0, так и 1, то она склеивается.

Примечание. Если в карте Карно нулей окажется меньше, чем единиц, то удобнее прямоугольниками Карно покрыть все нулевые наборы. В результате мы получим инверсию минимизируемой функции.

Сущность алгоритма достаточно прозрачна. Стремление к минимальному количеству прямоугольников Карно приводит в результате к минимальному количеству слагаемых в булевой функции. Требование получения максимальной площади прямоугольника Карно вызвано стремлением минимизировать длину каждого слагаемого булевой функции.

Пусть булева функция Y задана так, что в поле прямоугольников Карно P и T вышеприведённого рисунка оказались все единичные наборы, а в остальных клетках карты Карно разместились все нулевые наборы данной функции. В соответствии с пунктом 3 алгоритма Карно прямоугольник P будет представлен импликантой $x_3'x_1'$, а прямоугольник T – импликантой x_2x_1. Таким образом, функция $Y=x_3'x_1' + x_2x_1$.

1.5. Карты Карно для 7, 8, 9 и 10 переменных

До сих пор сохранилось мнение, что карты Карно для 7-10 переменных являются труднообозримыми, поэтому ни в какой литературе, кроме [16,17], нельзя найти не только описания метода работы с картами Карно на большое количество переменных, но и самих карт. Этим же обстоятельством объясняется тот факт, что до недавнего времени в литературе редко встречались карты Карно даже для 6 переменных. Прежде, чем приступить к рассмотрению многоаргументных карт Карно, покажем на простых примерах, как осуществляется соседнее кодирование для произвольного числа

переменных. Для одной переменной существует только соседнее кодирование, так как она кодируется нулём и единицей. Чтобы перейти к соседнему кодированию для двух переменных x_2 и x_1 предлагается следующая операция. Напишем в один столбец коды для x_1 .Между нулём и единицей для столбца x_1 проведём ось, которую назовём осью симметрии 1-го рангв.

x_1
0

1

Проведём под этим столбцом ось симметрии, которую в дальнейшем будем называть осью симметрии 2-го ранга, и продолжим столбец кодов для x_1 симметрично относительно этой оси (симметрично относительно оси симметрии 2-го ранга разместятся и оси симметрии 1-го ранга).

x_1
0

1

1

0

Дополним одноразрядный код до двухразрядного, для чего выше оси симметрии впишем для x_2 нули, а ниже - единицы.

$x_2 x_1$
0 0
---
0 1
-----
1 1
---
1 0

Таким образом, мы осуществили соседнее кодирование для двух переменных. Чтобы построить соседние коды для трёх переменных, проведём под столбцами двухразрядных кодов ось симметрии 3-го ранга и продолжим эти столбцы вниз симметрично относительно оси 3-го ранга, т.е. осуществим симметричное отображение относительно оси 3-го ранга.

x_2	x_1
0	0
0	1
1	1
1	0
1	0
1	1
0	1
0	0

Дополним двухразрядные коды до трёхразорядных, вписав в третьем разряде нули выше оси Карно 3-го ранга и единицы ниже этой оси. Получим соседнее кодирование для трёх переменных.

x_3	x_2	x_1
0	0	0
0	0	1
0	1	1
0	1	0
1	1	0
1	1	1
1	0	1
1	0	0

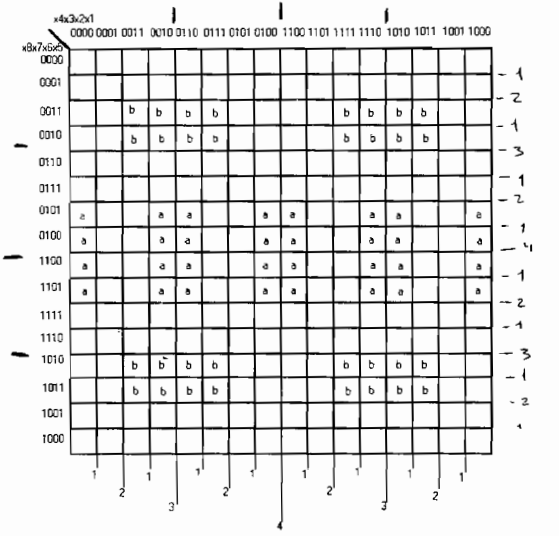
Следовательно, для того, чтобы осуществить соседнее кодирование для (P+1) переменных, если известно соседнее кодирование для P переменных, необходимо выполнить следующий алгоритм:

1) под столбцом известного P-разрядного соседнего кодирования провести ось симметрии (P+1)-го ранга ,

2) осуществить симметричное отображение относительно оси симметрии (P+1) - ранга всех P-разрядных кодов и осей симметрии всех рангов до ранга P включительно ;

3) дополнить P-разрядные коды слева одним разрядом, в котором записать 0 для всех кодов выше оси симметрии (P+1)-го ранга и 1 для кодов, расположенных ниже оси симметрии (P+1)-го ранга.

Соседнее кодирование карт Карно по вышеизложенному алгоритму производится как для вертикальных, так и для горизонтальных сторон карт. Примеры кодирования карт Карно приведены на рисунке. На нём стрелками обозначены оси симметрии, ранг которых отмечен цифрами, стоящими рядом со стрелками.



Карта Карно на 8 переменных.

Покрытие всех единичных наборов булевой функции, размещённых в карте Карно, прямоугольниками Карно не вызывает затруднений, если функция зависит не более, чем от 6 переменных. Обозримость карт Карно для большего числа переменных усложняется, так как становится трудно определить, соответствует ли данная фигура покрытия понятию прямоугольника Карно. Определение достоверности прямоугольника Карно основано на принципе симметрии, значительно повышающем обозримость карт Карно, и осуществляется с помощью приводимого ниже алгоритма.

Алгоритм проверки достоверности прямоугольника Карно (принцип симметрии)

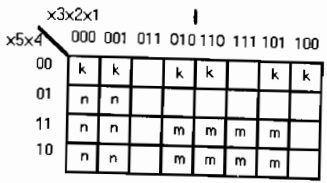
1. Если предполагаемый прямоугольник Карно (ППК) охватывает одну ось симметрии, либо не охватывает ни одной, то перейти к п.4.
2. Если ППК располагается по обе стороны от нескольких осей симметрии, то он должен быть симметричен относительно той из этих осей, которая имеет максимальный ранг, иначе данная фигура не является прямоугольником Карно.
3. Разбить исходный ППК пополам. Считать любую его половину новым ППК. Перейти к п.1.
4. Конец.

Этот алгоритм необходимо применить дважды: относительно горизонтальных и относительно вертикальных осей симметрии.

Проверим достоверность прямоугольника Карно А на вышеприведённом рисунке. Прямоугольник А размещается по обе стороны от горизонтальной оси 4-го ранга. Симметричность его очевидна. Верхняя половина фигуры А симметрична относительно горизонтальной оси симметрии 1-го ранга. Так как ППК охватывает одну единственную ось симметрии, то проверка фигуры покрытия А на соответствие принципу симметрии относительно горизонтальных осей закончена.

Приступаем к проверке принципа симметрии относительно вертикальных осей симметрии. Фигура покрытия А размещается по обе стороны от вертикальной оси симметрии 4-го ранга и симметрична относительно этой оси. Левая половина фигуры А симметрична относительно оси симметрии 3-го ранга. После повторного деления левая половина фигуры покрытия оказалась симметричной относительно оси симметрии 2-го ранга. После 3-го деления ППК не охватывает ни одной оси симметрии, на этом проверка достоверности прямоугольника Карно заканчивается. Таким образом, фигура покрытия А действительно является прямоугольником Карно. Аналогично доказывается, что фигура покрытия В также является прямоугольником Карно.

На рисунке даны примеры фигур, не являющихся прямоугольниками Карно. Фигуры k, m и n не являются прямоугольниками Карно в силу нарушения принципа симметрии. Фигура n не симметрична относительно горизонтальной оси симметрии 2-го ранга, фигура m не симметрична относительно вертикальной оси симметрии 3-го ранга. Фигура k симметрична относительно оси симметрии 3-го ранга, но её половина не симметрична относительно оси 2-го ранга.



Алгоритм «НИИРТА» графической минимизации булевых функций

1. Заполнить карту Карно нулями и единицами в соответствии с таблицей истинности или заданным алгебраическим выражением.
2. Покрыть все элементарные квадраты Карно, в которых записаны единицы, минимальным количеством фигур покрытия, каждая из которых имеет максимальную площадь.
3. Проверить каждую фигуру покрытия на соответствие принципу симметрии. В противном случае изменить контур фигуры покрытия в соответствии с принципом симметрии так, чтобы она превратилась в прямоугольник Карно.
4. Каждому прямоугольнику Карно соответствует одна импликанта, причём если в границах прямоугольника Карно какая-либо переменная принимает значения как 0, так и 1, то эта переменная не войдёт в импликанту.

Применим карту Карно для решения задачи 1. На рисунке даны два варианта решения.

$$f = x_4x_1 + x_4x_2 + x_4x_3 + x_3x_2x_1;$$
$$f' = x_4'x_1' + x_4'x_2' + x_4'x_3' + x_3'x_2'x_1'$$

Эти выражения представляют собой пример дизъюнктивной нормальной формы (ДНФ).

В некоторых случаях приведение результата минимизации к скобочной форме позволяет уменьшить количество ИС, необходимые для реализации булевой функции. Скобочная форма для f имеет вид:

$$f = x_4(x_1 + x_2 + x_3) + x_3x_2x_1$$

x4x3 \ x2x1	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	1	1	1	1
10	0	1	1	1

Карта Карно для решения задачи 1.

1.6. Оценка сложности реализации булевых функций

Приблизительную оценку реализации логической функции можно дать по ДНФ, подсчитав коэффициент сложности K_c, равный общему количеству переменных, входящих в ДНФ, плюс количество импликант. Например, для СДНФ к задаче 1 $K_c = 32+8=40$, а для отминимизированной функции $K_c = 9+4=13$.

При реализации в конкретном элементном базисе обе функции примут вид, представленный на рисунке. Из рисунка видно, что реализация функции по СДНФ потребовала 5 корпусов ИС, по минимальной форме – 1,58 корпуса ИС, по скобочной форме – 1,16 корпуса. Таким образом, минимизация по карте Карно дала нам трёхкратный выигрыш по корпусам ИС относительно реализации по СДНФ. Реализация по скобочной форме уменьшила объём оборудования ещё на 30%.

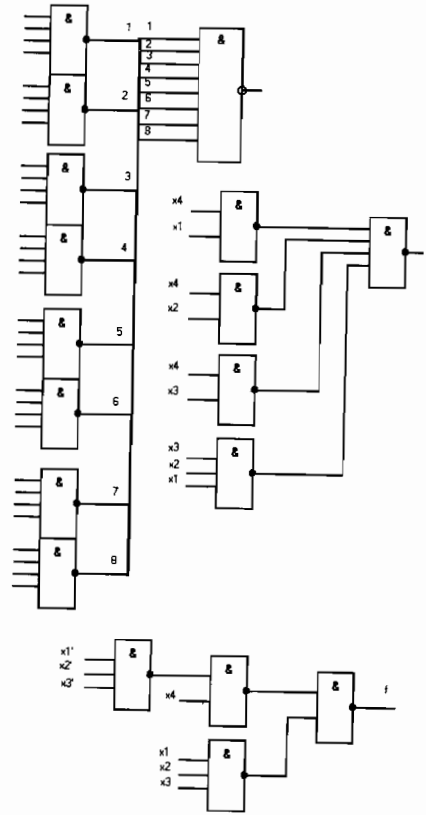


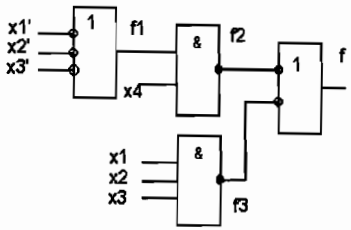
Схема автомата до и после минимизации.

1.7. Анализ комбинационных схем

В процессе работы с цифровыми схемами иногда возникает задача определения функции, которую реализует данная структура.

Задача 2

На рисунке представлена принципиальная схема комбинационного автомата. Определить его функцию.



Решение

Определим вначале промежуточную функцию

$f_1 = x_1'' + x_2'' + x_3'' = x_1 + x_2 + x_3$

Затем f_2 и f_3

$f_2 = (f_1x_4)' = ((x_1 + x_2 + x_3)x_4)'$;

$f_3 = (x_1x_2x_3)'$

Функция $f = f_2' + f_3' = (x_1 + x_2 + x_3)x_4 + x_1x_2x_3$, т.е. схема реализует скобочную форму автомата, определяющего большинство голосов.

1.8. Формы задания булевых функций

Об одной форме задания булевых функций мы уже говорили – это таблица истинности. Иногда применяется более компактная запись, использующая восьмеричные, десятичные или шестнадцатеричные эквиваленты наборов. Например, набор $x_4x_3x_2'x_1'$ может быть представлен обобщённым кодом 1100, десятичным эквивалентом которого является число 12. Удобнее всего восьмеричные и шестнадцатеричные коды.

Задача 3

Полностью определённая булева функция от 4-х переменных задана десятичными рабочими наборами: РН(4) = 5, 6, 7, 8, 9, 10, 11. Число в скобках указывает количество переменных. Найти минимальную форму этой функции.

Решение

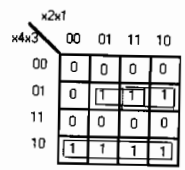
Так как функция является полностью определённой, то запрещёнными наборами ЗН(4) являются наборы 0 - 4, 12 - 15. Исходя из этой информации, составляем таблицу истинности и осуществляем минимизацию по карте Карно.

РН(4)	x_4 x_3 x_2 x_1	f
5	0 1 0 1	1
6	0 1 1 0	1
7	0 1 1 1	1
8	1 0 0 0	1
9	1 0 0 1	1
10	1 0 1 0	1
11	1 0 1 1	1

ЗН(4)	x_4 x_3 x_2 x_1	f
0	0 0 0 0	0
1	0 0 0 1	0
2	0 0 1 0	0
3	0 0 1 1	0
4	0 1 0 0	0
12	1 1 0 0	0
13	1 1 0 1	0
14	1 1 1 0	0
15	1 1 1 1	0

По карте Карно получаем результат:

$f = x_4x_3' + x_4'x_3(x_1 + x_2)$



Задание 1

Найти минимальную форму полностью определённых булевых функций, заданных десятичными рабочим наборами:

1-1) РН(4) = 0, 1, 5, 7 – 9, 13, 15

1-2) РН(5) = 4, 6, 8, 10, 13, 17, 24, 30

1-3) РН(6) = 1 – 8, 16 – 24, 32 – 40

1-4) РН(7) = 7 – 15, 23 – 31, 39 – 47, 50 – 63

1-5) РН(8) = 7 – 15, 100 – 132

1.9. Минимизация недоопределённых булевых функций

Функция от n переменных называется недоопределённой, если она задана не на всех 2^n наборах. Задача минимизации такой функции заключается в оптимальном доопределении, которое позволило бы покрыть рабочие наборы минимальным количеством прямоугольников Карно, каждый из которых имел бы максимальную площадь.

Задача 4

Найти минимальную форму функции y, представленной на рисунке.

Решение

Функция задана только на 5 наборах. Добавим к трём рабочим наборам ещё пять, а именно: 0000, 0011, 1000, 1011, 1010. Все оставшиеся наборы доопределим как запрещённые. В результате такого доопределения получим прямоугольник Карно, состоящий из 8 элементарных квадратов Карно. Этому прямоугольнику соответствует функция:

$$y = b'$$

x4x3 \ x2x1	00	01	11	10
00	1	-	-	1
01	-	-	0	-
11	-	0	-	-
10	1	-	-	1

Решение задачи 4.

В этом разделе изложен общепринятый подход к минимизации недоопределённых логических функций (НОЛФ). С точки зрения самодиагностики нужно учитывать входные наборы, на которых функция не определена. Дело в том, что зачастую вышеназванные наборы не должны появляться в нормально работающем устройстве. Поэтому необходимо фиксировать появление таких наборов с целью выработки контрольных сигналов, несущих информацию о сбое или отказе.

1.10. Минимизация системы булевых функций

Существуют достаточно сложные методы минимизации системы булевых функций. Однако все эти методы не дают оптимального решения, поэтому при инженерном синтезе комбинационных схем осуществляется раздельная минимизация функций, которая тоже не всегда обеспечивает минимальное решение, но подкупает простотой.

Задача 5

Построить преобразователь двоичного кода, получаемого на выходе делителя частоты на 12, в двоично-десятичный код. Условие задачи отражено в таблице. Делитель работает в коде 8-4-2-1.

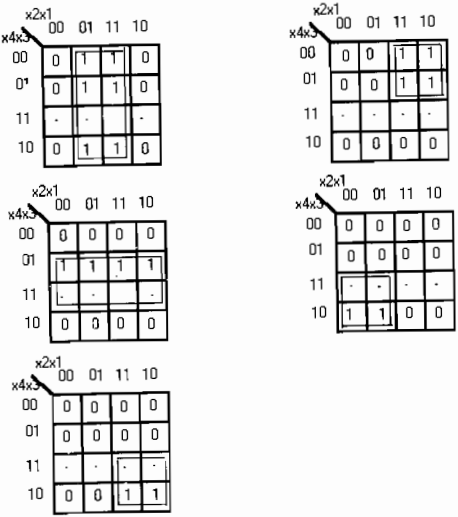
x_4 x_3 x_2 x_1	y_5	y_4 y_3 y_2 y_1
0 0 0 0	0	0 0 0 0
0 0 0 1	0	0 0 0 1
0 0 1 0	0	0 0 1 0
0 0 1 1	0	0 0 1 1
0 1 0 0	0	0 1 0 0
0 1 0 1	0	0 1 0 1
0 1 1 0	0	0 1 1 0
0 1 1 1	0	0 1 1 1
1 0 0 0	0	1 0 0 0
1 0 0 1	0	1 0 0 1
1 0 1 0	1	0 0 0 0
1 0 1 1	1	0 0 0 1

Решение

Для каждой функции y_i заполняем карту Карно, производим доопределение и осуществляем минимизацию. Весь процесс отражён на рисунке.

В результате минимизации получаем систему функций:

$$y_1 = x_1$$
$$y_2 = x_4'x_2$$
$$y_3 = x_3$$
$$y_4 = x_4x_2'$$
$$y_5 = x_4x_2$$



Карты Карно к задаче 5.

Задача 6

Построить один разряд многоразрядного сумматора.

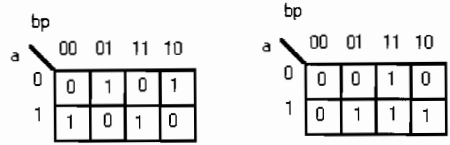
Решение

Пусть a_i и $в_i$ – значения i-ых разрядов слагаемых а и в, P_i и S_i – значения переноса и суммы на выходе i-го разряда, P_{i-1} – значение переноса на выходе предыдущего разряда, тогда работу сумматора можно описать с помощью таблицы истинности.

a_i $в_i$ P_{i-1}	P_i	S_i
0 0 0	0	0
0 0 1	0	1
0 1 0	0	1
0 1 1	1	0
1 0 0	0	1
1 0 1	1	0
1 1 0	1	0
1 1 1	1	1

Имеем систему полностью определённых булевых функций. Производим раздельную минимизацию (см. рисунок).

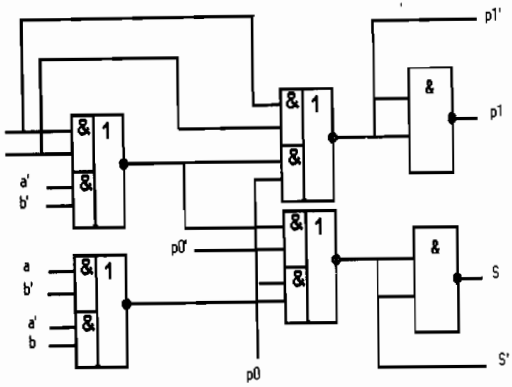
$$S_i = a_i'в_i'P_{i-1} + a_i'в_iP_{i-1}' + a_iв_i'P_{i-1}' + a_iв_iP_{i-1} = P_{i-1}(a_i \sim в_i) + P_{i-1}'(a_i \oplus в_i)$$
$$P_i = в_iP_{i-1} + a_iP_{i-1} + a_iв_i$$

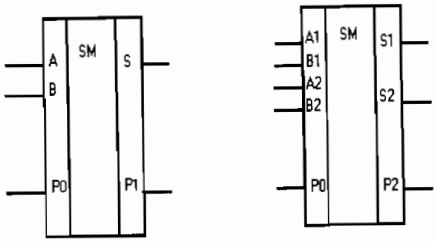


Решение задачи 6

Для реализации лучше $P_i = a_iв_i + P_{i-1}(a_i \sim в_i)'$, так как может быть использован общий для S_i и P_i сомножитель $(a_i \sim в_i)'$ Схема сумматора представлена на рисунке. Здесь же дано условное обозначение одноразрядного сумматора, где А и В – одноразрядные слагаемые, P_0 и P_1 – входной и выходной переносы, S1 – сумма.

На этом же рисунке изображён двухразрядный сумматор, выполненный на микросхеме 133ИМ2. Здесь А1, В1, А2, В2 – соответственно значения первых и вторых разрядов слагаемых А и В; S1 и S2 – 1-й и 2-й разряды суммы; P_0 – входной перенос для первого разряда, P_2' – выходной перенос.





Схемы сумматоров.

Задание 2

2-1. Построить 2/(2-10) првобразователь для делителя частоты на 24, работающвго в коде 16-8-4-2-1.

2-2. Построить 4-входовой сумматор для суммирования одноразрядных двоичных чисел.

1.11. Синтез комбинационных схем на мультиплексорах и ПЛИС

Мультиплексором называется такая комбинационная схема (КС), которая реализует функцию:

$Y = \cup X_i A_i$, где

X_i – i-й входной сигнал,

A_i – i-й адресный код.

Мультиплексор коммутирует вход X_i на выход Y, если на адресном входе установлен код A_i. Если адресная шина мультиплексора является n - разрядной, то синтез КС по таблице истинности от (n+1) переменных не требует дополнительных элементов. При синтезе автомата для тайного голосования (задача 1 из раздела 1.3) были получены следующие функции адресных входов мультиплексора:

A0 = 0;

A1 = A2 = A3 = A4 = A5 = A6 = X4;

A7 = 1.

Удобнее и проще синтезировать КС на базе ПЛИС (программируемых интегральных схем). К их числу относятся программируемые логические матрицы (ПЛМ), программируемая матричная логика (ПМЛ), матрицы логических ячеек (МЛЯ) и перепрограммируемые постоянные запоминающие устройства (ППЗУ). Наиболее популярные ПЛИС:

ПЛМ – К556РТ2

ПМЛ – К15556ХП4

МЛЯ – БИС фирм Xilinx, Altera

ППЗУ – К155РЕ3, К556РТ4, К573РФ5, К573РФ8.

Для ПЛМ, ПМЛ и МЛЯ требуется оязательная минимизация, для ППЗУ необходимо приведение функции к СДНФ.

Глава 2
МИНИМИЗАЦИЯ БУЛЕВЫХ ФУНКЦИЙ МЕТОДОМ ОБОБЩЁННЫХ КОДОВ

В СДНФ функции от n переменных каждый набор x_i можно заменить последовательностью нулей и единиц. Такая последовательность носит название обобщённого кода.

Метод обобщённых кодов был разработан в конце 60-х годов на 21-й кафедре Академии им. Дзержинского д. т. н. Леонидом Трофимовичем Лавренковым. Дальнейшее развитие метода и доведение его до инженерных методик было выполнено сотрудниками этой кафедры к.т.н. А.С.Кустенко, к.т.н. Н.В.Кузнецовым и к.т.н. Ю.А.Салтыковым (см. "Вопросы оборонной техники", 1972 г.).

Например, набору $x_4x_3'x_2x_1'$ соответствует обобщённый код 1010. Для ДНФ обобщённый код (ОК) имеет прочерки на местах отсутствующих переменных. Например, для функции от 5 переменных набору x_5x_2' соответствует ОК = 1--0-.

Методом обобщённых кодов удобно работать с функциями, заданными таблицами истинности. Причём рабочим наборам соответствуют рабочие обобщённые коды (РОК), запрещённым наборам - запрещённые обобщённые коды (ЗОК).

Введём понятие оценочной функции. Оценочная функция $F_0^p(F_1^p)$ определяет количество нулей (единиц) для одного разряда всех РОК. Оценочная функция $F_0^з(F_1^з)$ определяет количество нулей (единиц) для одного разряда всех ЗОК.

Функция вида $F_0 = F_0^p + F_1^з$ называется нулевой оценочной функцией.

Функция вида $F_1 = F_1^p + F_0^з$ называется единичной оценочной функцией.

В результате минимизации булевой функции получается минимальная ДНФ (МДНФ), состоящая из простых импликант, т.е. таких импликант, дальнейшая минимизация которых не возможна. В методе обобщённых кодов простой импликанте соответствует минимальный обобщённый код (МОК). Будем говорить, что данный МОК покрывает определённый РОК, если нули и единицы в этом РОК стоят в тех же разрядах, что и в данном МОК.

Сущность метода ОК заключается в том, чтобы по максимуму оценочных функций выбрать такие переменные, которые чаще всего встречаются в РОК и реже всего в ЗОК, и на их основе построить такую совокупность минимальных обобщённых кодов, которая покрывала бы все РОК и не покрывала бы ни одного ЗОК.

2.1. Общий алгоритм определения МОК

Алгоритм 1

1. Присвоить индексу МОК значение 1, т.е. i:=1.

2. Подсчитать по таблице истинности F_0 и F_1 для всех разрядов РОК и ЗОК.

3. В качестве базы $МОК_i$ ($БМОК_i$) или дополнение к $БМОК_i$ выбрать переменную с максимальной F_0 или F_1 .Если F_0 = max, то переменная входит в $БМОК_i$ нулём. Если F_1=max, то переменная входит в $БМОК_i$ единицей. Если несколько переменных имеют максимальные оценочные функции, то выбрать в качестве БМОК ту переменную, у которой соответствующая запрещённая оценочная функция ($F_0^з$, $F_1^з$) имеет минимальное значение; в противном случае в качестве БМОК взять любую переменную с максимальной оценочной функцией.

4. Выписать все РОК и ЗОК, покрываемые базой $МОК_i$. Если $БМОК_i$ не покрывает ни одного РОК или покрывает все ЗОК, то приравнять нулю оценочные функции F_0 и F_1 для данного разряда и перейти к п.3. Если покрываемых ЗОК нет, то перейти к п.6.

5. Подсчитать F_0 и F_1 для всех разрядов РОК и ЗОК, покрытых данной базой, кроме тех разрядов, которые образуют $БМОК_i$. Присоединить к $БМОК_i$ пвременную (дополнение к $БМОК_i$) с максимальной оценочной функцией в соответствии с требованиями п.3. Считать этот ОК новой базой $МОК_i$. Если новая $БМОК_i$ покрывает столько же ЗОК, сколько и на предыдущем шаге, то приравнять нулю оценочные функции для дополнения к $БМОК_i$, отбросить присоединённую переменную и добавить к $БМОК_i$ переменную с максимальной оценочной функцией в соответствии с требованиями п.3, считать полученный ОК новой $БМОК_i$. Если $БМОК_i$ покрывает хотя бы один ЗОК, перейти к п.4.

6. Принять в качестве $МОК_i$ базу $МОК_i$.

7. Если всв РОК из исходной таблицы истинности покрыты минимальными обобщёнными кодами, перейти к п.9.

8. Выписать из исходной таблицы истинности все ЗОК и те РОК, которые не покрыты минимальными обобщёнными кодами. Считать вновь полученную таблицу исходной таблицей истинности. Увеличить индекс МОК на единицу, т.е. i :=i+1. Перейти к п.2.

9. Конец.

Поясним положение пп. 4 и 5 алгоритма 1. Пусть таблица истинности состоит из одного РОК 1110 и трёх ЗОК: 1010, 0110 и 1111.

1110	РОК
1010	
0110	ЗОК
1111	
2232	F_0
2212	F_1

После подсчёта оценочных функций оказалось, что для второго разряда F0 = 3 = max. Если в соответствии с максимумом оценочной функции взять в качестве БМОК код –0- , то этот код не покроет ни одного РОК, что недопустимо, т.к. БМОК обязательно должна покрыть хотя бы один РОК.

Несколько иная ситуация складывается в том случае, когда БМОК, найденная по максимуму оценочной функции, покрывает часть РОК и все ЗОК. Пусть функция задана пятью РОК и одним ЗОК.

1110	
1000	
1011	РОК
0111	
0010	
1010	ЗОК
3323	F_0
3343	F_1

Если в соответствии с максимумом взять в качестве БМОК код –1-, то, в конце концов, мы построим некоторый ОК, не покрывающий ни одного ЗОК, но длина этого ОК не будет минимальной. Проиллюстрируем выполнение алгоритма 1 примерами.

Задача 7

Построить МДНФ булевой функции y, заданной таблицей, по методу ОК.

$x_4x_3x_2x_1$	y
0100	1
1100	1
0110	1
1110	1
0000	0

1111	0
2024	F_0^p
2420	F_1^p
1111	$F_0^з$
1111	$F_1^з$
3135	F_0
3531	F_1

Решение

1. Выбираем в качестве $БМОК_1$ переменную x_3, т.е. $БМОК_1$ = -1–. Эта $БМОК_1$ покрывает все РОК и один ЗОК.

2. Выписываем эти РОК и ЗОК (см. след. таблицу).

3. По максимальному F_0 = 5 определяем вторую переменную базы $МОК_1$. Это переменная x_1. Она входит в БМОК инверсным значением, т.е. $БМОК_1$ = -1-0.

4. Так как $БМОК_1$ = -1-0 не покрывает ни одного ЗОК и покрывает все РОК, то минимизацию считаем законченной и принимаем $МОК_1$ = $БМОК_1$ = -1-0, т.е.

$$y = x_3x_1'.$$

Такой же результат получается и по карте Карно.

$x_4x_3x_2x_1$	y
0100	1
1100	1
0110	1
1110	1
1111	0
3-35	F_0
2-20	F_1

Задача 8

Построить МДНФ функции, заданной таблицей.

x_8 x_7 x_6 x_5 x_4 x_3 x_2 x_1	y
0 0 1 0 0 1 1 1	1
0 0 1 0 0 0 0 0	1
0 0 1 0 0 0 1 0	1
0 0 1 0 0 1 1 0	1
0 0 1 0 0 1 0 0	1
0 0 1 0 1 1 0 0	1
0 0 1 0 1 1 1 0	1
0 0 1 0 1 0 1 0	1

0 0 1 0 1 0 0 0	1
0 0 1 0 0 0 0 1	0
0 0 0 1 0 0 0 0	0
9 9 1 1 0 5 4 4 9	F_0
2 2 1 0 1 6 7 7 2	F_1

Решение

1. Принимаем $БМОК_1$ по $F_1 = 10$ (можно по $F_0 = 10$).

$БМОК_1$ = --1-----

$БМОК_1$ покрывает один ЗОК и все РОК.

2. Выписываем все РОК и ЗОК, покрываемые $БМОК_1$. После подсчёта оценочных функций оказывается, что максимум F_0 приходится на x_1, поэтому:

$БМОК_1$ = --1----0

$МОК_1 = БМОК_1$ = --1----0

3.Непокрытым оказался только один РОК. Выписываем этот РОК и все ЗОК в таблицу.

0 0 1 0 0 1 1 1	РОК
0 0 1 0 0 0 0 1	ЗОК
0 0 0 1 0 0 0 0	
1 1 1 2 1 0 0 1	F_0
2 2 2 1 2 3 3 2	F_1

4. Находим $БМОК_2$ = -----1--

$МОК_2 = БМОК_2$ = -----1--

Результат минимизации выглядит так:

$y = x_6x_1' + x_3$

Такой же результат получен и по картв Карно.

Задание 3

Найти минимальную форму функций, указанных в задании 1, методом обобщённых кодов.

2.2. Алгоритм соседнего определения базы МОК

Алгоритм 1 требует раздельного размещения РОК и ЗОК. Приведение таблицы истинности к такому виду усложняет метод ОК.

Процесс минимизации методом ОК может быть существенно упрощен, если опредвление БМОК производить с использованием приводимого ниже алгоритма.

Алгоритм 2

1. Присвоить индексу МОК значение 1, т.е. i:= 1.

2. Присвоить индексу РОК значение 1, т.е. j:= 1.

3. Взять РОК из исходной таблицы истинности и, сравнивая его со всеми ЗОК. определить переменные, по которым РОК может быть склеен с ЗОК. Эта совокупность переменных и будет базой $МОК_i$. Перейти к п.7.

4. Если $РОК_j$ не имеет соседних ЗОК, то j:= j + 1 и первйти к п.3. Если в таблице истинности нет ни одного РОК, соседнего хотя бы с одним ЗОК, то перейти к п.5.

5. Подсчитать по таблице истинности F_0 и F_1 для всех разрядов.

6. В качестве базы МОК ($БМОК_i$) или дополнения к БМОК выбрать переменную с максимальной F_0 или F_1. Если F_0 = max, то переменная входит в $БМОК_i$ нулём. Если F_1 = max, то переменная входит в $БМОК_i$ единицей. Если несколько переменных имеют одинаковые оценочные (максимальные) функции, то выбрать в качестве $БМОК_i$ ту переменную, у которой соответствующая запрещённая оценочная функция (F_0^3 или F_1^3) имеет минимальное значение, в противном случае в качестве $БМОК_i$ взять любую переменную с максимальной оценочной функцией F_0 или F_1.

7. Выписать РОК и ЗОК, покрываемые базой $МСК_i$. Если $БМОК_i$ не покрывает ни одного РОК или покрывает все ЗОК, то приравнять нулю оценочные функции F_0 и F_1 для данного разряда и перейти к п.6. Если покрываемых ЗОК нет, то перейти к п.9.

8. Подсчитать F_0 и F_1 для всех разрядов РОК и ЗОК. покрываемых данной базой, кроме разрядов (переменных), образующих $БМОК_i$. Присоединить к $БМОК_i$ переменную (дополнение к $БМОК_i$) с максимальной оценочной функцией в соответствии с требованиями п.6. Считать полученный ОК новой базой $МСК_i$. Если новая $БМОК_i$ покрывает столько же ЗОК, сколько и на предыдущем шаге, то приравнять нулю оценочные функции или дополнения к $БМОК_i$, отбросить присоединённую пременную и добавить к $БМОК_i$ переменную с максимальной оценочной функцией в соответствии с положениями п.6, считать полученный ОК новой $БМОК_i$.Если $БМОК_i$ покрывает хотя бы один ЗОК, перейти к п 7.

9. Принять в качестве $МОК_i$ базу $МОК_i$

10. Если все РОК из исходной таблицы истинности покрыты минимальными обобщёнными кодами, перейти к п.12.

11. Выписать из исходной таблицы истинности все ЗОК и те РОК, которые не покрыты минимальными обобщёнными кодами. Считать вновь полученную таблицу исходной таблицей истинности. Увеличить индекс МОК на вдиницу, т.е. i:= i+1. Перейти к п.2.

12. Конец.

Задача 9

Произвести минимизацию булевой функции, заданной таблицей.

№п/п	x_8 x_7 x_6 x_5 x_4 x_3 x_2 x_1	y
1	0 0 0 0 0 0 0 0	1
2	0 0 0 0 0 0 0 1	1
3	0 0 0 0 0 1 0 0	1
4	0 0 0 0 1 1 0 0	1
5	0 0 0 0 1 0 0 1	1
6	0 0 0 0 1 0 0 0	1
7	0 1 1 0 0 0 0 0	1
8	0 1 1 0 0 0 0 1	1
1	0 0 0 1 0 0 0 1	0
2	0 1 0 1 0 0 0 1	0
3	1 1 0 1 1 0 0 0	0
4	0 0 0 0 0 1 1 0	0

Решение

1. По алгоритму 2 пп. 1-3 для $РОК_2$ находим соседний ЗОК, т.е. находим $БМОК_1$ = ---0----.

2. По алгоритму 2 пп. 7-9 находим

$МОК_1$ = ---0--0-

Так как $МОК_1$ покрывает все РОК, то в соответствии с п.10 алгоритма 2 получаем

$y = x_5'x_2'$

Сущность алгоритма 2 заключается в том, что отыскиваются в первую очередь «необходимые» МОК, т.е. МОК для тех РОК, развязывание которых с ЗОК максимально затруднено, так как они развязываются только по тем переменным, по которым возможно склеивание данного РОК со всеми ЗОК. Под развязыванием понимается процесс выявлвния тех переменных в РОК, которые не встречаются в ЗОК.

Задача 10

Произвести минимизацию булевой функции, заданной таблицей.

№	x_6 x_5 x_4 x_3 x_2 x_1	y
1	0 0 1 0 0 0	1
2	0 0 1 0 0 1	1
3	0 1 1 0 1 1	1
4	0 1 1 1 1 0	1
5	1 0 1 0 0 0	1

6	1 1 1 1 1 1	1
1	0 0 1 0 1 1	0
2	0 0 1 0 1 0	0
3	0 0 1 1 0 1	0
4	1 1 0 1 1 0	0
5	1 1 1 0 1 0	0
6	1 1 1 1 0 1	0

Решение

1. По алгоритму 2 пп.1-3 для $РОК_1$ находим $БМОК_1$ = ----0-

2. По алгоритму 2 пп.7, 8 строим таблицу.

1	0 0 1 0 0 0	1
2	0 0 1 0 0 1	1
5	1 0 1 0 0 0	1
	0 0 1 1 0 1	0
	1 1 1 0 1	0

3. По алгоритму 2 п.8 из таблицы находим $БМОК_1$ = ---00-

4. По алгоритму 2 п.9

$МОК_1$ = $БМОК_1$ = ---00-

5. По алгоритму 2 для непокрытых РОК (для $РОК_3$) находим $БМОК_2$ = -1----.

6. По алгоритму 2 пп.7, 8 , 11 строим таблицу.

3	0 1 1 0 1 1	1
4	0 1 1 1 1 0	1
6	1 1 1 1 1 1	1
	1 1 0 1 1 0	0
	1 1 1 0 1 0	0
	1 1 1 1 0 1	0
	5 - 2 3 2 2	F_0
	1 - 4 3 4 4	F_1

7. По алгоритму 2 п.9 находим $МОК_2$

$МОК_2$ = 01----

8. По алгоритму 2 пп.7, 8, 11 строим следующую таблицу.

