М. А. КАРЦЕВ

Арифметика цифровых машин

ИЗДАТЕЛЬСТВО «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
М О С К В А 1969

6П 2.15 К 27
УДК 681.142

Арифметика цифровых машин, К а р ц е в М. А., Главная редакция физико-математической литературы издательства «Наука», 1969, 576 стр.

В книге рассмотрен комплекс теоретических и практических вопросов, связанных с проектированием арифметических устройств электронных цифровых машин.

Рассмотрены способы представления чисел в машинах, влияние различных решений на количество оборудования и скорость выполнения операций в машине. Дано обобщение понятия системы счисления на дробные основания, введено понятие способов кодирования, близких к позиционным с естественными весами разрядов. Приведены способы двоичного кодирования десятичных цифр, помехозащищенные и рефлексные коды.

Далее рассмотрены сумматоры, счетчики и другие схемы выполнения элементарных операций; приводятся минимальные схемы двоичного сумматора, обобщение понятия комбинационного сумматора на случаи использования синхронных элементов, схемы счетчиков на многопозиционных полусчетных кольцах, схемы, основанные на принципе подвижных блокировок, а также способы ускорения суммирования.

Излагаются аппаратные способы выполнения сложения, вычитании, умножения, деления, извлечения корня, перевода чисел из одной системы счисления в другую и других операций. Для умножения рассмотрены логические методы ускорения, аппаратные методы ускорения в последовательных и параллельных устройствах; даны предельные оценки для логических методов и аппаратных методов 1-го порядка. Аналогичные методы рассмотрены и для деления. В заключение приведены соображения по формированию списка арифметических операций электронной цифровой машины.

Библиографических ссылок 340 назв., 101 рисунок, 16 таблиц.

2-2-4
71 69

Оглавление

Предисловие

По основному содержанию и даже по расположению глав настоящая книга напоминает предыдущую работу автора, посвященную той же теме [*]). Объем ее, однако, значительно больше.

Предметом изучения в книге является комплекс логических, математических и технических вопросов, относящихся к построению арифметических цепей электронных цифровых машин. При этом *арифметическими мы называем те цепи цифровой вычислительной машины, где непосредственно происходит образование новой информации.* Во всех других цепях машины (запоминающих, входных и выходных, управляющих) производится лишь перемещение информации во времени или пространстве, изменение физической формы сигналов и т. д., но никакая новая информация не возникает.

Разумеется, с точки зрения теории связи такое определение не выдерживает критики. Выходная информация арифметических цепей однозначно определена той информацией (исходными данными и командами управления), которая поступает на входы этих цепей. Поскольку никакой неопределенности здесь нет, то нет, казалось бы, и источника новой информации. Если строго следовать представлениям теории связи, то новая информация в вычислительной машине может возникнуть разве что в результате случайных сбоев.

Все дело в том, однако, что закон образования выходной информации чрезвычайно сложен. С точки зрения потребителя, если бы можно было заранее определить, какие данные должны получиться на выходе машины, то незачем было бы вообще пользоваться машиной. Принципиально ситуация здесь не очень сильно отличается от тех ситуаций, которые мы считаем классическими примерами случайных событий; скажем, если бы мы могли с достаточной точностью измерить начальные условия при бросании монеты и найти достаточно точное решение уравнения ее движения, то выпадение герба оказалось бы вполне детерминированным событием; однако, закономерности, влияющие на исход бросания монеты, настолько сложны, что мы предпочитаем считать это событие случайным.

Таким образом, выходная информация вычислительной машины является именно новой информацией, и источником ее внутри машины являются арифметические цепи. При этом нас не должно смущать то обстоятельство, что «новая» в указанном смысле информация возникает лишь в результате длинной последовательности вычислений, а каждая отдельная операция, выполняемая сумматором, цепью сдвигов и т. д. и даже арифметическим устройством в целом дает вполне детерминированный результат.

В этой книге мы не раз еще столкнемся с тем, что понятия и выводы теории связи нельзя однозначно переносить в вычислительную технику, как это пытаются делать иногда, хотя эти области техники действительно во многом близки между собой.

Определение наше имеет и другие недостатки. Приняв его формально, мы иногда встретились бы со значительными трудностями в решении вопроса о том, является ли некоторая конкретная цепь арифметической или нет. Скажем, числовой регистр арифметического устройства выполняет в основном функции кратковременного запоминания информации и все же включается нами в состав арифметических цепей. В то же время, скажем, дешифратор адресов в запоминающем устройстве, превращающий двоичный код адреса в позиционный, мы не относим к арифметическим цепям, поскольку целью этих операций является формирование сигналов выборки определенной ячейки памяти, а не формирование новой информации.

Тем не менее, автор надеется, что у читателей не возникнет особых сомнений относительно предмета данной книги. Фактически определение того, какие цепи называются арифметическими, не играет никакой роли в последующем изложении. Там, где это важно по существу, мы постараемся давать более точные определения. Вообще стройной и последовательной теории арифметических цепей пока не существует. Такая теория будет создана, возможно, в будущем, в пограничной между математикой и вычислительной техникой области — подобно теории информации, лежащей на границе между математикой и техникой связи, теоретической электротехнике, объединяющей физику и прикладную электротехнику, и другим дисциплинам аналогичного профиля. Пока что эта книга — одна из очередных попыток более или менее систематического изложения того, что уже сделано в этом направлении.

Естественно, что при этом на отбор материала, последовательность и методику изложения могли повлиять субъективные взгляды автора. Так, например, в книге почти не нашли отражения вопросы применения специальных мер для повышения надежности арифметических устройств. Почти не имея собственных результатов в этой области, автор не счел себя вправе просто пересказывать выводы чужих работ. Все-таки эта книга — не энциклопедия по арифметическим устройствам, хотя временами может показаться, что она претендует на эту роль.

В конце книги помещена библиография книг и журнальных статей по арифметическим устройствам и по смежным вопросам, составленная А. В. Гольдбергом. Библиография охватывает в основном работы последних 5—7 лет. Ссылки на оригинальные работы приводятся в подстрочных примечаниях в тексте.

В заключение хотелось бы выразить признательность В. А. Брику и Н. П. Брусенцову, прочитавшим в рукописи всю книгу, а также многим другим товарищам, которые читали отдельные ее части. Их предложения помогли существенно улучшить книгу. Автор был бы благодарен также за любые замечания, которые сочли бы нужным высказать читатели.

М. Карцев

---

[*]) К а р ц е в М. А., Арифметические устройства электронных цифровых машин, Физматгиз, 1958.

1. Изображение чисел

1.1. Введение

В повседневной практике для представления чисел мы пользуемся почти исключительно десятичной системой счисления. Лишь в очень редких случаях встречаются остатки других систем. Так, счет на дюжины является пережитком применявшейся некогда двенадцатеричной системы; деление часа на 60 минут, а минуты — на 60 секунд напоминает нам о шестидесятеричной системе; в редких случаях используются также римские цифры.

Особенности применяемого нами постоянно метода представления чисел (помимо того факта, что он основан на десятичной системе счисления) настолько примелькались, что часто воспринимаются как нечто само собой разумеющееся. При этом мы не всегда замечаем, что те или иные особенности в общепринятом способе изображения чисел не являются обязательным и что возможен ряд отличных решений.

Ясно, однако, что система изображения чисел, которая веками складывалась применительно к ручному счету, может оказаться далеко не наилучшей при использовании принципиально новых методов выполнения вычислений. Имеет смысл поэтому рассмотреть эту систему более подробно, обратив внимание на те ее особенности, которые не являются обязательными и допускают в принципе другие решения.

1°. Общепринятая система изображения чисел основана на *десятичной системе счисления.* Число десять есть *основание* системы счисления. Иногда считают определением десятичной системы равенство весов отдельных разрядов целым степеням числа десять (в целой части числа веса разрядов равны $1 = 10°$, $10 = 10^1$, $100 = 10^2$ и т. д., в дробной части 10^{-1}, 10^{-2} и т. д.). Аналогичным образом в системе счисления с основанием n веса разрядов должны быть целыми степенями числа n: в целой части числа веса разрядов равны $1 = n^0$, n, n^2 и т. д., в дробной части числа n^{-1}, n^{-2} и т. д. Но мы увидим в дальнейшем, что не только значения весов отдельных разрядов могут быть различными при одном и том же основании системы счисления, но и вообще не всегда каждый разряд обязан иметь свой определенный вес.

Более общим определением десятичной системы счисления является возможность использования в любом разряде одного из десяти различных символов (цифр). Приняв, что вообще основание системы счисления есть количество различных символов (цифр), допустимых для любого места, мы увидим, что в качестве основания системы счисления может быть выбрано любое целое положительное число, большее единицы (2, 3, 4, 5 и т.д.)[*]).

Основание системы счисления может быть в принципе и бесконечно велико. Так, с бесконечно большим основанием мы имеем дело при использовании римских цифр. В первом же разряде любого числа могут стоять символы:

I — единица;	C — сто;
V — пять;	D — пятьсот;
X — десять;	M — тысяча;
L — пятьдесят;	и т. д.

Не все символы, может быть, придуманы, но если мы хотим оперировать со сколь угодно большими числами, то и количество символов, которые могут нам потребоваться, неограниченно велико.

2°. Рассматривая далее общепринятый способ представления чисел, мы замечаем, что он основан на *однородной* системе счисления. Это означает, что количества допустимых символов (цифр) для всех разрядов одинаковы. Для каждого из разрядов допустимы десять цифр.

Наряду с этим возможны и *смешанные* системы — с различным количеством допустимых цифр для разных разрядов. Например, смешанной является система, применяемая для измерения времени: в разряде секунд и в разряде минут возможно по 60 разных символов (от «00» до «59»), в разряде часов — 24 разных символа (от «00» до «23»), в разряде суток — 7 разных символов (от «0» до «6» — потому что 7 суток составляют 1 неделю) и т. д. Смешанной является также система, связанная с английскими денежными единицами (12 пенсов составляют один шиллинг, 20 шиллингов — один фунт).

3°. Важно обратить внимание также и на тот факт, что в применяемой нами системе представления чисел количество различных знаков равно количеству цифр. Для каждой цифры имеется свой особый знак, всего таких знаков 10 (0, 1, 2,...,9). Такой способ изображения чисел мы будем называть *непосредственным.*

В отличие от этого возможны *кодированные* способы изображения чисел, когда количество различных знаков меньше, чем количество используемых цифр, а каждая цифра кодируется определенной комбинацией из нескольких таких знаков. Например, кодированной является упоминавшаяся выше система изображения времени: каждый из 60 символов, допустимых для разрядов минут и секунд, и каждый из 24 символов, допустимых для разряда часов, кодируется при помощи пары десятичных цифр. Всего разных знаков 10, хотя количество допустимых цифр в некоторых из разрядов намного выше.

Десятичная система тоже часто используется в кодированном виде. Например, при передаче десятичных цифр азбукой Морзе мы пользуемся фактически только тремя знаками (• точка, — тире, ; пропуск); из различных комбинаций этих знаков составляются все цифры:

---

[*]) В разделе 1.2 мы дадим более общее определение этому понятию.

0	— — — — —;	**5**	. . . .;
1	. — — — —;	**6**	— . . .;
2	. . — — —;	**7**	— — . . .;
3	. . . — —;	**8**	— — —. . ;
4	. . . . —;	**9**	— — — — .;

Если бы было заранее известно, что будут передаваться одни только цифры, то необходимости в пропусках не было бы, так как любая из цифр состоит всегда из 5 точек и тире; пропуски для обозначения концов комбинаций нужны потому, что при передаче букв возможны комбинации, состоящие из разного количества точек и тире. В телеграфном коде Бодо используется всего 2 знака (токовая или бестоковая посылка); любая цифра или буква изображается комбинацией из пяти таких знаков. В дальнейшем мы часто будем встречаться с кодированием десятичных цифр, в котором используются два знака.

4°. Важнейшей особенностью используемой нами системы представления чисел является то, что это система *позиционная.* Это означает, во-первых, что каждой цифре поставлено в соответствие некоторое число и, во-вторых, что каждому разряду приписан определенный вес. Любое число, записанное в позиционной системе, равно сумме чисел, соответствующих его цифрам, взятых с весами, соответствующими разрядам, в которых стоят эти цифры.

В применяемой нами повседневно системе представления чисел десяти цифрам поставлены в соответствие десять начальных неотрицательных чисел натурального ряда (включая нуль). Веса разрядов в целой части числа равны 1, 10, 100 и т. д., в дробной части $^1/_{10}$, $^1/_{100}$, $^1/_{1000}$ и т. д. Поэтому, например, число 3986,125 равно 3986,125 = $=3\cdot1000+ 9\cdot100+8\cdot10+6\cdot1 + 1\cdot {}^1/_{10} +2\cdot{}^1/_{100} + 5{}^1/_{1000}$.Аналогичным образом при использовании, например, двоичной системы счисления большей частью считают, что двум возможным цифрам поставлены в соответствие числа 0 и 1, а веса разрядов в целой части числа равны 1,2,4, 8 и т. д., а в дробной части $^1/_2$, $^1/_4$, $^1/_8$ и т. д. Поэтому, например, число 11001, 1011 равно

$11001,1011 = 1\cdot6 + 1\cdot8 + 0\cdot4 + 0\cdot2 + 1\cdot1 +1\cdot{}^1/_2 + + 0\cdot{}^1/_4 + 1\cdot{}^1/_8+ 1\cdot{}^1/_{16} = (25\ {}^1/_{16})_{10}.$

Вообще в позиционной системе счисления с основанием n цифры большей частью соответствуют числам 0, 1, 2 (n — 1), а веса разрядов равны: в целой части числа 1, n, n^2,..., в дробной части —n^{-1}, n^{-2}, n^{-3},... Однако ни указанный выбор чисел, поставленных в соответствие цифрам, ни указанный выбор весов разрядов не являются обязательными для позиционных систем. Те веса, которые приняты выше для различных разрядов, называются *естественными;* но возможен также и *искусственный* порядок весов. Более точные определения этих понятий будут даны в разделе 3.1.1.

Наряду с этим возможны и *символические* системы, в которых цифры вообще являются символами, каждый из которых в отдельности никак не связан с каким-либо числом; определенным комбинациям цифр условно поставлены в соответствие определенные числа. Ясно, что при этом и о весах разрядов не имеет смысла говорить. Мы встретимся также и с такими системами, в которых каждой цифре поставлено в соответствие некоторое число, но значение любого числа, записанного в этой системе, получается не суммированием цифр с определенными весами, а по более сложному закону; такие системы мы тоже будем относить к символическим.

Очевидно, что позиционный характер применяемой нами системы изображения чисел и выбор естественных весов разрядов существенным образом влияют на методику выполнения арифметических действий над числами, делая ее особенно простой и удобной для ручных вычислений. Однако для машинных вычислений и для некоторых специальных целей имеет смысл поискать, может быть, более удобные системы как среди позиционных систем с искусственными весами, так и среди символических систем.

Интересно отметить, что разделение числа на целую и дробную части имеет смысл вообще только в позиционной системе. Для указания границы целой и дробной частей мы применяем обычно специальный знак — запятую. Но когда речь пойдет об изображении чисел в вычислительной машине, этот способ нам придется пересмотреть.

5°. В качестве одной из особенностей обычного метода изображения чисел отметим еще, что в изображении отрицательных чисел записывается обычным порядком абсолютная величина числа, а перед ней ставится знак «минус». Такой способ называют *прямым кодом* для отрицательных чисел. Наряду с этим существуют и другие способы записи отрицательных чисел, с которыми мы познакомимся в разделе 1.4. и которые, как мы увидим, иногда удобнее при использовании в вычислительных машинах.

1.2. Основание системы счисления

1.2.1. Исходные положения.

В настоящее время общепринятым является следующее определение основания системы счисления: *основанием системы счисления называется количество различных символов (цифр), допустимых для каждого из разрядов.* Очевидно, что при этом основанием системы может быть любое целое положительное число, большее единицы. Наряду с приведенным автором предлагалось*) и более общее определение, которое понадобится нам при

*) К а р ц е в М. А. и Г л и в е н к о Е . В., Дробные основания системы счисления и их использование для ускорения операций в цифровых машинах. Сб. «Colloquium on the Foundations of Mathematics, Mathematical Machines and their Applications», Будапешт, 1965.

исследовании некоторых методов ускорения умножения (см. раздел 4.4). *Основанием системы счисления* предлагается называть *такую величину, логарифм которой равен количеству информации, приходящемуся на один разряд числа*; при этом все числа (все допустимые комбинации цифр) считаются равновероятными; основание, по которому берется логарифм, должно соответствовать единице измерения количества информации.

Читатель, который уже знает, что такое «количество информации», легко сообразит, что:

если все комбинации цифр допустимы, т. е. если любой возможной комбинации цифр соответствует какое-нибудь число, а все числа равновероятны, то для любого из разрядов числа все цифры тоже оказываются равновероятными и независимыми от цифр других разрядов; если различных цифр имеется n и все они равновероятны, то количество информации, заключенной в некотором разряде числа, равно одному n-ичному разряду; число, логарифм которого по основанию n равен единице, есть само n; следовательно, в этом случае старое и новое определения совпадают;

если не все комбинации цифр допустимы, то при наличии n различных цифр на каждый разряд числа будет приходиться менее одного n-ичного разряда информации; в соответствии с новым определением основание системы счисления будет меньше n; в частности, оно может оказаться и неправильной дробью (вообще любым действительным числом, большим +1).

Если читатель незнаком с понятием количества информации (оно рассматривается в разделе 1.7.), то ему придется вернуться к двум предыдущим абзацам при чтении раздела 4.4.

Во всех других разделах книги и вообще всегда, когда это не оговорено специально, мы пользуемся только общепринятым определением основания системы счисления, которое было приведено первым.

Число 10, выбранное в давние времена в качестве основания системы счисления по соображениям почти случайного характера (потому, что у человека на руках 10 пальцев), оказалось весьма удобным для ручного счета.

По существу ручной счет и не предъявляет особых требований к основанию системы счисления. Оно не должно быть слишком большим, иначе нам пришлось бы оперировать со слишком большим количеством разных цифр и производить слишком сложные вычисления в уме (например, таблица умножения одноразрядных чисел была бы слишком обширна). Но оно не должно быть и слишком маленьким, иначе запись обычных чисел получалась бы очень длинной и однообразной. Даже не очень большие числа состояли бы из большого количества разрядов, в которых часто повторялись бы одни и те же цифры. В такой записи числа трудно было бы читать и запоминать.

Позднее высказывалось мнение, что в качестве основания системы счисления удобно было бы выбрать число, имеющее много делителей, и что с этой точки зрения лучше было бы взять число 12, а не 10 (с соображениями, по которым это выгодно, мы познакомимся в разделе 4.4). Однако преимущества двенадцатеричной системы не столь велики, и давно укоренившаяся десятичная система не сдала фактически ни одной позиции.

Совсем по-иному обстоит дело, когда речь идет об электронных цифровых машинах. Создатели первой электронной цифровой вычислительной машины ENIAC использовали в своей машине без всяких изменений общепринятый способ изображения чисел. Но уже во всех последующих машинах способы изображения чисел были существенно отличны. Наиболее важную роль при выборе системы изображения чисел играют простота конструкции машины и скорость выполнения операций в ней.

Конечно, если машина получает исходные данные от человека и сообщает человеку конечные результаты вычислений, то должна быть обеспечена сравнительно простая возможность преобразования чисел из общепринятой формы их записи в ту форму, которая используется в машине, и наоборот. Но даже в машинах, предназначенных для решения различных математических задач (не говоря уже о машинах, предназначенных для управления производственными процессами), объем операций по преобразованию исходных данных и конечных результатов из одной формы в другую обычно оказывается небольшим по сравнению с общим объемом операций. Только в отдельных случаях, например при выполнении бухгалтерских расчетов, объем таких операций мог бы иметь сравнительно большой вес; если это так, то систему изображения чисел в машине стремятся максимально приблизить к общепринятой форме.

При выборе системы изображения чисел применительно к вычислительным машинам в первую очередь подвергся пересмотру выбор основания системы счисления. Правильный выбор системы счисления позволяет получить наиболее заметную экономию в количестве оборудования и наиболее значительный выигрыш по скорости.

В вычислительных машинах, строившихся до настоящего времени, чаще всего использовалась, кроме десятичной, двоичная система счисления. Некоторое применение нашли также четверичная, восьмеричная и шестнадцатеричная системы. Реже применялись троичная система и смешанная двоично-пятеричная система (в которой через один чередуются разряды с количествами допустимых символов 2 и 5). Обсуждались также возможности использования некоторых других систем.

1.2.2. Влияние выбора основания системы счисления на количество оборудования в машине.

1°. *Функция f (n).* Пусть количество различных чисел, с которым должна оперировать машина, равно *N.* Число *N* характеризует точность вычислений в машине: чем выше точность вычислений, тем с большим количеством отличных друг от друга чисел приходится иметь дело.

Если основание системы счисления равно *n,* то количество разрядов *m,* которое нужно иметь в машине, должно удовлетворять неравенству

$$m \geq \log_n N.$$

Действительно, в каждом разряде возможен один из *n* символов; количество разных комбинаций в *m* разрядах равно n^m; это число по условию должно быть не меньше *N.* Если число *N* достаточно велико, так что $\log_n N >> 1$, то можно приближенно считать, что $m \approx \log_n N$.

Предположим далее, что количество оборудования в каждом разряде пропорционально величине *n*. Например, если бы мы строили счеты в системе счисления с основанием *n,* то в каждом разряде (на каждой проволочке) было бы по n костяшек.

При этих предположениях общее количество оборудования, необходимого для изображения одного числа (любого из *N* возможных чисел) в системе счисления с основанием *n*, пропорционально произведению

$$nm \approx n \log_n N.$$

В частности, в двоичной системе ($n = 2$) количество оборудования пропорционально величине $2 \log_2 N$.

Функция

$$f(n) = \frac{n \log_n N}{2 \log_2 N} = \frac{n}{2 \log_2 n}$$

показывает, во сколько раз количество оборудования, необходимого для изображения одного числа в системе счисления с основанием *n*, больше количества оборудования, необходимого для изображения одного числа в двоичной системе, при одинаковой точности.

Рассматривая $f(n)$ как функцию непрерывного аргумента, легко найти, что в области $n \geq 1$ она имеет минимум при $n = e = 2{,}718...$ и монотонно возрастает влево и вправо от минимума. В таблице 1-1 приведены значения функции $f(n)$ для нескольких начальных целых значений *n*.

Таблица 1-1

			Функция $f(n)$						
n	2	3	4.	5	6	7	8	9	10
$f(n)$	1,000	0,946	1,000	1,078	1,148	1,247	1,333	1,420	1,505

Из таблицы видно, что с точки зрения количества оборудования наиболее выгодна троичная система счисления; близкие результаты могут дать двоичная и четверичная системы, которые между собой равноценны; десятичная система оказывается примерно в 1,5 раза расточительнее, чем двоичная или четверичная чистимы, и примерно в 1,6 раза расточительнее троичной системы.

Хотя оценки, подобные приведенным выше, можно часто встретить в литературе по вычислительным машинам, все же нужно помнить, что область применения этих оценок ограничена.

Главное ограничение связано с предположением, что количество оборудования в некотором разряде пропорционально основанию системы счисления *n.*

В действительности в течение многих лет вычислительная техника не имела в своем распоряжении таких элементов и должна была пользоваться почти исключительно элементами 2-позиционными. Только в последние годы появились достаточно устойчивые *n*-позиционные элементы, такие, сложность которых примерно пропорциональна n[*]).

Необходимо помнить, что функция *f(n)* (как, впрочем, и те оценочные функции, которые рассматриваются ниже) учитывает только оборудование в цифровых элементах и, может быть, элементах запоминания. Зависимость количества логических и других элементов схемы вычислительной машины от основания системы счисления нельзя учесть столь простыми соотношениями.

2°. *Функция* φ (*n*). Ввиду отсутствия достаточно надежных и экономичных многопозиционных цифровых и запоминающих элементов в вычислительных машинах почти никогда не прибегали к непосредственному изображению цифр систем счисления с большими основаниями. Обычно если основание системы счисления *n* больше двух, то каждая из *n* цифр кодируется определенной комбинацией нескольких двоичных цифр —подобно тому, как в телеграфном коде Бодо каждая десятичная цифра кодируется определенной комбинацией из токовых

[*]) См., например, Карцев М. А., Принцип подвижных блокировок при построении схем электронных цифровых машин, ДАН СССР т. 135, № 5, I960, стр. 1064—1076, а также раздел 2.6 настоящей книги.

и бестоковых посылок. Например, обозначив двоичные цифры символами «0» и «1», можно 10 десятичных цифр представить следующими десятью комбинациями из двоичных цифр:

0	— 0000,	5 —	0101,
1	— 0001,	6 —	0110,
2	— 0010,	7 —	0111,
3	— 0011,	8 —	1000,
4	— 0100,	9 —	1001.

При этом для изображения цифры какого-либо разряда числа в системе счисления с основанием n нам потребуется вместо одного n-позиционного элемента несколько 2-позиционных. Группа 2-позиционных элементов сможет хранить ту комбинацию двоичных цифр, которая соответствует нужной цифре системы счисления с основанием n. Пусть через $\lambda(n)$ обозначено минимальное количество двоичных разрядов, необходимых для представления одной цифры системы счисления с основанием n. Это число должно быть выбрано так, чтобы количество различных комбинаций из двоичных цифр в $\lambda(n)$ разрядах было не меньше, чем n, т. е. чтобы выполнялось неравенство $2^{\lambda(n)} \geq n$ (или $\lambda(n) \geq \log_2 n$). Ясно, что если n есть целая степень двойки, то $\lambda(n) = \log_2 n$; во всех других случаях $\lambda(n)$ — ближайшее большее целое число к $\log_2 n$. Иными словами, $\lambda(n)$ есть целое число, заключенное в интервале

$$\log_2 n \leq \lambda(n) < \log_2 n + 1.$$

Например, для изображения 10 десятичных цифр требуется по меньшей мере 4 двоичных разряда, так как $\log_2 10 = 3{,}32...$ (откуда $\lambda(10) = 4$). Всего 4 двоичные цифры могут образовать 16 разных комбинаций. Любые 10 из них могут быть любым способом поставлены в соответствие десяти десятичным цифрам. Это можно сделать либо так, как было показано выше на этой странице, либо одним из многих других способов (см. раздел 1.6.). Так или иначе 6 комбинаций будут оставаться «лишними».

Числа $\lambda(n)$ для различных значений n приведены в одной из граф таблицы 1-2.

Таблица 1-2

Функция $\varphi(n)$ и числа $\lambda(n)$										
n	2	3	4	5	6	7	8	9	10	...
$\lambda\{n\}$	1	2	2	3	3	3	3	4	4	...
$\varphi(n)$	1,000	1,262	1,000	1,294	1,148	1,069	1,000	1,262	1,204	...

Предположим по-прежнему, что точность вычислений характеризуется количеством N разных чисел, с которыми должна оперировать машина. Соответствующее количество разрядов, которыми должна располагать машина, равно $m = \log_n N$. При этом полное количество 2-позиционных элементов, необходимое для представления одного числа в системе счисления с основанием n, равно

$$\lambda(n) \cdot \log_n N.$$

В частности, в двоичной системе количество 2-позиционных элементов, необходимых для представления одного числа, равно $1 \cdot \log_2 N$.

Функция

$$\varphi(n) = \frac{\lambda(n) \cdot \log_n N}{1 \cdot \log_n N} = \frac{\lambda(n)}{\log_2 n}$$

показывает, во сколько раз количество 2-позиционных элементов, необходимых при использовании системы счисления с основанием n, больше количества 2-позиционных элементов, необходимых при использовании двоичной системы, при одинаковой точности и при кодировании цифр системы счисления с основанием n минимальным числом двоичных разрядов.

В нижней графе таблицы 1-2 приведены значения функции $\varphi(n)$ для нескольких начальных значений. Очевидно, что для любых n имеем $\varphi(n) \geq 1$. При этом $\varphi(n) = 1$ в тех и только тех случаях, когда n есть целая степень двойки; во всех остальных случаях $\varphi(n) > 1$. Таким образом, наиболее экономными с точки зрения необходимого оборудования являются системы счисления с основаниями 2, 4, 8, 16 и т. д., все остальные системы более расточительны. Это и понятно: если n не является целой степенью двойки, то $\lambda(n) > \log_2 n$ и часть тех комбинаций, которые могут быть в $\lambda(n)$ двоичных разрядах, остается «лишней»; таким образом, затраченное оборудование недоиспользуется.

Если продолжить таблицу 1-2, то можно убедиться в том, что наихудшими по затратам оборудования являются

пятеричная система (φ (5) = 1,294), а также троичная и девятеричная системы (φ(3) = φ (9)= 1,262). Десятичная система оказывается всего примерно на 20% расточительнее двоичной (а не в 1,5 раза, как можно было бы заключить, рассматривая функцию $f(n)$).

При возрастании n функция φ (n) стремится к единице ($\lim\limits_{n\to\infty} \varphi(n) = 1$). Дело в том, что, когда n достаточно велико, разница между $\lambda(n)$ и $\log_2 n$ не играет особой роли. Поэтому определенный интерес могут представлять системы счисления с высокими основаниями. Например, для $n = 100$ φ(n)= 1,053, для n =1000 φ(n) = 1,003. Однако возможности применения этих систем недостаточно исследованы.

Рассматривая далее свойства функции φ(n), интересно отметить, что при $n > 6$ всегда φ (n) < $f(n)$. Это означает, что если для $n > 6$ даже и имеются достаточно надежные многопозиционные схемы, которые по количеству оборудования пропорционально сложнее 2-позиционных устройств, то и тогда выгоднее использовать двоичное кодирование цифр n-ичной системы счисления. Например, если 10-позиционное кольцо содержит 10 триодов, а 2-позиционный триггер — два, то выгоднее кодировать десятичные цифры при помощи 4 двоичных, потому что четыре 2-позиционных триггера содержат всего 8 триодов. Правда, логические схемы при использовании 10-позиционных колец могли бы оказаться проще.

3°. *Затраты оборудования при использовании смешанных систем счисления* можно оценить функциями, аналогичными $f(n)$ и φ (n).

Пусть имеем смешанную систему счисления, содержащую p_1 разрядов по основанию n_1 (т. е. с n_1 допустимыми символами для каждого из них), p_2 разрядов по основанию n_2,...,p_k разрядов по основанию n_k. Всего количество разных чисел, которые можно изобразить при помощи этого набора, равно

$$n_1^{p_1} n_2^{p_2} \ldots n_k^{p_k}.$$

Для записи такого же количества чисел в двоичной системе потребовалось бы примерно $\log_2 (n_1^{p_1} n_2^{p_2} \ldots n_k^{p_k})$ разрядов.

Если бы каждая цифра в системе счисления с основанием n_i изображалась, например, n_i-позиционным кольцом с n_i триодами, то всего для осуществления смешанной системы потребовалось бы $p_1 n_1 + p_2 n_2 + \ldots + p_k n_k$ триодов. В то же время если бы каждая двоичная цифра изображалась одним 2-позиционным триггером с двумя триодами, то в двоичной системе при той же точности потребовалось бы $2 \log_2 (n_1^{p_1} n_2^{p_2} \ldots n_k^{p_k})$ триодов. Функция

$$f\begin{pmatrix} n_i \\ p_i \end{pmatrix} = f\begin{pmatrix} n_1, n_2, \ldots, n_k \\ p_1, p_2, \ldots p_k \end{pmatrix} = \frac{n_1 p_1 + n_2 p_2 + \ldots + n_k p_k}{2 \log_2 \left(n_1^{p_1} n_2^{p_2} \ldots n_k^{p_k} \right)}$$

показывает, во сколько раз количество оборудования, необходимого для осуществления данной смешанной системы счисления, больше количества оборудования, необходимого при использовании двоичной системы, при той же точности и при непосредственном изображении n_i-ичных цифр. Аналогичная функция для случая, когда цифры n_i-ичной системы счисления кодируются минимальным количеством двоичных цифр, имеет вид

$$\varphi\begin{pmatrix} n_i \\ p_i \end{pmatrix} = \varphi\begin{pmatrix} n_1, n_2, \ldots n_k \\ p_1, p_2, \ldots p_k \end{pmatrix} = \frac{\lambda(n_1) p_1 + \lambda(n_2) p_2 + \ldots + \lambda(n_k) p_k}{\log_2 \left(n_1^{p_1} n_2^{p_2} \ldots n_k^{p_k} \right)}.$$

Рассматривая свойства функций $f\begin{pmatrix} n_i \\ p_i \end{pmatrix}$ и $\varphi\begin{pmatrix} n_i \\ p_i \end{pmatrix}$, можно доказать, что смешанные системы счисления всегда занимают по количеству оборудования промежуточное место между соответствующими однородными системами. Это означает, что, расположив $n_1, n_2, \ldots, n_k$ в таком порядке, чтобы выполнялись неравенства

$$f(n_{s_1}) \le f(n_{s_2}) \le \ldots \le f(n_{s_k}),$$

мы обязательно получим при любых p

$$f(n_{s_1}) \le f\begin{pmatrix} n_i \\ p_i \end{pmatrix} \le f(n_{s_k}).$$

Аналогичным образом, расположив $n_x, n_2, \ldots, n_k$ в таком порядке, что $\varphi(n_{r_1}) \le \varphi(n_{r_2}) \le \ldots \le \varphi(n_{r_k})$, при любых p получим $\varphi(n_{r_1}) \le \varphi\begin{pmatrix} n_i \\ p_i \end{pmatrix} \le \varphi(n_{r_k})$.

Например, двоично-пятеричная система (система, в которой для части разрядов допустимы по 2 цифры, а для части разрядов — по 5 цифр) при любом способе изображения пятеричных цифр оказывается менее экономной, чем чисто двоичная система, и более экономной, чем чисто пятеричная система. Предположим, к примеру, что количество двоичных разрядов равно количеству пятеричных разрядов: $p_1 = p_2 = p$. Тогда для случая непосредственного изображения пятеричных цифр количество оборудования в двоично-пятеричной системе оценивается величиной

$$f\binom{2,5}{p,p}=\frac{2p+5p}{2\log_2\left(2^p\cdot 5^p\right)}=\frac{2+5}{2\log_2 10}\approx 1{,}054.$$

В то же время для двоичной системы $f(2)=1$, а для пятеричной системы $f(5)=1{,}078$; таким образом,

$$f(2)\langle f\binom{2,5}{p,p}\langle f(5)\cdot$$

Точно так же для случая кодирования пятеричных цифр при помощи двоичных цифр количество оборудования в двоично-пятеричной системе оценивается величиной

$$\varphi\binom{2,5}{p,p}=\frac{1\cdot p+3\cdot p}{\log_2\left(2^p\cdot 5^p\right)}=\frac{1+3}{\log_2 10}\approx 1{,}204\cdot$$

В то же время для чисто двоичной системы $\varphi(2)=1$, а для чисто пятеричной системы $\varphi(5)=1{,}294$, следовательно,

$$\varphi(2)\langle\varphi\binom{2,5}{p,p}\langle\varphi(5)\cdot$$

Такие же неравенства сохранились бы и при любых других соотношениях в количестве двоичных и пятеричных разрядов. Полного доказательства этих положений мы здесь не приводим.

Из сказанного видно, что те выводы по оценкам влияния основания системы счисления на количество оборудования в цифровых элементах, которые были сделаны в 1° и 2°, справедливы не только для однородных, но и вообще для любых систем счисления. Например, если цифры *n*-ичной системы счисления кодируются при помощи минимального количества двоичных цифр, то в любом случае (в том числе и при возможности использования смешанных систем) наилучшими с точки зрения затрат оборудования являются системы счисления с основаниями 2,4,8,16,..., а также системы с очень высокими основаниями.

1.2.3. Влияние выбора основания системы счисления на скорость выполнения операций.

В предыдущем разделе нам удалось провести оценки влияния выбора основания системы счисления на количество необходимого оборудования в весьма общем виде. Мы обращали внимание на основание системы счисления и не интересовались никакими другими признаками способа записи чисел: позиционной или символической является система изображения чисел, естественный ли порядок принят для весов разрядов и т. д.

К сожалению, в такой общей форме мы не сможем оценить влияние выбора основания системы счисления на скорость выполнения операций. Мы увидим из дальнейшего, как некоторые специальные свойства системы изображения чисел могут дать неожиданно высокие скорости выполнения определенных операций; такой случай приведен, например, в разделе 1.3.4. Все же чаще всего в вычислительных машинах используются позиционные системы изображения чисел с естественным порядком весов разрядов. Для этих систем некоторые оценки можно провести.

1°. *Скорость выполнения умножения* очень значительно сказывается на общем быстродействии машины. Как показывает рассмотрение многих разнообразных задач, умножение приходится выполнять в общем несколько реже, чем сложение или вычитание. Но так как выполнение одного умножения требует большего времени, чем выполнение одного сложения или вычитания, то оказывается, что при решении многих задач машина в основном занята выполнением умножений.

Представим себе, что умножение в машине выполняется самым простым способом — путем последовательных сложений и сдвигов. От обычного способа умножения на бумаге «столбиком» машинный метод умножения отличается только тем, что на листе бумаги мы сначала выписываем все частичные произведения и потом суммируем их, а в машине сумма частичных произведений будет накапливаться в специальном устройстве — регистре частичных произведений по мере их получения; кроме того, умножение множимого на цифры отдельных разрядов множителя будет заменяться последовательными сложениями.

Первоначально регистр частичных произведений гасится (ставится в «0») и множимое добавляется к нему столько раз, сколько единиц содержится в младшей цифре множителя; при этом в регистре частичных произведений образуется произведение множимого на младшую цифру множителя. Затем множимое сдвигается на один разряд влево, что в позиционной *n*-ичной системе счисления с естественными весами разрядов эквивалентно умножению на n — подобно тому, как в десятичной системе сдвиг числа на один разряд влево эквивалентен умножению на 10. После этого множимое добавляется к содержимому регистра частичных произведений столько раз, сколько единиц содержится во второй цифре множителя. Затем снова производится сдвиг множимого на один разряд влево и т. д.— пока не будут использованы все цифры множителя.

Следующий пример иллюстрирует этот метод на примере умножений 2-разрядных десятичных чисел:

```
  94 — множимое
× 23 — множитель
----
0000 — начальное состояние регистра частичных произведений
 +94
----
 0094
 +94
----
 0188   } — умножение на 1-ю цифру множителя (3)
 +94
----
 0282
 +94
----
 1222   } — умножение на 2-ю цифру множителя (2)
 +94
----
 2162   } — результат
```

Подсчитаем теперь, как зависит от основания системы счисления n количество сложений, необходимых в процессе выполнения одного умножения.

Максимальная цифра множителя — это n — 1. Например, в двоичной системе максимальная цифра есть 1, в десятичной системе 9. Поэтому и максимальное количество сложений, которые потребуются при умножении на один разряд множителя в системе счисления с основанием n, равно n —1.

Предположим далее, как это мы делали в 1.2.2, что точность вычислений характеризуется количеством N различных чисел, с которыми оперирует машина, и что соответствующее количество разрядов в n-ичной системе счисления равно $\log_n N$. Тогда полное количество сложений, которые максимально могут потребоваться в процессе выполнения одного умножения, равно

$$(n - 1)\log_n N.$$

В частности, в двоичной системе оно равно $1 \log_2 N$.

Функция

$$f_0(n) = \frac{(n-1)\log_n N}{1 \cdot \log_2 N} = \frac{n-1}{\log_2 n}$$

показывает, во сколько раз максимальное количество сложений, необходимых для выполнения одного умножения в n-ичной системе счисления, больше, чем максимальное количество сложений, необходимых для выполнения одного умножения в двоичной системе, при одинаковой точности и при использовании описанного выше метода выполнения умножения.

В таблице 1-3 приведены значения функции $f_0(n)$ для нескольких начальных значений n. С ростом n функция $f_0(n)$ монотонно возрастает. Например, для десятичной системы количество сложений, необходимых при выполнении одного умножения, более чем в 2,7 раза больше, чем в двоичной системе при равной точности.
Таким образом, из рассмотрения функции $f_0(n)$ видно, что двоичная система дает значительный выигрыш в быстродействии по сравнению со всеми другими системами счисления.

Таблица 1-3

		Функция $f_0(n)$							
n	2	3	4	5	6	7	8	9	10
$f_0(n)$	1,000	1,262	1,500	1,725	1,913	2,138	2,333	2,524	2,709

Нужно помнить, что функция $f_0(n)$ дает не соотношение отрезков времени, необходимых для выполнения умножения, а только соотношение количества сложений. С одной стороны, выполнение умножения включает не только сложения, но и сдвиги множимого. Количество сдвигов равно количеству разрядов в множителе. Оно, следовательно, тем меньше, чем больше основание системы счисления n. Это несколько улучшает временные соотношения для систем счисления с высокими основаниями. С другой стороны, сама длительность выполнения одного сложения при больших n может оказаться больше, чем в двоичной системе, что приведет, наоборот, к ухудшению временных соотношений для высоких значений n. Однако функция $f_0(n)$ дает в общем правильное представление о зависимости времени выполнения умножения от основания системы счисления.

Мы увидим из дальнейшего (4.2, 4.4), что и при использовании других методов выполнения умножения двоичная система при прочих равных условиях обеспечивает более высокое быстродействие, чем какая-либо другая система счисления. Существуют, правда, методы, позволяющие достигнуть в десятичной системе примерно таких же скоростей, как в двоичной. Но для осуществления этих методов требуется существенно более сложное оборудование,

чем обычно используется в двоичной системе.

2°. *Скорости выполнения других операций* в ряде случаев можно оценить таким же способом, как мы оценили скорость выполнения умножения. Для многих операций при этом и соотношения будут получаться примерно такими же, как для умножения. К подобным операциям относятся, например, деление и извлечение квадратного корня. Но обычно эти операции приходится выполнять не очень часто, и оценки такого рода не представляют особого интереса.

Важнее всего, конечно, было бы провести оценки скоростей выполнения сложения и вычитания в зависимости от основания системы счисления. Однако принципы выполнения сложения и вычитания, используемые в вычислительных машинах, очень разнообразны, а скорости выполнения этих операций сильно связаны с применяемой методикой.

В ряде случаев оказывается, что скорость выполнения сложения или вычитания находится в прямой зависимости от общего количества двоичных знаков в каждом из чисел, участвующих в операции. (Предполагается, что цифры n-ичной системы счисления кодируются группами двоичных знаков.) При этом, очевидно, соотношения скоростей, соответствующих различным основаниям системы счисления n, можно примерно оценить с помощью функции $\varphi(n)$ из раздела 1.2.2. Наилучшими окажутся системы счисления, в которых при той же точности требуется минимальное количество двоичных знаков для представления одного числа, т. е. двоичная, четверичная, восьмеричная и т. д. В десятичной системе длительность выполнения сложения или вычитания будет примерно на 20% больше, чем в двоичной системе.

При использовании других принципов выполнения сложения зависимость скорости выполнения сложения или вычитания от основания системы счисления n может оказаться более сильной — например, похожей на функцию $f_0(n)$.

Так или иначе при использовании одинаковых принципов выполнения операций двоичная система, как правило, обеспечивает более высокие скорости, чем какая-либо другая система счисления. Мы говорим, разумеется, об общей тенденции; можно, наверное, без особого труда найти и исключение из этого правила. Бывает и так (см., например, раздел 4.4), что, скажем, в десятичной системе удается применить методы ускорения операций, для которых нет аналогов в двоичной системе; при этом в десятичной системе могут получиться более высокие скорости, чем достигаются в двоичной системе. Но тогда и проигрыш в оборудовании при использовании десятичной системы будет больше, чем это можно было бы заключить из оценок раздела 1.2.2.

Необходимо еще раз подчеркнуть, что все, сказанное относительно влияния выбора системы счисления на скорость выполнения операций, относится к позиционным системам **с** естественным порядком весов.

1.2.4 Перевод чисел из одной системы счисления в другую.

Во всех случаях, когда в вычислительной машине используется не десятичная система счисления, приходится переводить некоторые числа из одной системы в другую. Переводить приходится те исходные данные, которые машина получает от человека, конечные результаты, сообщаемые машиной человеку, а также, возможно, некоторые промежуточные результаты, нужные для контроля.

Перевод чисел осуществляется либо вручную, на листе бумаги, с помощью, может быть, настольной клавишной вычислительной машины, либо самой электронной вычислительной машиной — путем выполнения специальной программы перевода, в которой используются такие операции, как сложение, вычитание, умножение и т. д. Иногда для перевода чисел из одной системы в другую строятся специальные устройства — самостоятельные или входящие в качестве одного из узлов в вычислительную машину. Наконец, в некоторых случаях перевод чисел выполняется в виде самостоятельной операции арифметического устройства машины, на равных правах со сложением, умножением и т. д.

В зависимости от указанных условий может быть различна и методика перевода.

Мы рассмотрим сейчас те способы перевода чисел из одной системы счисления в другую, которые обычно используют при выполнении этих операций вручную; аппаратные способы рассмотрены в разделе 5.4; программные способы перевода чисел в этой книге не рассматриваются. Речь будет идти о числах, представленных позиционной системой счисления с основанием n, с естественными весами разрядов и с цифрами 0, 1, ..., n — 1.

При выполнении операций вручную в общем случае перевод чисел из системы счисления с основанием n_1 в систему счисления с основанием n_2 выполняется через десятичную систему: из n_1-ичной системы число переводится в десятичную систему, затем из десятичной системы — в n_2-ичную. Только в некоторых специальных случаях (имеющих, однако, важное значение) такой перевод осуществляется непосредственно, минуя десятичную систему; эти случаи рассматриваются в конце настоящего раздела. Перевод выполняют обычно отдельно для целой и отдельно для дробной части числа.

1°. *Перевод целых чисел из системы счисления с основанием n в десятичную систему.* Пусть имеем целое число $A = \alpha_m\alpha_{m-1}...\alpha_2\alpha_1$, записанное позиционной системой счисления с основанием n с естественными весами разрядов. Как было объяснено в разделе 1.1., это означает, что данное число может быть записано в виде

$$A=\alpha_m n^{m-1} +\alpha_{m-1}n^{m-2} + ... + \alpha_2 n + \alpha_1$$

Иначе можно записать:

$$A = (...((\alpha_m n + \alpha_{m-1})\, n + \alpha_{m-2})\, n + \ldots + \alpha_2)\, n + \alpha_1.$$

Эта запись дает удобный алгоритм для перевода чисел в десятичную систему. Он состоит из следующих операций:

(1) умножить старшую цифру числа (α_m) на основание системы счисления (n);
(2) добавить к предыдущему результату следующую по порядку цифру числа;
(3) умножить результат предыдущей операции на n;
(4) повторять (2) и (3) до тех пор, пока при выполнении (2) не будет добавлена младшая цифра числа; на этом действии прервать операции.

Все операции выполняются при этом в десятичной системе.

П р и м е р . Перевод восьмеричного числа 76201 в десятичную систему

$$
\begin{aligned}
(76201)_8 &\to 7 \times 8 = 56\\
&+\ 6 \to 62\\
&\times\ 8 \to 496\\
&+\ 2 \to 408\\
&\times\ 8 \to 3984\\
&+\ 0 \to 3984\\
&\times\ 8 \to 31872\\
&+\ 1\\
&(31873)_{10}
\end{aligned}
$$

2°. *Перевод дробных чисел из системы счисления с основанием n в десятичную систему.* Дробное число $B = 0, \beta_1 \beta_2 ... ...\beta_{m-1}\beta_m$, представленное позиционной n-ичной системой счисления с естественными весами разрядов, можно записать в виде

$$B = \beta_1 \cdot n^{-1} + \beta_2 \cdot n^{-2} + \ldots + \beta_{m-1} \cdot n^{-(m-1)} + \beta_m \cdot n^{-m}$$

или иначе

$$B = ((\ldots (\beta_m : n + \beta_{m-1}) : n + \ldots + \beta_2) : n + \beta_1) : n.$$

Отсюда — аналогично предыдущему — алгоритм перевода выглядит следующим образом:

(1) разделить младшую цифру числа (β_m) на основание системы счисления (n) (или умножить на $1/n$);
(2) добавить к предыдущему результату следующую по порядку цифру числа;
(3) разделить предыдущий результат на основание системы счисления (или умножить на обратную величину);
(4) повторять (2) и (3) до тех пор, пока при выполнении (2) не окажется добавленной старшая цифра числа; после этого один раз выполнить (3).