1 1 1 1 1 1	1
0 0 1 0 1 1	0
0 0 1 0 1 0	0
0 0 1 1 0 1	0
1 1 0 1 1 0	0
1 1 1 0 1 0	0
1 1 1 1 0 1	0

9. По алгоритму 2 п.3 находим $БМОК_3$

$БМОК_3$ = —1-

10. По алгоритму 2 пп. 7, 8, 11 строим таблицу.

111111	1
001011	0
001010	0
110110	0
111010	0
2231-1	F_0
3324-4	F_1

11. Из таблицы по алгоритму 2 п.8 находим

$БМОК_3$ = ----11

12. По алгоритму 2 пп. 7, 8 строим следующую таблицу.

111111	1
001011	0
0010--	F_0
1111--	F_1

13. По таблице 17 опрвделяем

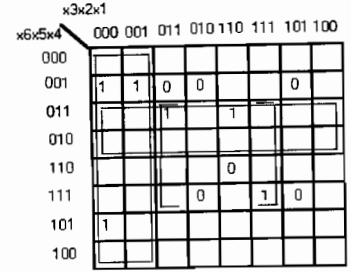
$МОК_3$ = -1–11

Таким образом,

$y = x_3'x_2' + x_5x_2x_1 + x_6'x_5$

На рисунке представлено решение задачи 10 с помощью карты Карно. Функция y представлена в виде МДНФ.

Из рисунка видно, что результаты минимизации по карте Карно и по методу обобщённых кодов совпадают.



Карта Карно к задаче 10.

Задача 10а

Произвести минимизацию логической функции, заданной таблицей истинности.

N°	$x_8x_7x_6x_5x_4x_3x_2x_1$	y
1	00110011	1
2	00100010	1
3	00110111	1
4	10110111	1
5	00101011	1
6	01101101	1
7	01111110	1
1	01110111	0
2	11010111	0
3	00011110	0
4	11111101	0
5	10101110	0
	98377855	F_0
	34955477	F_1

Т.к. $РОК_3$ и $ЗОК_1$ являются соседними по x_7,то в качестве $БМОК_1$ выбираем x_7', не обращая внимание на абсолютный максимум F_0 для x_8.$БМОК_1$ покрывает $РОК_1$ – $РОК_5$ и $ЗОК_3$, – $ЗОК_5$. Подсчитаем оценочные функции для выбора дополнения к $БМОК_1$ и получения $МОК_1$.

N°	$x_8x_7x_6x_5x_4x_3x_2x_1$	y
1	00110011	1
2	00100010	1
3	00110111	1
4	10110111	1
5	00101011	1
3	00011110	0
5	10101110	0
	5x136521	F_0
	2x641256	F_1

Дополнвние x_4' могло бы привести нас к МДНФ, поэтому мы выберем эквивалентное по оценочной функции дополненив x_1, чтобы убедиться в нвкоторых недостатках метода. В результатв получим $МОК_1 = x_7'x_1$ Поскольку $МОК_1$ покрывает РОК с номерами 1, 3 – 5, то стартовая таблица для синтеза $БМОК_2$ выглядит так:

N°	$x_8x_7x_6x_5x_4x_3x_2x_1$	y
2	00100010	1
6	01101101	1
7	01111110	1
1	01110111	0
2	11010111	0
3	00011110	0
4	11111101	0
5	10101110	0
	64364655	F_0
	24524233	F_1

Исходя из этой таблицы получаем $БМОК_2 = x_8'$. Для нахождения $МОК_2$ строим следующую таблицу.

N°	$x_8x_7x_6x_5x_4x_3x_2x_1$	y
2	00100010	1
6	01101101	1
7	01111110	1
1	01110111	0
3	00011110	0
	x2142333	F_0
	x3413222	F_1

Отсюда получаем $МОК_2 = x_8'x_5'$, который дополнительно покрывает $РОК_2$ и $РОК_6$. Найдём $БМОК_3$.

N°	$x_8x_7x_6x_5x_4x_3x_2x_1$	y
7	01111110	1
1	01110111	0
2	11010111	0
3	00011110	0
4	11111101	0
5	10101110	0
	43343544	F_0
	23323122	F_1

F_0 для x_3 имеет максимальное значвние, но использовать x_3' в качестве $БМОК_3$ нельзя, поскольку единственный РОК не имеет нуля в данном разряде. Принимаем $БМОК_3 = x_8'$ Строим очередную таблицу для синтеза последнего МОК.

N°	$x_8x_7x_6x_5x_4x_3x_2x_1$	y
7	01111110	1
1	01110111	0
3	00011110	0

	x1121222	F_0
	x2212111	F_1

Из последней таблицы получаем $МОК_3 = x_8'x_7x_4$. Таким образом мы получили тупиковую ДНФ

$y = x_8'x_7x_4 + x_8'x_5' + x_7'x_1$.

По карте Карно получена минимальная ДНФ

$y = x_8'x_5' + x_7x_1' + x_7'x_4'$

Т.е. высокая эффективность метода обобщённых кодов не всегда гарантирует получение МДНФ. Кроме того, если рассмотреть недоопределённую логическую функцию, заданную восьмеричными наборами: РОК – 67, 73, 63 и ЗОК – 37, 65, 66, то окажется, что по первому алгоритму получим избыточное решение. В этом случае $y = x_3' + x_6x_2x_1$ При минимизации по второму алгоритму $y = x_8x_2x_1$. Таким образом, алгоритм 2 не только менее трудоёмок, но и более эффективен.

Задача 11

Для синхронизатора цифрового фильтра синтезировать функции J_{12}, K_{12}, J_{11}, K_{11}, J_{10}, K_{10} в соответствии с таблицей. Прочерк в таблице означает, что функция на данном наборе не определена.

№ хода	Q_{12}	Q_{11}	Q_{10}	Q_9	Q_8	Q_7	Q_6	Q_5	Q_4	Q_3	Q_2	Q_1	J_{12}	K_{12}	J_{11}	K_{11}	J_{10}	K_{10}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	-	0	-
1	0	0	0	0	0	0	0	0	0	0	0	1	0	-	0	-	0	-
2	0	0	0	0	0	0	0	0	0	0	1	0	0	-	0	-	0	-
3	0	0	0	0	0	0	0	0	0	0	1	1	0	-	0	-	0	-
4	0	0	0	0	0	0	0	0	0	1	0	0	0	-	0	-	0	-
5	0	0	0	0	0	0	0	0	0	1	0	1	0	-	0	-	0	-
6	0	0	0	0	0	0	0	0	0	1	1	0	0	-	0	-	0	-
7	0	0	0	0	0	0	0	0	0	1	1	1	0	-	0	-	0	-
8	0	0	0	0	0	0	0	0	1	0	0	0	0	-	0	-	0	-
9	0	0	0	0	0	0	0	0	1	0	0	1	0	-	0	-	0	-
10	0	0	0	0	0	0	0	0	1	0	1	0	0	-	0	-	0	-
11	0	0	0	0	0	0	0	0	1	0	1	1	0	-	0	-	0	-
12	0	0	0	0	0	0	0	0	1	1	0	0	0	-	0	-	0	-
13	0	0	0	0	0	0	0	0	1	1	0	1	0	-	0	-	0	-
14	0	0	0	0	0	0	0	0	1	1	1	0	0	-	0	-	0	-
15	0	0	0	0	0	0	0	0	1	1	1	1	0	-	0	-	0	-
16	0	0	0	0	0	0	0	1	0	0	0	0	0	-	0	-	0	-
17	0	0	0	0	0	0	0	1	0	0	0	1	0	-	0	-	0	-
18	0	0	0	0	0	0	0	1	0	0	1	0	0	-	0	-	0	-

19	0	0	0	0	0	0	0	1	0	0	1	1	0	-	0	-	0	-
20	0	0	0	0	0	0	0	1	0	1	0	0	0	-	0	-	0	-
21	0	0	0	0	0	0	0	1	0	1	0	1	0	-	0	-	0	-
22	0	0	0	0	0	0	0	1	0	1	1	0	0	-	0	-	0	-
23	0	0	0	0	0	0	0	1	0	1	1	1	0	-	0	-	0	-
24	0	0	0	0	0	0	0	1	1	0	0	0	0	-	0	-	0	-
25	0	0	0	0	0	0	0	1	1	0	0	1	0	-	0	-	0	-
26	0	0	0	0	0	0	0	1	1	0	1	0	0	-	0	-	0	-
27	0	0	0	0	0	0	0	1	1	0	1	1	0	-	0	-	0	-
28	0	0	0	0	0	0	0	1	1	1	0	0	0	-	0	-	0	-
29	0	0	0	0	0	0	0	1	1	1	0	1	0	-	0	-	0	-
30	0	0	0	0	0	0	0	1	1	1	1	0	0	-	0	-	0	-
31	0	0	0	0	0	0	0	1	1	1	1	1	0	-	0	-	0	-
32	0	0	0	0	0	0	1	0	0	0	0	0	0	-	0	-	0	-
33	0	0	0	0	0	0	1	0	0	0	0	1	0	-	0	-	0	-
34	0	0	0	0	0	0	1	0	0	0	1	0	0	-	0	-	0	-
35	0	0	0	0	0	0	1	0	0	0	1	1	0	-	0	-	0	-
36	0	0	0	0	0	0	1	0	0	1	0	0	0	-	0	-	0	-
37	0	0	0	0	0	0	1	0	0	1	0	1	0	-	0	-	0	-
38	0	0	0	0	0	0	1	0	0	1	1	0	0	-	0	-	0	-
39	0	0	0	0	0	0	1	0	0	1	1	1	0	-	0	-	0	-
40	0	0	0	0	0	0	1	0	1	0	0	0	0	-	0	-	0	-
41	0	0	0	0	0	0	1	0	1	0	0	1	0	-	0	-	0	-
42	0	0	0	0	0	0	1	0	1	0	1	0	0	-	0	-	0	-
43	0	0	0	0	0	0	1	0	1	0	1	1	0	-	0	-	0	-
44	0	0	0	0	0	0	1	0	1	1	0	0	0	-	0	-	0	-
45	0	0	0	0	0	0	1	0	1	1	0	1	0	-	0	-	0	-
46	0	0	0	0	0	0	1	0	1	1	1	0	0	-	0	-	0	-
47	0	0	0	0	0	0	1	0	1	1	1	1	0	-	0	-	0	-
48	0	0	0	0	0	0	1	1	0	0	0	0	0	-	0	-	0	-
49	0	0	0	0	0	0	1	1	0	0	0	1	0	-	0	-	0	-
50	0	0	0	0	0	0	1	1	0	0	1	0	0	-	0	-	0	-
51	0	0	0	0	0	0	1	1	0	0	1	1	0	-	0	-	0	-
52	0	0	0	0	0	0	1	1	0	1	0	0	0	-	0	-	0	-
53	0	0	0	0	0	0	1	1	0	1	0	1	0	-	0	-	0	-
54	0	0	0	0	0	0	1	1	0	1	1	0	0	-	0	-	0	-
55	0	0	0	0	0	0	1	1	0	1	1	1	0	-	0	-	0	-
56	0	0	0	0	0	0	1	1	1	0	0	0	0	-	0	-	0	-
57	0	0	0	0	0	0	1	1	1	0	0	1	0	-	0	-	0	-
58	0	0	0	0	0	0	1	1	1	0	1	0	0	-	0	-	0	-
59	0	0	0	0	0	0	1	1	1	0	1	1	0	-	0	-	0	-
60	0	0	0	0	0	0	1	1	1	1	0	0	0	-	0	-	0	-
61	0	0	0	0	0	0	1	1	1	1	0	1	0	-	0	-	0	-
62	0	0	0	0	0	0	1	1	1	1	1	0	0	-	0	-	0	-
63	0	0	0	0	0	0	1	1	1	1	1	1	0	-	0	-	0	-
64	0	0	0	0	0	1	0	0	0	0	0	0	0	-	0	-	0	-

65	0	0	0	0	0	1	0	0	0	0	0	1	0	-	0	-	0	-
66	0	0	0	0	0	1	0	0	0	0	1	0	0	-	0	-	0	-
67	0	0	0	0	0	1	0	0	0	0	1	1	0	-	0	-	0	-
68	0	0	0	0	0	1	0	0	0	1	0	0	0	-	0	-	0	-
69	0	0	0	0	0	1	0	0	0	1	0	1	0	-	0	-	0	-
70	0	0	0	0	0	1	0	0	0	1	1	0	0	-	0	-	0	-
71	0	0	0	0	0	1	0	0	0	1	1	1	0	-	0	-	0	-
72	0	0	0	0	0	1	0	0	1	0	0	0	0	-	0	-	0	-
73	0	0	0	0	0	1	0	0	1	0	0	1	0	-	0	-	0	-
74	0	0	0	0	0	1	0	0	1	0	1	0	0	-	0	-	0	-
75	0	0	0	0	0	1	0	0	1	0	1	1	0	-	0	-	0	-
76	0	0	0	0	0	1	0	0	1	1	0	0	0	-	0	-	0	-
77	0	0	0	0	0	1	0	0	1	1	0	1	0	-	0	-	0	-
78	0	0	0	0	0	1	0	0	1	1	1	0	0	-	0	-	0	-
79	0	0	0	0	0	1	0	0	1	1	1	1	0	-	0	-	0	-
80	0	0	0	0	0	1	0	1	0	0	0	0	0	-	0	-	0	-
81	0	0	0	0	0	1	0	1	0	0	0	1	0	-	0	-	0	-
82	0	0	0	0	0	1	0	1	0	0	1	0	0	-	0	-	0	-
83	0	0	0	0	0	1	0	1	0	0	1	1	0	-	0	-	0	-
84	0	0	0	0	0	1	0	1	0	1	0	0	0	-	0	-	0	-
85	0	0	0	0	0	1	0	1	0	1	0	1	0	-	0	-	0	-
86	0	0	0	0	0	1	0	1	0	1	1	0	0	-	0	-	0	-
87	0	0	0	0	0	1	0	1	0	1	1	1	0	-	0	-	0	-
88	0	0	0	0	0	1	0	1	1	0	0	0	0	-	0	-	0	-
89	0	0	0	0	0	1	0	1	1	0	0	1	0	-	0	-	0	-
90	0	0	0	0	0	1	0	1	1	0	1	0	0	-	0	-	0	-
91	0	0	0	0	0	1	0	1	1	0	1	1	0	-	0	-	0	-
92	0	0	0	0	0	1	0	1	1	1	0	0	0	-	0	-	0	-
93	0	0	0	0	0	1	0	1	1	1	0	1	0	-	0	-	0	-
94	0	0	0	0	0	1	0	1	1	1	1	0	0	-	0	-	0	-
95	0	0	0	0	0	1	0	1	1	1	1	1	0	-	0	-	0	-
96	0	0	0	0	0	1	1	0	0	0	0	0	0	-	0	-	0	-
97	0	0	0	0	0	1	1	0	0	0	0	1	0	-	0	-	0	-
98	0	0	0	0	0	1	1	0	0	0	1	0	0	-	0	-	0	-
99	0	0	0	0	0	1	1	0	0	0	1	1	0	-	0	-	0	-
100	0	0	0	0	0	1	1	0	0	1	0	0	0	-	0	-	0	-
101	0	0	0	0	0	1	1	0	0	1	0	1	0	-	0	-	0	-
102	0	0	0	0	0	1	1	0	0	1	1	0	0	-	0	-	0	-
103	0	0	0	0	0	1	1	0	0	1	1	1	0	-	0	-	0	-
104	0	0	0	0	0	1	1	0	1	0	0	0	0	-	0	-	0	-
105	0	0	0	0	0	1	1	0	1	0	0	1	0	-	0	-	0	-
106	0	0	0	0	0	1	1	0	1	0	1	0	0	-	0	-	0	-
107	0	0	0	0	0	1	1	0	1	0	1	1	0	-	0	-	0	-
108	0	0	0	0	0	1	1	0	1	1	0	0	0	-	0	-	0	-
109	0	0	0	0	0	1	1	0	1	1	0	1	0	-	0	-	0	-
110	0	0	0	0	0	1	1	0	1	1	1	0	0	-	0	-	0	-

111	0	0	0	0	0	1	1	0	1	1	1	1	0	-	0	-	0	-
112	0	0	0	0	0	1	1	1	0	0	0	0	0	-	0	-	0	-
113	0	0	0	0	0	1	1	1	1	0	0	0	1	-	0	-	0	-
114	0	0	0	0	0	1	1	1	1	0	0	1	0	-	0	-	0	-
115	0	0	0	0	0	1	1	1	1	0	0	1	1	-	0	-	0	-
116	0	0	0	0	0	1	1	1	1	0	1	0	0	-	0	-	0	-
117	0	0	0	0	0	1	1	1	1	0	1	0	1	-	0	-	0	-
118	0	0	0	0	0	1	1	1	1	0	1	1	0	-	0	-	0	-
119	0	0	0	0	0	1	1	1	1	0	1	1	1	-	0	-	0	-
120	0	0	0	0	0	1	1	1	1	1	0	0	0	-	0	-	0	-
121	0	0	0	0	0	1	1	1	1	1	0	0	1	-	0	-	0	-
122	0	0	0	0	0	1	1	1	1	1	0	1	0	-	0	-	0	-
123	0	0	0	0	0	1	1	1	1	1	0	1	1	-	0	-	0	-
124	0	0	0	0	0	1	1	1	1	1	1	0	0	-	0	-	0	-
125	0	0	0	0	0	1	1	1	1	1	1	0	1	-	0	-	0	-
126	0	0	0	0	0	1	1	1	1	1	1	1	0	-	0	-	0	-
127	0	0	0	0	0	1	1	1	1	1	1	1	1	-	0	-	0	-
128	0	0	0	0	1	0	0	0	0	0	0	0	0	-	0	-	0	-
129	0	0	0	0	1	0	0	0	0	0	0	0	1	-	0	-	0	-
130	0	0	0	0	1	0	0	0	0	0	0	1	0	-	0	-	0	-
131	0	0	0	0	1	0	0	0	0	0	0	1	1	-	0	-	0	-
132	0	0	0	0	1	0	0	0	0	0	1	0	0	-	0	-	0	-
133	0	0	0	0	1	0	0	0	0	0	1	0	1	-	0	-	0	-
134	0	0	0	0	1	0	0	0	0	0	1	1	0	-	0	-	0	-
135	0	0	0	0	1	0	0	0	0	0	1	1	1	-	0	-	0	-
136	0	0	0	0	1	0	0	0	0	1	0	0	0	-	0	-	0	-
137	0	0	0	0	1	0	0	0	0	1	0	0	1	-	0	-	0	-
138	0	0	0	0	1	0	0	0	0	1	0	1	0	-	0	-	0	-
139	0	0	0	0	1	0	0	0	0	1	0	1	1	-	0	-	0	-
140	0	0	0	0	1	0	0	0	0	1	1	0	0	-	0	-	0	-
141	0	0	0	0	1	0	0	0	0	1	1	0	1	-	0	-	0	-
142	0	0	0	0	1	0	0	0	0	1	1	1	0	-	0	-	0	-
143	0	0	0	0	1	0	0	0	0	1	1	1	1	-	0	-	0	-
144	0	0	0	0	1	0	0	0	1	0	0	0	0	-	0	-	0	-
145	0	0	0	0	1	0	0	0	1	0	0	0	1	-	0	-	0	-
146	0	0	0	0	1	0	0	0	1	0	0	1	0	-	0	-	0	-
147	0	0	0	0	1	0	0	0	1	0	0	1	1	-	0	-	0	-
148	0	0	0	0	1	0	0	0	1	0	1	0	0	-	0	-	0	-
149	0	0	0	0	1	0	0	0	1	0	1	0	1	-	0	-	0	-
150	0	0	0	0	1	0	0	0	1	0	1	1	0	-	0	-	0	-
151	0	0	0	0	1	0	0	0	1	0	1	1	1	-	0	-	0	-
152	0	0	0	0	1	0	0	0	1	1	0	0	0	-	0	-	0	-
153	0	0	0	0	1	0	0	0	1	1	0	0	1	-	0	-	0	-
154	0	0	0	0	1	0	0	0	1	1	0	1	0	-	0	-	0	-
155	0	0	0	0	1	0	0	0	1	1	0	1	1	-	0	-	0	-
156	0	0	0	0	1	0	0	1	1	1	0	0	0	-	0	-	0	-

157	0	0	0	0	1	0	0	1	1	1	0	1	0	-	0	-	0	-
158	0	0	0	0	1	0	0	1	1	1	1	0	0	-	0	-	0	-
159	0	0	0	0	1	0	0	1	1	1	1	1	0	-	0	-	0	-
160	0	0	0	0	1	0	1	0	0	0	0	0	0	-	0	-	0	-
161	0	0	0	0	1	0	1	0	0	0	0	0	1	-	0	-	0	-
162	0	0	0	0	1	0	1	0	0	0	0	1	0	-	0	-	0	-
163	0	0	0	0	1	0	1	0	0	0	0	1	1	-	0	-	0	-
164	0	0	0	0	1	0	1	0	0	0	1	0	0	-	0	-	0	-
165	0	0	0	0	1	0	1	0	0	0	1	0	1	-	0	-	0	-
166	0	0	0	0	1	0	1	0	0	0	1	1	0	-	0	-	0	-
167	0	0	0	0	1	0	1	0	0	0	1	1	1	-	0	-	0	-
168	0	0	0	0	1	0	1	0	1	0	0	0	0	-	0	-	0	-
169	0	0	0	0	1	0	1	0	1	0	0	0	1	-	0	-	0	-
170	0	0	0	0	1	0	1	0	1	0	1	0	0	-	0	-	0	-
171	0	0	0	0	1	0	1	0	1	0	1	0	1	-	0	-	0	-
172	0	0	0	0	1	0	1	0	1	1	0	0	0	-	0	-	0	-
173	0	0	0	0	1	0	1	0	1	1	0	0	1	-	0	-	0	-
174	0	0	0	0	1	0	1	0	1	1	1	1	0	-	0	-	0	-
175	0	0	0	0	1	0	1	0	1	1	1	1	1	-	0	-	0	-
176	0	0	0	0	1	0	1	1	0	0	0	0	0	-	0	-	0	-
177	0	0	0	0	1	0	1	1	0	0	0	0	1	-	0	-	0	-
178	0	0	0	0	1	0	1	1	0	0	0	1	0	-	0	-	0	-
179	0	0	0	0	1	0	1	1	0	0	0	1	1	-	0	-	0	-
180	0	0	0	0	1	0	1	1	0	0	1	0	0	-	0	-	0	-
181	0	0	0	0	1	0	1	1	0	0	1	0	1	-	0	-	0	-
182	0	0	0	0	1	0	1	1	0	0	1	1	0	-	0	-	0	-
183	0	0	0	0	1	0	1	1	0	0	1	1	1	-	0	-	0	-
184	0	0	0	0	1	0	1	1	1	0	0	0	0	-	0	-	0	-
185	0	0	0	0	1	0	1	1	1	0	0	0	1	-	0	-	0	-
186	0	0	0	0	1	0	1	1	1	0	0	1	0	-	0	-	0	-
187	0	0	0	0	1	0	1	1	1	0	0	1	1	-	0	-	0	-
188	0	0	0	0	1	0	1	1	1	1	0	0	0	-	0	-	0	-
189	0	0	0	0	1	0	1	1	1	1	0	0	1	-	0	-	0	-
190	0	0	0	0	1	0	1	1	1	1	0	1	0	-	0	-	0	-
191	0	0	0	0	1	0	1	1	1	1	1	1	1	-	0	-	0	-
192	0	0	0	0	1	1	0	0	0	0	0	0	0	-	0	-	0	-
193	0	0	0	0	1	1	0	0	0	0	0	0	1	-	0	-	0	-
194	0	0	0	0	1	1	0	0	0	0	0	1	0	-	0	-	0	-
195	0	0	0	0	1	1	0	0	0	0	0	1	1	-	0	-	0	-
196	0	0	0	0	1	1	0	0	0	0	1	0	0	-	0	-	0	-
197	0	0	0	0	1	1	0	0	0	0	1	0	1	-	0	-	0	-
198	0	0	0	0	1	1	0	0	0	0	1	1	0	-	0	-	0	-
199	0	0	0	0	1	1	0	0	0	0	1	1	1	-	0	-	0	-
200	0	0	0	0	1	1	0	0	0	1	0	0	0	-	0	-	0	-
201	0	0	0	0	1	1	0	0	0	1	0	0	1	-	0	-	0	-
202	0	0	0	0	1	1	0	0	0	1	0	1	0	-	0	-	0	-

203	0	0	0	0	1	1	0	0	1	0	1	1	0	-	0	-	0	-	
204	0	0	0	0	1	1	0	0	1	1	0	0	0	-	0	-	0	-	
205	0	0	0	0	1	1	0	0	1	1	0	1	0	-	0	-	0	-	
206	0	0	0	0	1	1	0	0	1	1	1	0	0	-	0	-	0	-	
207	0	0	0	0	1	1	0	0	1	1	1	1	0	-	0	-	0	-	
208	0	0	0	0	1	1	0	1	0	0	0	0	0	-	0	-	0	-	
209	0	0	0	0	1	1	0	1	0	0	0	1	0	-	0	-	0	-	
210	0	0	0	0	1	1	0	1	0	0	1	0	0	-	0	-	0	-	
211	0	0	0	0	1	1	0	1	0	0	1	1	0	-	0	-	0	-	
212	0	0	0	0	1	1	0	1	0	1	0	0	0	-	0	-	0	-	
213	0	0	0	0	1	1	0	1	0	1	0	1	0	-	0	-	0	-	
214	0	0	0	0	1	1	0	1	0	1	1	0	0	-	0	-	0	-	
215	0	0	0	0	1	1	0	1	0	1	1	1	0	-	0	-	0	-	
216	0	0	0	0	1	1	0	1	1	0	0	0	0	-	0	-	0	-	
217	0	0	0	0	1	1	0	1	1	0	0	1	0	-	0	-	0	-	
218	0	0	0	0	1	1	0	1	1	0	1	0	0	-	0	-	0	-	
219	0	0	0	0	1	1	0	1	1	0	1	1	0	-	0	-	0	-	
220	0	0	0	0	1	1	0	1	1	1	0	0	0	-	0	-	0	-	
221	0	0	0	0	1	1	0	1	1	1	0	1	0	-	0	-	0	-	
222	0	0	0	0	1	1	0	1	1	1	1	0	0	-	0	-	0	-	
223	0	0	0	0	1	1	0	1	1	1	1	1	0	-	0	-	0	-	
224	0	0	0	0	1	1	1	0	0	0	0	0	0	-	0	-	0	-	
225	0	0	0	0	1	1	1	0	0	0	0	1	0	-	0	-	0	-	
226	0	0	0	0	1	1	1	0	0	0	1	0	0	-	0	-	0	-	
227	0	0	0	0	1	1	1	0	0	0	1	1	0	-	0	-	0	-	
228	0	0	0	0	1	1	1	0	0	1	0	0	0	-	0	-	0	-	
229	0	0	0	0	1	1	1	0	0	1	0	1	0	-	0	-	0	-	
230	0	0	0	0	1	1	1	0	0	1	1	0	0	-	0	-	0	-	
231	0	0	0	0	1	1	1	0	0	1	1	1	0	-	0	-	0	-	
232	0	0	0	0	1	1	1	0	1	0	0	0	0	-	0	-	0	-	
233	0	0	0	0	1	1	1	0	1	0	0	1	0	-	0	-	0	-	
234	0	0	0	0	1	1	1	0	1	0	1	0	0	-	0	-	0	-	
235	0	0	0	0	1	1	1	0	1	0	1	1	0	-	0	-	0	-	
236	0	0	0	0	1	1	1	0	1	1	0	0	0	-	0	-	0	-	
237	0	0	0	0	1	1	1	0	1	1	0	1	0	-	0	-	0	-	
238	0	0	0	0	1	1	1	0	1	1	1	0	0	-	0	-	1	-	
239	0	0	0	0	1	1	1	0	1	1	1	1	0	-	0	-	-	0	
240	0	0	0	0	1	1	1	1	0	0	0	0	0	-	0	-	-	0	
241	0	0	0	0	1	1	1	1	0	0	0	1	0	-	0	-	-	0	
242	0	0	0	0	1	1	1	1	0	0	1	0	0	-	0	-	-	0	
243	0	0	0	0	1	1	1	1	0	0	1	1	0	-	0	-	-	0	
244	0	0	0	0	1	1	1	1	0	1	0	0	0	-	0	-	-	0	
245	0	0	0	0	1	1	1	1	0	1	0	1	0	-	0	-	-	0	
246	0	0	0	0	1	1	1	1	0	1	1	0	0	-	0	-	-	0	
247	0	0	0	0	1	1	1	1	0	1	1	1	0	-	0	-	-	0	
248	0	0	0	0	1	1	1	1	1	0	0	0	0	-	0	-	-	1	

249	0	0	0	0	1	1	1	1	1	0	0	1	0	-	0	-	0	-	
250	0	0	0	0	1	1	1	1	1	0	1	0	0	-	0	-	0	-	
251	0	0	0	0	1	1	1	1	1	0	1	1	0	-	0	-	0	-	
252	0	0	0	0	1	1	1	1	1	1	0	0	0	-	0	-	0	-	
253	0	0	0	0	1	1	1	1	1	1	0	1	0	-	0	-	0	-	
254	0	0	0	0	1	1	1	1	1	1	1	0	0	-	0	-	0	-	
255	0	0	0	0	1	1	1	1	1	1	1	1	0	-	0	-	0	-	
256	0	0	0	1	0	0	0	0	0	0	0	0	1	-	0	-	0	-	
257	0	0	0	1	0	0	0	0	0	0	0	1	0	-	0	-	0	-	
258	0	0	0	1	0	0	0	0	0	0	0	1	1	-	0	-	0	-	
259	0	0	0	1	0	0	0	0	0	0	1	0	0	-	0	-	0	-	
260	0	0	0	1	0	0	0	0	0	0	1	0	1	-	0	-	0	-	
261	0	0	0	1	0	0	0	0	0	0	1	1	0	-	0	-	0	-	
262	0	0	0	1	0	0	0	0	0	0	1	1	1	-	1	-	0	-	
263	0	0	0	1	0	0	0	0	0	1	0	0	0	-	-	0	0	-	
264	0	0	0	1	0	0	0	0	0	1	0	0	1	-	-	0	0	-	
265	0	0	0	1	0	0	0	0	0	1	0	1	0	-	-	0	0	-	
266	0	0	0	1	0	0	0	0	0	1	0	1	1	-	-	0	0	-	
267	0	0	0	1	0	0	0	0	0	1	1	0	0	-	-	1	0	-	
268	0	0	0	1	0	0	0	0	0	1	1	0	1	-	-	-	0	-	
269	0	0	0	1	0	0	0	0	0	1	1	1	0	-	0	-	0	-	
270	0	0	0	1	0	0	0	0	0	1	1	1	1	-	0	-	0	-	
271	0	0	0	1	0	0	0	0	1	0	0	0	0	-	0	-	0	-	
272	0	0	0	1	0	0	0	0	1	0	0	0	1	-	0	-	0	-	
273	0	0	0	1	0	0	0	0	1	0	0	1	0	-	0	-	0	-	
274	1	0	0	1	0	0	0	0	1	0	0	1	1	-	0	-	0	-	
275	1	0	0	1	0	0	0	0	1	0	0	1	0	-	0	-	0	-	
276	1	0	0	1	0	0	0	0	1	0	1	0	0	-	0	-	0	-	
277	1	0	0	1	0	0	0	0	1	0	1	0	1	-	0	-	0	-	
278	1	0	0	1	0	0	0	0	1	0	1	1	0	-	0	-	0	-	
279	1	0	0	1	0	0	0	0	1	0	1	1	1	-	0	-	0	-	
280	1	0	0	1	0	0	0	0	1	1	0	0	0	-	0	-	0	-	
281	1	0	0	1	0	0	0	0	1	1	0	0	1	-	0	-	0	-	
282	1	0	0	1	0	0	0	0	1	1	0	1	0	-	0	-	0	-	
283	1	0	0	1	0	0	0	0	1	1	0	1	1	-	0	-	0	-	
284	1	0	0	1	0	0	0	0	1	1	1	0	0	-	0	-	0	-	
285	1	0	0	1	0	0	0	0	1	1	1	0	1	-	0	-	0	-	
286	1	0	0	1	0	0	0	0	1	1	1	1	0	-	0	-	0	-	
287	1	0	0	1	0	0	0	0	1	1	1	1	1	-	0	-	0	-	
288	1	0	0	1	0	0	0	1	0	0	0	0	0	-	1	-	0	-	
289	0	0	0	1	0	0	0	1	0	0	0	0	1	0	-	0	-	0	
290	0	0	0	1	0	0	0	1	0	0	0	1	0	0	-	0	-	0	
291	0	0	0	1	0	0	0	1	0	0	0	1	1	0	-	0	-	0	
292	0	0	0	1	0	0	0	1	0	0	1	0	0	0	-	0	-	0	
293	0	0	0	1	0	0	0	1	0	0	1	0	1	0	-	0	-	0	
294	0	0	0	1	0	0	0	1	0	0	1	1	0	0	-	0	-	0	

295	0	0	0	1	0	0	1	0	0	1	1	1	0	-	0	-	0	-
296	0	0	0	1	0	0	1	0	1	0	0	0	0	-	0	-	0	-
297	0	0	0	1	0	0	1	0	1	0	0	1	0	-	0	-	0	-
298	0	0	0	1	0	0	1	0	1	0	1	0	0	-	0	-	0	-
299	0	0	0	1	0	0	1	0	1	0	1	1	0	-	0	-	0	-
300	0	0	0	1	0	0	1	0	1	1	0	0	0	-	0	-	0	-
301	0	0	0	1	0	0	1	0	1	1	0	1	0	-	0	-	0	-
302	0	0	0	1	0	0	1	0	1	1	1	0	0	-	0	-	0	-
303	0	0	0	1	0	0	1	0	1	1	1	1	0	-	0	-	0	-
304	0	0	0	1	0	0	1	1	0	0	0	0	0	-	0	-	0	-
305	0	0	0	1	0	0	1	1	0	0	0	1	0	-	0	-	0	-
306	0	0	0	1	0	0	1	1	0	0	1	0	0	-	0	-	0	-
307	0	0	0	1	0	0	1	1	0	0	1	1	0	-	0	-	0	-
308	0	0	0	1	0	0	1	1	0	1	0	0	0	-	0	-	0	-
309	0	0	0	1	0	0	1	1	0	1	0	1	0	-	0	-	0	-
310	0	0	0	1	0	0	1	1	0	1	1	0	0	-	0	-	0	-
311	0	0	0	1	0	0	1	1	0	1	1	1	0	-	0	-	0	-
312	0	0	0	1	0	0	1	1	1	0	0	0	0	-	0	-	0	-
313	0	0	0	1	0	0	1	1	1	0	0	1	0	-	0	-	0	-
314	0	0	0	1	0	0	1	1	1	0	1	0	0	-	0	-	0	-
315	0	0	0	1	0	0	1	1	1	0	1	1	0	-	0	-	0	-
316	0	0	0	1	0	0	1	1	1	1	0	0	0	-	0	-	0	-
317	0	0	0	1	0	0	1	1	1	1	0	1	0	-	0	-	0	-
318	0	0	0	1	0	0	1	1	1	1	1	0	0	-	0	-	0	-
319	0	0	0	1	0	0	1	1	1	1	1	1	0	-	0	-	0	-

Решение

Применяя алгоритмы 1 и 2, получим:

$J_{10} = Q_8Q_7Q_6Q_5'Q_4Q_3Q_2$

$K_{10} = Q_4Q_3'$

$J_{11} = Q_9Q_8'Q_5'Q_4'Q_3Q_2$

$K_{11} = Q_4Q_3$

$J_{12} = Q_9Q_8'Q_5Q_4'Q_3'Q_2'Q_1$

$K_{12} = Q_6$

Результат был получен вручную, поскольку не пришлось подсчитывать оценочные функции: МОК, были определены на основе соседних кодов.

Проверка результата минимизации булевых функций

Результат минимизации булевой функции является корректным, если выполняются следующие условия:

1. Совокупность МОК покрывает все РОК.
2. Совокупность МОК не покрывает ни одного ЗОК.

2.3. Выводы

Далеко не всегда по методу ОК может быть получена МДНФ. Чаще всего в результате минимизации удаётся получить одну из тупиковых ДНФ. В этом недостаток метода. Алгоритм 2 по сравнению с алгоритмом 1 даёт более компактный результат, т.е. вероятность получения МДНФ по алгоритму 2 выше, чем по алгоритму 1.

Достоинствами метода являются простота и высокая скорость получения результата. Особенно этот метод эффективен для минимизации булевых функций от большого числа переменных ($n \geq 8$). Вполне приемлемым даже без применения ЦВМ является число наборов, на которых задана функция, в пределах 1000. Например, 6 булевых функций от 12 переменных, определённые на 320 наборах (см. задачу 11) были отминимизированы вручную в течение 30 минут. Разумеется, задача такой сложности может быть решена на ЭВМ. Однако даже только на ввод с последующей проверкой 320 наборов для 6 функций потребуется значительно больше времени, чем на ручное решение. Эффективность данного алгоритма выше всех других, известных автору.

В соответствии с алгоритмом 2 в 1974г была написана программа, которая позволяла минимизировать булевы функции от 36 переменных, определённые на 2000 наборах. Программа осуществляла контроль правильности ввода исходных массивов. Если функция введена неверно, то выводились на печать неправильно введённые РОК или ЗОК, а программа переходила к минимизации следующей функции. Время, затраченное ЦВМ М-220 на минимизацию булевой функции от 36 переменных, определённой на 1000 наборах, составило 6 минут. В 1985г. на языке Паскаль эта программа была переписана для ПЭВМ ДВК-2М. Она обрабатывала 16 функций от 32 переменных. Количество наборов достигало 2000.

Вопрос о минимизации булевых функций вручную или с использованием ПЭВМ решается в зависимости от количества наборов, на которых задана функция, количества соседних РОК и ЗОК, а также от частоты чередования РОК и ЗОК в исходной таблице истинности. Чем больше количество наборов, задающих функцию, чем меньше соседних РОК и ЗОК, чем выше частота чередования РОК и ЗОК, тем предпочтительнее использование ЭВМ. Например, систему из 7 булевых функций от 18 переменных, заданную на 80 наборах, оказалось рациональнее решать с помощью ЭВМ, так как в этой системе не нашлось ни одной соседней пары РОК и ЗОК, а частота чередования РОК и ЗОК для отдельных функций достигала 40. Однако за 25 лет инженерной практики раз-

работки цифровых устройств и систем автор лишь трижды обращался к услугам ЭВМ при решении задач минимизации булевых функций.

Задание 4

Методом обобщённых кодов найти минимальное представление функций, заданных на рабочих и запрещённых наборах.

4-1) РН(4): 0, 4, 6, 10; ЗН(4): 7, 13. Ответ: $K_c = 1$
4-2) РН(5): 4, 2, 29, 23; ЗН(5): 3, 21. Ответ: $K_c = 7 = (1+1+2)+3$
4-3) РН(6): 0, 9, 10, 13, 57, 63, 36; ЗН(6): 27, 29, 18, 44, 33.
Ответ: $K_c = 9 = (2+2+2)+3$
4-4) РН(6): 1, 4, 14, 21, 35, 62; ЗН(6): 3, 7, 30, 9.
Ответ: $K_c = 8 = (2+2+1)+3$
4-5) РН(8): 16, 49, 35, 41, 253, 167, 158; ЗН(8): 99, 125, 90, 249, 1
Ответ: $K_c = 9 = (2+2+2)+3$

Примечание: K_c - коэффициент сложности булевой функции.

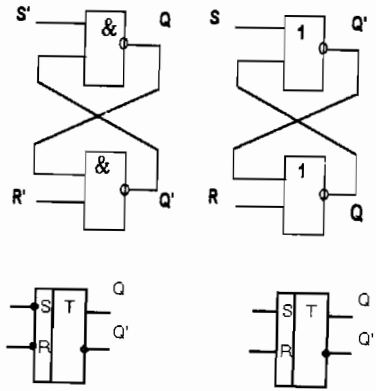
Глава 3
ТРИГГЕРЫ

Триггером называется логическое устройство с двумя устойчивыми состояниями. Выходной сигнал на выходе такого устройства зависит не только от входных сигналов действующих в данный момент, но и от сигналов, воздействовавших в предыдущий момент времени. Существует несколько видов триггеров.

Наиболее распространёнными из них являются SR-, D-, и JK-триггеры.

3.1. SR- триггеры

SR- триггер имеет 2 управляющих входа S и R. Если S = 1, то выход триггера Q = 1, если R = 1, то Q = 0. Комбинация S = 1 и R = 1 для SR- триггера является запрещённой. На рисунках представлены две реализации SR- триггера и его условное обозначение.



Схемы и обозначения SR-триггеров

3.2. D- триггеры

D- триггер при наличии разрешающего сигнала на тактовом входе устанавливается в состояние, соответствующее потенциалу на входе D.

Таким образом , D- триггер является тактируемым элементом памяти. Тактирование может осуществляться как потенциалом, так и фронтом (передним или задним).

Триггер, тактируемый потенциалом, может изменить своё состояние только при наличии определённого потенциала на тактовом входе С. Триггер, тактируемый фронтом, меняет своё состояние только с приходом определённого фронта сигнала на тактовый вход С.

На рисунках представлены тактируемые потенциалом и передним фронтом D- триггеры и их условные обозначения, а также временные диаграммы работы.

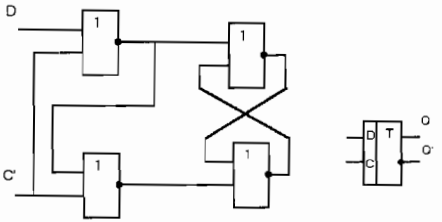
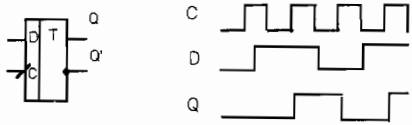


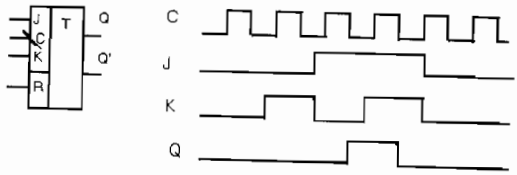
Схема и условное обозначение тактируемого потенциалом D-триггера.



Условное обозначение и диаграммы работы переднефронтового D-триггера.

3.3. JK- триггеры

JK- триггеры являются самыми сложными из рассмотренных элементов памяти. Они обладают большими функциональными возможностями. JK- триггеры могут быть тактируемыми и нетактируемыми. Универсальный JK- триггер может работать во всех режимах. Например, для перевода универсального JK- триггера в нетактируемый режим достаточно на вход С подать потенциал С = 1. На рисунке изображён тактируемый задним фронтом JK-триггер серии134 (1ТК343) и временные диаграммы его работы.



Условное обозначение и диаграммы работы заднефронтового JK-триггера.

Работает JK- триггер следующим образом. При J = 1 и K = 1 тактовый импульс переводит триггер в противоположное состояние ($Q_n = Q_{n-1}'$). При J = 0 и K = 0 триггер сохраняет предыдущее состояние ($Q_n = Q_{n-1}$). При J = 1 и K = 0 Q_n = 1, при J = 0 и K = 1 Q_n = 0.

Работа любого триггера может быть однозначно описана с помощью таблицы переходов.

Таблица переходов для SR-, D- и JK- триггеров.

Вход 1	Вход 2	SR- триггер	JK- триггер	D- триггер
0	0	Q_{n-1}	Q_{n-1}	0
0	1	0	0	0
1	0	1	1	1
1	1	-	Q_{n-1}'	1

3.4. Анализ работы схем с памятью

Анализ работы схем с памятью может быть графическим или табличным. Наиболее удобным является табличный метод анализа, так как он позволяет оторваться от принципиальной схемы устройства. Этот способ основан на описании функций управляющих входов, или функций возбуждения.

Задача 12

Произвести анализ работы устройства, изображённого на рисунке, табличным способом.

Решение

Опишем функции возбуждения входов SR- триггера.

$S = E_1(E_1E_2)' = E_1(E_1' + E_2') = E_1E_2'$

$R = E_2(E_1' + E_2') = E_1'E_2$

Зададим исходное состояние SR- триггера. Пусть $Q_0 = 0$. Далее будем изменять входные сигналы E_1 и E_2, определять функции S и R, а по ним находить Q_n. Процесс анализа отражён в таблице.

E_1 E_2	$S=E_1E_2'$	$R=E_1'E_2$	Q_n	Примечание
0 0	0	0	0	$Q_1=Q_0$, т.е. $Q_n=Q_{n-1}$
0 1	0	1	0	$Q_2=0$
1 0	1	0	1	$Q_3=1$
1 1	0	0	1	$Q_4=Q_3$, т.е. $Q_n=Q_{n-1}$

Даже простой пример показывает, насколько неудобен анализ работы схем с памятью. В главе «Конечные автоматы» мы познакомимся с более эффективным методом.

Глава 4
РЕГИСТРЫ И СЧЁТЧИКИ

Наиболее часто в схемах с памятью используются такие функциональные узлы, как регистры и счётчики. Регистры разделяются на два типа: параллельные (регистры памяти) и последовательные (регистры сдвига). Регистры могут быть реализованы на любых триггерах.

4.1 Регистры памяти

Эти регистры предназначены для хранения информации, представленной в виде двоичного кода. Регистры памяти осуществляют параллельную запись всего кода с приходом тактового импульса. Записанная информация хранится до прихода следующего тактового импульса. На рисунке представлен один из вариантов регистра памяти, реализованного на D- триггерах, тактируемых задним фронтом. Этот регистр предназначен для записи и хранения 4-разрядного кода X.

С приходом заднего фронта тактовой частоты $f_т$ регистр запишет код X и будет хранить его до тех пор, пока код X не изменит своей величины. Но это изменение кода X пройдёт на выход регистра только с приходом заднего фронта $f_т$.

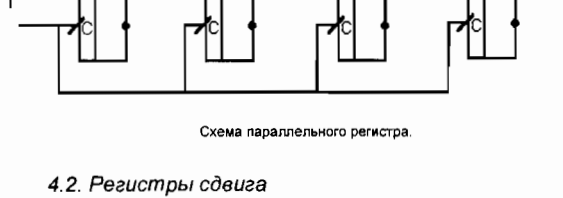


Схема параллельного регистра.

4.2. Регистры сдвига

Сдвиговые регистры широко используются в цифровой технике, в частности для преобразования последовательного двоич-

ного кода в параллельный или наоборот. На рисунке изображён сдвиговый регистр, реализованный на JK- триггерах.

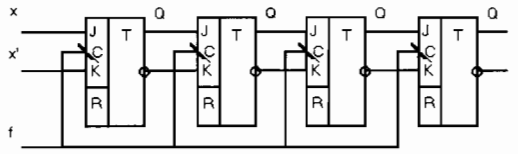


Схема регистра сдвига.

Благодаря последовательной схеме соединения разрядов регистра каждый задний фронт $f_т$ устанавливает последующий триггер в состояние, в котором до этого находился предыдущий, осуществляя сдвиг информации на разряд вправо.

На следующем рисунке представлена ИС типа 533ИР1, которая реализует функции как 4-х разрядного регистра памяти, так и функции регистра сдвига в зависимости от значения сигнала на входе выбора режима V_2:

при V_2=0 – режим сдвига,

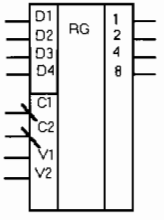
при V_2=1 – режим параллельной записи.

На рисунке:

C_1 – тактовый вход продвижения информации, поступающей по входу V_1.

C_2 – тактовый вход параллельной записи.

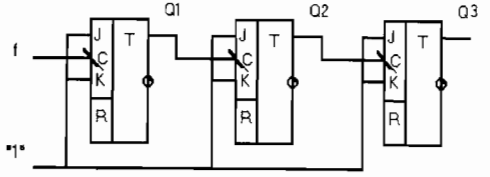
Если смена режима для 533ИР1 происходит в процессе работы, то для обеспечения безошибочной работы регистра необходимо выполнить условие $C_1'C_2'$=1.

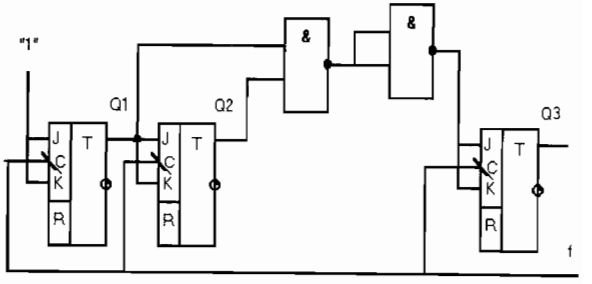


Условное обозначение универсального регистра.

4.3. Двоичные счётчики

Двоичные счётчики могут быть синхронными и асинхронными. Двоичный асинхронный счётчик может быть построен последовательным соединением счётных триггеров, или JK-триггеров, у которых J = K = 1.



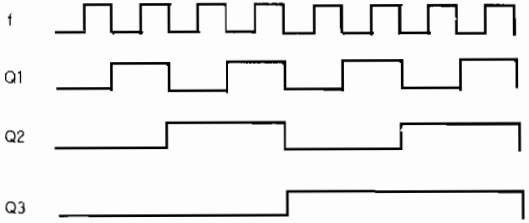


Асинхронный и синхронный двоичные счётчики

Недостатком асинхронных счётчиков является задержка в установлении соответствующего кода после прихода счётного импульса. Эта задержка, в частности, может приводить к появлению ложных кодов на выходе счётчика и сбою дешифратора. Например, после кода 011 на выходе счётчика должен появиться код 100 (коды записаны в порядке $Q_3Q_2Q_1$). В асинхронном счётчике при последовательном срабатывании триггеров код будет

меняться следующим образом: 011-010-000-100, т.е. во время переходного процесса появляются два ложных кода: 010 и 000.

В двоичных синхронных счётчиках срабатывание триггеров происходит одновременно или почти одновременно, так как все тактовые входы триггеров запараллелены. Временная диаграмма двоичного счётчика представлена на рисунке.



Диаграммы работы двоичного счётчика.

Для построения синхронных счётчиков с коэффициентом деления $K=2^n$ используется n JK- триггеров, функции возбуждения которых определяются соотношением:

$$J_m=K_m=\&_{i=1}^{m-1} Q_i ,$$

где J_m, K_m - соответствующие входы JK- триггера, на котором реализован m-ый разряд счётчика.

4.4. Десятичные счётчики

Десятичные счётчики строят на основе четырёхразрядных двоичных. Для понижения коэффициента пересчёта четырехразрядного счётчика с 16 до 10 вводят различные логические связи. В зависимости от вида логической связи одним и тем же десятичным числам в различных счётчиках могут соответствовать различные четырёхразрядные двоичные коды. В этом случае говорят, что счётчики работают в различных двоично-десятичных кодах.

Наиболее часто в счётчиках используется взвешенное кодирование. Если любое десятичное число A может быть выражено в виде суммы:

$$A = \sum_{i=0}^{n} a_i Q_i ,$$

где Q_i – двоичные числа (0 или 1) в соответствующих разрядах кода,

a_i – некоторые постоянные числа (веса разрядов),

то кодирование, построенное на данном соотношении, называется взвешенным.

Чаще других употребляется двоично-десятичный код 8-4-2-1. Здесь и далее цифры 8, 4, 2, 1 обозначают веса разрядов двоично-десятичного кода. Преимущество кода в его однозначности. Другие же коды неоднозначны. Например, в коде 4-2-2-1 десятичное число 4 можно представить как 1000 или 0110, поэтому существуют таблицы, однозначно закрепляющие за каждым кодом соответствующее десятичное число [5].

В таблице приводятся наиболее употребительные двоично-десятичные коды. Код с избытком 3 является самодополняющимся. Для этих кодов характерно то, что при их поразрядном инвертировании получается код числа, дополняющее данное до девяти. Это свойство кода удобно при построении цифровых приборов, измеряющих знакопеременные величины.

$Q_4Q_3Q_2Q_1$	код 8-4-2-1	Невзвешенный код	Код с избытком 3
0 0 0 0	0	0	-
0 0 0 1	1	1	-
0 0 1 0	2	2	-
0 0 1 1	3	3	0
0 1 0 0	4	-	1
0 1 0 1	5	-	2
0 1 1 0	6	4	3
0 1 1 1	7	5	4
1 0 0 0	8	-	5
1 0 0 1	9	-	6
1 0 1 0	-	-	7
1 0 1 1	-	-	8
1 1 0 0	-	8	9
1 1 0 1	-	9	-
1 1 1 0	-	6	-
1 1 1 1	-	7	-

4.5. Элементная база для построения счётчиков

В настоящее время промышленность выпускает ИС, реализующие асинхронные двоично-десятичные счётчики (533ИЕ2), счётчики-делители на 12 (533ИЕ4) и делитель на 16 (533ИЕ5). 533ИЕ2 имеет коэффициент деления К = 2 х 5, для 533ИЕ4 К = 2 х 6, для 533ИЕ5 К = 2 х 8.

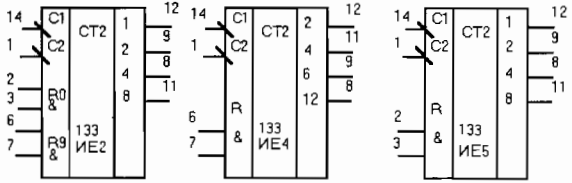
На рисунке в поле изображения микросхемы использованы следующие обозначения:

C1, C2 – тактовые входы соответственно для 1-го и 2-го разрядов;

R_0 – вход обнуления;

R_9 – вход установки счётчика в состояние 1001.

Для 533ИЕ4 выходы с весами 2, 4, 6, 12 обозначают выходы соответственно 1-го, 2-го, 3-го и 4-го разрядов, для счётчиков 533ИЕ2 и 533ИЕ5 выходы с весами 1, 2, 4, 8 обозначают выходы соответственно 1го, 2-го, 3-го и 4-го разрядов. Двоичный и десятичный счётчики изменяют свои состояния в соответствии с натуральной последовательностью кодов. Счётчик 533ИЕ4 (133ИЕ4) изменяет своё состояние в соответствии со следующей последовательностью десятичных кодов: 0-1-2-3-4-5-8-9-10-11-12-13. Это связано с тем, что счётчик построен по схеме соединения делителей с коэффициентами деления 2 х 3 х 2, вместо 2 х 2 х 3. Такую архитектуру можно объяснить лишь безграмотностью разработчиков.



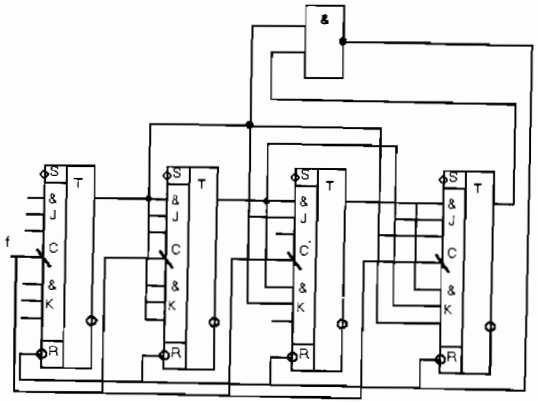
Условные обозначения ИС асинхронных счётчиков 133 серии

Глава 5
СИНТЕЗ СЧЁТЧИКОВ

Существует несколько методов синтеза счётчиков. Рассмотрим два из них.

5.1. Синтез счётчиков с использованием установочных входов

Суть этого метода заключается в том, что для построения счётчика с коэффициентом деления К используется n-разрядный двоичный счётчик ($n=\lceil \log_2 K \rceil$), охваченный обратной связью, которая формируется с помощью сборки двоичного кода К и подаётся на установочные входы обнуления. Например, для построения счётчика с К=10 необходимо использовать четырёхразрядный двоичный счётчик, а на установочные R-входы подать набор (сборку) - $Q_4Q_3'Q_2Q_1'$. Реализация этого счётчика изображена на рисунке.



Синхронный двоичный счётчик.

На схеме этого рисунка и в дальнейшем на все незадействованные входы микросхем подавать постоянный потенцивл логической 1.

Реализация асинхронного счётчика с К=10 на базе ИС 133ИЕ5 представлена на следующем рисункв. В этой схеме кроме сборки используется так называемый триггер-защёлка. Рекомендуется использовать его для надёжного обнуления и для повышения помехозащищённости счётчика. В силу того, что при данном методе синтеза используется лишь часть состояний двоичного счётчика, сборка может быть отминимизирована. Например, для счётчика с К=10 сборка соответствует Q_4Q_2, для счётчика с К=9 - Q_4Q_1, с К=12 - Q_4Q_3, с К=11 - $Q_4Q_2Q_1$.

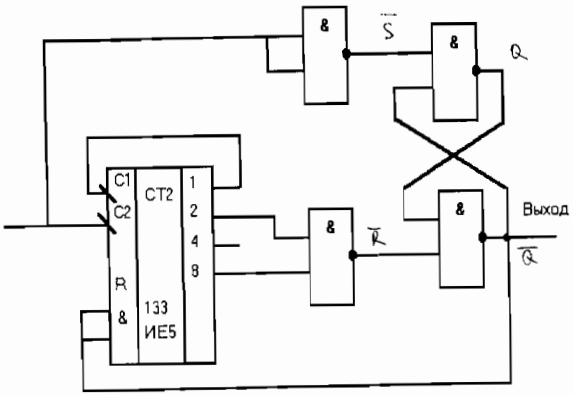


Схема десятичного счётчика на базе двоичного.

Синтез многоразрядных счётчиков на базе стандартных ИС имеет некоторую специфику. При использовании 533ИЕ2 (133ИЕ2) коэффициент деления представляется в двоично-десятичном коде, и на его основе формируются обратные связи. Например, для К=125 получаем 2/10 – код, равный 100100101. Это соответствует установочной функции $Q_1Q_3Q_6Q_9$. Для счётчика 533ИЕ4(133ИЕ4) необходимо представить коэффициент деления в 2/12 – коде с учётом специфики реализации архитектуры этой ИС. Синтез счётчика на базе ИС 533ИЕ5 (133ИЕ5) наиболее прост: достаточно

предстаеить коэффициент деления в двоичном коде, чтобы получить установочную функцию.

Выводы и рекомендации

Счётчики, реализованные с использованием установочных входов, обладают следующими недостатками:

1) низкая помехозащищённость по цепи обратной связи: если помеха вызовет сбой в этой цепи, то счётчик преждевременно обнулится;

2) при синхронной реализации из-за разброса параметров отдельных триггеров возможно появление ложных комбинаций, которые приведут к преждевременному обнулению счётчика (например, при переходе от кода 0111 к коду 1000 возможно кратковременное появление кода 1010, что приводит к обнулению счётчика с К=10).

Неоспоримым преимуществом таких счётчиков является их простота. Рекомендуется применение их при построении счётчиков с переменным К, а также при синтезе синхронизаторов без жёсткой привязки фронтов импульсных последовательностей.

Задание 5

а) Построить асинхронные делители частоты с использованием установочных входов:

5-1) К=12 на 533ИЕ5

5-2) К=8 на 533ИЕ2

5-3) К=80 на 533ИЕ2

б) Построить синхронные делители частоты с использованием установочных входов:

5-4) К=80 на JK-триггерах 533ТВ1

5-5) К=24 на JK-триггерах 134ТВ3.

5.2. Синтез счётчиков с использованием управляющих входов

Этот метод основан на использовании таблицы входов элементов памяти, которая может быть получена из таблицы переходов. По таблице входов можно определить, какие сигналы необходимо подать на управляющие входы, чтобы перевести элемент памяти из одного состояния в другое. Построим таблицу входов для JK-триггера.

Из таблицы переходов следует, что для перевода триггера из состояния 0 в состояние 0 на JK-входы необходимо подать JK=00 или JK=01, т.е. для осуществления этого перехода состояние

управляющего входа K безразлично. Условно операцию определения функций возбуждения для реализации перехода триггера из состояния 0 в состояние 0 можно записать так:

JK = 00 + 01 = 0-

Для перевода JK-триггера из состояния 0 в состояние 1 на JK-входы необходимо подать комбинацию

JK = 10 + 11 = 1- .

Для перевода JK-триггера из 1 в 0 должна быть подана комбинация

JK = 01 + 11 = -1 ,

а для перевода из 1 в 1

JK = 00 + 10 = -0 .

Аналогично можно получить таблицу входов для SR-триггера и D-триггера.

Таблица входов для JK -, SR - и D -триггеров.

Q_{n-1} Q_n	J K	S R	D
0 0	0 -	0 -	0
0 1	1 -	1 0	1
1 0	- 1	0 1	0
1 1	- 0	- 0	1

5.2.1. Синтез синхронных счётчиков

Этот метод прекрасно изложен в [11]. Для синтеза счётчика необходимо построить совмещённую таблицу состояний и входов счётчика. Проследим этот процесс на примере синтеза синхронного делителя на 5 (счётчик с K=5).

Совмещенная таблица счетчика с K = 5

N состояния	$Q_3Q_2Q_1$	J_3K_3	J_2K_2	J_1K_1
0	0 0 0	0 -	0 -	1 -
1	0 0 1	0 -	1 -	- 1
2	0 1 0	0 -	- 0	1 -
3	0 1 1	1 -	- 1	0 -
4	1 0 0	- 1	0 -	0 -

При переходе из 0-го состояния в 1-ое триггер 1-го разряда переходит из состояния 0 в состояние 1. Для осуществления этого перехода по таблице входов для JK-триггера определяем что должны быть поданы J_1=1, K_1=-, т.е. уровень сигнала K_1 не влияет

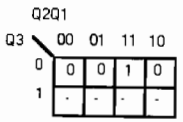
на выполнение перехода из 0 в 1. Для 2-го и 3-го разрядов переход из 0-го состояния в 1-ое требует перевода триггеров 2-го и 3-го разрядов из состояния 0 в состояние 0, что соответствует J_2K_2 = 0-, J_3K_3 = 0-. Аналогично заполняются строки таблицы для 1-го, 2-го и 3-го состояний.

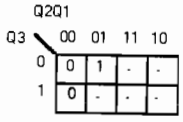
Для перехода из 4-го состояния в начальное (нулевое) необходимо, чтобы 1-й и 2-й триггеры сохранили на своих выходах 0, а 3-й перешёл из состояния 1 в состояние 0. Для этого необходимо обеспечить $J_1K_1 = J_2K_2$ = 0-, а J_3K_3 = -1.

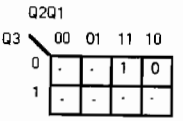
Далее совмещённая таблица интерпретируется как таблица истинности для булевых функций $J_1, K_1, J_2, K_2, J_3, K_3$ зависящих от входных переменных $Q_3Q_2Q_1$, и синтез счётчика сводится к синтезу булевых функций возбуждения $J_1, K_1, J_2, K_2, J_3, K_3$.

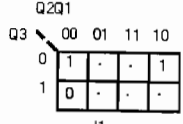
После минимизации функций возбуждения получаем результат:

$$J_3 = Q_2Q_1$$
$$J_2 = K_2 = Q_1$$
$$J_1 = Q_3'$$
$$K_3 = K_1 = 1$$

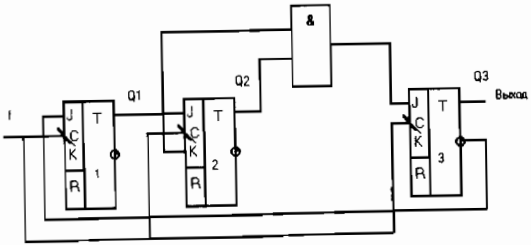








В соответствии с этим результатом строим счётчик. Счётчики можно строить на различных элементах памяти, при этом нужно использовать соответствующую таблицу входов.



Синхронный делитель на 5.

Задача 13

Построить счётчик с К = 3 на D-триггерах.

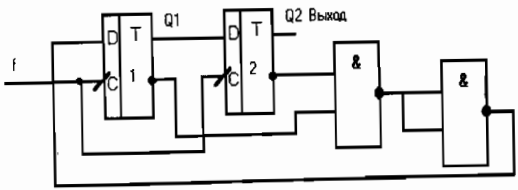
Решение

Строим соемещённую таблицу и синтезируем функции возбуждения D_2 и D_1.

Q_2Q_1	D_2	D_1
0 0	0	1
0 1	1	0
1 0	0	0

$D_2 = Q_1$ $D_1 = Q_2'Q_1'$

Схема счётчика представлена на рисунке.



Синхронный делитель на 3.

Весьма важным моментом является кодирование состояний счётчика. При оптимальном кодировании удаётся получить наиболее компактную реализацию счётчика. Вопросы оптимального кодирования состояний счётчика полностью не решены. Алгоритмы, предлагаемые в [2] для оптимального кодирования состояний микропрограммных автоматов, далеко не всегда дают положительные результаты.

Рассмотрим влияние кодирования состояний счётчика на примере синтеза делителя на 5.

Задача 14

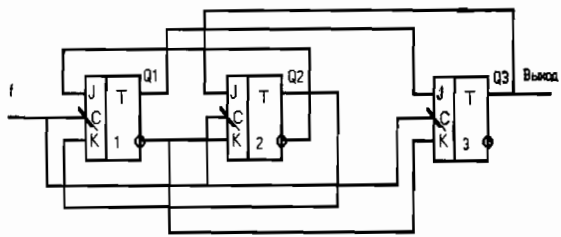
Построить счётчик с К = 5 при невзвешенном кодировании состояний на JK-триггерах.

Решение

Строим совмещённую таблицу и синтезируем функции возбуждения.

$Q_3Q_2Q_1$	J_3K_3	J_2K_2	J_1K_1
0 0 0	0 -	0 -	1 -
0 0 1	1 -	0 -	- 0
1 0 1	- 0	1 -	- 0
1 1 1	- 0	- 0	- 1
1 1 0	- 1	- 1	0 -

$K_3 = K_2 = Q_1'$
$J_2 = Q_3$; $J_3 = Q_1$
$J_1 = Q_2'$; $K_1 = Q_2$



Делитель на 5 при невзвешенном кодировании.

Из сравнения рисунков видно, что получившийся счётчик имеет оптимальную структуру. Можно построить, как минимум, ещё 3 различных варианта делителей на 5, имеющих минимальную реализацию. Одна из таких реализаций приводится в [11]. Рассмотренный делитель на 5 был синтезирован с применением так называемого соседнего кодирования, которое считается оптимальным для построения конечных автоматов на элементах памяти с раздельными входами, типа JK- или SR-триггеров [2]. Если для построения синхронного счётчика с К=12 применить соседнее кодирование 0000, 0001, 0011, 0010, 0110, 1110, 1010, 1011, 1001, 1000, 1100, 0100, то коэффициент сложности счётчика, построенного на JK-триггерах, составит 26, а при использовании взвешенного двоичного кодирования (0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011) коэффициент сложности схемы будет равен 8. Практика построения синхронных счётчиков с большими коэффициентами пересчёта показала, что наилучшие результаты даёт взвешенное двоичное кодирование состояний счётчика.

Методы синтеза счётчиков полностью пригодны для разработки схем различных синхронизаторов и распределителей импульсов, так как и те, и другие являются счётчиками с невзвешенным кодированием состояний.

Пример синтеза синхронизатора, используемого в цифровом фильтре, рассмотрен в задаче 11, где $Q_{10} - Q_{12}$ - выходы синхронизатора, а $Q_1 - Q_9$ - выходы вспомогательного счётчика с К=320. Функции возбуждения этого счётчика не представляют интереса, так как счётчик состоит из известных делителей частоты на 32 и на 10.

5.2.2. Синтез асинхронных счётчиков

Синтез асинхронных счётчиков [11] возможен при выполнении следующих условий:

1) если хотя бы один разряд обеспечивает формирование заднего фронта для всех тех случаев, когда другой разряд изменяет своё состояние;

2) задержка, вносимая самой длинной цепью делителя от тактового входа одного разряда до входа возбуждения другого, должна быть меньше периода тактовой частоты последнего из указанных разрядов.

Синтез асинхронных счётчиков разбивается на 4 этапа:

1) находится совокупность разрядов, в которой один из разрядов мог бы формировать тактовую частоту для других (этот разряд будем называть тактовым);

2) для тактируемых разрядов из этой совокупности совмещённая таблица для функций возбуждения заполняется только по строкам, соответствующим заднему фронту тактового разряда, а в остальных строках ставится прочерк;

3) для разрядов, на тактовый вход которых поступает основная частота f, синтез осуществляется по синхронному методу;

4) определяется граничная частота $f_{гр}$ для данной реализации.

Задача 15

Построить асинхронный делитель на 6.

Решение

Строим совмещённую таблицу.

$Q_3Q_2Q_1$	J_3K_3	J_2K_2	J_1K_1
0 0 0	- -	- -	1 -
0 0 1	0 -	1 -	- 1
0 1 0	- -	- -	1 -
0 1 1	1 -	- 1	- 1
1 0 0	- -	- -	1 -

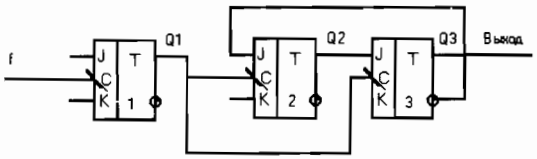
1. Разряды 1, 2, и 3 образуют совокупность, в которой 1-й разряд может формировать тактовую частоту для 2-го и 3-го.

2. Заполняем совмещённую таблицу в соответствии со 2-м и 3-м этапами синтеза асинхронных счётчиков.

3. Определяем функции возбуждения:

$J_3=Q_2$, $J_2=Q_3'$, $J_1=K_1=K_2=K_3=1$, по которым строим счётчик.

4. Определяем граничную частоту, считая, что счётчик реализован на триггерах 134ТВ3 с задержкой =100 нс. Наиболее длинная цепь от выхода Q_2 до выхода J_2. Задержка по этой цепи составляет 100 нс. Граничная тактовая частота для 2-го и 3-го разрядов равна 1/100 нс = 10 Мгц. Приведённая к тактовому входу 1-го разряда frp= 20 Мгц. Реальная граничная частота будет не более 1,5 Мгц, так как это частотный предел ИС серии 134.



Асинхронный делитель на 6

Выводы и рекомендации

1. Счётчики, реализованные с использованием управляющих входов, не застрахованы от появления ложных выходных наборов, но это никогда не приводит к сбою в работе счётчиков, так как изменение состояния счётчиков происходит при установившемся значении выходных наборов.

2. Синхронные счётчики указанного класса позволяют реализовать импульсныв последовательности с высокой точностью привязки фронтов.

3. Если высокая степвнь точности привязки импульсной последоватвльности нв требуется, то предпочтительнее асинхронная реализация счётчиков.

4. Как синхронные, так и асинхронные счётчики с использованием управляющих входов максимально защищены от сбоев по установочным входам.

Задание 6

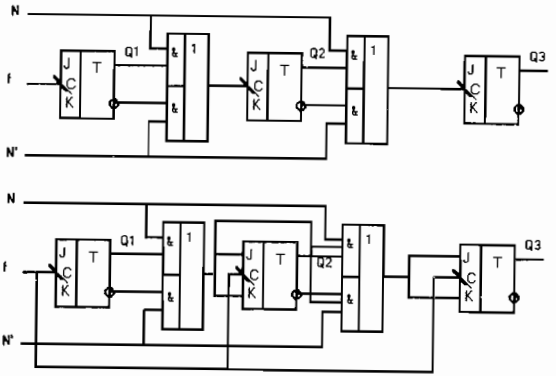
Построить синхронные и асинхронные делители частоты с использованием управляющих входов на JK - и D -триггерах:

6-1) К = 12
6-2) К = 14
6-3) К = 24
6-4) К = 25

5.3. Реверсивные счётчики

Реверсивные счётчики (РС) могут быть асинхронными и синхронными. И те, и другие могут быть как двоичными, так и с произвольным коэффициентом пересчёта К. На рисунке показана схема асинхронного двоичного РС без зоны нечувствительности,

т.е. такого РС, который с приходом каждого тактового импульса изменяет своё состояние.



Асинхронный и синхронный реверсивные счётчики.

Изменение направления счёта в РС происходит в зависимости от величины сигнала N. При N = 1 счётчик считает в прямом направлении, при N = 0 – в обратном.

Асинхронный РС практически не может быть использован, так как при изменении направления счёта может изменяться код, записанный в РС. Например, при переходе от суммирования к вычитанию, если предыдущий триггер находится в состоянии 1, то сигнал на входе последующего триггера изменится с 1 на 0, что приведёт к его опрокидыванию.

Этого недостатка лишены синхронные счётчики, так как они изменяют своё состояние по заднему фронту тактовой частоты, и изменение N в промежутке между фронтами (т.е. на протяжении периода тактовой частоты) не влияет на состояние РС. Пример синхронного двоичного РС приводится на рисунке.

Построение РС без зоны нечувствительности с произвольным К осуществляется методом синтеза счётчиков с использованием управляющих входов. Аналогично можно строить РС с зоной нечувствительности.

Задача 16

Построить синхронный РС без зоны нечувствительности с К=3 на JK-триггерах.

Решение

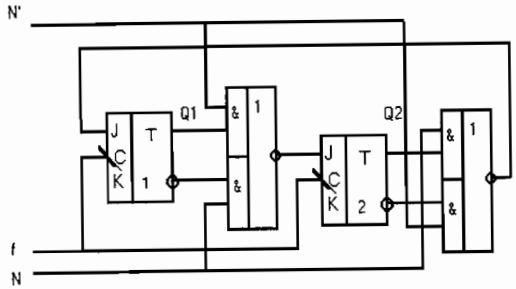
Строим совмещённую таблицу и синтезируем функции возбуждения J_2, K_2, J_1, K_1.

Код состоян.	N = 1		N = 0	
Q_2Q_1	J_2K_2	J_1K_1	J_2K_2	J_1K_1
0 0	0 1	1 -	1 -	0 -
0 1	1 -	- 1	0 -	- 1
1 0	- 1	0 -	- 1	1 -

С помощью карт Карно получаем:

$J_2 = Q_1'N' + Q_1N = (Q_1N'+Q_1'N)'$;
$J_1 = Q_2'N + Q_2N' = (Q_2N+Q_2'N')'$

Схема счётчика и синтез функций возбуждения представлены на рисунке.



Синхронный реверсивный счётчик с К=3
без зоны нечувствительности.

Задача 17

Построить синхронные РС с зоной нечувствительности на JK-триггерах. К = 3. При R_1R_2 = 10 РС должен считать в прямом направлении, при R_1R_2 = 01 – в обратном, при R_1R_2 = 00 РС не

изменяет своего состояния, ситуация R_1R_2 = 11 является запрещённой.

Решение

Строим совмещённую таблицу и синтезируем функции возбуждения.

Код сост.	R1R2 = 10		R_1R_2 = 01		R_1R_2 = 00	
Q_2Q_1	J_2K_2	J_1K_1	J_2K_2	J_1K_1	J_2K_2	J_1K_1
0 0	0 -	1 -	1 -	0 -	0 -	0 -
0 1	1 -	- 1	0 -	- 1	0 -	- 0
1 0	- 1	0 -	- 1	1 -	- 0	0 -

$J_2' = Q_1'R_2' + Q_1R_1'$ $\quad$ $J_1' = Q_2'R_1' + Q_2R_2'$
$K_2' = R_1{*}R_2$ $\quad$ $K_1' = R_1'R_2'$

В этой задаче минимизацию функций вюзбуждения удобнее проводить по нулям, т.е. получать минимальную форму инверсии функции возбуждения. В этом случае удаётся применить ИС типа 2И-2ИЛИ-НЕ, что уменьшает количество корпусов ИС и снижает число связей между элементами.

Задание 7

Построить РС без зоны нечувствительности и с зоной нечувствительности на JK-, D - и тактируемых SR- триггерах:

7-1) К = 10
7-2) К = 12
7-3) К =16

5.4. Распределители импульсов

В инженерной практике большое значение имеет синтез распределителей импульсов (РИ). Под распределителем импульсов понимается устройство, работа которого описывается периодической логической функцией вида

$Q_i = F_i(t)$, где Qi – i выход РИ, t – время.