Все операции выполняются в десятичной системе.

П р и м е р . Перевод восьмеричного числа 0,76201 в десятичную систему

$$
\begin{aligned}
(0,76201)_8 &\to 1 : 8 = 0,125\\
&+ 0 \to 0,125\\
&: 8 \to 0,015625\\
&+ 2 \to 2,015625\\
&: 8 \to 0,251953\ldots\\
&+ 6 \to 6,251953\ldots\\
&: 8 \to 0,781494\ldots\\
&+ 7 \to 7,781494\\
&: 8\\
&\overline{\qquad\qquad}\\
&(0,972687\ldots)_{10}
\end{aligned}
$$

3°. *Перевод целых чисел из десятичной системы в систему счисления с основанием n* основывается на следующих соображениях.

Если некоторое целое число A записано в n-ичной позиционной системе с естественными весами разрядов в виде $A = \alpha_m\alpha_{m-1}...\alpha_2\alpha_1$, то эта запись, как мы говорили, обозначает

$$A = \alpha_m n^{m-1} + \alpha_{m-1} n^{m-2} + \ldots + \alpha_2 n + \alpha_1.$$

Ясно, что младшая цифра α_1, $\alpha_1 < n$, есть остаток от деления A на n. Целая часть частного от деления A на n равна

$$A_1 = \frac{A - \alpha_1}{n} = \alpha_m n^{m-2} + \alpha_{m-1} n^{m-3} + \ldots + \alpha_2 .$$

Вторая цифра числа α_2 есть, очевидно, остаток от деления A_1 на n и т. д. Отсюда алгоритм перевода целого числа из десятичной системы в позиционную n-ичную систему счисления выглядит следующим образом:

(1) разделить исходное число A (заданное в десятичной системе) на n; полученный остаток является младшей цифрой в представлении числа n-ичной системой счисления;
(2) разделить целую часть частного, полученного при предыдущем делении, на n; остаток является очередной цифрой в n-ичном представлении числа;
(3) повторять (2) до тех пор, пока целая часть частного не окажется равной нулю.

Все операции, как и в предыдущих случаях, выполняются в десятичной системе.

П р и м е р . Перевод десятичного числа 31873 в восьмеричную систему

$(31873)_{10}$: 8 → 3984 (остаток 1)
:8 → 498 (остаток 0)
:8 → 62(остаток 2) :
:8 → 7 (остаток 6):
:8→ 0 (остаток 7)

Искомое представление числа — $(76201)_8$.

4°. *Перевод дробных чисел из десятичной системы в систему счисления с основанием n* выполняется по аналогии с переводом целых чисел.

Если некоторое дробное число B записано в n-ичной позиционной системе счисления с естественными весами разрядов в виде

$$B = 0,\beta_1\beta_2\ldots\beta_{m-1}\beta_m,$$

то это означает, что

$$B = \beta_1 \cdot n^{-1} + \beta_2 n^{-2} + \ldots + \beta_{m-1} n^{-(m-1)} + \beta_m n^{-m}.$$

Отсюда вытекает следующий алгоритм преобразования:

(1) умножить исходное число B (заданное в десятичной системе) на n; целая часть полученного результата является старшей цифрой в представлении числа n-ичной системой;
(2) умножить дробную часть предыдущего результата на n; целая часть полученного произведения является очередной цифрой в n-ичном представлении числа;
(3) повторять (2) до тех пор, пока дробная часть произведения не окажется равной нулю или пока не будет получено достаточное количество цифр в n-ичном представлении числа.

Все операции можно выполнять в десятичной системе счисления.

П р и м е р 1. Перевод десятичного числа 0,972687 в восьмеричную систему

(0.972687)10 × 8 = 7,781496 (целая часть 7)

0,781496 × 8 = 6,251968 (целая часть 6)

0,251968 × 8 = 2,015744 (целая часть 2)

0,015744 × 8 = 0,125952 (целая часть 0)

0,125952 × 8 = 1,007616 (целая часть 1)

и т. д.

Искомое представление числа — $(0,76201...)_8$.

П р и м е р 2. Перевод числа 1/3 в двоичную систему

$$\frac{1}{3} \times 2 = \frac{2}{3} \quad \text{(целая часть 0)}$$

$$\frac{2}{3} \times 2 = 1\frac{1}{3} \quad \text{(целая часть 1)}$$

$$\frac{1}{3} \times 2 = \frac{2}{3} \quad \text{(целая часть 0)}$$

и т. д. (периодически).

Искомое представление числа — $(0,010101...)_2$.

5°. *Некоторые специальные случаи.* Как указывалось в 1.2.1, наряду с широко распространенной двоичной системой счисления в вычислительных машинах нашли некоторое применение системы счисления с основаниями 4, 8 и 16.

Эти системы применяются в основном во входных и выходных устройствах двоичных машин. Главное их достоинство как раз в том и состоит, что перевод чисел из двоичной системы в систему счисления с основанием 2^k (k — целое положительное число) и обратный перевод очень просты, не требуют почти никаких вычислений и могут выполняться либо в уме, если это делает человек, либо с помощью простейших средств, если перевод производится в машине. С другой стороны, применение на входе и на выходе машины двоичной системы неудобно потому, что двоичные числа — даже при невысокой точности — содержат очень много знаков и очень однообразны.

Если, например, числа, которые нужно ввести в машину, заранее переведены в двоичную систему (или, например, команды для машины составлены сразу по двоичной системе), то их все равно трудно было бы перенести на перфоленту или перфокарты: для этого пришлось бы затратить слишком много времени, причем вряд

ли удалось бы избежать большого количества ошибок. Лучше данные готовить в четверичной, восьмеричной или шестнадцатеричной системе. При этом перевод в двоичную систему выполняется большей частью прямо аппаратами, на которых выполняется перфорация лент или карт.

Аналогичным образом обстоит дело и на выходе машины.

Очевидно, правило перевода чисел из двоичной системы счисления в систему счисления с основанием 2^k выглядит следующим образом:

— разбить исходное двоичное число на группы по k разрядов, начиная от запятой (вправо и влево);

— рассматривая каждую группу из k двоичных разрядов как целое число, заменить его одной цифрой (так как число, представленное группой из k двоичных разрядов, может быть заключено в пределах от 0 до 2^k— 1, то всего различных цифр для каждого разряда при этом может получиться 2^k).

П р и м е р 1. Перевод двоичного числа

11001000101,11101001010011

в восьмеричную систему.

Разбиваем разряды исходного числа на группы по 3 разряда в каждой группе:

011 001 000 101, 111 010 010 100 110

(в неполные группы справа и слева дописываем нули). Заменяем каждую группу одной цифрой. Так как $(011)_2 = 3$; $(001)_2 = 1$; $(000)_2 = 0$; $(101)_2 = 5$ и т. д., восьмеричную запись получаем в виде

3105, 72246.

В шестнадцатеричной системе недостающие 6 цифр (соответствующих числам 10, 11, ..., 15) записывают обычно либо знаками $\overline{0}, \overline{1}, \ldots \overline{5}$, либо соответственно буквами *a*, *b*,... ...,*f*. Мы будем применять второй из этих способов, так как знаки $\overline{0}, \overline{1}, \ldots$ в дальнейшем используются в другом смысле.

Для перевода числа, представленного системой счисления с основанием 2^κ, в двоичную систему необходимо каждую цифру в исходной форме числа заменить k-разрядным двоичным числом.

П р и м е р 2. Перевод в двоичную систему шестнадцатеричного числа 3*e*0*a*,6725*b*.

Так как $3 = (0011)_2$; $e = (1110)_2$; $0 = (0000)_2$; $a = 1010$ и т. д., двоичная запись числа имеет вид (нули перед целой частью опущены)

11 1110 0000 1010, 0110 0111 0010 0101 1011

В разделах 4 и 5 мы увидим, что системы счисления с основаниями 4, 8 и 16 иногда используются как вспомогательное средство и в арифметических устройствах двоичных машин.

Аналогично этому в троичной машине для вспомогательных целей могла бы использоваться система счисления с основанием 9.

Мы упоминали уже о том, что в некоторых вычислительных машинах применялась смешанная двоично-пятеричная система, в которой через один чередуются разряды с количествами допустимых цифр 2 и 5. При использовании многопозиционных колец двоично-пятеричная система дает заметную экономию в оборудовании по сравнению с десятичной системой: для десятичной системы имеем $f(10) = 1{,}505$, для двоично-пятеричной системы $f\begin{pmatrix} 2, 5 \\ p, p \end{pmatrix} = 1{,}054$ (см. 1.2.2); таким образом, в двоично-пятеричной системе количество оборудования всего на 5,4% больше, чем в двоичной системе, и примерно в $1\frac{1}{2}$ раза меньше, чем в десятичной.

В то же время правила преобразования, связывающие двоично-пятеричную систему с десятичной системой счисления, не отличаются в принципе от правил, связывающих двоичную систему, например, с восьмеричной. В этом, собственно, и состоит основное достоинство двоично-пятеричной системы.

Чтобы перейти от десятичной системы к двоично-пятеричной системе, достаточно каждую десятичную цифру заменить одной двоичной цифрой и одной пятеричной цифрой; двоичная цифра (0 или 1) показывает, сколько целых пятерок входит в данную десятичную цифру, пятеричная цифра (0, 1,2,3 или 4) дает дополнительно количество единиц. Например, десятичное число 624,17503 может быть записано по двоично-пятеричной системе в виде

*1*1 *0*2 *0*4, *0*1 *1*2 *1*0 *0*0 *0*3

(курсивом выделены двоичные цифры, прямые цифры — пятеричные). Столь же просто выполняется и обратный перевод.

Возможна также пятерично-двоичная система, в которой каждая десятичная цифра тоже заменена одной пятеричной и одной двоичной цифрой, но пятеричная цифра (0, 1, 2, 3 или 4) дает целое количество двоек, а двоичная— дополнительное количество единиц. То же число 624,17503 в пятерично-двоичной системе выглядит следующим образом (где снова курсивом выделены двоичные цифры):

3*0* 1*0* 2*0*, 0*1* 3*1* 2*1* 0*0* 1*1*.

1.3. Позиционные и символические способы изображения чисел

1.3.1. Определения.

Мы упоминали уже о том, что в практике большей частью используется позиционное представление чисел с естественными весами разрядов, но что в принципе возможно применение и других способов изображения чисел.

Напомним, что *позиционным* называется такой способ изображения чисел, когда *каждой из возможных для некоторого разряда цифр поставлено в соответствие определенное число, а каждому разряду приписан определенный вес; любое число, записанное позиционным способом, равно сумме чисел, соответствующих его цифрам, взятых с весами, соответствующими разрядам, в которых стоят эти цифры.*

Все другие способы изображения чисел мы будем называть *символическими.*

Естественными называются такие веса разрядов, когда *в целой части числа вес первого (младшего) разряда равен единице, а вес любого другого разряда равен произведению количеств цифр, допустимых для каждого из разрядов, находящихся в целой части правее данного разряда; в дробной части числа вес каждого разряда равен величине, обратной произведению из количества цифр, допустимых для данного разряда и для разрядов дробной части числа, находящихся слева от него.*

Если, например, число представлено десятичной системой счисления, то естественными являются следующие веса разрядов. В целой части числа вес первого (младшего) разряда равен 1. Вес второго разряда равен 10, потому что справа от него в целой части находится один разряд, для которого допустимо 10 различных цифр. Вес третьего разряда равен 100, потому что справа от него в целой части числа имеются два разряда, для каждого из которых возможно по 10 цифр ($10 \times 10 = 100$) и т. д. В дробной части числа вес старшего разряда равен $^1/_{10}$, потому что в нем самом может быть 10 различных цифр, а слева от него в дробной части числа других разрядов нет; вес второго разряда в дробной части числа равен $^1/_{100}$, потому что в нем самом может быть 10 различных цифр и слева от него находится один разряд дробной части, для которого тоже допустимо 10 различных цифр $\left(\frac{1}{10\times10}=\frac{1}{100}\right)$, и т.д.

На первый взгляд кажется, что такое определение естественных весов слишком сложно. Однако трудно дать другое определение, которое подходило бы не только для однородных, но и для смешанных систем счисления.

Например, для двоично-пятеричной системы, рассмотренной в разделе 1.2.4. естественные веса разрядов должны быть следующими: в целой части числа:

— вес первого (младшего) разряда— 1;
— вес второго разряда — 5 (потому что справа от него в целой части находится один разряд с пятью допустимыми цифрами для него);
— вес третьего разряда — 10 (потому что справа от него в целой части находится один разряд с двумя допустимыми цифрами для него и один разряд с пятью допустимыми цифрами: $2\times5=10$);
— вес четвертого разряда — 50 (потому что справа от него в целой части находится один разряд с пятью допустимыми цифрами для него, один разряд с двумя допустимыми цифрами и еще один разряд с пятью допустимыми цифрами: $5 \times 2 \times 5 = 50$);

и т. д.

в дробной части числа:

— вес первого (старшего) разряда —$^1/_2$ (потому что для него допустимы 2 различные цифры);
— вес второго разряда — $^1/_{10}$ (потому что для него самого допустимы 5 различных цифр и слева от него имеется один разряд дробной части с двумя допустимыми цифрами: $\frac{1}{5\times2}=\frac{1}{10}$);
— вес третьего разряда — $^1/_{20}$ (потому что для него самого допустимы 2 различные цифры, а слева от него имеется один разряд дробной части с 5 допустимыми цифрами и один разряд дробной части с двумя допустимыми цифрами: $\frac{1}{2\times5\times2}=\frac{1}{20}$);

и т.д.

Таким образом, число, записанное по двоично-пятеричной системе,

$$1102040121210000 3,$$

(где курсивом выделены двоичные цифры, прямые цифры — пятеричные) может быть представлено в виде

$$1\cdot500+1\cdot100+0\cdot50+2\cdot10+0\cdot5+4\cdot1+0\cdot\frac{1}{2}+1\cdot\frac{1}{10}+1\cdot\frac{1}{20}+2\cdot\frac{1}{100}+1\cdot\frac{1}{200}+0\cdot\frac{1}{1000}+0\cdot\frac{1}{2000}+$$
$$+0\cdot\frac{1}{10000}+0\cdot\frac{1}{20000}+3\cdot\frac{1}{100000}=(624{,}17503)_{10}$$

(ср. с 1.2.4).

Из приведенного определения естественного порядка весов вытекают следующие рекуррентные формулы. Если естественный вес некоторого i-го разряда обозначить через δ_i, а количество допустимых символов в нем через n_i, то для разряда, находящегося слева от данного разряда,

$$\delta_{i_{\text{лев}}} = \delta_i \cdot n_i,'$$

для разряда, находящегося справа от данного разряда,

$$\delta_{i_{\text{прав}}} = \frac{\delta_i}{n_{i_{\text{прав}}}}$$

Эти формулы справедливы и для целой, и для дробной части числа. Так как в дробной части числа нумерация разрядов идет слева направо, а в целой части — справа налево, то в дробной части числа $i_{\text{лев}} = i$ — 1, $i_{\text{прав}} = i$+l, в целой части числа наоборот. Впрочем, рекуррентные формулы справедливы и при переходе от первого разряда дробной части к первому разряду целой части и наоборот.

Задавшись для некоторого (i-го) разряда произвольным весом δ'_i (может быть, отличающимся от его естественного веса δ_i), можно затем воспользоваться рекуррентными формулами для определения весов всех остальных разрядов ...δ'_{i-3}, δ'_{i-2}, δ'_{i-1}, δ'_{i+1}, δ'_{i+2}, δ'_{i+3},...; при этом будет получена система изображения чисел, которая во всем сходна с позиционным изображением чисел с естественными весами разрядов, но в которой все числа как бы умножены на постоянный коэффициент. Если разрядам числа приписаны какие-нибудь другие веса, не соответствующие приведенному определению, то такие веса разрядов называются *искусственными.*

1.3.2. Некоторые свойства позиционного способа представления чисел с естественными весами разрядов.

Рассматриваемый способ представления чисел сыграл выдающуюся роль в развитии методов вычислений в период до появления электронных цифровых машин. В значительной мере важность его сохранилась и в настоящее время. Мы рассмотрим сейчас основные свойства этого способа изображения чисел.

Прежде всего должен быть разобран вопрос о том, какие числа могут быть поставлены в соответствие цифрам. Выше мы предполагали, что различным цифрам n-ичной системы счисления соответствуют числа 0, 1, 2,..., (n— 1). Однако в разделе 1.1 указывалось, что этот выбор не является единственно возможным.

Выбор чисел, которые ставятся в соответствие цифрам (в дальнейшем для краткости мы говорим «выбор цифр») должен обеспечить выполнение следующих очевидных требований:

(1) если система счисления однородна, то одни и те же цифры должны употребляться во всех разрядах числа;

(2) при ограниченном количестве разрядов m и определенных весах разрядов*) все числа, которые можно представить с помощью этих разрядов при выбранном наборе цифр, должны располагаться на числовой оси через равные интервалы;

(3) среди чисел, представленных ограниченным количеством разрядов m при помощи выбранного набора цифр, должно содержаться число нуль;

(4) условия (2) и (3) должны соблюдаться при любом ограниченном количестве разрядов m.

Заметим еще, что умножение всех цифр на некоторый масштабный множитель M приводит к умножению на M всех чисел, записанных с помощью указанного набора цифр.

Можно показать, что *для того, чтобы набор цифр удовлетворял четырем перечисленным условиям, необходимо и достаточно* (с точностью до масштабного множителя), *чтобы цифрам был поставлен в соответствие отрезок натурального ряда, включающий число* 0.

Например, в двоичной позиционной системе счисления с естественными весами разрядов в качестве цифр могут быть выбраны либо числа — 1, 0, либо числа 0, +1. Однако эти два набора отличаются один от другого только масштабным множителем —1 и, следовательно, по существу идентичны. Таким образом, в двоичной системе выбор цифр для позиционного способа изображения чисел с естественными весами разрядов может быть сделан единственным способом с точностью до масштабного множителя.

В троичной системе цифрам могут быть приписаны либо значения —2, —1, 0, либо значения —1, 0, +1. либо значения 0, +1, +2. Но первый и третий из возможных наборов различаются лишь масштабным множителем —I, Следовательно, в троичной системе возможны два существенно различных набора для изображения чисел позиционным способом с естественными весами разрядов.

Вообще в n-ичной системе счисления выбор цифр, пригодных при использовании позиционного способа

*) Веса разрядов должны быть естественными. Свобода в выборе здесь имеется только в том отношении, что запятая в указанных разрядах может располагаться различно. Определив положение запятой, мы тем самым определим и веса всех m разрядов.

изображения чисел с естественными весами разрядов, может быть сделан либо $(n + 1)/2$ различными способами (если n нечетно), либо $n/2$ различными способами (если n четно).

Доказательство приведенного правила выбора цифр сводится к следующему. Сначала доказывается, что для соблюдения условий (1) и (2) необходимо и достаточно, чтобы в качестве цифр были выбраны числа, отстоящие одно от другого на равные интервалы. Это требование не налагает пока никаких ограничений на выбор цифр для двоичной системы счисления. Далее доказывается, что условие (3) могло бы выполняться и без того, чтобы одна из цифр соответствовала числу 0 (достаточно среди цифр иметь положительные и отрицательные числа, подобранные специальным образом), но тогда не выполнялось бы условие (4). Полное доказательство слишком громоздко, и мы его здесь приводить не будем.

Если цифры выбраны указанным образом, то правила счета при позиционном способе изображения чисел очень просты. Правило *прямого счета* выглядит следующим образом:

для того чтобы от какого-либо числа перейти к ближайшему большему числу, нужно в первом (младшем) из разрядов, цифра которого не является наибольшей допустимой для данного места, заменить находящуюся в нем цифру ближайшей большей, а во всех разрядах, находящихся правее от него, заменить наибольшие допустимые для них цифры наименьшими допустимыми.

Правило *обратного счета* аналогично:

для того чтобы от какого-либо числа перейти к ближайшему меньшему числу, нужно в первом (младшем) из разрядов, цифра которого не является наименьшей допустимой для данного места, заменить находящуюся в нем цифру ближайшей меньшей, а во всех разрядах, находящихся правее от него, заменить наименьшие допустимые для них цифры наибольшими допустимыми.

Пусть, например, число 62,498 записано позиционным способом с естественными весами разрядов в десятичной системе счисления, причем цифрам соответствуют числа 0, 1, 2,..., 9. Иначе говоря, способ представления чисел — общепринятый в повседневной практике. Первым из разрядов, цифра которого не является наибольшей допустимой для данного места, является самый младший разряд числа (в нем стоит цифра «8»). Поэтому ближайшее большее число, выраженное теми же разрядами, есть число 62,499. Для этого последнего числа ближайшим большим является число 62,500. Младшим из разрядов, цифра которого не являлась наибольшей допустимой для данного места, был третий справа разряд; имевшаяся в нем цифра («4») заменена на ближайшую большую («5»); во всех разрядах, находящихся правее от него, наибольшие допустимые для них цифры («9») заменены наименьшими допустимыми («0»).

Конечно, при общепринятом способе изображения чисел приведенные правила счета представляются слишком очевидными. Читателю предлагается, однако, попробовать применить их, например, в троичной системе счисления цифрами —1, 0, +1 или в двоично-пятеричной системе с цифрами 0, 1 для двоичных разрядов и цифрами —2, —1, 0, +1, +2 для пятеричных разрядов, чтобы убедиться, что правила эти не так тривиальны, как могло бы показаться.

Приведенные правила счета сравнительно просты. Они легко реализуются в схемах счетчиков. Так же достаточно удобны для схемной реализации вытекающие из них правила выполнения сложения и вычитания. В этом состоит одно из важнейших преимуществ позиционного способа представления чисел с естественными весами разрядов перед другими способами изображения чисел. Кстати, и тот простой алгоритм умножения, который был приведен в разделе 1.2.3, существенным образом использовал позиционный характер способа представления чисел и естественный порядок весов разрядов. То же относится и к правилам перевода чисел из одной системы счисления в другую, рассмотренным в 1.2.4.

Другое очень важное достоинство рассматриваемого способа изображения чисел состоит в наличии простого правила округления. Отбросив в позиционном представлении числа (с естественными весами разряда) все разряды, находящиеся правее некоторого разряда, мы получим число, которое отличается от исходного менее чем на единицу младшего из оставшихся разрядов. Например, отбрасывая в десятичном числе 62,498 (представленном позиционным способом с естественными весами разрядов) две младшие цифры, получим число 62,4, . которое отличается от исходного менее чем на единицу младшего из оставшихся разрядов, т. е. менее чем на $^1/_{10}$. Простым общеизвестным приемом ошибку округления можно сократить до величины, не превышающей половины младшего из оставшихся разрядов.

Возможность округления чисел имеет очень важное значение при выполнении операций с ограниченным количеством разрядов. В 1.3.4 рассматривается один пример символического способа представления чисел, имеющего весьма существенные достоинства, но не нашедшего до сих пор широкого применения главным образом из-за отсутствия удобных способов округления.

Наличие простых правил счета (а также выполнения других операций) и простого правила округления чисел является причиной того, что в вычислительных машинах в подавляющем большинстве случаев используются именно позиционные способы представления чисел с естественными весами разрядов — или близкие к ним способы. Никем, однако, не доказано, что среди бесконечного разнообразия возможных способов изображения чисел, отличающихся от позиционных с естественными весами разрядов, не могут быть найдены такие способы, которые тоже обеспечивали бы удобные правила счета и округления и, может быть, имели бы какие-нибудь дополнительные преимущества.

1.3.3. Способы изображения чисел, близкие к позиционному изображению с естественными весами разрядов.

В вычислительной технике сравнительно часто используются способы изображения чисел, близкие к позиционному. Строго говоря, эти способы во многих случаях являются символическими; иногда сохраняется позиционный характер изображения чисел, но веса разрядов отличаются от естественных. Общим для них является наличие простых правил перехода к позиционному изображению чисел с естественными весами разрядов, в связи с чем правила счета и округления чисел похожи на аналогичные правила, рассмотренные в 1.3.2.

Здесь не дано точного определения тому, что называется изображением чисел, *близким к позиционному способу с естественными весами разрядов.* Однако приводимый ниже пример поясняет смысл, который вкладывается в это понятие.

В качестве примера рассмотрим двоичную систему с цифрами «—1» и «+1» (вместо обычных «0» и «1»). В дальнейшем цифру «—1» мы будем записывать знаком «1», цифру «+1», как обычно, знаком «1». В 1.3.2. говорилось о том, что набор цифр $\overline{1},1$ нельзя использовать в двоичной системе при позиционном способе изображения чисел и естественных весах разрядов, потому что, имея ограниченное количество разрядов *m,* цифрами $\overline{1}$, 1 нельзя записать число нуль.

Но если бы количество разрядов было бесконечно велико, такое возражение отпало бы. При этом разрядную сетку достаточно только не ограничивать справа.

Представим себе, например, что разрядная сетка начинается, скажем, от третьего разряда слева от запятой (т. е. от разряда с весом 4), а вправо простирается неограниченно. Тогда с помощью цифр $\overline{1}$, 1 число нуль можно записать в виде

$$\overline{1}\,\overline{1}\,\overline{1},\overline{1}\,\overline{1}\,\overline{1}\ldots$$

(Целая часть этого числа равна единице:

$$1\,\overline{1}\,\overline{1}. = 1\times 4 + (-1)\times 2 + (-1)\times 1;$$

дробная часть равна минус единице:

$$.\overline{1}\,\overline{1}\,\overline{1}\ldots = (-1)\cdot\frac{1}{2} + (-1)\cdot\frac{1}{4} + (-1)\cdot\frac{1}{8} + \ldots = -1.)$$

Иначе число нуль можно записать так:

$$\overline{1}11,111\ldots$$

При бесконечном количестве разрядов неоднозначность изображения имеется и для других чисел.

Оборвем теперь снова разрядную сетку на каком-либо разряде — скажем, на разряде с весом 2^{-k} (*k*-й разряд дробной части числа). Ясно, что погрешность, которая вносится при этом в любое из чисел (величина отброшенной части числа), может быть заключена в пределах от

$$\underbrace{.00\ldots00}_{k\text{ разрядов}}\,\overline{1}\,\overline{1}\,\overline{1}\ldots = -2^{-k}$$

до

$$\underbrace{.00\ldots00}_{k\text{ разрядов}}\,111\ldots = +2^{-k}.$$

Точного изображения нуля мы, конечно, не получим, но с указанной степенью точности нуль можно будет записать.

Удобнее, однако, подойти к этому вопросу несколько иначе. Можно условиться, что для всех чисел, записанных оставшимися разрядами разрядной сетки, «отброшенная» часть имеет вид

$$\underbrace{.00\ldots00}_{k\text{ разрядов}}\,\overline{1}\,\overline{1}\,\overline{1}\ldots = -2^{-k}.$$

Тогда любое число, записанное оставшимися разрядами, фактически увеличено на 2^{-k}. Далее, для того чтобы получить истинное значение какого-либо числа, нужно будет вычесть 2^{-k} из числа, представленного оставшимися разрядами разрядной сетки. Величина -2^{-k} будет играть роль постоянной аддитивной поправки по всем числам. При этом и для числа нуль мы будем иметь точное изображение.

Обрывая разрядную сетку на разряде с весом 2^{-k}, нужно иметь в виду, что ближайшие по величине числа будут отстоять одно от другого не на величину $2^{-\kappa}$ (как было бы, если бы мы пользовались цифрами 0,1), а на величину $2\cdot 2^{-k}$. Это видно из того, что, изменяя цифру разряда с весом 2^{-k} с 0 на 1, мы меняем число на величину 2^{-k} изменяя же цифру того же разряда с $\overline{1}$ на 1, мы меняем число на величину — (—1) $\cdot 2^{-k}$ + 1$\cdot$ 2^{-k}, т. е. вдвое

большую. Поэтому, используя цифры $\bar{1}$, 1 и желая получить заданный интервал между ближайшими числами, мы должны обрывать разрядную сетку на один разряд правее, чем в том случае, когда используются цифры 0,1. Но это, как мы сейчас увидим, не ведет к увеличению общего количества разрядов при той же точности вычислений.

Пусть, например, мы хотим оперировать с целыми числами, отстоящими друг от друга на единицу. Если бы мы пользовались цифрами 0, 1, разрядную сетку нужно было бы оборвать на первом с л е в а от запятой разряде. При использовании цифр $\bar{1}$, 1, как сказано выше, разрядную сетку нужно оборвать на первом разряде с п р а в а от запятой.

В таблице 1-4 приведены числа, записанные с помощью оставшихся разрядов (старшим, как мы условились выше, является третий слева от запятой разряд). Аддитивная поправка для всех чисел в этом случае, очевидно, равна -2^{-1}. Ясно, что, имея больше разрядов, можно было бы аналогичным образом записать и большие числа. При этом, хотя в изображении чисел имеется один «лишний» разряд справа (по сравнению с изображением чисел цифрами 0,1), зато нет необходимости в специальном разряде алгебраического знака. Таким образом, лишних затрат оборудования нет.

Таблица 1-4

Целые числа, записанные двоичной системой с цифрами $\bar{1}$, 1

Изображение числа в двоичной системе с цифрами $\bar{1}$,1	Значение числа, записанного в разрядной сетке (по десятичной системе)	Истинное значение числа (по десятичной системе) с учетом поправки $-1/2$	Изображение числа в двоичной системе с цифрами $\bar{1}$,1	Значение числа, записанного в разрядной сетке (по десятичной системе)	Истинное значение числа (по десятичной системе) с учетом поправки $-1/2$
$\bar{1}\bar{1}\bar{1},\bar{1}$	$-7\frac{1}{2}$	-3	$1\bar{1}\bar{1},\bar{1}$	$+\frac{1}{2}$	0
$\bar{1}\bar{1}\bar{1},1$	$-6\frac{1}{2}$	-7	$1\bar{1}\bar{1},1$	$+1\frac{1}{2}$	$+1$
$\bar{1}1\bar{1},\bar{1}$	$-5\frac{1}{2}$	-6	$1\bar{1}1,\bar{1}$	$+2\frac{1}{2}$	$+2$
$\bar{1}1\bar{1},1$	$-4\frac{1}{2}$	-5	$1\bar{1}1,1$	$+3\frac{1}{2}$	$+3$
$\bar{1}11,\bar{1}$	$-3\frac{1}{2}$	-4	$11\bar{1},\bar{1}$	$+4\frac{1}{2}$	$+4$
$\bar{1}11,1$	$-2\frac{1}{2}$	-3	$11\bar{1},1$	$+5\frac{1}{2}$	$+5$
$\bar{1}11,\bar{1}$	$-1\frac{1}{2}$	-2	$111,\bar{1}$	$+6\frac{1}{2}$	$+6$
$\bar{1}11,1$	$-\frac{1}{2}$	-1	$111,1$	$+7\frac{1}{2}$	$+7$

Легко убедиться, что числа, приведенные в таблице 1-4, записаны в действительности в символической системе. Предположив противное, мы могли бы обозначить веса разрядов через x_1, x_2, x_3, x_4 соответственно. Тогда согласно таблице 1-4 числа x_1, x_2, x_3, x_4 должны были бы удовлетворять следующим 16 уравнениям:

$$\begin{cases} -x_1 - x_2 - x_3 - x_4 = -8, \\ -x_1 - x_2 - x_3 + x_4 = -7, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ +x_1 + x_2 + x_3 + x_4 = +7. \end{cases}$$

Сравнивая, например, первое уравнение с последним, нетрудно видеть, что эти уравнения несовместны.

С другой стороны, и правила округления, и правила счета здесь такие же, как при позиционном способе изображения чисел. Эти правила одинаково справедливы и для положительных, и для отрицательных чисел, В разделе 1.4 мы увидим, что при обычной двоичной записи чисел (с использованием цифр 0,1) получить одинаковые правила счета для положительных и отрицательных чисел не очень просто. Используя запись чисел цифрами $\bar{1}$,1, можно, например, начать счет от какого-нибудь отрицательного числа и, многократно применяя правило прямого счета, выбирать последовательные числа в порядке возрастания, причем переход через нуль не потребует никаких изменений в выполнении операций.

Точно так же и правила выполнения других операций оказываются одинаковыми при любых сочетаниях алгебраических знаков чисел. Например, сложение двух положительных чисел и сложение положительного числа с отрицательным при использовании рассматриваемого способа изображения чисел выполняются по одним и тем же правилам.

В этом, собственно, и состоит основное преимущество рассматриваемого способа изображения чисел. Указанный способ, вероятно, никогда не применялся самостоятельно; но во многих случаях в арифметических устройствах вычислительных машин числа, записанные первоначально обычным способом, в ходе выполнения тех или иных операций преобразуются в указанную форму, или результаты получаются в указанной форме и затем

преобразуются в обычную запись. Впрочем, не видно каких-нибудь причин, по которым рассматриваемый способ изображения чисел не мог бы использоваться в качестве основного, а не вспомогательного.

Переход от позиционного двоичного изображения чисел с естественными весами разрядов и цифрами 0, 1 к записи чисел цифрами $\overline{1}$,1 выполняется по очень простым правилам. Для положительных чисел правило перехода следующее:

(1) приписать к позиционной записи числа цифру 1 слева;

(2) сдвинуть код числа на один разряд вправо;

(3) заменить все цифры «0» цифрами « $\overline{1}$ ».

Например, для целого положительного числа +5 = $(101)_2$ переход к записи цифрами $\overline{1}$, 1 совершается по этому правилу так:

(1) 101.→ 1101.
(2) 1101.→ 110,1
(3) 110,1 → 11 $\overline{1}$,1

Правильность полученного результата можно проверить по таблице 1-4. Однако читатель мог бы проверить приведенное правило преобразования и в общем виде.

Для отрицательных чисел между операциями (1) и (2) необходимо осуществить преобразование числа в дополнительный код. Смысл этой операции поясняется в разделе 1.4; формально же она состоит в следующем:

(1а) заменить все цифры «0» цифрами «1» и все цифры «1» цифрами «0»;

(1б) добавить к полученному числу единицу младшего разряда (т. е. перейти от него к ближайшему большему).

Например, для целого отрицательного числа —5 = $(—101)_2$ переход к записи цифрами 1, 1 совершается по этому правилу так:

(1) 101. 1101.
(1а) 1101.→ 0010.
(1б) 0010 + 0001 = 0011.
(2) 0011.→ 001,1
(3) 001,1 → $\overline{1}$ $\overline{1}$ 1,1

Переход от записи чисел цифрами $\overline{1}$,1 к обычной двоичной записи чисел (позиционным способом с естественными весами разрядов и цифрами 0, 1) может быть выполнен в обратном порядке.

Аналогичным образом может быть построена, например, десятичная система с цифрами —9, —7, —5, —3, — 1,1, 3, 5, 7, 9 вместо обычных цифр 0, 1, 2,...,9. Преимущества и свойства ее аналогичны преимуществам и свойствам рассмотренной двоичной системы с цифрами $\overline{1}$,1.

С другими примерами способов изображения чисел, близких к позиционному изображению с естественными весами разрядов, мы встретимся в разделе 1.4, а также в разделе 4.

1.3.4. Один из символических способов изображения чисел — запись чисел в остатках.

Несмотря на то, что позиционные способы изображения чисел с естественными весами разрядов и близкие к ним системы имеют ряд очевидных достоинств, в настоящее время продолжаются поиски новых способов изображения чисел, которые позволили бы достигнуть тех или иных преимуществ в скорости выполнения операций или в количестве оборудования или в других отношениях. Символические способы изображения чисел чрезвычайно разнообразны; какой-либо системы в их исследовании нет, и трудно предсказать, к каким результатам могут привести эти поиски. Ниже рассматривается одна интересная находка среди символических способов изображения чисел.

1°. Предположим, что вычислительная машина должна оперировать с N целыми числами (от 0 до N — 1). Выберем несколько взаимно простых чисел $s_1, s_2, .., s_m$ с таким расчетом, чтобы их произведение было не меньше N:

$$s_1 s_2 \ldots s_m \geq N.$$

Далее каждое число можно будет однозначно записать остатками от деления на $s_1, s_2, \ldots, s_m$. При этом в первом разряде числа, дающем остаток от деления на s_1 может быть s_1 различных цифр (от 0 до s_1 — 1), во втором разряде числа (остаток от деления на s_2) может быть s_2 различных цифр (0, 1,..., s_2 — 1) и т. д. Таким образом, мы будем пользоваться смешанной системой счисления с различными основаниями для каждого из разрядов.

Пусть для примера N = 30. (В действительности, конечно, количество различных чисел, с которыми должна оперировать вычислительная машина, обычно во много раз больше.) Тогда в качестве взаимно простых чисел $s_1, s_2,...$ могут быть выбраны наименьшие простые числа 2, 3 и 5. Условие $s_1 s_2 s_3 \geq N$ при этом выполняется (2 × 3 × 5 = =30). Запись всех 30 чисел в остатках отделения на 2, 3 и 5 приведена в таблице 1-5. В этой таблице правая цифра в записи числа дает остаток от деления на 2, средняя — остаток от деления на 3, а левая цифра — остаток от деления на 5; впрочем, можно было бы избрать и любой другой порядок записи разрядов.

Таблица 1-5

Пример записи чисел в остатках

Число (по десятичной системе)	Запись в остатках	Число (по десятичной системе)	Запись в остатках
0	000	15	001
1	111	16	110
2	220	17	221
3	301	18	300
4	410	19	411
5	021	20	020
6	100	21	101
7	211	22	210
8	320	23	321
9	401	24	400
10	010	25	011
11	121	26	120
12	200	27	201
13	311	28	310
14	420	29	421

Правила счета при записи чисел в остатках оказываются более простыми, чем при позиционном способе изображения чисел. Они основаны на том, что при увеличении или уменьшении числа на единицу остаток от деления этого числа на s_i тоже соответственно возрастает или уменьшается на единицу. Особым при прямом счете является случай, когда остаток от деления исходного числа на s_i равен s_i — 1: при возрастании числа на единицу деление на s_i будет выполняться нацело (остаток станет равным нулю). Точно так же при обратном счете, если остаток от деления исходного числа на s_i равен нулю, то после уменьшения числа на единицу этот остаток станет равным s_i — 1. Отсюда правило *прямого счета:*

для того чтобы перейти от данного числа к ближайшему большему, нужно во всех разрядах, где цифра не является наибольшей допустимой для данного места, заменить имеющуюся цифру ближайшей большей; в разрядах, где цифра является наибольшей допустимой для данного места, заменить ее на наименьшую допустимую[*]).

Правило обратного счета аналогично.

По сравнению с правилами счета для позиционных способов изображения чисел существенное упрощение состоит в том, что счет оказывается поразрядной операцией: цифра, которая получается в каждом из разрядов, никак не зависит от цифр в остальных разрядах числа.

Однако наиболее важные достоинства рассматриваемого способа изображения чисел состоят в том, что не только счет, но и *сложение, вычитание* и *умножение* чисел оказываются *поразрядными* операциями.

Действительно, пусть имеем числа x_1 и x_2 такие, что остаток от деления первого из них на s_i равен a_1 остаток от деления второго на s_i равен a_2. Их можно записать в виде

$$x_1 = k_1 s_i + a_1,$$

$$x_2 = k_2 s_i + a_2,$$

где k_1, k_2 —целые числа. Тогда

$$x_1 + x_2 = (k_1 + k_2)s_i + (a_1 + a_2),$$

$$x_1 - x_2 = (k_1 - k_2)s_i + (a_1 - a_2),$$

$$x_1 x_2 = (k_1 k_2 s_i + a_1 k_2 + a_2 k_1)s_i + a_1 a_2.$$

Здесь $(k_1 + k_2)$, $(k_1 - k_2)$ и $(k_1 k_2 s_i + a_1 k_2 + a_2 k_1)$ — целые числа. Отсюда ясно, что остаток от деления на s_i суммы (разности, произведения) двух чисел равен сумме (разности, произведению) по модулю s_i остатков от деления на s_i исходных чисел. Слова «по модулю s_i» означают, что при сложении (вычитании, умножении) остатков от деления на s_i исходных чисел результат можно увеличить или уменьшить на величину rs_i, где r — любое целое число.

Пусть, например, числа 11 и 2 записаны в остатках от деления на 2, 3 и 5:

$$(11)_{10} = 121,$$

$$(2)_{10} = 220$$

(см. таблицу 1-5).

Сложим эти числа. Остаток от деления на 2 искомой суммы равен сумме (по модулю 2) остатков от деления

[*]) В приведенных выше рассуждениях наибольшая допустимая для данного разряда цифра есть s_i — 1, наименьшая допустимая цифра — всегда 0. Можно, однако, считать, что, например, остаток от деления на 3 может быть равен не 0, 1 или 2, а —1,0 или + 1; такими же будут и цифры в разряде с основанием 3.

на 2 исходных чисел:

$$1 + 0 = 1 ;$$

остаток от деления 3 суммы равен сумме (по модулю 3) остатков исходных чисел:

$$2 + 2 = 4 \equiv 1 \pmod 3;$$

точно так же для остатка от деления на 5:

$$1 + 2 = 3.$$

Таким образом,

$$+\begin{array}{r} 121 \\ 220 \\ \hline 311 \end{array}.$$

Результат, как и следовало ожидать, представляет число $(13)_{10}$ (см. таблицу 1-5).

Аналогичным образом вычитание выполняется следующим образом:

$$1 - 0 = 1 ,$$
$$2 - 2 = 0,$$
$$1 - 2 = -1 \equiv 4 \pmod 5;$$

Следовательно,

$$(11)_{10} - (2)_{10} = \begin{array}{r} 121 \\ -\ 220 \\ \hline 401 \end{array} = (9)_{10}.$$

При умножении

$$1 \times 0 = 0,$$
$$2 \times 2 = 4 \equiv 1 \pmod 3,$$
$$1 \times 2 = 2;$$

поэтому

$$(11)_{10} \times (2)_{10} = \begin{array}{r} 121 \\ \times\ 220 \\ \hline 210 \end{array} = (22)_{10}$$

Результат, который получается при сложении, вычитании или перемножении чисел в любом из разрядов, никак не зависит от цифр в других разрядах числа. Операции во всех разрядах можно выполнять одновременно, причем с каждым разрядом можно действовать независимо от всех остальных разрядов.

Очевидно, что при этом могут быть достигнуты существенно более высокие скорости выполнения операций, чем при использовании позиционного способа изображения чисел. Наиболее удивительным является, вероятно, тот факт, что даже умножение при изображении чисел в остатках оказывается поразрядной операцией; в позиционных системах умножение является весьма сложной операцией (см., например, 1.2.3, а также раздел 4).

В то же время попытки использовать на практике рассмотренный способ изображения чисел наталкиваются на ряд серьезных трудностей.

Практически все преимущества представления чисел в остатках сводятся на нет тремя обстоятельствами:

— отсутствием удобного способа сравнения чисел;

— отсутствием удобного способа для того, чтобы обнаружить выход результата некоторой операции за пределы диапазона допустимых чисел;

— отсутствием удобного метода округления чисел.

Перечисленные трудности представляют собой, по сути дела, три части одной и той же проблемы. Отыскание решения одной из них сняло бы и другие затруднения. Например, если бы был найден удобный способ сравнения чисел, то нетрудно было бы и обнаруживать случаи выхода результатов операций из разрешенного диапазона чисел — путем сравнения каждого результата с максимальным и минимальным числами диапазона. Однако при известных пока способах выполнение сравнения чисел в остатках примерно так же сложно, как выполнение умножения в позиционных системах. Повторять сравнения после каждой операции невозможно, а без этого невозможно в длинной цепочке операций удержать результаты в пределах разрешенного диапазона чисел, который должен быть конечным, если конечно количество разрядов (см. об этом также раздел 4.1.3, п. 3°).

Вследствие этого представление чисел в остатках использовалось до сих пор либо в малых вычислительных машинах без программного управления (где оператор видит каждый промежуточный результат и в уме производит пересчет масштабов и округление), либо в специализированных машинах с короткой постоянной программой, где длинные цепочки вычислений отсутствуют и диапазоны чисел заранее определены для исходных данных и всех промежуточных и конечных результатов. В чехословацкой машине «Эпос», построенной на основе позиционной

десятичной системы счисления, представление чисел в остатках использовано для изображения десятичных цифр.

2°. Приведем для справок один из алгоритмов сравнения чисел в остатках, предложенный некогда М. Л. Цейтлиным.

Пусть числа представлены остатками от деления на $s_1, s_2,...,s_m$, где $s_1, s_2,...,s_m$ — взаимно простые числа $s_1 > s_2 > ... > s_m$, максимально возможное число равно $N - 1 = s_1 s_2 ... s_m - 1$. Ограничение в применении алгоритма состоит в том, что должно быть $s_m = 2$.

Задачу сравнения двух чисел можно свести к задаче определения того, меньше ли некоторое число, чем $N/2 = s_1 s_2 ... s_{m-1}$: если одно из двух сравниваемых чисел меньше, чем $N/2$, а другое больше, то ответ сравнения ясен; если оба числа меньше $N/2$ или оба больше $N/2$, то для выполнения сравнения нужно определить, какова их. разность по модулю N (если разность меньше $N/2$, то уменьшаемое больше или равно вычитаемому).

Итак, пусть имеем число $a_1, a_2,...,a_m$, где a_i — остаток от деления на s_i, $a_i \leq s_i - 1$; требуется определить, меньше ли это число, чем $N/2$.

Отбросим (заменим звездочкой) последнюю цифру числа $a_1, a_2,...,a_{m-1}, *$. Заметим, что по одному числу вида $a_1, a_2, ...,a_{m-1},*$ имеется как среди чисел, меньших $N/2$, так и среди чисел, больших или равных $N/2$; одно из этих двух чисел четно, другое нечетно (ср. левую и правые части таблицы 1-5).

Сделаем предположение, что исходное число меньше $N/2$. Будем вычитать из числа $a_1, a_2,...,a_{n-1},*$ единицу (т. е. число 1,1,..., 1,1) до тех пор, пока во втором справа разряде не получится 0, т. е. пока не получится число $b_1, b_2, ...,b_{m-2}, 0,*$. При каждом таком вычитании четность числа будет изменяться на противоположную. Далее, полученное число разделим на s_{m-1} (оно делится нацело, поскольку в $(m - 1)$-м разряде стоит 0). При этом цифру m-го разряда определить нельзя, поскольку уже в исходном числе в m-м разряде стоит звездочка; цифру $(m - 1)$-го разряда нам также определить не удастся, и мы ее тоже заменим звездочкой. В остальных же разрядах правило деления обратно правилу умножения: к цифре a_i нужно добавить целое количество (r) раз число s_i так, чтобы $rs_i + b_i$ делилось нацело на s_{m-1}, после чего выполнить деление. В результате получится число $a_1^{(1)}, a_2^{(1)},..., a^{(1)}_{m-2}, *, *$, четность которого совпадает с четностью $b_1,b_2,...,b_{m-2}, 0,*$ и о котором известно, что оно меньше $N/2s_{m-1}$.

В следующем шаге из числа $a_1^{(1)}, a_2^{(1)},..., a^{(1)}_{m-2}, *, *$ будем вычитать единицу до тех пор, пока в $(m - 2)$-м разряде не получится 0 (т. е. $a^{(1)}_{m-2}$ раз). В результате этих вычитаний получим число $b_1^{(1)}, b_2^{(1)},..., b^{(1)}_{m-3}, 0, *, *$; при этом каждое вычитание единицы меняет четность числа на противоположную. Аналогично предыдущему разделим его затем на s_{m-2}, в результате чего получится число $a_1^{(2)}, a_2^{(2)},...,a^{(2)}_{m-3}, *, *, *$, четность которого совпадает с четностью числа $b_1^{(1)}, b_2^{(1)},..., b^{(1)}_{m-3}, 0, *, *$ и о котором известно, что оно меньше $N/2s_{m-1}s_{m-2}$.