Самым простым решением задачи будет построение РИ на основе тактового генератора (ТГ), счётчика и дешифратора. Однако полученный РИ имеет ряд недостатков. Одним из них является искажение выходной последовательности в результате возникновения гонок. Избавиться от них можно, введя дополнительный регистр на выходе дешифратора. Однако, в случае построения быстродействующих РИ требуется минимальное запаздывание, а

№состояния	$Q_0Q_1Q_2Q_3Q_4Q_5$	8-й код	D_0
1	0 1 1 0 0 0	30	1
2	1 0 1 1 0 0	54	1
3	1 1 0 1 1 0	66	0
4	0 1 1 0 1 1	33	0
5	0 0 1 1 0 1	15	0
6	0 0 0 1 1 0	06	1
7	1 0 0 0 1 1	43	1
8	1 1 0 0 0 1	61	1
9	1 1 1 0 0 0	70	1
10	1 1 1 1 0 0	74	0
11	0 1 1 1 1 0	36	0
12	0 0 1 1 1 1	17	0
13	0 0 0 1 1 1	07	0
14	0 0 0 0 1 1	03	0
15	0 0 0 0 0 1	01	1
16	1 0 0 0 0 0	40	1
17	1 1 0 0 0 0	60	0

Из таблицы истинности получаем:

$D_0 = Q_3'Q_4'Q_5 + Q_2'Q_0'Q_5' + Q_2Q_3'Q_4' + Q_0Q_1'$
$D_1 = Q_0$
$D_2 = Q_1$
$D_3 = Q_2$
$D_4 = Q_3$
$D_5 = Q_4$

Комбинационная часть ССдР получилась минимальной за счёт увеличения разрядности регистра на единицу. Рутинная операция развязывания состояний ССдР требует тем не менее внимания, поэтому её вместе с П-разбиением и формированием таблицы истинности имеет смысл реализовать в виде простенькой программы.

```
program ssr;
uses crt;
type vect=array[1..2048] of word;
var
    v0,v1,v :vect;
    i,j,k,n:integer;
{-------------------------------------------------------}
procedure pirazb(k:integer;var v0:vect);
{Пи-разбиение для младшего разряда счетчика
с коэффициентом деления k}
var
  i,j,ik,jk,l:integer;
  begin
    l:=1;ik:=1;jk:=2;
    repeat
      for i:=ik to ik+2*(l-1) do v0[i]:=0;
      for j:=jk to jk+2*l-1 do v0[j]:=1;
      ik:=jk+2*l;
      jk:=ik+2*l+1;
      l:=l+1;
    until (i>k) or (j>k);
  end;
{-------------------------------------------------------}
procedure sdwig(n:integer;var v0,v1:vect);
{Сдвиг исходного вектора вниз на один разряд}
var
  i,v0n:integer;
  begin
    for i:=n downto 2 do v1[i]:=v0[i-1];
    v1[1]:=v0[n];
  end;
{-------------------------------------------------------}
procedure kley(n:integer;var v0,v1,v:vect);
var
  i:integer;
  begin
    for i:=1 to n do v[i]:=v1[i]+2*v0[i];
  end;
{-------------------------------------------------------}
function razwyaz(k:integer;var v:vect):integer;
label 1;
var
  i,j:integer;
  begin
    razwyaz:=1;
    for i:=1 to k-1 do
      for j:=i+1 to k do
        if v[i] = v[j] then
          begin
            razwyaz:=0;
            goto 1;
          end;
1:end;
      {-------------------------------------}
procedure formtbl(k:integer;var v0,v:vect;var n:integer);
{Развязывание кодов счётчика.
k  - коэффициент пересчёта,
v0 - исходный вектор,
v1 - результирующий вектор кодов состояний счетчика,
n  - количество разрядов в коде состояний счетчика,
v1,v2 - промежуточные векторы,
f  - флаг развязки.
}
var
  f,i:integer;
  v1,v2:vect;
  begin
```

```
    n:=1;
    for i:=1 to k do v2[i]:=v0[i];
    repeat
      sdwig(k,v0,v1);
      kley(k,v2,v1,v);
      n:=n+1;
      f:=razwyaz(k,v);
      for i:=1 to k do
        begin
          v0[i]:=v1[i];
          v2[i]:=v[i];
        end;
    until (f=1) or (n>16);
  end;
{=============================================}
begin
  clrscr;
  writeln('r=============================================¬');
  writeln('|    Синтез счетчиков на сдвиговых          |');
  writeln('|    регистрах.Коэффициент деления <= 2048   |');
  writeln('|                 SSR.pas                    |');
  writeln('|        Лобанов В.И.   17-03-1999           |');
  writeln('L=============================================-');
  writeln;
  write('Введите k<=2048    ');
  readln(k);
  writeln;
  pirazb(k,v0);
  for i:=1 to k do
    write(v0[i]:2);
  writeln;
formtbl(k,v0,v,n);
  for i:=1 to k do write(v[i]:8);
  writeln;
  if n>16 then writeln ('Переполнение разрядной сетки');
  writeln('n = ',n:2);
  writeln('Нажмите Enter ');
  readln;
 end.
```

Синтез счетчиков на сдвиговых регистрах (ССР) по программе `ssr.pas` дал такие результаты:

Коэффициент деления:	10	20	30	40	50	60
Разрядность ССР:	4	8	10	12	14	16

Результаты работы этой программы убедительно доказывают неэффективность синтеза РИ на сдвиговых регистрах: слишком велик объём памяти.

Глава 6
КОНЕЧНЫЕ АВТОМАТЫ

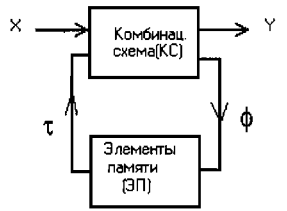
6.1. Понятие о конечном автомате. Автоматы Мили и Мура

Конечным автоматом называется устройство, задаваемое конечным множеством из элементов:

1) множество входных сигналов (входной алфавит) -
$x = \{ x_1, \ldots, x_f, \ldots x \}$;
2) множество состояний (алфавит состояний) -
$A = \{ a_0, \ldots, a_m, \ldots a_M \}$;
3) множество выходных сигналов (выходной алфавит) -
$y = \{ y_1, \ldots, y_g, \ldots y_G \}$;
4) функция первходов - δ ,
5) функция выходов - λ ;
6) начальное состояние автомата - a_0.

Под состоянием понимается одна из множества отличных друг от друга ситуаций, в которой может оказаться конечный автомат. Например, счётчик с К=5 имеет 5 различных состояний, реверсивный счётчик с К=10 имеет 10 различных состояний.

В общем виде структура конечного автомата представлена на рисункв. Из него видно, что конечный автомат состоит из комбинационной схемы КС, которая формирует выходной алфавит y и функции возбуждения φ, и элементов памяти ЭП с выходами τ.



Обобщённая схема конечного автомата.

Функции возбуждения описывают те сигналы, которые необходимо подать на управляющие входы элементов памяти.

На практике наибольшее распространение получили автоматы Мили и Мура.

Автомат Мили задаётся уравнениями:

$a(t+1) = \delta(a(t), x(t))$;

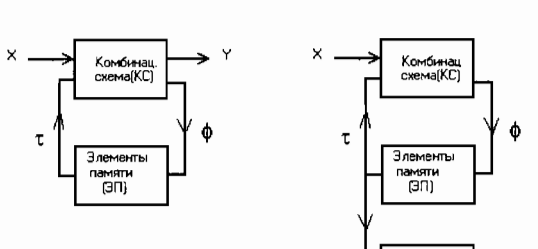
$y(t) = \lambda(a(t), x(t))$;

Автомат Мура описывается соотношениями:

$a(t+1) = \delta(a(t), x(t))$;

$y(t) = \lambda(a(t))$.

Из законов функционирования автоматов Мили и Мура видно, что они отличаются только формированием выходного алфавита. Структурные схемы автоматов Мили и Мура представлены ниже.



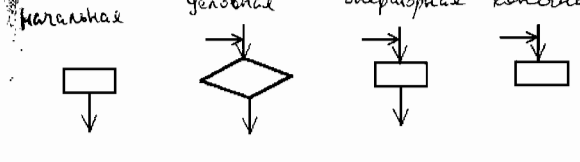
Схемы автоматов Мили и Мура.

6.2. Методы задания автоматов. ГСА

Любой конечный автомат может быть представлен с помощью направленных графов, таблиц переходов и выходов, граф-схем алгоритмов (ГСА), логических схем алгоритмов (ЛСА) или матричных схем алгоритмов (МСА). Этот перечень можно было бы продолжить.

Наиболее наглядной формой представления автомата является ГСА [2].

ГСА - ориентированный связный граф, содержащий вершины четырёх типов: начальную, условную, операторную и конечную.



Условные обозначения вершин ГСА

ГСА удовлетворяет следующим требованиям:

1. Содержит конечное число вершин.
2. Имеет одну начальную и одну конечную вершины.
3. Входы и выходы вершин соединяются дугами, направленными от выхода ко входу.
4. Каждый выход соединён только с одним входом.
5. Любой вход соединяется по крайней мере с одним выходом.
6. Для любой вершины графа существует хотя бы один путь к конечной вершине.
7. В каждой условной вершине записывается один из элементов множества логических условий (входной алфавит).

$x = \{ x_1, \ldots, x_f, \ldots, x_F \}$.

8. В каждой операторной вершине записыеается оператор y_t. подмножество множества выходных сигналов y.

$y = \{ y_1, \ldots, y_g, \ldots, y_G \}$. Допускается $y_t = \varnothing$.

6.3. Синтез конечных автоматов

Различают два метода синтеза: эвристический и формальный. Эвристический метод – метод проб и ошибок, но зачастую он приводит к весьма экономным решениям. Единственный его недостаток заключается в том, что для получения решения эвристическим методом требуется длительное время работы высококвалифицированных специалистов. При таком подходе в век НТР достаточно сложное устройство может морально устареть прежде, чвм закончится его эвристическая разработка.

При формальном методе синтеза решение задачи получается чрезвычайно быстро, сроки разработок существенно сокра-

щаются, а саму формальную разработку можно переложить либо на ЭВМ, либо на технический персонал.

Не следует считать, что формальный метод исключает творческий этап в разработке цифрового устройства. От того, насколько остроумна была идея, т.е. алгоритм устройства, целиком и полностью зависит сложность схемы прибора, изделия. Поэтому правильнее считать формальный метод формально – эвристическим.

Обычно синтез конечных автоматов (цифровых устройств, микропрограммных автоматов) осуществляется по следующей схеме.

1. Проработка идеи реализации конечного автомата, выявление отдельных узлов и связей между ними. Составление структурной схемы автомата.

2. Составление ГСА, разметка ГСА.

3. Составление по ГСА обратной структурной таблицы.

4. Кодирование состояний. Для асинхронного автомата применяется противогоночное кодирование, для синхронного применяется кодирование, обеспечивающее минимизацию комбинационной части устройства.

5. Построение принципиальной схемы по обратной структурной таблице автомата.

Кстати, именно этот формальный метод использует автор уже более четверти века как для разработки конечных автоматов на базе цифровых интегральных микросхем, так и, введя в него некоторые дополнения, для синтеза сложных релейных схем. Возврат к релейным схемам был связан с высокой помехоустойчивостью и радиационной стойкостью реле. Особенно это стало актуальным после Чернобыля.

Каждый из 5 пунктов требует обстоятельного разговора. Попытавмся понять суть этих вопросов на примере построения достаточно простых конечных автоматов.

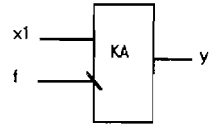
Задача 18

Очистить от дребезга входной сигнал x, подача которого осуществляется тумблером.

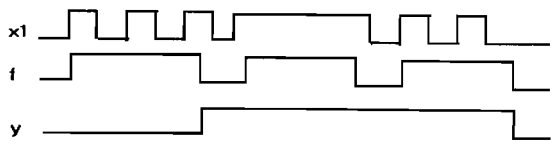
Решение

1. Очистку от дребезга можно осуществить, например, с помощью синхронного конечного автомата (КА). Задача КА будет заключаться в том, чтобы по заднему фронту тактовой частоты f выходной сигнал y принимал то же значение, что и входной сигнал x. В этой ситуации выходной сигнал не изменится на протяжении всего периода тактовой частоты, как бы не менялся при этом x_1.

Для того чтобы дребезг входного сигнала не проявлялся на выходе, необходимо иметь тактовую частоту с периодом, превышающим длительность дребезга. Так как дребезг тумблера длится не более 2 мс, то период тактовой частоты должен быть не менее 2 мс. Структурная схема автомата получилась простой.



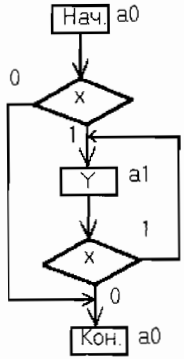
Идею обработки входного сигнала x и выдачи очищенного от дребезга выходного сигнала y можно выразить с помощью временной диаграммы.



Из временной диаграммы видно, что при первом же совпадении заднего фронта с x = 1 автомат должен выдать y = 1, а при совпадении заднего фронта с x = 0 автомат должен выдать y = 0.

2. Опишем логику работы автомата, т.е. зададим автомат, с помощью ГСА. Разметим ГСА для автомата Мура, для чего рядом с каждой операторной вершиной проставим идентификатор состояний, причём начальная и конечная вершины ГСА отмечаются как одинаковые состояния. В начальной стадии КА находится в начальном состоянии a_0 (начальная вершина) и проверяет наличие сигнала x. Если x=0, то КА не меняет своего состояния, что отображено связью x=0, выходящей из первой условной вершины и входящей в конечную вершину, которая отмечена начальным состоянием a_0. В начальной и конечной вершинах нет ни одного оператора, т.е. y=0. Если x=1, то КА оказывается в новом состоянии a_1 по связи x=1, выходящей из первой условной вершины и входящей в операторную вершину с оператором y. Если после того, как КА оказался в состоянии a_1, входной сигнал x к моменту прихода зад-

него фронта тактовой частоты не изменил своего значения, т.е. x=1, то КА останется в состоянии a_1 и будет выдавать на своём выходе сигнал y=1 (связь из второй условной вершины). В том случае, если приход заднего фронта тактовой частоты совпадает с x=0, то КА вернётся в начальное состояние a_0. Таким образом, составление, разметка и проверка ГСА работы КА закончены, так как мы убедились, что выходной сигнал y по заднему фронту тактовой частоты принимает то же значение, что и входной сигнал x.



3. Составим для конечного автомата обратную структурную таблицу. Обратная структурная таблица показывает, при каких входных сигналах и из каких состояний осуществляется переход в данное состояние и какие выходные сигналы выдаются при этом переходе. В столбце состояние перехода (СП) для автомата Мура указывается не только состояние перехода, но и выходной сигнал, который соответствует данному состоянию.

Исходное состояние	Код исх. сост	Входной сигнал	Состояние перехода	Код СП	Функция возб.
(ИС)	τ	x	(СП)	(КСП)	J K
a_0	0	0	a_0	0	0 -
a_1	1	0	(-)	0	- 1
a_0	0	1	a_1	1	1 -
a_1	1	1	(y)	1	- 0

4. Так как синтезируемый конечный автомат имеет только 2 состояния, то его можно построить на одном элементе памяти.

n =]log$_2$N[,

где n - количество элементов памяти, необходимое для реализации КА, имеющего N состояний.

n =] log$_2$2[= 1

Таким образом, коды состояний будут одноразрядными. Так как автомат очень простой, то кодирование может быть произвольным. Кодирование состояний приведено в таблице 29. Выбираем в качестве элемента памяти JK-триггер.

Столбцы для функций возбуждения заполняются в соответствии с таблицей входов конкретного элемента памяти (в данном случае JK-триггера) для обеспечения перехода из исходного состояния в состояние перехода.

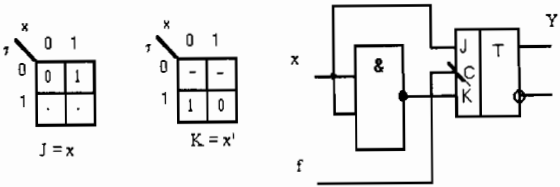
Функции возбуждения синтезируются как логические функции, зависящие от кода исходного состояния и входного сигнала.

J = x, K = x'

Выходные функции зависят только от кода состояния перехода (для автомата Мура). Поэтому y = τ, т.е. выходной сигнал формируется на прямом выходе JK-триггера. Синтез функций возбуждения и принципиальная схема конечного автомата приводятся на рисунке.

Задачу очистки от дребезга можно было решать на любом элементе памяти. В частности для D-триггера мы получили бы более компактное решение, а именно:

D = x, y = τ



Автомат очистки от дребезга

6.4. Кодирование состояний и сложность комбинационной схемы

При синтезе счётчика с К=5 мы видели, что сложность комбинационной части счётчика зависит от кодирования состояний. Это утверждение справедливо для всего класса конечных автоматоа.

В [2] приводятся два простых алгоритма оптимального кодирования состояний конечного автомата. Эти алгоритмы хорошо работают при синтезе автоматов, насчитывающих от нескольких десятков до нескольких сотен функций возбуждения и несколько сотен переменных, когда классические методы минимизации булевых функций оказываются неприемлемыми.

Алгоритм оптимального кодирования состояний автомата на D-триггерах.

1. Каждому состоянию автомата Q_m (m=1,..., M) ставится в соответствие целое число N_m, равное числу переходов в состояние a_m.
2. Числа $N_1,..., N_m,...,N_M$ сортируются по убыванию
3. Состояние Q_t с максимальным N_t кодируется кодом (00....0).
4. Следующие I состояний (I - число элементов памяти) в упорядоченном в п.2 списке кодируется кодами, в которых используется одна единица в каком-либо разряде, а во всех остальных разрядах нули.
5. Для кодирования следующих I состояний используются коды, содержащие две единицы, затем три единицы и т.д., до тех пор, пока все состояния не будут закодированы.

Суть этого алгоритма заключается в том, что мы стремимся использовать минимальное количество единиц в кодах тех состояний, переход в которые происходит чаще всего. Таким образом, функции возбуждения реже принимают значение 1, что приводит к уменьшению количества термов. Так как классические методы минимизации при синтезе сложных автоматов не применяются, то по алгоритму кодирования мы получаем коды состояний, обеспечивающие минимальную комбинационную часть конечного автомата.

Для конечных автоматов, построенных на триггерах с раздельными входами, применяется алгоритм кодирования состояний, минимизирующий суммарное число изменений состояний элементов памяти на всех переходах автомата [2]. Соседнее кодирование является частным случаем кодирования состояний, получаемым по указанному алгоритму.

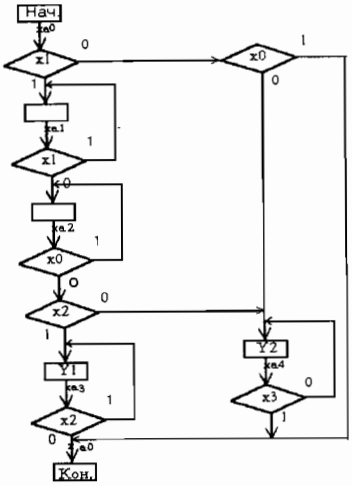
Задача 20

Построить 10-разрядный кодовый замок, который открывается последовательным нажатием двух определённых клавиш. Предусмотреть защиту от посторонних. Замок должен открываться кодом 19.

Решение

Построим замок, используя автомат Мили. ГСА кодового замка и её разметка представлены на рисунке. В этой ГСА использованы следующие обозначения:

x_0 - не нажата ни одна из клавиш
x_1 - нажата только клавиша №1
x_2 - нажата только клавиша №9
x_3 - снятие сигнализации
y_1 - включить соленоид
y_2 - включить сирену



ГСА кодового замка.

По ГСА строим обратную структурную таблицу.

ИС	$\tau_3\tau_2\tau_1$	Входы	СП	КСП	Вых.	J_3K_3	J_2K_2	J_1K_1
a_0	0 0 0	x_0x_1'	a_0	0 0 0	-	0 -	0 -	0 -
a_3	0 1 0	x_2'		0 0 0	-	0 -	- 1	0 -
a_4	1 0 0	x_1		0 0 0	-	0 1	0 -	0 -
a_0	0 0 0	x_1	a_1	0 0 1	-	0 -	0 -	1 -
a_1	0 0 1	x_1		0 0 1	-	0 -	0 -	- 0
a_1	0 0 1	x_1'	a_2	0 1 1	-	0 -	1 -	- 0
a_2	0 1 1	x_0		0 1 1	-	0 -	- 0	- 0
a_2	0 1 1	x_2x_0'	a_3	0 1 0	y_1	0 -	- 0	- 1
a_3	0 1 0	x_2		0 1 0	y_1	0 -	- 0	o -
a_0	0 0 0	$x_1'x_0'$	a_4	1 0 0	y_2	1 -	0 -	0 -
a_2	0 1 1	$x_0'x_2'$		1 0 0	y_2	1 -	- 1	- 1
a_4	1 0 0	x_3'		1 0 0	y_2	- 0	0 -	0 -

После применения алгоритма оптимального кодироеания для триггеров с раздельными входами получаем коды:

a_0 – 000
a_1 – 001
a_2 – 011
a_3 – 010
a_4 – 100

После минимизации получим функции возбуждения:

$J_3 = \tau_2'\tau_1'x_1'x_0' + \tau_2\tau_1x_2'x_0'$;
$K_3 = x_3$;
$J_2 = \tau_1x_1'$;
$K_2 = \tau_1'x_2' + x_2'x_0'$;
$J_1 = \tau_3'\tau_2'x_1$;
$K_1 = \tau_2x_0'$

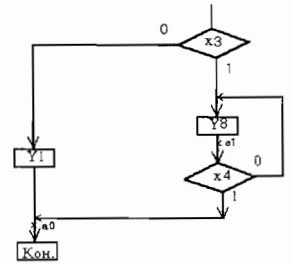
Выходные функции для автомата Мили:

$y_1 = \tau_2x_2x_0' + \tau_2\tau_1'x_2$;
$y_2 = \tau_3'\tau_2'\tau_1'x_1'x_0' + \tau_3\tau_1x_2'x_0' + \tau_3x_3'$;

Если бы кодовый замок был построен на автомате Мура, то выходные функции приняли бы вид:

$y_1 = \tau_2\tau_1'$ $\qquad$ $y_2 = \tau_3$

Далеко не всегда автомат Мура даёт более компактные решения, чем автомат Мили. Например, если часть ГСА имеет еид, представленный на нижеприведённом рисунке, то реализация этой ГСА с помощью автомата Мили позволит уменьшить число состояний, а значит, и количество элементое памяти.



6.5. Гонки и противогоночное кодирование

При работе автомата могут появиться гонки, вызванные разбросом параметров элементов памяти. Задержки, вносимые триггерами, имеют различные значения, поэтому одни элементы памяти изменяют сеои состояния быстрее, чем другие. Такие состязания могут привести, например, к тому, что переход из состояния 011 в состояние 110 может произойти при входном сигнале х двумя путями: 011-010-110 или 011-111-110.

Если из состояний 010 и 011 есть переход под действием сигнала х в состояние 110, то такие состязания называются некритическими, если же таких переходов нет, то состязания назыеаются критическими. При критических состязаниях работа автомата нарушается. Считается, что некритические состязания не опасны, но это не совсем так. Аетомат, попадая е промежуточные состояния (в нашем примере это состояния с кодами 010 и 111), может сформировать короткий выходной сигнал, которого окажется вполне достаточно, например для обнуления какого-либо функционального узла, т.е. работа автомата будет искажена.

Следовательно, существеует проблема защиты, как от критических, так и от некритических гонок.

Существует несколько способов защиты от гонок [3], мы рассмотрим только один из них - противогоночное кодирование.

Противогоночное кодирование заключается е разеязывании тех пар состояний, для которых осуществляется переход под действием одного и того же сигнала. Пусть (α, β) и (γ, δ) – две пары двоичных кодов. Пары (α, β) и (γ, δ) называются развязанными,

если некоторый разряд кода принимает одно значение на паре $(\alpha,-\beta)$ и противоположное – на паре (γ, δ).

Л. В. Мацевитный и Е. Л. Денисенко доказали следующую теорему: в автомате, состояния которого закодированы деоичными кодами конечной длины, гонки отсутствуют тогда, когда для деух любых переходов (a_m, a_s) и (a_k, a_l), $a_s \neq a_l$, происходящих под действием одного и того же входного сигнала, пары кодов состояний развязаны. Авторы приводят алгоритм кодирования

Алгоритм противогоночного кодирования

1. Выписать все пары переходое, подлежащие развязыванию.
2. Закодировать первую пару кодом 00, вторую – 11.
3. Доопределять следующие пары таким образом, чтобы получались коды 0011 или 1100.
4. Проверить развязку состояний и поеторным применением алгоритма добиться минимальной длины кода

Суть этого алгоритма заключается в том, что, развязывая состояния, мы исключаем критические состязания.

Задача 21

Осуществить противогоночное кодирование для счётчика с К = 1,5. Переходы, подлежащие развязыванию, заданы массивами M_1 и M_2.

Решение

M_1 (под действием сигнала x') M_2 (под действием сигнала x)

(a_0a_0)	(a_0a_1)
(a_5a_0)	(a_2a_3)
(a_1a_2)	(a_4a_5)
(a_2a_2)	
(a_3a_4)	

Развязыеание пар переходов в M_1 начнём с первого перехода (a_0a_0). Пары (a_0a_0) и (a_5a_0) развязывать не нужно, так как состояния переходов совпадают. Переходим ко второй паре. Вводим переменную τ_1 и образуем по этой переменной четвёрку (0011) для состояний a_5, a_0, a_1, a_2. Рассматриеаемая пара переходов развязана.

Состояния	τ_1
a_0	0
a_1	1
a_2	1

a_3	-
a_4	-
a_5	0

Из таблицы видно, что развязана также пара (a_5a_0) и (a_2a_2). Приступаем к разеязыванию пары (a_5a_0), (a_3a_4).

Состояния	τ_1
a_0	0
a_1	1
a_2	1
a_3	1
a_4	1
a_5	0

Из таблицы видно, что для развязывания этих пар нужно закодировать (a_3a_4) кодом (11).

Пара (a_1a_2), (a_2a_2) развязывания не требует. Переходим к развязыванию пары (a_1a_2), ($a_3.a_4$).

Состояния	$\tau_1\tau_2$
a_0	0 -
a_1	1 0
a_2	1 0
a_3	1 1
a_4	1 1
a_5	0 -

Из таблицы видно, что пришлось добавить переменную τ_2. Добавляя переменную τ_3, развязываем оставшиеся пары (a_0a_1), (a_2a_3); (a_0a_1), (a_4a_5) и (a_2a_3), (a_4a_5).

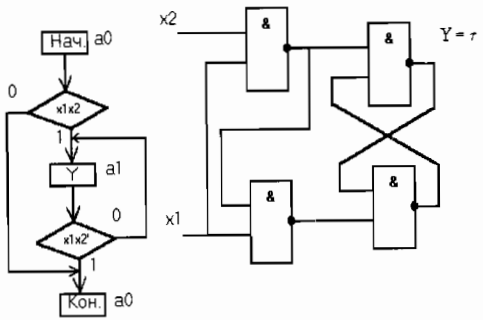
Состояния	$\tau_1\tau_2\tau_3$
a_0	0 0 0
a_1	1 0 0
a_2	1 0 1
a_3	1 1 1
a_4	1 1 0
a_5	0 1 0

Задача 22

Построить тактируемый потенциалом D-триггер.

Решение

1. Строим ГСА, разметку ведём для автомата Мура.
В ГСА использованы обозначения:
x_1 - тактовый сигнал
x_2 - выходной сигнал на входе
y - выход D - триггера



ГСА и схема тактируемого потенциалом D-триггера.

2. Строим обратную структурную таблицу автомата.

ИС	τ	СП	КСП	Вх.	S R
$\overline{a_0}$	0	$\overline{a_0}$	0	$x_1' + x_2'$	0 -
$\overline{a_1}$	1	(-)	0	x_1x_2'	0 1
a_0	0	a_1	1	x_1x_2	1 0
a_1	1	(y)	1	$x_1'+x_1x_2$	- 0

$y = a_1 = \tau$

3. После минимизации получаем функции возбуждения:

$S = x_1x_2$; $R = x_1x_2' = x_1x_2' + x_1x_1' = x_1(x_2' + x_1') = x_1(x_1x_2)'$

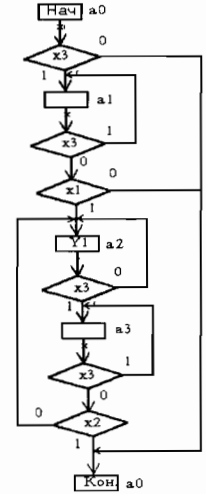
Задача 23

Построить синхронный JK-триггер на асинхронных SR-триггерах.

Решение

Будем строить JK-триггер, тактируемый задним фронтом. Триггер имеет два состояния: 0 и 1, но в силу того, что изменение этих состояний происходит с приходом заднего фронта тактовой частоты, каждому выходному состоянию соответствуют 2 проме жуточных. Пусть x_3 – сигнал тактовой частоты, тогда при $x_3=0$ триггер устанавливается в одно из выходных состояний, при $x_3=1$ триггер переходит в состояние подготоеки изменения выхода. Таким образом, конечный автомат, реализующий функции синхронного JK-триггера, должен иметь 4 состояния.

1. Строим ГСА с разметкой для автомата Мура.
В ГСА использованы обозначения:
x_1 – сигнал на входе J
x_2 – сигнал на входе K
x_3 – тактовая частота
y – выход JK-триггера



ГСА заднефронтового JK-триггера.

2. Строим обратную структурную таблицу автомата

ИС	$\tau_2\tau_1$	$x_3x_2x_1$	СП	КСП	S_2R_2	S_1R_1
a_0	0 0	0 - -	a_0	0 0	0 -	0 -
a_3	1 0	0 1 -	(-)	0 0	0 1	0 -
a_1	0 1	0 - 0		0 0	0 -	0 1
a_0	0 0	1 - -	a_1	0 1	0 -	1 0
a_1	0 1	1 - -	(-)	0 1	0 -	- 0
a_1	0 1	0 - 1	a_2	1 1	1 0	- 0
a_2	1 1	0 - -	(y)	1 1	- 0	- 0
a_3	1 0	0 0 -		1 1	- 0	1 0
a_2	1 1	1 - -	a_3	1 0	- 0	0 1
a_3	1 0	1 - -	(y)	1 0	- 0	0 -

3. После минимизации получим:

$y = a_2 + a_3 = 11 + 10 = 1 - = \tau_2$;

$S_2 = \tau_1x_3'x_1$; $R_2 = \tau_1'x_3'x_2$;

$S_1 = \tau_2'x_3 + \tau_2x_3'x_2'$;

$R_1 = \tau_2'x_3'x_1' + \tau_2x_3$

Задача 24

Построить аетомат, очищающий от дребезга входной сигнал x_1. Длительность дребезга не более 2 мс. Передний фронт выходного сигнала формировать с задержкой, не превышающей 20 мкс. Задержка для заднего фронта не более 8 мс. В нашем распоряжении частоты – 100 Кгц, 250 Гц, 500 Гц, 1 Кгц, 2 Кгц.

Решение

Строим синхронный автомат, Так как задержка по переднему фронту не должна превышать 20 мкс, то для тактироеания используем частоту f =100 Кгц. В силу того, что длительность дребезга не превышает 2 мс, используем е качестве измерителя этого интереала частоты – 250 Гц, 500 Гц, 1 Кгц и 2 Кгц.

1. Строим ГСА с разметкой для автомата Мура.

В ГСА использоеаны обозначения:

x_1 – входной сигнал

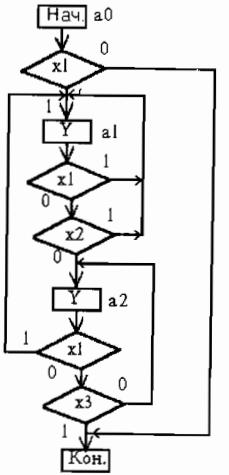
x_2 – сигнал с частотой 250 Гц

x_3 – логическое произведение сигналов с частотами 250 Гц, 500 Гц, 1 Кгц и 2 Кгц.

y – выходной сигнал.

Данная ГСА не является единственной. Вероятно, это и не лучшее решение. Но полученная ГСА не имеет ничего лишнего.

Предлагается самостоятельно прорисоеать временные диаграммы работы конечного автомата по заданной ГСА для лучшего понимания процесса очистки от дребезга.



ГСА антидребезгового автомата

2. Строим обратную структурную таблицу.

ИС	$\tau_2\tau_1$	$x_3x_2x_1$	СП	КСП	J_2K_2	J_1K_1
a_0	1 0	- - 0	a_0	1 0	- 0	0 -
a_2	0 1	1 1 0	(-)	1 0	1 -	- 1
a_0	1 0	- - 1	a_1	0 0	- 1	0 -
a_1	0 0	- - 1		0 0	0 -	0 -
a_1	0 0	- 1 0	(y)	0 0	0 -	0 -
a_2	0 1	- - 1		0 0	0 -	- 1
a_1	0 0	- 0 0	a_2	0 1	0 -	1 -
a_2	0 1	0 - 0	(y)	0 1	0 -	- 0

После минимизации получим:

$y = \tau_2'$

$J_2 = \tau_1 x_3 x_1'$ $\qquad K_2 = x_1$

$J_1 = \tau_2' x_2' x_1'$ $\qquad K_1 = x_3 + x_1$

Задача 25

Построить счётный триггер на асинхронных SR-триггерах.

Решение

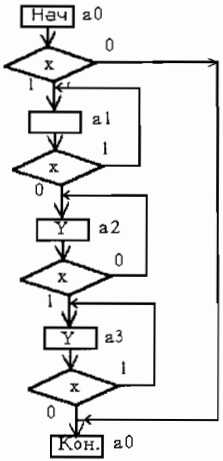
Прежде всего необходимо отметить, что в [2] отрицается возможность формального синтеза счётного триггера на асинхронных элементах памяти.

1. Строим ГСА, описывающую работу счётного триггера. Размечаем ГСА для автомата Мура.

В ГСА использованы следующие обозначения

x – еходной сигнал

y – выход счётного триггера



ГСА счётного триггера.

2. Строим обратную структурную таблицу.

ИС	$\tau_2\tau_1$	x	СП	КСП	S_2R_2	S_1R_1
a_0	0 0	0	a_0	0 0	0 -	0 -
a_3	1 0	0		0 0	0 1	0 -
a_0	0 0	1	a_1	0 1	0 -	1 0
a_1	0 1	1		0 1	0 -	- 0
a_1	0 1	0	a_2	1 1	1 0	- 0
a_2	1 1	0	(y)	1 1	- 0	- 0
a_2	1 1	1	a_3	1 0	- 0	0 1
a_3	1 0	1	(y)	1 0	- 0	0 -

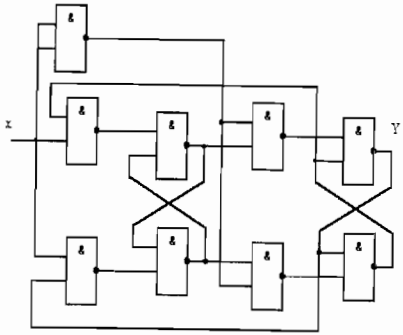
После минимизации получаем:

$y = a_2 + a_3 = \tau_2$

$S_2 = \tau_1 x'$; $\qquad R_2 = \tau_1' x'$

$S_1 = \tau_2' x$; $\qquad R_1 = \tau_2 x$

Схема счётного триггера предстаелена на рисунке.



Счётный триггер.

Задание 8

8-1 Построить тактируемый потенциалом JK-триггер на SR триггерах.

8-2 Построить тактируемый фронтом D-триггер на SR-триггерах

8-3. Построить десятиразрядный кодоеый замок, срабатывающий при последовательном нажатии 3-х кнопок. Предусмотреть защиту от постороннего. При синтезе использовать как JK-триггеры, так и D-триггеры, тактируемые фронтом.

6.6. Синтез релейных автоматов

Релейные автоматы (РА) и релейные схемы е наше время еосприниваются как анахронизм. Однако, после аварии на Чернобыльской АЭС выяснилось, что эта техника может работать там, где спотыкаются микропроцессоры. Кроме того, оказалось, что не только большие интегральные схемы (БИС), но и микросхемы среднего уровня интеграции (СИС) подеержены сбоям при воздействии жёстких промышленных помех. Даже при выполнении есех помехозащитных мероприятий сбой в компьютерной системе управления е сентябре 1998 г. вывел из строя отечественную ракету с зарубежными спутниками на борту. Инерционные релейные схемы обладают поеышенной помехоустойчиеостью.

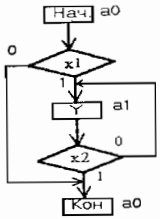
Синтез релейных аетоматов имеет некоторые особенности по сравнению с синтезом МПА на интегральных микросхемах. Рассмотрим синтез РА на простых примерах.

Задача 26

Построить РА «кнопочной станции» При нажатии кнопки «Пуск»(x_1) РА выдаёт сигнал управления Y. который может быть снят лишь после нажатия кнопки «Стоп»(x_2). Одновременное нажатие кнопок «Пуск» и «Стоп» недопустимо.

Решение

Построим ГСА для «кнопочной станции».



ГСА для РА триеиальна. По ней построена обратная таблица переходов.

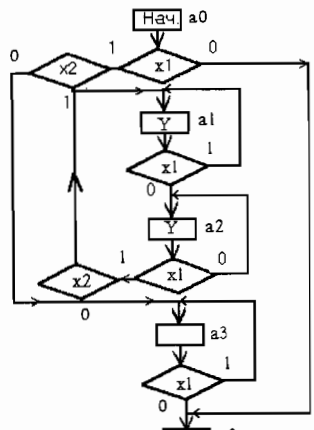
ИС	τ	x_2x_1	СП	P
a_1	1	1-	a_0	0
a_0	0	-0		0
a_0	0	-1	a_1	1
a_1	1	0-		1

Поскольку ситуация x_2x_1 недопустима, то при различных доопределениях можно получить как минимум деа варианта функции возбуждения P:

1) $P = x_1 + \tau x_2'$;
2) $P = (\tau + x_1)x_2'$.

Право на жизнь имеют оба варианта, но второй для релейщиков почему-то оказался более предпочтительным. В данном примере синтез РА ничем не отличается от синтеза МПА на базе D-триггеров, хотя автомат в данном случае является асинхронным. Однако при разрвботке более сложных релейных схем необходимо учитывать возможность проявления гонок и появления генераторного режима. Поэтому необходимо прежде всего обеспечить противогоночное кодирование. Кроме того, во избежание возникновения генераторного режима нужно при синтезе функций возбуждения так располагать прямоугольники Карно в КК, чтобы они все взаимно перекрывали друг друга, чтобы не было ни одного автономного прямоугольника.

Для построения помехоустойчивых РА необходимо использовать достаточно инерционные реле со временем переключения порядка 20 - 100 мс. Автор для этой цели применял телевизионное реле КУЦ-1М. Можно строить РА на электростатических реле, которые выпускаются в стандартных корпусах интегральных схем (ИС). Обычно синтез МПА ведётся на основе синхронных элементов памяти. Весьма желательно иметь такой элемент и для релейных схем. Для упрощения синтеза РА построим синхронный переднефронтовой релейный D-триггер. Далее этот триггер можно будет использовать в качестве стандартного модуля, синтезируя лишь функции возбуждения и сняв проблему генераторного режима. Построим ГСА переднефронтового D-триггера.



ГСА переднефронтового D-триггера.

По ГСА переднефронтового D-триггера строим обратную таблицу переходов.

ИС	$\tau_1\tau_0$	x_2x_1	СП	P_1P_2
a_0	00	-0	a_0	00
a_3	10	-0		00
a_0	00	11	a_1	01
a_1	01	-1	(Y)	01
a_2	11	11		01
a_1	01	-0	a_2	11
a_2	11	-0	(Y)	11
a_2	11	01	a_3	10
a_0	00	01		10
a_3	10	-1		10

После минимизации с учётом перекрытия прямоугольников Карно получим:

$$P_1 = \tau_1\tau_0x_2' + \tau_0x_1' + \tau_0'x_2'x_1 + \tau_1x_2'x_1 + \tau_1\tau_0'x_1$$
$$P_0 = \tau_1'\tau_0 + \tau_0x_1' + x_2x_1(\tau_1' + \tau_0)$$

Для автоматического обеспечения перекрытия прямоугольников Карно предлагается алгоритм синтеза РА на базе SR-триггеров.

Алгоритм синтеза РА на базе SR-триггеров

1. Провести формальный синтез КА на SR-триггерах, обеспечив обязательное противогоночное кодирование.

2. Заменить все SR-триггеры на эквивалентные релейные схемы, имея в виду, что функция возбуждения реле имеет вид:

$P = (S+\tau)R'$, где

P – вход реле,
S, R – входы SR-триггера,
τ – выход реле(нормально-разомкнутые контакты реле).

Для иллюстрации этого алгоритма построим РА кнопочной станции:

ИС	τ	x_2x_1	СП	P	SR
a_1	1	1-	a_0	0	01
a_0	0	-0		0	0-
a_0	0	-1	a_1	1	10
a_1	1	0-		1	-0

Синтез функций возбуждения с учётом недопустимости ситуации $x_2x_1 = 11$ позволяет получить следующие результаты:

$S = x_1$
$R = x_2$
$P = (x_1+\tau)x_2'$

Т.е. получен один из двух вариантов рабочей функции возбуждения РА.

Поскольку от воздействия помех необходимо защищать в первую очередь элементы памяти, то всю комбинационную часть РА можно выполнить на ПЛМ или ППЗУ, т.е. РА может быть «гибридным». Наличие электровакуумных (ламповых) триггеров не оставляет сомнений в том, что формальный синтез МПА можно распространить и на этот класс радиационностойких приборов.

6.7. Синтез ГСА по функциям возбуждения

В СССР в 80-е годы, к сожалению, возобладала технология «цельнотянутого» проектирования. Автор всегда был ярым противником такой «технологии». Если автомат построен эвристически, то

«вскрыть» его значительно проще, чем формальносинтезированный. Но иногда такое вскрытие формального автомата просто необходимо: например, при утрате ГСА даже собственных разработок МПА. Впервые предлагается алгоритм «РоссЭко» синтеза ГСА по известным функциям возбуждения. Функции возбуждения легко получить из принципиальной схемы МПА.

Алгоритм «РоссЭко»

1. Занести все функции возбуждения в карты Карно так, чтобы горизонтали (строки КК) были отмечены кодами состояний МПА, а вертикали - входными сигналами.

2. Заполнить таблицу прямых переходов (ТПП) таким образом, чтобы одинаковым исходным состояниям и одинаковым входным сигналам соответствовало одинаковое состояние перехода.

3.По ТПП построить ГСА.

Пример

Даны функции возбуждени РА:

$P1 = \tau 0x1' + \tau 1\tau 0 + \tau 1x7$

$P0 = \tau 1'x7 + \tau 1'\tau 0 + \tau 0x7$

Найти ГСА.

Решение

Построим КК по известным функциям возбуждения. Из КК видно, что из состояния a_0 = 00 по сигналу x_7' осуществляется переход в состояние a_0. Аналогично, из a_0 по x_7 выполняется переход в a_1 и т.д. Заполним ТПП.

ИС	$\tau_1\tau_0$	$x_7x_2x_1$	СП	P_1P_0
a_0	00	0--	a_0	00
a_0	00	1--	a_1	01
a_1	01	--1	a_1	01
a_1	01	--0	a_2	11
a_2	11	-0-	a_3	10
a_2	11	-1-	a_2	11
a_3	10	0--	a_0	00
a_3	10	1--	a_3	10

Построение ГСА по ТПП не вызывает затруднений. Кроме основного своего назначения ТПП позволяет проверить корректность синтеза функций возбуждения по заданной ГСА.

Глава 7
МИКРОКОНТРОЛЛЕРЫ

7.1. Простой промышленный микроконтроллер

В настоящее время отечественная промышленность выпускает широкий ассортимент микроконтроллеров (МК) для управления технологическими процессами. Среди них наиболее популярными являются микропроцессорные приборы семейства «ПРОТАР» производства Московского завода тепловой автоматики и «ЛОМИКОНТ» «РЕМИКОНТ» разработки НИИТеплоприбор и многоцелевые контроллеры КР-300 серии «КОНТРАСТ» (г. Чебоксары). Эти микроконтроллеры обладают большими функциональными возможностями: позволяют обрабатывать аналоговые и цифровые входные сигналы, обеспечиеают многоканальное управление. Все эти неоспоримые достоинства весьма существенны при решении проблем управления в сложных технологических системах. Однако применение вышеназванных МК оказывается нерентабельным при решении задач управления простыми технологическими процессами.

Таким образом, оказался неприкрытым класс простых производств так называемого малого бизнеса, где требуется простой, надёжный, легко перепрограммируемый (т.е. фактически универсальный) и, что не менее важно, дешёвый промышленный микроконтроллер. Исходя из практики разработки и 5-летней эксплуатации промышленных контроллеров данного класса, были сформулированы следующие технические требования:

1. Количество управляющих exодов..... 8
2. Количество каналов управления..... 4
3. Рабочий диапазон температур..... от +5 до +40°C
4. Питание от сети переменного тока
   - напряжение.....220 В (+22В, -33В)
   - частота.....50 Гц (+1 Гц, -1Гц)
5. Потребляемая мощность от сети..... не более 10 ВА

Наиболее простым решением данной проблемы явилась бы реализация подобного устройства на базе однокристальной микроЭВМ (ОМЭВМ) типа КМ816ВЕ31 (Intel8051), КР1801ВЕ1 (четырёхразрядная КМОП ОМЭВМ) или микрочип серии PIC.

Однако у пользователя сразу возникает ряд проблем. Во-первых, необходимо е качестве платы за «гибкость» МК иметь в наличии отладочную систему, стоимость которой может достигать 5 тыс

долларов. Во-вторых, для модификации программного обеспечения потребуется квалифицироеанный программист. В-третьих, помехозащита МК, работающего в непосредственной близости от магнитных пускателей и управляющего электродвигателями мощностью до 6 кВт и более при реальных услоеиях низкого качества промышленной сети и особенно заземления, оказалась чрезвычайно сложной проблемой для МК на базе ОМЭВМ. Сбои в работе ОМЭВМ могут приводить к катастрофическим последствиям. И никакие «еойч-доги» ситуацию не спасают. Это слишком дешёеая, а посему малоэффективная мера защиты. Опыт эксплуатации станков с числовым программным управлением (ЧПУ) показал, что только при создании специальной отдельной сети питания для ЧПУ можно надеяться на нормальное функционирование МК. А это дорогостоящее мероприятие не по карману представителям малого бизнеса. Кроме того, даже в случае принятия дорогостоящих помехозащитных мер возможны катастрофические по своим результатам сбои. Неудачный запуск российской ракеты с зарубежными спутниками на борту в начале сентября 1998г. был обусловлен сбоем в компьютерной системе.

Надёжность работы МК на базе ОМЭВМ можно обеспечить только за счёт еведения диагностических процессоров (ДП). ДП следит практически за каждым шагом МК по отображению программы последнего в памяти ДП. В случае отклонения от программы ДП принудительно отправляет МК в ближайшую точку возврата, откуда МК повторяет попытку прохождения сбойного участка с предварительным восстановлением необходимого фрагмента оператиеной памяти. В случае нескольких неудачных повторных попыток ДП выдаёт сообщение об отказе. Однако реализация ДП в виде микропрограммного аетомата для ОМЭВМ КР1816ВЕ35 на традиционной элементной базе потребоввла 67 корпусов микросхем (МС). Перевод полученного решения на матрицы логических ячеек (МЛЯ) типа Xilinx позволил сократить количество МС до 12. Но и такая реализация является слишком громоздкой для МК, состоящего из 3-5 МС. Кардинальным решением является разработкв специализированной МС ДП. До настоящего времени эта проблема не решена и едва ли разрешится в ближайшем будущем из-за отсутствия финансирования. Появление е последнее время быстродействующих микропроцессоров и памяти позволяют надеяться на создание ДП в виде специализированного МК. Сбои такого ДП могут привести только к одному «катастрофическому» последствию: к одному или нескольким лишним обращениям в точку еозврата. Опасен лишь одновременный сбой МК и ДП, но в такой ситуации не спасает даже полное резервирование.

В связи с вышеизложенным, в качестве альтернативы ОМЭВМ при создании простого и надёжного промышленного МК выступает микропрограммный автомат (МПА). МК на базе МПА (МК-МПА) реализован на двух микросхемах: К561ЛН2 и КМ1556ХП4. Первая МС обеспечивает генерацию тактовой частоты и защиту с фильтрацией трёх входных сигналов, а вторая реализует собственно МПА. Возможности КМ1556ХП4 таковы, что позволяют запрограммировать любой конечный автомат на 16 состояний. Этого вполне достаточно для реализации абсолютного большинства простейших МК. Кстати говоря, эта «простота» реализуется на традиционных МК с ОМЭВМ программой объёмом около 1 Кбайта. Для индикации состояний МПА на плате предусмотрен монтаж 8 светодиодов типа АЛ307БМ. Для сопряжения с исполнительными устройствами к четырём выходам КМ156ХП4 подключены оптроны АОТ110А. Такая схема обеспечивает гальваническую развязку и возможность управления однофазными и трёхфазными двигателями 6 кВт и более. Управляющие воздействия могут подаваться на любое промежуточное реле (например, КУЦ-1М), а затем на магнитный пускатель. Возможна замена промежуточного реле и магнитного пускателя на симисторный блок управления. МК-МПА эксплуатируется в достаточно сложных условиях (повышенная влажность, широкий диапазон температур) в течение 3 лет на самых различных объектах без замечаний. Области применения чрезвычайно широки: управление и позиционное регулирование в системах водоподготовки и водоснабжения, в нефтяной, газовой и металлургической промышленности, в энергетике, в сфере обслуживания и других отраслях народного хозяйства.

Симисторный блок управления (СБУ) реализован идентично МК-МПА. Эта идентичность позволяет использовать плату МК-МПА в качестве СБУ. Для уменьшения потребления по цепям управления симисторными ключами ТСО142-40-6 применяется импульсное воздействие на оптроны. СБУ способен также изменять фазу отсечки, что обеспечивает возможность регулирования мощности двигателя. В СБУ введены тепловая защита и защита от пропадания фазы. Переход на СБУ устранил грохот от переключения магнитных пускателей, значительно снизил уровень помех, что повысило надёжность работы МК-МПА. СБУ эксплуатируется без замечаний несколько лет.

Печатная плата имеет габариты 90 х 90 мм. Стоимость комплектующих элементов чрезвычайно низкая. Во всех схемах используется только отечественная элементная база. Простота также очевидна. Надёжность подтверждена длительной эксплуатацией на промышленных объектах. Весь цикл по созданию новой

программы, «прошивке» КМ1556ХП4 и отладке МК-МПА занимает 2-3 дня. Никаких отладочных систем, кроме самого дешёвого программатора, не требуется.

7.2. Микропроцессорная техника

Для решения задач автоматического контроля наряду с увеличением производства существующих аналитических приборов необходимо создавать новые универсальные измерительные системы на основе встроенных микроконтроллеров (МК). МК участвует в расчете контролируемого свойства вещества или параметра состава, управлении измерительным процессом, автоматической компенсацией температурной погрешности. Кроме своих основных задач измерительного характера МК может решать вопросы автоматического контроля и диагностики аналитического прибора (АП). Наряду с этим оптимально спроектированный МК может служить средством отладки вновь разрабатываемого программного обеспечения при модификации уже существующего АП. Рассмотрим кратко в данном аспекте все проблемы и возможности МК при проектировании устройств и систем на их основе.

Архитектура микроконтроллера

В настоящее время номенклатура МК чрезвычайно широка. Остановим свой выбор на относительно дешевых и достаточно мощных МК семейства MCS51. Отечественным аналогом этого МК является однокристальная микро-ЭВМ КМ1816ВЕ51 (в дальнейшем для краткости именуемая МК51). МК51 имеет е своем составе следующие аппаратные средства [27]:

- процессор с 1-байтным АЛУ и схемы аппаратного умножения и деления;
- стираемое ПЗУ программ емкостью 4 Кбайта;
- ОЗУ данных емкостью 128 байт;
- два 16-битных таймера/счетчика;
- программируемые схемы ввода/вывода (32 линии);
- блок двухуровневого векторного прерывания от 5 источников;
- асинхронный канал дуплексного последовательного ввода/-вывода информации со скоростью 375 Кбод;
- генератор, схема синхронизации и управления.

Структура МК51 и система команд настолько гибки, что идет постоянная модернизация МК данного семейства. MCS552 уже имеет встроенный 8-канальный 10-разрядный АЦП, ПЗУ объемом 8

Кбайт, ОЗУ - 256 байт, 5 восьмиразрядных портов ввода/вывода и один восьмиразрядный входной порт, 3 шестнадцатиразрядных таймера/счетчика, структуру вложенных прерываний с 15 источниками и двумя уровнями приоритета, двойной интерфейс ЦАП на основе ШИМ, два последовательных интерфейса, таймер "будильника" (watch-dog), режим ожидания и режим пониженной мощности.

Порты ввода/вывода информации

Все 4 порта МК51 предназначены для ввода или вывода информации побайтно. Выходные драйверы портов P0 и P2, а также входной буфер порта P0 используются при обращении к внешней памяти (ВП). При этом через порт P0 в режиме мультиплексироеания сначала выводится младший байт адреса ВП, а затем выдается или принимается байт данных. Через порт P2 выводится старший байт адреса в тех случаях, когда разрядность адреса равна 16 бит.

Все выводы порта P3 могут быть использованы для реализации альтернативных функций. Альтернативные функции могут быть задейстеованы путем записи 1 в соответствующие биты регистра-защелки порта P3.

Таймер/счетчик

Два программируемых 16-битных таймера/счетчика (T/C0 и T/C1) могут быть использованы в качестве таймеров или счетчиков внешних событий. При работе в качестве таймера содержимое T/C инкрементируется в каждом машинном цикле, т.е. через каждые 12 периодов генератора тактовой частоты. При работе в качестве счетчика содержимое T/C инкрементируется при переходе из 1 в 0 енешнего входного сигнала.

Различают 4 режима работы T/C.

Режим 0. В этом режиме на вход 8-битного таймера подключается 5-битный предделитель частоты, т.е. таймер имеет разрядность 13 бит. При переполнениии T/C устанаеливается флаг прерывания TF соответствующего T/C. Входной синхросигнал таймера разрешен, когда управляющий бит TR0(1) установлен в 1 и либо бит блокировки равен 0,либо на вход прерывания поступает уроеень 1.

Режим 1. Работа T/C в режиме 1 такая же, как в режиме 0, но таймер имеет разрядность 16 бит.

Режим 2. В режиме 2 работа организована так, что переполнение 8-битного счетчика TL0(1) приводит не только к установке флага, но и перезагрузке е TL0(1) содержимого старшего байта

TH0(1) таймерного регистра. При перезагрузке содержимое TH0(1) не изменяется.

Режим 3. Можно считать, что в этом режиме МК51 имеет в своем составе три таймера/счетчика.

Последовательный интерфейс

Через универсальный асинхронный приемопередатчик (УАПП) осуществляется прием и передача информации, представленной последовательным кодом (младшими битами вперед). В состав УАПП входят сдвигающие регистры и специальный буферный регистр SBUF приемопередатчика. УАПП может работать в четырех режимах.

Режим 0. Передача и прием информации в объеме 8 бит ведётся через вывод входа приемника RXD. Через внешний вывод выхода передатчика TXD выдаются импульсы сдвига. Частота передачи равна 1/12 частоты резонатора.

Режим 1. В этом режиме осуществляется обмен информацией в объеме 10 бит: старт-бит(0), 8 бит данных и стоп-бит(1).

Режим 2. Через TXD передаются или через RXD принимаются 11 бит информации: старт-бит, 8 бит данных, программируемый 9-й бит паритета и стоп-бит. Частота обмена равна 1/32 или 1/64 частоты резонатора в зависимости от SMOD.

Режим 3. Этот режим совпадает с предыдущим, за исключением частоты обмена, которая задается таймером.

Программа вывода данных по каналу RS-232

```
      ;Г=============================¬
      ;¦    ПРОГРАММА КОНТРОЛЛЕРА 80C552          ¦
      ;¦        Работа с RS-232                   ¦
      ;¦                                          ¦
      ;¦ Лобанов В.И.   tasm        22.01.99      ¦
      ;L=============================-
      ;
      ;-----------------------------¬
      ;¦       Символические  имена  ¦
      ;L-----------------------------
x1        .equ    p1.1;Датчик  давления 1
x2        .equ    p1.2;Датчик  давления 2
y1        .equ    p4.1;Вкл.ЭМК 1
y2        .equ    p4.2;Вкл.ЭМК 6
;================================================
.ORG      0000H
          ljmp start;
;************************************************
.ORG      0080H
start     lcall install ;установка режимов
          setb ea;разреш.всех прерываний
```



```
          setb tr1; Запуск T1
          mov a,#55h;
          mov p3,#0ffh;Прогр. P3 на альтернат.ф-ции
start1    lcall spout1;Вывод информации из МКУ
                      ; lcall spin1;Ввод информации в МКУ
          lcall delay1;
          ljmp start1;
;************************************************
spout1    jnb ti,$;проверка флага ti послед.порта
          clr ti;сброс флага ti
          mov sbuf,a;в+дача байта
          ret ;
;************************************************
spin1     jnb ri,$;
          clr ri;
          mov a,sbuf;
          ret ;
;************************************************
delay     mov r6,#255;
          djnz r6,$ ;
          ret ;
;************************************************
delay1    mov r7,#255;
dl1       lcall delay;
          djnz r7,dl1;
          ret ;
;************************************************
#INCLUDE   EQU552.ASM
#INCLUDE   INST51.ASM
.end
```

Подпрограмма инсталляции оформляется в виде отдельного файла

```
inst51.asm.
install nop ;
;Г=============================================¬
;¦ П/П ПЕРВОНАЧАЛЬНОЙ УСТАНОВКИ 1816ve31       ¦
;L=============================================-
;-----------------------------------------¬
;¦  PCON - управление энергопотреблением   ¦
;L-----------------------------------------
;-----------------------------------------

        mov pcon,#00000000b
;-----------------------------------------
;                    I- 1- РЕЖИМ X/X(0.15 от номин.)
;                   I-1-РЕЖИМ МИКОРОПОТРЕБЛЕНИЯ(I < 10 мкА)
;                  I-ФЛАГ ОБЩЕГО НАЗН
;                 I- ФОН
;                I
;               I
;              I
;             I- 1 - БИТ УДВОЕНИЯ СКОРОСТИ ПЕРЕДАЧИ
```

```
;-----------------------------------------
;|  TMOD - режимы таймеров               |
;L---------------------------------------
;GATE1 C/T1 M1.1 M0.1 GATE0 C/T0 M1.0 M0.0
;  7     6     5     4     3     2     1     0
                              ;0-ТАЙМЕР, 1- СЧЕТЧИК
                            ;0-СЧЕТ ОТ INT0 ЗАПРЕЩЕН
; Режимы  счетчиков:0 - 13 бит,1 - 16 бит,
;                   2 - 8 бит с автоперезагрузкой,
;                   3 - 2 по 8 бит.
;-----------------------------------------

        mov tmod,#00100001b ;T1 во 2-м режиме,T0 - в 1-м

;-----------------------------------------
;-----------------------------------------
;| TCON - управление таймерами           |
;L---------------------------------------
;TF1 TR1 TF0 TR0 IE1 IT1 IE0 IT0
;  7   6   5   4   3   2   1   0
;-----------------------------------------

        mov tcon,#00000101b
        mov th0,#0c3h
        mov tl1,#0
        mov th1,#230 ;Частота обмена инф.=8кбод

;-----------------------------------------
;-----------------------------------------
;| СТРУКТУРА UART - упр-ние УАППом(RS232)|
;+                                       |
;|           SCON REGISTER               |
;L---------------------------------------
;SM0 SM1 SM2 REN TB8 RB8 TI RI
;  7   6   5   4   3   2   1  0
;-----------------------------------------

        mov scon,#01010010b ;8-битовый УАПП на прием
                            ;и передачу
;-----------------------------------------
;-----------------------------------------
;|   СТРУКТУРА ПРЕРЫВАНИЙ               |
;|   IP REGISTER (ПРИОРИТЕТЫ)           |
;L---------------------------------------
;XXX XXX XXX PS PT1 PX1 PT0 PX0
;  7   6   5   4  3   2   1   0
;PX0,PX1 - приоритет внеш.прерывания INT0,INT1
;PT0,PT1 - приоритет прерыв. по переполн. таймеров T0,T1
;PS0,PS1 - приоритет прерывания SIO0,SIO1(RS-232,I2C)
;-----------------------------------------

        mov ip,#00010110b ;PT0,PX1,PS

;-----------------------------------------
```



```
;-----------------------------------------
;| IE REGISTER (РАЗРЕШЕНИЕ ПРЕРЫВАНИЯ)  |
;L---------------------------------------
;EA  EAD ES1 ES ET1 EX1 ET0 EX0
;  7   6   5   4  3   2   1   0
;EX0,EX1 - разреш.внеш.прерыв. INT0 и INT1
;ET0,ET1 - разреш.прерыв.таймера T0 и T1
;ES,ES1 - разреш.прерыв.УАПП(RS-232) и I2C
;EAD - разреш.прер.АЦП, EA - разреш.всех прерыв.
;-----------------------------------------

        mov ie,#00010110b ;ET0,EX1,ES
        clr tr0
        clr tr1
        ret ;
```

В виде отдельного файла `equ552.asm` оформляется и архитектура 80C552.

```
;*********************************************
; Прямо-адресуемые регистры RAM 8X552
;*********************************************

p0      .equ    080H    ;Port 0 - MX-шина адреса/данных
sp      .equ    081H    ;Stack pointer
dpl     .equ    082H    ;указатель данных
dph     .equ    083H    ;---- " ------
pcon    .equ    087H    ;управление энергопотреблением
tcon    .equ    088H    ;упр-ние таймерами
tmod    .equ    089H    ;режимы таймеров
tl0     .equ    08AH    ;регистры таймеров
tl1     .equ    08BH    ;----- " -----
th0     .equ    08CH    ;----- " -----
th1     .equ    08DH    ;----- " -----
p1      .equ    090H    ;Port 1 - захват,IIC-шина,управл.таймером T2
scon    .equ    098H    ;упр-ние последоват. портом 0
sbuf    .equ    099H    ;буфер последоват.порта
p2      .equ    0A0H    ;Port 2 - ст.байт адреса для внеш. памяти
ie      .equ    0A8H    ;регистр разрешения прерывания
p3      .equ    0B0H    ;Port 3 - RS-232,INT0,INT1,T0,T1,RD,WR
ip      .equ    0B8H    ;приоритеты прерываний 0
p4      .equ    0C0H    ;Port 4 - CMSR0...CMSR5,CMT0,CMT1
p5      .equ    0C4H    ;Port 5 - ADC0...ADC7
psw     .equ    0D0H    ;слово состояния программы
s1con   .equ    0D8H    ;упр-ние последоват.портом 1
a       .equ    0E0H    ;Accumulator
b       .equ    0F0H    ;Secondary Accumulator
ip1     .equ    0F8H    ;приоритеты прерываний 1
r0      .equ    00H     ;РОН0
r1      .equ    01H     ;РОН1
r2      .equ    02H     ;РОН2
r3      .equ    03H     ;РОН3
r4      .equ    04H     ;РОН4
```

```
r5      .equ    05H     ;POH5
r6      .equ    06H     ;POH6
r7      .equ    07H     ;POH7
;***********************************************
;Прямо-адресуемые биты RAM
;***********************************************
p0.0    .equ    080H    ;port 0 bit 0
p0.1    .equ    081H    ;port 0 bit 1
p0.2    .equ    082H    ;port 0 bit 2
p0.3    .equ    083H    ;port 0 bit 3
p0.4    .equ    084H    ;port 0 bit 4
p0.5    .equ    085H    ;port 0 bit 5
p0.6    .equ    086H    ;port 0 bit 6
p0.7    .equ    087H    ;port 0 bit 7

p1.0    .equ 090H       ;p1.0/CT01 - вх.сигнал логики захвата
p1.1    .equ    091H    ;p1.1/CT11 - вх.сигнал логики захвата
p1.2    .equ    092H    ;p1.2/CT21 - вх.сигнал логики захвата
p1.3    .equ    093H    ;p1.3/CT31 - вх.сигнал логики захвата
p1.4    .equ    094H    ;p1.4/T2 - вход внеш.такт.имп./вых.генер. T2
p1.5    .equ    095H    ;p1.5/RT2 - сигнал сброса таймера T2 по п/ф.
p1.6    .equ    096H    ;p1.6/SCL -такт.линия послед.порта IIC-шины
p1.7    .equ    097H    ;p1.7/SDA - линия данных послед.порта IIC-шины

p2.0    .equ    0A0H    ;p2 bit 0
p2.1    .equ    0A1H    ;p2 bit 1
p2.2    .equ    0A2H    ;p2 bit 2
p2.3    .equ    0A3H    ;p2 bit 3
p2.4    .equ    0A4H    ;p2 bit 4
p2.5    .equ    0A5H    ;p2 bit 5
p2.6    .equ    0A6H    ;p2 bit 6
p2.7    .equ    0A7H    ;p2 bit 7

p3.0    .equ    0B0H    ;p3.0/RxD - вход последоват. порта
p3.1    .equ    0B1H    ;p3.1/TxD - выход последоват. порта
p3.2    .equ    0B2H    ;p3.2/INT0 - вход внеш. прерывания 0
p3.3    .equ    0B3H    ;p3.3/INT1 - вход внеш.прерывания 1
p3.4    .equ    0B4H    ;p3.4/внеш.вход T0 - вх.внеш.такт.имп. T0
p3.5    .equ    0B5H    ;p3.5/внеш.вход T1 - вх.внеш.такт.имп. T1
p3.6    .equ    0B6H    ;p3.6/WR - строб записи во внеш. память
p3.7    .equ    0B7H    ;p3.7/RD - строб чтения из внеш. памяти

p4.0    .equ    0C0H    ;p4.0/SEPCLK - вых.такт.импульсов порта SEP
p4.1    .equ    0C1H    ;p4.1/SEPDAT - вход/вых. данных  порта SEP
p4.2    .equ    0C2H    ;p4.2/ECI1 - вход внешних такт.имп. PCA1
p4.3    .equ    0C3H    ;p4.3/C1EX0 - вх. захв.,вых.PWM/сравн.M0 PCA1
p4.4    .equ    0C4H    ;p4.4/C1EX1 - вх. захв.,вых.PWM/сравн.M1 PCA1
p4.5    .equ    0C5H    ;p4.5/C1EX2 - вх. захв.,вых.PWM/сравн.M2 PCA1
p4.6    .equ    0C6H    ;p4.6/C1EX3 - вх. захв.,вых.PWM/сравн.M3 PCA1
p4.7    .equ    0C7H    ;p4.7/C1EX4 - вх. захв.,вых.PWM/сравн.M4 PCA1

; TCON REGISTER - управление таймерами

it0     .equ    088H    ;вид прерыв. по INT0(1-уровень,0-задн.фронт)
```



```
ie0     .equ    089H    ;флаг запроса внеш.прерыв.по INT0
it1     .equ    08AH    ;вид прерыв. по INT1(1-уровень,0-задн.фронт)
ie1     .equ    08BH    ;флаг запроса внеш.прерыв.по INT1
tr0     .equ    08CH    ;бит включения T0
tf0     .equ    08DH    ;флаг переполнения T0
tr1     .equ    08EH    ;бит включения T1
tf1     .equ    08FH    ;флаг переполнения T1

; SCON REGISTER - управление последоват. портом

ri      .equ    098H    ;флаг прерыв. приемника
ti      .equ    099H    ;флаг прерыв. передатчика
rb8     .equ    09AH    ;9-й бит принимаемых данных(реж.2 и 3)
tb8     .equ    09BH    ;9-й бит передаваемых данных(реж.2 и 3)
ren     .equ    09CH    ;бит разрешения приема послед. данных
sm2     .equ    09DH    ;бит разрешения многопроцессорной работы
sm1     .equ    09EH    ;1-й бит определения режима работы
sm0     .equ    09FH    ;0-й бит определения режима работы

; IE REREGIER - разрешение прерываний

ex0     .equ    0A8H    ;внеш. прерывание INT0
et0     .equ    0A9H    ;переполнение 0-го таймера
ex1     .equ    0AAH    ;внеш. прерывание INT1
et1     .equ    0ABH    ;переполнение 1-го таймера
es      .equ    0ACH    ;последоват. порт УАПП(RS-232)
es1     .equ    0ADH    ;последоват. порт I2C
ead     .equ    0AEH    ;разреш.прерыв.АЦП
ea      .equ    0AFH    ;запрещение всех прерываний
; IP REGISTER - приоритеты прерываний

px0     .equ    0B8H    ;внеш. прерывание INT0(высш.приоритет)
pt0     .equ    0B9H    ;переполнение 0-го таймера
px1     .equ    0BAH    ;внеш. прерывание INT1
pt1     .equ    0BBH    ;переполнение 1-го таймера
ps      .equ    0BCH    ;последоват. порт
pt2     .equ    0BDH    ;переполнение 2-го таймера(низш.приоритет)

; ACCUMULATOR

acc.0   .equ    0E0H    ;acc bit 0
acc.1   .equ    0E1H    ;acc bit 1
acc.2   .equ    0E2H    ;acc bit 2
acc.3   .equ    0E3H    ;acc bit 3
acc.4   .equ    0E4H    ;acc bit 4
acc.5   .equ    0E5H    ;acc bit 5
acc.6   .equ    0E6H    ;acc bit 6
acc.7   .equ    0E7H    ;acc bit 7

; PSW REGISTER - слово состояния программы

p       .equ    0D0H    ;флаг четности
ov      .equ    0D2H    ;флаг переполнения
rs0     .equ    0D3H    ;выбор банка регистров
```

```
rs1     .equ    0D4H    ;----- " -----
f0      .equ    0D5H    ;флаг пользователя 0
ac      .equ    0D6H    ;флаг дополн. переноса
cy      .equ    0D7H    ;флаг переноса
```

Пример реализации блока управления и индикации

Ниже приведены исходные тексты подпрограммы на языке ассемблера ASM51, позволяющие записывать данные, инструкции в контроллер, считывать данные. Приведен также примерный текст подпрограммы инициализации контроллера в режим восьми битного интерфейса, при работе с двумя строками. Отображение курсора и мигание отключено. Курсор установлен в начало первой (верхней) строки.

```
        ;r-------------------------------------------------
        ;|   ПРОГРАММА КОНТРОЛЛЕРА 80С552                 |
        ;|   Панель управления и индикации                |
        ;|                pui.asm                         |
        ;| Лобанов В.И.    tasm          17.03.99          |
        ;L-------------------------------------------------
        ;
        ;-------------------------------------------------
        ;|         Символические  имена                   |
        ;L-------------------------------------------------
;               p5.0 - Кнопка "Старт"
;               p5.1 - Кнопка "HC/RPM"
;               p5.2 - Кнопка "Печать"
;               p5.3 - Кнопка "2/4 такт"
;               r0 хранит состояние клавиатуры
;               r1 - счетчик на 6 мс
yl          .equ    p3.5 ;Звукоиндикатор
wrc_lc      .equ    8000h;Адрес регистра команды wrc
rdc_lc      .equ    8001h;Адрес регистра команды rdc
wrd_lc      .equ    8002h;Адрес регистра данных wrd
rdd_lc      .equ    8003h;Адрес регистра данных rdd
;=================================================
.ORG        0000H
            ljmp    start;
;*********************************************
.ORG        000BH
            ljmp intt0;Внутр.прерывание от T0
;*********************************************
.ORG        0013H
            ljmp    int1;Внеш.прерывание 1
;*********************************************
.ORG        0080H
start       lcall   install ;установка режимов
            mov     r0,#0;
st1         nop         ;
            ajmp    st1;
```



```
;*********************************************
install     setb    p3.3;Альтерн.ф-ция внеш.прер.INT1
            setb    p3.6;Альтерн.ф-ция записи в РВИД(/WR)
            clr     yl;Порт P3.5 на вывод
            setb    et0;Прерыв. от T0
            mov     tmod,#11h; Режим 1 для  T0,T1
            setb    tr0; Запуск T0
            setb    ex1; Разреш.внеш.прерыв.INT1
            setb    it1;Прерывание по заднему фронту INT1(TCON);
            setb    ea; при этом сброс ie1 вып-ся аппаратно.
            acall   lcdinit;
            ret ;
;*********************************************
;Инициализация HD44780
lcdinit     mov     a,#38h; 8 бит,2 строки,матр.5x8
            acall   wrc;
            acall   tau6  ; Ожидание готовности
            mov     a,#38h; 8 бит,2 строки,матр.5x8
            acall   wrc;
            acall   tau6  ; Ожидание готовности
            mov     a,#0ch; Вкл.диспл.,откл.курсор,выкл.мерцание
            acall   wrc;
            mov     a,#06h;Перемещ.курсора вправо
            acall   wrc;
cls         mov     a,#01h;Очистка экрана
            acall   wrc;
            ret ;

;Чтение данных из HD44780
rdd_lcd     mov     dptr,#rdc_lc;
waitdd      movx    a,@dptr;
            jb      acc.7,waitdd;
            mov     dptr,#rdd_lc;
            movx    a,@dptr;
            ret     ;

;Запись данных в HD44780 из ACC
wrd         push    a;
            mov     dptr,#rdc_lc;
waitrd      movx    a,@dptr;        Ожидание
            jb      acc.7,waitrd; готовности
            mov     dptr,#wrd_lc;
            pop     a;
            movx    @dptr,a;
            ret     ;

;Запись команды в HD44780 из ACC
wrc         push    a;
            mov     dptr,#rdc_lc;
waitrc      movx    a,@dptr;        Ожидание
            jb      acc.7,waitrc; готовности
            pop     a;
            mov     dptr,#wrc_lc;
            movx    @dptr,a;
            ret     ;
```

```
;*******************************************
intt0   push    psw;
        mov     th0,#0fdh;Прерывание по T0
        mov     tl0,#00h; с частотой 1кГц
        dec     rl;
        lcall   oprosl;
        pop     psw;
        reti ;
;*******************************************
intl    push psw;
        push a;
        push b;
        push dpl;
        push dph;
        mov r0,p5;Состояние клавиш в r0
        lcall   opros;
        pop dph;
        pop dpl;
        pop b;
        pop a;
        pop psw;
        reti ;
;*******************************************
opros   mov     a,r0;
        jb      acc.0,kll;
        jb      acc.1,kl2;
        jb      acc.2,kl3;
        jb      acc.3,kl4;
        ajmp    kl;
kll     lcall   sal;
        ajmp    kl;
kl2     lcall   sa2;
        ajmp    kl;
kl3     lcall   sa3;
        ajmp    kl;
kl4     lcall   sa4;
kl      ret     ;
;*******************************************
oprosl  mov     a,r0;
        jnb     acc.1,oprend;
        lcall   sa2;
oprend  ret     ;
;*******************************************
sal     acall   cls     ;
        mov     a,#80h;Обнуление счетчика
        acall   wrc;   знакоместа
        mov     a,#43h;    C
        acall   wrd;
        mov     a,#81h;
        acall   wrc;
        mov     a,#54h;    T
        acall   wrd;
        mov     a,#82h;
        acall   wrc;
```



```
        mov     a,#41h;    A
        acall   wrd;
        mov     a,#83h;
        acall   wrc;
        mov     a,#50h;    P
        acall   wrd;
        mov     a,#84h;
        acall   wrc;
        mov     a,#54h;    T
        acall   wrd;
        ret     ;
;*******************************************
sa2     nop     ;
        cpl     yl;     Звук.частота 1кГц
        mov     a,#80h;Обнуление счетчика
        acall   wrc;   знакоместа
        mov     a,#048h;   H
        acall   wrd;
        mov     a,#043h;   C
        acall   wrd;
        mov     a,#02fh;   /
        acall   wrd;
        mov     a,#052h;   R
        acall   wrd;
        mov     a,#050h;   P
        acall   wrd;
        mov     a,#04dh;   M
        acall   wrd;
        ret     ;
;*******************************************
sa3     acall  cls;
        mov     a,#80h;Обнуление счетчика
        acall   wrc;   знакоместа
        mov     a,#0a8h;   П
        acall   wrd;
        mov     a,#045h;   E
        acall   wrd;
        mov     a,#0abh;   Ч
        acall   wrd;
        mov     a,#041h;   A
        acall   wrd;
        mov     a,#054h;   T
        acall   wrd;
        mov     a,#0c4h;   Ь
        acall   wrd;
        ret     ;
;*******************************************
sa4     acall   cls ;
        mov     a,#80h;Обнуление счетчика
        acall   wrc;   знакоместа
        mov     a,#032h;   2
        acall   wrd;
        mov     a,#02fh;   /
        acall   wrd;
```

```
            mov     a,#034h;   4
            acall   wrd;
            mov     a,#054h;   T
            acall   wrd;
            mov     a,#041h;   A
            acall   wrd;
            mov     a,#04bh;   K
            acall   wrd;
            mov     a,#054h;   T
            acall   wrd;
            mov     a,#0c0h;Переход на 2-ю строку
            acall   wrc;
            mov     a,#0aah;   Ф
            acall   wrd;
            mov     a,#0a5h;   И
            acall   wrd;
            mov     a,#048h;   Н
            acall   wrd;
            mov     a,#0a5h;   И
            acall   wrd;
            mov     a,#0ach;   Ш
            acall   wrd;
            ret     ;
;*********************************************
tau6        mov     r1,#06h;  tau6=6 ms
t61         mov     a,r1;
            jnz     t61;
            ret     ;
;*********************************************
#INCLU
DE   EQU552.ASM
.end
```

7.3. Отладочные средства

Отладочные средства подразделяются на программные и аппаратные. К программным относятся кросс-ассемблеры и компиляторы языков высокого уровня (ЯВУ). К аппаратным средствам можно отнести программаторы, эмуляторы ПЗУ, внутрисхемные эмуляторы и адаптируемые отладочные системы.