Продолжим этот процесс до тех пор, пока не будет получено число $a_1^{(m-2)}, *, *,..., *$, о котором известно, что оно меньше $N/2s_{m-1}s_{m-2}...s_2$, т. е. меньше s_1 (поскольку

$$N = s_m s_{m-1} s_{m-2} \ldots s_1 = 2s_{m-1}s_{m-2} \ldots s_2 s_1.$$

Вычитая из него $a^{(m-2)}$ раз единицу, получим нуль. Нуль-число четное. В зависимости от того, четным или нечетным было общее количество вычитаний в описанном выше процессе, соответственно четным или нечетным должно быть то из чисел $a_1,a_2,\ldots a_{m-1}, *$, которое меньше $N/2$; если цифра в m-м разряде исходного числа (a_m) совпадает с определенной таким образом четностью (a_m — остаток от деления на 2 — равен 0 для четных чисел и 1 —для нечетных), то исходное число на самом деле меньше $N/2$; в противном случае оно больше или равно $N/2$.

П р и м е р . Пусть $s_1 = 7$, $s_2 = 5$, $s_3 = 3$, $s_4 = 2$; $N = s_1s_2s_3s_4 = (210)_{10}$; $N/2 = (105)_{10}$. Задано число 4011. Меньше ли оно, чем $N/2$?

Отбросим крайний справа разряд

$$4011 \rightarrow 401*$$

Предположим, что $401* < (105)_{10}$.

Далее у каждого из промежуточных результатов знаком «+» отмечаем совпадение по четности с исходным числом, знаком «—» — несовпадение.

$$\begin{array}{rl}
-\,\begin{array}{r}401*\\1111\end{array} & \\
\hline
340* & (-)\\
:3 = \begin{array}{r}-\,13**\\1111\end{array} & (-)\\
\hline
02** & (+)\\
-\,1111 & \\
\hline
61** & (-)\\
-\,1111 & \\
\hline
50** & (+)\\
:5 = \begin{array}{r}-\,1***\\1111\end{array} & (+)\\
\hline
0 & (-)
\end{array}$$

Если $401* < (105)_{10}$, то последний результат равен нулю; нуль — четное число, его четность противоположна четности того из пары чисел 401*, которое меньше, чем $N/2$. Таким образом, 4011 (исходное число) меньше, чем

$N/2 = (105)_{10}$, число 4010 больше, чем $N/2$.

В действительности $4011 = (25)_{10}$; $4010 = (130)_{10}$.

Максимальное количество операций в процессе выполнения сравнения одного числа с $N/2$ равно $(s_1 — 1) + (s_2 — — 1) +... + (s_{m-1}— 1)$ вычитаний и m — 2 делений, т. е. всего $s_1 + s_2 + . . . + s_{m-1} — 1$. Все это операции однотактные. Тем не менее полный алгоритм сравнения весьма громоздок. Возможно, что затруднение это имеет принципиальный характер и что когда-нибудь будет доказана невозможность получить существенное ускорение сравнения чисел в остатках без существенного усложнения аппаратуры. Представление чисел в остатках рассмотрено здесь сравнительно подробно главным образом для иллюстрации того, что вообще может быть достигнуто удачным способом изображения чисел.

1.4. Представление отрицательных чисел

Вопрос о представлении отрицательных чисел рассматривается в настоящем разделе применительно к позиционным способам изображения чисел с естественными весами разрядов или к близким к ним способам.

1.4.1. Прямые коды.

В разделе 1.3.2 говорилось о том, что в случае, когда используется позиционный способ изображения чисел с естественными весами разрядов, в качестве цифр n-ичной системы счисления должны быть выбраны n последовательных целых чисел, одно из которых должно быть нулем.

Пусть наименьшая из цифр есть a_1 наибольшая из цифр a_n (где $a_n = a_1 + n — 1$). Если вес старшего разряда есть δ, то все m-разрядные числа, которые можно представить с помощью этих цифр, заключены в интервале от

$$a_1\delta + a_1\delta n^{-1} + ... + a_1\delta n^{-m+1} = a_1\delta\frac{1 - n^{-m}}{1 - n^{-1}}$$

до

$$a_n\delta + a_n\delta n^{-1} + ... + a_n\delta n^{-m+1} = a_n\delta\frac{1-n^{-m}}{1-n^{-1}}.$$

Чтобы иметь возможность оперировать и с положительными, и с отрицательными числами, удобнее всего выбрать цифры так, чтобы среди них были и положительные, и отрицательные. Если при этом будет соблюдено условие

$$a_1 = — a_n,$$

то мы получим симметричный относительно нуля диапазон чисел, как это большей частью и требуется.

К сожалению, такой путь возможен только при нечетном основании системы счисления я (например, в троичной системе в качестве цифр могут употребляться числа —1,0, +1, в пятеричной системе — числа — 2, —1, 0, +1, +2). Так как $a_n = a_1 + n — 1$, то при четном n не может соблюдаться условие $a_1 = —a_n$ (все цифры, как мы условились, соответствуют целым числам).

В системах счисления с четными основаниями (двоичной, десятичной и т. д.), строго говоря, нельзя полностью выдержать позиционный способ изображения чисел с естественными весами разрядов и при этом получить симметричный относительно нуля диапазон чисел.

Способ, которым мы пользуемся повседневно (в десятичной системе), состоит в том, что в качестве цифр n-ичной системы выбираются n неотрицательных целых чисел 0, 1, 2,..., n — 1 и к каждому числу приписывается один специальный двоичный разряд — разряд алгебраического знака, с допустимыми символами для него «+» и «—». Числа, выраженные с помощью m разрядов n-ичной системы, заключены при этом в интервале от 0 до

$$a_n\delta\frac{1-n^{-m}}{1-n^{-1}} = \delta n\left(1-n^{-m}\right) \approx \delta n,$$

где δ — вес старшего разряда. Этими разрядами записывается абсолютная величина числа. Разряду алгебраического знака никакого определенного веса приписать нельзя; для того чтобы получить истинное значение числа, нужно абсолютную величину числа умножить либо на +1, если в разряде знака стоит «+», либо на —1, если «—». В этом, собственно, и состоит небольшое отступление от позиционного способа изображения чисел.

Такой способ записи относительных чисел называется прямым кодом.

Недостатки этого способа связаны в основном с тем, что правила счета оказываются разными для положительных и отрицательных чисел. Те правила счета, которые сформулированы в 1.3.2 для позиционной записи чисел с естественными весами разрядов, справедливы лишь по отношению к абсолютной величине числа. Если же рассматривать числа целиком, вместе с алгебраическим знаком, то для перехода от некоторого числа к ближайшему большему нужно применить либо правило обратного счета (если данное число отрицательно), либо правило прямого счета (если число равно нулю или положительно); для того чтобы перейти от некоторого числа к ближайшему меньшему, нужно либо применить правило прямого счета (если число отрицательно или равно

нулю), либо применить правило обратного счета (если число положительно). При этом во всех случаях, когда исходное число не равно нулю, алгебраический знак сохраняется неизменным; если исходное число равно нулю, то ближайшее меньшее отрицательно.

Вытекающие из правил счета правила выполнения сложения и вычитания тоже оказываются существенно различными при разных комбинациях алгебраических знаков чисел. Например, правило сложения двух положительных чисел сильно отличается от правила сложения положительного числа с отрицательным; эта последняя операция тоже может выполняться различно — в зависимости от того, какое из чисел больше по абсолютной величине.

Выполняя вычисления вручную, мы не замечаем указанных затруднений, так как все операции с алгебраическими знаками чисел производим в уме. Однако даже простое перечисление этих хорошо известных и по существу тривиальных правил показывает, что использование общепринятого способа изображения отрицательных чисел может привести к существенному усложнению схемы вычислительного устройства.

Поэтому в вычислительных машинах для систем счисления с четными основаниями наряду с прямыми кодами часто используются другие способы записи отрицательных чисел.

1.4.2. Дополнительные коды.

Пусть, как и в предыдущем подразделе, в качестве цифр n-ичной системы счисления выбраны n неотрицательных целых чисел 0,1,2,..., n— 1. При использовании позиционного способа изображения чисел с естественными весами разрядов все m-разрядные числа, которые можно записать с помощью этих цифр, заключены в интервале от 0 до примерно δn (где δ — вес старшего разряда); общее количество различных чисел в этом интервале равно n^m.

Если мы хотим оперировать не только с положительными, но и с отрицательными числами, то это значит, что количество различных чисел должно удвоиться. Для этого так или иначе к $\boldsymbol{m}$ основным разрядам нужно добавить еще один двоичный разряд. Все дело только в том, как использовать этот разряд. При записи относительных чисел прямым кодом дополнительный двоичный разряд является просто разрядом алгебраического знака; ему не приписывается никакого определенного веса, поэтому и расположение его относительно других разрядов может быть выбрано чисто условно: знак можно писать и слева от числа, и справа от него — никакой роли это не играет.

При использовании дополнительных кодов добавочный двоичный разряд записывается всегда слева от старшего из n-ичных разрядов. Разрядная сетка машины расположена обычно так, что добавочный разряд оказывается в целой части числа. Если бы этому разряду приписывался естественный вес, то его вес был бы равен $\delta_0 = n\delta$, где δ — вес старшего n-ичного разряда (см. 1.3.1) [*]).

Во многих случаях разрядная сетка располагается так, что старший из n-ичных разрядов является первым разрядом дробной части числа, его вес при этом равен $1/n$; добавочный двоичный разряд оказывается тогда первым разрядом целой части числа, его естественный вес должен был бы равняться единице:

$$\delta_0 = n \cdot \frac{1}{n} = 1.$$

Дополнительный код далее может быть построен одним из двух способов, которые приводят почти к одинаковым результатам.

1°. *Первый вариант* состоит в том, что добавочному двоичному разряду приписывается вместо его естественного веса δ_0 вес — δ_0. Ясно, что при наличии цифры «0» в добавочном разряде все число в целом положительно или равно нулю; его основные разряды изображают абсолютную величину числа точно так же, как и при использовании прямых кодов. При наличии цифры «1» в добавочном разряде (с весом — $б_0$) все число в целом отрицательно, потому что величина, записанная всеми остальными разрядами числа, какие бы цифры ни были в этих разрядах, непременно меньше, чем δ_0. В основных разрядах при этом тоже записана, как и при использовании прямых кодов, абсолютная величина числа, но записана она в виде дополнения до δ_0; иначе говоря, для числа $x < 0$ в основных разрядах вместо величины $|x|$ записана величина $\delta_0 - |x|$. Отсюда и название: «дополнительный» код. Добавочный двоичный разряд при использовании дополнительных кодов может рассматриваться как разряд знака: если в нем стоит цифра «0», число положительно, если «1» — отрицательно.

П р и м е р 1. Пусть, например, используется десятичная система счисления (n = 10) и в качестве основных

---

[*]) Если бы разрядная сетка была расположена так, что добавочный разряд оказался бы в дробной части числа, то его естественный вес все равно можно было бы определить по рекуррентной формуле из 1.3.1— как $\delta_o = \delta n$. Но при этом сама величина б (вес старшего из основных разрядов) может оказаться, строго говоря, установленной неверно. Если, например, в разрядную сетку входят в качестве основных 2-й, 3-й и т. д. десятичные разряды дробной части числа, то мы приписываем этим разрядам веса 1/100, 1/1000 и т. д. и тем самым предполагаем, что отсутствующий в разрядной сетке 1-й разряд дробной части числа есть тоже десятичный разряд. Поставив теперь слева от основных разрядов (т. е. на первое место в дробной части числа) не десятичный, а двоичный разряд, следовало бы приписать ему естественный вес 1/2, следующему за ним разряду (старшему из основных) — вес 1/20, следующему затем — вес 1/200 и т. д. Однако, сохранив за основными разрядами веса 1/100, 1/1000 и т. д. и приписав добавочному двоичному разряду вес $\delta n = 10 \cdot 1/100 = 1/10$, мы получили бы все равно систему, обладающую всеми свойствами позиционного способа изображения чисел с естественными весами разрядов (см. предпоследний абзац 1.3.1 —стр. 41—42).

разрядов числа выбраны 1-й, 2-й и 3-й разряды дробной части. Величина, записанная этими разрядами, находится в интервале от 0 (число. 000) до почти 1 (число. 999). Чтобы иметь возможность оперировать не только с положительными, но и с отрицательными числами, нужно к трем десятичным разрядам добавить еще один двоичный разряд. Если мы пользуемся п р я м ы м к о д о м, то этот дополнительный разряд является просто разрядом алгебраического знака (+ или —). К примеру, число +0,537 записывается в виде

+.537,

число —0,537 — в виде

—.537.

Если мы пользуемся д о п о л н и т е л ь н ы м кодом, то добавочный двоичный разряд (с допустимыми для него цифрами 0, 1) размещается обязательно слева от старшего из основных разрядов, т. е. на месте первого разряда целой части числа. Его естественный вес δ_0 должен быть равен единице; но мы, как говорилось, приписываем этому разряду вес —1. При этом, к примеру, число +0,537 записывается в виде

0.537

(где курсивом выделен двоичный разряд), число —0,537 должно быть записано в виде

1.463.

(Действительно,

$$1\cdot(-1)+4\cdot\frac{1}{10}+6\cdot\frac{1}{100}+3\cdot\frac{1}{1000}=-0{,}537.)$$

Основные (десятичные) разряды отрицательного числа при этом представляют его абсолютную величину в виде дополнения до $\delta_0 = 1$(.463 = 1.000—.537).

Число нуль при использовании прямого кода может быть записано либо в виде

+.000,

либо в виде

—.000.

При использовании дополнительного кода число нуль записывается однозначно, в виде

0.000.

Комбинация

1.000

соответствует при этом максимальному по абсолютной величине отрицательному числу:

$$1.000 = 1\cdot(-1)+0\cdot\frac{1}{10}+0\cdot\frac{1}{100}+0\cdot\frac{1}{1000}=-1.$$

(В прямом коде максимальное по абсолютной величине отрицательное число при этом же количестве разрядов равно —.999; одна лишняя комбинация израсходована на неоднозначность в изображении нуля).

П р и м е р 2. Используется двоичная система счисления; в качестве основных разрядов выбраны 1-й, 2-й, 3-й и 4-й разряды дробной части. В прямом коде число, скажем, $+{}^{11}/_{16}$ записывается в виде

+.1011,

число $-{}^{11}/_{16}$ в виде

—.1011.

В дополнительном коде число $+{}^{11}/_{16}$ имеет вид

0.1011,

а число $-{}^{11}/_{16}$

1.0101.

Здесь вес добавочного (первого слева от запятой) разряда равен —1, поэтому записанное указанным образом число равно на самом деле

$$1\cdot(-1)+0\cdot\frac{1}{2}+1\cdot\frac{1}{4}+0\cdot\frac{1}{8}+1\cdot\frac{1}{16}=-\frac{11}{16}.$$

При этом в основных разрядах находится величина . 0101 = $= {}^{5}/_{16}$, являющая дополнением от абсолютной величины отрицательного числа до единицы ($\delta_0 = 1$):

$${}^{5}/_{16} = 1 - {}^{11}/_{16}$$

В рассмотренном варианте построения дополнительного кода мы допустили фактически небольшое отступление от естественного порядка весов, установив для добавочного (знакового) разряда вес — δ_0 вместо его естественного веса δ_0. Во втором варианте мы отойдем вообще от позиционного способа изображения чисел, хотя получим систему, б л и з к у ю (см. 1.3.3) к позиционному изображению чисел с естественными весами разрядов.

2°. *Второй вариант* построения дополнительного кода состоит в следующем. Добавочному двоичному разряду, который размещается слева от старшего из основных разрядов, приписывается его естественный вес δ_0. Но при этом предполагается, что число, записанное добавочным и основными разрядами, отличается от истинного числа на величину δ_0. Величина —δ_0 играет роль постоянной аддитивной поправки: чтобы получить истинное значение

любого числа, нужно из числа, записанного в разрядной сетке (включая добавочный разряд), вычесть величину δ_0.

Основными разрядами, как говорилось, может быть представлена величина, лежащая в интервале от 0 до примерно δ_0. Добавив слева один двоичный разряд с весом δ_0, мы получаем возможность записывать величины от 0 до примерно $2\delta_0$. При этом истинные значения чисел будут находиться в интервале от —δ_0 до примерно +δ_0.

Ясно, что если в изображении числа имеется единица в добавочном разряде (с весом δ_0), то, вычитая поправку δ_0, мы получим неотрицательное число; его абсолютная величина записана в основных разрядах точно так же, как при использовании прямых кодов. Если в изображении числа в добавочном разряде имеется нуль, то величина, записанная в разрядной сетке, меньше δ_0; вычитая поправку δ_0, мы получим отрицательное число, абсолютная величина которого представлена основными разрядами разрядной сетки в виде дополнения до δ_0.

Возвращаясь к примеру 1 на стр. 29, мы видим, что при построении дополнительного кода по второму варианту число + 0,537 должно быть записано в виде

1.537.

Здесь вес добавочного двоичного разряда δ_0 равен единице. Поэтому постоянная аддитивная поправка ко всем числам есть — 1. Истинное значение числа, представленного в виде *1*,537, равно 1,537—1 = +0,537. Число —0,537 должно записываться в виде *0*.463.

В такой записи истинное значение числа равно 0,463 — —1 = — 0,537.

Число нуль изображается в виде

1.000

(так как 1,000 — 1 = 0,000).

Максимальное по абсолютной величине отрицательное число записывается в виде

0.000.

Его истинное значение равно 0,000 — 1 = —1.

Таким образом, дополнительный код, построенный по второму варианту, отличается от дополнительного кода, построенного по первому варианту, формально только тем, что в добавочном (знаковом) разряде для положительных чисел записывается 1 вместо 0, а для отрицательных чисел — 0 вместо 1.

Точно так же в примере 2 (стр. 30) при построении дополнительного кода по второму варианту число $+^{11}/_{16}$ должно записываться в виде

1.1011,

число — $^{11}/_{16}$ — в виде

0.0101.

Здесь так же различие между дополнительными кодами, построенными по первому и по второму вариантам, оказывается чисто формальным.

3°. Интересно заметить, что дополнительный код, построенный по первому варианту, можно при определенных условиях свести к дополнительному коду, построенному как бы по второму варианту. Для этого нужно к изображению числа в первом варианте дополнительного кода приписать слева еще один добавочный двоичный разряд и установить в этом разряде цифру, противоположную той, которая имеется в первом добавочном разряде. Далее можно считать, что все разряды имеют естественные веса (в частности, добавочный разряд, имевшийся прежде,— вес + δ_0, новый добавочный разряд — вес + $2\delta_0$), но что величина, записанная в разрядной сетке, отличается от истинного значения числа на величину $2\delta_0$. Такой код называют иногда *модифицированным дополнительным кодом.*

В примере 1 в дополнительном коде, построенном по первому варианту, число + 0,537 записывалось в виде

0.537;

число —0,537 — в виде

1.463.

Действуя указанным способом, записываем изображения этих чисел в виде соответственно

10.537

и

01.463.

Курсивом, как и прежде, выделены двоичные разряды. Полагая, что они имеют естественные веса ($2\delta_0 = 2$; $\delta_0 = 1$), найдем величины, записанные в разрядной сетке:

$$1 \cdot 2 + 0 \cdot 1 + 5 \cdot \frac{1}{10} + 3 \cdot \frac{1}{100} + 7 \cdot \frac{1}{1000} = 2{,}537,$$

$$0 \cdot 2 + 1 \cdot 1 + 4 \cdot \frac{1}{10} + 6 \cdot \frac{1}{100} + 3 \cdot \frac{1}{1000} = 1{,}463\,.$$

Добавляя к этим величинам постоянную аддитивную поправку — $2\delta_0$ = —2, найдем, что истинные значения чисел равны соответственно + 0,537 и — 0,537.

4°. Нетрудно убедиться, что при использовании дополнительных кодов, построенных по второму варианту, нельзя подобрать таких весов разрядов, чтобы истинное значение любого числа получалось суммированием его

цифр с этими весами. Таким образом, дополнительные коды, построенные по второму варианту, не являются, строго говоря, позиционным способом записи чисел. Однако и в первом, и во втором вариантах дополнительные коды близки по своим свойствам к позиционному изображению чисел с естественными весами разрядов.

В частности, и правила округления, и правила счета здесь такие же, как правила, сформулированные в 1.3.2 для позиционного изображения чисел с естественными весами разрядов. Одни и те же правила счета могут одинаково применяться как к положительным, так и к отрицательным числам (чем дополнительные коды выгодно отличаются от прямых кодов).

Справедливость последнего утверждения для дополнительных кодов, построенных по второму варианту, очевидна непосредственно: изображения всех чисел (и положительных, и отрицательных) положительны и представлены позиционным способом с естественными весами разрядов; наличие для всех чисел постоянной аддитивной поправки никак не сказывается при применении правил счета.

При использовании дополнительных кодов, построенных по первому варианту, правила счета следовало бы вообще пересмотреть в той части, в которой они касаются старшего разряда числа, так как вес этого разряда искусственно принят равным — δ_0 (вместо естественного веса + δ_0). К счастью, однако, в двоичной системе счисления (когда речь идет об одном двоичном разряде) нет разницы между сложением и вычитанием:

$$0 + 0 = 0 - 0 = 0,$$
$$0 + 1 \equiv 0 - 1 \equiv 1 \pmod 2,$$
$$1 + 0 = 1 - 0 = 1,$$
$$1 + 1 \equiv 1 - 1 \equiv 0 \pmod 2.$$

Поэтому и в дополнительных кодах, построенных по первому варианту, непосредственно применимы те правила счета, которые сформулированы в 1.3.2 для позиционного изображения чисел с естественными весами разрядов.

Пусть, например, имеем такую же разрядную сетку, как в примере 1 на стр. 29 (см. также стр. 31). При использовании дополнительного кода, построенного по первому варианту, изображение наименьшего по абсолютной величине отрицательного числа имеет вид

1.999.

Это — число —0,001. Применяя непосредственно правило прямого счета из 1.3.2, найдем изображение ближайшего большего числа:

0.000;

как и следовало ожидать, мы получили число нуль.

Точно так же при использовании дополнительного кода, построенного по второму варианту: изображение наименьшего по абсолютной величине отрицательного числа (—0,001) имеет вид

0.999;

применение правила прямого счета (см. 1.3.2.) дает в результате

1.000,

что является изображением числа нуль.

5°. Для двоичной системы счисления нетрудно усмотреть связь между дополнительным кодом, построенным по второму варианту, и системой с цифрами $\overline{1}$,1, рассмотренной в 1.3.3. Если в любом двоичном числе, представленном дополнительным кодом (который построен по второму варианту), заменить все цифры 0 цифрами 1, то получится изображение у д в о е н н о г о исходного числа в системе с цифрами $\overline{1}$,1. Эти соотношения подробно рассмотрены в 1.3.3.

Аналогичные связи имеются и в десятичной системе счисления (и вообще в любой системе счисления с четным основанием *n).* Если в изображении любого десятичного чцсла, записанного дополнительным кодом (который построен по второму варианту) заменить во всех десятичных разрядах цифры 0, 1,2, 3, 4, 5, 6, 7, 8, 9 соответственно цифрами —9, —7, —5, —3, —1, 1, 3, 5, 7, 9, а в двоичном разряде — цифры 0, 1 цифрами —1, 1, то получится изображение удвоенного исходного числа системой с цифрами —9, —7, ..., 9 для десятичных разрядов и с цифрами —1, 1 для двоичного разряда.

Пусть, например, в той же разрядной сетке, которая использована в примере 1 (стр. 29 и 31) число +0,537 записано в виде

1.537.

Произведя замену цифр по указанному правилу, получим

1.135,

т. е. $1\cdot 1 + 1\cdot {}^{1}/_{10} - 3\cdot {}^{1}/_{100} + 5\cdot {}^{1}/_{1000} = 1{,}075$. Поскольку мы предполагаем, что отброшенные справа разряды числа содержат величину 0,00099999... = —0,001, постоянная аддитивная поправка в этом изображении чисел равна — 0,001. Таким образом, запись 1,135 соответствует числу 1,075—0,001 = 1,074, т. е. удвоенному исходному числу (0,537 × 2).

Точно так же, например, для отрицательного числа —0,537 запись дополнительным кодом (по второму варианту) имеет вид

*0.*463.

Произведя указанную замену цифр, получим

$\overline{1},\overline{1}3\overline{3},$

т.е. —1·1—$1\cdot{}^{1}/_{10}$ + $3\cdot{}^{1}/_{100}$ — $3\cdot{}^{1}/_{1000}$ = —1,073. Учтя аддитивную поправку —0,001, найдем, что запись $\overline{1},\overline{1}3\overline{3},$ соответствует числу —1,074, т. е. удвоенному исходному числу (—0,537 × 2).

6°. При построении дополнительных кодов по второму варианту для систем счисления с четным основанием $n > 2$ возможна некоторая модификация этого варианта. Модификация состоит в том, что в качестве старшего разряда кода (с весом δ_0) используется не двоичный разряд, а обычный n-ичный разряд, не отличающийся ничем от основных разрядов числа; при этом аддитивная поправка считается равной не величине — δ_0, а величине — $\delta_0\, n/2$.

Например, в рассмотренном выше примере (для десятичной системы) можно заменить добавочный двоичный разряд добавочным десятичным разрядом — первым разрядом целой части числа, и при этом считать, что аддитивная поправка ко всем числам равна —5. Число +0,537 должно при этом записываться в виде

5.537

(5,537 — 5 = 0,537); число — 0,537 будет представлено в виде

4.463

(4,463 — 5 = —0,537).

Число нуль будет записано в виде

5.000.

Наибольшее по абсолютной величине отрицательное число в данной разрядной сетке имеет вид

0.000,

что соответствует числу—5,000 (так как 0,000 — 5 = —5); наибольшее по абсолютной величине положительное число изображается в виде

9.999,

что соответствует числу +4,999.

Все свойства дополнительных кодов при этом сохраняются. Аналогично предыдущему такая система связана с изображением десятичных чисел цифрами $\overline{9},\overline{7},\overline{5},...,7,9$. Преимущество этой модификации состоит в полной идентичности всех разрядов в изображении числа.

7°. При использовании дополнительных кодов правила счета и, как мы увидим в разделе 3, вытекающие из них правила сложения и вычитания оказываются проще, чем при использовании прямых кодов. В то же время такие операции, как умножение и деление, проще выполняются в прямых кодах.

Поэтому при построении вычислительных машин часто используется следующий прием: нормально числа записываются прямым кодом; перед выполнением сложения или вычитания они преобразуются в дополнительный код; результат операции затем снова преобразуется в прямой код.

При преобразовании чисел из прямого кода в дополнительный или обратно главное затруднение связано с определением цифр основных разрядов для отрицательных чисел. Как указывалось, в изображении отрицательного числа x прямым кодом основные разряды числа содержат величину $|\, x\, |$, в изображении того же числа дополнительным кодом — величину δ_0 —$|\, x\, |$ (где δ_0 — вес знакового разряда: $\delta_0 = \delta n$; δ — вес старшего из основных разрядов, n — основание системы счисления). При переходе от прямого кода к дополнительному, как и при обратном переходе, для отрицательных чисел приходится выполнить вычитание величины, содержащейся в основных разрядах числа, из δ_0.

Способ, которым это делается, основан на том, что число $\delta_0 = \delta n$ можно представить в виде

$$\delta_0 = \delta n = \delta(n-1) + \frac{\delta}{n}(n-1) + \frac{\delta}{n^2}(n-1) + ... + \frac{\delta}{n^{m-1}}(n-1) + \frac{\delta}{n^{m-1}}\cdot 1.$$

Здесь $\delta, \frac{\delta}{n}, \frac{\delta}{n^2}, ..., \frac{\delta}{n^{m-1}}$ — веса m основных разрядов числа, величины $(n$ — 1) — старшие допустимые цифры в них, величина $\frac{\delta}{n^{m-1}}\cdot 1$ — единица младшего разряда.

Число $\delta_0 - \frac{\delta}{n^{m-1}} = \delta_0(1 - n^{-m})$, равное $\delta(n-1) + \frac{\delta}{n}(n-1) + ... + \frac{\delta}{n^{m-1}}(n-1)$, содержит наибольшие допустимые цифры (n— 1) во всех основных разрядах. Найти дополнение до него очень просто. Для этого нужно в каждом из

основных разрядов найти дополнение от имеющейся в нем цифры до (n — 1). Затем к полученному результату достаточно добавить единицу младшего разряда (величину $\frac{\delta}{n^{m-1}}$), чтобы получить дополнение до δ_0.

Пусть, например (см. пример 1 на стр. 29), отрицательное десятичное число—0,537 записано прямым кодом в виде

—.537.

Для нахождения цифр основных разрядов этого числа в дополнительном коде найдем сначала дополнение от каждой цифры до 9:

.462

(эта величина — дополнение от величины .537 не до 1,000, а до 0,999); затем добавим единицу младшего разряда:

$$\begin{array}{r} \cdot 462 \\ +\ \cdot 001 \\ \hline \cdot 463 \end{array}$$

При обратном переходе на первом этапе из величины . 463 получим

.536,

на втором этапе

$$\begin{array}{r} \cdot 536 \\ +\ \cdot 001 \\ \hline \cdot 537 \end{array}$$

Аналогичным образом в примере 2 (стр. 30) имеем отрицательное двоичное число $-{}^{11}/_{16}$, записанное прямым кодом в виде

—.1011.

При переходе к дополнительному коду цифры основных разрядов находятся в два этапа:

— сначала в каждом разряде находится дополнение от имеющейся цифры до 1; иначе говоря, каждая цифра заменяется на обратную; при этом получим

.0100,

что является дополнением от исходной величины не до 1,0000, а до 0,1111;

— затем к полученному числу добавляется единица младшего

$$\begin{array}{r} \cdot 0100 \\ +\ \cdot 0001 \\ \hline \cdot 0101. \end{array}$$

Заметим, что если первый этап преобразования является поразрядной операцией, то второй этап выполняется по обычному правилу прямого счета, так что каждая цифра результата зависит, вообще говоря, от всех младших цифр.

1.4.3. Обратные коды.

Обратные коды нигде, по-видимому, не используются в качестве самостоятельного способа изображения относительных чисел. Однако в тех случаях, когда в качестве основного способа используется прямой код, при выполнении сложения и вычитания часто вместо перехода к дополнительным кодам производится переход к обратным кодам (см. 3.1).

Обратные коды отличаются от дополнительных тем, что при записи отрицательных чисел основные разряды содержат вместо дополнения от абсолютной величины числа до δ_0 дополнение от абсолютной величины числа до $\delta_0 (1 - n^{-m})$. Все обозначения здесь такие же, как в 1.4.2:

δ_0 — вес добавочного двоичного разряда,
n — основание системы счисления,
m — количество основных (n-ичных) разрядов.

Относительно содержимого добавочного (знакового) разряда возможны, как и для дополнительных кодов, различные условия.

Можно, например, условиться — по аналогии с первым вариантом построения дополнительных кодов—, что для положительных чисел добавочный разряд содержит «0», для отрицательных — цифру «1». При этом следует считать, что все основные разряды имеют естественные веса, а вес добавочного (знакового) разряда равен $-\delta_0 (1 - n^{-m})$ (а не — δ_0, как в дополнительных кодах); числа записаны позиционным способом, но с искусственным весом для старшего разряда.

Если — по аналогии со вторым вариантом построения дополнительных кодов — договориться, что для положительных чисел знаковый разряд содержит цифру «1», а для отрицательных «0», то это эквивалентно условию, что все основные разряды имеют естественные веса, знаковый разряд имеет вес $+\delta_0(1 - n^{-m})$, а все числа увеличены на $+\delta_0(1 - n^{-m})$ (т. е. величина — $\delta_0 (1 - n^{-m})$ является аддитивной поправкой ко всем числам). Второй

вариант, строго говоря, не является позиционным способом изображения чисел.

Вернемся к примерам 1 и 2 на стр. 29—30. Положительные числа в дополнительном и в обратном кодах (а по существу и в прямом коде) записываются одинаково. Скажем, если дополнительные коды строятся по первому варианту и обратные коды построены аналогично, то в разрядной сетке примера 1 положительное число +0,537 записывается в виде

в прямом коде +.537,
в дополнительном и обратном коде *0.*537.

Курсивом выделен двоичный (знаковый) разряд. Для отрицательных чисел разница более существенна. В той же разрядной сетке отрицательное число —0,537 имеет вид

в прямом коде...—.537,
в дополнительном коде *1.*463,
в обратном коде ...../.462.

В обратном коде вес добавочного двоичного разряда равен здесь — $\delta_0 (1 — n^{-m}) = — 1.(1—0{,}001) = —0{,}999$; поэтому представленное указанным способом число действительно равно

$$1 \cdot (-0{,}999) + 4 \cdot \frac{1}{10} + 6 \cdot \frac{1}{100} + 2 \cdot \frac{1}{1000} = -0{,}537.$$

Нуль в обратном коде может записываться двояко. В разрядной сетке примера 1 он либо имеет вид

*0.*000

(здесь $0 \cdot (-0{,}999) + 0 \cdot {}^{1}/_{10} + 0 \cdot {}^{1}/_{100} + 0 \cdot {}^{1}/_{1000} = 0$),
либо может быть записан в виде

*1.*999

(здесь $1 \cdot (-0{,}999) + 9 \cdot {}^{1}/_{10} + 9 \cdot {}^{1}/_{100} + 9 \cdot {}^{1}/_{1000} = 0$).
Наибольшее по абсолютной величине отрицательное число в этой разрядной сетке при использовании обратного кода есть число —0,999; оно записывается в виде

*1.*000;наименьшее по абсолютной величине отрицательное число, равное —0,001, имеет вид

*1.*998.

Аналогичным образом в разрядной сетке примера 2 положительное число $+ {}^{11}/_{16}$ записывается в виде

в прямом коде .. +.1011,
в дополнительном и обратном коде 0.1011;

отрицательное число $—{}^{11}/_{16}$ имеет вид

в прямом коде..... —.1011,
в дополнительном коде.....1.0101,
в обратном коде..... 1.0100;

вес добавочного разряда здесь равен $—1 \cdot (1 — {}^{1}/_{16}) = = —{}^{15}/_{16}$ (по двоичной системе —0,1111); нуль в обратном коде может быть записан либо в виде 0.0000, либо в виде 1.1111.

Как и при использовании дополнительных кодов, в обратных кодах можно применять и к положительным, и к отрицательным числам те правила счета, которые в 1.3.2 были сформулированы для позиционного способа изображения чисел с естественными весами разрядов. Однако при переходе через нуль правила счета усложняются.

Например, в разрядной сетке примера 1 наименьшее по абсолютной величине отрицательное число, как указывалось, записывается в виде

*1.*998.

Применяя один раз правило прямого счета, найдем изображение ближайшего большего числа:

*1.*999.

Как и следовало ожидать, получили изображение нуля (в одной из двух возможных форм). Однако, применяя еще раз правило прямого счета, вместо ближайшего большего числа получаем другое изображение нуля:

*0.*000.

Только следующее применение правила прямого счета дает изображение ближайшего большего числа:

*0.*001.

Аналогичная картина наблюдается и при обратном счете. Таким образом, выполняя счет в обратных кодах, нужно

при переходе через нуль сделать один лишний (холостой) просчет. Соответствующим образом несколько усложняются правила сложения и вычитания (по сравнению с правилами сложения и вычитания в дополнительных кодах). Но, как будет показано в разделе 3.1, эти усложнения не очень значительны и при определенных построениях арифметических устройств легко реализуются схемно. Преимуществом же обратных кодов перед дополнительными является более простая связь с прямыми кодами. Из 1.4.2 (см. стр. 33) видно, что преобразование числа из прямого кода в обратный из обратного кода в прямой является поразрядной операцией.

1.5. Указание положения запятой

Как и в предыдущем разделе, изложение здесь ведется применительно к позиционным способам изображения чисел или к близким к ним способам. Ясно, что при использовании символической записи чисел вопрос о положении запятой, отделяющей целую часть числа от дробной, в общем случае не имеет смысла.

Для указания положения запятой при построении вычислительных машин применяется обычно один из двух способов, первый из которых называется *фиксированной* запятой, второй — *плавающей* запятой. Среди вычислительных машин последних моделей сравнительно часто встречаются машины, которые оперируют как с числами с фиксированной запятой, так и с числами с плавающей запятой.

1.5.1 Фиксированная запятая.

При использовании этого метода указания положения запятой место запятой в разрядной сетке машины заранее обусловлено — раз и навсегда для всех чисел. При этом запятую большей частью помещают перед старшим из основных разрядов числа, так что этот разряд считается первым разрядом дробной части числа. Машина в указанном случае может оперировать только с дробными числами. Если используются дополнительные коды, то добавочный (знаковый) разряд является первым разрядом в целой части числа (как это и было принято нами в примерах, рассмотренных в 1.4).

Запятая физически никак не обозначается. Но в алгоритмах выполнения таких операций, как умножение и деление, заранее учтено место запятой в разрядной сетке машины.

Расположение фиксированной запятой перед старшим из основных разрядов имеет то преимущество, что результат умножения при этом никогда не выходит за пределы разрядной сетки машины. Разумеется, если сомножители слишком малы, то может оказаться, что в результате умножения мы получим в разрядной сетке машины только нули; но это просто означает, что с той точностью, с которой машина производит операции, произведение равно нулю. Если же мы хотим выполнить умножение с максимальной возможной точностью, то мы должны записать сомножители в таком масштабе, чтобы все или почти все разряды разрядной сетки были значащими; иначе говоря, сомножители должны быть взяты с такими коэффициентами, чтобы они были близки к единице. Тогда и их произведение окажется близким к единице; при размещении произведения в разрядной сетке машины все или почти все разряды окажутся значащими, и точность произведения будет почти такой же, как точность сомножителей.

Пусть, например, разрядная сетка машины содержит 4 десятичных разряда, и пусть нам нужно найти произведение 38,418 x 0,042363. (Точное значение этого произведения равно 1,627501734.) Предположим, что запятая в разрядной сетке фиксирована перед старшим из разрядов, т. е. что машина оперирует только с дробными числами. Исходные числа могут быть введены в машину только с некоторыми коэффициентами (число 38,418 иначе просто нельзя разместить в разрядной сетке). Чтобы точность была возможно большей, выберем эти коэффициенты так, чтобы все цифры разрядной сетки были значащими; для этого первый сомножитель введем с коэффициентом 10^{-2}, второй сомножитель— с коэффициентом 10^{1}; округление придется провести только в последнем разряде. Таким образом, в разрядной сетке машины сомножители будут размещены в виде

$$.3842 \times .4236.$$

Произведение этих чисел равно 0,16274712; в разрядной сетке машины смогут разместиться 4 старших разряда произведения в виде

$$.1627,$$

что с точностью до единицы младшего разряда разрядной сетки представляет искомый результат (в масштабе 10^{-1}). Совсем другая картина получилась бы, если бы запятая была зафиксирована где-нибудь в другом месте. Предположим, например, что запятая фиксируется справа от младшего из разрядов разрядной сетки; при этом младший разряд был бы первым разрядом целой части числа, машина оперировала бы только с целыми числами. Чтобы исходные числа рассматриваемого примера разместились в разрядной сетке с наибольшим числом значащих цифр, нужно было бы первое из них ввести с коэффициентом 10^{2}, второе — с коэффициентом 10^{5}. Но тогда произведение

$$3842. \times 4236.$$

получится слишком большим и выйдет за пределы разрядной сетки. Чтобы этого не произошло, нужно для первого сомножителя выбрать масштаб 1, для второго 10^{3}; исходные числа (с учетом округления) разместятся в разрядной сетке в виде 0038. и 0042.; их произведение получится равным 1596. С коэффициентом $10"^{3}$ это число должно представлять искомый результат; но точность его получилась очень низкой.

С другой стороны, если машина оперирует только с дробными числами, то результат сложения, вычитания и деления может легко выйти за пределы разрядной сетки. Достаточно, например, попытаться разделить большее число на меньшее, чтобы получился результат, который нельзя разместить в разрядной сетке машины (число, большее единицы). Впрочем, и при любом другом фиксированном расположении запятой имеется аналогичная опасность.

Подготавливая задачу для решения на машине с фиксированной запятой, программист должен очень тщательно продумать выбор масштабов для всех чисел. Масштабы должны быть такими, чтобы результаты операций не выходили за пределы разрядной сетки и в то же время имели достаточную точность. При этом речь идет не только об исходных числах и конечных результатах, но и обо всех промежуточных числах, которые встретятся по ходу дела. Для выполнения необходимых требований приходится внимательно следить за порядком операций в программе, предусматривать в программе специальные операции по изменению масштабов чисел, применять некоторые специальные приемы программирования (о которых частично говорится ниже — см. 1.5.2). Так или иначе эта подготовительная работа требует много времени и часто является источником многих ошибок при программировании.

В этом и состоит основной недостаток фиксированного расположения запятой.

Машина обычно строится так, что при переполнении разрядной сетки (т. е. при получении в какой-либо операции результата, не укладывающегося в разрядную сетку) вычисления автоматически останавливаются. Другой возможный вариант состоит в том, что при переполнении разрядной сетки в машине вырабатывается специальный признак; дальше в программе можно предусмотреть операцию условного перехода, которая — в зависимости от того, было ли переполнение в предыдущей операции,— определенным образом изменит порядок выполнения последующих команд.

1.5.2. Плавающая запятая.

При использовании этого метода указания положения запятой разрядная сетка машины разбивается на две части; в одной из них записывается само число, другая часть служит для указания положения запятой.

Обычный прием при этом состоит в следующем. Если используется система счисления с основанием *n,* то любое число a можно представить в виде

$$\alpha — n^a A,$$

где $1 > |A| \geq 1/n$. Например, в десятичной системе число 462,83 может быть представлено в виде $10^3 \cdot 0{,}46283$, число 0,0046283 — в виде $10^{-2} \cdot 0{,}46283$ и т. д. Исключение часто делают для числа нуль, для которого принимается $A = 0$.

Далее в той части разрядной сетки, которая отводится для изображения самого числа, записывается величина *A,* в той части, которая отведена для указания положения запятой,— величина *a.* Величину *A* называют *мантиссой числа* (а разряды, отведенные для нее в разрядной сетке,— разрядами мантиссы), величину *a* — *порядком числа* (а соответствующую часть разрядной сетки машины— разрядами порядка). Термин *n-ичный порядок* означает, что порядок является показателем степени при *n.* Как порядок, так и мантисса числа записываются в своих разрядах в виде чисел с фиксированной запятой; мантисса записывается в виде дробного числа, с запятой перед старшим из основных разрядов (не считая, может быть, знакового разряда), порядок — обычно в виде целого числа, с запятой после самого младшего разряда.

Условие, принятое нами относительно мантиссы,

$$1 > |A| \geq 1/n$$

означает, что старший из основных разрядов мантиссы (с весом $1/n$) должен содержать ненулевую цифру. Если это условие выполняется, то число называется *нормализованным.*

Поскольку порядок представляет собой показатель степени при *n (n* — основание системы счисления), величина порядка фактически показывает, на сколько разрядов нужно перенести запятую в мантиссе, чтобы получить истинное значение числа. Знак порядка указывает, в какую сторону нужно перенести запятую: если порядок положителен, запятую нужно передвигать вправо, если отрицателен — влево.

При выполнении операций над числами с плавающей запятой машина должна по-разному обращаться с разрядами порядка и с разрядами мантиссы. Например, при выполнении умножения порядки сомножителей нужно сложить, а мантиссы — перемножить, при выполнении деления в порядках производится вычитание, а в мантиссах — деление и т. д.

В ходе выполнения арифметических операций в машине могут появляться ненормализованные числа, для которых абсолютная величина мантиссы меньше, чем $1/n,$ или больше единицы. Обычно в машине с плавающей запятой каждая операция заканчивается автоматической нормализацией результата. Для нормализации числа его мантиссу нужно сдвигать влево или вправо до тех пор, пока ее абсолютная величина не окажется в заданных пределах; при этом каждый сдвиг мантиссы влево на один разряд сопровождается вычитанием единицы из порядка, каждый сдвиг мантиссы вправо на один разряд — добавлением единицы к порядку. Поскольку, например, сдвиг мантиссы вправо на один разряд эквивалентен уменьшению числа в *n* раз, а добавление единицы к порядку — увеличению числа в *n* раз, в процессе нормализации истинное значение числа не изменяется (с точностью до младших разрядов мантиссы), а изменяется лишь его форма.

Порядок выполнения операций над числами с плавающей запятой подробно рассмотрен в разделах 3.2 и 4.8.

В машине с плавающей запятой легко получить очень широкий диапазон чисел. При использовании десятичной системы счисления для изображения десятичного порядка обычно отводится 1 ÷ 2 десятичных разряда и разряд алгебраического знака; при использовании двоичной системы для изображения двоичного порядка в обычных случаях отводят 5 ÷ 6 двоичных разрядов и разряд знака.

Если, например, для изображения двоичного порядка отведено 5 двоичных разрядов (кроме знака), то это означает, что величина порядка может быть заключена в пределах от +31 до —31 (по двоичной системе от +11111 до —11111), либо, может быть, в пределах от +31 до —32 (если используется дополнительный код для_отрицательных порядков). Поскольку мантиссы по абсолютной величине заключены в пределах от 1 до $^1/_2$, диапазон абсолютных величин чисел при этом будет ограничен сверху величиной

$$2^{+31}\cdot 1 \approx 2{,}15\cdot 10^9,$$

снизу — величиной

$$2^{-31}.{}^1/_2 \approx 2{,}33\cdot 10^{-10}$$

или, может быть, величиной

$$2^{-32}.\ {}^1/_2 \approx 1{,}17.\ 10^{-10}.$$

Ясно, что, имея такой диапазон чисел, программист может при подготовке большинства задач почти не задумываться над выбором масштабов для входящих в нее величин. В этом — главное преимущество представления чисел с плавающей запятой перед фиксированной запятой.

Здесь, однако, необходимо учитывать одно важное обстоятельство. Подготавливая для машины с фиксированной запятой задачу, в которой расчет масштабов чисел представляет значительные трудности, программист может использовать следующий искусственный прием. Пусть в машине с фиксированной запятой все числа изображаются так же, как в машине с плавающей запятой: разрядная сетка разделена на две части, одна часть содержит порядок числа, другая — мантиссу. В машине при этом не предусмотрена, разумеется, возможность непосредственно выполнять арифметические операции над числами в такой записи. Но эти операции можно выполнять по специальным подпрограммам.

Подпрограмма любого арифметического действия будет начинаться с разделения чисел на порядки и мантиссы; далее с порядками и мантиссами можно будет оперировать отдельно, как с числами с фиксированной запятой. Например, в подпрограмме умножения вслед за разделением сомножителей на порядки и мантиссы должны быть предусмотрены такие операции, как сложение порядков, перемножение мантисс, сравнение произведения мантисс с некоторой константой (для того чтобы выяснить, не требуется ли нормализация результата), затем — если необходимо — сдвиг произведения мантисс на один разряд влево и вычитание единицы из суммы порядков. Заканчивается подпрограмма объединением результата операций над порядками и результата операций над мантиссами в единую ячейку результата, который должен быть записан в такой же форме, как исходные числа.