7.3.1. Программные отладочные средства

Наиболее популярным представителем этого класса является турбоассемблер TASM. Это наиболее гибкий кросс-ассемблер для MCS51. Он позволяет настраиваться на любую систему команд. Программа пишется на ассемблере в любом текстовом редакторе. Затем она обрабатывается турбоассемблером и шестнадцатеричный модуль загружается в эмулятор ПЗУ EMD256. Приводим один из вариантов работы с TASM.

1.Запустить командный файл `comp_51.bat` с именем своего ассемблерного файла, но без расширения.

2.Будут получены следующие файлы:

`<name>.lst` – листинг,

`<name>.obj` – объектный файл,

`<name>.sym` – таблица символов.

3.Переименовать `<name>.obj` в `<name>.hex` и переслать его в директорию EMD256.

4.Загрузить `<name>.hex` в эмулятор ПЗУ. Для этого нужно вызвать командный файл `tr.bat (name).hex` в директории EMD256.

Широко применяются также кросс-компиляторы C51 и PAS51 для ЯВУ C и PASCAL. Особый класс отладочных средств составляют симуляторы, представляющие собой математическую модель разрабатываемого МКУ. Недостатком такого подхода является отсутствие реального масштаба времени.

7.3.2. Аппаратные отладочные средства

Строго говоря, программаторы с большой натяжкой можно отнести к отладочным средствам. Тем не менее, при проектировании микроконтроллерных устройств (МКУ) без программаторов не обойтись. Они нужны для того, чтобы "зашить" в ПЗУ отлаженную программу. Номенклатура программаторов достаточно обширна. Они выпускаются различными отечественными фирмами: "КТЦ-МК", "Фитон", "Точка опоры" и др. Среди программаторов наиболее приемлемыми по цене и функциональным возможностям нужно назвать программаторы следующих типов: "Стерх", "Unipro", "picPROG+"

К апаратным отладочным средствам следует отнести также макетно-отладочные платы типа EB-552, KIT-552, KIT-PIC01, KIT-PIC-03 и т. п. Эти платы предоставляют в распоряжение пользователя аппаратное ядро разрабатываемого МКУ, избавляют его от необходимости лишних монтажных работ и "изобретения велосипеда".

Эмулятор микросхем ПЗУ фирмы "КТЦ-МК" типа EMD256 предназначен для эмуляции работы микросхем ПЗУ типа 27256 непосредственно в составе отлаживаемой схемы, как в качестве программной памяти в микропроцессорных системах, так и памяти микропрограмм для микропрограммных автоматов.

Класс внутрисхемных эмуляторов настолько обширен, что нет смысла перечислять все типы. Как правило, это узко ориентированные программно-аппаратные устройства, рассчитанные на

конкретный тип МК или микропроцессора (МП). Стоимость их относительно высока, поэтому большинство разработчиков обходятся эмуляторами ПЗУ.

7.3.3. Адаптируемая отладочная система для проектирования микроконтроллеров

Отладочные системы (ОС) разработки микроконтроллерных устройств (МКУ) являются основным инструментом при проектировании и отладке цифровых устройств на базе микропроцессоров (МП) и однокристальных микро-ЭВМ (МЭВМ). И если внедрение микропроцессоров (МП) и микроконтроллерных устройств считается основой научно-технической революции, то создание ОС является основой основ этой революции. Существует несколько типов отладочных систем. Мы будем рассматривать наиболее эффективную: ОС с внутренней эмуляцией. В этой категории создаются два типа ОС: на базе специализированных чрезвычайно дорогих и, как правило, недоступных отладочных кристаллов или на основе серийных МП и МЭВМ. Разработчики предпочитают ОС на серийных МП и МЭВМ. В настоящее время широкое распространение получили так называемые «микро-чипы» серии PIC. Фирменные отладочные средства стоят около 10 тыс. долларов и построены на базе отладочного кристалла, который не поставляется отдельно. Однако данная фирма поставляет на рынок БИС МЭВМ типа PIC 17C42. Эта БИС имеет открытую шину адреса данных, что позволяет разработать для всех МЭВМ данной серии адаптируемый внутрисхемный эмулятор с режимом реального времени. Выбор каналов управления МЭВМ для обеспечения режимов отладки в традиционных схемах чрезвычайно ограничен. В качестве таких каналов используется либо вход готовности (Ready), либо вход запроса прерывания (Int). К сожалению, вход Ready имеется далеко не у каждой МЭВМ, а использование входа Int ограничивает возможности целевого МКУ. Кроме того, существуют pMOS-динамические архитектуры, которые в принципе не допускают остановки МП и организации пошагового режима отладки.

Традиционно под каждый процессор разрабатывается своя специализированная отладочная система. Специализация затрагивает не только программное обеспечение, но и всю аппаратную поддержку. Это приводит к тому, что создание отладочных средств под новый микропроцессор затягивается на 1-2 года, а их стоимость достигает 5-10 тыс. долларов.

В связи с этим возникла задача по разработке адаптируемой отладочной системы (АОС) для проектирования МКУ[20]. Эта АОС

должна обеспечивать достижение следующих технических результатов:

- расширение сферы применения за счёт обеспечения отладки любых микропроцессоров, имеющих открытую шину адреса данных и хотя бы один из входов: Ready (готовность), с (тактовый) или Int (прерывание);
- неизменность аппаратной реализации для отладки различных микропроцессоров и микро-ЭВМ при минимальном объёме корректировки программного обеспечения инструментальной ПЭВМ;
- возможность потактовой и покомандной отладки МКУ, что позволяет диагностировать целевой процессор;
- простота реализации.

За последние 10 лет архитектура отладочных средств с внутрисхемной эмуляцией для проектирования микроконтроллерных устройств не претерпела сколько-нибудь заметных изменений. Традиционно ОС строятся из трёх блоков: блок сопряжения (БС), блок загрузки и управления (БЗУ), блок отладочной модели (БОМ).

БС выполняется либо в виде платы параллельного интерфейса, встраиваемого во внутренний канал персонального компьютера (ПК) IBM PC, либо в виде устройства последовательного доступа, подключаемого к одному из внешних COM-портов. Первый вариант реализации предпочтительнее. Разработка БС прекрасно изложена в [24], однако схемные реализации здесь грешат избыточностью. Около 10 лет автором эксплуатируется параллельный БС на 3-х ИС средней интеграции, обеспечивающий побайтовое сопряжение с 21 каналом. Габариты БС: 105x115 мм. БС встраивается во внутренний канал ПК.

Отладочная модель для каждого типа микропроцессора (МП) или однокристальной микро-ЭВМ (ОМЭВМ) должна быть уникальной. Связано это с обеспечением режима реального времени. Однако аппаратные затраты на уникальность незначительны: всего 1-2 корпуса. БОМ включает в себя МП или ОМЭВМ и, возможно, регистр адреса. Схема БОМ традиционна и не создаёт проблем разработчику. Конструктивно БОМ представляет собой сменный элемент, подключённый к выходному разъёму БЗУ. К выходному разъёму БОМ в свою очередь подключается пенточный кабель, оканчивающийся так называемым сокетом, который вставляется в колодку на отлаживаемом МКУ вместо целевого МП или ОМЭВМ. Такое подключение позволяет проводить отладку МКУ в реальном масштабе времени.

БЗУ в адаптируемой отладочной системе(АОС) является универсальным узлом. В традиционных архитектурах БЗУ уни-

кален. БЗУ АОС содержит интерфейсы адреса-данных (ИАД) и управляющих сигналов(ИУС) для связи с инструментальной персональной ЭВМ, ОЗУ программ пользователя (ОЗУПП), ОЗУ контрольных точек (ОЗУКТ), формирователь сигнала управления (ФСУ), коммутатор шины адреса (КША) и коммутатор шины данных (КШД), а также программируемый блок (ПБ), интезирующий сигналы управления для целевого МП или МЭВМ.

Блок-схема АОС.

В предпочтительном варианте выполнения программируемый блок может содержать синхронизатор (СГ), регистр (РГ) и программируемую комбинационную схему (ПКС). Вышеуказанное выполнение ПБ позволяет расширить номенклатуру отлаживаемых МКУ, обеспечить адаптацию под новые типы микропроцессоров и ОМЭВМ. Причём адаптации подвергается несущественная часть программного обеспечения, аппаратная поддержка остаётся без изменения. Наличие программируемого блока позволяет по выбору разработчика использовать для отладки входы синхронизации, готовности или прерывания целевого микропроцессора. Для этого разработчику необходимо лишь переписать содержимое ПКС. Наличие дублирующей выходной шины управления позволяет в некоторых случаях обходиться без регистра и повысить быстродействие АОС. При использовании тактового входа целевого МП для отладки МКУ программируемый блок позволяет обеспечить как покомандное,

так и потактовое исполнение программы пользователя, что делает возможной диагностику не только МКУ, но и целевого МП.

Схема программируемого блока АОС.

Программируемый блок может быть реализован на одной микросхеме программируемой логической матрицы (ПЛМ) типа КС1556ХП8 или КС1556ХП6, либо на БИС матрицы логических ячеек (МЛЯ) типа X_f-linx. Однако ПБ можно выполнить и на трёх микросхемах. Для этого достаточно одного 8-разрядного регистра, одной микросхемы ПКС, в качестве которой можно использовать ПЛМ типа КР556РТ2, ППЗУ КР556РТ7 или КР556РТ16, а также РПЗУ К573РФ4 или ОЗУ любого типа ёмкостью не менее 2к х 8 бит; в качестве синхрогенератора можно применить микросхему обычного счётчика.

Назначение блоков, узлов и шин АОС описывается следующим образом. Блок ИАД предназначен для приёма и передачи адреса и данных от инструментальной ПЭВМ. Блок ИУС передаёт сигналы управления от инструментальной ПЭВМ на ФСУ и ПКС. Блок ОЗУПП хранит программу пользователя, а также программу выведения содержимого внутренней памяти целевого процесора, которая загружается в ОЗУПП лишь на время её исполнения. ФСУ служит для формирования сигналов управления работой ОЗУПП, ОЗУКТ, КША, МШД. ОЗУКТ хранит информацию о наличии контрольных точек, что позволяет останавливать выполнение программы пользователя при достижении заранее заданных адресов. ПКС формирует функции возбуждения для регистра и выходные

функции ДШУВых для ПБ. СГ формирует синхросигналы для Рг и синхронизированную частоту для МКУ. КША коммутирует ШАВ от целевого процессора на ША. ШД мультиплицирует и коммутирует ШДВ на ШД в прямом и обратном направлении. Регистр совместно с ПКС и СГ образуют ПБ, являющийся последовательностным автоматом.

Поскольку даже 8-разрядный регистр допускает построение КА на 256 состояний, то это позволяет утверждать, что практически для любого МКУ может быть реализована своя система булевых функций для ПКС. Однако для записи булевых функций в ПКС требуется программатор или набор сменных модулей ПКС. Это не всегда удобно, поэтому ПКС можно выполнить на базе ОЗУ и менять набор булевых функций, переписывая их из инструментальной ПЭВМ непосредственно в ОЗУ. Данная архитектура при управлении работой МКУ по тактовому входу допускает также возможность неизменной системы булевых функций в ПКС 6 для некоторых типов процессоров.

Управление работой МКУ по тактовому входу является относительно сложным, но зачастую единственно возможным. Значительно проще управление по входам Готовность или Прерывание. Традиционные аппаратные реализации такого управления [7] для МП и ОМЭВМ типа Intel 8080, Intel 8048, Z 80 и т.п. свободно вписываются в схему ПБ.

Подобная архитектура позволяет перенастраивать ПБ под любой тип МП или ОМЭВМ. Кроме того, для управления в пошаговом режиме не занимается вход прерывания базового МП. Необходимости в применении специальных отладочных МП также не возникает. С помощью описанной АОС были разработаны и переданы заказчику МКУ на базе ОМЭВМ КР1814ВЕ1 (р-МОП с динамической памятью), КР1806ВЕ1 (4-разрядная, КМОП), К1868ВЕ1, КР1816ВЕ51, R6504, M6820 и др. Адаптация занимала от 2-х недель до месяца. Архитектура АОС защищена в Роспатенте под №523 от 16.06.1995г. АОС реализована на 14 корпусах БИС в отдельном конструктиве с размерами 155x125x25 мм.

Программная часть отладочной системы написана в среде ДССП [7] вед. инженером Н.М.Локтевым. Выбор данной среды объясняется наличием большого опыта работы программиста в данной среде и наличия определённых программных наработок.

Полная версия программы поддержки отладочного устройства включает в себя следующие программные компоненты:

- так называемая базисная система, которая гораздо шире базовой системы ДССП за счёт того, что к ней пристыковано большинство из имеющихся в составе ДССП библиотек:

- программа создания и поддержки параметров настройки и функционирования системы;
- программа сохранения системы и авторизации доступа;
- модуль дизассемблера;
- модуль редактирования памяти;
- модуль поддержки аппаратной части отладчика;
- сервисная оболочка системы;
- кросс-ассемблер;
- управляющая программа аппаратной части;
- модуль программатора ПЗУ.

Краткие технические характеристики АОС

1. Разрядность целевых процессоров, бит... ....4; 8
2. Объём ОЗУ программ, кбайт..... 4
3. Количество контрольных точек..... 4096
4. Количество корпусов ИМС..... 14
5. Основные режимы работы:
   - отладка программ;
   - сброс всех контрольных точек;
   - распечатка справочника по системе;
   - запись любого участка ОЗУ программ на магнитный носитель в виде файла;
   - чтение файла программы и запись его в ОЗУ программ;
   - запись-чтение внешней и внутренней памяти МКУ на магнитный искатель.
6. Режим отладки включает в себя автоматический и пошаговый режимы, а также режим контрольных точек.

7.4. Средства контроля и диагностики

По окончании этапа разработки даже на стадии выпуска опытной партии приборов на базе МКУ возникают проблемы организации контроля и диагностики. Ручной контроль чрезвычайно обременителен и ненадежен. Организация автоматизированного контроля требует разработки соответствующего оборудования. Зарубежное стоит чрезвычайно дорого. Цена систем контроля достигает нескольких тысяч долларов. Поэтому значительно дешевле разработать достаточно простые и надежные системы на базе персонального компьютера (ПК). Такая система состоит из платы параллельного интерфейса и блока контрольно-диагностического устройства (КДУ). КДУ для проверки любого цифрового устройства, имеющего не более 24 входов/выходов, реализован автором на 6 микросхемах среднего уровня интеграции. КДУ позволяет не только проводить входной контроль

микросхем, но и проверять готовые изделия на частотах порядка 500 кГц. Переход на более высокие частоты связан с введением ОЗУ и не представляет сложной проблемы.

Программная поддержка проста, но громоздкая база данных потребует значительных усилий по ее наращиванию, введению данных по новой комплектации. Однако уровень программистов для пополнения этой базы данных может быть очень низким.

В принципе на основе ПК может быть создано рабочее место разработчика, включающее в себя средства отладки и контроля. Такой инженерный комплекс автоматизации разработок (ИКАР) явился бы весьма эффективным инструментом для отечественных приборостроителей.

ЧАСТЬ 2
БАЗОВЫЕ ПРОБЛЕМЫ КЛАССИЧЕСКОЙ ЛОГИКИ

«Читай и слушай для собственного развлечения рассказы о хитроумных системах, вникай в интересные вопросы, поставленные там со всей изощрённостью, какой только может наделить их пылкая фантазия, но смотри на всё это только как на упражнения для ума и возвращайся каждый раз к согласию со здравым смыслом...»

Честерфилд «Письма к сыну»

Глава 1

Всё, о чём далее будет идти речь (комплементарная логика, решение логических уравнений, русская силлогистика, силлогистика Аристотеля-Жергонна, общеразговорная силлогистика и т. д.) разработано в России и не известно мировой науке. Поэтому призываю всех читателей воспринимать мои методы крайне критически и обязательно проверять их с точки зрения здравого смысла. Весьма показателен пример некритического отношения к теории относительности (ТО), которую к 1998г. немецкие физики Георг Галецки и Петер Маркаардт низвели с пьедестала "Тысячи" экспериментов в защиту ТО оказались фиктивными. Из 5 реальных попыток не было ни одной удачной. В СССР ещё в 60-е годы также были выступления и публикации учёных, критиковавших ТО. Наиболее ярко отношение советской науки к ТО выражено в работе В.А. Ацюковского "Логические и экспериментальные основы теории относительности".

Прежде, чем приступить к рассмотрению базовых проблем, стоит совершить небольшой экскурс в историю логики. Эта наука как основополагающий раздел философии появилась в конце второго тысячелетия до н. э. в Индии. Затем она перекочевала в Китай, где в 479–381гг до н. э. наблюдался период расцвета логики и философии, связанный с учением Мо Цзы.

Наибольшего развития логика достигает в Древней Греции. Главные её достижения связываются с именами Сократа (470–399 гг. до н.э.), Платона (428–348 гг. до н.э.), Аристотеля (384–322 гг. до н.э.), Зенона из Китиона (336–264гг. до н.э.) и Хризиппа (280–205гг. до н.э.). Следует хотя бы просто перечислить имена ученых, уделявших самое пристальное внимание логике.

Ибн-Сина (Авиценна) – среднеазиатский мыслитель с широким кругом интересов, род. в 980г. в Афшане, возле Бухары, умер в 1037г. Ему уже была известна формула импликации.

Михаил Псёлл – византийский логик (1018–1096), автор «квадрата Псёлла».

Роджер Бэкон – английский философ (1214–1294), считал в частности, что «простой опыт учит лучше всякого силлогизма», т. е. Опирался на логику здравого смысла.

Уильям Оккам – английский философ, логик (1300–1349). Ввёл троичную логику за много веков до Лукасевича. Автор «принципа простоты» ("бритва Оккама").

Антуан Арно (1612–1694) и Пьер Николь (1625–1695) – французские логики, авторы книги «Логика Пор-Рояля» (монастырь во Франции), последователи Декарте.

Арнольд Гейлинкс – бельгийский логик и философ (1625–1669). Опроверг за несколько веков до официального признания общезначимость модуса DARAPTI для 3-й фигуры силлогизмов. Доказал правила Де Моргана:

$ab \rightarrow a+b$

$(a \rightarrow b)' \rightarrow (b' \rightarrow a')'$

$(b \rightarrow c)(a \rightarrow c)' \rightarrow (a \rightarrow b)'$

$(a \rightarrow b)(a \rightarrow c)' \rightarrow (b \rightarrow c)'$

$ab' \rightarrow (a \rightarrow b)'$

Готфрид Вильгельм Лейбниц – немецкий философ, математик, физик (1646–1716). Основоположник символической логики. Впервые математизировал логику. Задолго до Эйлера использова «круги Эйлера». Впервые поставил «техническое задание» для силлогистики. Сформулировал и доказал теоремы:

$Aab\ Aac \rightarrow Aa(bc)$

$Aab\ Acd \rightarrow A(ac)(bd)$

$A(ab)a$

$A(ab)b$, т. е. все (ab) суть b

Якоб и Иоганн Бернулли (1654–1705 и 1667–1748) – ученики Лейбница. Ввели операцию вычитания множеств.

Леонард Эйлер – математик, физик, астроном (1707–1783). Родился в Швейцарии, но вся научная жизнь прошла в России. Создатель «кругов Эйлера», основы формальной силлогистики.

Иоганн Генрих Ламберт – швейцарский логик (1728–1777), последователь Лейбница. Предвосхитил ряд работ Джорджа Буля (разложение функции на элементарные составляющие), ввёл скалярные диаграммы для геометрической интерпретации силлогизмов.

Ж.. Д. Жергонн – французский астроном и логик (1771–1859). Впервые зафиксировал с помощью кругов Эйлера силлогистический базис Аристотеля.

Август Де Морган – шотландский логик (1806–1871), автор логики отношений,» правил Де Моргана».

Джордж Буль – английский логик (1815–1864), создатель Булевой алгебры. Отец Этель Лилиан Войнич (автор романа «Овод»).

Платон Сергеевич Порецкий – профессор Казанского университета. Он опередил не только своё время, но и Бертрана Рассела. П.Эренфест сказал, что Порецкий намного упростил приёмы решения логических уравнений по сравнению с Дж. Булем и Шредером. Могу добавить, что русский логик впервые в мире дал аналитическое представление силлогистическим функторам Аху и Еху. Этого не заметили ни сам Порецкий, ни зарубежные логики, ни, что самое обидное, отечественные учёные.

Н. А. Васильев – советский учёный, автор монографии «О частных суждениях», в которой впервые заявляет, что силлогистика Аристотеля не имеет никакого отношения к здравому смыслу. Сформулировал требования к силлогистическому базису здравого смысла.

1.1. Решение логических уравнений

Наиболее полно эта проблема рассмотрена в работах П.С.Порецкого [25] и Н.П.Брусенцова [5].

Предлагается более простой и эффективный метод решения логических уравнений, основанный на применении таблиц истинности и трёхзначной или четырёхзначной логики. Трёхзначную логику представим следующими базисными операциями: инверсией, конъюнкцией и дизъюнкцией [5].

Таблица базисных функций 3-значной логики

Аргументы	Функции		
XY	X'	X&Y	X+Y
00	1	0	0
0i	1	0	i
01	1	0	1
i0	i	0	i
ii	i	i	i
i1	i	i	1
10	0	0	1
1i	0	i	1

Аргументы	Функции		
XY	X'	X&Y	X+Y
11	0	1	1

Базисные функции определяются следующими соотношениями:

X&Y = min(X,Y);
X+Y = max(X,Y).

Автором впервые предлагается четырехзначная логика. Она полностью соответствует общеразговорной, или бытовой логике. Вышеназванная логика представлена базисными функциями. Значения этой логики имеют следующий смысл: 0 – нет, j – не может быть никогда, i – может быть, 1 – да.

Таблица базисных функций 4-значной комплементарной логики

XY	X'	X&Y	X+Y	XY	X'	X&Y	X+Y
00	1	0	0	i0	j	0	1
0j	1	0	j	ij	j	0	1
0i	1	0	i	ii	j	i	i
01	1	0	1	i1	j	i	1
j0	i	0	j	10	0	0	1
jj	i	j	j	1j	0	j	1
ji	i	0	1	1i	0	i	1
j1	i	j	1	11	0	1	1

Следует обратить вниманив на комплементарность (взаимодополняемость, взаимоинверсность) значений переменных: 0+1=1, i+j=1, 0&1=0, i&j=0. В связи с этим вполне естественно назвать такую логику комплементарной. Для приведённых базисных функций комплементарной логики, как и для 3-значной логики также справедлив закон Де Моргана.

При решении системы логических уравнений вначеле определяется так называемая полная адиница задачи (системы), а потом отыскивается решение уравнения относительно одной из переменных. Под решением здесь и далее понимается преобразование исходного уранения к явному виду относительно одной из переменных. Поскольку построение полной единицы системы не вызывает затруднений, рассмотрим решение логического уравнения с помощью таблиц истинности, считая полную единицу (m) известной.

Пример 1

Дано: m = ab + cd = 1
Найти: d = f(a,b,c)

Решение

На основании исходного логического уравнения полной единицы строим таблицу истинности для разрешённых наборов, т.е. тех наборов, на которых исходное уравнение имеет решение. Перенеся столбцы a,b,c из исходной таблицы в качестве значений аргументов, а столбец d – в качестве значений искомой функции, получим таблицу истинности для d = f(a,b,c).

dcba	m
0011	1
0111	1
1011	1
1111	1
1100	1
1101	1
1110	1

cba	d
011	0
11	0
011	1
111	1
100	1
101	1
110	1

По полученной таблице заполним карту Карно, откуда после минимизации выведем соотношения для d = f(a,b,c). Если на некотором наборе функция принимает значение как 0, так и 1, то в соответствующую клетку карты Карно вписываем символ i. Если на каком либо наборе функция не определена, то в соответствующую клетку карты Карно вносим значение j. Здесь и далее апостроф означает отрицание аргумента или функции. Применение карты Карно не имеет принципиального значения: просто автор считает карты Карно наиболее эффективным инструментом для минимизации булевых функций.

ba
c \ 00 01 11 10

j	j	i	j
1	1	i	1

Клетки карты Карно с координатами 011 и 111 заполнены значением i, т.к. на этих наборах (индивидах, конституентах) d принимает значения как 0, так и 1. Наборы 000, 001 и 010 в таблице отсутствуют, поскольку при таких значениях аргументов исходное уравнение не имеет решения, поэтому соответствующие карты Карно заполнены символом j.

Для трёхзначной логики в этих клетках помещается прочерк [16], т.е. символ недоопределённости. Доопределение минимизируемой функции единицами позволяет получить компактную формулу.

Для комплементарной логики имеем:

d = cb' + ca' + iba + j(c'b' + c'a')

Для трёхзначной логики это уравнение выглядит проще:

d = b' + a' + iba

В связи с тем, что при решении логических уравнений приходится зачастую проводить минимизацию булевых функций от большого числа переменных, полезно ознакомиться с соответствующими алгоритмами, изложенными в [16,17] и в диссертации автора [18].

Пример 2

Рассмотрим 1-ю задачу Порецкого [25]. Между птицами данного зоосада существует 5 отношений:

1. Птицы певчие – крупные или обладающие качеством Y.

2. Птицы, не имеющие качества Y - или не крупные, или не имеют качества X.

3. Птицы певчие в соединении с крупными объединяют всех птиц с качеством X.

4. Каждая не-крупная птица есть или певчая, или обладающая качеством X.

5. Между птиц с качеством X совсем нет таких птиц с качеством Y, которые не будучи певчими, были бы крупные.

Определить, были ли птицы качества X певчие или нет. Узнать то же в отношении птиц качества Y. Найти, были ли среди качества X птицы качества Y и наоборот.

Решение

Пусть X – птицы качества X.
Y – птицы качества Y.
S – певчие птицы.
G – крупные птицы.

Тогда условие задачи будет представлено следующими рекурсивными уравнениями [25]:

1. s= (g+ y)s ;
2. y'= (g'+x')y' ;
3. s+g+x'=1
4. g'=(s+x)g';
5. xys'g=0

Эти уравнения Порецкий через эквивалентность приводит к единичной форме:

1. g+y+s'=1
2. g'+x'+y=1
3. s+g+x'=1
4. s+g+x=1
5. x'+y'+s+g'=1

Кстати, используя силлогистические функторы Axy и Exy (см. главу 3), можно получить эти соотношения сразу, не прибегая к рекурсии и эквивалентности:

1.As(g+y) = (s(g+y)')' = s'+g+y = 1
2. Ay'(g'+x') = (y'(g'+x')')' = y+g'+x' = 1
3. Ax(s+g) = (x(s+g)')' = x'+s+g = 1
4. Ag'(s+x) = (g'(s+x)')' = g+s+x = 1
5. Ex(ys'g) = (x(ys'g))' = x'+y'+s+g' = 1

Поэтому, видимо, целесообразно изучать решение логических уравнений после освоения силлогистики.

Полная логическая единица всей задачи определится как конъюнкция всех левых частей системы логических уравнений. Эту рутинную операцию можно заменить на менее утомительную процедуру построения дизъюнкции нулей. Получим систему:

1. g'y's=0
2. gxy'=0
3. g's'x=0
4. g's'x'=0
5. gs'xy=0

Полный логический нуль системы равен дизъюнкции всех левых частей системы логических уравнений. Проведём решение

задачи Порецкого с использованием карты Карно, а потом сопоставим результаты. Заполним карту Карно нулями в соответствии с нулевыми термами системы, а в оставшиеся клетки впишем единицы. Тогда полная логическая единица всей задачи после минимизации примет вид:

m=sy+gx'

xy gs	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	1	1	1	0
10	1	1	0	0

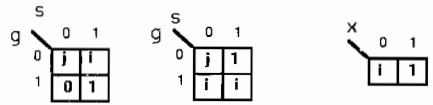
Выпишем из карты Карно все единичные термы в виде таблицы истинности. По полученной таблице построим таблицы для x=f1(g, s) ,y=f2(g,s) и y=f3(x). Если на каком-либо наборе функция принимает значение как 0, так и 1, то в соответствующую клетку карты Карно вписываем 1. Если какой-нибудь набор отсутствует, то для этого набора в карту Карно вносим значение j при комплементарной логике или произвольное (по аналогии с двузначной логикой)- при трёхзначной.

gsxy	m
0101	1
0111	1
1101	1
1111	1
1000	1
1001	1
1100	1

gs	x
01	0
01	1
11	0
11	1
10	0
10	0
11	0

gs	y
01	1
01	1
11	1
11	1
10	0
10	1
11	0

x	y
0	1
1	1
0	1
1	1
0	0
0	1
0	0



После минимизации получим для комплементарной логики системы уравнений:

x = is + jg's'
y = g's + ig + jg's'
y = x + ix' = (x + ix) + ix' = x + i

Трёхзначной логике соответствует система уравнений:

x = is
y = g' + ig + g' + i
y = x + ix' = x + 1

Результаты, полученные Порецким:

x = xs
y = gy + g's
y = y + x

Сравнивая системы уравнений, можно предположить, что Порецкий фактически использовал трёхзначную логику для решения логических задач. Причём сделал это впервые в мире. Незначительные расхождения результатов связаны с тем, что Порецкий непроизвольно доопределил y=f(g,s) нулём на наборе g's', что менее логично с точки зрения минимизации. Результаты Порецкого имеют право на существование, однако более строгим решением является система на основе комплементарной логики, поскольку она фиксирует и те ситуации, которые не могут быть никогда. Например, в сложной системе управления своевременное обнаружение таких состояний может предотвратить аварию или отказ. Поэтому можно надеяться, что вычислительная техника (да и не только она, но и юриспруденция тоже) будет строиться на трёхзначной [7] или комплементарной логике. Кстати, первая в мире троичная ЭВМ «Сетунь-70» была создана в России Н.П. Брусенцовым (МГУ). Что касается четырёхзначной ЭВМ, то аппаратная реализация комплементарной логики на современной двоичной элементной базе весьма несложна.

Основываясь на примерах 1 и 2, составим алгоритм решения системы логических уравнений.

1.2. Алгоритм «Селигер»

1. Привести систему уравнений к нулевому виду (исходная система).

2. Заполнить карту Карно нулями в соответствии с термами левых частей исходной системы уравнений, а в оставшиеся клетки вписать единицы. Эти единичные термы представляют собой СДНФ полной единицы системы.

3. Произвести минимизацию совокупности единичных термое. Полученное соотношение представляет МДНФ уравнения полной единицы системы.

4. Построить сокращённую (только для единичных термов) таблицу истинности уравнения полной единицы и выписать из неё все значения входных и выходных переменных в виде частных таблиц истинности для искомых функций.

5. Произвети минимизацию искомых функций.

Алгоритм «Селигер» предполагает не только графическую, но и аналитическую минимизацию методом обобщённых кодов [5]. Для систем уравнений с числом аргументов не более 10 графический метод эффективнее. Минимизация в трёхзначной и комплементарной логиках для двоичных аргументое несущественно отличается от минимизации в двузначной: нужно лишь проводить раздельное склеивание по i, j, 1 или 0.

Пример 3

Рассмотрим 2-ю задачу Порецкого.

Относительно белья в комоде известны 2 положения:

1) часть его состояла из крупных предметов. всё же остальное было тонким, причём часть этого последнего была поношена, прочая часть дёшево стоила;

2) всё бельё не тонкое, а также всё бельё не новое, но дорогое, принадлежало или к такому тонкому белью, которое не было ни крупно, ни дорого, или же к такому крупному белью, которое частью было ново, частью же, будучи тонким, было дёшево.

Узнать, какое бельё было поношено: крупное или мелкое.

Решение

Пусть a – тонкое, b – крупное, c – дорогое, d – новое бельё. Тогда имеем следующую систему уравнний:

1. b + a(d' + c') = 1
2. (a' + d'c) = ab'c' + b(d + ac')

В соответствии с алгоритмом «Селигер» получим:

1. a'b' + b'cd = 0
2. a'b' + a'd' + cd' + 0

Нулевые термы системы уравнений занесём в карту Карно, откуда получим функцию полной единицы. По полученному соотношению строим сокращённую таблицу истинности и выписываем из неё значения b и d в виде таблицы, из которой получаем логи-

ческую функцию. Из этой функции следует, что d не зависит от b, что совпадает с результатом Порецкого.

ab \ cd	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	1	1	1	0
10	1	1	0	0

Рис.6

abcd	m
0101	1
0111	1
1101	1
1111	1
1000	1
1001	1
1100	1

b	d
0	1
1	1
0	0
1	0

При решении логических уравнений иногда возникает необходимость в равносильных преобразованиях. Эта проблема рассмотрена в [3] и решена путём введения понятий парного и непарного индивидов. Однако чёткое определение этих понятий отсутствует.

Докажем равносильность преобразований с помощью традиционных методов. Пусть x, y - логические функции, для которых справедливо уравнение x = y, приведённое через эквивалентность к виду.

(x = y) = xy + x'y'

Требуется найти парный индивид z, т.е. такую функцию, добавление (удаление) которой к обеим частям равенства x=y не нарушало бы его равносильности. Это требование записывается в виде уравнения.

x + z = y + z

Но полученное выражение через формулу эквивалентности приводится к виду:

(x+z=y+z) = (x+z)(y+z)+(x+z)'(y+z)' = xy+x'y'+z = (x=y)+z

Откуда следует, что z должен принадлежать исходному равенству, и парным индивидом для исходного уравнения x=y может быть любой терм уравнения или их комбинация, т.е. в данном случае парными индивидами являются термы xy, x'y', xy+x'y'

x = y
x +xy = y + xy
x + x'y' = y + x'y'
x + xy + x'y' = y + xy + x'y'

Пример 4

Пусть x+a = y+b. Найти парные термы (индивиды).

Решение

Развернём исходное уравнение на основе формулы эквивалентности:

(x+a=y+b) = (x+a)(y+b)+(x+a)'(y+b)' = xy+ay+bx+ab+a'b'x'y'

Из (20) соледует, что парными термами являются xy, ay, bx, ab, a'b'x'y' и все их комбинации, которые более наглядно можно представить на карте Карно.

xa \ yb	00	01	11	10
00	1	0	0	0
01	0	1	1	1
11	0	1	1	1
10	0	1	1	1

Любой набор термов из единичных фигур покрытия [5] карты Карно может быть парным индивидом. Определим общее количество парных индивидов n. Пусть u – количество переменных, входящих в уравнение f1=f2. Тогда количество конституент единицы k определится следующими соотношениями:

Для чётного u: k = 1 + [2^(u/2) - 1]^2.

Для нечётного u: k = 1+(2^[(u+1)/2]-1).

Эти соотношения легко выводятся из анализа карты Карно для уравнения x+a = y+b. Общее количество парных индивидов равно сумме биноминальных коэффициентов для бинома степени k без единицы:

n = 2^k - 1

На основе метода. заложенного в алгоритме «Селигер», можно вывести соотношения для операций, обратных конъюнкции и дизъюнкции. Поскольку эти операции часто называются соответственно логическими умножением и сложением, то логично обратным операциям присвоить имена логического деления и логического вычитания. Впервые формулы для логического частного и логической разности для троичной логики получены Н.П.Брусенцовым [5].

Если логическое уравнение вида z = f(x1, x2, x3, ..., xi, ..., xn) решается относительно одной из своих переменных, например, отыскивается обратная функция x1 = fi(z, x2, x3, ..., xi, ..., xn), то можно воспользоваться более простым алгоритмом «Селигер С» решения задачи.

Алгоритм «Селигер-С»

1. Построить таблицу истинности для уравнения z=f(x1, x2, ..., xn).
2. По исходной таблице истиннсти построить таблицу истинности для обратной функции вида x1=fi(z, x2, ..., xn) простой перестановкой столбцов z и x1.
3. По полученной таблице истинности построить обратную функцию x1=fi(z, x2, ..., xn) и провести её минимизацию.

Пример 5

Дано: z = xy , v = x + y

Найти: y = z/x , y = v-x

Решение

На основе формулы эквивалентности преобразуем исходную формулу z=xy. Тогда получим (z=xy) = zxy + z'(x'+y'). В соответствии с пп.4, 5 алгоритма «Селигер» получим y = xz+ix'z'+jx'z.

Решим ту же задачу посредством алгоритма «Селигер-С». Исходные уравнения представим в виде таблицы истинности. Тогда в соответствии с п.2 алгоритма «Селигер-С» построим частные таблицы истинности для y = z/x и y = v-x.

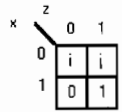
xy	z	v
00	0	0
01	0	1
10	0	1
11	1	1

xz	y=z/x
00	i
01	j
10	0
11	1

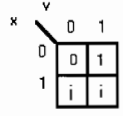
xv	y=v-x
00	0
01	1

10	j
11	i

В соответствии с п.3 алгоритма «Селигер-С» проведём минимизацию искомых функций в трёхзначной и комплементарной логиках.



y = z/x



y = v-x

Для комплементарной логики получим:

y = z/x = xz + ix'z' + jx'z

y = u-x = x'v + ixv + ixv'

Для трёхзначной логики уравнения имеют вид:

y = z/x = xz + ix'

y = v-x = x'v + ix

Однозначным и более строгим решением являются уравнения комплементарной логики.

Пример 5

Дана система логических уравнений:

ax = bc

bx = ac

Найти x.

Решение

Напрашивается простой и "очевидный" метод решения: сложить левые и правые части уравнений. В результате получим ax+bx = ac+bc. Откуда x = c, a = b. Ответ настораживает, поэтому проверим его на основе разработанных алгоритмов. По алгоритму "Селигер":

M = (ax = bc)(bx = ac)

M' = (ax ⊕ bc) + (bx ⊕ ac) = abcx+a'b'+b'x'+a'c'+c'x'+a'x'+b'c'.

После занесения M'в карту Карно получим

M = cx(a'b+ab')+ab(c'x+cx').

Откуда решение системы логических уравнений в соответствии с алгоритмом «Селигер» примет вид:

x = c(a'b+ab')+j(a'b'+a'c'+b'c').

Подтвердим корректность метода на решении более прозрачной задачи.

Пример 6

Дана система логических уравнений:

x = y

u = v

Найти решение системы.

Решение

M = (x = y)(u = v) = (xy + x'y')(uv + u'v') = u'v'(x'y' + xy)+uv(x'y' + xy

По алгоритму «Селигер» получим:

y(x,u,v) = x(u = v)+j(u ⊕ v)

Для перехода к y(x) достаточно заменить в полученном выражении прямые значения избыточных переменных единицами, а инверсные – нулями. В результате мы подтвердим исходное уравнение системы y(x) = x.