Конечно, использование такого приема сильно снизит эффективную скорость машины: вместо выполнения одной операции придется каждый раз выполнять целую подпрограмму, в которую входят многочисленные обращения к оперативной памяти (чтение команд, запись и чтение промежуточных результатов). Возрастает и количество ячеек, занятых в оперативной памяти при решении задачи. Зато, пользуясь указанным приемом, можно для каждой задачи применять наиболее выгодное для нее распределение разрядной сетки машины. В задачах, где требуется не очень широкий диапазон чисел, но желательно получить возможно большую точность вычислений, можно для представления порядков отвести всего 3—4 двоичных разряда, а остальные сохранить для мантисс. Если, наоборот, нужен очень широкий диапазон чисел, а точность не играет особой роли, можно отвести на порядки, скажем, половину всех разрядов числа и т. д. При использовании плавающей запятой распределение разрядов разрядной сетки заложено заранее в конструкции машины; такая машина оперирует с большим, но раз и навсегда заданным диапазоном чисел и с заранее определенной точностью. Иногда это является недостатком машины с плавающей запятой.

Плавающая запятая менее удобна, чем фиксированная, и в других случаях, когда распределение разрядов в разрядной сетке не соответствует той схеме, которая принята заранее для чисел с плавающей запятой. Наиболее важный из этих случаев — это операции над командами. В универсальных вычислительных машинах команда размещается, как правило, в обычной разрядной сетке машины. При этом часть разрядов отводится для изображения кода операции, определенные группы разрядов — для изображения адресов и т. д. Обычно команда занимает всю разрядную сетку — и разряды порядка, и разряды мантисс, если речь идет о машине с плавающей запятой,— но при выполнении операций над командами величины, которые находятся в этих разрядах, не имеют, конечно, смысла порядков и мантисс и оперировать с ними нужно по-другому.

Некоторые логические (а не вычислительные) функции вычислительных машин также проще осуществлять с помощью машин с фиксированной запятой.

Поэтому области применения фиксированной и плавающей запятой можно примерно описать следующим образом:

только фиксированная запятая —

— в специализированных машинах, предназначенных для решения какой-либо одной задачи или узкого круга задач (для одной данной задачи расчет масштабов чисел, как бы сложен он ни был, можно

выполнить заранее; при этом достигается упрощение конструкции машины и экономия оборудования);

— в сверхмалых вычислительных машинах (выполняемая автоматически часть программы невелика и легко обозрима; расчет масштабов чисел не представляет труда);

— в машинах, предназначенных для статистических, бухгалтерских и других подобных расчетов (диапазон чисел невелик);

— в машинах, предназначенных специально для решения логических задач — игровых, тактических и т. д.;

только плавающая запятая —

— в малых, и может быть, средних универсальныхмашинах, предназначенных для научных и инженерных расчетов (для большинства задач диапазон чисел и точность, которые обычно обеспечиваются машиной с плавающей запятой, оказываются достаточными, никаких искусственных приемов при программировании применять ненужно; применение только фиксированной запятой сильно затруднило бы подготовку задач; совмещение плавающей и фиксированной запятой неоправданно усложнило бы машину);

плавающая и фиксированная запятая —

— в больших и, может быть, средних универсальных машинах, предназначенных для научных и инженерных расчетов (обеспечивается максимальная универсальность машины при решении самых различных классов задач; некоторое усложнение арифметического устройства машины не играет особой роли в общем объеме оборудования).

1.6. Двоичные коды для десятичных цифр

1.6.1. Количество различных двоичных кодов для десятичных цифр.

При использовании в вычислительных машинах десятичной системы счисления десятичные цифры почти никогда не изображаются непосредственно. Как указывалось в разделе 1.2, большей частью применяется изображение десятичных цифр с помощью четырех двоичных цифр; иногда применяется также изображение десятичных цифр посредством пяти, шести или даже семи двоичных цифр. Четверку двоичных разрядов, используемых для изображения одной десятичной цифры, часто называют *тетрадой.*

Если вместо каждой десятичной цифры записываются четыре двоичные, то наиболее естественным кажется представить в двоичной системе числа, соответствующие десятичным цифрам, позиционным способом с естественными весами разрядов. Изображения десятичных цифр при этом будут такими, как в качестве примера было показано в 1.2.2:

$(0)_{10} = 0000$, $(5)_{10} = 0101$,
$(1)_{10} = 0001$, $(6)_{10} = 0110$,
$(2)_{10} = 0010$, $(7)_{10} = 0111$,
$(3)_{10} = 0011$, $(8)_{10} = 1000$,
$(4)_{10} = 0100$, $(9)_{10} = 1001$.

Этот код называют иногда *кодом* «8, 4, 2, 1» (так как именно таковы естественные веса двоичных разрядов).

Однако ясно, что это далеко не единственный способ изображения десятичных цифр посредством четырех двоичных.

В четырех двоичных разрядах всего может быть 16 различных комбинаций. Из них для изображения десяти десятичных цифр нужно выбрать только 10, остальные 6 комбинаций остаются неиспользованными (в коде «8, 4, 2, 1», например, не использованы комбинации 1010, 1011, 1100, 1101, 1110, 1111). Количество способов, которыми могут быть выбраны 10 комбинаций из 16, равно числу сочетаний из 16 элементов по $10 : C_{16}^{10} = \frac{16!}{10!\cdot 6!}$.

Но и после того, как выбор 10 комбинаций сделан, возможно еще 10! различных вариантов построения кода для десятичных цифр. Число 10! —это количество способов, которым выбранные 10 комбинаций могут быть поставлены в соответствие 10 десятичным цифрам (число перестановок из 10 элементов, $P_{10} = 10!$). Итого, общее количество различных вариантов записи 10 десятичных цифр с помощью 4 двоичных равно количеству размещений из 16 элементов по 10;

$$A_{16}^{10} = C_{16}^{10} P_{10} = \frac{16!}{6!} \approx 2{,}9 \cdot 10^{10}.$$

Еще во много раз больше число вариантов, которыми можно десять десятичных цифр записать с помощью пяти, шести или семи двоичных цифр. Например, количество различных вариантов 5-разрядных двоичных кодов для десяти десятичных цифр равно (поскольку в пяти двоичных разрядах возможны 32 комбинации):

$$A_{32}^{10} = \frac{32!}{22!} \approx 2{,}35 \cdot 10^{14}.$$

Из этого количества различных двоичных кодов для десятичных цифр изучено всего 2—3 десятка кодов, а практическое применение получили едва 5—6 разных вариантов. Рассматриваемые ниже примеры показывают,

что среди множества возможных вариантов могут быть найдены коды с очень интересными свойствами, существенно упрощающие выполнение арифметических операций над числами или дающие какие-либо другие важные преимущества. Однако достижения, которые имеются в этом направлении, носят пока характер более или менее случайных находок; какой-либо систематизации двоичных кодов для десятичных цифр еще нет, отсутствуют и обобщающие работы по этому вопросу.

1.6.2. Код с излишком 3 и другие коды с аналогичными свойствами.

Наряду с кодом «8, 4, 2, 1» широкое распространение получил также способ двоичного кодирования десятичных цифр, известный под названием *код «с излишком* 3». В этом коде десятичные цифры записываются следующим образом:

$(0)_{10} = 0011$, $(5)_{10} = 1000$,
$(1)_{10} = 0100$, $(6)_{10} = 1001$,
$(2)_{10} = 0101$, $(7)_{10} = 1010$,
$(3)_{10} = 0110$, $(8)_{10} = 1011$,
$(4)_{10} = 0111$, $(9)_{10} = 1100$.

Если двоичным разрядам приписать их естественные веса 8, 4, 2, 1, то величина, записанная в четверке двоичных разрядов, отличается на 3 от истинного значения десятичной цифры, которой она соответствует. Например, комбинация ООП является двоичным представлением числа 3, соответствует же она десятичной цифре 0; величина 0100 представляет в двоичной системе число 4, соответствует же она цифре 1 и т. д.

В коде с излишком 3 удобно осуществляются сложение и вычитание по десятичной системе. Действительно, пусть необходимо, например, сложить десятичные цифры *b* и *c*. Их двоичные изображения, как условлено, соответствуют величинам

$$b + 3$$

и

$$c + 3$$

Произведя сложение по правилам д в о и ч н о й арифметики, получим величину

$$b + c + 6.$$

Если при этом возникает перенос влево от старшего двоичного разряда тетрады, то это значит, что

$$b + c + 6 \geq 16.$$

Но это в то же время означает, что

$$b + c \geq 10,$$

т. е. что при сложении цифр *b* и *c* по правилам д е с я т и ч ной арифметики должен возникнуть перенос в следующий десятичный разряд.

В данном десятичном разряде результат сложения должен быть равен либо *b* + с (если нет переноса в следующий разряд), либо *b* + *c* — 10 (если есть перенос в следующий разряд). Так как результат сложения должен быть представлен — подобно слагаемым — кодом с излишком 3, тетрада двоичных разрядов в результате должна содержать соответственно либо

$$(b + c) + 3,$$

либо

$$(b + c — 10) + 3.$$

На самом же деле, действуя по правилам двоичной арифметики, мы получим в данной тетраде соответственно либовеличину *b* + *c* + 6 (если из данной тетрады не было переноса в следующую), либо *b* + с— 10 (если из данной тетрады был перенос); для получения правильного результата нужно в первом случае произвести в данной тетраде вычитание числа 3, во втором —добавление числа 3.

Итак, для того чтобы выполнить сложение двух десятичных чисел в коде с излишком 3 нужно сначала сложить изображения этих чисел по правилам двоичной арифметики, а затем во всех тетрадах произвести добавление или вычитание числа 3 в зависимости от того соответственно, был перенос из данной тетрады или не был.

П р и м е р . Пусть необходимо выполнить сложение 3542 + 1476. В коде с излишком 3 слагаемые имеют вид 0110 1000 0111 0101 и 0100 0111 1010 1001.

Сложение по правилам двоичной арифметики дает

```
 0110  1000  0111  0101
+0100  0111  1010  1001
------------------------
 1011  0000  0001  1110
(0)   (1)   (1)   (0)
```

Цифры, которые записаны внизу в скобках, показывают, был ли перенос из находящейся справа тетрады. Добавление и вычитание по тетрадам числа 3 (в двоичной системе 11) в соответствии с указанным выше правилом дает

1011	0000	0001	1110
(—11)	(+11)	(+11)	(—11)
1000	0011	0100	1011

В коде с излишком 3 результат является изображением числа 5018, как и должно было получиться при сложении.

В следующих разделах мы увидим, что при использовании кода с излишком 3 удобно выполнять и некоторые другие операции, например удвоение десятичных чисел.

В этом коде весьма просто найти дополнение от любой десятичной цифры до 9 (что необходимо при преобразовании десятичного числа в дополнительный или в обратный код —см. 1.4.2 и 1.4.3). Для этого достаточно в двоичном изображении десятичной цифры заменить все двоичные цифры на обратные (т. е. нули заменить единицами, единицы — нулями).

Действительно, двоичное изображение какой-либо десятичной цифры a в рассматриваемом коде соответствует величине $a + 3$. Заменяя в данной тетраде все двоичные цифры на противоположные, мы тем самым находим дополнение от этой величины до числа 15 (по двоичной системе 1111). Таким образом, замена всех двоичных цифр на противоположные дает (в указанной тетраде) вместо величины $a + 3$ величину

$$15 — (a + 3) = (9 — a) + 3,$$

что является изображением десятичной цифры $9 — a$ в коде с излишком 3.

Например, заменив на противоположные все двоичные цифры в изображении десятичной цифры 0 (0011), получим 1100 — изображение цифры 9; заменив на противоположные все двоичные цифры в изображении десятичной цифры 1 (0100), получим 1011 —изображение десятичной цифры 8 и т. д.

Последним свойством обладает не только код с излишком 3, но и множество других кодов. Чтобы построить код, обладающий указанным свойством, нужно поступить следующим образом.

Все десятичные цифры необходимо сгруппировать в 5 взаимно дополнительных пар:

0 и 9; 1 и 8; 2 и 7; 3 и 6; 4 и 5.

16 комбинаций, которые возможны в четырех двоичных разрядах, следует сгруппировать в 8 взаимно обратных пар:

0000 и 1111; 0001 и 1110; 0010 и 1101; 0011 и 1100;

0100 и 1011; 0101 и 1010; 0110 и 1001; 0111 и 1000.

Далее нужно выбрать любые пять взаимно обратных пар комбинаций двоичных цифр и поставить их в соответствие пяти взаимно дополнительным парам десятичных цифр причем одна из двоичных комбинаций пары должна соответствовать одной десятичной цифре, другая—другой. Например, в коде с излишком 3 паре взаимно дополнительных десятичных цифр 0 и 9 соответствует пара взаимно обратных двоичных комбинаций 0011 и 1100 (причем комбинация 0011 изображает цифру 0, а комбинация 1100 — цифру 9); паре десятичных цифр 1 и 8 соответствует пара двоичных комбинаций 0100 и 1011 и т. д. Ясно, что и в любом другом двоичном коде для десятичных цифр, построенном по этому способу, отыскание дополнений от десятичных цифр до 9 будет выполняться так же, как в коде с излишком 3,— путем обращения всех двоичных цифр.

Количество способов, которыми из восьми взаимно обратных двоичных комбинаций можно выбрать пять и поставить их в соответствие пяти взаимно дополнительным парам десятичных цифр, равно количеству размещений из 8 элементов по 5:

$$A_8^5 = \frac{8!}{3!} = 6720 .$$

Когда пары двоичных комбинаций каким-либо образом поставлены в соответствие парам десятичных цифр, возможны еще 32 различных варианта построения кода (так как в каждой из пяти пар соответствие между двоичными комбинациями и десятичными цифрами может быть установлено двумя различными способами; $32=2^5$).

Итого количество различных двоичных кодов для десятичных цифр, в которых дополнение- до 9 получается обращением двоичных цифр, равно

$$A_8^5 \cdot 2^5 = \frac{8!}{3!} \cdot 32 = 215040 .$$

Очевидно, что количество различных 5-разрядных двоичных кодов для десятичных цифр, обладающих аналогичным свойством, равно

$$A_{16}^5 \cdot 2^5 = \frac{16!}{11!} \cdot 32 \approx 1{,}67 \cdot 10^6 .$$

Практическое приложение из этого множества кодов нашел, кроме кода с излишком 3, еще так называемый

код «2, 4, 2, 1». В нем десятичные цифры представлены четырьмя двоичными разрядами позиционным способом — правда, с искусственным весом для старшего из разрядов; цифры в названии кода как раз и соответствуют весам двоичных разрядов (старший разряд вместо естественного веса 8 имеет вес 2). Десятичные цифры в коде «2, 4, 2, 1» записываются в виде

$(0)_{10} = 0000$, $(5)_{10}=1011$,
$(1)_{10} = 0001$, $(6)_{10}=1100$,
$(2)_{10} = 0010$, $(7)_{10}=1101$,
$(3)_{10} = 0011$, $(8)_{10}=1110$,
$(4)_{10} = 0100$, $(9)_{10}=1111$.

Впрочем, здесь возможны и некоторые варианты. Например, десятичную цифру 4 можно записать в виде 1010, а цифру 5 — в виде 0101 и т. д.

Если требовать, чтобы веса разрядов были положительны, то, как можно показать, имеется еще только три кода, в которых десятичные цифры записываются позиционным способом четырьмя двоичными разрядами и в которых дополнения от десятичных цифр до 9 получаются обращением двоичных цифр. Это — коды с весами двоичных разрядов 5, 2, 1, 1; 4, 3, 1, 1; 3, 3, 2, 1. Общим для них (и для кода «2, 4, 2, 1») является то, что сумма весов равна 9.

Заметим здесь, что код с излишком 3 является по существу не позиционным, а символическим способом записи десятичных цифр. Однако сложение и другие операции над десятичными числами выполняются в нем удобнее, чем, например, в позиционном коде «2, 4, 2, 1».

Если не требовать, чтобы веса двоичных разрядов были положительны, то можно предложить еще несколько позиционных способов записи десятичных цифр, в которых дополнения до 9 образуются по тем же правилам, что и в указанных выше кодах. Таков, например, код с весами двоичных разрядов 8, 4, —2, —1 *(код* «8, 4, —2, —1»).

Десятичные цифры в нем записываются в виде

$(0)_{10} = 0000$, $(5)_{10}= 1011$,
$(1)_{10} = 0111$, $(6)_{10}= 1010$,
$(2)_{10} = 0110$, $(7)_{10}= 1001$,
$(3)_{10} = 0101$, $(8)_{10}= 1000$,
$(4)_{10} = 0100$, $(9)_{10}= 1111$.

Легко видеть, что от записи десятичных цифр кодом с излишком 3 приведенная запись отличается тем, что две младшие двоичные цифры заменены на обратные. Это замечание может оказаться полезным при разработке способов выполнения операций над числами в коде с излишком 3 или в рассмотренном коде («8, 4, —2, —1»).

Среди кодов, в которых каждая десятичная цифра изображается п я т ь ю двоичными разрядами, имеется два кода, очень схожих по своим свойствам с кодом с излишком 3.

Один из них — *код* «с *излишком* 11» (или *код «а* + 11»). В нем пяти двоичным разрядам присвоены их естественные веса 16, 8, 4, 2, 1, но вместо любой десятичной цифры *а* в этих разрядах записывается величина *а* + 11. Свойства этого кода полностью аналогичны свойствам кода с излишком 3.

В другом из кодов двоичным разрядам тоже приписаны их естественные веса, но вместо любой десятичной цифры *а* пятерка двоичных разрядов содержит величину 3*а* + 2 (так называемый *код* «З*а* + 2»). Десятичные цифры при этом, очевидно, изображаются следующим образом:

$(0)_{10} = 00010$, $(5)_{10} = 10001$,
$(1)_{10} = 00101$, $(6)_{10} = 10100$,
$(2)_{10} = 01000$, $(7)_{10} = 10111$,
$(3)_{10} = 01011$, $(8)_{10} = 11010$,
$(4)_{10} = 01110$, $(9)_{10} = 11101$.

Легко видеть, что дополнения от десятичных цифр до У здесь также получаются заменой на обратные всех двоичных цифр: замена всех двоичных цифр на обратные эквивалентна вычитанию из числа 31 (так как $(11111)_2 = (31)_{10}$); при этом вместо величины З*а*+2 получается величина 31 — (З*а* + 2) = 3·(9 — *а*) + 2, которая является изображением цифры 9 — *а.*

При сложении двух десятичных цифр (*b* и *c*), записанных указанным образом, если сложение выполняется по правилам двоичной арифметики, получим величину (3*b* + + 2) + (3*c* + 2) = 3 *(b* + *c*) + 4. При этом наличие двоичного переноса влево от старшего разряда пятерки двоичных разрядов означает, что

$3(b+ c) + 4 \geq 32$.

Так как при $b + c$ — 9 имели бы 3 $(b + c)$ +4 = 31**,** то это неравенство одновременно означает, что $b + c > 9$, т. е. что $b + c \geq 10$ (потому что *b* и *c* —"целые). Таким образом, двоичный перенос влево от старшего разряда пятерки возникает тогда и только тогда, когда при сложении записанных в них десятичных цифр по правилам десятичной арифметики должен возникнуть перенос в следующий десятичный разряд. В данном десятичном разряде в этом случае должна остаться цифра суммы, равная $b + c$ — 10; действуя по правилам двоичной арифметики, мы получаем в данной пятерке двоичных разрядов величину

$$(3b + 2) + (3c + 2) — 32 = 3\,(b + c — 10) + 2,$$

что как раз и является изображением цифры $b + c — 10$.

В случае, когда перенос из данного десятичного разряда отсутствует, десятичная цифра суммы должна быть равна просто $b + c$. Чтобы получить правильное изображение этой цифры ($3\,(b + c) + 2$), нужно вычесть 2 из результата сложения, выполненного по правилам двоичной арифметики ($(3b + 2) + (3c + 2) = 3\,(b + c) + 4$).

Важным достоинством кода «3а + 2» по сравнению с кодом с излишком 3 является возможность обнаружения наиболее вероятных ошибок при передаче чисел, о чем сказано в разделе 1.7. Именно ради этого, может быть, и имеет смысл применять 5-разрядный двоичный код для изображения десятичных цифр вместо 4-разрядного кода.

1.6.3. Примеры других двоичных кодов десятичных цифр.

Для кодов, приведенных в 1.6.2, общим преимуществом является то, что дополнения от десятичных цифр до 9 получаются обращением двоичных цифр. У некоторых из них это едва ли не единственное достоинство.

Вообще говоря, если цель состоит в том, чтобы максимально упростить отыскание дополнений от десятичных цифр до 9, то можно составить такие коды, в которых эта операция будет даже проще, чем в рассмотренных выше. Можно, например, потребовать, чтобы дополнение от десятичной цифры до 9 получалось путем замены на обратную о д н о й п о с л е д н е й двоичной цифры кода. Аналогично предыдущему можно показать, что таких двоичных кодов для десятичных цифр может быть тоже 215040 (4-разрядных) или $1{,}67\text{-}10^6$ (5-разрядных).

Приводимые ниже примеры, однако, должны показать, что удачный выбор двоичного кода для десятичных цифр может обеспечить получение самых разнообразных — порой совершенно неожиданных — преимуществ. Примеры сведены в таблицу 1-6.

Т а б л и ц а 1-6

Примеры двоичных кодов для десятичных цифр

Десятичные цифры	а	б	в	г	д	е
	код 7,4,2,1	два из пяти	(*w, x, y, z*)			код 5,4,2,1
0	0000	11000	0000	0000	0000	0000
1	0001	00011	0001	0110	0001	0001
2	0010	00101	0011	0011	0011	0010
3	0011	00110	1000	0111	0010	0011
4	0100	01001	0110	1111	0110	0100
5	0101	01010	1111	0101	0111	1000
6	0110	01100	1001	1101	0101	1001
7	1000	10001	0010	1001	0100	1010
8	1001	10010	1100	1100	1100	1011
9	1010	10100	0100	1010	1000	1100

Код «7, 4, 2, 1», приведенный в столбце «а», предлагался для применения в релейных (электромеханических) вычислительных машинах, где двоичную единицу изображает замкнутое реле, а двоичный нуль — разомкнутое реле. При этом замкнутое реле потребляет ток от источника питания, разомкнутое не потребляет. В рассматриваемом коде использованы комбинации двоичных цифр с минимальным количеством единиц: комбинация 0000 (для десятичной цифры 0), все 4 комбинации, содержащие по одной единице (0001—для цифры 1, 0010—для цифры 2, 0100 —для цифры 4, 1000—для цифры 7), и комбинации, содержащие по две единицы для остальных десятичных цифр. Поэтому использование этого кода обеспечивает в среднем минимальное потребление тока от источника питания.

Аналогичная задача может возникнуть и в электронных машинах — например, при использовании в них динамических триггеров. Потребление тока от источника питания для динамических триггеров тоже зависит от того, в каком состоянии находится триггер.

Иногда важно не столько уменьшить потребление тока от источника питания, сколько обеспечить ровную нагрузку на источник (чтобы предотвратить большие изменения падения напряжения на его внутреннем сопротивлении). При этом может потребоваться код, в котором изображения всех десятичных цифр содержат примерно одинаковые количества единиц. Его можно получить, например, заменив в столбце «а» таблицы 1-6 комбинацию 0000 (для изображения цифры 0) комбинацией 1100; при этом окажется, что в изображении любой десятичной цифры содержится одна или две единицы. Для сравнения напомним, что в коде «8, 4, 2, 1» изображение десятичной цифры может содержать от нуля до трех единиц ($(0)_{10} = 0000$, $(7)_{10} = 0111$).

С этой точки зрения еще лучший результат дают так называемые коды «два из пяти» и «три из пяти». Один из таких кодов приведен в столбце «б» таблицы 1-6. Каждая десятичная цифра в нем изображается пятью двоичными

разрядами, из которых два разряда содержат единицы, а три разряда — нули. Приведенный код получен из кода «7, 4, 2, 1», (с заменой комбинации для десятичного нуля на 1100) путем добавления справа одного разряда. Цифра дополнительного разряда выбрана так, чтобы всего в любой комбинации было две единицы. Можно считать, что вес дополнительного разряда равен нулю; соответственно приведенный код называют иногда кодом «7, 4, 2, 1, 0» (хотя десятичный 0 под общее правило не подходит).

Вообще построение кодов «два из пяти» и «три из пяти» возможно потому, что количество сочетаний из пяти элементов по два или из пяти элементов по три оказывается достаточным:

$$C_5^2 = C_5^3 = 10.$$

Десять сочетаний из пяти элементов по два могут быть поставлены в соответствие десяти десятичным цифрам 101 различными способами. Поэтому возможно 101 = 3 628 800 различных вариантов построения кода «два из пяти». Столько же возможно кодов «три из пяти».

Коды «два из пяти» и «три из пяти» дают также возможность обнаруживать наиболее вероятные ошибки при передаче чисел (о чем говорится в разделе 1.7).

Очень интересными свойствами обладает код, приведенный в столбце «в» таблицы 1-6. Если двоичные цифры обозначить соответственно *w, x, y, z,* как сделано в таблице, то для любой десятичной цифры комбинация *z, w, x, y* дает младшую цифру произведения на 3, комбинация *x, y, z, w* — младшую цифру произведения на 7, комбинация *y, z, w, x* — младшую цифру произведения на 9. Таким образом, младший разряд произведения десятичной цифры на 3, на 7 или на 9 может быть получен круговой перестановкой двоичных цифр.

Возьмем, например, десятичную цифру 2; ее изображение имеет вид 0011; выполнив перестановку *w, x, y, z* → →*z, w, x, y,* получим 1001, что является изображением цифры 6 (2 × 3 = 6); перестановка *w, x, y, z* → *x, y, z, w* дает комбинацию 0110, представляющую цифру 4 (младший разряд произведения 2×7= 14); перестановка *w, x, y, z*→ *y, z, w, x* дает 1100, т. е. цифру 8 (младший разряд произведения 2 × 9= 18).

Приведенный код заимствован из книги «Синтез электронных вычислительных и управляющих схем» (перев. с англ. под ред. В. И. Шестакова, ИЛ, 1953). Не видно, однако, каким образом он был найден.

В столбцах «г» и «д» таблицы 1-6 приведены еще два двоичных кода для десятичных цифр. Эти коды мы рассмотрим в разделах 1.7.5 и 1.8.1; здесь же никаких пояснений к ним приводить не будем. Читатель может сделать попытку самостоятельно догадаться, какими полезными свойствами обладают эти коды.

Построение кода «5, 4, 2, 1», приведенного в столбце «е» таблицы 1-6, очевидно. Интересно, что десятичные числа, записанные в этом коде, можно рассматривать как двоично-пятеричные; при этом три младших двоичных разряда каждой десятичной тетрады изображали бы один пятеричный разряд, старшая двоичная цифра соответствовала бы двоичному разряду. Код этот имеет некоторые достоинства при выполнении арифметических операций.

1.7. Помехозащищенные коды

Вопросы, рассматриваемые в настоящем разделе, возникли первоначально в основном в технике связи, а не в вычислительной технике. Именно по этой причине типовая ситуация, рассматриваемая здесь, состоит в том, что выполняется п е р е д а ч а (а не п е р е р а б о т к а) информации и что при этом с некоторой вероятностью могут возникать случайные ошибки. Передачу информации можно понимать в широком смысле этого слова — включая изменение физической формы сигналов, задержку их по времени, изменение порядка следования сигналов и т. п. Тем не менее практическую важность этих вопросов для вычислительной техники и их приложимость к вычислительной технике не следует переоценивать.

В большинстве случаев в дальнейшем предполагается, что при передаче числа наиболее вероятной является ошибка в одном разряде, менее вероятной — ошибка одновременно в двух разрядах, еще менее вероятной — одновременная ошибка в трех разрядах и т. д. Зачастую, однако, такое предположение оказывается слишком искусственным.

В зависимости от назначения и возможностей помехозащищенных кодов различают коды самокорректирующиеся и самоконтролирующиеся. Коды, позволяющие автоматически обнаруживать наиболее вероятные ошибки при передаче чисел, называются *самоконтролирующимися,* а коды, в которых возможно автоматическое исправление ошибок,— *самокорректирующимися.*

1.7.1. Коды Хемминга.

Коды Хемминга — наиболее известные и, вероятно, первые*) из самоконтролирующихся и самокорректирующихся кодов. Построены они применительно к двоичной системе счисления.

Пусть каждое число представлено *m* двоичными разрядами, и пусть наиболее вероятная ошибка состоит в искажении одного из них (вместо нуля принимается единица или наоборот).

Для построения *самоконтролирующегося* кода достаточно приписать к каждому числу один добавочный (контрольный) двоичный разряд и выбрать цифру этого разряда так, чтобы общее количество единиц в изображении

*) См. H a m m i n g R. W. Error correcting and error detecting codes, Bell Syst. Techn. J., 1950, 2.

любого числа было, например, четным. Одиночная ошибка в каком-либо из разрядов передаваемого числа (в том числе, может быть, и в контрольном разряде) изменит четность общего количества единиц. Счетчик по модулю 2, подсчитывающий количество единиц, которые содержатся среди двоичных цифр числа, мог бы давать сигнал о наличии ошибок.

При этом, разумеется, мы не получаем никаких указаний о том, в каком именно разряде произошла ошибка, и, следовательно, не имеем возможности исправить ее. Остаются незамеченными также ошибки, возникающие одновременно в двух, или в четырех или вообще в четном количестве разрядов. Впрочем, двойные, а тем более четырехкратные ошибки полагаются маловероятными.

Если количество информационных разрядов нечетно, то выгодно цифру контрольного разряда выбирать так, чтобы общее количество единиц в коде было нечетно. Тогда при наличии всех нулей в информационных разрядах контрольный разряд должен будет содержать единицу, а при наличии всех единиц в информационных разрядах цифра контрольного разряда будет нуль; это поможет обнаруживать ошибки, возникающие, может быть, по вине цепей управления, когда во всех разрядах вместо нужной информации передаются нули или, наоборот, единицы.

Иногда различают «контроль по четности» и «контроль по нечетности», но чаще, независимо от того, какое условие принято для выбора цифры контрольного разряда, этот метод контроля называют контролем по четности.

Для построения *самокорректирующегося* кода, рассчитанного на исправление одиночных ошибок, одного контрольного разряда недостаточно. Как видно из дальнейшего, количество контрольных разрядов k должно быть выбрано так, чтобы удовлетворялось неравенство

$$2^k \geq k + m + 1$$

или

$$k \geq \log_2 (k + m + 1),$$

где m — количество основных двоичных разрядов числа. Минимальные значения k при заданных значениях m, найденные в соответствии с этим неравенством, приведены в таблице 1-7.

Таблица 1-7

Минимальные количества контрольных разрядов, необходимые при заданном числе основных разрядов для построения самокорректирующегося кода

m	k	m	k	m	k	m	k
1	2	11	4	21	5	31	6
2	3	12	5	22	5	32	6
3	3	13	5	23	5	33	6
4	3	14	5	24	5	34	6
5	4	15	5	25	5	35	6
6	4	16	5	26	5	36	6
7	4	17	5	27	6	37	6
8	4	18	5	28	6	38	6
9	4	19	5	29	6	39	6
10	4	20	5	30	6	40	6

Имея $m + k$ разрядов, самокорректирующийся код можно построить следующим образом.

Присвоим каждому из разрядов свой номер — от 1 до $(m + k)$; запишем эти номера по двоичной системе. Поскольку $2^k > m + k$, каждый номер можно представить, очевидно, *k-разрядным* двоичным числом. Для определенности будем полагать, что при записи номеров разрядов по двоичной системе использован позиционный способ изображения чисел с естественными весами разрядов; вообще это условие не является необходимым.

Предположим далее, что все $m + k$ разрядов кода разбиты на контрольные группы (которые частично перекрываются), причем так, что единицы в двоичном представлении номера разряда указывают на его принадлежность к определенным контрольным группам. Например, разряд № 5 принадлежит к 1-й и 3-й контрольным группам, потому что в двоичном представлении его номера

$$(5)_{10} = (...000101)_2$$

1-й и 3-й разряды содержат единицы; разряд № 7 принадлежит 1-й, 2-й и 3-й контрольным группам, потому что в двоичном представлении его номера

$$(7)_{10} = (...0111)_2$$

содержатся единицы в 1-м, 2-м и 3-м разрядах и т. д.

Среди $m + k$ разрядов кода при этом имеется k разрядов, каждый из которых принадлежит только к одной контрольной группе. Это разряды, номера которых являются целыми степенями двойки и потому в двоичном представлении содержат по одной единице: разряд № 1, принадлежащий только к 1-й контрольной группе (потому что его номер по двоичной системе имеет вид ...000001), разряд № 2, принадлежащий только ко 2-й контрольной группе (так как $(2)_{10} = (...000010)_2$), ..., разряд $2^{(k-1)}$, принадлежащий только к *k-й* контрольной группе. Эти k разрядов мы и будем считать контрольными. Остальные m разрядов, каждый из которых принадлежит по меньшей мере к двум контрольным группам, будут основными разрядами числа (информационными разрядами).

При этом в каждой из k контрольных групп будем иметь по одному контрольному разряду. Например, в 1-ю контрольную группу входят все разряды, номера которых в двоичном представлении содержат единицу младшего разряда: 1-й, 3-й, 5-й, 7-й и т. д.; из них контрольным является 1-й. Во 2-ю контрольную группу входят все разряды, номера которых в двоичном представлении содержат единицу 2-го разряда: 2-й, 3-й, 6-й, 7-й и т. д.; из них контрольным является 2-й.

Содержимое основных m разрядов числа заранее задано: в них размещается передаваемая информация. В каждый из контрольных разрядов поместим такую цифру (0 или 1), чтобы общее количество единиц в его контрольной группе было четным. Так как каждый контрольный разряд принадлежит только к одной контрольной группе, цифры в контрольных разрядах не зависят друг от друга.

Таким образом формируется код числа.

После приема кода произведем контроль по четности отдельно в каждой контрольной группе. Если код был принят верно, то во всех контрольных группах количество единиц будет четным. Если в каком-либо разряде при передаче кода была допущена ошибка, то в тех контрольных группах, в которые этот разряд входит, количество единиц окажется нечетным. Но любой разряд кода входит в те контрольные группы, которые соответствуют единицам в его номере, записанном по двоичной системе. Следовательно, по результатам контроля, проведенного внутри контрольных групп, можно судить о номере разряда, принятого неверно. Изменив цифру этого разряда на противоположную, получим правильный код.

Пример. Пусть должны передаваться 4-разрядные двоичные числа — от 0000 до 1111 (т. е. от 0 до 15 по десятичной системе). Для построения самокорректирующегося кода, рассчитанного на исправление одиночных ошибок, необходимо добавить 3 контрольных разряда (см. таблицу 1-7: при m = 4, k= 3). Введем общую нумерацию для разрядов кода — от № 1 до № 7. Согласно сказанному выше, в 1-ю контрольную группу включим 1-й, 3-й, 5-й, 7-й разряды, во вторую — 2-й, 3-й, 6-й и 7-й разряды, в третью — 4-й, 5-й, 6-й и 7-й разряды. Контрольными разрядами будем считать 1-й, 2-й и 4-й. Само число разместим в 3-м, 5-м, 6-м и 7-м разрядах. Установив цифры в контрольных разрядах так, чтобы количество единиц в каждой контрольной группе было четным, получим такие изображения чисел, как показано в таблице 1-8. (В столбце «Д. К.» приведены цифры 8-го разряда—разряда «двойного контроля»— для кода, который рассматривается позже. Пока на этот столбец не нужно обращать внимания.)

Таблица 1-8

Изображение чисел от 0 до 15 в самокорректирующемся коде, рассчитанном на исправление одиночных ошибок

Обычная запись числа		Изображение в самокорректирующемся коде	
десятичная	двоичная	Номера разрядов 7 6 5 4 3 2 1	Д. К.
0	0000	0 0 0 0 0 0 0	0
1	0001	0 0 0 0 1 1 1	1
2	0010	0 0 1 1 0 0 1	1
3	0011	0 0 1 1 1 1 0	0
4	0100	0 1 0 1 0 1 0	1
5	0101	0 1 0 1 1 0 1	0
6	0110	0 1 1 0 0 1 1	0
7	0111	0 1 1 0 1 0 0	1
8	1000	1 0 0 1 0 1 1	0
9	1001	1 0 0 1 1 0 0	1
10	1010	1 0 1 0 0 1 0	1
11	1011	1 0 1 0 1 0 1	0
12	1100	1 1 0 0 0 0 1	1
13	1101	1 1 0 0 1 1 0	0
14	1110	1 1 1 1 0 0 0	0
15	1111	1 1 1 1 1 1 1	1

Предположим теперь для примера, что при передаче изображения числа 9 (т. е. кода 1001100) произошла ошибка в 5-м разряде, так что был принят код 1011100. Произведя в принятом коде контроль по четности внутри контрольных групп, мы обнаружили бы, что количество единиц нечетно в 1-й и 3-й контрольной группах и четно во 2-й контрольной группе. Это указывает, во-первых, на наличие ошибки и, во-вторых, означает, что номер ошибочно принятого разряда в двоичном представлении содержит единицу на первом и на третьем местах и нуль — на втором месте. Иначе говоря, номер разряда, в котором произошла ошибка, равен 5 (так как $(101)_2 = (5)_{10}$). Заменив в 5-м разряде принятую цифру на противоположную, получим правильный код.

Построенный код, разумеется, никак не рассчитан на возможность одновременной ошибки в двух разрядах числа. Когда в принятом коде производится контроль по четности внутри контрольных групп, случай двойной ошибки ничем внешне не отличается от случая одиночной ошибки; при этом вырабатывается некий номер разряда, в котором якобы имеется ошибка, но не имеющий на самом деле никакого отношения к тем разрядам, где ошибка

действительно произошла*). Исправление кода по общему правилу не только не улучшило бы, но даже ухудшило бы дело.

Если бы, скажем, в рассмотренном выше примере при передаче кода числа 9 (т. е. кода 1001100) ошибка произошла не только в 5-м, но и в 7-м разряде, то был бы принят код 0011100. Количество единиц в 1-й контрольной группе четно, во 2-й нечетно, в 3-й четно. Внешне дело выглядит так, как будто произошла ошибка при передаче 2-го разряда (так как $(010)_2 = (2)_{10}$). Однако исправление 2-го разряда дало бы код 0011110 — еще более далекий от правильного, чем принятый.

Можно построить и такой код, который обнаруживал бы двойные ошибки и исправлял одиночные. Для этого к самокорректирующемуся коду, рассчитанному на исправление одиночных ошибок, нужно приписать еще один контрольный разряд (разряд двойного контроля). Полное количество разрядов кода при этом будет равно $m + k + 1$, где m—количество разрядов информации, k определяется по таблице 1-7. Цифра в разряде двойного контроля устанавливается такой, чтобы общее количество единиц во всех $m + k + 1$ разрядах кода было четным. Этот разряд не включается в общую нумерацию разрядов и не входит ни в одну контрольную группу.

После приема кода контроль по четности будет производиться раздельно по контрольным группам и для всего кода в целом. При этом могут быть следующие случаи:

(1) в принятом коде в целом и по всем контрольным группам количество единиц четно;
(2) в принятом коде в целом количество единиц нечетно, но во всех контрольных группах количество единиц четно;
(3) в принятом коде в целом и в некоторых из контрольных групп количество единиц нечетно;
(4) в принятом коде в целом количество единиц четно, но в некоторых контрольных группах имеется нечетное количество единиц.

Если тройные ошибки и ошибки в большем количестве разрядов исключаются, то первый случай соответствует безошибочному приему кода, второй случай — ошибке только в разряде двойного контроля, третий случай — одиночной ошибке в каком-либо из остальных разрядов (которую можно исправить в соответствии с приведенными выше правилами), четвертый случай—двойной ошибке. Исправление двойных ошибок здесь, конечно, невозможно. Пример такого кода приведен в таблице 1-8, где колонка «Д. К.» дает цифры разряда двойного контроля.

Увеличивая дальше количество контрольных разрядов, можно было бы построить коды, рассчитанные на исправление двойных ошибок, на исправление двойных ошибок и обнаружение тройных и т. д. Однако методы построения этих кодов не вполне разработаны.

1.7.2. Геометрическая интерпретация.

Иногда полезно геометрическое представление кодов Хемминга и других помехозащищенных кодов.

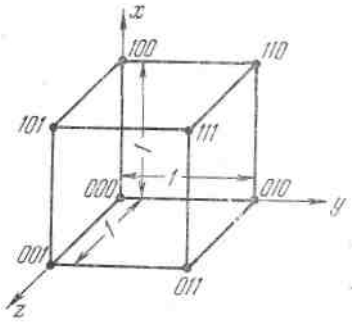


Рис. 1-1. Единичный 3-мерный куб, соответствующий 3-разрядному двоичному коду.

Каждый разряд n-разрядного двоичного числа можно поставить в соответствие с одной из координат в n-мерном пространстве. Тогда любая из 2^n комбинаций двоичных цифр, которые могут быть записаны в n двоичных разрядах, будет соответствовать одной из вершин n-мерного единичного куба, расположенного так, что одна из вершин находится в начале координат, а ребра направлены вдоль осей.

На рис. 1-1 показан такой куб в 3-мерном пространстве. Считается, что ось z соответствует первому справа разряду кода, ось y — второму разряду, ось x— третьему. Восемь вершин куба соответствуют восьми различным комбинациям цифр, возможным в трех двоичных разрядах.

Расстоянием между двумя вершинами называется минимальное количество ребер, которые нужно пройти, чтобы попасть из одной вершины в другую. Например, на рис. 1-1 расстояние от вершины 000 до вершины 001 равно одной единице, расстояние от вершины 000 до вершины 011 равно двум единицам, а до вершины 111 —трем. Расстояние между двумя комбинациями цифр равно фактически количеству разрядов, в которых эти комбинации различны. Если при передаче кодов допускаются все 2^n комбинаций, которые могут быть в n двоичных разрядах, то минимальное расстояние между двумя допустимыми комбинациями равно одной единице. Допустимые комбинации занимают при этом все вершины куба. Ни контроль, ни обнаружение каких-либо ошибок невозможны: ошибка в любом разряде передаваемой комбинации сдвигает ее на единицу, причем снова получается допустимая комбинация; никаких признаков того, что при передаче комбинации произошла ошибка, нет.

Предположим теперь, что из n двоичных разрядов кода один является контрольным, а передаваемое число содержится в остальных n — 1 разрядах. При этом допустимы всего 2^{n-1} комбинаций: в основных n — 1 разрядах возможны любые комбинации, а цифра контрольного разряда по какому-либо правилу однозначно связана с

*) Очевидно, что двойная ошибка не может компенсироваться во всех контрольных группах. В случае двойной ошибки хотя бы в одной контрольной группе количество единиц окажется нечетным. Отсюда видно, что приведенный код можно использовать вместо исправления одиночных ошибок для обнаружения (без исправления) одиночных и двойных ошибок.

цифрами остальных разрядов. Из 2^n вершин n-мерного куба задействована только половина (2^{n-1}), остальным не соответствуют какие-нибудь используемые комбинации. Есть надежда так расставить используемые вершины, чтобы минимальное расстояние между ними было равно двум единицам. Одиночная ошибка при передаче кода сдвинет соответствующую комбинацию на одну единицу; таким образом мы попадем в вершину куба, которая не соответствует никакой из допустимых комбинаций. Это и будет являться признаком одиночной ошибки.

Доказательством того, что 2^{n-1} комбинаций действительно можно разместить в вершинах n-мерного куба указанным образом, является построение самоконтролирующегося кода Хемминга, содержащего один контрольный разряд и рассчитанного на исправление одиночных ошибок (см. 1.7.1).

Обратимся вновь к примеру, который иллюстрируется рис. 1-1. Пусть из трех разрядов кода два являются информационными (второй и третий справа), а один — первый справа — контрольным; цифра контрольного разряда выбирается так, чтобы общее количество единиц было четным. При этом допустимыми будут комбинации 000, 011, 101, 110. На рис. 1-2,*а* соответствующие вершины куба отмечены точками.

Расстояние между двумя ближайшими допустимыми комбинациями теперь равно двум единицам. Следовательно, одиночная ошибка, которая сдвигает комбинацию на расстояние 1, может быть обнаружена. Однако при этом ее нельзя исправить, потому что каждая неотмеченная вершина куба отстоит на равные расстояния от трех отмеченных вершин. Если, скажем, принята комбинация 111, то нельзя выяснить, почему это произошло: вследствие ошибки в 1-м справа разряде при передаче комбинации 110, или вследствие ошибки во 2-м

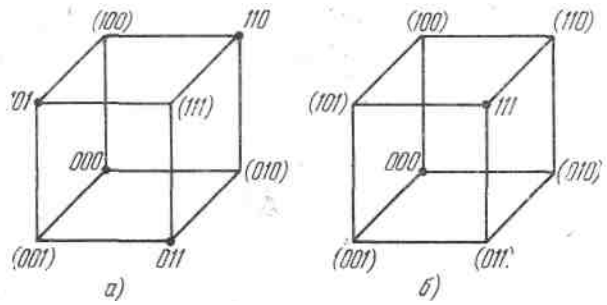


Рис. 1-2. Расположение допустимых комбинаций: *а)* в коде, рассчитанном на обнаружение одиночных ошибок; *б)* в коде, рассчитанном на исправление одиночных ошибок или на обнаружение одиночных и двойных ошибок.

разряде при передаче комбинации 101, или вследствие ошибки в 3-м разряде при передаче комбинации 011; нет, следовательно, способа исправить допущенную ошибку.

Чтобы одиночные ошибки можно было не только обнаруживать, но и исправлять, расстояние между допустимыми комбинациями должно быть не меньше трех единиц. Такой код для $n = 3$ иллюстрируется рис. 1-2,*б*. В случае одиночной ошибки получается комбинация, соответствующая неотмеченной вершине, но она находится «ближе» к одной из отмеченных вершин, чем к другим; последнее обстоятельство и позволяет исправлять одиночные ошибки.

Если возможны не только одиночные, но и двойные ошибки, то код с минимальным расстоянием 3 между допустимыми комбинациями не пригоден для исправления ошибок, но дает возможность обнаруживать и одиночные, и двойные ошибки.

Продолжив это рассуждение дальше, можно утверждать, что для исправления одиночных и обнаружения двойных ошибок или для обнаружения одиночных, двойных и тройных ошибок необходим код с минимальным расстоянием между допустимыми комбинациями в 4 единицы, для исправления двойных ошибок (или для обнаружения всех ошибок вплоть до четырехкратных) минимальное расстояние должно быть 5 единиц и т. д. К этим критериям мы вернемся еще в 1.7.3.

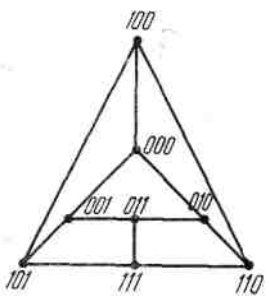


Рис. 1-3. Сетка для графического изображения 3-разрядного кода.

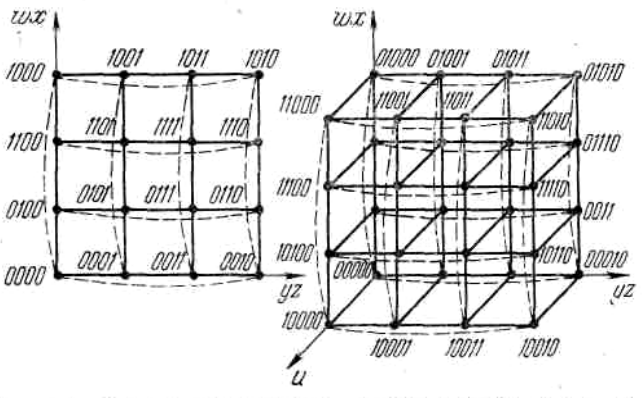


Рис. 1-4. Условная развертка n-мерного куба (n=4 и n=5).

Приведенная геометрическая интерпретация дает возможность получить наглядное пространственное изображение кода только в очень ограниченных случаях — для n=2 и n = 3. Впрочем, вместо n-мерного куба для графического изображения кода можно воспользоваться сеткой, имеющей 2^n узлов, каждый из которых соединен

с n другими узлами. Пример такой сетки для $n = 3$ показан на рис. 1-3. Для случаев $n = 4$ и $n = 5$ сетку можно изобразить в виде условной развертки n-мерного куба, как показано на рис. 1-4. Пунктирными линиями на этом рисунке показаны места, где должна «сшиваться» развертка; расстояние между двумя узлами, соединенными пунктирной линией, равно единице.

1.7.3. Некоторые теоретико-информационные аспекты. Общие замечания.

Задача, рассматриваемая нами в настоящем разделе, по сути дела, совпадает с одной из основных проблем теории информации.

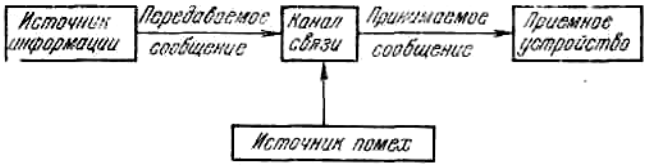


Рис. 1-5. Система связи при наличии помех.

Рис. 1-5 иллюстрирует ситуацию, которая изучается теорией информации: источник информации по имеющемуся каналу связи передает некоторое сообщение, воспринимаемое приемным устройством; вследствие наличия помех в канале (или на его входе, или на его выходе) сообщение, получаемое приемным устройством, отличается от сообщения, переданного источником информации. Задача состоит в том, чтобы, зная принятое сообщение, определить, какое сообщение было передано. Тот факт, что в интересующем нас случае каждое переданное сообщение является одним из к о н е ч н о г о множества возможных сообщений, существенно облегчает дело.

Закономерности, управляющие выбором того или иного сообщения из множества возможных, связаны со смысловым значением этих сообщений и настолько сложны, что такой выбор можно считать случайным. Точно так же случайный характер носят искажения, создаваемые помехами. Этим оправдывается применение вероятностных методов исследования.

Приведем определения некоторых основных понятий и некоторые важные для нас выводы теории информации*).

Прежде всего необходимо дать определение понятию «количество информации».

Пусть имеется n возможных сообщений, и пусть вероятности появления их равны соответственно $p_1, p_2,..., p_n$ (где $\sum_{i=1}^{n} p_i = 1$). Мера количества информации, содержащейся в одном сообщении, некоторым образом характеризует свободу выбора того или иного сообщения или неопределенность появления некоторого сообщения. Если такая мера существует, скажем $H(p_1, p_2,..., p_n)$, то разумно требовать, чтобы эта величина обладала следующими свойствами:

(1) H должна быть непрерывной функцией p_i;

(2) если все p_i равны между собой, $p_i = 1/n$, $i = 1, 2,... n$, то H должна быть монотонно возрастающей функцией n (при равновероятных сообщениях количество информации, содержащейся в каждом из них, тем больше, чем больше число сообщений);

(3) если выбор разбивается на два последовательных выбора, то общая H должна быть взвешенной суммой частичных H.

Смысл последнего требования иллюстрируется рис. 1-6. Слева приведена схема выбора одного из четырех равновероятных сообщений ($p_1 = p_2 = p_3 = p_4 = {}^1/_4$). Справа тот же выбор разбивается на два последовательных выбора: сначала с вероятностями ${}^1/_4$, ${}^3/_4$ производится выбор между 1-м сообщением и объединением 2-го, 3-го и 4-го сообщений; если выбрано объединение 2-го, 3-го и 4-го сообщений, то далее с равными вероятностями ${}^1/_3$, ${}^1/_3$, ${}^1/_3$ выбирается одно из этих сообщений. В соответствии с условием (3) в этом частном случае должно выполняться соотношение

Рис. 1-6. Разбиение одного выбора на два последовательных выбора.

$$H({}^1/_4, {}^1/_4, {}^1/_4, {}^1/_4) = H({}^1/_4, {}^3/_4) + {}^3/_4\, H({}^1/_3, {}^1/_3, {}^1/_3)$$

коэффициент ${}^3/_4$ при $H({}^1/_3, {}^1/_3, {}^1/_3)$ стоит потому, что второй выбор появляется лишь с вероятностью ${}^3/_4$.

Можно доказать, что *единственная функция, удовлетворяющая трем предположениям, сделанным относительно H,* имеет вид

*) Мы следуем в дальнейшем изложении этого раздела ходу изложения К. Шеннона (см. сб. Шеннон К., Работы по теории информации и кибернетике, ИЛ, 1963). Никому, кажется, не удалось изложить эти вопросы лучше, чем это сделал сам автор теории информации.

$$H = -C \sum_{i=1}^{n} p_i \log p_i,$$

где C — постоянная положительная величина.

Выбор C, как и выбор основания логарифмов, эквивалентен выбору единицы измерения. В дальнейшем принимается $C = 1$.

Функцию $H(p_1, p_2, ..., p_n) = -\sum_{i=1}^{n} p_i \log p_i$ по аналогии с функциями того же вида, применяемыми в статистической физике, называют *энтропией* множества сообщений. Энтропия множества сообщений является мерой количества информации, содержащейся в каждом из них.

Если, например, передаваемые сообщения имеют вид n-разрядных двоичных чисел, то это еще не обязательно означает, что каждое из них содержит n двоичных разрядов информации. Такое количество информации содержалось бы в каждом сообщении при условии, что все 2" комбинаций равновероятны. Если, к примеру, допустимыми являются лишь некоторые комбинации, а вероятность передачи других комбинаций равна нулю, то каждое передаваемое сообщение содержит меньше информации. Таким образом, количество информации, содержащееся в каждом сообщении, зависит не столько от формы этого сообщения, сколько от того, каким способом производится выбор сообщений из множества возможных.

Однако и форма (способ кодирования) сообщений играет существенную роль. Разность между максимальным количеством информации, которое могло бы содержаться в каждом сообщении при выбранном способе кодирования, и тем количеством информации, которое действительно содержится в нем, называется *избыточностью* кода. Если, например, сообщения передаются в виде 7-разрядных двоичных кодов, но имеется всего 16 возможных сообщений (равновероятных между собой), то количество информации в каждом сообщении равно 4 двоичным разрядам $\log_2 16$), а избыточность кода составляет 3 двоичных разряда. Такое положение мы имели в примере, рассмотренном в 1.7.1 (см. таблицу 1-8 — код, позволяющий исправлять одиночные ошибки).

Если при передаче сообщений возможны искажения вследствие помех, то избыточность может быть использована для восстановления правильного сообщения.

Можно показать, что *любому конечному уровню помех соответствует определенная величина избыточности кода, являющаяся необходимой и достаточной для того, чтобы вероятность правильного приёма сообщений была как угодно близка к единице.* Это утверждение является одним из наиболее важных выводов теории информации.

Покажем прежде всего, каким способом можно подсчитать величину потерь информации, происходящих вследствие помех.

Пусть принято некоторое сообщение i. Если бы помех не было, то отсюда непременно вытекало бы, что переданное сообщение тоже есть i. При наличии помех может существовать несколько различных сообщений j, передача любого из которых могла бы привести к приему сообщения i. Обозначим через $p_i(j)$ условную вероятность того, что было передано сообщение j, если известно, что принятое сообщение есть i. В соответствии со сказанным выше, неопределенность того, какое сообщение было передано, при условии, что принято сообщение i, должна измеряться суммой

$$-\sum_{(i)} p_i(j) \log p_i(j).$$

Ее среднее значение

$$H_y(x) = -\sum_{(i)} p_i \sum_{(j)} p_i(j) \log p_i(j)$$

(где p_i — вероятность принять сообщение i) является мерой того, какова в среднем неопределенность переданного сообщения, если известно принятое сообщение. Эту величину называют *условной энтропией.* (В обозначении $H_y(x)$, как и в дальнейшем, буква x относится к передаваемым сообщениям, y — к принимаемым).

Условная энтропия переданных сообщений при известных принятых сообщениях — $H_y(x)$ — является мерой потери информации в результате воздействия помех. Таким образом, количество информации в каждом принимаемом сообщении равно

$$H(x) - H_y(x),$$

где $H(x)$ — энтропия передаваемых сообщений, т. е. количество информации, содержащейся в каждом передаваемом сообщении.

Рассмотрим следующий пример. Пусть каждое сообщение представляет собой 8-разрядное двоичное число, и пусть вероятность ошибки в каждом из разрядов p равна $^1/_8$. Каково максимальное количество информации, которое может содержаться в каждом принятом сообщении?

Поскольку вероятность ошибки в любом из 8 разрядов сообщения равна $^1/_8$, в среднем в каждом сообщении принимается неверно 1 двоичный разряд из восьми. Но это не означает, конечно, что количество информации в каждом принятом сообщении может быть равно 7 двоичным разрядам, так как заранее не известно, какой именно из восьми двоичных разрядов принят неверно. В соответствии со сказанным выше, максимально возможное количество информации подсчитывается следующим образом.

В переданном сообщении каждый двоичный разряд может содержать максимум 1 двоичный разряд

информации (если вероятности передачи 0 и 1 равны между собой). Тогда и вероятности приема 0 и 1 равны между собой ($p_0 = p_1 = {}^1/_2$). Но если принят 0, то вероятность того, что был передан 0, равна

$$p_0(0) = {}^7/_8,$$

а вероятность того, что была передана 1, равна

$$p_0(1) = {}^7/_8.$$

Аналогичным образом

$$p_1(1) = {}^7/_8,$$

$$p_1(0) = {}^1/_8.$$

Поэтому потеря информации при передаче каждого двоичного разряда (измеренная в двоичных разрядах) равна

$$H_y(x) = -p_0[p_0(0)\log_2 p_0(0) + p_0(1)\log_2 p_0(1)] -$$
$$- p_1[p_1(1)\log_2 p_1(1) + p_1(0)\log_2 p_1(0)] = -\frac{7}{8}\log_2\frac{7}{8} - \frac{1}{8}\log_2\frac{1}{8} \approx 0{,}54$$ двоичного разряда.

Потеря информации во всем сообщении равна 8·0,54= =4,32 двоичного разряда, а максимальное количество информации во всем принятом сообщении составляет 8— —4,32=3,68 двоичного разряда.

Можно показать, далее, что *вычисленная указанным образом потеря информации* $H_y(x)$ *равна такой величине избыточности кода, которая является необходимой и достаточной для того, чтобы вероятность правильного приема сообщения была как угодно близка к единице.* Таким образом, например, передавая 3,68 двоичного разряда информации при помощи 8-разрядного двоичного кода, возможно в принципе так составить код, что все возможные ошибки будут исправлены с вероятностью как угодно близкой к единице (при условии, что при передаче одного сообщения искажается в среднем один двоичный разряд); если избыточность кода будет меньше, то при данном уровне помех это невозможно.

Более низкому уровню помех соответствует меньшая потеря информации и, следовательно, меньшая необходимая и достаточная избыточность кода. Например, если бы вероятность ошибки в каком-либо разряде равнялась $p = {}^1/_9$, то потеря информации в каждом разряде была бы равна

$$H_y(x) = -\frac{8}{9}\log_2\frac{8}{9} - \frac{1}{9}\log_2\frac{1}{9} \approx \frac{1}{2}$$ двоичного разряда.

В 8-разрядном сообщении можно было бы иметь примерно 4 разряда информации (при избыточности в 4 двоичных разряда) и т. д.

Нетрудно видеть, что сформулированный здесь критерий существенно отличается от критериев, приведенных в 1.7.1 и 1.7.2.

С точки зрения критериев теории информации коды Хемминга весьма далеки от оптимальных. Из таблицы 1-7 видно, например, что 7-разрядный код Хемминга с исправлением одиночных ошибок может содержать 4 разряда информации, а его избыточность равна 3 двоичным разрядам ($m = 4$, $k = 3$, $m + k = 7$).

Именно такой величина избыточности должна была бы получиться, если бы мы предположили, что вероятность приема сообщения без ошибок равна $p^{(0)} = {}^1/_8$. вероятность приема сообщения с одиночной ошибкой в каком-либо i-м разряде ($i = 1, 2,..., 7$) тоже равна $p^i = {}^1/_8$, а случаи приема сообщений с ошибками более чем в одном разряде невероятны. Согласно сказанному выше потеря информации при этом была бы равна

$$H_y(x) = -\sum_{i=0}^{7} p^{(i)} \log_2 p^{(i)} = -\log_2 {}^1/_8,$$

т. е. 3 двоичным разрядам. Код Хемминга идеально соответствует такому предположению: он содержит ровно 3 избыточных разряда и с полной достоверностью исправляет все одиночные ошибки. Однако само предположение не выдерживает критики. Если случаи приема сообщения без ошибок и приема сообщения с одиночной ошибкой в каком-либо i-м разряде равновероятны, то это означает, что вообще вероятность ошибки при приеме каждого из двоичных разрядов p должна быть весьма велика[*]): из равенства $(1 — p)^7 = (1 — p)^6 p$ вытекает $p = {}^1/_2$. Но тогда, конечно, нельзя пренебрегать возможностью одновременных ошибок в двух, трех и более разрядах. При $p = {}^1/_2$ их суммарная вероятность равна ${}^{15}/_{16}$.

Коды Хемминга, исправляющие одиночные ошибки, в действительности применяются, конечно, для намного меньших уровней помех — когда двойными, тройными и т. д. ошибками можно на самом деле пренебречь. Но тогда и избыточность кода может быть много меньшей. Мы сильно завысили ее потому, что при построении кода несоразмерно преувеличили роль одиночных ошибок. В то же время, допустив такую большую избыточность, мы не получили никаких возможностей для исправления двойных, тройных и т. д. ошибок.

Необходимо учитывать, однако, следующее обстоятельство. Важный вывод теории информации о том, что

[*]) Ошибки в различных разрядах считаются независимыми.

избыточность кода, равная потере информации вследствие помех, является необходимой и достаточной для уверенного приема сообщений, носит характер теоремы существования. В его доказательстве (оно проводится во многих работах по теории информации), собственно построение кода, обладающего указанной избыточностью и позволяющего исправлять все возможные ошибки при данном уровне помех с вероятностью, как угодно близкой к единице, отсутствует. Коды Хемминга при этом имеют уже то важное преимущество, что они построены и известны.

Очень многие работы последних лет посвящены построению различных кодов, которые были бы возможно ближе к оптимальным (так называемые *циклические* и другие коды).

Не имея возможности подробно останавливаться на этих работах, мы хотели бы все же предостеречь читателя от попыток механического перенесения их результатов в теорию и тем более практику построения цифровых вычислительных машин. Переоценка роли выводов теории информации (в части защиты кодов от помех) для вычислительной техники является весьма частой ошибкой новичков.

Дело в том, что при всем внешнем сходстве ситуации, которая рассматривается в теории информации, и ситуации, с которой мы имеем дело в вычислительных машинах, положение здесь имеет и одно очень существенное различие.

В системе связи (см. рис. 1-5 на стр. 49) основным узким местом является канал связи. Именно в нем возникают те помехи, с которыми мы пытаемся бороться при построении помехозащищенных кодов, и именно пропускная способность канала ограничивает нас, когда мы ставим задачу получить код с минимально необходимой избыточностью. По сравнению с каналом связи то оборудование, которое находится на передающем и приемном концах системы, предполагается — по крайней мере в теории — бесконечно надежным и бесконечно дешевым. Поэтому критерии для оценки кодов исходят только из возможности искажения сообщений при их передаче и из того, насколько компактно кодируются сообщения, проходящие по каналу связи; насколько сложным должно быть оборудование для кодирования и декодирования сообщений на передающем и приемном концах, сколько времени потребуют эти операции, какова вероятность ошибки при их выполнении — все эти вопросы в оценке кодов игнорируются.

В вычислительной машине канал связи как таковой отсутствует. Те «помехи», с которыми нам приходится иметь дело, представляют собой в сущности сбои в работе (неправильные срабатывания) цифровых элементов, элементов запоминания, элементов формирования сигналов и логических элементов машины.

Если для устранения этих помех используются помехозащищенные коды, то нельзя не учитывать, что операции по кодированию и декодированию сообщений (чисел) и по обнаружению и исправлению ошибок будут выполняться такими же элементами схем — столь же надежными или столь же ненадежными, как и те элементы, которые осуществляют передачу чисел. Иногда, впрочем, некоторая разница в надежности есть: помехозащищенные коды используются часто для увеличения надежности запоминающих устройств, причем схемы кодирования и декодирования чисел строятся из более надежных цифровых и логических элементов; но различие в надежности здесь не так велико, как предполагается в теории информации.

Наконец, в технике связи задача обычно состоит в том, чтобы максимально загрузить канал связи. Кодирование сообщений на передающем конце системы и их декодирование на приемном конце могут выполняться одновременно многими людьми (или многими автоматическими устройствами), и время выполнения этих операций, как и количество оборудования, необходимое для их осуществления, несущественно. В вычислительной технике положение, очевидно, совсем иное.

Таким образом, оценка тех или иных кодов с точки зрения вычислительной техники непременно должна включать оценку *сложности* операций кодирования и декодирования. Задача здесь состоит не в том, чтобы получить код с минимальной избыточностью, а в том, чтобы минимизировать общий объем оборудования (включая и оборудование кодирования и декодирования чисел) или, возможно, произведение количества оборудования на время выполнения операций. При этом нужно достигнуть не столько максимальной вероятности исправления всех возможных ошибок при передаче чисел, сколько повышения общей надежности — с учетом того, что оборудование для исправления ошибок при передаче чисел само может явиться источником новых ошибок.

Численные методы таких оценок к настоящему времени не разработаны и представляют собой, вероятно, очень интересный и важный объект будущих исследований. Пока что в вычислительной технике часто применяются коды, избыточность которых намного больше, чем та потеря информации, на которую они рассчитаны. С точки зрения критериев теории информации эти коды весьма далеки от оптимальных; зато преимуществом их является то, что операции кодирования и декодирования сообщений осуществляются сравнительно просто. С одним из примеров такого кода мы познакомимся в 1.7.4. Более сложные коды — даже такие, как коды Хемминга для исправления одиночных ошибок — почти никогда не применяются и, как считают, особого эффекта не могут дать.

1.7.4. Ортотронный контроль*).

Пусть имеем m-разрядное двоичное число или, может быть, несколько чисел, содержащих всего m двоичных

*) По поводу происхождения термина «ортотронный» *(англ.* — orthotronic) в литературе имеются противоречивые сведения. Скорее всего это слово означает контроль по взаимно перпендикулярным (ортогональным) направлениям.

разрядов. Разобьем все имеющиеся разряды на s равных групп, по r двоичных разрядов в каждой группе $(m = rs)$. Наименьшая избыточность кода получится при $r \approx s \approx \sqrt{m}$. Запишем далее число (набор чисел) в виде прямоугольной таблицы из s строк и r столбцов — в соответствии с разбивкой имеющихся разрядов на группы. Добавим по одному контрольному разряду к каждой из s строк и к каждому из r столбцов. Цифры контрольных разрядов выбираются так, чтобы количество единиц в каждой строке и в каждом столбце было, скажем, четным.

Если при передаче числа исказится цифра в каком-нибудь разряде, то это будет обнаружено контролем по четности в соответствующей строке и в соответствующем столбце. Поскольку при этом будут известны и строка, и столбец, в которых произошла ошибка, ошибку можно будет исправить.

П р и м е р . Пусть имеем 30-разрядные двоичные числа, которые разбиваются на 6 групп, по 5 разрядов в каждой группе (m = 30, s = 6, r = 5). Количество контрольных разрядов при этом равно $s + r$ = 11. Например, двоичное число

11101 10101 10110 00011 10000 01110 должно быть записано в виде

	(контрольные разряды строк)
11101	0
10101	1
10110	1
00011	0
10000	1
01110	1
00011	

(контрольные разряды столбцов)

Если при передаче кода произойдет ошибка, например, в двенадцатом слева разряде (вместо «0» будет принято «1»), то принятый код будет иметь вид

11101	0
10101	1
←11110—	—1—
00011	0
10000	1
01110	1
00011	

Стрелками здесь перечеркнуты строка и столбец, в которых контроль по четности обнаруживает ошибку. На пересечении стрелок находится неверно принятый разряд (двенадцатый); замена принятой в нем цифры на противоположную дает правильный код.

Избыточность рассматриваемого кода больше, чем избыточность кода Хемминга, позволяющего исправлять одиночные ошибки: для кода Хемминга при m разрядах информации количество k контрольных разрядов должно быть не меньше $\log_2 (m+k+1)$ (см. 1.7.1), для рассматриваемого кода — не меньше, чем $2\sqrt{m}$. Зато и операции кодирования и исправления ошибок здесь проще.

Кроме того, необходимо учесть, что рассматриваемый код позволяет исправлять не только одиночные ошибки, но и ошибки в нескольких разрядах строки (при условии, что остальные строки приняты верно) или в нескольких разрядах столбца (если приняты верно остальные столбцы).

Ортотронный контроль был предложен для защиты информации на магнитных лентах. Строка соответствует ряду цифр, записываемых и считываемых одновременно несколькими магнитными головками, расположенными по ширине ленты; столбец соответствует ряду цифр, записываемых и считываемых последовательно одной головкой (вдоль длины ленты). Возможные ошибки связаны с местными неоднородностями покрытия ленты или с попаданием мелких пылинок между магнитной головкой и лентой. При этом наиболее вероятными являются ошибки в 1—2 соседних строках. Если по какой-либо из указанных причин неверно читаются маркерные сигналы строк (служебные отметки, обозначающие те моменты времени, когда должны восприниматься сигналы строки), то ошибочной оказывается целая строка или, весьма вероятно, 2 соседние строки. Для того чтобы иметь возможность исправлять ошибки в двух соседних строках, обычно предусматривают не один, а два контрольных разряда в каждом столбце: один для четных строк, другой для нечетных.

Не исключена возможность, однако, применения аналогичных методов и для защиты информации в других устройствах машины.

1.7.5. Защита двоичных кодов десятичных цифр.

Если десятичная цифра кодируется при помощи нескольких двоичных цифр, то для защиты двоичных кодов от помех можно применить обычные методы, изложенные выше. Например, при использовании 4-разрядного двоичного кода для представления десятичных цифр можно добавить 5-й — контрольный — разряд, осуществить контроль по четности и таким образом обнаруживать одиночные ошибки.

Однако для этого случая известны и другие, более выгодные методы построения помехозащищенных кодов.

В разделе 1.6 упоминался код «два из пяти» (табл. 1-6 на стр. 43, столбец «б»). Легко видеть, что этот код, содержащий также 5 двоичных разрядов на одну десятичную цифру, позволяет обнаруживать, кроме всех одиночных ошибок, 40% двойных и 4-кратных ошибок, все 3-кратные и 5-кратные ошибки. Признаком ошибки при этом будет являться изменение количества единиц в группе из пяти двоичных разрядов. О других полезных свойствах этого кода сказано в разделе 1.6.3.

В разделе 1.6.2 рассмотрен другой 5-разрядный код десятичных цифр, позволяющий обнаруживать одиночные ошибки — код «3a + 2». Возможность обнаружения одиночных ошибок вытекает из того, что расстояние между двумя ближайшими из используемых комбинаций равно двум единицам, т. е. что любые две допустимые комбинации различаются не менее чем в двух разрядах. Последнее легко проверить непосредственным сравнением изображений десятичных цифр в этом коде (см. стр. 42). Наряду с этим код «3a + 2» позволяет удобно выполнять арифметические операции над десятичными числами, о чем говорилось в свое время.

Чтобы имелась возможность не только обнаруживать, но и исправлять одиночные ошибки либо обнаруживать двойные ошибки, необходимо обеспечить расстояние между двумя ближайшими комбинациями минимум в три единицы. Как видно из таблицы 1-7 (стр. 45), для этого на каждые 4 двоичных разряда информации нужно иметь по меньшей мере 3 контрольных разряда, т. е. изображать каждую десятичную цифру с помощью 7 двоичных разрядов. Можно, однако, показать, что среди 7-разрядных двоичных кодов десятичных цифр, обеспечивающих минимальное расстояние в 3 единицы, нет таких кодов, которые в отношении выполнения арифметических операций были бы аналогичны коду «3a + 2». Зато существует один 8-разрядный код такого типа — код «27a + 6»[*]).

По существу уже в изображении десятичной цифры четырьмя двоичными разрядами имеется некоторая избыточность, которую можно попытаться использовать Для защиты кода. В одной из работ в этом направлении[**]) поставлены следующие цели построения такого кода:

— максимальная вероятность обнаружения одиночных ошибок;
— минимальная величина наибольшего возможного отклонения в значении десятичной цифры при необнаруживаемых одиночных ошибках;
— минимальная величина среднего отклонения в значении десятичной цифры при необнаруживаемых одиночных ошибках.

Если мы пользуемся, например, кодом «8, 4, 2, 1», то все эти показатели оказываются очень плохими. Вероятность обнаружения одиночных ошибок оказывается равной всего 25%. Скажем, при передаче десятичной цифры «0» (0000) ни одна из четырех возможных одиночных ошибок не обнаруживается (вместо 0000 принимаются комбинации 1000, 0100, 0010 или 0001, каждая из которых является допустимой); при передаче кода «9» (1001) одиночные ошибки в первом или четвертом разрядах не обнаруживаются (потому что при этом получаются допустимые комбинации 0001 или 1000), а одиночные ошибки во втором или третьем разрядах можно обнаружить (так как комбинации 1101 и 1011 недопустимы) и т. д. Наибольшее отклонение в значении десятичной цифры при необнаруживаемых одиночных ошибках может быть равно восьми (например, если при передаче цифры «0» [0000] произойдет ошибка в старшем двоичном разряде, то будет принята цифра «8» [1000]). Среднее значение таких отклонений, как нетрудно подсчитать, равно трем.

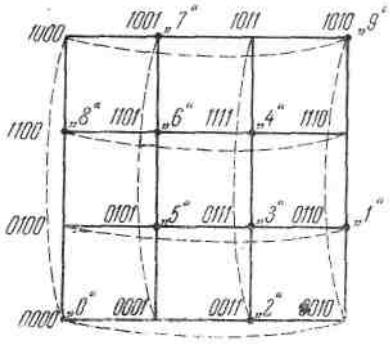


Рис 1-7. Графическое представление оптимального (с точки зрения защиты от помех) 4-разрядного двоичного кода десятичных цифр.

Можно показать, что наилучшим с точки зрения указанных критериев является код, приведенный в столбце «г» таблицы 1-6 (см. стр. 43). На рис. 1-7 приведена графическая интерпретация этого кода, причем условная развертка 4-мерного куба выполнена в соответствии с рис. 1-4 на стр. 48. Из рисунка видно, что допустимые комбинации выбраны по возможности так, чтобы расстояние между ними было не меньше двух единиц (если бы использовалось не 10, а 8 комбинаций, то можно было бы не задействовать комбинации 0111 и 1101, соответствующие цифрам «3» и «6», и тогда это условие выполнялось бы для всех случаев). В некоторых местах, однако, приходится все-таки использовать комбинации, находящиеся друг от друга на расстоянии в одну единицу; следовательно, не все одиночные ошибки будут

[*]) D i a m o n d J. M., Checking Codes for Digital Computers. Proc. IRE, 1955, 43, N 4 (стр. 487—488).

[**]) K a u t z W . H., Optimized Data Encoding for Digital Computers, Conv. Rec. IRE, 1954, Part. 4 — Electronic Computer and Information Theory (стр. 47—57).

обнаруживаться. Но в этих случаях близким комбинациям поставлены в соответствие близкие по значению десятичные цифры.

Легко проверить, что для рассматриваемого кода вероятность обнаружения одиночной ошибки равна 60%, максимальное отклонение в значении десятичной цифры при необнаруживаемых одиночных ошибках равно двум, а среднее отклонение — полутора.

1.7.6. Контроль по модулю.

Контрольные коды Хемминга, ортотронный контроль и другие помехозащищенные коды, рассматривавшиеся ранее, могли использоваться только для повышения надежности передач чисел, но не для контроля выполнения операций над ними. Если, к примеру, для одного и для другого числа известны контрольные разряды в каком-либо из кодов Хемминга, то по ним никак нельзя образовать контрольные разряды для суммы, или разности, или произведения данных чисел; выполнив арифметическую операцию, приходится заново, независимым путем, формировать контрольные разряды для результата, которые далее позволят контролировать передачу этого результата в память, хранение его в памяти, обратный прием в арифметическое устройство и т. д.; само же выполнение арифметической операции контролем не охватывается.

Контроль по модулю более универсален, поскольку он может быть использован и для проверки правильности передач чисел, и для проверки правильности выполнения операций.

Идея состоит в том, что в качестве контрольной группы к каждому числу присоединяется остаток от деления этого числа на какое-нибудь заранее заданное целое число *r*. При этом сами исходные числа рассматриваются как целые. Например, в машине с фиксированной запятой контрольные цепи оперируют со всеми числами так, как будто запятая в этих числах находится справа от младшего разряда. Это не мешает основным цепям машины оперировать со всеми числами как с дробными.

Для контроля передач чисел вместе с каждым числом передается и его остаток от деления на *r.* Для принятого числа заново вычисляется остаток от деления на *r* и сравнивается с принятым остатком.

При контроле операций мы пользуемся тем, что остаток от деления на *r* суммы (разности, произведения) двух чисел должен быть равен сумме (разности, произведению) по модулю *r* остатков от деления на *r* исходных чисел (см. 1.3.4). Каждая операция выполняется одновременно над исходными числами и над их остатками от деления на *r*. Затем для результата основной операции вычисляется остаток от деления на *r*, который сверяется с результатом операции над остатками исходных чисел.

Конечно, возможности такого контроля операций не следует преувеличивать: алгоритмы контроля сложения, вычитания и умножения сильно усложняются при наличии округления, при использовании дополнительных кодов и т. д.; алгоритмы для контроля по модулю логических операций, деления и других операций вообще очень сложны и не обеспечивают достаточной достоверности.

Вообще достоверность контроля передач и арифметических операций тем выше, чем больше число г. Но чем больше *r*, тем сложнее и контрольные цепи. Поэтому в тех случаях, когда пользуются контролем по модулю, приходится выбирать некоторое компромиссное значение для величины *r*. Для десятичной системы, например, выбирают *r* равным 3, или 9, или 11, для двоичной — 3, или 5, или 7, или 15.

При выборе величины *r* желательно соблюдать следующее соотношение:

$$r = \frac{n^k - 1}{l},$$

где *n* — основание системы счисления, *k, l*— произвольные целые числа, *k* по возможности невелико. Иначе говоря, $n^k \equiv 1 \pmod r$. Например, для десятичной системы

$$3 = \frac{10^1 - 1}{3} \qquad \text{(т. е. } 10 \equiv 1 \pmod 3\text{)},$$

$$9 = \frac{10^1 - 1}{1} 9 \qquad \text{(т. е. } 10 \equiv 1 \pmod 9\text{)},$$

$$11 = \frac{10^2 - 1}{9} \qquad \text{(т.е. } 100 \equiv 1 \pmod{11}\text{)}.$$

Для двоичной системы

$$3 = \frac{2^2 - 1}{1} \qquad \text{(т. е. } 4 \equiv 1 \pmod 3\text{)},$$

$$5 = \frac{2^4 - 1}{3} \qquad \text{(т. е. } 16 \equiv 1 \pmod 5\text{)},$$

$$7 = \frac{2^3 - 1}{1} \qquad \text{(т. е. } 8 \equiv 1 \text{ (mod 7))},$$

$$15 = \frac{2^4 - 1}{1} \qquad \text{(т. е. } 16 \equiv 1 \text{ (mod 15))}.$$

Выполнение этого соотношения позволяет получить простую схему для вычисления остатка от деления произвольного числа, представленного позиционной системой с естественными весами разрядов, на *r*. Например, для вычисления остатка от деления любого десятичного числа на 3 или на 9 достаточно сложить между собой по модулю 3 (или соответственно 9) остатки от деления каждой десятичной цифры числа на 3 (на 9); при этом остаток от деления десятичной цифры на 9 равен, естественно, ей самой. Остаток от деления любого десятичного числа на 11 находится путем разбиения этого числа на пары разрядов *(k* = 2), вычисления остатков от деления на 11 полученных таким образом двузначных чисел и суммирования затем этих остатков по модулю 11. (Например, для числа 7 839 756 840

$$40 \equiv 7 \text{ (mod 11)},$$

$$68 \equiv 2 \text{ (mod 11)},$$

$$75 \equiv 9 \text{ (mod 11)},$$

$$39 \equiv 6 \text{ (mod 11)},$$

$$78 \equiv 1 \text{ (mod 11)},$$

$$7 + 2 + 9 + 6 + 1 = 25 \equiv 3 \text{ (mod 11)}.$$

Остаток от деления 7 839 756 840 на 11 равен 3.)

Аналогичным образом остаток от деления любого двоичного числа на 3 получится разбивкой этого числа на пары двоичных разрядов и суммированием полученных двузначных чисел по модулю 3 (остаток отделения двузначного двоичного числа на 3 равен ему самому); остаток отделения двоичного числа на 7 (на 15) получится разбиением этого числа на тройки (на четверки) двоичных разрядов и суммированием полученных трехзначных (четырехзначных) двоичных чисел по модулю 7 (соответственно 15). Для вычисления остатка от деления двоичного числа на 5, нужно распределить разряды исходного числа на четверки, найти для каждого из полученных таким образом четырехзначных двоичных чисел остаток от деления на 5 и затем сложить эти остатки по модулю 5.

Иначе эти правила формулируются следующим образом:

остаток от деления на 3 любого двоичного числа (представленного позиционной системой с естественными весами разрядов и с цифрами 0,1) находится суммированием по модулю 3 его цифр с весами

..., 2, 1, 2, 1, 2, 1;

остаток от деления на 5 — суммированием по модулю 5 цифр исходного числа с весами

..., 3, 4, 2, 1, 3, 4, 2, 1;

остаток от деления на 7 — суммированием по модулю 7 цифр исходного числа с весами

..., 4, 2, 1, 4, 2, 1, 4, 2, 1;

остаток от деления на 15 — суммированием по модулю 15 цифр исходного числа с весами

.... 8, 4, 2, 1, 8, 4, 2, 1.

Исходя из этих правил, нетрудно подсчитать и возможности указанных кодов в отношении обнаружения ошибок. Одиночная ошибка (ошибка при передаче двоичного числа в одном его разряде или ошибка в одном двоичном разряде результата арифметической операции) обнаруживается, очевидно, в любом случае.

Если полагать, что ошибки в отдельных разрядах равновероятны, независимы между собой и не зависят от информации, содержащейся в этих разрядах, то использование для контроля двоичных чисел остатков от деления на 3 позволяет обнаруживать двойные ошибки с вероятностью $^1/_2$: двойные ошибки, состоящие в одновременном появлении или одновременном пропадании двух единиц, обнаруживаются в тех случаях, когда оба разряда, в которых происходят ошибки, имеют одинаковые веса (либо 1, либо 2), и не обнаруживаются, когда вес одного из этих разрядов 1, а другого 2 (так как 2+1 ≡ 0 (mod 3)); двойные ошибки, состоящие в появлении лишней единицы в одном из разрядов и исчезновении единицы в другом, обнаруживаются в тех случаях, когда веса этих разрядов различны, и не обнаруживаются — когда одинаковы.

Аналогичные рассуждения для тройной ошибки сведены в таблицу 1-9, где знак «+» на пересечении строки и столбца показывает, что ошибка данного вида обнаруживается при контроле, а знак «—» — что не обнаруживается.

Из таблицы видно, что вероятность обнаружения тройной ошибки составляет $^3/_4$.

Аналогичным образом можно подсчитать вероятности обнаружения четверных и других ошибок при использовании для контроля остатков от деления на 3, а также вероятности обнаружения ошибок различных видов при использовании остатков от деления на 5, на 7, на 15 и т. д.

Таблица 1-9

Подсчет вероятности обнаружения тройных ошибок при использовании для контроля двоичных чисел остатков от деления на 3

Веса разрядов, в которых произошли ошибки, применяемые при подсчете остатка от деления данного числа на 3	Характер ошибки							
	+ + +	− + +	+ − +	− − +	+ + −	− + −	+ − −	− − −
1, 1, 1	−	+	+	+	+	+	+	−
1, 1, 2	+	−	+	+	+	+	−	+
1, 2, 1	+	+	−	+	+	−	+	+
1, 2, 2	+	+	+	−	−	+	+	+
2, 1, 1	+	+	+	−	−	+	+	+
2, 1, 2	+	+	−	+	+	−	+	+
2, 2, 1	+	−	+	+	+	+	−	+
2, 2, 2	−	+	+	+	+	+	+	−

1.8. Рефлексный код

1.8.1. Постановка задачи.

Рефлексный код возник из задачи преобразования данных из непрерывной формы **в** дискретную.

Представим себе, например, что в качестве исходной в вычислительную машину должна быть введена величина угла поворота некоторого вала. Эта величина должна быть выражена, для примера, по двоичной системе, в долях от полного оборота вала.

Задача эта может быть решена так, как показано на рис. 1-8. С валом, угол поворота которого должен быть измерен, жестко соединен прозрачный диск. По одну сторону диска размещен источник света, по другую сторону—фотоэлементы, закрепленные неподвижно вдоль радиуса диска. Количество фотоэлементов равно количеству двоичных разрядов, которое необходимо получить в цифровом представлении измеренной величины (на рисунке их три).

На диске прочерчены концентрические кольца — каждое против своего фотоэлемента. Первое кольцо разделено пополам: половина его зачернена, половина оставлена прозрачной. Второе кольцо разделено на 4 равных сектора, третье — на 8. Если бы количество разрядов превышало 3, то четвертое кольцо делилось бы на 16 секторов, пятое — на 32 и т. д. Деление всех колец на секторы начинается от одного и того же радиуса, который условно назван нулевым. В каждом кольце зачерненные и прозрачные секторы чередуются через один.

Напряжение на любом из фотоэлементов зависит от того, какой сектор находится против него — зачерненный или прозрачный. Можно условиться, что напряжение на фотоэлементе, против которого находится зачерненный участок диска, соответствует двоичной цифре «0», а напряжение на фотоэлементе, против которого находится прозрачный участок,— цифре «1».

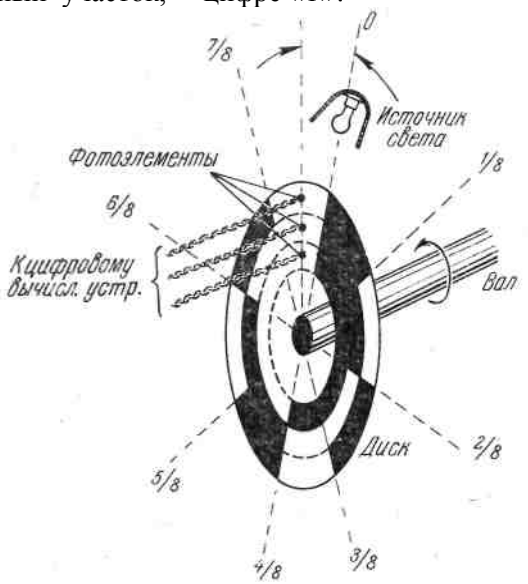


Рис. 1-8. Принцип построения преобразователя вал-цифра.

За нулевое положение вала принимается то положение, когда против фотоэлементов находится нулевой радиус диска. Если вал повернется от этого положения менее чем на $^1/_8$ оборота против часовой стрелки, то против всех фотоэлементов будут находиться зачерненные участки диска, и напряжения на них составят комбинацию .000. Если диск повернулся более чем на $^1/_8$ и менее чем на $^2/_8$ оборота, то освещен один 3-й фотоэлемент, а 1-й и 2-й затемнены; напряжение на фотоэлементах составляет комбинацию .001 и т. д. В положении, которое изображено на рисунке, диск совершил более $^7/_8$ оборота, все фотоэлементы освещены, и напряжения на них образуют комбинацию. 111. Таким образом, комбинация напряжений на фотоэлементах обычным двоичным кодом изображает угол поворота вала (в нашем случае — с точностью до $^1/_8$ оборота).

При этом, однако, имеется одна весьма существенная трудность. Представим себе, например, что диск повернут от нулевого состояния ровно на четверть оборота, так что против фотоэлементов находится радиус, который на рис. 1-8 имеет обозначение «$^2/_8$». При этом 1-й фотоэлемент затемнен, а 2-й и 3-й фотоэлементы освещены наполовину. Напряжения на них имеют некоторое промежуточное значение, однако последующие схемы

могут воспринимать их только двояко — либо как «0», либо как «1». Какая именно цифра будет воспринята от данного фотоэлемента, когда он освещен наполовину, зависит от многих случайных причин: от характеристики фотоэлемента, от точности его установки в пространстве, от настройки и характеристик последующих схем. Поэтому, когда диск находится в рассматриваемом промежуточном положении, напряжения на выходах фотоэлементов могут случайно образовать и комбинацию .000, и комбинацию .001, и комбинацию .010, и комбинацию .011. Первая соответствует углу поворота 0, вторая — $^1/_8$ оборота, третья — $^1/_4$ поворота, четвертая — $^3/_4$. Следовательно, ошибка в измерении угла поворота здесь может быть уже не $^1/_8$ оборота, а $^1/_4$. Еще больше ошибка может быть, когда против фотоэлементов находится в точности радиус «0» или радиус «$^4/_8$»; напряжения на выходах фотоэлементов могут при этом случайно образовать л ю б у ю из 8 возможных комбинаций, а ошибка может достигать $^1/_2$ оборота.

Для того чтобы обойти это затруднение, в преобразователях подобного типа используют вместо обычного двоичного кода специально построенные коды. Условие, которое должно быть соблюдено, состоит в том, что любым двум соседним значениям угла поворота должны соответствовать кодовые комбинации, отличающиеся одна от другой не более чем в одном двоичном разряде. Пользуясь представлениями раздела 1.7.2 настоящего параграфа, можно сказать, что расстояние между двумя комбинациями, соответствующими соседним значениям угла поворота, должно быть всегда равно одной единице*).

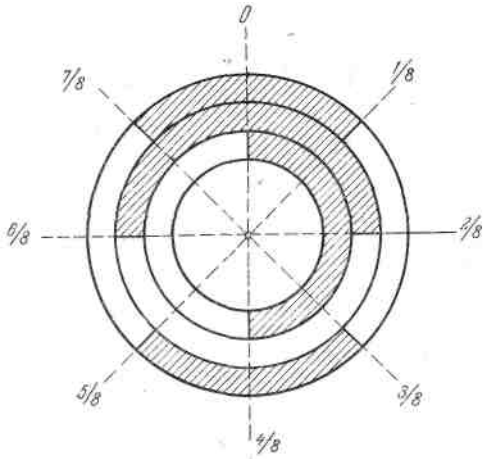


Рис. 1-9. Диск преобразователя вал-цифра при использовании 3-разрядного рефлексного кода (код столбца «а» таблицы 1-10).

В таблице 1-10 представлено 12 различных 3-разрядных кодов, удовлетворяющих указанному условию, а на рис. 1-9 показано, как должен быть расчерчен диск, если используется, например, код столбца «а» таблицы 1-10.

Таблица 1-10

3-разрядные двоичные коды, в которых две комбинации, соответствующие соседним значениям изображаемой величины, различаются в одном двоичном разряде

Значения угла поворота (в долях от полного оборота)	Двоичные изображения					
	а	б	в	г	д	е
0	000	000	000	000	000	000
$^1/_8$	001	001	001	001	010	010
$^2/_8$	011	101	101	011	011	110
$^3/_8$	010	111	100	111	001	111
$^4/_8$	110	011	110	101	101	011
$^5/_8$	111	010	111	100	111	001
$^6/_8$	101	110	011	110	110	101
$^7/_8$	100	100	010	010	100	100
	а'	б'	в'	г'	д'	е'
0	000	000	000	000	000	000
$^1/_8$	100	100	010	010	100	100
$^2/_8$	101	110	011	110	110	101
$^3/_8$	111	010	111	100	111	001
$^4/_8$	110	011	110	101	101	011
$^5/_8$	010	111	100	111	001	111
$^6/_8$	011	101	101	011	011	110
$^7/_8$	001	001	001	001	010	010

*)Другая возможность состоит в использовании так называемого V-кода. Она здесь не рассматривается.

Рассматривая рисунок, легко видеть, что любой радиус диска может оказаться границей между зачерненным и прозрачным секторами не более чем в одном кольце. С таким расчетом, собственно, и строился код. Поэтому при любом положении диска полуосвещенным может оказаться не более чем один какой-либо фотоэлемент. Как бы ни было воспринято напряжение на его выходе, ошибка не превысит точности измерений. Например, если против фотоэлементов находится радиус, обозначенный «$^2/_8$», то 1-й фотоэлемент на внутреннем кольце полностью затемнен, 3-й (на внешнем кольце) полностью освещен, а 2-й фотоэлемент (на среднем кольце) находится в промежуточном состоянии. Комбинация напряжений на фотоэлементах может быть воспринята либо как 001, либо как 011. Первая из них соответствует углу поворота в $^1/_8$ полного оборота, вторая — в $^2/_8$; и то и другое близко к истине.

Заметим, что и на радиусе «0» лишь для одного кольца проходит граница между зачерненным и прозрачным секторами. Если фотоэлементы находятся в точности против этого радиуса, то напряжения с их выходов могут восприниматься либо как 100 (т. е. $^7/_8$), либо как 000 (т. е. 0). Поскольку измерение идет по модулю 1, то с точностью до $^1/_8$ и то и другое является правильным результатом.

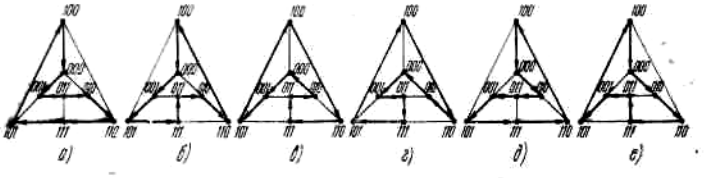


Рис. 1-10. Графическое построение кодов таблицы 1-10.

Рис. 1-10 иллюстрирует способ, которым были построены коды, приведенные в таблице 1-10. Для построения использована сетка из 8 узлов, которая может служить графическим изображением 3-разрядного двоичного кода (см. раздел 1.7.2, рис. 1-3 на стр. 48). Задача построения 3-разрядного двоичного кода, в котором 8 возможных комбинаций расположены так, что две соседние комбинации отличаются не более чем в одном двоичном разряде, графически интерпретируется следующим образом. Начиная от какого-либо узла сетки (например, 000) и двигаясь только вдоль линий сетки, нужно пройти по одному разу через все узлы и вернуться к исходной точке.

На рис. 1-10, *а* — *е* показаны 6 различных путей, которыми это можно сделать. Если не принимать во внимание направление движения, то легко убедиться, что никакие другие пути невозможны. Шесть соответствующих кодов записаны в столбцах «а» — «е» таблицы 1-10 (буквы в заголовках столбцов таблицы и на рисунках находятся в соответствии). Коды в столбцах «а'» — «е'» соответствуют тем же рисункам, но обратному порядку обхода узлов.

Из каждого кода, приведенного в таблице, круговой перестановкой символов можно получить еще 7 кодов с аналогичными свойствами. Таким образом, имеется всего 12 × 8 = 96 различных 3-разрядных кодов, обладающих тем свойством, что комбинации, соответствующие двум соседним значениям измеряемой величины, отличаются одна от другой в одном двоичном разряде. Еще больше может быть различных вариантов 4-разрядных, 5-разрядных и т. д. кодов. В разделе 1.6.3 настоящего параграфа приведен один из множества д е с я т и ч н ы х кодов с аналогичными свойствами, который удобно использовать в том случае когда угол поворота вала должен быть измерен с точностью до одной десятой от полного оборота (см. столбец «д» таблицы 1-6 на стр. 43).