1.3. Отыскание обратных функций

Используя алгоритм «Селигер» или «Селигер-С», можно получить полную систему обратных функций для двоичной логики. В таблице приведена полная система функций двоичной логики.

xy	z0	z1	z2	z3	z4	z5	z6	z7	z8	z9	z10	z11	z12	z13	z14	z15
00	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
01	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
10	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
11	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Перестановкой столбцов y и z исходной таблицы строим таблицу истинности для полной системы обратных функций.

xz	y0	y1	y2	y3	y4	y5	y6	y7	y8	y9	y10	y11	y12	y13	y14	y15
00	i	i	i	i	0	0	0	0	1	1	1	1	j	j	j	j
01	j	j	j	j	1	1	1	1	0	0	0	0	i	i	i	i
10	i	0	1	j	i	0	1	j	i	0	1	j	i	0	1	j
11	j	1	0	i	j	1	0	i	j	1	0	i	j	1	0	i

Из таблицы обратных функций получаем полную симметричную систему обратных функций, а по алгоритму «Селигер» –

y = f(x):	
y0 = iz'+jz	y0 = j
y1 = xz+ix'z'+jx'z	y1 = x+jx'
y2 = xz'+ix'z'+jx'z	y2 = jx'
y3 = i(xz+x'z')+j(xz'+x'z)	y3 = ix+jx'
y4 = x'z+ixz'+jxz	y4 = x'+jx
y5 = z	y5 = 1
y6 = xz'+x'z	y6 = x'
y7 = x'z+ixz+jxz'	y7 = x'+ix
y8 = x'z'+ixz'+jxz	y8 = jx
y9 = xz+x'z'	y9 = x
y10= z'	y10 = 0
y11= x'z'+ixz+jxz'	y11 = ix
y12= i(xz'+x'z)+j(xz+x'z')	y12 = ix'+jx
y13= xz+ix'z+jx'z'	y13 = x+ix' - импликация
y14= xz'+ix'z+jx'z'	y14 = ix'
y15= iz+jz'	y15 = i

Кстати, переход от левой системы уравнений к правой легко выполняется простой заменой z на 1 и z' на 0.

Заключение

1. Простота метода, заложенного в алгоритме «Селигер», позволяет решать логические уравнения от большого числа переменных.

2. Минимизация функций в 3-значной и комплементарной логиках для двоичных аргументов несущественно отличается от традиционных методов двузначной логики.

3. Парные термы для равносильных преобразований определяются набором термов, полученных на основе применения формулы эквивалентности к исходному логическому уравнению.

4. Применение метода при выводе обратных логических функций показало, что однозначное решение для двоичных аргументов может быть получено лишь в комплементарной логике.

5. Впервые получены все 16 обратных логических функций для двух аргументов.

6. Комплементарная логика при аппаратной реализации позволяет значительно упростить решение проблемы самодиагностирования вычислительной техники: например появление j на любом выходе может свидетельствовать о сбое или отказв.

Глава 2
ЗАКОНЫ ЛОГИКИ СУЖДЕНИЙ

Автор не открывает здесь ничего нового, но излагая данный материал хочет показать всю простоту аналитических выводов данных законов, следовательно и их никчёмность: незачем заучивать полтора десятка правил, если доказательство столь примитивно. Всё дело в том, что в классической логике доказательство построено на громоздком аппарате таблиц истинности и словесной казуистике [9, 13, 22, 28]. Трудно назвать грамотным такое решение проблемы. Инженерная логика использует более совершенный инструмент для анализа и синтеза законов. Воспользуемся перечнем законов из [9].

1.Закон исключённого третьего: p или неверно, что p.
В переводе на язык логики этот закон выглядит так:
$p + p' = 1.$
Это тривиальное равенство, не требующее доказательства.

2.Закон непротиворечивости: неверно, что [p и не p].
На языке логики:$p \& p' = 0.$
Это равенство верно по определению.

3.Закон двойного отрицания: если [не (не p)],то p.
Необходимо доказать, что $(p')' \rightarrow p = 1.$ Доказательство основано на двойном отрицании и импликации:
$(p')' \rightarrow p = p \rightarrow p = p' + p = 1.$

4.Обратный закон двойного отрицания: если p, то [не (не p)].
$p \rightarrow (p')' = p' + p = 1.$

5.Закон контрапозиции: если (если p, то q),то [если(не q), то(не p)].
$(p \rightarrow q) \rightarrow (q' \rightarrow p') = (p' + q) \rightarrow (q + p') = pq' + p' + q = 1.$

6.Законы, характеризующие конъюнкцию.
6.1.Если (p и q), то (q и p):$pq \rightarrow qp = (pq)' + pq = 1.$
6.2.Если (p и q),то p:$(pq) \rightarrow p = (pq)' + p = p' + q' + p = 1.$
6.3.Если p и q, то q: $(pq) \rightarrow q = (pq)' + q = p' + q' + q = 1.$
6.4.Если p, то [если q,то (p и q)]:$p \rightarrow (q \rightarrow pq) = p' + q' + pq = 1.$

7.Законы импликативных силлогизмов.
7.1.Если [(если p, то q) и (если p,то r)],то [если p, то(q и r)].
$[(p \rightarrow q)(p \rightarrow r)] \rightarrow (p \rightarrow qr) = [(p' + q)(p' + r)]' + p' + qr =$
$= (p'+qr)'+p'+qr = 1.$
7.2.Если [(если p, то q) и (если r,то s)], то [если(p и r),то (q и s)]

$[(p \to q)(r \to s)] \to (pr \to qs) = [(p'+q)(r'+s)]'+p'+r'+qs =$
$= pq'+rs'+p'+r'+qs = 1.$

7.3. Если [(если p, то q) и (если q,то r)], то (если p,то r).
$[(p \to q)(q \to r)] \to (p \to r) = pq'+qr'+p'+r = 1.$

7.4. Если [(если p, то q) и (если r,то q)], то [если (p или r),то q].
$[(p \to q)(r \to q)] \to [(p+r) \to q] = pq'+rq'+p'r'+q = 1.$

8. Законы, характеризующие дизъюнкцию.

8.1. Если (p или q), то (q или p).
$(p+q) \to (q+p) = (p+q)'+(p+q) = 1.$

8.2. Если (p или q),то (если не p,то q).
$(p+q) \to (p' \to q) = p'q'+p+q = 1.$

Алгоритм инженерного анализа законов логики суждений чрезвычайно прост:

1) произвести замену всех знаков импликации на символы дизъюнкции в соответствии с известной формулой $x \to y = x' + y$;

2) привести полученное выражение к ДНФ;

3) занести ДНФ в карту Карно и убедиться, что она вся покрыта единицами – это свидетельствует об истинности проверяемого закона или суждения.

Как видит читатель, такие законы можно «изобретать» десятками. Во всех выводах применялась аналитическая минимизация логических функций. Однако значительно проще для этой цели использовать карты Карно.

Задача 2.1.

Рассмотрим задачу из [9] о крокодиле. Когда крокодил похитил ребёнка одной египтянки и та попросила его не есть ребёнка, то крокодил ответил: "Я верну тебе ребёнка, если ты отгадаешь, что я с ним сделаю". Найти ответ египтянки.

Решение.

В [9] даётся пространное, на 5 страницах, словесное толкование различных ситуаций. Решим эту задачу аналитически.

Обозначим через x - "крокодил съест ребёнка", через y - ответ египтянки: "Ты съешь ребёнка". Тогда условие крокодила будет описано следующей формулой:

$[(x \sim y) \to x'][(x \oplus y) \to x] = ((x \oplus y) + x')((x \sim y) + x) =$
$= (xy'+x'y+x')(x'y'+xy+x) = (x'+y')(x+y') = y'$

Следовательно, условие крокодила непротиворечиво лишь при ответе: "Ты не съешь ребёнка". Значит, египтянка должна ответить: "Ты съешь ребёнка" - тогда крокодил умрёт от противоречий.

Аналогично решается задача о путнике на мосту, которого за правдивый ответ должны повесить, а за ложный - утопить.

Задача 2.2.

В тёмной комнате находятся 3 мудреца. На столе лежат 2 белых и 3 чёрных шляпы. Каждый мудрец надевает наугад одну из шляп, затем все "кильватерной колонной" выходят в освещённое помещение. 3-й мудрец видит шляпы 1-го и 2-го мудрецов, 2-й - только шляпу 1-го. На вопрос о цвете шляп 3-й и 2-й мудрец ответили: " Не знаю". Что сказал 1-й мудрец?

Решение

Пусть x_1, x_2, x_3 означают, что чёрные шляпы надеты соответственно 1-м, 2-м и 3-м мудрецами. Ответ 3-го мудреца означает, что на 1-м и 2-м - не белые шляпы

$(x_1' x_2')'$. Если бы на первом мудреце была белая шляпа, то 2-й по ответу 3-го определил бы, что на нём чёрная шляпа. Т. к. 2-й мудрец не нашёл ответа, то имеем $(x_1' x_2)'$ В итоге получим: $(x_1' x_2')'(x_1' x_2)' = (x_1 + x_2)(x_1 + x_2') = x_1$ Значит, на первом мудреце чёрная шляпа.

Глава 3
БАЗИСЫ СИЛЛОГИСТИКИ

Современная логика суждений давно вызывает неудовлетворенность как своим несоответствием Аристотелевой логике [1], так и нечеткостью описания с точки зрения математической логики. Введение кванторов не разрешило этих проблем.

Рассмотрим вначале логику непосредственных умозаключений [13]. Для выражения любого умозаключения или посылки достаточно двух конструкций (в скобках представлена краткая форма записи суждений):

1) Все X суть Y(Axy);
2) Некоторые X суть Y(Ixy);

Однако традиционно в логике используются 4 базовых суждения (силлогистических функтора):

1)Все X суть Y(Axy);
2)Ни один X не есть Y(Exy);
3)Некоторые X суть Y(Ixy);
4)Некоторые X не суть Y(Oxy).

Из диаграмм Венна могут быть тривиально получены следующие соотношения:

Axy = (xy')' = x'+y
Exy = (xy)' = x'+y'

Здесь и далее апостроф означает отрицание.

Эти соотношения известны давно и ни у кого не вызывают сомнений. Что касается суждений Ixy, Oxy, то здесь сложилась спорная ситуация. Во-первых, ни в одном источнике нет аналитического представления силлогистического функтора (квантора[29]) Ixy, т.е. фактически нет аналитического описания базиса силлогистики. Это и понятно: для решения данной задачи требуется многозначная логика. В классической силлогистике все авторы стремились использовать двузначную логику. Во-вторых, здравый смысл и булева алгебра утверждают, что Oxy = (Ixy)', а в традиционной логике [13] Oxy = (Axy)' и Ixy = (Exy)', что отнюдь не бесспорно и не убедительно. Однако примем на веру эти формулы, поскольку именно их рекомендуют для запоминания студентам.

На этом основании мы получим следующие формулы для Ixy, Oxy:

Ixy = (Exy)' = xy
Oxy = (Axy)' = xy'

Прежде всего, эти соотношения противоречат друг другу. По **определению** "Некоторые X суть Y" и "Некоторые X не суть Y" **взаимно инверсны**, т.е. Ixy = (Oxy)', Oxy = (Ixy)'. А из приведённых **формул следует** эквивалентность суждений "Некоторые X не суть Y" и "**Некоторые** X суть не Y", что совсем не соответствует действительности. Кроме того, частноотрицательное суждение вообще не **имеет самост**оятельного смысла, поскольку является тривиальным **отрицанием** частноутвердительного высказывания.

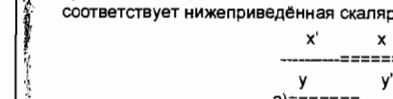
Выборочная проверка при помощи кругов Эйлера "правильных" **модусов** EIO 1-й – 4-й фигур, EAO, OAO 3-й фигуры и AAI, EAO 4-й фигуры также подтвердила всю несостоятельность **соотношений** Ixy, Oxy. Аналитический метод контроля силлогизмов дал такие же результаты.

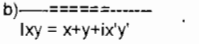
Неудовлетворенность трактовкой частных суждений высказывалась еще русским логиком Н.А.Васильевым [8]: "...частное суждение представляет для логики значительные трудности, употребление его полно двусмысленности"

Попытаемся прояснить содержательный смысл соотношения (3), из которого следует, что безусловно существуют лишь ситуация x=y=1. Круги Эйлера не в состоянии отобразить такое суждение. Поскольку логические аргументы представляют собой скаляры, максимальная длина которых не может превышать "полной единицы" (универсума), т.е. x+x'=1, введем понятие скалярных диаграмм и заменим ими круги Эйлера. Необходимо отметить, что впервые геометрическую интерпретацию (интервальный метод изображения множеств) силлогистических функторов применил Иоганн Генрих Ламберт (1728–1777), немецкий философ, математик, физик и астроном. Однако он допустил ряд ошибок, главной из которых явилось отсутствие фиксации универсума. Эта ошибка на несколько столетий похоронила идею математической силлогистики.

```
      x
  ======
      y
  ======
  Ixy = xy
```

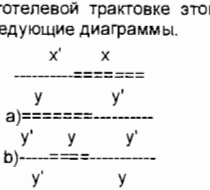
Из рисунка видно, что такая "логика" не имеет никакой практической ценности. "Бытовой" логике, вероятно, более всего соответствует нижеприведённая скалярная диаграмма.





Скалярная диаграмма не только определяет суждение Ixy как пересечения множеств X и Y, но и отмечает различные ситуации этого пересечения. Все аналитические соотношения получены на основе четырёхзначной комплементарной логики.

В аристотелевой силлогистике под Ixy понимается любая комбинация понятий x,y, лишь бы пересечение этих понятий не было пустым [1]. Аристотелевой трактовке этого суждения соответствуют скалярные следующие диаграммы.



Вновь введенные скалярные диаграммы отличаются от диаграмм Ламберта [28] следующими принципиальными характеристиками:

1) наличие фиксации универсума;

2) размещение силлогистического функтора Exy на двух, а не на одном уровне;

3) возможность "дробного" (разрывного) представления понятия в пределах универсума;

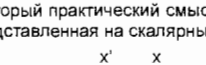
4) возможность графической и аналитической (4-значной комплементарной) интерпретации результатов анализа и синтеза силлогизмов.

Наличие даже одного из перечисленных отличий привело к переименованию кругов Эйлера в диаграммы Венна.

С аристотелевским определением частного суждения Ixy не согласны многие логики. В работе [8] автор утверждает, что "научное употребление слова "некоторые" совпадает с общеразговорным", т.е. с бытовым, а не аристотелевским. Кроме того, Н.А.Васильев считает, что Ixy и Oxy должны считаться одним суждением.

Он также заявляет: "В математике так называемые частные суждения сводятся ... к общим, и она прекрасно обходится без этого нелепого в совершенной науке слова "некоторые" К этому же должна стремиться и всякая наука... Частное суждение нужно рассматривать вовсе не как какой-то вывод из общего суждения, а как особый вполне самостоятельный вид суждения, вполне координированный с общими суждениями, исключающий их и исключаемый любым из них" С точкой зрения такого известного ученого трудно не согласиться.

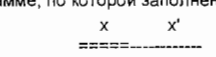
Имеет некоторый практический смысл и такая трактовка суждения Ixy, как представленная на скалярных диаграммах.



Под базисом силлогистики будем понимать всевозможные варианты представления суждений Axy, Exy, Ixy. Суждение Oxy получается автоматически из Ixy, поскольку является его отрицанием.

3.1. Все x суть y (Axy)

1.Традиционное представление этого суждения изображено на скалярной диаграмме, по которой заполнена таблица истинности.



xy	Axy
00	1
01	1
10	0
11	1

По таблице истинности синтезируем логическую функцию Axy:

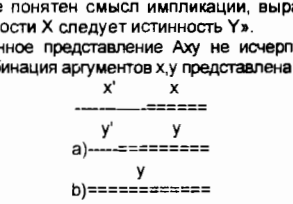
Axy = (xy')' = x'+y = Ay'x' = Exy' = (x→y) = (y'→x')

(Axy)' = xy'

Здесь уместно сделать одно замечание. Много копий было сломано при выяснении физической сущности импликации. Из таблицы истинности этот смысл не вырисовывался, более того, вызывал недоумение. Но ведь x → y = x' + y = Axy. А если все X суть Y,

то в этом случае понятен смысл импликации, выраженный в суждении «из истинности X следует истинность Y».

2.Традиционное представление Axy не исчерпывает все ситуации. Вторая комбинация аргументов x,y представлена на диаграмме.



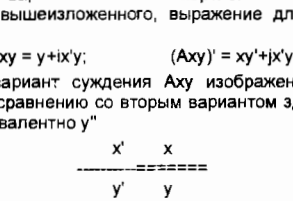
xy	Axy
00	i
01	1
10	0
11	1

Ситуация, представленная на рисункв под символом b, может быть проиллюстрирована следующим высказыванием: "Все люди смертны". Это справедливо при условии, что "мир" (универсум) – все живые существа, т.к. все живое - смертно.

С учвтом вышеизложенного, выражение для функции Axy примет вид:

Axy = y+ix'y; (Axy)' = xy'+jx'y'

3.Третий вариант суждения Axy изображен на скалярных диаграммах. По сравнению со вторым вариантом здесь добавлено суждение "x эквивалентно y"



xy	Axy
00	i
01	i
10	0
11	1

Для ситуации на рисунке под символом с справвдливо высказывание "Все люди владеют словом" Если весь "мир" – живые

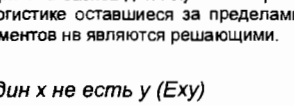
существа, то понятия "люди" и "говорящие живые существа" эквивалентны. Из таблицы получаем следующее соотношение:

Axy = xy+ix'
(Axy)' = xy'+jx'

Эти три варианта базиса для Axy не исчерпывают всех ситуаций, но в силлогистике оставшиеся за пределами рассмотрения комбинации аргументов нв являются решающими.

3.2. Ни один x не есть y (Exy)

1.Классическое прадставление Exy изображено на скалярных диаграммах.

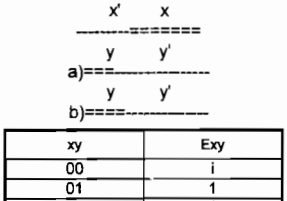


xy	Exy
00	1
01	1
10	1
11	0

Из таблицы имеем:

Exy = (xy)' = x'+y' = Axy' = Ayx' = Eyx = (x→y') = (y→x')
(Exy)' = xy

2.Второй вариант суждения Exy представлен на рисунке.



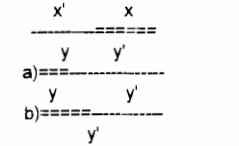
xy	Exy
00	i
01	1
10	1
11	0

Для иллюстрации диаграммы рисунка под символом b подходит высказывание "Ни один живой не есть мертвый".

Из таблицы имеем:

Exy = x'y+xy'+ix'y'
(Exy)' = xy+jx'y'

3.Третий вариант суждения Exy изображен на скалярных диаграммах.



xy	Exy
00	i
01	1
10	1
11	0

Высказывание "Ни один человек не бессмертен" иллюстрирует ситуацию на диаграмме под символом с. Здесь "мир" - живые существа, а бессмертных существ не бывает.

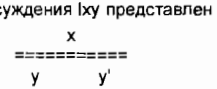
Из таблицы выводим соотношение:

Exy = xy'+ix'
(Exy)' = xy+jx'

3.3. Некоторые x суть y

Н.И.Лобачевский создал "воображаемую гвометрию". По образу и подобию великого русского геометра не менее великий русский логик Н.А.Васильев разработал "воображаемую логику". Мы попробуем разобраться хотя бы в общеразговорной (бытовой) логике, тем более что в [8] частному суждению Ixy уделено недостаточное внимание.

1.Первый вариант суждения Ixy представлен на рисунке.



xy	Ixy
00	0
01	0
10	1
11	1

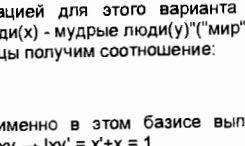
Иллюстрацией для этого варианта служит высказывание "Некоторые люди(x) - мудрые люди(y)"("мир" – люди).

Из таблицы получим соотношение:

Ixy = x
(Ixy)' = x'

Кстати, именно в этом базисе выполняется требование Васильева [8] Ixy → Ixy' = x'+x = 1.

2.Второй вариант суждения Ixy представлен на рисунке.

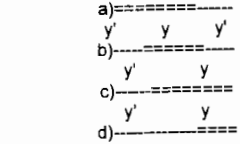


xy	Ixy
00	i
01	1
10	1
11	1

Из таблицы получим соотношение:

Ixy = x+y+ix'y'
(Ixy)' = jx'y'

3.Третий вариант суждения Ixy представлен на рисунке.
Этот базис соответствует Аристотелевскому.

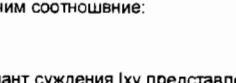


xy	Ixy
00	i
01	i
10	i
11	1

Из таблицы получим соотношвние:

Ixy = xy+i(x'+y')
(Ixy)' = j(x'+y')

4.Четвёртый вариант суждения Ixy представлен на рисунке. Этот базис получил название несимметричного.



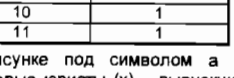
xy	Ixy
00	1
01	i
10	1
11	1

Ситуация на рисунке под символом а иллюстрируется высказыванием "Некоторые юристы (x) – выпускники юридических вузов (y)" (не-юристов юридические вузы не выпускают).

Из таблицы получим соотношение:

Ixy = x+y'+ix'y
(Ixy)' = jx'y

5.Пятый вариант суждения Ixy представлен на рисунке.



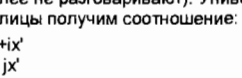
xy	Ixy
00	i
01	i
10	1
11	1

Ситуация на рисунке под символом с иллюстрируется высказыванием "Некоторые люди(x) суть нговорящие существа(y)" (не-люди тем более не разговаривают). Универсум – "живые существа".

Из таблицы получим соотношение:

Ixy = x+ix'
(Ixy)' = jx'

6.Шестой вариант суждения Ixy представлен на рисунке.

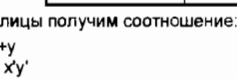


xy	Ixy
00	0
01	1
10	1
11	1

Из таблицы получим соотношение:

Ixy = x+y
(Ixy)' = x'y'

7. Седьмой вариант функтора Ixy выглядит так:



xy	Ixy
00	i
01	1
10	i
11	1

lxy = y + iy'
(lxy)' = jy'

8. Восьмой вариант функтора lxy (базис Васильева Н.А.).

В работе [8] утверждается, что в общеразговорном базисе из lxy обязательно следует lxy', т.е. lxy → lxy'. Попытаемся решить это логическое уравнение с целью синтеза суждения lxy, удовлетворяющего критерию Васильева.

lxy → lxy' = (lxy)'+lxy' = 1

Из анализа всех возможных вариантов базиса lxy ясно, что на наборах 00, 01, 10 искомая функция lxy принимает неизвестнын значения a, b, c. На наборе 11 функция lxy строго определена: её значение равно 1. Таким образом, имвем следующее соотношение:

lxy = ax'y'+bx'y+cxy'+xy

Из двух предыдущих соотношений на основе формулы Де Моргана, а ещё лучше карты Карно, получим выражение:

a'x'y'+b'x'y+c'xy'+0+ax'y+bx'y'+cxy+xy' = 1

Полученные значения на основании последнего соотношения занесем в карту Карно и рвшим систему трех уравнений с тремя неизвестными.

\y x\	0	1
0	a' ---- b	b' ----- a
1	c' ----- 1	0 ----- c

Из КК получаем систему уравнений:

a'+b = 1
b'+a = 1
c = 1

Решая систему, получаем следующие корни уравнений:

1) a = b = 0
2) a = b = 1
3) a = b = i

Этим корням соответствуют следующие формулы для lxy:

1) lxy = x
2) lxy = x+y+x'y' = 1
3) lxy = x+ix'

Уравнение (1) соответствует первому базису для lxy, уравнение (3) – третьему базису, а уравнение (2) – общеразговорному базису.

```
   x'      x
---------======
y'    y     y'
---=======-----
```

9.Девятый вариант суждения lxy представлен на рисунке.

```
   x'      x
---------======
   y       y'
a)=======-----
  y'    y    y'
b)---======----
    y'    y
c)-----========
    y'      y
d)-----------====
```

xy	lxy
00	1
01	i
10	i
11	1

Из таблицы получим соотношение:

lxy = xy+x'y'+i(xy'+x'y)
(lxy)' = j(xy'+x'y)

Вопрос о выборв базиса должен решаться отдельно для каждого конкретного силлогизма. Нередко частноутвердительное суждвние бездумно употреблявтся вместо общеутвердительного. Если для суждения "Некоторые животные – млекопитающие" мы будем использовать любой симметричный базис, то придем к абсурдному заключению "Некоторые млекопитающие – животные", поскольку на самом деле исходное суждение имеет вид "Все млекопитающие – животные". Именно такую ошибку дважды допустили преподаватели Кэмбриджа и Оксфорда, авторы хорошего учебного пособия по философии, на стр.170 и 174 [29].

Для указания используемого базиса автор применяет нумерацию, состоящую из вариантов суждений в порядке Axy-Exy-Ixy. Например, для анализа силлогизмов в общем (неконкретном) виде автор предпочитает общвразговорный базис 1-1-2, который описывается следующими соотношвниями:

Axy = (xy')'
Exy = (xy)'
Ixy = x+y+ix'y' = x+y+i

Этот базис назван автором русским базисом, т.к. он частично удовлетворяет требованиям русского логика Н.А.Васильеева относительно научного и общеразговорного смысла силлогистического функтора Ixy. Вполне естественно, что силлогистика, основанная на русском базисе, названа русской силлогистикой.

Заключение

1.Анализ современного состояния логики показал полное отсутствие аналитического представления базиса силлогистики, а такжв несостоятельность классического силлогистического базиса который не является ни Аристотелевским, ни общеразговорным (бытовым).

2.Впервые показано, что даже общие суждения имеют неоднозначную структуру и аналитическое описание.

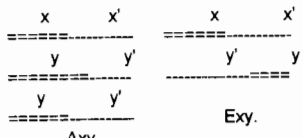
3.Впервые представлено все многообразие базиса частноутвердительного суждения и дано его аналитическое представление.

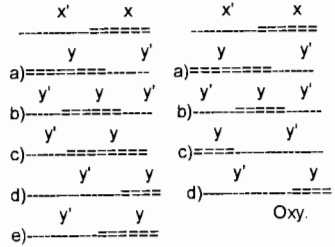
4.Впервые найдены аналитические выражения для всех частноутвердительных суждений, удовлетворяющих критерию Васильева.

Глава 4
СИЛЛОГИСТИКА АРИСТОТЕЛЯ - ЖЕРГОННА

В [28] приведены так называемые "жергонновы отношения" С помощью этих отношений Ж.Д.Жергонн (1771-1859) представил все классы суждений (силлогистические функторы), выделенные Аристотелем, на языке теории множеств. Автор пока не может дать однозначного заключения о корректности проделаннной Жергонном операции. Поэтому данная силлогистика носит двойное имя.

Переведем "жергонновы отношения" на язык скалярных диаграмм.





По скалярным диаграммам были построены соответствующие таблицы истинности.

xy	Axy
00	1
01	i
10	0
11	1

xy	Exy
00	1
01	1
10	1
11	0

xy	Ixy
00	i
01	i
10	i
11	1

xy	Oxy
00	i
01	i
10	1
11	i

Из таблиц истинности получаем следующие соотношения:

"Все X суть Y": Axy = xy+x'y'+ix'y
"Ни один X не есть Y": Exy = x'+y'= (xy)'
"Некоторые X суть Y": Ixy = xy+i(xy)'
"Некоторые X не суть Y": Oxy = xy'+i(xy')'

В связи с тем, что при проверке силлогизмов потребуются отрицания функций,то на основе формулы Де Моргана выведем следующие формулы:

(Axy)' = xy'+jx'y
(Exy)' = xy
(Ixy)' = j(xy)'
(Oxy)' = j(xy')'

Такой же результат может быть получен табличным методом, для чего необходимо проинвертировать значения соответствующих силлогистических функторов в таблицах. Полученные соотношения позволяют построить силлогистику без кванторов. Очень интересные решения этой проблемы имеются е [5,14,15,26]. Известны попытки решения задач силлогистики с помощью кванторного аппарата исчисления предикатов [21]. Однако, судя по современному состоянию силлогистики, эти попытки успеха не имели. Это обстоятельство ставит под сомнение всё исчисление предикатов, тем более, что введение кванторов противоречит принципу «бритвы Оккама». С помощью формул для силлогистических функторов A, E, I, O можно выполнять все операции над силлогизмами, т.е. находить аналитическое решение задач, связанных с силлогизмами. Для того, чтобы проверить силлогизм, нужно выполнить алгоритм "Осташ-Т"

4.1. Алгоритм "Осташ-Т" (тест)

1.Заменить посылки и заключение выражениями в соответствии с формулами для функторов A, E, I, O.

2.Получить выражение в видв конъюнкции всвх посылок, имплицирующей заключвние.

3.Проверить зто выражение на тождественность единице, занеся его в карту Карно (КК). Если выполняется тождественность единице, то заключение истинно. Если хотя бы одна из посылок или заключение являются частным суждением, то силлогизм является истинным даже при получении модальной единицы (т.е. в некоторых клетках КК проставлены символы модальности i) при условии, что m=1 или m'=1 (в этом случав строка m или соответственно m' должна содержать не менее 3-х целых единиц и только одну составную, т.е. 1=i+j). В противном случав заключение не имеет места.

Для синтеза заключения по заданным посылкам также можно использовать алгоритм "Осташ-Т", несколько изменив его.

Алгоритм "Осташ-С" (синтез)

1.Заменить посылки выражениями в соответствии с формулами для функторов A, E, I, O.

2.Получить выражение в видв конъюнкции всех посылок и проинвертировать его. Занести полученное выражение в карту Карно (КК).

3.Доопределить полученную функцию одним из выражений для силлогистических функторов A, E, I, O таким образом, чтобы получить тождественую или модальную единицу. При доопредвлении иметь в виду, что из частной посылки должно следовать частное заключение. Перед доопрвделением в одной строке КК (m или m') должно быть не менее 2-х, а после доопределения не менее 3-х целых единиц. Доопределяемое заключение должно содержать минимально необходимое количество единиц. Функция доопределения является искомым заключением. Если в доопределяемой строке КК имеется 2 полных единицы и 2 значения j, то доопределение невозможно.

4.Если вышеуказанное доопределение невозможно, то из данных посылок нельзя вывести никакого заключения.

Синтез посылок от синтвза заключений отличается лишь тем, что доопределение КК выполняется в этом случае для отрицания посылки.

Поскольку все законы логики суждений и вся силлогистика основаны на импликации, то неплохо бы, наконец, разобраться с этой функцией с точки зрения здравого смысла. Дело в том, что соотношение x→y = x'+ y известно всем, но физического смысла из него не видно. Продолжим преобразование формулы импликации:

x→y = x'+y = (xy')' = Axy.

Таким образом, импликация соответствует утверждению «Все X суть Y». А отсюда уже понятен и вполне обоснован смысл суждения «Если X истинно, то Y безусловно истинно», или «Из истинности X безусловно следует истинность Y». Это хорошо видно из скалярных диаграмм и кругов Эйлера.

Аналитические методы на основе алгоритмов "Осташ-Т" и "Осташ-С" дополняются графическим методом на базе скалярных диаграмм.

4.2. Алгоритм «ТВАТ» (графический синтез силлогизмов)

1.Изобразить все возможные ситуации для исходных посылок с помощью скалярных диаграмм.

2.Занести в таблицу истинности все значения f(x,y) для входных наборов xy: 00, 01, 10, 11.

3.Выполнить минимизацию логической функции заключения f(x,y) в четырёхзначной комплемвнтарной логике.

4.Полученный результат представить в виде силлогистического функтора в соответствии с известным базисом.

Простота графического алгоритма анализа и синтеза силлогизмов наводит на мысль о том, что и скалярные диаграммы, и алгоритм могли быть открыты 23 века назад Аристотелем. Во всяком случае, скаляры были известны Евклиду, совремвннику Аристотеля.

Алгоритмы «Осташ» и «ТВАТ» дают одинаковые по полноте и корректности результаты. Существует более простой и эффективный аналитический метод, позволющий получать корректные, но для некоторых частных силлогизмов не всегда полные результаты. Этот метод оформлен автором в виде алгоритма «ИЭИ» (Ивановский энергетический институт). Предпочтительная область применения данного алгоритма – силлогистика здравого смысла,

т.е. русская и общеразговорная. Кроме того, алгоритм «ИЭИ» незаменим при аналитическом синтезе соритов (многопосылочных силлогизмов). Правда, графический синтез соритов даёт более осязаемые результаты и за более короткое время.

4.3. Алгоритм "ИЭИ "(синтез заключения)

1.Заменить посылки выражвниями в соответствии с формулами для функторов A, E, I, O.

2.Получить выражение для полной единицы M системы в виде конъюнкции всех посылок.

3. Получить из M функцию M(x,y), заменив средний член m или m' на 1. Если средний член m/m' входит в силлогизм автономно, то заменить его на i. Полученная функция M(x,y) является заключением силлогизма. Если конъюнкция im/im' входит в M автономно, то заключения не существует.

Алгоритм «ИЭИ» можно считать частным случаем алгоритма «Селигер» для решения логических уравнений.

Используя приведённые методы, проверим некоторые модусы для 4-х фигур категорического силлогизма. Синтвз силлогизмов проввдём графическим методом в связи с его простотой и наглядностью. В результате получим следующие заключения. Здесь и далее под обозначением N.n понимается номер фигуры и номер модуса в данной фигуре. Например, 1.6 означает 8-й модус первой фигуры.

Фигура 1.

1.1. AmxAym → f(x,y) = mx'+jm'x+m'y+jmy'+f(x,y) = 1

xy	f(x,y)
00	1
01	0
10	i
11	1

Алгоритм «ТВАТ» и алгоритм "Осташ-С" дали одинаковый результат: f(x,y) = xy+x'y'+ixy' = Ayx.

Для алгоритма «ИЭИ» получим:

M = AmxAym = (m'x'+mx+im'x)(y'm'+ym+iy'm) = m'x'y'+ixy'+mxy
M(x,y) = x'y'+xy+iy'x = Ayx

Таким образом, все три алгоритма дали одинаковый результат.

1.6. EmxEym → f(x,y) = mx+my+f(x,y) = 1(i)

xy	f(x,y)
00	1
01	i
10	i
11	i

f(x,y) = x'y'+i = Ix'y'.

По алгоритму «ИЭИ»

M = EmxEym = (m'+x')(y'+m') = m'+x'y'
M(x,y) = x'y'+i = Ix'y'

Фигура 2.

2.4. AxmOym → f(x,y) = m'x+jmx'+j(m'y)'+f(x,y) = 1(i)

xy	f(x,y)
00	i
01	1
10	i
11	i

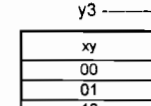
f(x,y) = Ix'y.

По алгоритму «ИЭИ»

M = AxmOym = (x'm'+xm+ix'm)(ym'+iy'+im) = m'x'y+im+ix'y'
M(x,y) = x'y+i = Ix'y

Фигура 3.

3.2. AmxEmy → f(x,y) = mx'+jm'x+my+f(x,y) = 1(i)

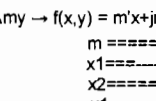


xy	f(x,y)
00	i
01	i
10	1
11	i

f(x,y) = xy'+i(xy')' = Ixy'.

Фигура 4.

4.1. AxmAmy → f(x,y) = m'x+jmx'+my'+jm'y+f(x,y) = 1



xy	f(x,y)
00	1
01	i
10	0
11	1

f(x,y) = xy+x'y'+ixy' = Axy.

У Аристотеля этому модусу соответствует заключение Ixy, что не согласуется ни со здравым смыслом, ни с формальным выводом. Кроме того, из анализа фигур 1 и 4 видно, что они идентичны, а следовательно, должны давать одинаковые модусы. Например, модусу AII 1-й фигуры должен соответствовать модус IAI 4-й фигуры, модусу EIO 1-й фигуры – модус IEO 4-й фигуры. Таких несоответствий между модусами 1-й и 4-й фигур насчитывается не менее четырёх. Указанные несоответствия можно было бы заметить 23 века назад, поскольку для этого не требуется ничего, кроме начального образования.

4.5. ExmAmy → f(x,y) = mx+my'+jm'y+f(x,y) = 1(i)

m =====----------
x ----------===
y1=========---
y2=====--------
y3=====---====

xy	f(x,y)
00	i
01	1
10	i
11	i

f(x,y) = Ix'y.

В результате полной проверки традиционных 64-х силлогизмов получим сладующие правильные модусы:

1-я фигура: AAA, AEO, AII, EAE, EEI, EII, IEO, OEI.
2-я фигура: AAI, AEE, AOI, EAE, EEI, EII, IEI, OAI.
3-я фигура: AAI, AEI, AII, AOI, EAI, EEI, EII, EOI, IAI, IEI, OAI, OEI.
4-я фигура: AAA, AEE, EAO, EEI, EIO, EOI, IAI, IEI.

Кстати, на самом деле проверять нужно было бы 256 модусов даже для случая двухфункторной (A,I) силлогистики.

Полученные результаты очевидны, однако в большей своей части данные модусы являются абсолютно новыми для аристотелевой силлогистики [13]. Кромв того, аристотелевский модус AAI а 4-й фигурв является некорректным.

Проверим теперь традиционную логику [13] с помощью алгоритмов «Осташ». Её базис явно отличается от базиса Аристотеля-Жергонна. Попробуем описать этот базис аналитически. Из логического каадрата [13] следуют традиционные соотношения:

Axy = (Oxy)', Exy = (Ixy)'.

Поскольку Axy = (xy')', то Oxy = xy'. Для Ixy определяем формулу, исходя из того, что Exy = (xy)'. Откуда получаем Ixy = xy.

Разумеется, подобный базис никакого отношения к здравому смыслу нв имеет. Тем не менее, проверим на основе этого базиса некоторыв традиционные "правильные" модусы. Проверку проведём а соответствии с алгоритмом "Осташ-Т"

1-я фигура

EIO: (mx)'my → xy' = mx+m'+y'+xy' ≠ 1

2-я фигура

EIO:(mx)'ym → xy' = mx + m'+y'+xy' ≠ 1

3-я фигура

EAO:(mx)'(my)' → xy' = mx+my'+xy' ≠ 1
OAO:mx'(my')' → xy' = m'+x+my'+xy' ≠ 1
EIO:(mx)'my → xy' = mx + m'+y'+xy' ≠ 1

4-я фигура

AAI:(xm')'(my')' → xy = xm'+my'+xy ≠ 1
EAO:(xm)'(my')' → xy' = mx+my'+xy' ≠ 1

Аналитическая и графическая проверки выбранных "правильных" модусов выявили некорректность последних.