Разумеется, различия, имеющиеся в этом множестве кодов, не все одинаково принципиальны. Например, коды, которые можно получить из одного какого-либо кода круговой перестановкой символов, соответствуют все одному и тому же чертежу диска; разница состоит лишь в том, какой радиус считается нулевым. Для пар кодов таблицы 1-10, полученных различным направлением обхода узлов сетки («а» и «а'», «б» и «б'» и т. д.), чертеж диска тоже один и тот же, но диск нужно поставить обратной стороной. Наконец, например, код «в'» таблицы 1-10 отличается от кода «г» только тем, что в нем переставлены средняя и правая цифры.

Однако если бы мы поставили задачу отыскать такой код, который, с одной стороны, обеспечивал бы максимальную точность преобразования данных из непрерывной формы в дискретную, а с другой стороны, давал бы возможность легко выполнять арифметические операции или, например, легко переходить к обычной двоичной записи чисел, то нам нужно было бы рассмотреть все указанное множество кодов. Такая задача пока не решена. В устройствах преобразования данных из непрерывной формы в дискретную используется один из возможных кодов, подходящих для этой цели,— так называемый рефлексный код (или «код Грэя»). 3-разрядный рефлексный код приведен в столбце «а» таблицы 1-10; общий случай построения рефлексного кода рассматривается ниже.

1.8.2. Рефлексный код.

Рефлексный код в общем случае строится по следующему правилу. Пусть имеем обычное двоичное представление числа (в позиционной системе, с естественными весами разрядов и цифрами «0», «1»). Если цифра некоторого разряда в таком представлении числа совпадает с цифрой соседнего старшего разряда, то цифра соответствующего разряда в рефлексном коде есть «0», в противном случае — «1». Что касается самого старшего разряда, то предполагается, что в обычном двоичном представлении числа слева от самого старшего разряда всегда находится 0; поэтому в старшем разряде рефлексного кода цифра всегда такая же, как в старшем разряде

обычного двоичного представления числа.

Например, если обычное двоичное представление числа есть 10011101011, то в рефлексном коде то же число записывается в виде 11010011110. Здесь цифра старшего разряда — такая же, как в обычном двоичном представлении («1»). Во втором слева разряде имеем «1», потому что в обычной записи цифра второго слева разряда не совпадает с цифрой первого разряда («0» и «1»); в третьем слева разряде имеем «0» потому, что в обычной записи цифры третьего и второго разрядов совпадают («0»и «0»), и т. д.

Легко доказать, что в коде, построенном по этому правилу, двум соседним по величине числам соответствуют комбинации, различающиеся в одном двоичном разряде.

Действительно. Любое число в обычной двоичной записи в самом общем случае имеет вид

$$...\underbrace{0111...111}_{k\text{ разрядов}}.$$

Мы полагаем здесь, что несколько старших разрядов могут содержать вообще произвольные цифры, затем в (k +1)-м (считая справа) разряде имеется цифра «0», а k младших разрядов содержат цифры «1»; в частности, может оказаться, что $k = 0$ (т. е. что в первом же справа разряде содержится «0»). В другом крайнем случае k равно количеству разрядов числа. Соответствующую запись в рефлексном коде получим в виде

$$... * \underbrace{100...000}_{k-1\text{ разрядов}},$$

где многоточие слева соответствует старшим разрядам, * — цифра $(k + 1)$-го разряда (она зависит от цифры (k + 2)-го слева разряда в обычной записи: если в (k + 2)-м разряде имелся «0», то *—это «0», если «1» —то «1»), последующие k цифр образованы в соответствии с указанным выше правилом.

Перейдем теперь от исходного числа к ближайшему большему. Воспользовавшись правилом прямого счета (раздел 1.3.2), получим обычную двоичную запись ближайшего большего числа в виде

$$....\underbrace{1000...000}_{k\text{ разрядов}},$$

где старшие разряды (обозначенные многоточием слева) — те же, что в исходном числе. Отсюда, следуя обычному правилу, запись этого числа в рефлексном коде получим в виде

$$.... \overset{-}{*} \underbrace{100...000}_{k-1\text{ разрядов}},$$

где многоточием слева обозначены старшие разряды — такие же, как в изображении рефлексным кодом исходного числа, * — цифра (k + 1)-го справа разряда, противоположная цифре этого разряда в изображении исходного числа, k младших цифр — снова такие же, как в изображении рефлексным кодом исходного числа. Таким образом, при переходе от любого числа к. ближайшему большему в его изображении рефлексным кодом всегда изменяется одна и только одна двоичная цифра.

Само правило построения рефлексного кода дает простой способ перевода чисел из рефлексного кода в обычную двоичную запись: старшая цифра в обычной двоичной записи числа — та же, что и старшая цифра в рефлексном коде; вторая (слева) цифра в обычной записи числа совпадает с первой, если вторая цифра в рефлексном коде есть «0», и противоположна первой цифре, если «1», и т. д. Например, если рефлексный код числа имеет вид 00110100010, то его обычная двоичная запись есть 00100111100. Каждая цифра в обычной двоичной записи числа либо совпадает с предыдущей (старшей) цифрой, либо противоположна ей в зависимости от того, является ли соответствующая цифра рефлексного кода нулем или единицей.

По указанному способу сравнительно легко построить схему, которая выполняла бы автоматически перевод чисел из рефлексного кода в обычную двоичную запись. Существенное неудобство состоит только в том, что операция перевода начинается от старших разрядов числа, а младший разряд обычной двоичной записи появляется в последнюю очередь. В то же время выполнение арифметических действий большей частью должно начинаться от младших разрядов чисел (например, выполнение сложения, вычитания и других операций).

Перевод чисел из рефлексного кода в обычную двоичную форму можно попытаться начинать и от младших разрядов. Для этого следовало бы поступить следующим образом. Младшей цифрой обычной записи числа сначала зададимся произвольно; вторая (справа) цифра либо совпадает с младшей цифрой, если младшая цифра в рефлексном коде есть «0», либо противоположна ей, если в рефлексном коде в младшем разряде имеем «1», и т. д. Получив таким образом все цифры обычной записи числа, нужно проверить, совпадает ли ее старшая цифра со старшей цифрой рефлексного кода; если совпадение получилось, то все цифры обычной записи числа найдены верно, в противном случае все найденные цифры нужно изменить на противоположные. Следовательно, и здесь окончательное значение младшей цифры обычной двоичной записи числа можно получить не раньше, чем будет выполнено преобразование по всем разрядам. Однако с технической точки зрения такое решение может иметь определенные преимущества.

Младшую цифру обычной двоичной записи числа можно было бы заранее точно определить, если бы заранее было известно, четно ли количество единиц в рефлексном коде числа. Ясно, что если в рефлексном коде числа

содержится четное количество единиц, то младшая цифра в обычной записи числа есть «0»; если в рефлексном коде количество единиц нечетно, то младшая цифра в обычной записи числа есть «1». Например, преобразование числа из рефлексного кода в обычную форму удобно было бы начинать от младших разрядов, если бы вместе с рефлексным кодом поступал разряд контроля по четности. Цифра этого разряда могла бы служить прямо младшим разрядом обычной записи числа; при этом совпадение старшего разряда, полученного в конце операции преобразования, со старшим разрядом рефлексного кода являлось бы признаком отсутствия ошибок при передаче кода.

2. Сумматоры и другие схемы для выполнения элементарных операций

Процесс выполнения любых арифметических и логических операций в арифметическом устройстве слагается из выполнения более простых *элементарных операций.* К числу последних относятся, например, сдвиг числа, обращение кода числа, передачи чисел и др.; наиболее сложной и, может быть, наиболее важной из элементарных операций является суммирование.

Каждая из элементарных операций осуществляется, как правило, своей специальной схемой; в частности, схема для выполнения суммирования называется сумматором. Иногда, впрочем, схемы для выполнения двух или нескольких элементарных операций могут частично состоять из общих для одной и для другой схемы элементов (особенно если эти операции не предполагается выполнять одновременно).

2.1. Сумматоры. Основные понятия

Выделяя *суммирование* в качестве элементарной операции, мы понимаем под этим термином простое *арифметическое сложение двух чисел, представленных одинаковой системой счисления, позиционным способом с естественными весами разрядов, с одинаковым количеством разрядов в каждом из чисел и с запятой, фиксированной в одном и том же месте.*

Полная операция сложения, выполняемая арифметическим устройством как самостоятельное законченное действие, в общем случае состоит из нескольких элементарных операций, среди которых имеется и суммирование. Усложнение по сравнению с простым суммированием получается за счет того, что числа могут быть различных знаков, что представлены они, возможно, в системе с плавающей запятой и по другим подобным причинам. Методы выполнения полной операции сложения рассматриваются в разделе 3; здесь же речь пойдет о выполнении только элементарной операции суммирования и о *сумматорах* — устройствах, предназначенных для этой цели.

2.1.1. Принцип построения сумматоров.

Пусть имеем два m-разрядных числа, каждое из которых представлено системой счисления с основанием n. Если n-ичные цифры изображаются с помощью минимального количества двоичных знаков, то каждое из чисел представляет собой набор $m\lambda(n)$ двоичных переменных, где $\lambda(n)$ — ближайшее не меньшее целое число к величине $\log_2 n$ (см. 1.2.2.). Например, 8-разрядное десятичное число, в котором каждая десятичная цифра изображается четырьмя двоичными цифрами, представляет собой набор из 32 двоичных переменных

Сумма двух таких чисел содержит на один разряд больше, чем исходные числа. При этом цифра дополнительного разряда — слева от старшего разряда слагаемых — может получиться вообще нулем, плюс единицей или минус единицей (если среди цифр n-ичной системы счисления имеются и положительные, и отрицательные). Поэтому для представления цифры дополнительного разряда суммы может потребоваться 2 двоичных разряда, а всего сумма будет представлена $m\lambda(n) + 2$ двоичными разрядами.

Каждую из цифр суммы можно вообще рассматривать как функцию от всех цифр слагаемых. Правда, в простом суммировании младшие разряды суммы не зависят от старших цифр слагаемых; однако для выполнения полной операции сложения иногда приходится применять так называемое «суммирование с круговым переносом» (см. 3.1.2), и тогда этого упрощения нет.

Таким образом, сумматор представляет собой схему, которая по определенным правилам формирует $m\lambda(n) + 2$ переключательных функций от $2m\lambda(n)$ двоичных переменных. Например, сумматор для 8-разрядных десятичных чисел получает входную информацию в виде 64 двоичных переменных ($2m\lambda(n) = 2 \cdot 8 \cdot 4$) и формирует 33 переключательные функции, каждая из которых вообще зависит от всех входных переменных. (Сумма содержит 33, а не 34 двоичных разряда, потому что в десятичной системе обычно все цифры неотрицательны — 0, 1, 2,..., 9 —и, следовательно, старший разряд суммы может содержать цифру 0 или 1, но не —1.)

Задача об оптимальном построении сумматора, т. е. о таком построении, которое обеспечивало бы наибольшее возможное быстродействие при минимальном количестве оборудования, не решена ни в общем виде, ни для какого-нибудь конкретного случая (скажем, для какой-нибудь определенной системы счисления). Как правило, полные сумматоры строят из так называемых одноразрядных сумматоров.

Одноразрядный сумматор — это схема, на входы которой поступают цифры о д н о г о разряда слагаемых (b и c) и цифра переноса из соседнего младшего разряда (e) и которая вырабатывает цифру суммы данного разряда *(B)* и перенос в следующий старший разряд (E). Выходной сигнал переноса (E), образующийся при сложении цифр данного разряда, является входным сигналом переноса *(e)* для сложения цифр следующего разряда.

Таким образом, принцип построения сумматоров в общем повторяет ту идею, которой мы пользуемся при выполнении сложения вручную. Не известно, однако, насколько такое решение близко к оптимальному с точки зрения количества необходимого оборудования и скоростей.

Разновидность элементарной операции суммирования, о которой мы упоминали выше,— «суммирование с круговым переносом» — отличается тем, что сигнал переноса с выхода самого старшего разряда сумматора является входным сигналом переноса для самого младшего разряда. Смысл введения такой разновидности будет выяснен в разделе 3. Здесь, однако, рассматривая различные построения сумматоров, мы будем иметь в виду, что в некоторых случаях может потребоваться указанное видоизменение суммирования.

В n-ичной системе счисления каждая из цифр слагаемых и суммы b, c, B может принимать одно из n возможных значений. Цифры переносов e, E могут принимать вообще*) одно из трех значений: +1,0 или —1; но если все значения цифр n-ичной системы имеют один и тот же знак, то для e и E остаются только две, а не три возможности. Обозначив через $a_{\min}$ наименьшее из возможных значений цифр n-ичной системы счисления, а через $a_{\max}$ — наибольшее, $a_{\max} = a_{\min} + n$—1, запишем правила работы одноразрядного сумматора в виде

$$E = \begin{cases} 1, \text{если } a_{\max} < b + c + e, \\ 0, \text{если } a_{\min} \le b + c + e \le a_{\max}, \\ -1, \text{если } b + c + e < a_{\min}, \end{cases}$$

$$B = b + c + e - En.$$

Например, в десятичной системе с цифрами 0, 1, 2,..., 9

$$E = \begin{cases} 1, \text{если } 9 < b + c + e, \\ 0, \text{если } b + c + e \le 9, \end{cases}$$

$$B = b + c + e - 10E$$

случай, когда $b + c + e < 0$ невозможен).

2.1.2. Параллельные и последовательные сумматоры.

Полная схема сумматора может строиться из одноразрядных сумматоров разными способами. В зависимости от того, как это делается, различают параллельные и последовательные сумматоры. Возможен также промежуточный вариант — параллельно-последовательный сумматор.

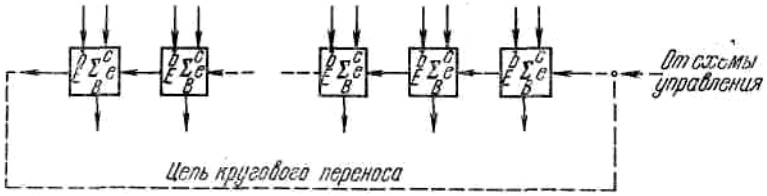


Рис. 2-1. Построение параллельного сумматора.

Построение *параллельного* сумматора показано на рис. 2-1. Прямоугольниками с буквой 2 внутри здесь, как и в дальнейшем, обозначаются одноразрядные сумматоры. Количество одноразрядных сумматоров равно количеству разрядов в складываемых числах. На выход b и c каждого из них подаются цифры одного из разрядов слагаемых. Входы e (кроме сумматора самого младшего разряда) соединены с выходами E соседних младших разрядов. Сумма появляется на выходах B одноразрядных сумматоров и на выходе E самого старшего разряда (добавочный разряд суммы). На вход e самого младшего разряда при обычном суммировании должна подаваться цифра «0», а при суммировании «с круговым переносом» — цифра с выхода E самого старшего разряда. В некоторых специальных случаях (см. раздел 3) на этот вход нужно подавать сигналы от схемы управления. В разделе 3.1.2 сказано также о некоторых тонкостях в построении параллельных сумматоров «с круговым переносом».

Рис. 2-2 иллюстрирует построение *последовательного* сумматора. В отличие от параллельной схемы здесь имеется всего один одноразрядный сумматор. Цифры слагаемых поступают на входы b и c по очереди: сначала цифры самого младшего разряда, затем цифры следующего по старшинству разряда и т. д. В первом такте на выходе B появляется младший разряд суммы, а на выходе E — цифра переноса во второй разряд. Цифра переноса специальным устройством (триггером, линией задержки) запоминается на один такт и затем поступает на

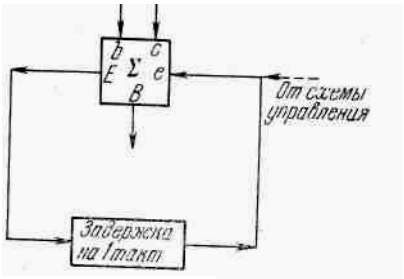


Рис. 2-2. Построение последовательного сумматора.

*) Если цифры, допустимые для каждого из разрядов, выбраны в соответствии с 1.3.2.

вход *e* одноразрядного сумматора одновременно с поступлением цифр второго разряда слагаемых на входы *b* и *c*; при этом на выходе *B* появляется вторая цифра суммы, а на выходе *E* — цифра переноса в третий разряд и т. д. В первом такте на входе *e* при обычном суммировании должна быть цифра «0»; однако, если в каких-либо специальных случаях это необходимо, от схемы управления может быть введена и какая-нибудь другая цифра.

На рис. 2-3 показано построение *последовательно-параллельного* сумматора. Для определенности предполагается, что в параллель суммируются четверки разрядов. Соответственно полная схема содержит 4 одноразрядных сумматора, соединенных между собой так же, как в параллельной схеме. Но — подобно тому как это имеет место в последовательной схеме — выход переноса *E* старшего из одноразрядных сумматоров соединен с устройством запоминания цифры переноса на 1 такт, с которого она может передаваться на вход *e* самого младшего одноразрядного сумматора. При сложении двух чисел на входы *b* и *c* одноразрядных сумматоров

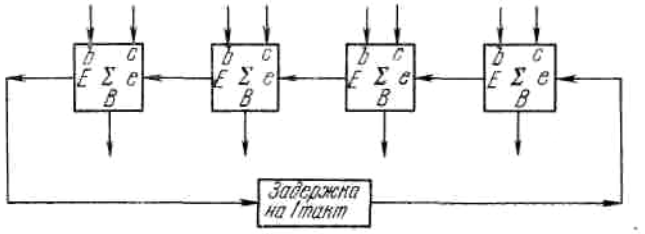


Рис. 2-3. Построение последовательно-параллельного сумматора.

сначала подаются четверки младших разрядов слагаемых. При этом на выходах *B* образуются 4 младших разряда суммы, а на выходе *E* самого старшего разряда — цифра переноса в 5-й разряд. В следующем такте на входы *b* и *c* поступают цифры 5-го, 6-го, 7-го и 8-го разрядов слагаемых; на вход *e* младшего из одноразрядных сумматоров возвращается цифра переноса в 5-й разряд; в результате на выходах *B* образуются 5-й — 8-й разряды суммы, а устройство запоминания воспринимает цифру переноса в 9-й разряд и т. д.

2.1.3. Комбинационные, накапливающие и амплитудные сумматоры.

В зависимости от того, по какому принципу построены одноразрядные сумматоры, различают схемы комбинационные, накапливающие и амплитудные. В любом случае может быть в принципе применено параллельное, последовательное или параллельно-последовательное построение, хотя в действительности, например, накапливающие сумматоры большей частью используют в параллельной схеме.

Одноразрядный сумматор *комбинационного типа* строится из логических элементов «и», «или» и «нет», либо, может быть, из каких-нибудь других логических элементов, используемых в машине. Элементы эти соединяются между собой так, чтобы из независимых входных переменных *b*, *c*, *e* (цифр слагаемых и переноса в данный разряд) образовать функции *B* и *E* (цифру суммы данного разряда и перенос в следующий разряд). В схеме имеются свои особые входы для каждой из входных переменных. Если одноразрядный комбинационный сумматор строится из потенциальных элементов, входные переменные должны подаваться все одновременно; выходные сигналы при этом появляются в принципе одновременно с появлением входных сигналов, а в действительности с некоторым запаздыванием, равным времени срабатывания логических элементов. При использовании элементов других типов (феррито-диодных и феррито-транзисторных, параметронов и т. д.) одновременность в подаче входных и появлении выходных сигналов может отсутствовать. Однако признаком комбинационного сумматора является то, что замена составляющих его элементов на тождественные по логике потенциальные элементы (может быть, гипотетические) приводит к схеме, в которой входные и выходные сигналы одновременны.

Основой одноразрядного сумматора *накапливающего типа* является счетчик импульсов. Для работы в системе счисления с основанием *n* счетчик должен вести счет по модулю *n.* Счетчик по модулю *n* представляет собой схему, составленную из нескольких 2-позиционных или каких-нибудь других цифровых элементов либо, может быть, из одного *n*-позиционного цифрового элемента; схема имеет *n* устойчивых состояний, которые проходятся последовательно, когда на вход счетчика поступают электрические импульсы. Подробнее устройство счетчиков рассматривается в разделе 2.6. Сумматоры накапливающего типа используются обычно тогда, когда значения всех цифр *n*-ич-ной системы счисления имеют один и тот же знак; большей частью при этом они все неотрицательны: 0, 1, 2,..., *n* — 1.

До начала суммирования счетчик устанавливается в положение «0». Затем на его вход подается столько импульсов, сколько единиц содержится в цифре первого слагаемого, после этого — столько импульсов, сколько единиц содержится в цифре второго слагаемого, и наконец,— импульс переноса, если в данный разряд переносится единица. Впрочем, порядок следования импульсов может быть и иным. Конечное состояние счетчика дает цифру суммы данного разряда. Если в процессе счета счетчик переходил из состояния *n* — 1 в состояние 0, то это указывает на наличие переноса в следующий разряд.

Имея параллельный накапливающий сумматор и получив на счетчиках сумму двух чисел, легко далее к этой сумме добавить третье число. Для этого операцию нужно вести точно так же, как и при сложении двух чисел, начиная ее от того момента, когда на сумматор уже поданы цифры первого слагаемого (роль первого слагаемого

здесь играет предыдущая сумма). Затем к полученному результату можно таким же образом добавить четвертое число и т. д. Отсюда, собственно, и происходит название «накапливающий сумматор».

Но и сумматоры других типов во многих случаях включаются так, что слагаемые поступают в них из двух регистров, а сумма затем возвращается в один из регистров, где заменяет, скажем, первое слагаемое. В этом регистре также легко организовать накопление суммы нескольких чисел. Чтобы избежать путаницы в терминологии, мы будем в дальнейшем регистр, из которого в сумматор поступает одно из слагаемых и куда возвращается сумма, называть *регистром-накопителем.* В случае использования параллельного накапливающего сумматора регистр-накопитель как бы входит частью в сам сумматор: его роль выполняют счетчики одноразрядных суммирующих схем.

Одноразрядный сумматор *амплитудного типа* по своим «внешним» свойствам более походит на схему комбинационного типа. В нем тоже имеются раздельные входы для каждой из входных переменных *(b, c* и *e),* а выходные переменные *(B* и *E)* формируются в принципе одновременно с поступлением входных сигналов, а в действительности с небольшой задержкой, равной времени срабатывания схемы. Однако внутри схемы вместо логических элементов (или наряду с ними) имеются устройства для сложения амплитуд токов, или напряжений, или каких-либо других физических величин, а также ограничители и другие элементы преобразования сигналов.

Накапливающие и амплитудные сумматоры в настоящее время в практике построения вычислительных машин используются сравнительно редко.

2.2. Построение двоичного одноразрядного комбинационного сумматора

2.2.1. Варианты построения из элементов «и», «или», «нет».

Двоичный одноразрядный сумматор комбинационного типа строится в соответствии с таблицей двоичного сложения (таблица 2-1).

Таблица 2-1

Двоичное сложение

b	*c*	*e*	*B*	*E*	*b*	*c*	*e*	*B*	*E*
0	0	0.	0	0	1	0	0	1	0
0	0	1	1	0	1	0	1	0	1
0	1	0	1	0	1	1	0	0	1
0	1	1	0	1	1	1	1	1	1

В таблице, как и прежде, через *b* и *c* обозначены цифры одного разряда слагаемых, через *e* — цифра переноса в данный разряд из предыдущего (младшего) разряда; *b*, *c* и *e* представляют независимые входные переменные, *B* и *E* — это соответственно сумма, которая должна получиться в рассматриваемом разряде, и перенос в следующий (старший) разряд; переменные *B* и *E* являются переключательными функциями, каждая из которых зависит от трех аргументов (*b c* и e). Формирование этих переключательных функций и является задачей двоичного одноразрядного сумматора. В соответствии с таблицей 2-1 можно записать канонические (нормальные дизъюнктивные) формы для функций *B* и *E:*

$$B = \bar{b}\,\bar{c}e + \bar{b}\,c\bar{e} + b\bar{c}\,\bar{e} + bce\,,$$

$$E = \bar{b}ce + b\bar{c}e + bc\bar{e} + bce\,.$$

Непосредственно по этим уравнениям из логических элементов «и», «или» и «нет» может быть составлено построение одноразрядного сумматора, приведенное на рис. 2-4. Инвертор (логический элемент «нет»), обозначенный на рисунке кружком, введен в схему потому, что на входах ее наряду с входными переменными *b, c* и *e* используются

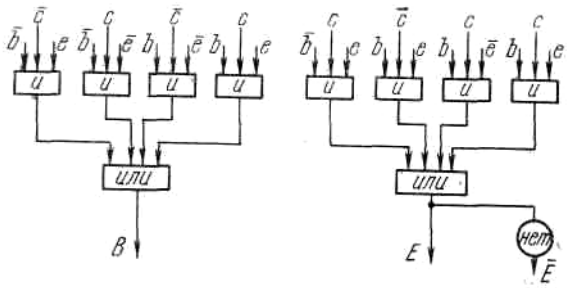


Рис. 2-4. Двоичный одноразрядный сумматор, построенный непосредственно по каноническим формам уравнений для *B* и *E.*

и все инверсии ($\bar{b}, \bar{c}$ и $\bar{e}$). Если цифры слагаемых *(b* и *c)* получаются от триггерных регистров, то их инверсии $\bar{b}$

и $\bar{c}$, возможно, так же доступны, как и прямые выходы b и c. Но для того, чтобы выходной сигнал переноса E мог использоваться в следующем разряде в качестве входного сигнала e, необходимо наряду с прямым выходом сигнала E иметь также и инверсный выход $\bar{E}$. Как мы увидим из дальнейшего, количество оборудования в схеме рис. 2-4 неоправданно велико: если бы элементы «и» и «или» выполнялись из диодов, то всего в схеме их содержалось бы 32 (по одному диоду на каждый вход элемента «и» или «или»). Другие недостатки этой схемы обсуждаются в разделе 2.4.

Фактически все работы по логике построения двоичных одноразрядных сумматоров комбинационного типа сводятся к тождественным преобразованиям канонических форм уравнений для B и E с целью получения схем, требующих меньшего количества оборудования или обладающих какими-нибудь другими достоинствами. Интересно, однако, что даже в такой простой на первый взгляд задаче ни один
из многочисленных формальных методов синтеза логических схем не дает оптимального решения хотя бы даже только с точки зрения количества оборудования. Например, использование метода минимизирующих карт Айкена*) приводит к уравнениям

$$B = (b + c + e)(b + \bar{c} + \bar{e})(\bar{b} + c + \bar{e})(\bar{b} + \bar{c} + e),$$

$$E = (b + c)(b + e)(c + e),$$

где знаки сложения и умножения используются в смысле логического сложения и логического умножения.

Соответствующее построение выполнено на рис. 2-5. Инвертор для формирования функции $\bar{E}$ в этой схеме

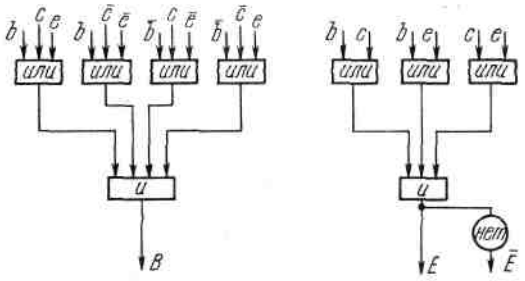


Рис. 2-5. Двоичный одноразрядный сумматор, построенный по «минимальным формам» для B и E.

поставлен по той же причине, что и в схеме рис. 2- 4: как и в предыдущей схеме, на входах наряду с независимыми переменными используются все инверсии. Количество оборудования здесь меньше, чем в схеме рис. 2-4, но все еще велико: если бы все элементы «и» и «или» выполнялись из диодов, то количество диодов в схеме было бы 25. При этом схема образования суммы *(B)* содержит столько же диодов, сколько и на рис. 2-4. Не использовано также то обстоятельство, что функции B и E частично связаны между собой, что дает возможность несколько сэкономить в оборудовании: на рис. 2-5, как и на рис. 2-4, схемы формирования функций B и E полностью самостоятельны.

Пример построения, в котором для сокращения количества оборудования использована взаимосвязь функций B и E, показан на рис. 2-6. Проверить, что приведенное построение является действительно одноразрядным сумматором, можно непосредственно по таблице двоичного

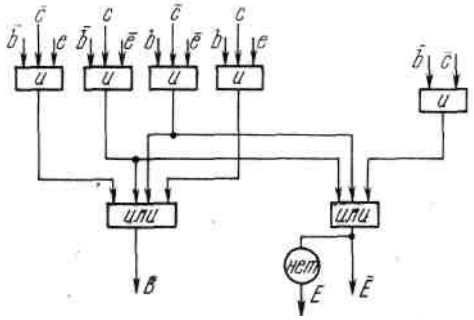


Рис. 2-6. Пример построения двоичного одноразрядного сумматора, в котором использована взаимосвязь между функциями B и E.

сложения (таблица 2-1 на стр. 64). Можно также записать уравнения, соответствующие схеме, и затем рядом тождественных преобразований привести их к канонической форме. Эти уравнения для схемы рис. 2-6 имеют вид

$$B = \bar{b}\bar{c}e + \bar{b}c\bar{e} + b\bar{c}\bar{e} + bce,$$

$$\bar{E} = \bar{b}c\bar{e} + b\bar{c}\bar{e} + \bar{b}\bar{c},$$

$$(E = \bar{E}).$$

*) См. «Синтез электронных вычислительных и управляющих схем», перев. с англ. под ред. В. И. Шестакова, ИЛ, 1955.

По существу только уравнение формирования функции *E* записано в форме, отличающейся от канонической.

В схеме рис. 2-6, если бы элементы «и» и «или» выполнялись из диодов, был бы всего 21 диод. Однако при этом требуется, чтобы выходной сигнал элемента «и» можно было подавать на входы двух элементов «или», что не всегда возможно. Наиболее удачные построения двоичных одноразрядных сумматоров были получены эмпирическим путем.

На рис. 2-7 приведены построения одноразрядных сумматоров, впервые использованные в вычислительных машинах BINAC и IBM-701; в течение длительного времени эти построения считались наилучшими. Схемы эти

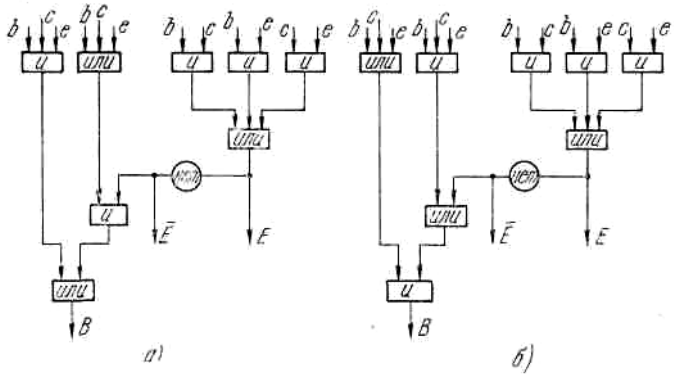


Рис. 2-7. Варианты построения двоичного одноразрядного сумматора:
а) схема вычислительной машины BINAK;
б) схема вычислительной машины IBM-701.

очень похожи между собой. Участки формирования сигнала переноса *(E)* в них одинаковы; участки формирования суммы *(B)* различаются тем, что вместо элементов «и» в одной схеме в другой стоят элементы «или», а вместо элементов «или» — элементы «и».

Как мы увидим из дальнейшего, такая замена возможна вообще в любом двоичном одноразрядном сумматоре комбинационного типа — как в участке формирования функции *B,* так и в участке формирования *E* (принцип взаимности).

По количеству оборудования схемы рис. 2-7 более экономны, чем рассмотренные выше. Если элементы «и» и «или» строятся из диодов, то для осуществления этих схем требуется по 19 диодов. При этом (в отличие, например, от схемы рис. 2-6) ни один из элементов не нагружается на входы двух других элементов. Другое достоинство этих построений состоит в том, что на их входах используются только прямые значения независимых переменных (*b*, *c* и *e),* но не их инверсии. На рисунке показана возможность использования наряду с выходным сигналом переноса *E* также его инверсии $\bar{E}$. Однако в этом — по крайней мере принципиально — нет необходимости.

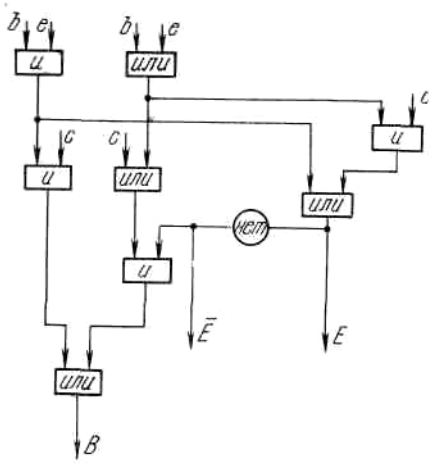


Рис. 2-8. Построение двоичного одноразрядного сумматора вычислительной машины NAREC.

Если работа одного элемента на входы двух других возможна, то можно еще более сократить количество оборудования. Схема, для осуществления которой требуется всего 16 диодов и 1 инвертор, приведена на рис. 2-8. Это — построение одноразрядного сумматора, впервые примененное некогда" в вычислительной машине NAREC. Как и в схемах рис. 2-7, на ее входах используются только прямые сигналы *b*, *c* и *e,* но не их инверсии.

Как показал М. Я. Вайнштейн, построить двоичный одноразрядный сумматор комбинационного типа, который содержал бы менее 16 диодов при 1 инверторе, невозможно. При этом использование на входах инверсий от входных переменных не помогло бы уменьшить количество диодов. Кроме схемы рис. 2-8, существует еще только одно принципиально отличное построение, содержащее 16 диодов и 1 инвертор и не использующее инверсий входных переменных; оно приведено на рис. 2-9.

Доказательство этих утверждений, включающее некоторые элементы перебора, сравнительно сложно, и мы

его здесь не воспроизводим[*]).

Несколько дополнительных вариантов построения двоичного одноразрядного сумматора, содержащих такое же количество оборудования, можно получить из схем рис. 2-8 и 2-9 путем применения принципа взаимности (см. 2.2.3), а также перестановками входных переменных, которые в указанные схемы входят несимметрично (см. например, рис. 2-15, *а* на стр. 71). Кроме того, можно найти несколько вариантов, требующих такого же количества оборудования, но использующих наряду с входными переменными их инверсии[**]).

С двумя инверторами количество диодов может быть еще меньше. На рис. 2-10 приведено построение, содержащее 14 диодов и два инвертора. Как и в предыдущих схемах, на входах используются только прямые сигналы *b*, *c* и *e,* но не их инверсии.

По сравнению со схемами, рассмотренными выше, построение рис. 2-10 отличается одной характерной особенностью. Как видно из рисунка, двоичный одноразрядный сумматор состоит здесь из двух совершенно одинаковых частей и дополнительно одного элемента «или». На чертеже эти две части обведены пунктирными линиями. Первая из них (слева) производит суммирование двух цифр слагаемых (*b* и *c)* без учета переноса в данный разряд. При этом вырабатывается промежуточный сигнал переноса (ε_1) и промежуточная цифра суммы (β). Перенос в следующий разряд при суммировании двух двоичных цифр может возникнуть только при условии, что обе эти цифры единицы:

$$\varepsilon_1 = bc;$$

сумма двух двоичных цифр равна единице, если какая-нибудь одна из этих цифр есть единица (но не обе одновременно):

$$\beta = (b + c)\bar{\varepsilon}_1 .$$

Устройство, которое производит суммирование двух двоичных цифр и вырабатывает по указанному правилу цифру суммы и перенос в следующий разряд, называется *полусумматором.*

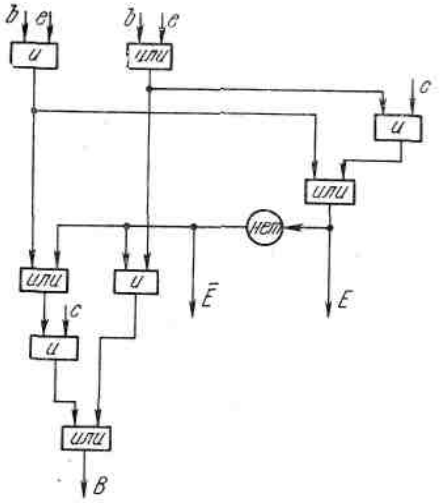


Рис. 2-9. Второй из возможных вариантов построения одноразрядного сумматора, содержащего 16 диодов и инвертор и не использующего инверсий входных переменных (вариант М. Я. Вайнштейна).

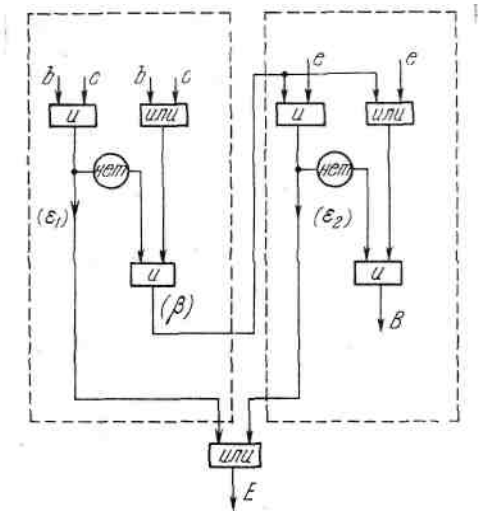


Рис. 2-10. Двоичный одноразрядный сумматор, построенный из двух полусумматоров

Второй полусумматор (на рис. 2-10 — справа) по тем же правилам суммирует цифру суммы двух слагаемых (β) с сигналом переноса в данный разряд *(e).* Если в первом полусумматоре сигнал переноса в следующий разряд не возник ($\varepsilon_1 = 0$), то он может возникнуть во втором полусумматоре — при условии, что одна из цифр слагаемых *b* или с равна единице (т. е. $\beta = 1$) и имеется перенос в данный разряд ($e = 1$); если перенос в следующий разряд возник в первом полусумматоре ($\varepsilon_1 = 1$), то при этом $\beta = 0$ и перенос во втором полусумматоре не может появиться. Окончательный сигнал переноса в следующий разряд (*E*) должен передаваться в том случае, когда сигнал переноса возник либо на первом, либо на втором полусумматоре:

$$E = \varepsilon_1 + \varepsilon_2;$$

сигнал суммы, появляющийся на выходе второго полусумматора, является, очевидно, суммой трех двоичных цифр *(b*, *c* и *e)* по модулю 2, т. е. выходным сигналом двоичного одноразрядного сумматора *(B).* Построение сумматора рис. 2-10 было использовано в вычислительной машине «Стрела».

[*]) Вайнштейн М. Я., О схемах одноразрядного сумматора, ДАН СССР, 1960, 135, № 5, стр. 1031-1034.

[**]) Вайнштейн М. Я., О схемах одноразрядного сумматора, ДАН СССР, 1960, 135, № 5, стр. 1031-1034.

На сокращенных функциональных схемах будем изображать полусумматоры в виде прямоугольников с буквой *о* внутри. Таким образом, полный одноразрядный сумматор рис. 2-10 может быть представлен так, как показано на рис. 2-11. Полусумматоры иногда называют *сумматорами на два входа;* при этом полный одноразрядный сумматор называется *сумматором на три входа.*

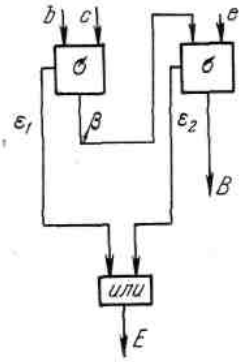


Рис. 2-11.. Представление двоичного одноразрядного сумматора в виде двух полусумматоров.

В ряде случаев полусумматоры используются в качестве самостоятельных функциональных устройств. При этом с целью унификации элементов машины каждый полный двоичный одноразрядный сумматор имеет смысл строить из двух полусумматоров. Но если отдельных полусумматоров требуется немного, то может оказаться, что по количеству оборудования (в частности, по количеству инверторов) более экономно было бы принять в качестве унифицированного типового элемента полный одноразрядный сумматор; там, где требуется полусумматор, можно просто не задействовать один из его входов (присоединить к нему постоянно сигнал «0»).

Другое возможное построение полусумматора, требующее несколько большего количества диодов, чем на рис. 2-10, приведено на рис. 2-12 (схема, впервые примененная в машине EDVAC).

Вообще прямым перебором нетрудно показать, что построение полусумматора, приведенное на рис. 2-10, является наиболее экономным из всех возможных (6 диодов и 1 инвертор).

2.2.2. Использование других элементов.

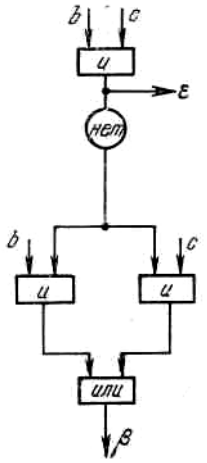


Рис. 2-12. Построение полусумматора вычислительной машины EDVAC.

Сравнительные оценки, которые мы приводили в разделе 2.2.1, исходили в основном из количества оборудования, необходимого для реализации того или иного построения одноразрядного сумматора, причем подсчитывалось отдельно количество инверторов и отдельно количество диодов. Ясно, однако, что эти оценки весьма условны.

В качестве характерного примера рассмотрим применение элементов типа «и-не». В последнее время появилось несколько разновидностей таких элементов, конструктивно оформленных в виде микромодулей. Первая логическая ступень элемента — «и» — может быть выполнена из сопротивлений, но чаще выполняется, как обычно, из диодов; существенно, однако, что выход этой первой ступени не может быть нигде использован непосредственно, поскольку он не выводится наружу из микромодуля, а соединен внутри со входом второй логической ступени — инвертора (обычно — полупроводниковый триод). В качестве выходного сигнала микромодуля используется выходной сигнал инвертора («не»), причем имеется возможность, объединив общим проводом выходы нескольких инверторов, получить без дополнительных затрат оборудования вторую ступень «и». Таким образом, типовая цепочка элементов имеет вид «и-не-и» либо просто «и-не» (если объединение выходов не используется).

С точки зрения логики цепочка «и-не-и» эквивалентна цепочке «и-или-не», т. к. $\overline{x_1}\overline{x_2}\overline{x_3} = \overline{x_1 + x_2 + \ldots + x_n}$; поэтому в принципе некоторые построения одноразрядных сумматоров, приведенные в разделе 2.2.1, могли бы быть повторены и на рассматриваемых элементах. Но при этом, если бы в качестве критерия для сравнения было выбрано по-прежнему количество оборудования, нас интересовало бы только количество микромодулей. Сколько входов задействовано в первой ступени «и» и сколько выходов инверторов объединено между собой, не имеет значения, если только не превышены ограничения, которые задаются конкретно для каждой системы микромодулей. Если микромодуль, для примера, позволяет осуществить входную схему «и» на 3 входа, а мы используем «и» на 2 входа, то диоды неиспользуемых входов первой ступени «и» все равно остаются внутри микромодуля. Даже если схема «и» вообще не нужна, а нужен только инвертор, затраты оборудования не уменьшаются: на входе инвертора используется как бы схема «и» с одним входом, другие входы «и» не задействованы.

На рис. 2-13 приведено одно из возможных построений двоичного комбинационного одноразрядного сумматора из элементов «и-не», повторяющее в основном идею построения рис. 2-7,*а* на стр. 66. Элементы «и-не» обозначены

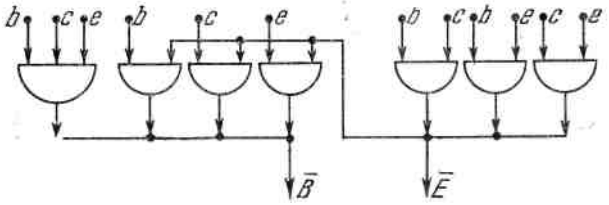


Рис. 2-13. Построение двоичного одноразрядного сумматора из элементов «и-не».

полукругами; максимальное количество входов «и» в одном элементе — 3. Вторые ступени «и» показаны в виде общих линий, соединяющих выходы элементов, причем, как видно из рисунка, максимальное количество объединенных между собой выходов в этом построении сумматора — четыре.

Одноразрядный сумматор, изображенный на рис. 2-13, получает только прямые сигналы входных переменных *b*, *c, e* и вырабатывает инверсные сигналы *B* и *E.* При необходимости их можно было бы инвертировать с помощью двух дополнительных элементов «и-не»; но, как будет видно из раздела 2.2.3, в этом может и не быть нужды. Без этих дополнительных инверторов схема рис. 2-13 содержит 7 элементов «и-не». Возможно, однако, что существуют и более экономные построения.

Если вместо элементов «и», «или» и «нет» применяется какая-нибудь другая система элементов, то построение сумматора будет, разумеется, совсем не таким, как рассматривалось выше. Не имея возможности останавливаться на разнообразных логических схемах, которые приходится строить в этих случаях, приведем лишь один интересный пример. На рис. 2-14 показано построение одноразрядного двоичного сумматора из параметронов. Каждый параметрон представлен здесь кружком с нечетным количеством входов (один или три) и одним выходом. Напомним, что выходной сигнал параметрона совпадает по фазе с б о л ь ш и н с т в о м сигналов на входах; таким образом, если параметрон имеет три входа, то при наличии единиц на двух или трех из них на выходе появляется единица, а при наличии на входах двух или трех нулей выходной сигнал есть нуль. Черточка на линии, соединяющей выход одного параметрона со входом другого, означает на рисунке, что обмотки здесь соединены накрест, т. е. что на вход второго параметрона подается инверсия от выходного сигнала первого.

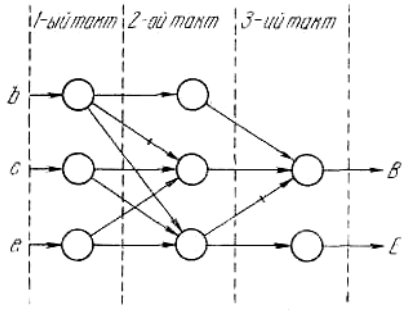


Рис. 2-14. Построение двоичного одноразрядного сумматора из параметронов.

В том, что схема рис. 2-14 действительно является одноразрядным двоичным сумматором, легко убедиться перебором всех комбинаций входных переменных в соответствии с таблицей 2-1 (стр. 64).

В схеме рис. 2-14 имеется ряд «лишних» ступеней — параметроны с одиночными входами, не выполняющие никаких логических функций. Стоят они потому, что схема сумматора работает в три такта (как и большинство схем на параметронах), и по всем цепям должны быть обеспечены равные запаздывания.

Входные сигналы в схему сумматора подаются здесь все одновременно, выходные же сигналы *(B, E)* появляются оба одновременно, но с опозданием на 3 такта относительно входных. Однако это запаздывание не носит принципиального характера, а связано только с техническими особенностями работы параметронов. Если бы вместо параметронов применялись элементы потенциального типа, выполняющие те же логические функции, что и параметроны, то никакого запаздывания сигналов в построении рис. 2-14 не было бы, кроме естественных задержек на срабатывание элементов. Аналогичное положение возникает при построении комбинационных сумматоров из феррито-диодных или феррито-транзисторных элементов.

Вообще говоря, имея любую функционально полную систему элементов, всегда можно построить из этих элементов схемы, выполняющие логические операции «и», «или» и «нет». Затем из таких схем можно составить сумматор в соответствии с одним из построений рис. 2-4 — 2-12. Читателю предлагается испытать этот путь для случая использования параметронов. Сопоставление полученных результатов с построением рис. 2-14 покажет, что такой путь является далеко не наилучшим и что каждый раз, когда применяется новая система логических элементов, приходится заново искать подходящие построения сумматоров.

2.2.3. Некоторые общие свойства двоичных одноразрядных сумматоров комбинационного типа.

Рассматривая таблицу 2-1 на стр. 64, нетрудно убедиться, что обе переключательные функции, формируемые двоичным одноразрядным сумматором, являются функциями *самодвойственными.* Это означает, что *инвертирование всех входных переменных—цифр слагаемых b и c и цифры переноса в данный разряд e — приводит к тому, что на выходах схемы вместо цифры суммы B и цифры переноса в следующий разряд E будут формироваться их инверсии.*

Например, при $b = c = e = 0$ на выходах схемы нормально получаем $B = 0$ и $E = 0$ (первая строка таблицы 2-1); инвертирование входных переменных, т. е. подача на входы вместо сигналов *b, c, e* их инверсий $\bar{b} = \bar{c} = \bar{e} = 1$, приведет к тому, что на обоих выходах получатся сигналы «1» (последняя строка таблицы), т. е. фактически $\bar{B}$ и $\bar{E}$. Если $b = c = 0$, а $e = 1$, то на выходах нормально получаем $B = 1$, $E = 0$ (вторая строка таблицы); при подаче на входы *b*, *c*, *e* инверсий от входных величин, т. е. сигналов $\bar{b} = \bar{c} = 1, \bar{e} = 0$, на выходе *B* получим «0», на выходе *E* — «1»(см. предпоследнюю строку таблицы), т. е. инверсии выходных сигналов *B* и *E,* и т. д.

Это свойство двоичных одноразрядных сумматоров иногда используется следующим образом.

Представим себе, что параллельный сумматор строится из одноразрядных сумматоров такого типа, как показано на рис. 2-7, *а* или *б* или на рис. 2-8, причем элементы «и» и «или» выполняются из диодов, а инвертором служит ламповый или транзисторный усилитель постоянного тока; предположим, кроме того, что триггерные

регистры, от которых получаются цифры слагаемых, выполнены из симметричных статических триггеров, так что в любом разряде инверсии входных переменных *b* и *c* легко доступны.

На рисунках, иллюстрирующих построение одноразрядных сумматоров, показано, что выходной сигнал переноса *E* получается на выходе одного из диодных логических элементов, а входной сигнал переноса *e* поступает на входы других диодных логических элементов. Однако в действительности в рассматриваемом случае нельзя непосредственно соединить входы переноса каждого из разрядов с выходом переноса соседнего младшего разряда, как это по логике должно быть сделано в параллельном сумматоре: слишком много каскадов диодных логическихсхем оказались бы включенными друг за другом. Длявосстановления перепадов сигналов нужно в каждом разряде или через каждые несколько разрядов ставить усилитель.

Имея в каждом разряде инверсии цифр слагаемых $\bar{b}$ и $\bar{c}$, можно было бы воспользоваться самодвойственностью функций *B* и *E* и обойтись без дополнительных усилителей. Для этого с входом переноса каждого из разрядов (*e*) нужно соединять не прямой выход переноса предыдущего младшего разряда (*E*), а инверсный выход ($\bar{E}$): уровни сигнала на инверсном выходе могут восстанавливаться инвертором, имеющимся в схеме. При этом оказалось бы, что, например, на вход 2-го разряда вместо сигнала *e* поступает сигнал $\bar{e}$; соответственно и на входы *b* и *c* 2-го разряда нужно было бы подать вместо цифр слагаемых их инверсии $\bar{b}$ и $\bar{c}$. Тогда на прямом выходе сигнала переноса 2-го разряда получался бы сигнал $\bar{E}$, а на его инверсном выходе — сигнал *E*. Сигнал $\bar{E}$ с инверсного выхода 2-го разряда является входным сигналом *e* для 3-го разряда; на входы слагаемых 3-го разряда подаются прямые сигналы *b* и *c,* на его прямом выходе при этом получается сигнал *E*, а на инверсном — сигнал $\bar{E}$, который и передается на 4-й разряд, и т. д.

Таким образом, в нечетные разряды поступали бы прямые сигналы цифр слагаемых и переноса, в четные — инверсные. Инверторы, имеющиеся по логике в одноразрядных сумматорах, выполняли бы роль усилителей сигналов в цепи переносов, так что никакие добавочные усилители не были бы нужны. Одноразрядные сумматоры при этом по конфигурации все совершенно одинаковы и всюду одинаково соединены между собой, но разводка сигналов слагаемых *(b* и *c*) на их входы в четных и нечетных разрядах различна. Различной должна быть и разводка выходных сигналов суммы на приемный регистр: из нечетных разрядов сумматора сигналы суммы должны поступать на входы установки «1» в триггерах, из четных — на входы установки «0» (потому что в первых образуются сигналы *B*, во вторых — сигналы $\bar{B}$).

Аналогичный прием используется иногда и при построении множительных устройств.

Из самодвойственности функций *B* и *E* вытекает и *принцип взаимности* для двоичных одноразрядных сумматоров комбинационного типа, о котором мы уже упоминали выше.

В алгебре логики широко известна теорема о том, что в любой логической схеме, составленной из элементов «и», «или» и «нет» (инверторов), замена всех элементов «и» элементами «или», всех элементов «или» элементами «и» и всех входных переменных их инверсиями (при сохранении без изменений конфигурации схемы и элементов «нет») приводит к тому, что на выходах схемы формируются инверсии тех переключательных функций, которые формировались исходной схемой.

Справедливость этого утверждения вытекает непосредственно из того, что оно, как легко проверить, справедливо для каждого элемента «и» и каждого элемента «или» в отдельности.

Например, переключательная функция *x* на выходе элемента «и» равна «1», если все входные переменные (скажем, x_1 и x_2) равны единице; инверсия от *x* равна единице, если либо x_1 либо x_2 является нулем, т. е. либо $\bar{x}_1$, либо $\bar{x}_2$ есть «1»:

$$\overline{x_1 x_2} = \bar{x}_1 + \bar{x}_2;$$

таким образом, заменив элемент «и» элементом «или» и подав на его входы инверсии от входных переменных, получим инверсию от функции *x.* Аналогичным образом для элемента «или»

$$\overline{x_1 + x_2} = \bar{x}_1 \bar{x}_2.$$

Путем индукции эта теорема обобщается на сколь угодно сложную*) схему, составленную из элементов «и» и «или». Если в такой схеме поменять местами элементы «и» и «или» и инвертировать входные переменные, то по крайней мере на выходах тех элементов, на которые поступают только входные переменные, будут получаться инверсии от функций, которые получались в исходной схеме. Эти функции являются, может быть, входными переменными для других элементов. Следовательно, и эти другие элементы будут теперь формировать инверсии от тех функций, которые формировались ими в исходной схеме, и т. д. Следовательно, и на конечных выходах схемы после указанной замены будут формироваться инверсии от выходных функций прежней схемы. Наличие в схеме инверторов, если при трансформации схемы они остаются на своих местах, не меняет дела.

Применим теперь эту теорему к двоичному одноразрядному сумматору комбинационного типа (или вообще к любой схеме, формирующей самодвойственные функции). Произведя сначала по обычному правилу замену всех элементов «и» элементами «или», всех элементов «или» элементами «и» и всех входных переменных (*b*, *c* и *e*) их

*) Но без перекрестных связей.

инверсиями $(\overline{b},\overline{c},\overline{e})$, мы получим на выходах схемы вместо *B* и *E* инверсии выходных функций $\overline{B}$ и $\overline{E}$. Затем, не меняя больше схему, заменим вновь на ее входах инверсии входных переменных $\overline{b},\overline{c}$ и $\overline{e}$ прямыми сигналами *b*, *c*, *e.* Так как функции *B* и *E* самодвойственны, то при этом на выходах новой схемы вместо $\overline{B}$ и $\overline{E}$ получатся снова сигналы *B* и *E.* Полученное построение, таким образом, снова является двоичным одноразрядным сумматором. Это и есть принцип взаимности: *если, в некотором логическом построении двоичного одноразрядного сумматора комбинационного типа, составленном из элементов «и», «или» и «нет», заменить все элементы «и» элементами «или», а все элементы «или» — элементами «и», сохранив при этом без изменения конфигурацию схемы, все элементы «нет» и все входные переменные, то при этом будет получено другое возможное построение двоичного одноразрядного сумматора.*

Теорема эта может применяться как ко всему одноразрядному сумматору в целом, так и отдельно к участкам, формирующим функции *B* и *E,* если это возможно. Иллюстрацией частичного использования принципа взаимности (только применительно к участку формирования суммы) является рис. 2-7 на стр. 66. На рис. 2-15 показаны примеры полного использования принципа взаимности.

Заметим здесь, что выходные функции, формируемые полусумматором, не являются самодвойственными, и к отдельному полусумматору принцип взаимности неприменим. Хотя полный одноразрядный сумматор рис. 2-15, *б,* подобно схеме рис. 2-10, состоит из двух одинаковых частей, каждая часть в отдельности, заключенная в пунктирную рамку, здесь не является полусумматором.

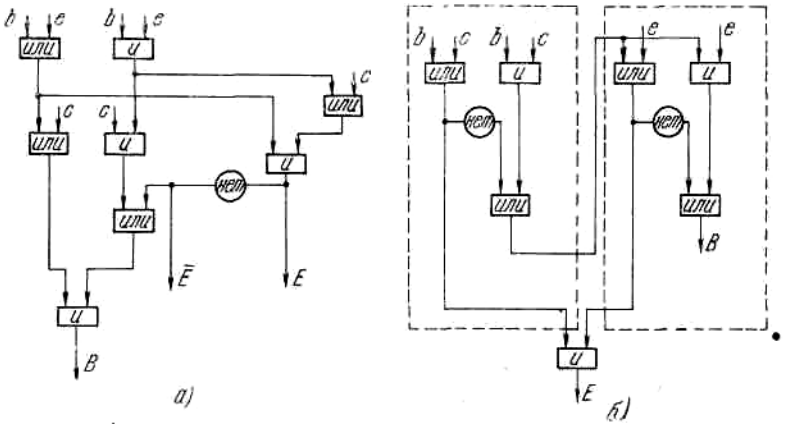


Рис. 2-15. Построения двоичных одноразрядных сумматоров, полученные путем применения принципа взаимности: *а)* из схемы рис. 2-8; б) из схемы рис. 2-10.

2.3. Одноразрядные комбинационные сумматоры для десятичной и других систем счисления

2.3.1. Принципы построения десятичных сумматоров комбинационного типа.

Задача о построении десятичного одноразрядного сумматора могла бы в принципе решаться аналогично задаче о построении двоичного одноразрядного сумматора. Однако если там шла речь о получении двух переключательных функций, каждая из которых зависела от трех аргументов, то здесь задача состоит в получении по меньшей мере 5 переключательных функций от 9 аргументов. (Каждая из десятичных цифр слагаемых и десятичная цифра суммы представляются не менее чем четырьмя двоичнымицифрами; кроме того, должен быть учтен сигнал переноса в данный разряд и сформирован сигнал переноса в следующий разряд.) Правда, не все комбинации входных переменных возможны. Вообще в 9 двоичных разрядах может быть $2^9 = 512$ различных комбинаций. Однако если эти 9 разрядов используются для представления двух десятичных цифр (слагаемых) и одной двоичной цифры (переноса в данный разряд), то в них может быть только 10×10×2=200 различных комбинаций. Следовательно, таблица десятичного сложения, построенная по аналогии с таблицей двоичного сложения (таблица 2-1 на стр. 64), должна содержать вместо восьми двести строк. Столбцов в ней тоже должно быть не два, как в таблице двоичного сложения, а пять (по числу выходных функций).

Имея такую таблицу, можно было бы далее выписать канонические формы для пяти выходных функций и тождественными преобразованиями этих уравнений попытаться найти оптимальное построение десятичного одноразрядного сумматора. Но если формальные методы синтеза логических схем оказались неэффективными применительно к построению двоичных сумматоров, то тем более трудно воспользоваться ими в рассматриваемой задаче. При этом вряд ли можно надеяться и на эмпирические методы (подбор оптимальной схемы, перебор всех возможностей), используемые при построении двоичных сумматоров.

Задача об отыскании оптимального построения десятичного одноразрядного сумматора оказалась бы еще значительно более сложной (но при этом и намного более важной), если бы заранее не был задан способ кодирования десятичных цифр. В этом случае может быть поставлена цель отыскать такой способ кодирования, при котором схема десятичного одноразрядного сумматора получается наиболее простой, а также, разумеется, построить эту оптимальную схему сумматора. Но даже и для какого-нибудь определенного способа кодирования

задача о построении оптимальной схемы десятичного одноразрядного сумматора не решена. При использовании обычных кодов задачу суммирования десятичных цифр большей частью удается подразделить на два этапа:

(1) суммирование двоичных кодов десятичных цифр по правилам двоичной арифметики;

(2) коррекция результата путем добавления или вычитания некоторой поправки, величина которой не слишком сложным образом связана с величиной некорректированного результата.

Пусть, например, десятичные цифры записываются 4-разрядным двоичным кодом «8, 4, 2, 1» (см. раздел 1.6). Предположим, что на первом этапе две десятичные цифры слагаемых и цифра переноса в данный десятичный разряд просуммированы по правилам двоичной арифметики. Далее возможны 3 случая:

(1) Если полученный результат меньше чем 10, то никакой дальнейшей коррекции не требуется: четыре двоичных разряда суммы, полученные по правилам двоичной арифметики, непосредственно являются изображением десятичной цифры суммы; переноса в следующий десятичный разряд нет.

(2) Если результат выполнения первого этапа заключен в пределах от 10 до 15, то необходимо передать сигнал переноса единицы в следующий десятичный разряд. При этом в данном десятичном разряде из полученной суммы нужно вычесть 10.

(3) Если результат выполнения первого этапа больше или равен 16 (вообще он может получиться до 19, так как 9+9+1 = 19, где 9 — это максимальное значение цифры каждого из слагаемых, а 1 — максимальное значение цифры переноса в данный разряд), то также необходимо передать сигнал переноса единицы в следующий десятичный разряд. Но при этом в данном десятичном разряде сумма оказывается сразу уменьшенной на 16, так как сумма представлена всего четырьмя двоичными разрядами, а пятый двоичный разряд с весом 16 в ней отсутствует. В действительности же она должна быть уменьшена только на 10. Следовательно, коррекция должна состоять в добавлении числа 6.

На самом деле в случае (2) коррекцию можно выполнять по тому же правилу, что и в случае (3). Если сумма, полученная в результате выполнения первого этапа, заключена в пределах от 10 до 15, то добавление к ней числа 6 приведет к тому, что она станет больше или равна 16. При этом из нее автоматически вычтется число 16 (так как двоичный разряд с весом 16 отсутствует). В итоге получится, что сумма изменена на величину — 16+6 = —10, как это и требуется.

Аналогичные правила для выполнения суммирования в коде «с излишком 3» были приведены в разделе 1.6.2. Если десятичные цифры слагаемых представлены каждая четверкой двоичных разрядов в коде «с излишком 3» и таким же кодом должна быть представлена десятичная цифра суммы, то перенос в следующий десятичный разряд образуется при суммировании четверок двоичных разрядов по правилам двоичной арифметики, а коррекция суммы состоит либо в вычитании числа 3, если перенос в следующий десятичный разряд отсутствует, либо в добавлении числа 3, если имеется перенос в следующий десятичный разряд.

Необходимо, однако, учесть, что код «8, 4, 2, 1» и код «с излишком 3» обладают тем общим свойством, что внутри каждой тетрады двоичным разрядом приписываются их естественные веса (правда, в коде «с излишком 3» число «3» является постоянной аддитивной поправкой). Именно по этой причине сложение двоичных представлений пары десятичных цифр выполняется внутри тетрады по правилам двоичной арифметики, а коррекция связана только с отсутствием или наличием переноса в следующую тетраду, причем сигнал переноса вырабатывается нестандартным образом. В других кодах положение может быть несколько иным.

Например, в кодах «7, 4, 2, 1» и «5, 4, 2, 1» (см. таблицу 1-6 на стр. 43) старший двоичный разряд тетрады имеет искусственный вес. Поэтому сложение трех младших двоичных разрядов может выполняться по правилам двоичной арифметики, а перенос в четвертый двоичный разряд должен формироваться специальными схемами.

При этом в коде «7, 4, 2, 1» требуется коррекция (причем разная) как при возникновении переноса в 4-й разряд тетрады, так и при возникновении переноса в следующий десятичный разряд; формирование как одного, так и другого сигнала переноса требует специальных схем. В коде «5, 4, 2, 1» коррекция требуется только при возникновении переноса в старший разряд тетрады, а при возникновении переноса в следующий десятичный разряд никакой коррекции не нужно (так как вес старшего разряда тетрады равен 5); перенос в следующий десятичный разряд образуется по правилам двоичной арифметики при сложении старших двоичных разрядов тетрад, с учетом сигнала переноса в этот разряд из младших двоичных разрядов; специальной схемы для образования десятичного переноса не требуется.

Читателю предлагается самостоятельно рассмотреть правила сложения в кодах «5, 4, 2, 1» и «7, 4, 2, 1».

2.3.2. Однотактные десятичные сумматоры.

При построении десятичных сумматоров в соответствии с принципами, изложенными в 2.3.1, широко используется двоичная техника. Как первый этап (суммирование двоичных кодов цифр слагаемых и цифры переноса в данный разряд), так и второй этап (добавление корректирующей поправки) выполняются по правилам двоичной арифметики и могут быть реализованы с помощью двоичных сумматоров.

На рис. 2-16 приведена схема десятичного одноразрядного комбинационного сумматора для кода «8, 4, 2, 1», построенная указанным способом.

На этом рисунке верхний ряд двоичных сумматоров образует 4-разрядный параллельный сумматор, осуществляющий первый этап операции — сложение двоичных представлений цифр слагаемых и цифры переноса

в данный десятичный разряд. Сложение это выполняется по правилам двоичной арифметики, причем 4-разрядная сумма появляется на выходах *B* одноразрядных сумматоров.

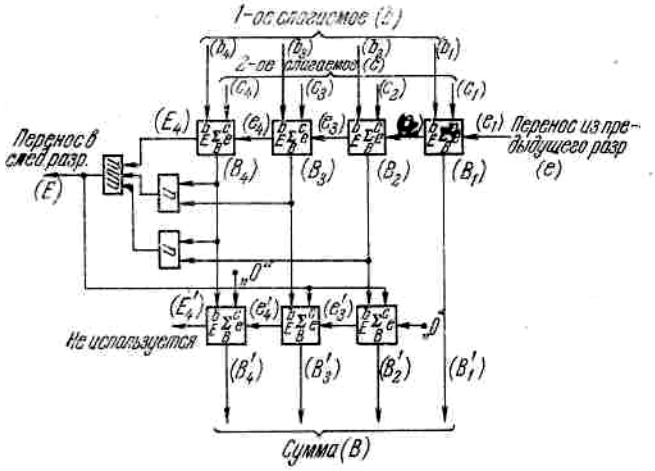


Рис. 2-16. Десятичный одноразрядный сумматор комбинационного типа для кода «8, 4, 2, 1», построенный из двоичных сумматоров.

Наличие сигнала переноса на выходе *E* старшего (левого) одноразрядного сумматора верхнего ряда означает, что результат первого этапа больше или равен 16. В этом случае, как говорилось, должен передаваться сигнал переноса в следующий десятичный разряд. Кроме того, перенос в следующий десятичный разряд должен передаваться и в тех случаях, когда сумма получается в пределах от 10 до 15, т. е. когда в ее двоичном представлении имеются единицы в четвертом и втором или четвертом и третьем разрядах (или, может быть, одновременно в четвертом, третьем и втором разрядах). В соответствии с этими правилами на рисунке построена схема образования сигнала переноса в следующий десятичный разряд.

Каким бы способом ни был получен сигнал переноса в следующий десятичный разряд, в данном разряде при наличии этого сигнала необходимо выполнить коррекцию, прибавив поправку +6; если при добавлении поправки результат станет больше или равен 16, то число 16 нужно отбросить.

Коррекция выполняется вторым рядом двоичных одноразрядных сумматоров (нижний ряд сумматоров на рис. 2-16). В первом справа (младшем) двоичном разряде сумматор не нужен, так как число 6 четно и в его двоичном представлении (0110) младший разряд равен нулю. В остальных разрядах в нижнем ряду имеется по одноразрядному двоичному сумматору, которые в совокупности образуют 3-разрядный параллельный сумматор. На один из его входов (входы «*b*» одноразрядных сумматоров) подается некорректированная сумма с выходов верхнего ряда сумматоров. На второй вход (входы «c») подается либо число 000 (0), если перенос в следующий десятичный разряд отсутствует, либо число 011 (0), т. е. 6, если имеется перенос в следующий десятичный разряд.

Правильное (скорректированное) двоичное представление десятичной цифры суммы получается на выходах *B* одноразрядных сумматоров нижнего ряда. Выход *E* старшего из одноразрядных сумматоров нижнего ряда не используется; поэтому в случае, когда при добавлении корректирующей поправки результат должен получаться больше или равным 16, он автоматически уменьшается на 16.

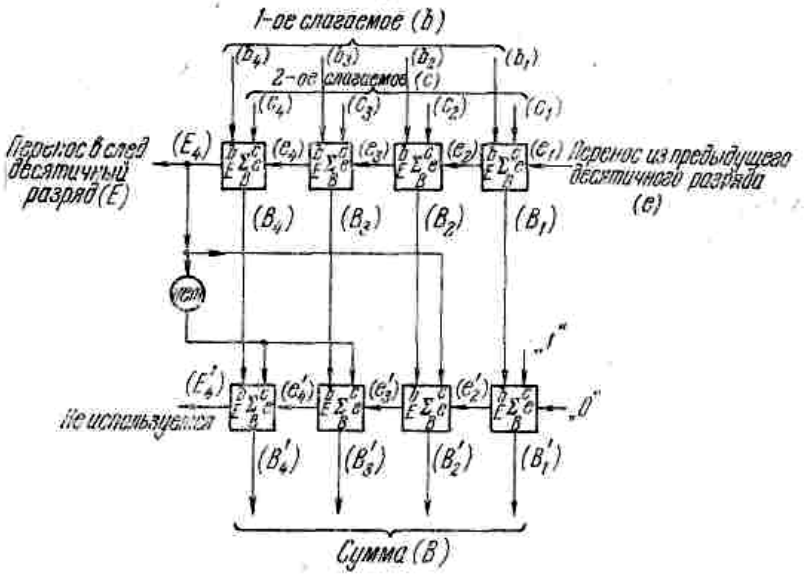


Рис. 2-17. Десятичный одноразрядный сумматор комбинационного типа для кода с излишком 3, построенный из двоичных сумматоров.

Аналогичным образом на рис. 2-17 построен десятичный одноразрядный сумматор для кода с излишком 3. Верхний ряд двоичных одноразрядных сумматоров на рисунке производит по правилам двоичной арифметики сложение двоичных представлений десятичных цифр слагаемых и цифры переноса в данный десятичный разряд. Сигнал переноса, образующийся на выходе старшего одноразрядного сумматора верхнего ряда, является непосредственно сигналом переноса в следующий десятичный разряд. Нижний ряд сумматоров производит либо сложение «+3» (если имеется перенос из данного десятичного разряда в следующий), либо вычитание «—3» (если переноса из данного десятичного разряда нет). Вместо вычитания «—3» фактически выполняется сложение «+13»; при этом результат должен был бы получиться непременно больше 16; ввиду отсутствия двоичного разряда с весом 16 величина 16 автоматически отбрасывается, так что в итоге поправка оказывается равной —16 +13 = — 3, как это и требуется. Схемы рис. 2-16 и 2-17 содержат, очевидно, много излишнего оборудования. Не говоря уже о том, что вообще разделение суммирования десятичных цифр на два этапа (суммирование по правилам двоичной арифметики и коррекция) не является, вероятно, наилучшим решением, каждый из этих двух этапов в приведенных схемах реализуется слишком сложным способом.

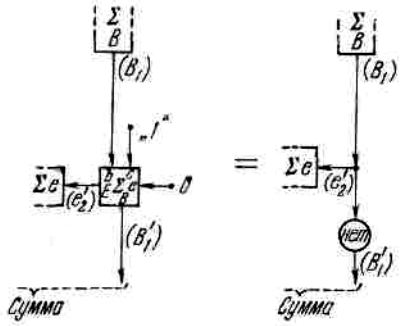


Рис. 2-18. Упрощение к схеме рис. 2-17.

Верхний ряд одноразрядных сумматоров вообще мог бы производить сложение любых чисел от 0 до 15; учитывая также, что сигнал переноса может принимать одно из двух значений, найдем, что всего на входах может быть $16^2 \times 2 = 512$ различных комбинаций. В действительности же в рассмотренных схемах на каждую группу входов может подаваться лишь одно из 10 чисел, так что всего на входах возможно только $10^2 \times 2 = 200$ различных комбинаций слагаемых.

Еще слабее использованы возможности нижнего ряда сумматоров. Например, в схеме рис. 2-16 трехразрядный сумматор нижнего ряда мог бы работать с $8^2 \times 2 = 128$ различными комбинациями входных переменных; используется же он так, что на одной группе входов может быть одно из 8 различных чисел (входы «*b*»), на другой — одно из двух чисел (входы «с» для корректирующей поправки), а на входе переноса — всегда «0», т. е. всего может быть только 10 комбинаций.

Таким образом, оборудование, задействованное в схемах рис. 2-16 и 2-17, может выполнять значительно более сложные функции, чем это необходимо; по-видимому, и само оно намного сложнее, чем нужно.

В схеме рис. 2-16 можно достичь некоторого сокращения оборудования, заменив в нижнем ряду крайний правый и крайний левый одноразрядные двоичные сумматоры полусумматорами. Как видно из рисунка, в обоих этих одноразрядных сумматорах имеется по одному незадействованному входу (на них постоянно подводятся сигналы «0»).

В схеме рис. 2-17 особенно плохо использован крайний правый одноразрядный двоичный сумматор нижнего ряда. В нем выходная цифра крайнего одноразрядного сумматора верхнего ряда всегда складывается с единицей. Ясно, что результат сложения по модулю 2 некоторой двоичной цифры с единицей в любом случае противоположен исходной цифре ($0 \oplus 1 = 1$; $1 \oplus 1 = 0$); перенос в следующий разряд возникает тогда и только тогда, когда исходная двоичная цифра есть «1». Поэтому указанный одноразрядный двоичный сумматор может быть заменен просто одним инвертором, а на вход переноса второго справа одноразрядного сумматора нижнего ряда можно непосредственно подать входной сигнал инвертора (рис. 2-18). Более радикальный путь сокращения оборудования состоит в том, чтобы по крайней мере в отдельных частях вообще отказаться от использования двоичных сумматоров и вместо этого применить специальные логические построения.

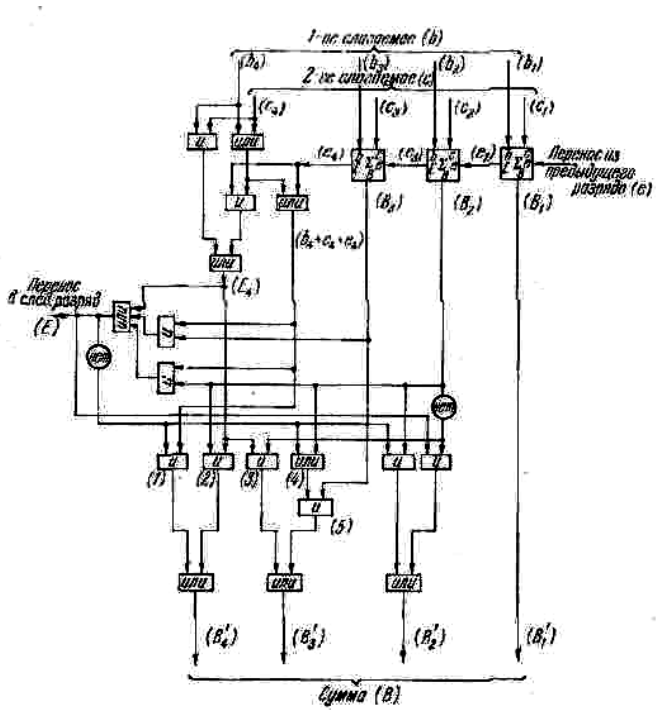


Рис. 2-19. Десятичный одноразрядный сумматор комбинационного типа для кода «8, 4,2, 1», в котором часть двоичных сумматоров заменена специальными логическими схемами.

На рис. 2-19 показано построение, полученное из рис. 2-16 заменой всего нижнего ряда двоичных сумматоров и левого из сумматоров верхнего ряда специальными логическими схемами*).

*) Это построение, как и схемы рис. 2-16 —2-17, в общих чертах заимствовано из книги Р. К. Ричардса «Арифметические операции на цифровых вычислительных машинах», ИЛ, 1957. В деталях имеются отличия, дающие некоторую дополнительную экономию в оборудовании.

Читателю предлагается самостоятельно проверить, что построение, приведенное на этом рисунке, действительно представляет собой схему десятичного одноразрядного сумматора для кода «8, 4, 2, 1». Схема рис. 2-19 скорее всего не является еще оптимальной с точки зрения количества оборудования. Получена она эмпирическим путем, и недостатком ее является значительно меньшая степень унификации отдельных частей, чем в схеме рис. 2-16.

Если экономии в количестве оборудования не придается решающего значения, то можно построить схемы десятичных одноразрядных сумматоров с еще большей степенью унификации отдельных частей, чем, скажем, в схемах рис. 2-16 или 2-17. Пример такого построения приведен на рис. 2-20[*]).

Десятичный одноразрядный сумматор для кода с излишком 3, представленный на этом рисунке, состоит из четырех одинаковых частей. Эти части изображены в виде прямоугольников со значком $\sum 5$ внутри. Каждая из них представляет собой *двоичный одноразрядный сумматор на пять входов.* Входы для пяти двоичных цифр обозначены соответственно *b*, *c*, *d*, *e′*, *e″*.

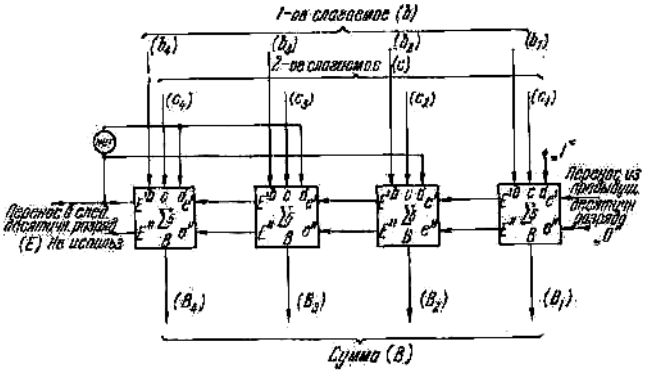


Рис. 2-20. Десятичный одноразрядный сумматор комбинационного типа для кода с излишком 3, выполненный из четырех двоичных сумматоров на 5 входов.

Выходными переменными такого сумматора являются цифра суммы данного разряда (B) и две цифры переноса в следующий старший разряд (*E′* и *E″*). Сумматор устроен так, что если все пять входных цифр равны нулю, то $B = 0$ и $E' = E'' = 0$; если на одном из входов имеется единица, то $B = 1$, $E' = E'' = 0$; если на двух из пяти входов имеются единицы, то $B = 0$, а один из сигналов *E′* или *E″* равен единице; если на трех из пяти входов имеются единицы, то $B = 1$ и один из сигналов *E′* или *E″* равен единице; если на четырех входах имеются единицы, то $B = 0$, $E' = E'' = 1$; наконец, при наличии единиц на всех пяти входах должно получаться $B = 1$, $E' = E'' = 1$.

В общем случае при построении сумматора на 5 входов выходы *E′* и *E″* равноправны: если единицы имеются на двух или на трех входах, то в принципе все равно, где появится сигнал переноса — на выходе *E′* или на выходе *E″*. Однако для схемы рис. 2-20 существенно, чтобы наличие хотя бы двух единиц на входах *b*, *c, e′* давало сигнал переноса на выходе *E′*, а наличие единиц на входах *d* и *e″* вызывало бы появление сигнала переноса на выходе *E″*. Случаи использования сумматоров на 5 входов, где это не играет роли, имеются, например, в разделе 4.5.

В рассматриваемой схеме двоичные одноразрядные сумматоры на 5 входов используются так, что на входы *b* и *c* подаются цифры двоичных представлений слагаемых, на вход *e′* — сигнал переноса из предыдущего двоичного разряда, который может передаваться при сложении двоичных представлений слагаемых; *E′* есть сигнал переноса в следующий двоичный разряд, возникающий при суммировании в данном разряде цифр *b* и *c* с учетом *e′*. На вход *d* подается цифра корректирующей поправки, а *e″* и *E″* — это соответственно входной и выходной сигналы переноса, возникающие при выполнении коррекции.

Теперь ясно, что схема рис. 2-20 по существу ничем не отличается от схемы рис. 2-17. Каждую пару двоичных одноразрядных сумматоров на 3 входа, которые на рис. 2-17 изображены один под другим, соединенных так, как показано на этом рисунке, можно представить в виде одного одноразрядного двоичного сумматора на 5 входов. Входы *b*, *c* и *e* верхнего одноразрядного сумматора на 3 входа будут являться соответственно входами *b*, *c* и *e″* одноразрядного сумматора на 5 входов; входы *c* и *e* нижнего одноразрядного сумматора на 3 входа являются соответственно входами *d* и *e″* сумматора на 5 входов; выход *B* нижнего из пары сумматоров на 3 входа является выходом *B* сумматора на 5 входов, а выходы *E* верхнего и нижнего сумматоров на 3 входа являются соответственно выходами *E′* и *E″* сумматора на 5 входов. Возможно, однако, что сумматор на 5 входов можно строить не из пары сумматоров на 3 входа, а каким-либо более экономным путем (подобно тому, как сумматор на 3 входа не обязательно строить из пары полусумматоров). Если бы было найдено удачное построение двоичного одноразрядного сумматора на 5 входов, то схема рис. 2-20 могла бы оказаться даже более экономной, чем, скажем, схема рис. 2-17. Пока, однако, такие построения неизвестны.

Схема рис. 2-20 представляет, по-видимому, определенный интерес с точки зрения применения микроэлектроники. При разработке микромодулей важно, чтобы устройство можно было разделить на

[*]) Идея схемы заимствована из работы Stone. Electronic Adder-accumulator (пат. США, кл. 235-61, № 2705108 от 29.03.1955 г.), хотя там речь идет не о комбинационном, а о накапливающем сумматоре.

сравнительно крупные однотипные части, причем количество внешних связей между такими частями должно быть относительно невелико. Сумматор рис. 2-20 как раз отвечает таким требованиям.

В связи с развитием микроэлектроники вновь пробудился интерес и к схемам прямого суммирования десятичных цифр. На рис. 2-21 показана, в частности, квадратная матрица, выполняющая прямое суммирование двух десятичных цифр*); при этом, правда, не учтена цифра переноса из младшего разряда и не формируется цифра переноса в следующий разряд, так что матрица не является еще полным одноразрядным сумматором. Элементы «и», стоящие на пересечениях вертикальных и горизонтальных проводов матрицы, могут быть выполнены в виде транзисторов с управлением по базе и по эмиттеру, а присоединение коллекторов нескольких (в данном случае 10) транзисторов к одному (диагональному) проводу с общей нагрузкой образует элементы «или». Предполагается, что в микроисполнении все транзисторы матрицы можно изготовить на общей подложке; с точки зрения технолога удобно, что все элементы в матрице однородны (100 одинаковых транзисторов и 10 одинаковых сопротивлений), что внутренние соединения просты и весьма систематичны и что количество внешних выводов (30) невелико по отношению к количеству элементов внутри матрицы (110).

Несколько рассмотренных выше схем сумматоров для кода «8, 4, 2, 1» и для кода с излишком 3 и одна схема прямого суммирования десятичных цифр далеко не исчерпывают, конечно, всего разнообразия десятичных сумматоров. Но ни систематизации возможных вариантов, ни обобщающей теории нет, а рассмотреть хотя бы все известные варианты не представляется возможным.

Приведенные в настоящем разделе примеры должны в основном проиллюстрировать принципы, используемые вообще при построении схем десятичных сумматоров.

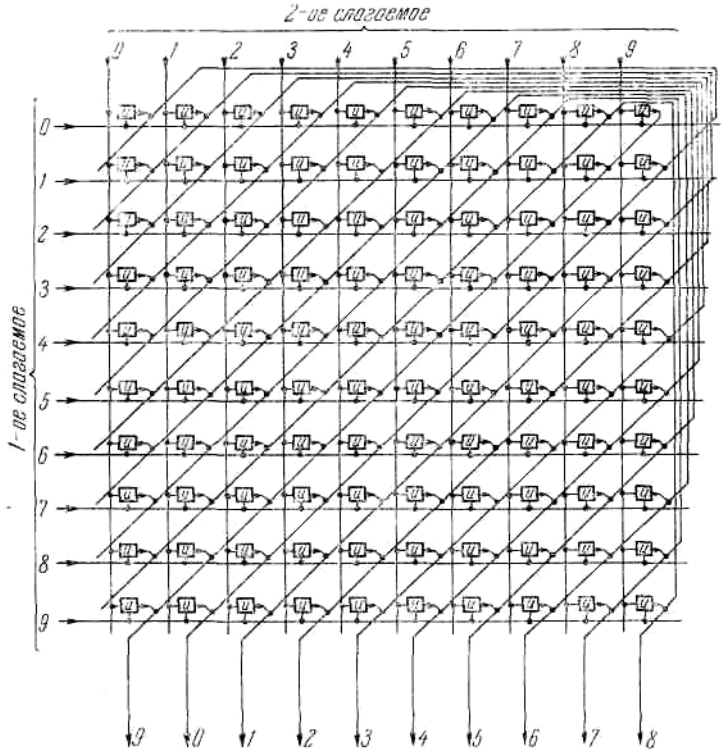


Рис. 2-21. Однородная матрица для прямого суммирования десятичных цифр.

2.3.3. Многотактные схемы десятичных сумматоров. Некоторые общие соображения.

Сравнивая различные системы счисления с точки зрения количества необходимого оборудования, мы нашли, что при использовании двоично-кодированной десятичной системы количество цифровых элементов или элементов запоминания возрастает примерно на 20,4% по сравнению с чисто двоичной системой (см. 1.2.2). Этот вывод получен в предположении, что для представления одной десятичной цифры требуется 4 двоичных элемента, т. е. что оборудование на один десятичный разряд равно оборудованию на четыре двоичных разряда. Рассматривая, однако, рис. 2-16 или 2-17, нетрудно убедиться, что количество логических элементов, приходящееся на один разряд десятичного комбинационного сумматора, эквивалентно примерно оборудованию 8-разрядного параллельного двоичного сумматора. Поэтому параллельный десятичный комбинационный сумматор по количеству оборудования превосходит параллельный двоичный сумматор не в 1,204 раза, а примерно в 2,4—2,5 раза. Наиболее радикальным средством сокращения количества оборудования в десятичных сумматорах является применение многотактных схем.

Представим себе, например (см. рис. 2-22, *а),* что параллельный десятичный сумматор для кода «8, 4, 2, 1» строится из одноразрядных сумматоров рис. 2-16. Цифры слагаемых поступают в сумматор в виде уровней напряжения из двух регистров — *В и С;* двоичные триггеры, из которых составлены эти регистры, обозначены на

*) Alexander I., Array networks for a parallel adder and its control, «IEEE Trans. Electron. Comput.», 1967, v. 16, No 2, стр. 226—229.

рис. 2-22 квадратами с буквой «Т» внутри. По команде «выдача суммы», вырабатываемой устройством управления, цифры суммы через специальный ряд вентилей (элементов «и») выдаются в регистр *В,* где заменяют цифры одного из слагаемых. Между вентилями и входами триггеров включены линии задержки (Л. 3.). Длительность задержки выбрана такой, что после импульса «выдача суммы» триггеры регистра *В* можно еще установить в «0» командой «гашение», и только через некоторое время после этого на входы триггеров поступят импульсы установки в «1» с линий (в тех разрядах, где сумма равна 1). Передачу суммы в регистр *В* здесь, как и на рис. 2-22, *б,* можно устроить и иначе, так чтобы триггеры не должны были при этом переключаться дважды (на «0» и на «1») и чтобы длительность задержки могла бы быть меньше длительности одного такта срабатывания триггера; но эти технические подробности нас сейчас не интересуют. На рисунке не показаны также цепи ввода информации в регистры и другие устройства, которые не относятся прямо к рассматриваемому вопросу.

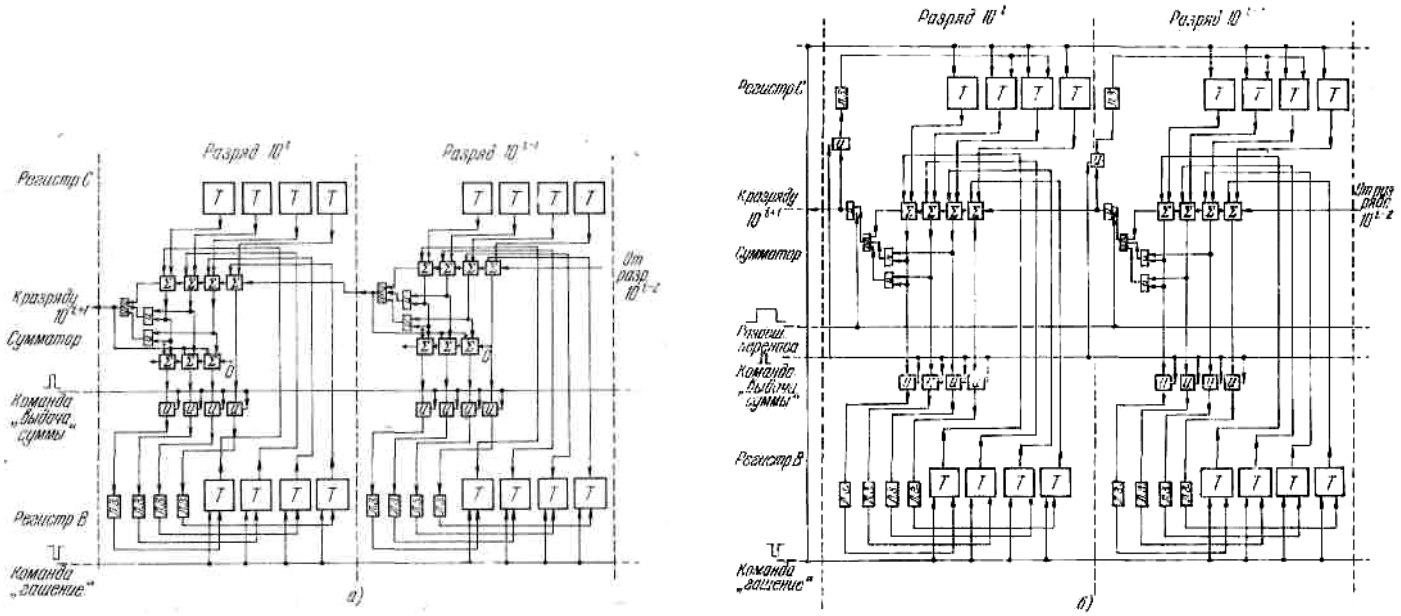


Рис. 2-22. Параллельный десятичный сумматор с числовыми регистрами: *а)* однотактный сумматор; б) двухтактный сумматор

Так как за все время суммирования (от приема слагаемых в регистры до выдачи суммы) логические элементы сумматора срабатывают только по одному разу (напряжения на их входах не изменяются), сумматор рис. 2-22, *а* называется однотактным.

На рис. 2-22,б показаны те же два регистра — *В* и *С* — и сумматор, но сумматор здесь двухтактный.

Внешне схема отличается главным образом тем, что в каждом из десятичных разрядов отсутствуют нижние ряды одноразрядных двоичных сумматоров, которые в схеме рис. 2-22,*а* служат для добавления корректирующей поправки. Оставшиеся верхние ряды сначала производят в каждой тетраде сложение цифр слагаемых по правилам двоичной арифметики, а потом — коррекцию результата.

В первом такте в регистрах *В* и *С* находятся исходные слагаемые. Складывая внутри каждой тетрады изображения десятичных цифр по правилам двоичной арифметики, сумматор в то же время формирует переносы из тетрады в тетраду по правилам десятичной арифметики. Цепь образования переносов здесь такая же, как на рис. 2-22, *а*; в каждом разряде имеется дополнительный элемент «и», который позволяет прервать образование сигналов переноса, подав запрещающий уровень напряжения на шину «разрешение переноса». Однако в течение первого такта схема управления удерживает на этой шине разрешающее напряжение.

Затем, точно так же как в схеме рис. 2-22, *а,* производится выдача суммы в регистр *В* (но при этом выдается Некорректированная сумма). Одновременно в регистр *С* выдаются корректирующие поправки, где они заменяют другое слагаемое. Для этого в каждом десятичном разряде имеется дополнительный элемент «и», который пропускает импульс «выдача суммы» в том случае, если из данного десятичного разряда имеется перенос в следующий десятичный разряд, и устанавливает в «1» 2-й и 3-й триггеры данной тетрады (т. е. ставит в ней число 6).

Во втором такте некорректированная сумма, находящаяся в регистре *В,* складывается с корректирующими поправками из регистра *С.* Цепь десятичных переносов при этом должна быть закрыта (запрещающий уровень на шине «разрешение переноса»). Правильная сумма затем прежним путем передается в регистр *В.* Одновременно с гашением *В* вновь устанавливается в «0» регистр С; но теперь дополнительные элементы «и» ни в одном из разрядов не дают сигналов (так как десятичные переносы отсутствуют), и в регистр *С* ничего больше не принимается.

Сумматор на рис. 2-22,*б* требует почти вдвое меньше оборудования, чем сумматор на рис. 2-22,*а*. Но и времени для выполнения сложения требуется примерно вдвое больше, так как и логические элементы в сумматоре

должны сработать по два раза и дважды должен произойти прием в регистр *B*. Впрочем, временные соотношения сильно зависят от того, какую роль играют различные компоненты в общем времени выполнения сложения. Например, если основную роль играет время установления сигналов десятичного переноса, то второй такт может быть существенно короче, чем первый, так как во втором такте десятичные переносы отсутствуют.

Рис. 2-23 иллюстрирует другой принцип построения многотактного десятичного сумматора. На этом рисунке представлен последовательный десятичный сумматор для кода с излишком 3. При этом предполагается, что не только десятичные цифры поступают в сумматор последовательно, но и все двоичные разряды, которые составляют изображение одной десятичной цифры, поступают последовательно, один за другим, по одному и тому же проводу, с периодом тактовой частоты τ.

Если бы двоичные цифры, которые относятся к одному десятичному разряду, поступали в сумматор одновременно (по 4 разным проводам), то полная схема сумматора должна была бы состоять из одноразрядного десятичного сумматора, выполненного, например, в соответствии с рис. 2-17, и элемента задержки на 1 такт — для запоминания цифры переноса в следующий десятичный разряд (см. рис. 2-2). Такой сумматор мы называли бы однотактным, так как суммирование одной пары десятичных цифр выполнялось бы в нем в течение одного такта. При этом, как видно из рис. 2-17, схема содержала бы 8 одноразрядных двоичных сумматоров.

В схеме рис. 2-23 имеется всего 2 одноразрядных двоичных сумматора — в четыре раза меньше. Зато и суммирование выполняется вчетверо медленнее, занимая по 4 такта на каждый десятичный разряд.

Работает схема рис. 2-23 следующим образом.

Цифры слагаемых (в виде импульсных сигналов) поступают последовательно на входы *b* и *c* верхнего одноразрядного двоичного сумматора. Вместе с линией задержки Л.З.-1 на 1 такт он образует последовательный двоичный сумматор, который производит сложение двоичных изображений слагаемых по правилам двоичной арифметики. Так как мы пользуемся кодом с излишком 3, то сигнал двоичного переноса, образующийся на выходе *E* верхнего сумматора при сложении старших двоичных цифр тетрады, является сигналом переноса в следующий десятичный разряд. Указанные моменты времени отмечаются импульсами «конец тетрады», которые поступают из устройства управления с частотой, равной $^1/_4$ частоты тактов.