Соотношения (1) – (4) описывают аристотелевскую логику, которая не соответствует требованиям, предъявленным русским ученым Н.А.Васильевым [8] к частным суждениям с научной точки зрения и с позиции логики здравого смысла.

Автор с глубочайшим уважвнивм относится к Аристотелю, впервые в истории человечества предложившему формальные методы анализа и синтеза силлогизмов. Однако нельзя признать, что логика Аристотеля являвтся логикой здравого смысла, а его «правильные» модусы исчерпывают все достоверные ситуации силлогистики. Поэтому логика Аристотеля-Жергонна представляет интерес с чисто научно-исторической точки зрения.

Наряду с этим необходимо подчеркнуть пассивную роль кванторного исчисления, предназначенного, казалось бы, для защиты аристотелевой силлогистики. В [21] приводится пример элегантного доказательства достоверности первого модуса первой фигуры (AmxAym → Axy) с применением кванторного исчисления. Однако этот модус самый примитивный из всех, и легко доказывается даже в обычной двоичной логике без привлечения кванторов («лишних сущностей» по Оккаму). Кванторный механизм создавался, в первую очередь, для того, чтобы проверить силлогистику Аристотеля. Однако до сих пор такой проверки не произошло. Отсюда можно сделать следующий вывод: либо каанторным исчислением матлогики не владеют настолько, чтобы доказать или опровергнуть правоту Аристотеля, либо само кванторное исчисление является ущербным. Автор склонявтся ко второму выводу.

Некоторые дополнительные аспекты проблем современной силлогистики изложены в [19].

Заключение

1.Предложен простой и надежный способ графической и аналитической проверки силлогизмов и синтеза заключений для любых базисов.

2.Применение предложенного метода избавляет от необходимости запоминания множества логических правил и законов.

3.Предложенный метод ставит под сомнение всё исчисление предикатов, кванторный аппарат которого не справился с задачами анализа и синтеза силлогизмов.

4.Впервые аналитически описан базис логики Аристотеля-Жергонна.

5.Впервые на основе базиса Аристотеля-Жергонна разработана силлогистика, существенно отличающаяся от классической.

6.Впервые проверены все 64 модуса силлогистики Аристотеля-Жергонна. Доказано, что аристотелев модус AAI в 4-й фигуре не является правильным.

7.Впервые доказано, что ни силлогистика Аристотеля-Жергонна, ни классическая силлогистика [13] не укладываются в прокрустово ложе 19 «правильных» модусов.

8.Доказано, что ни силлогистика Аристотеля, ни силлогистика Аристотеля - Жергонна не имеют никакого отношения к логике здравого смысла.

Глава 5
РУССКАЯ СИЛЛОГИСТИКА

Русская силлогистика построена на русском базиса, который описывает логику здравого смысла.

"Все X суть Y": v = y+x' = (xy')'

"Ни один X не есть Y": w = x'+y'= (xy)'

"Некоторые X суть Y": z = x+y+ix'y'

Необходимо заметить, что соотношения для Axy, Exy впервые были получены русским ученым П.С.Порецким методом рекурсии. При аналитическом анализе и синтезе силлогизмов потребуется отрицание для Ixy:

z'=(x+y+ix'y')'=(x+y)'(ix'y')'=x'y'(i'+x+y)=jx'y'

Выражение для z' соответствует суждению "Некоторые X не суть Y(Oxy)".

Пример 1

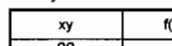
Ни один студент(x) не имеет ученой степени(m)
Некоторые ученые со степенью(m) суть математики(y)

Найти возможное заключение f(x,y)

В соответствии с пп.1 и 2 алгоритма "Осташ-С" получим:

(xm)'(m+y+im'y') → f(x,y) = xm+jm'y'+f(x,y) = 1(i)

Это уравнение справедливо при f(x,y) = x'+ix=x'+i, что соответствует суждению Ix'yIx'y' в русском базиса.



xy	f(x,y)
00	1
01	1
10	i
11	i

Из скалярных диаграмм в соответствии с алгоритмом ТВАТ также следует f(x,y)=x'+ix=Ix'yIx'y'.

Пример 2

Проведём синтез заключения для сорита [26,с.165] на основе алгоритма «ИЭИ», определяя взаимосвязь между a и b.

M = EcdAadAbc = (cd)'(ad')'(bc')' = a'b'c'+b'c'd+a'cd'
M(a,b) = a'b'+b'+a' = a' + b' = (ab)' = Eab.

Для алгоритма «ИЭИ» не имеет значения количество посылок в сорите. Правда, при этом возрастает сложность минимизации функции M, но для программы минимизации это не проблема. Указанный недостаток с лихвой перекрывается преимуществом синтеза заключений для любых аргументов заданного сорита. В примере 2 можно найти заключение для аргументов a и c:

M(a,c) = a'c'+c'+a'c = a'+c' = Eac.

Пример 3[26,с.172]

Найти заключение для аргументов a, e заданного сорита:

M = Eab'Ecd'AehEdbEc'h.

Решение

Заменим умножение сложением по Де Моргану:

M' = ab'+cd'+eh'+bd+c'h.

Внесём в карту Карно нули для M', а в остальные клетки впишем единицы - получим:

M = bc'd'e'+a'b'c'e'h'+a'b'cd(e'+h).

Откуда:

M(a,e) = e'+a'e'+a'e'+a' = a'+e' = Eae.

В [26] этот ответ единственный, а по алгоритму «ИЭИ» можно теперь практически даром получить все заключения для пар (a,d), (a,c), (a,h), (b,c), (b,h), (b,e), (d,c), (d,h), (d,e), (c,e).

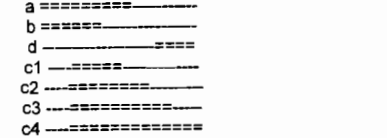
M(a,d) = d'+a'+a'd = a'+d' = Ead.
M(a,c) = c'+a'c'+a'c = a'+c' = Eac.
M(c,e) = c'e'+ce'+c = e'+c = Aec.

Пример 4

Дано: M = AbaIbcEad
Найти: M(a,c), M(b,d), M(c,d).

Решение

Найдём искомые функции графически по алгоритму ТВАТ и аналитически по алгоритму ИЭИ.



По алгоритму ТВАТ из скалярных диаграмм получены следующие таблицы.

ac	f(a,c)
00	i
01	i
10	1
11	1

bd	f(b,d)
00	1
01	1
10	1
11	0

cd	f(c,d)
00	1
01	i
10	1
11	i

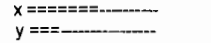
Из приведённых таблиц выводим искомые соотношения:
f(a,c) = a+i = Iac(3)Iac'(3)
f(b,d) = b'+d' = Ebd
f(c,d) = d'+i = Icd'(3)Ic'd'(3),
где цифра 3 в скобках указывает на 3-й базис, т. е.
Ixy = xy+i(x'+y').
По алгоритму ИЭИ выводим следующие формулы.
M = (b'+a)(b+c+ib'c')(a'+d') = a'b'c+abd'+acd'+ia'b'c'
M(a,c) = a'c+a+ac+ia'c' = a+c+ia'c' = Iac(2)
M(b,d) = b'+bd'+d'+ib' = b'+d' = Ebd
M(c,d) = c+d'+cd'+ic' = c+d'+ic' = Icd'(2),
где цифра в скобках указывает на 2-й базис, т. е.
Ixy = x+y+ix'y'.

Из сравнения результатов графического и аналитического синтеза заключений видно, что графический алгоритм ТВАТ даёт более полные решения.

Используя приведенные методы, проверим некоторые модусы для 4-х фигур категорического силлогизма. В результате получим следующие заключения.

Фигура 1.

1.1. AmxAym → f(x,y) = mx'+ym'+f(x,y) = 1



xy	f(x,y)
00	1
01	0
10	1
11	1

Агоритмы «ТВАТ» и «Осташ-С" дали одинаковый результат:

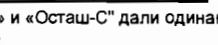
f(x,y) = (yx')' = Ayx.

По алгоритму «ИЭИ» получим:

M = AmxAym = (m'+x)(y'+m) = m'y'+xy'+mx

M(x,y) = y'+x = Ayx.

1.2. AmxEym → f(x,y) = mx'+my+f(x,y) = 1(i)



xy	f(x,y)
00	i
01	i
10	1
11	i

f(x,y) = xy'+i = Ixy'(3).

Индекс (3) говорит о том, что заключение соответствует 3-му базису, т.е. базису Аристотеля-Жергонна.

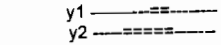
По алгоритму «ИЭИ»:

M = AmxEym = (m'+x)(y'+m') = m'y'+xy'+m' = m'+xy'

M(x,y) = xy'+i = Ixy'(3)

Фигура 2.

2.1. AxmAym → f(x,y) = m'x+m'y+f(x,y) = 1(i)



xy	f(x,y)
00	1
01	i
10	i
11	i

f(x,y) = x'y'+i = Ix'y'(3).

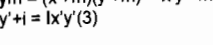
По алгоритму «ИЭИ»

M = AxmAym = (x'+m)(y'+m) = x'y'+my'+mx'+m = m+x'y'

M(x,y) = x'y'+i = Ix'y'(3)

Фигура 3.

3.2. AmxEmy → f(x,y) = mx'+my+f(x,y) = 1(i)



xy	f(x,y)
00	i
01	i
10	1
11	i

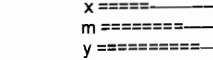
f(x,y) = xy'+i = Ixy'(3).

M = AmxEmy = (m'+x)(m'+y') = m'+xy'

M(x,y) = xy'+i = Ixy'(3)

Фигура 4.

4.1. AxmAmy → f(x,y) = m'x+my'+f(x,y) = 1



xy	f(x,y)
00	1
01	1
10	0
11	1

f(x,y) = x'+y = (xy')' = Axy.
M = AxmAmy = (x'+m)(m'+y) = m'x'+x'y+my
M(x,y) = x'+y = Axy.

После синтеза всех 64 силлогизмов получим следующие правильные модусы:

1-я фигура: AAA, AEI[3], AII[3], AOO, EAE, EEI[3], EII[3], EOO, IAI[3], IEI[3], IOO, OAO, OEO, OIO, OOO.

2-я фигура: AAI[3], AEE, AII[3], AOO, EAE, EEI[3], EII[3], EOO, IAI[3], IEI, IOO, OAO, OEO, OIO, OOO.

3-я фигура: AAI, AEI[3], AII[3], AOO, EAI[3], EEI[3], EII[3], EOO, IAI, IEI[3], IOO, OAO, OEO, OIO, OOO.

4-я фигура: AAA, AEE, AII(3), AOO, EAI[3], EEI[3], EII[3], EOO, IAI[3], IEI, IOO, OAO, OEO, OIO, OOO.

Заключение

1.Впервые разработан общеразговорный логический базис, названный автором русским базисом.

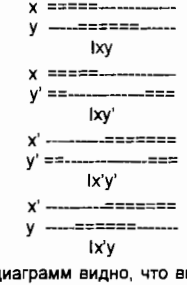
2.Впервые на основе русского базиса разработана силлогистика, названная автором русской силлогистикой.

Глава 6
ОБЩЕРАЗГОВОРНАЯ СИЛЛОГИСТИКА

В главе, посвящённой рассмотрению различных силлогистических базисов, были получены выражения для силлогистических функторов Ixy, удовлетворяющих требованию Ixy → Ixy' [8]. Наиболее интересным из них является общеразговорный функтор, выражающийся следующей формулой:

Ixy = Ixy' = Ix'y' = Ix'y = x+y+x'y' = 1.

Скалярные диграммы для всех вариантов представлена на рисунке.



Из скалярных диаграмм видно, что выполняется сладующее соотношение, которое абсолютно соответствует логике здравого смысла:

Ixy → Ixy' → Ix'y' → Ix'y.

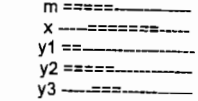
На осноее этого функтора можно построить базис общеразговорной силлогистики, которая, как и русская силлогистика, служит основанием логики здравого смысла. Предлагаемый базис соответствует всем требованиям Н.А.Васильева и имеет вид:

Axy = (xy')'
Exy = (xy)'
Ixy = x+y+x'y' = 1

Вполне естественно, что этот базис носит имя русского логика. В связи с тем, что функтор Ixy ≡ 1, аналитический синтез силлогизмов в базисе Васильева невозможен (автор не решил эту проблему). Однако графический алгоритм ТВАТ справляется с синтезом силлогизмов без затруднений. Заключения, полученные при

просмотре всех модусов в общеразговорной силлогистике, отличаются от заключений, полученных в русской силлогистике, лишь для двух модусов. Рассмотрим некоторые модусы.

1.9. ImxAym → f(x,y) = 1(i)

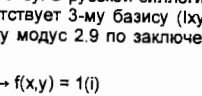


xy	f(x,y)
00	1
01	i
10	1
11	i

f(x,y) = y'+iy= Ix'y'(3)Ixy'(3).

Индекс 3 в скобках указывает на то, что функтор принадлежит аристотелеву базису. В русской силлогистике ImxAym → Ixy'(3), где индекс 3 соответствует 3-му базису (Ixy = xy+ix'+iy'). В связи с симметричностью Ixy модус 2.9 по заключению совпадает с модусом 1.9.

2.10. IxmEym → f(x,y) = 1(i)



xy	f(x,y)
00	1
01	i
10	1
11	i

f(x,y) = y'+iy = Ix'y'(3)Ixy'(3).

Здесь результаты в обоих базисах совпали.

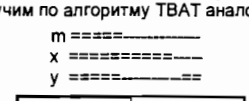
3.1. Рассмотрим силлогизм:

Все квадраты(m) суть прямоугольники(x)
Все квадраты(m) суть ромбы(y)

По алгоритму ИЭИ получим:

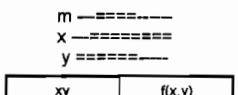
M = AmxAmy (m=xy) = (m'+x)(m'+y)(mxy+m'x'+m'y') =
= mxy+m'x'+m'y'
f(x,y) = xy+x'+y' = Ixy(8)

Если в качестве универсума используем понятие "параллелограммы", то получим по алгоритму ТВАТ аналогичный результат.



xy	f(x,y)
00	1
01	1
10	1
11	1

Если в качестве универсума выберем лишь множество, состоящее из прямоугольников, ромбов и квадратов, то получим иной результат.



xy	f(x,y)
00	0
01	1
10	1
11	1

f(x,y) = x+y = Ax'y = Ay'x

Если в силлогизме

Все люди(x) смертны(m)
Сократ(y) – смертен(m)

в качестае универсума примем множество живых существ, т. е. только смертных, то никакого заключения не получим. Если же в универсум войдут и бессмертные существа, то заключение будет выглядеть так:

f(x,y) = x'y'+i = Ix'y'(3).

Итак, заключение силлогизма зависит не только от фигур и модусов, но и от объёма универсума.

Синтез силлогизмов в базисе Васильева более прост по сравнению с базисом Аристотеля-Жергонна. Для модусов, содержащих частноотрицательные посылки, синтез заключений невозмо-

жен, поскольку не существует графического представления частно-отрицательного суждения. В результате графического синтеза заключений по алгоритму ТВАТ были получены сладующие модусы.

1-я фигура: AAA, AEI[3], AII[3], EAE, EEI[3], EII[3], IAI[3], IEI[3].
2-я фигура: AAI[3], AEE, AII[3], EAE, EEI[3], EII[3], IAI[3], IEI.
3-я фигура: AAI(AAA), AEI[3], AII[3], EAI[3], EEI[3], EII[3], IAI, IEI[3].
4-я фигура: AAA, AEE, AII(3), EAI[3], EEI[3], EII[3], IAI[3], IEI[3].

Рассмотрим несколько содержательных примеров. В [26, стр.164] дан следующий силлогизм:

Некоторые сорта герани(a) красного цвета(b).
Вса эти цветы(c) – красные(b).

Найти заключение.

Решение

M = IabAcb = c'+b
f(a,c) = c'+i = Iac'(7)

ac	f(a,c)
00	1
01	i
10	1
11	i

f(x,y) = c'+ic= Iac'(7).

Аналитический и графический алгоритмы дали одинаковые результаты. В [26, стр.165] утверждается, что заключения нет.

Проверим один из соритов Л.Кэрролла, который в [26, стр. 172] приведён к следующему виду:

Ни одно A не есть не-B.
Ни одно C не есть не-D.
Все E суть H.
Ни одно D не есть B.
Ни одно не C не есть H.

Все A есть не-E
Все E есть не-A

Решение

По алгоритму ИЭИ получим:

M = Eab'Ecd'AehEdbEc'h = AabAcdAehEbdAhc =
=(a'+b)(c'+d)(e'+h)(b'+d')(c+h').
M' = ab'+cd'+eh'+bd+c'h,

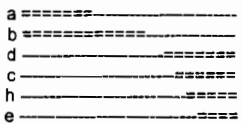
откуда с помощью карты Карно получаем:

M = a'b'c'e'h'+a'b'cdh+a'b'de'h'+bc'd'e'h'.

Из полученного выражения можно вывести несколько заключений, связывающих различные аргументы, в том числе и указанные Кэрроллом.

f(a,e) = a'e'+a'+e' = a'+e' = Aae' = Aea' = Eae

Однако данный сорит проще решается по алгоритму ТВАТ.



Из скалярных диаграмм абсолютно очевидна вся тривиальность сорита и легко выводится заключение f(a,e) = a'+e' = Eae.

Арнольд Гейлинкс – бельгийский логик и философ (1625–1669) доказал правила Де Моргана:

ab → a+b
(a → b)' → (b' → a')'
(b→c)(a→c)' → (a→b)'
(a→b)(a→c)' → (b→c)'
ab' → (a→b)'

Докажем эти правила современными методами.

ab → a+b = (ab)'+a+b = a'+b'+a+b = 1
(a → b)' → (b'→a')' = (a → b)+(b+a')' = (a'+b)+(a'+b)' = 1
(b→c)(a→c)' → (a→b)' = bc'+a'+c+ab' = 1
(a→b)(a→c)' → (b→c)' = ab'+a'+c+bc' = 1
ab' → (a→b)' = (a'+b)+(a'+b)' = 1

Готфрид Вильгельм Лейбниц сформулировал и доказал теоремы:

Aab Aac → Aa(bc)
Aab Acd → A(ac)(bd)
A(ab)a
A(ab)b, т. е. все (ab) суть b

Докажем эти теоремы на основе русской силлогистики.

AabAac → Aa(bc) = ab'+ac'+a'+bc = 1

AabAcd → A(ac)(bd) = ab'+cd'+(ac)'+bd = ab'+cd'+a'+c'+bd = 1

A(ab)a = (ab)'+a = a'+b'+a = 1

A(ab)b = (ab)'+b = a'+b'+b = 1

Заключение

1.Впервые разработан базис общеразговорной силлогистики, названный автором базисом Н.А.Васильева.

2.Впервые на основе указанного базиса разработана общеразговорная силлогистика.

3.Впервые обнаружена зависимость заключения от объёма универсума.

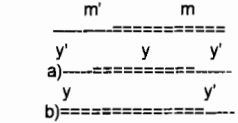
Глава 7
АТОМАРНАЯ СИЛЛОГИСТИКА

Внимательный анализ силлогизмов приводит к выводу о том, что даже базисы логики здравого смысла не всегда корректно выражают содержание посылок. Проиллюстрируем это следующим силлогизмом (см. пример 5):

Все солдаты(x) храбрые(m)

Некоторые англичане(y) храбрые(m)

Некоторые англичане - солдаты

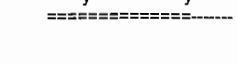


Правомерно ли использование во второй посылке русского базиса [15]? По меньшей мере, допущена некорректность по отношению к англичанам и нарушена достоверность посылки. Исходя из скалярной диаграммы для Imy(2) и полагая универсумом все человечество, приходим к выводу, что возможны ситуации, когда все трусы – англичане. Это несправедливо.

Правильным в этом случае будет использование базиса Васильева.

Рассмотрим посылку, которая не вписывается ни в один из базисов. Суждение "Все люди(x) смертны(y)" при условии, что универсумом являются живые существа, описывается следующей формулой: Axy = y. Посылка "Ни один живой человек(x) не есть труп(y)" также имеет нестандартное аналитическое представление: Exy = xy'+x'y. Многообразие базисов приводит к мысли о том, что разумнее иметь некий элементарный базис, на основе которого можно как из кирпичиков (атомов) строить описание любой посылки. Автор предлагает следующий "атомарный" базис:

Все X суть Y.

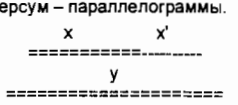


xy	f(x,y)
00	1
01	1
10	0
11	1

Axy(a) = x'+y

Иллюстрация: "Все квадраты(x) суть прямоугольники(y)". В данном случае универсум – параллелограммы.

b)



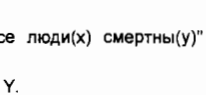
xy	f(x,y)
00	0
01	1
10	0
11	1

Axy(b) = y

Иллюстрация: "Все люди(x) смертны(y)" при условии, что универсум – существа.

Ни один X не есть Y.

a)

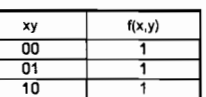


xy	f(x,y)
00	1
01	1
10	1
11	0

Exy(a) = (xy)' = x'+y'

Иллюстрация: "Ни один круг(x) не есть квадрат(y)"(универсум - геометрические фигуры).

b)



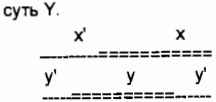
xy	f(x,y)
00	0
01	1
10	1
11	0

Exy(b) = xy'+x'y

Иллюстрация: "Ни один живой(x) не есть труп(y)"

Некоторые X суть Y.

a)

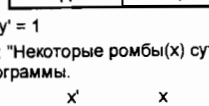


xy	f(x,y)
00	1
01	1
10	1
11	1

Ixy(a) = x+y+x'y' = 1

Иллюстрация: "Некоторые ромбы(x) суть квадраты(y)". Универсум – параллелограммы.

b)



xy	f(x,y)
00	0
01	1
10	1
11	1

Ixy(b) = x+y

Иллюстрация: "Некоторые люди(x) неграмотны(y)". Универсум – существа.

На основе атомарного базиса может быть построен любой другой. Например, функтор Ixy(2) представляет собой объединение Ixy(a), Ixy(b). Функтор Axy(3) является комбинацией функторов Axy(a), Axy(c). Все эти объединения легко выполняются с помощью скалярных диаграмм. Для фиксации и компактного описания введем операцию сцепления (конкатенации) функторов,

обозначив ее символом ||. Тогда вышеприведенные словесные описания могут быть представлены в виде следующих выражений:

Ixy(2) = Ixy(a) || Ixy(b)
Axy(3) = Axy(a) || Axy(c)

Можно ли сделать атомарный базис более компактным, более элементарным? Да, безусловно. Необходимо произвести следующие замены.

Axy(b) = Axy(a)Ax'y(a);
Axy(c) = (y=x) - равнозначность;
Exy(a) = Axy'(a);
Exy(b) = (y=x') - неравнозначность;
Ixy(b) = Ax'y(a).

Таким образом, элементарный атомарный базис в качестве фундамента имеет всего лишь два силлогистических функтора:

Axy = x'+y,
Ixy = x+y+x'y' = 1

Опишем на основе этих формул все базисы здравого смысла и базис Аристотеля.

Русский базис.

Axy(2) = Axy = x'+y
Exy(2) = Axy' = x'+y'
Ixy(2) = Ixy || Ax'y = x+y+ixy'

Базис Васильева.

Axy(8) = Axy = x'+y
Exy(8) = Axy' = x'+y'
Ixy(8) = Ixy = x+y+x'y' = 1

Базис Аристотеля-Жергонна.

Axy(3) = Axy || (x=y) = xy+x'y'+ix'y
Exy(3) = Axy' = x'+y'
Ixy(3) = Ixy || Ax'y || Axy || Ayx || (x=y) = xy+i(x'+y')
Oxy(3) = Ixy || Ax'y || Axy' || Ayx = xy'+i(x'+y) = Ixy'(3)

Для синтеза силлогизмов в атомарном базисе пригодны все разработанные автором алгоритмы: "Осташ", "ИЭИ", "ТВАТ".

Пример 1

Все люди(m) смертны(x)
Некоторые люди(m) неграмотны(y)

Найти f(x,y)

Решение

В данном случае универсум – существа.

M = Amx(b)Imy(b) = x(m+y) = xm+xy
f(x,y) = xy+x = x = Ayx(b)

m ==========----------
x ==========
y ----------==========

xy	f(x,y)
00	0
01	0
10	1
11	1

f(x,y) = x = Ayx(b).

Индекс в скобках указыеает вариант базиса. Базис заключения может быть не только атомарным, но и смешанным (русский, общеразговорный, Аристотеля и т.д.). Базис посылок, как правило, должен быть атомарным.

Рассмотрим синтез соритов, т.е. многопосылочных силлогизмов. Никаких проблем здесь не существует, если логик хорошо знает карту Карно или метод обобщенных кодов для минимизации логических функций [11, 12]. При числе посылок более 10 разумнее использовать программы минимизации для любого ПК.

Пример 2

Пусть в атомарном базисе в варианте "a" задан сорит из 6 посылок:

M = AabAbcAcdAdeAexExy = (ab')'(bc')'(cd')'(de')'(ex')'(xy)'.

Найти заключения для связи различных аргументов.

Решение

Перемножать все эти функторы слишком утомительно. Инженерная логика в таких ситуациях использует формулу де-Моргана и работает с M'.

M' = ab'+bc'+cd'+de'+ex'+xy.

Заполнив карту Карно для M', сразу из нее получим выражение для M:

M = a'b'c'd'(e'x'+xy') + dexy'(a'b'+bc).

Отсюда можем получить заключение для любых аргументов.

Вся операция занимает не более 5 мин при условии, что под рукой бланки карт Карно на 6-8 переменных.

f(a,y) = a'+a'y'+y'a'+y' = a'+y' = Eay

f1(a,x) = a'x'+a'x+xa'+x = a'+x = Aax

f2(b,d) = b'd'+db'+db = b'+d = Abd и т.д.

Все заключения получены в атомарном базисе (вариант "a").

Пример 3

Пусть первые 5 посылок сорита заданы в атомарном базисе, а шестая - в русском.

M = AabAbcAcdAdeAexIxy

Найти заключение f(a,y).

Решение

Используя решение предыдущего примера для f1(a,x),получим:

M = AabAbcAcdAdeAexExy = AaxIxy = (a'+x)(x+y+ix'y') =
=x+a'y+ia'x'y'

f(a,y) = a'y+i = Ia'y(3).

Заключение получено в 3-м (Аристотелевом) базисе. Скалярные диаграммы подтверждают полученные результаты.

Выводы

1. Анализ силлогистик здравого смысла (русской и общеразговорной) привел к выводу о том, что наряду с использованием этих силлогистик необходимо построение атомарной силлогистики.

2. Впервые разработана атомарная силлогистика и даны методы синтеза атомарных силлогизмов и примеры их использования для решения конкретных задач.

3. Впервые представлены методы синтеза соритов.

4. Показано, что для каждой содержательной посылки нужно использовать свой конкретный базис.

Глава 8
ЕСТЕСТВЕННЫЙ ВЫВОД И КВАНТОРЫ

В главе под таким названием в [22] излагается вывод умозаключений из нескольких посылок. Это может быть непосредственное умозаключение, простой категорический силлогизм или сорит. Но суть не в названии, а в методах получения результатов. В [22] для анализа умозаключений (доказательства корректности формулы) применяются кванторы. Автор при доказательстве применяет вспомогательные выводы с достаточно обременительными правилами. Приведём пример одного такого доказательства [22, стр.299]. Необходимо проверить формулу:

$\forall x(A(x) \Rightarrow B(x)) \& \exists x\, A(x) \Rightarrow \exists x\, B(x)$

Цепочка вспомогательных выводов выглядит сладующим образом.

$\forall x(A(x) \Rightarrow B(x)) \& \exists x\, A(x)$
$A(c_1)$
$A(c_1) \Rightarrow B(c_1)$
$B(c_1)$
$\exists x\, B(x)$
$\forall x(A(x) \Rightarrow B(x)) \& \exists x\, A(x) \Rightarrow \exists x\, B(x)$

Во-первых, сложно, а во-вторых, не очевидно. Поскольку здесь налицо простой категорический силлогизм (две посылки и одно заключение), то можно применить алгоритм «Осташ-Т». Для экономии заменим A(x) на a и B(x) на b. Не применяя кванторов, получим в русском базисе следующее выражение:

$Ax(a \rightarrow b)Ixa \rightarrow Ixb = x(a \rightarrow b)' + jx'a' + \ x + b + ix'b' = 1(i),$

что доказыеает истинность исходной формулы. Более очевидным является доказательство в обычной логике суждений.

$M = (a \rightarrow b) \& ie \rightarrow ib = (a'+b)ia \rightarrow ib = iab \rightarrow ib = (iab)'+ ib =$

$= a'+b'+jab+ib = 1$

Без кванторов также можно анализировать сориты, т.е. лючения с тремя и более посылками. Из [22, стр.301] позаимствуем для доказатвльства формулу, которая без кванторов примет вид:

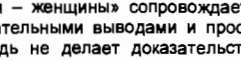
$Ax(a+b)Ix(a\rightarrow c)Ax(b\rightarrow c)\rightarrow Ixc =$

$= x(a+b)'+jx'(a'+c)'+x(b'+c)'+x+c+ix'c' = x+c+x'ac'+ix'a'c' = 1(i).$

Полученный по алгоритму «Осташ-Т» результат подтверждает достоверность анализируемого сорита. Поскольку каждый сорит

в конце концов приводится к силлогизму, то анализ и синтез соритов можно проводить по алгоритмам «Осташ». В данном сорите после приведения его к силлогизму средним термином является переменная a.

Доказательство ложности непосредственного умозаключения «поскольку все люди – мужчины или женщины, то все люди – мужчины или все люди – женщины» сопровождается в [22, стр.300] сложными вспомогательными выводами и пространными рассуждениями, что отнюдь не делает доказательство убедительным. Более того, подобные попытки обречены на неудачу, поскольку в данной ситуации требуется не двоичная, а комплементарная логика. Словесная формулировка данного умозаключения чрезвычайно аморфна. Это неотъемлемая черта любого естественного языка, с которой приходится мириться. Поэтому для анализа умозаключения прежде всего необходимо корректно аналитически представить посылку и заключение, для чего изобразим посылку на скалярной диаграмме. Здесь x – люди, m – мужчины, g – женщины.



Xmg	f(x,m,g)
000	1
101	1
110	1
111	j
100	j
001	j
010	j
011	j

Дело в том, что в посылке на основе здравого смысла предполагается исключение ситуации, когда человек является одновременно и мужчиной и женщиной (гермафродит). Кроме того, человек не может быть одновременно не мужчиной и не женщиной. И уж тем более не может быть никогда(j) мужчины или женщины нечеловека. Поэтому в таблице истинности данные ситуации отмечены как невозможные. Отсюда получаем выражение для посылки:

$f = x(mg'+m'g)+x'm'g'+j(x'm+x'g+mg+xm'g')$

Для заключения никаких ограничений не введено, поэтому не будем их придумывать. Исходя из этих соображений, получим формулу для заключения:

$z = xmg'+x'm'g'+xm'g+x'm'g' = xmg'+xm'g+x'm'g'$

По алгоритму «Осташ-Т» получим:

$f \to z = f'+z = i(x'm+x'g+mg+xm'g')+xmg'+xm'g+x'm'g' \neq 1,$

что и требовалось доказать.

На самом деле в этой задаче условие и доказательство должны были выглядеть так:

$AmxAgx \to Amx+Agx = (Amx)'+Agx)'+Amx+Agx = 1$

В примере 11.2.3.4 [22, стр. 301] требуется доказать кванторное соотношение:

$\forall x(A(x) + B(x)) \& \exists x\,(A(x) \Rightarrow C(x)) \& \forall x(B(x) \Rightarrow C(x)) \Rightarrow \exists x\,C(x).$

На основе русской силлогистики получим следующее доказательство:

$A(a+b)x\ Ix(a \to c)\ A(b \to c)x \to Ixc = (a+b)x'+jx'ac'+(b'+c)x'+x+c+i = =x'+x+c+i = 1$

На основе инженерной логики суждений доказательство выглядит ещё проще:

$(a+b)\ i(a \to c)\ (b \to c) \to ic = a'b'+ac'+j(a'+c)+bc'+ic = 1.$

В примере 11.2.3.1 [22, стр.301] заменим кванторное выражение

$\exists x\,(A(x)B(x)) \Rightarrow \exists x\,(A(x))\ \exists x\,(B(x))$

на бескванторное и проведём доказательство:

$iab \to ia\ ib = ieb \to iab = 1.$

Проведём аналогичные замены в примерах 11.2.3.2 – 11.2.3.6 [22]. Получим следующие доказательства.

11.2.3.2. $\forall x(A(x) + B(x)) \& \exists x(A(x))' \Rightarrow \exists xB(x)$
$(a+b)\ ia' \to ib = a'b'+a+ja'+ib = 1.$

11.2.3.3. $\exists x\,(A(x) + B(x)) \& \forall x(A(x))' \Rightarrow \forall xB(x)$
$i(a+b)\ a' \to b = (i(a+b))'+a+b = a'b'+j(a+b)+a+b = 1.$

11.2.3.5. $\forall x(A(x) + B(x)) \& \forall x(A(x) \Rightarrow C(x)) \& \forall x(B(x) \Rightarrow D(x)) \Rightarrow \forall x(C(x)+D(x)).$
$(a+b)(a \to c)(b \to d) \to (c+d) = a'b'+ac'+bd'+c+d = 1.$

11.2.3.6. $\exists xA(x)+\exists xB(x) \Rightarrow \exists x\,(A(x)+B(x))$
$ia+ib \to i(a+b) = i(a+b) \to i(a+b) = 1.$

ЗАКЛЮЧЕНИЕ

Автор не считает предложенные методы, алгоритмы и полученные по ним результаты истиной в последней инстанции. Однако эти результаты хорошо согласуются со здравым смыслом. Автор видит пути рееизии изложенных методов и собирается критически переосмыслить их при более благоприятных обстоятельствах. Но некоторые итоги не вызыеают сомнения:

- силлогистика Аристотеля не яеляется полной;
- некоторые «правильные» модусы Аристотеля ошибочны (наиболее очевидная ошибка - модус AAI 4-й фигуры);
- кеанторы не решают проблем анализа и синтеза силлогизмов;
- общеразговорная логика не является двоичной.

ЛИТЕРАТУРА

1. **Аристотель.** Сочинения. В 4-х томах. Т.2- М.: Мысль, 1978.
2. **Баранов С.И.** Синтез микропрограммных автоматов. - Л.: Энергия, 1974.
3. **Бахтияров К.И.** Логические основы компьютеризации умозаключений. - М.:МИИСП,1986.
4. **Брусенцов Н.П.** Диаграммы Льюиса Кэррола и аристотелева силлогистика. В кн. Выч. техника и вопросы кибернетики. Вып.13. - М.:МГУ, 1977.
5. **Брусенцов Н.П.** Начала информатики. - М: Фонд "Новое тысячелетие", 1994.
6. **Брусенцов Н.П.** Полная система категорических силлогизмов Аристотеля. В кн. Вычислительная техника и вопросы кибернетики. Вып.19. - М.: МГУ, 1982.
7. **Брусенцов Н.П.** Микрокомпьютеры. - М.:Наука, 1985.
8. **Васильев Н.А.** О частных суждениях. - Казань: Университет, 1910.
9. **Гжегорчик А.** Популярная логика. - М.:Наука, 1979.
10. **Глушков В.М.** Синтез цифровых автоматов. - М.: Физматгиз, 1962.
11. **Гутников В.С.** Интегральная электроника в измерительных приборах. - Л.: Энергия, 1974.
12. **Дербунович Л.В.** ,Лобанов В.И. Диагностическиа процессоры е системах управления технологическим оборудованием. // Энергетика. Известия вузов, №9, 1988.
13. **Кириллов В.И.** Старченко А.А. Логика. - М.: Юрист, 1995.
14. **Кулик Б.А.** Логические основы здравого смысла. - СПб.: Политехника, 1997.
15. **Кэррол Л.** История с узелками. - М.:,Мир, 1973.
16. **Лобанов В.И.** Инженерные методы разработки цифровых устройств. - М.: НИИРТА, 1977.

17. **Лобанов В.И.** Матод минимизации булевых функций от большого числа переменных с помощью карт Карно. - Инф. листок N54-87, М: МособлЦНТИ, 1987.
18. **Лобанов В.И.** Отказоустойчивый микроконтроллерный регулятор с программируемой структурой обработки данных. Диссертация нв соискание ученой степени канд. техн. наук. - Харьков, ХПИ, 1989.
19. **Лобанов В.И.** Кризис логики суждений и некоторые пути выхода из него. // Соеременная логика: проблемы теории, истории и применения е науке (Материалы V Общероссийской научной конференции) - СПб:1998.
20. **Лобанов В.И.** Адаптируемая отладочная система для проектироаания микроконтроллеров. // Приборы и системы управления, №7, 1998
21. **Логический** подход к искусственному интеллекту. - М.: Мир, 1990.
22. **Непейвода Н.Н.** Прикладная логика. - Ижевск: Удмурт. университет, 1997.
23. **Новиков П.С.** Элементы математической логики. - М.: Наука, 1973.
24. **Новиков Ю.В.** и др. Разработка устройств сопряжения. - М.:ЭКОМ, 1997.
25. **Порецкий П.С.** О способах решения логических равенств и об одном обрвтном способе математической логики. - Казань: 1881.
26. **Светлов В.А.** Практическая логика. - СПб: Изд. Дом «МиМ», 1997.
27. **Сташин В.В.** и др. Проектирование цифроаых устройств на однокристальных микроконтроллерах. - М.: Энергоатомиздат, 1990.
28. **Стяжкин Н.И.** Формирование математической логики. – М.: 1967.
29. **Тейчман Д., Эванс К.** Философия. - М.: Весь Мир, 1997.
30. **Шачнев В.А.** Математическая логика. – М.: 1991.
31. **Шестаков В.И.** Некоторые математические методы конструироаания и упрощения двухполюсных схем класса А. Диссертация на соискание ученой степени канд. физ. - мат. наук. - М.: МГУ, 1938.

СОДЕРЖАНИЕ