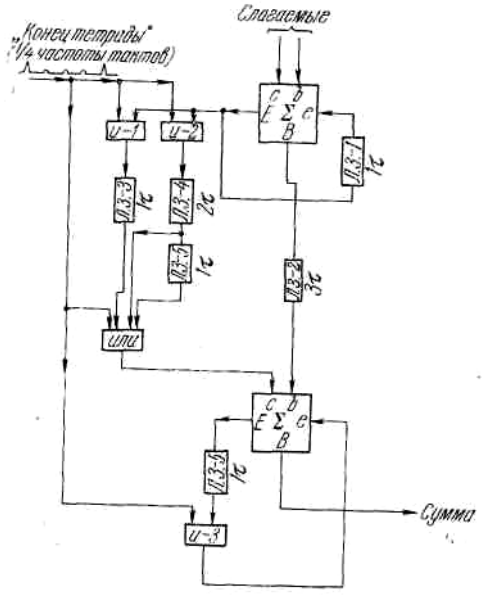


Рис. 2-23. Последовательный 4-тактный десятичный сумматор для кода с излишком 3.

В зависимости от того, есть ли перенос в следующий десятичный разряд, импульс «конец тетрады» проходит через элемент «и-1», либо через элемент «и-2» (в элементе «и-1» сигнал переноса р а з р е ш а е т прохождение импульса «конец тетрады», в элементе «и-2» он, наоборот, поступает на вход з а п р е т а, отмеченный на рисунке точкой). Кроме того, импульс «конец тетрады» через элемент «или» поступает на вход с нижнего двоичного сумматора. На другой его вход (*b*) поступает некорректированная сумма с выхода верхнего сумматора; так как она при этом проходит через линию задержки Л.З.-2 (на 3 такта), то момент поступления импульса «конец тетрады» на вход *c* совпадает с моментом прихода импульса м л а д ш е г о двоичного разряда тетрады суммы на вход *b*. Линии задержки (Л.З.-3, Л.З.-4 и Л.З.-5) расположены так, что если импульс «конец тетрады» прошел через элемент «и-1», то на входе с нижнего сумматора появится также импульс одновременно со в т о р ы м двоичным разрядом тетрады, поступающей на вход *b*, а в третьем и четвертом тактах на входе *c* никаких сигналов не будет; если же импульс «конец тетрады» прошел через элемент «и-2», то во втором такте на вход *c* нижнего сумматора ничего не поступает, а в третьем и четвертом тактах поступают импульсы.

Таким образом, на вход *c* нижнего сумматора поступает либо код 0011 (где младшие разряды, идущие по времени раньше, записаны справа), т. е. число 3, либо код 1101, т. е. число 13. Складывая эту корректирующую поправку с некорректированной суммой, которая подается на вход *b*, нижний сумматор формирует правильную сумму. При этом переносы из тетрады в тетраду, которые могли бы возникнуть в нижнем сумматоре, блокируются в элементе «и-3» импульсом «конец тетрады», который поступает на вход запрета как раз в тот момент времени, когда на входы *b* и *c* поступают младшие двоичные разряды десятичного разряда.

Суммирование одной пары десятичных цифр в схеме рис. 2-23 занимает, очевидно, в общей сложности 7 тактов — от момента поступления младших двоичных цифр слагаемых на входы верхнего сумматора до получения старшей двоичной цифры суммы на выходе нижнего сумматора. Однако мы все же называем сумматор 4-тактным потому что в течение последних трех тактов суммирования данной пары десятичных цифр, когда в нижнем сумматоре заканчивается добавление корректирующей поправки к некорректированной сумме, в верхнем сумматоре уже идет суммирование двоичных цифр следующего десятичного разряда.

Для схемы рис. 2-22 *(б)* было характерно:

параллельное сложение десятичных разрядов, параллельное сложение двоичных цифр внутри десятичных разрядов,

последовательное сложение сначала слагаемых между собой, затем некорректированной суммы с корректирующей поправкой;

в схеме рис. 2-23 имеем:

последовательное сложение десятичных разрядов, последовательное сложение двоичных цифр внутри десятичных разрядов,

параллельное сложение слагаемых между собой и некорректированной суммы с корректирующей поправкой.

В то же время, например, схема рис. 2-22,*а* дает:

параллельное сложение десятичных разрядов,

параллельное сложение двоичных цифр внутри десятичных разрядов,

параллельное сложение слагаемых между собой и некорректированной суммы с корректирующей поправкой.

Всего же возможно, очевидно, 2^3=8 различных комбинаций этих трех условий. Комбинация, обеспечивающая наибольшую скорость и требующая наибольшего количества оборудования — это схема рис. 2-22, *а* (все операции — параллельно). Наименьшего количества оборудования и наибольшего времени для выполнения сложения потребовал бы сумматор, в котором все операции выполнялись бы последовательно; в его состав мог бы входить всего один одноразрядный двоичный сумматор; зато и суммирование десятичных чисел занимало бы в среднем по 8 тактов на каждый десятичный разряд. В любом случае сокращение количества оборудования в многотактном сумматоре достигается за счет увеличения количества тактов суммирования.

Нужно учесть, что во многих случаях увеличение количества тактов суммирования не пропорционально сокращению количества оборудования в сумматоре.

В следующих разделах для сравнения различных вариантов схемы выполнения одной и той же операции мы будем использовать оценки по *относительной эффективности Q* разных вариантов, которая представляет собой отношение произведения количества оборудования на время выполнения операции в одном из рассматриваемых вариантов (принятом за эталон для сравнения) к аналогичному произведению для другого варианта (оцениваемого). Применение многотактных схем для выполнения десятичного суммирования ведет большей частью к снижению эффективности.

Например, при переходе от схемы рис. 2-22, *а* к схеме рис. 2-22, *б* количество тактов суммирования увеличилось вдвое, а количество одноразрядных двоичных сумматоров уменьшилось лишь в отношении 7 : 4. Что касается вспомогательного оборудования (линии задержки, элементы «и»), то его в схеме рис. 2-22, *б* даже больше, чем в схеме рис. 2-22, *а.* Проигрыш по эффективности здесь получается потому, что во втором такте суммирования оборудование сумматора в схеме рис. 2-22, *б* недоиспользуется (им выполняются более простые операции, чем оно может выполнить). Правда, по этой причине, как уже говорилось, второй такт суммирования мог бы оказаться по времени короче первого, так что суммарная потеря времени может получиться меньше чем вдвое, а потеря в эффективности — меньше, чем кажется с первого взгляда.

Однако при проектировании электронных вычислительных машин далеко не всегда ставится задача получения наиболее эффективных (в указанном смысле) схем. Иногда требуется получить быстродействие любой ценой, иногда — максимально возможное сокращение оборудования, иногда — выполнение каких-нибудь других условий. Любое проектирование представляет собой поиск некоторых компромиссных решений; и если требуется компромисс между скоростью выполнения десятичного сложения и экономичностью устройства, многотактные схемы десятичных сумматоров могут оказаться подходящим решением.

2.3.4. Сумматоры с переменным основанием системы счисления.

Принципы, использованные нами при построении десятичных сумматоров, можно применить и для создания сумматоров, работающих в любой другой системе счисления. Иногда, однако, приходится решать и более сложную задачу — построение сумматоров, работающих в произвольной системе счисления (в том числе, может быть, и смешанной), заданной программным путем. Такие сумматоры мы будем называть сумматорами с переменным основанием.

Впервые такая задача возникла, по-видимому, в Англии при создании машин для коммерческих расчетов, которые должны вести вычисления и в десятичной системе и в английской денежной системе. Сумматоры с переменным основанием могут оказаться полезными и при использовании в машине системы представления чисел в остатках (см. 1.3.4) и, возможно, в каких-нибудь других случаях.

Построение одноразрядного комбинационного сумматора с переменным основанием приведено на рис. 2-24. Полная схема сумматора строится далее из одноразрядных сумматоров указанного типа обычным путем.

Хотя основание системы счисления заранее неизвестно, тем не менее предполагается, что числа всегда изображаются позиционным способом с естественными весами разрядов и что цифрам системы счисления с основанием n поставлены в соответствие числа 0, 1, 2 (n — 1). Для определенности в схеме рис. 2-24 принято также, что все возможные основания системы счисления n не больше 16 ($n \leq 16$), а цифры 0, 1, 2,..., (n — 1) изображаются четверками двоичных разрядов; при этом внутри каждой тетрады для изображения *n-ичных* цифр двоичным кодом также применяется позиционный способ с естественными весами двоичных разрядов и с цифрами 0,1.

Если $n > 16$, но раскладывается на множители ($n = = n_1\ n_2 ... n_к$), каждый из которых не больше 16, то разряд по основанию n может быть представлен в виде нескольких разрядов с основаниями n_1, n_2,..., n_k и суммирование можно будет выполнять тем же сумматором. Например, разряд по основанию 21 (1 гинея=21 шиллингу) может быть представлен в виде одного семеричного и одного троичного разряда. Впрочем, принцип, использованный при построении схемы рис. 2-24, легко применить также для построения сумматоров, которые могли бы работать и с большими основаниями системы счисления.

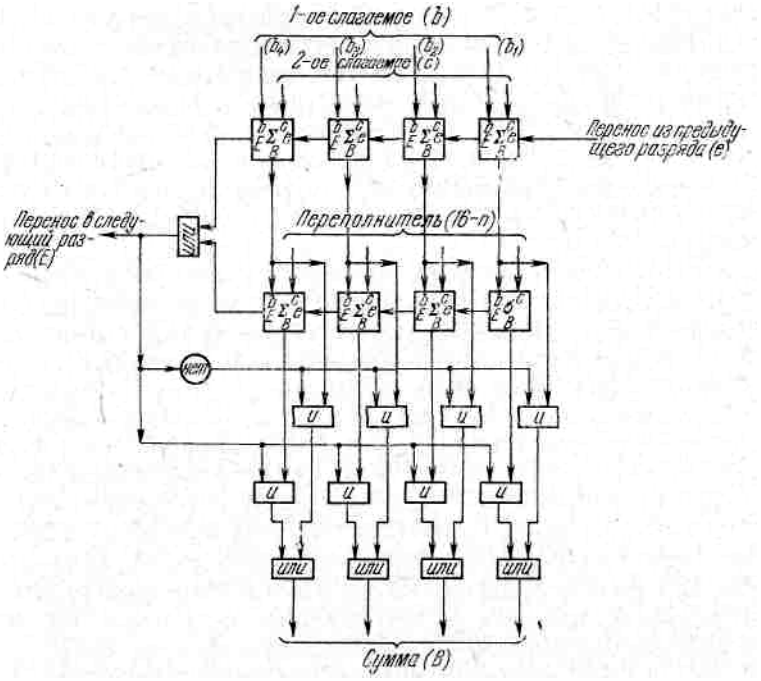


Рис. 2-24. Одноразрядный сумматор комбинационного типа с переменным основанием n ($n \leq 16$).

Для выполнения сложения в сумматор рис. 2-24 наряду с цифрами слагаемых подается цифра еще одного числа. Это специальное, третье, число называется *переполнителем.* Каждая тетрада переполнителя содержит величину (16 — *n),* где n — то основание системы счисления, по которому должно выполняться суммирование в данном разряде. Если мы пользуемся однородной системой счисления, то все тетрады переполнителя содержат одну и ту же величину (например, для десятичной системы 0110, т. е. 6); для смешанных систем разные тетрады переполнителя содержат разные коды. Если все тетрады переполнителя содержат нули, то сумматор работает в обычной двоичной системе счисления (формально — в двоично-кодированной шестнадцатеричной системе). В общем случае переполнители заносятся в отведенные для них ячейки памяти при составлении программы или образуются программным путем; перед выполнением сложения из памяти (или, может быть, из регистров арифметического устройства) извлекаются и слагаемые, и переполнитель, соответствующий той системе счисления, в которой эти слагаемые записаны. Результат суммирования всегда будет получаться в той же системе, в которой представлены слагаемые.

Верхний ряд одноразрядных двоичных сумматоров на рис. 2-24 производит суммирование по правилам двоичной арифметики двух n-ичных цифр слагаемых (b и c) и цифры переноса из предыдущей тетрады ($e = 0$ или $E = 1$). При этом суммирование выполняется по модулю 16, так что при $b + c + e < 16$ на выходах B сумматоров верхнего ряда получаем величину $b + c + e$, при $b + c + e \geq \geq 16$ — величину $b + c + e$—16. В последнем случае образуется перенос в следующую тетраду: так как $n \leq 16$, то из $b + c + e \geq 16$ вытекает непременно $b + c + e \geq n$.

Нижний ряд одноразрядных двоичных сумматоров складывает величину, полученную на верхних сумматорах ($b + c + e$ или, может быть, $b + c + e$ — 16) с величиной 16 — n.

Если с верхнего ряда поступает величина $b + c + e$ — — 16, то на выходах B нижних сумматоров получаем

$$(b + c + e — 16) + (16 — n) = b + c + e — n,$$

что является правильной цифрой суммы для данного n-ичного разряда (с учетом того, что в следующую тетраду перенесена единица, — что следует из уравнений для B и E в 2.1.1). Эта величина наверняка меньше чем 16: максимальные значения для b и c есть n — 1, максимальное значение e — это 1, поэтому $b + c + e — n \leq (n — 1) + (n — 1) + 1 — n = n — 1 < 16$; следовательно, на выходе E старшего одноразрядного сумматора нижнего ряда сигнал переноса в этом случае появиться не может. Если с верхнего ряда сумматоров передается величина $b + c + e$ (т. е. если оказалось, что $b + c + e<16$, и на верхнем ряду сумматоров перенос в следующую тетраду не возник), то при суммировании с величиной (16 — n) могут быть два случая:

— если $b + c + e \geq n$, то $(b + c + e) + (16 — n) \geq \geq 16$; при этом необходимость переноса единицы в следующую тетраду обнаруживается по наличию единицы на выходе E старшего двоичного сумматора нижнего ряда; на выходах B сумматоров нижнего ряда образуется величина $b + c + e — n$ (потому что суммирование здесь тоже выполняется по модулю 16), соответствующая искомой цифре суммы для данного n-ичного разряда;

— если $b + c + e < n$, то $(b+ c + e) + (16 — n)<16$; на выходе старшего двоичного сумматора нижнего ряда, как и на выходе старшего двоичного сумматора верхнего ряда, имеем «0»; в следующую тетраду единица переноситься не должна, а цифра суммы в данном n-ичном разряде равна $b + c + e$.

Окончательно правило суммирования, по которому построена рассматриваемая схема, может быть сформулировано следующим образом:

если на выходе E старшего одноразрядного двоичного сумматора верхнего ряда или на аналогичном выходе нижнего ряда появляется сигнал «1», то должен быть перенос в следующую тетраду; если на обоих указанных выходах получаются нули, то перенос в следующую тетраду отсутствует; при наличии переноса в следующую тетраду n-ичная цифра суммы в данном разряде должна быть прочитана с выходов B сумматоров нижнего ряда, при отсутствии переноса — с выходов B верхнего ряда.

Сравнивая схему рис. 2-24 со схемой, например, рис. 2-16, можно видеть, что сумматор с переменным основанием получается немногим сложнее, чем обычный десятичный сумматор. Это объясняется тем, что в схеме рис. 2-16 возможности входящего в нее оборудования используются неполностью, здесь же, если n произвольно, на входах всех двоичных сумматоров могут быть все возможные комбинации входных сигналов.

Известны и другие варианты построения сумматоров **с** переменным основанием[*]).

2.4. Особенности параллельных сумматоров

До сих пор мы рассматривали построение одноразрядных сумматоров безотносительно к тому, каким способом эти одноразрядные сумматоры объединяются в полную схему сумматора. Между тем при создании параллельного сумматора имеется ряд специфических затруднений, приводящих, в частности, к особым требованиям к построению применяемых одноразрядных сумматоров. Их мы рассмотрим в настоящем разделе.

2.4.1. Требования к построению цепи переносов. Оптимальные схемы.

Параллельный сумматор для сложения двух m-разрядных чисел содержит, грубо говоря, в m раз больше оборудования, чем последовательный сумматор. Естественно, что от него желательно получить и выигрыш по скорости примерно в m раз.

Во многих случаях исходные числа на входы сумматора поступают из триггерных регистров. Если минимальное время выполнения сдвига в регистре обозначить через τ_c, то сложение двух m-разрядных чисел с помощью последовательного сумматора, присоединенного к двум таким регистрам, будет выполняться за время $m\bullet\ \tau_c$. Применяя с теми же регистрами параллельный сумматор, желательно получить время выполнения суммирования порядка τ_c.

Но и в параллельном сумматоре образование сигналов переноса выполняется последовательно, разряд за разрядом: цифру переноса из данного разряда в следующий нельзя получить прежде, чем известен перенос из предыдущего (младшего) разряда в данный разряд. Время образования в одноразрядном сумматоре выходного сигнала переноса E, отсчитываемое от момента подачи на его вход сигнала переноса из предыдущего разряда e (сигналы цифр слагаемых могут быть поданы заранее), называется временем задержки переноса и обозначается τ_E. Чтобы суммирование m-разрядных чисел можно было выполнить за время порядка τ_c, задержка переноса τ_E должна быть порядка τ_c/m.

Фактически отношение τ_c: $\tau_E = k$ показывает, сколько разрядов имеет смысл суммировать параллельно. Если $k \approx 1$, то, очевидно, нужно строить последовательный сумматор. Если $k \gtrsim m$, то оправдано построение параллельного сумматора. Вообще наименьшее возможное время суммирования равно примерно $m\tau_E = \frac{m}{k}\tau_c$[**]).

При промежуточных значениях отношения τ_c: $\tau_E = k$ минимум оборудования, при котором может быть обеспечена такая скорость, достигается при использовании последовательно-параллельного сумматора, в котором в каждом такте в параллель суммируется примерно по k разрядов.

При создании параллельного сумматора наиболее трудной задачей является обеспечение достаточной скорости выполнения переноса, т. е. получения достаточной малой величины задержки переноса x_E. С этой целью применяются самые разнообразные средства: использование в цепях переносов наиболее быстродействующих логических элементов (может быть, более сложных, чем в других цепях), специальная логика построения этих схем, тщательно продуманный монтаж, исключающий длинные соединения и большие паразитные емкости, и т. д.

С точки зрения логики построения главное и наиболее очевидное требование состоит в том, что между входом сигнала переноса в данный разряд из соседнего младшего разряда (e) и выходом сигнала переноса в следующий разряд (E) должно быть возможно меньше промежуточных ступеней.

В частности, для двоичного сумматора, построенного из элементов «и», «или» и «нет», оптимальным является логическое построение, приведенное на рис. 2-25, или двойственное ему. В приведенной схеме сигнал переноса проходит через две ступени («и»-«или»); на другие входы этих ступеней подаются переключательные функции D и

---

[*]) См. T o w n s e n d R . , Serial Digital Adders for a Variable Radix of Notation, Automat. Digital Comp., London, 1954, стр. 120—124, Discuss., стр. 124.

[**]) Если не применять сверхпараллельные и параллельно-параллельные схемы (см. 2.5.2, 2.5.3).

R, зависящие только от цифр слагаемых данного разряда (*b* и *c*), но не от переноса в данный разряд. Эти функции можно сформировать одновременно по всем разрядам сумматора — сразу же вслед за тем, как в регистры приняты слагаемые. Следовательно, время образования функций *D* и *R* входит в полное время суммирования один раз, а не умножается на количество разрядов *m,* как время задержки переноса в каждом разряде τ_E.

Возможность построения участка формирования сигналов переноса двоичного одноразрядного сумматора в виде, показанном на рис. 2-25, вытекает непосредственно из преобразования канонической формы функции *E*

$$E = \bar{b}ce + b\bar{c}e + bc\bar{e} + bce$$

к виду

$$E = (\bar{b}c + b\bar{c})e + bc,$$

откуда

$$R=\bar{b}c+b\bar{c};\ D=bc$$

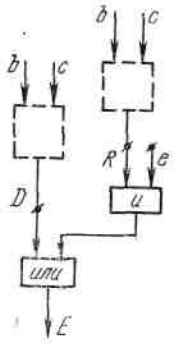


Рис. 2-25. Оптимальное построение участка формирования сигналов переноса для одноразрядного двоичного сумматора, использующего элементы «и»,«или» и «нет».

Можно получить и другие выражения для *R* и *D ,* которые, возможно, более удобны для построения схем в тех или иных случаях. Если, например, в схеме рис. 2-8 (стр. 66) поменять местами входы *c* и *e,* то окажется, что участок переносов в ней построен в соответствии с рис. 2-25; но функция *R* формируется при этом по уравнению*)

$$R = b + c$$

Как говорилось выше, участок переноса в схеме рис. 2-8 построен (как, впрочем, и вся схема одноразрядного сумматора) наиболее экономным образом из всех возможных построений, использующих элементы «и», «или», нет».

Также в соответствии с рис. 2-25 выполнен участок переносов в схеме одноразрядного сумматора, приведенный на рис. 2-10 (стр. 67). Однако в этой схеме *R* формируется по уравнению

$$R = (b + c)\,\bar{b}c$$

(тождественному приведенной ранее форме $R = \bar{b}c + b\bar{c}$).

Построение участка двоичных переносов в виде, представленном на рис. 2-25, при использовании элементов «и», «или» и «нет» является оптимальным по многим причинам.

Прежде всего в этом построении имеются всего 2 ступени между входом переноса *e* и выходом *E.* Нетрудно убедиться, что меньшим количество промежуточных ступеней не может быть: если бы между входом *e* и выходом *E* была всего одна ступень «нет», либо «и», либо «или», то это означало бы соответственно, либо что *E* является инверсией от *e,* либо что при *e* = 0 непременно должно быть *E* = 0, либо что при *e* = 1 обязательно должно быть *E* = 1; ни одно из этих соотношений не имеет места.

В двухступенной схеме каждая ступень могла бы в принципе состоять из нескольких логических элементов. Например, в построении рис. 2-4 на стр. 64 первая ступень схемы образования переноса (*E*) состоит из четырех элементов «и», каждый из которых имеет вход сигнала *e* или $\bar{e}$. В построении рис. 2-25 каждая ступень содержит минимальное количество элементов, на которые поступают сигналы *e* или функции, зависящие от *e* (по одному такому элементу в каждой ступени). Если цепи, определяющие скорость распространения переносов, предполагается выполнить из более быстродействующих (и, возможно, более дорогих) элементов, чем другие цепи, то построение рис. 2-25 является оптимальным с точки зрения необходимого количества быстродействующих элементов.

Выгодно в этом построении и то, что каждый из элементов в быстродействующей части схемы образования переносов имеет минимально возможное число входов (два), а по своему выходу нагружен на минимальное количество входов других элементов (каждый элемент соединен всего с одним последующим: элемент «и» — со входом элемента «или», элемент «или» — со входом «и» следующего разряда). Так как быстродействие элементов во многих случаях зависит от количества входов (а иногда быстродействующий элемент «и» или «или» просто не может иметь большого числа входов) и так как увеличение нагрузки на выходы элементов тоже может привести к снижению быстродействия, построение рис. 2-25 является оптимальным и с этой точки зрения.

Некоторые дополнительные возможности схемы рис. 2-25 рассматриваются ниже (см. 2.5).

Стремясь получить минимальное количество ступеней между входом сигнала переноса *e* и выходом *E* в каждом разряде параллельного сумматора, нельзя забывать и о том, что в полной схеме сумматора друг за другом включено большое количество разрядов и что поэтому в каждом разряде выходной сигнал переноса должен мало отличаться от входного по амплитуде (или перепаду напряжений), длительности фронтов и другим физическим параметрам. В связи с этим желательно, чтобы одна из логических ступеней в схеме образования переноса в

*) Разумеется, эта функция *R* не тождественна приведенной выше функции *R* = *be* 4- *be;* однако, преобразуя выражение для искомой функции *E* иначе, чем мы делали прежде, легко убедиться, что функция *b* -j-*c* тоже может быть использована в схеме рис. 2-25 на входе *R*.

каждом разряде могла бы заодно выполнять функции усиления и формирования сигналов переноса. Если это не так (скажем, если все элементы «и» и «или» выполнены из диодов,) то в каждом разряде или, может быть, в каждой группе, включающей несколько разрядов, в цепь переносов необходимо вводить специальные ступени усилителей-формирователей.

Построения, аналогичные рис. 2-25, могут быть найдены не только для двоичных сумматоров, но и для сумматоров, работающих в других системах счисления. На рис. 2-26 показан десятичный одноразрядный сумматор такого рода для кода «8, 4, 2, 1». Схема эта во многом похожа на схему рис. 2-16 (стр. 73), но участок формирования сигнала переноса в следующий разряд построен в ней иначе.

В схеме рис. 2-16 сигнал переноса из предыдущего десятичного разряда поступал на вход первого (правого двоичного сумматора верхнего ряда. На первый взгляд кажется, что по времени наиболее трудным для этой схемы случаем является тот, когда сумма десятичных цифр слагаемых в данном разряде равна 9 (без учета переноса из предыдущего разряда): если сумма цифр слагаемых больше девяти, то перенос в следующий десятичный разряд может образоваться еще до того, как получен сигнал переноса из предыдущего разряда.

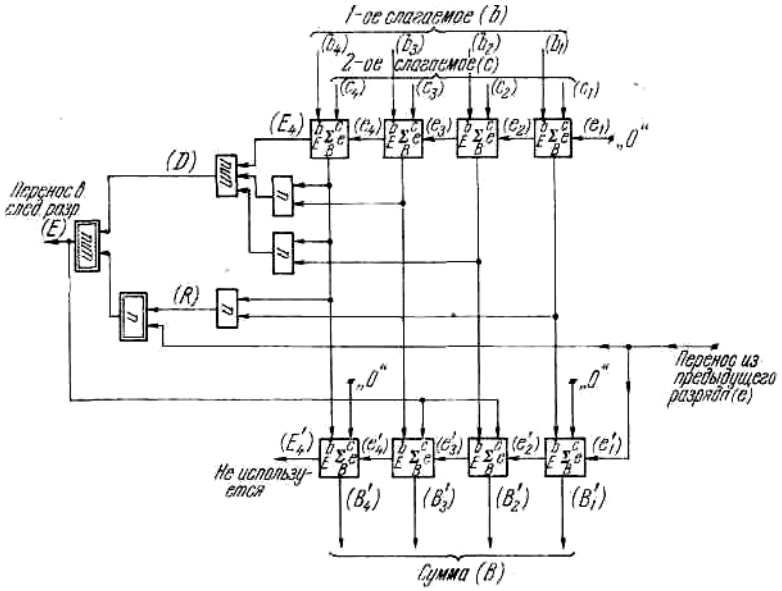


Рис 2-26. Десятичный одноразрядный сумматор для кода «8, 4, 2,1», в котором участок переноса построен по аналогии со схемой рис. 2-25.

На самом деле, наиболее трудным может оказаться случай, когда сумма десятичных цифр слагаемых равна 15. Представим себе, что схема рис. 2-16 выполнена с потенциальными связями. Если сигнал переноса из предыдущего десятичного разряда запаздывает по отношению к сигналам слагаемых, то на выходах верхнего ряда сумматоров сначала устанавливается комбинация «1111» (т. е. 15); при этом, как и должно быть, вырабатывается сигнал переноса в следующийдесятичный разряд (потому что $B_4B_2 = 1$ и $B_4B_3 = 1$). Однако, когда из предыдущего десятичного разряда приходит сигнал переноса (e), на выходах двоичных сумматоров верхнего ряда B_1, B_2, B_3, B_4 последовательно устанавливаются нули. При этом сигнал переноса в следующий десятичный разряд на некоторое время исчезает; затем он вновь появляется, когда на выходе E старшего двоичного сумматора верхнего ряда устанавливается единица. Таким образом, окончательное установление сигнала переноса в следующий десятичный разряд в схеме рис. 2-16 требует прохождения сигнала переноса из предыдущего разряда через 4 двоичных одноразрядных сумматора верхнего ряда и еще дополнительно через схему «или»; если в каждом двоичном разряде сигнал переноса проходит через 2 ступени, то всего на десятичный разряд приходится 9 ступеней.

В схеме рис. 2-26 перенос из предыдущего десятичного разряда поступает не на верхний, а на нижний ряд двоичных одноразрядных сумматоров. Для этого к нижнему ряду добавлен справа один сумматор (или, может быть, полусумматор). С точки зрения получения правильной десятичной цифры суммы *(B)* такое изменение не играет никакой роли. Зато с точки зрения скорости переноса оно дает существенный выигрыш.

Так как сигнал переноса на верхний ряд двоичных сумматоров не поступает, то группа логических элементов, формирующих функции D и R (см. рисунок), срабатывает сразу вслед за получением сигналов слагаемых, одновременно по всем десятичным разрядам. Эти логические элементы устроены так, что при $b + c \geq 10$ получается $D = 1$, при $b + c = 9$, $D = 0$, $R = 1$, при $b + c \leq 8$ $D = 0$, $R = 0$. В первом случае *(*$D = 1$) перенос в следующий разряд равен единице, в третьем — (D = 0, $R = 0$) — нулю, независимо от того есть ли перенос в данный разряд; в обоих этих случаях нужный сигнал на выходе E устанавливается еще до того, как поступил сигнал переноса из предыдущего разряда *(e)*. Во втором случае *(*$D = 0$, $R = 1$, т. е. $b + c = 9$) перенос в следующий десятичный разряд определяется наличием переноса из предыдущего десятичного разряда (e). При этом между входом e и выходом E имеются всего две логические ступени — элементы «и» и «или», обведенные на рисунке двойными рамками; от их быстродействия и зависит задержка переноса T_E на один десятичный разряд. По

сравнению со схемой рис. 2-16 количество промежуточных ступеней в цепи переносов здесь в $4\frac{1}{2}$ раза меньше.

Если основную часть времени суммирования составляет время распространения сигналов переноса, то по быстродействию такой десятичный сумматор примерно в 3,32 раза лучше, чем двоичный при той же точности вычислений: задержка переноса на каждый десятичный разряд получается такой же, как на один двоичный разряд, а количество десятичных разрядов при равной точности в $\log_2 10 = 3{,}32...$ раза меньше, чем количество двоичных разрядов. Интересно, что и по относительной эффективности Q десятичный сумматор может оказаться лучше, чем двоичный сумматор. Количество оборудования в одном десятичном разряде сумматора примерно в 8 раз больше, чем в одном разряде двоичного сумматора (возможность заменить два крайних двоичных сумматора нижнего ряда полусумматорами примерно компенсирует наличие цепей десятичного переноса). Так как количество десятичных разрядов в $\log_2 10$ раз меньше, чем двоичных, то всего оборудования в десятичном сумматоре примерно в $8/\log_2 10$ раз больше, чем в двоичном. Приняв произведение количества оборудования на время суммирования в двоичном сумматоре за 1, найдем то же произведение для десятичного сумматора: $\frac{8}{\log_2 10} \cdot \frac{1}{\log_2 10}$; отсюда относительная эффективность десятичного сумматора по отношению к двоичному равна

$$Q = 1 : \frac{8}{\log_2^2 10} \approx 1{,}37.$$

2.4.2. О применении синхронных элементов.

Создание параллельных сумматоров из синхронных элементов, т. е. из элементов, которые могут опрашиваться или переключаться лишь в течение определенной части периода тактовых сигналов, представляет значительные трудности. К этому типу элементов, относятся динамические триггеры с линиями задержки, параметроны, феррито-диодные и феррито-транзисторные элементы и др. Основная трудность здесь связана с конечностью времени распространения сигналов переноса. Но теперь уже дело не только в том, что задержки сигналов переноса не позволяют реализовать высокие скорости, но и в том, что вообще трудно построить работоспособную схему.

Представим себе, например, что регистры, из которых слагаемые должны поступать в параллельный сумматор, построены из динамических триггеров с линиями задержки.

Если все триггеры питаются от одного и того же источника сигналов синхронизации, то все они одновременно вырабатывают свои выходные сигналы («0» или «1») в виде импульсов длительностью в $^1/_2$ такта; в течение другой половины такта никаких сигналов нет. В младшем разряде сумматора из сигналов слагаемых формируется сигнал переноса в следующий разряд. Этот сигнал поступает во второй разряд с некоторой задержкой относительно сигналов слагаемых; так как во втором разряде он участвует в формировании сигналов переноса в следующий разряд, то сигнал переноса в третий разряд поступает с еще большей задержкой относительно сигналов слагаемых и т. д. Задержки эти в каждом разряде, может быть, и невелики по сравнению с длительностью такта (иначе вообще не стоило бы строить параллельный сумматор), но для далеких разрядов сигнал переноса может прийти уже в тот момент, когда сигналы слагаемых закончились. При этом нельзя будет получить ни правильной цифры суммы, ни •правильной цифры переноса в следующий разряд.

Эту трудность можно преодолеть, сдвинув фазы сигналов синхронизации для различных разрядов регистров или включив линии задержки разной длины между выходами триггеров регистров и входами сумматора. Но тогда важно, чтобы сигнал переноса в некоторый разряд поступал всегда с вполне определенной задержкой относительно начала суммирования (в определенной фазе относительно тактовой частоты). В рассмотренных выше схемах сумматоров это условие не всегда соблюдается. Скажем, при построении участка двоичного переноса по типу рис. 2-25 сигнал переноса из данного разряда выдается либо сразу вслед за поступлением цифр слагаемых (если $b = 1$ и $c = 1$ и, следовательно, $D = 1$), либо только после получения сигнала переноса из предыдущих разрядов (если $b = 1$ и c — 0 либо $b = 0$ и $c = 1$, т. е. если $R = 1$). Правда, фазы сигналов синхронизации для триггеров или линии задержки подбираются так, чтобы в каждом разряде цифры слагаемых поступали примерно в тот же момент времени, когда может прийти сигнал переноса из предыдущего разряда; следовательно, по обоим каналам перенос формируется примерно в одно и то же время. Но для определенности лучше в схемы одноразрядных сумматоров ввести еще и сигналы синхронизации — тоже в разных фазах для разных разрядов.

Более простой путь при построении параллельного сумматора с регистрами из динамических триггеров — это использование на входах цепей переноса потенциальных выходов триггеров.

Феррито-транзисторные и феррито-диодные ячейки тоже используются большей частью как элементы синхронные: выходные сигналы в них образуются в момент подачи опросного импульса и имеют вид сравнительно коротких импульсов (по длительности, как правило, не больше половины периода импульсов опроса). Если бы регистры, связанные с параллельным сумматором, были построены из феррито-транзисторных или феррито-диодных элементов, опрашиваемых в один и тот же момент времени, то при этом возникли бы те же

трудности, что и при построении регистров из работающих синфазно динамических триггеров с линией задержки. Но в отличие от динамических триггеров эти элементы не обязательно опрашивать периодически. Последним обстоятельством можно воспользоваться следующим образом.

Представим себе, что цепь переносов выполнена как парафазная схема. Это значит, что в каждый разряд сигнал переноса поступает по двум проводам: на одном из них импульс присутствует, когда в данный разряд из предыдущего переносится единица ($e = 1$), на другом — когда в данный разряд переносится нуль ($e = 0$). Схему сумматора можно построить так, чтобы в каждом разряде сигнал $E = 0$ или $E = 1$ вырабатывался только при условии, что на входы данного разряда поступил либо сигнал $e = 0$, либо сигнал $e = 1$; пока предыдущий разряд не сработал, т. е. пока нет ни импульса $e = 0$, ни импульса $e = 1$, в данном разряде никакие сигналы не вырабатываются. Такое построение сумматора для случая использования двоичной системы счисления (один из возможных вариантов) иллюстрируется рис. 2-27.

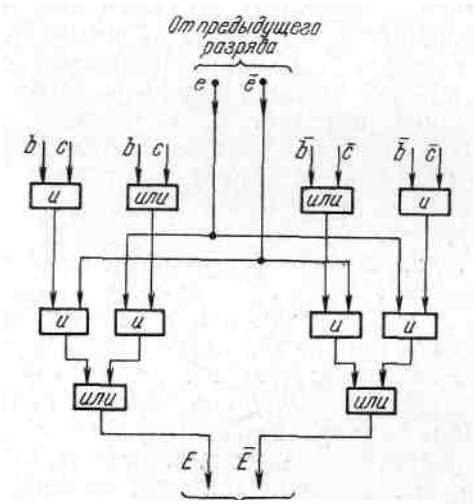


Рис. 2-27. Участок образования сигналов двоичного переноса, построенный как двухпроводная схема

Выполнение переносов начинается тогда, когда на один из входов e или $\bar{e}$ младшего разряда поступает импульс от схемы управления. При обычном арифметическом суммировании импульс должен поступать, конечно, на вход $\bar{e}$ (перенос в самый младший разряд есть нуль); однако в различных специальных случаях использования сумматора в младший разряд иногда подается сигнал $e = 1$ (импульс на входе e).

Регистры, связанные с сумматором такого типа, могут быть выполнены, скажем, из феррито-транзисторныхячеек. Для опроса ячеек первого разряда удобно использовать импульс, поступающий от схемы управления на вход e или на вход $\bar{e}$ данного разряда сумматора. Тогда сигналы на всех входах первого разряда сумматора (b, c и $\bar{e}$ или e) окажутся одновременными. С некоторым запаздыванием по времени первый разряд вырабатывает один из сигналов $\bar{E}$ или E, которые для второго разряда являются входными сигналами $\bar{e}$ или e. Если эти импульсы использовать для опроса ячеек второго разряда регистров, то и во втором разряде сумматора импульсы на всех входах окажутся одновременными. Выходные сигналы $\bar{E}$ или E второго разряда сумматора далее используются для опроса ячеек третьего разряда регистров и т. д.

Таким образом, при выполнении суммирования каждый разряд регистров опрашивается с некоторым запаздыванием относительно предыдущего разряда, причем запаздывания в точности соответствуют естественным задержкам сигналов переноса в сумматоре. Необходимости специально генерировать многофазную систему опросных импульсов нет.

При выполнении других элементарных операций, когда смещения по времени не требуется, все разряды регистров можно опрашивать одновременно.

2.5. Методы ускорения сложения в параллельных сумматорах

2.5.1. Ускорение суммирования в асинхронных устройствах.

Двухпроводная схема участка образования сигналов переноса (рис. 2-27), применение которой при использовании синхронных элементов обсуждалось в 2.4.2, заимствована нами фактически из одной из ранних работ по ускорению суммирования в а с и н х р о н н ы х устройствах[*]).

Для выполнения переносов при суммировании двух m-разрядных чисел параллельным сумматором обычно отводится время порядка $(1{,}1 \div 1{,}2)\, m\tau_E$, где τ_E —задержка переноса в одном разряде сумматора. Коэффициент $1{,}1 \div \div 1{,}2$ берется для создания запаса на случай изменения величины τ_E из-за ухода питающих напряжений или характеристик элементов схемы, а также на случай отклонений в работе элемента задержки, определяющего указанный интервал времени.

Поскольку в двухпроводной схеме участка переносов рис. 2-27 появление сигнала на одном из выходов (E или $\bar{E}$) самого старшего разряда может служить сигналом об окончании переноса, применение этой схемы позволяет отводить на выполнение переносов ровно столько времени,, сколько оно действительно занимает. Ясно, что выигрыш по скорости, который можно получить таким образом (порядка 10—20%), не очень значителен.

Более значительный выигрыш по быстродействию можно получить, используя статистическую структуру переносов. С этой целью двухпроводная схема цепи образования переносов перестраивается так, чтобы всегда, когда это возможно, сигналы наличия или отсутствия переноса из некоторого разряда в следующий формировались бы до того, как появятся соответствующие сигналы из предыдущего разряда. Одно из возможных построений участка переносов одноразрядного двоичного сумматора, удовлетворяющее этому требованию,

[*]) См. G i l c h r i s t B. и др., Fast carry logic for digital computers, IRE Trans. Electronic Comp, 1955 4, 4.

приведено на рис. 2-28. Если в данном разряде слагаемых имеются две единицы ($b = 1$ и $c = 1$), то перенос в следующий разряд равен единице ($E = 1$) независимо от того, имеется ли перенос в данный разряд; аналогичным образом при $b = 0$ и $c = 0$ получим $E = 0$ независимо от наличия переноса в данный разряд. Только тогда, когда $b \neq c$, наличие переноса в следующий разряд определяется переносом в данный разряд.

В самом трудном случае, когда во всех разрядах складываемых чисел (кроме, может быть, самого младшего разряда) имеем $b \neq c$, выполнение переноса занимает время $m\tau_E$ (m — по-прежнему количество разрядов слагаемых, T_E — задержка переноса на один разряд). Однако, если все возможные пары слагаемых равновероятны, вероятность такого случая очень мала. Можно подсчитать*), что при сложении 40-разрядных двоичных чисел максимальная длина цепочки разрядов, в которой перенос из данного разряда определяется наличием или отсутствием переноса в него из предыдущего разряда, в среднем равна 5,6 разряда. Вероятность появления таких цепочек длиной более 12 разрядов ничтожно мала.

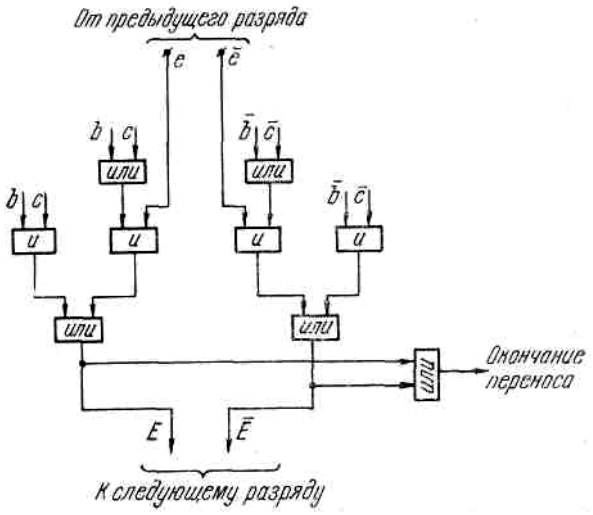


Рис. 2-28. Возможное построение участка переноса для схемы двоичного сумматора, учитывающей статистическую структуру переносов.

Таким образом, выполнение переносов при сложении 40-разрядных двоичных чисел занимает в среднем примерно в 7 раз меньше времени, чем максимально возможное время выполнения этой операции. В схеме рис, 2-28.появление сигнала на выходе E или на выходе $\overline{E}$ некоторого разряда свидетельствует о том, что формирование сигнала переноса в данном разряде закончено; если сигналы окончания переносов от всех m разрядов ввести в общий элемент «и» (имеющий m входов), то его выходной сигнал будет являться сигналом окончания переносов по всему сумматору. Таким образом, на выполнение переносов можно будет отводить ровно столько времени, сколько требуется в каждом конкретном случае, а не каждый раз максимальное время, как делается обычно.

Разумеется, этот метод ускорения сложения может быть использован в полной мере лишь тогда, когда схема арифметического устройства построена из асинхронных элементов, т. е. когда следующая (после выполнения переносов) элементарная операция может начинаться в произвольный момент времени.

2.5.2. Сверхпараллельные сумматоры.

При использовании синхронных элементов эффективным методом ускорения сложения является применение сверхпараллельных сумматоров. В равной мере эти схемы дают значительный эффект и при использовании асинхронных элементов. Главный недостаток сверхпараллельных схем — большое количество оборудования, необходимого для их осуществления.

Идея сверхпараллельного сумматора состоит в том, чтобы выполнить максимальное количество подготовительных операций в старших разрядах сумматора до получения сигналов переноса из младших разрядов. Имея в старших разрядах заранее сформированные «подготовительные функции» (зависящие только от цифр слагаемых, но не от цифр переноса в данные разряды), можно затем ускорить прохождение через эти разряды сигналов переноса.

Эта идея частично уже знакома нам по схеме цепи переносов, приведенной на рис. 2-25.

В схеме рис. 2-25 в каждом разряде формируются подготовительные функции D и R, зависящие только от цифр слагаемых данного разряда. Функция D равна единице в том случае, когда перенос из данного разряда есть единица независимо от того, имеется ли перенос в данный разряд; если функция R равна единице, то это означает, что перенос из данного разряда в следующий должен получиться при условии, что будет перенос в данный разряд из предыдущего. Наличие заготовленных заранее функций D и R позволило нам сформировать сигнал переноса E из данного разряда в следующий всего в две ступени по уравнению

$$E = D + Re,$$

где e — сигнал переноса в данный разряд. Функции D и R мы будем называть *подготовительными функциями нулевого порядка.*

*) См. сноску на стр. 85.

Развивая эту идею дальше, поступим следующим образом*). Подготовительные функции нулевого порядка D и R формируются по всем разрядам одновременно — вслед за приемом слагаемых в регистры. В младших разрядах они сразу же используются для образования сигналов переноса. Но пока сигнал переноса пройдет через несколько младших разрядов, в следующих разрядах из функций D и R можно сформировать *подготовительные функции первого порядка.*

С этой целью все разряды, старше некоторого s-го разряда, разбиваются на группы по p разрядов (о выборе величин s и p мы скажем несколько позже). Пусть имеем группу из p последовательных разрядов — от номера $k + 1$ (младший разряд этой группы) до номера $k + p$. Перенос из $(k + p)$-го разряда в следующий можно записать в виде

$$E_{k+p} = D_{k+p} + R_{k+p}\, e_{k+p},$$

где D_{k+p}, R_{k+p} — подготовительные функции нулевого порядка, сформированные в данном разряде, e_{k+p} — перенос в данный разряд. Но входным сигналом переноса для данного разряда является выходной сигнал переноса предыдущего разряда; поэтому

$$e_{k+p} = E_{k+p-1} = D_{k+p-1} + R_{k+p-1} e_{k+p-1}.$$

Подставляя это выражение e_{k+P} в уравнение для E_{k+p}, найдем

$$E_{k+p} = (D_{k+p} + R_{k+p} D_{k+p-1}) + (R_{k+p} R_{k+p-1})\, e_{k+p-1}.$$

Действуя таким же образом дальше, найдем

$$E_{k+p} = (D_{k+p} + R_{k+p} D_{k+p-1} + R_{k+p} R_{k+p-1} D_{k+p-2} + \ldots + R_{k+p} R_{k+p-1} \ldots R_{k+2} D_{k+1}) + (R_{k+p} R_{k+p-1} R_{k+p-2} \ldots R_{k+1})\, e_{k+1}.$$

Введем теперь обозначения

$$D^{(1)}_{k+1,\,k+p} = D_{k+p} + R_{k+p} D_{k+p-1} + R_{k+p} R_{k+p-1} D_{k+p-2} + \ldots R_{k+p} R_{k+p-1} \ldots R_{k+2} D_{k+1},$$

$$R^{(1)}_{k+1,\,k+p} = R_{k+p} R_{k+p-1} \ldots R_{k+2} R_{k+1}.$$

Функции $D^{(1)}_{k+1,\,k+p}$ и $R^{(1)}_{k+1,\,k+p}$ называются подготовительными функциями первого порядка для группы разрядов от $(k + 1)$-го до $(k + p)$-го. Они зависят только от подготовительных функций нулевого порядка, т. е. в конечном

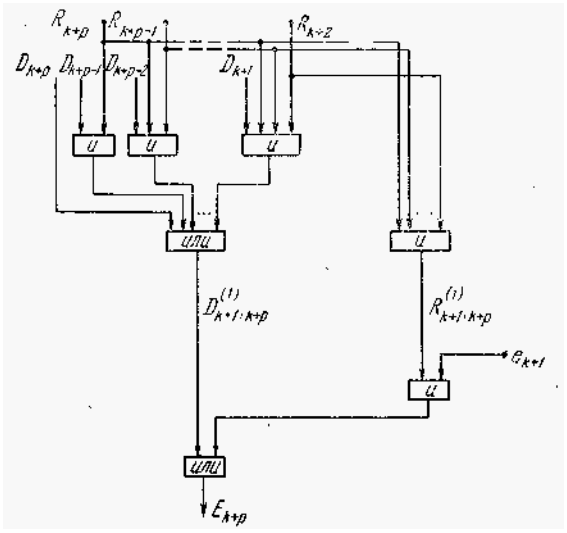


Рис. 2-29. Формирование подготовительных функций первого порядка
$D^{(1)}$ и $R^{(1)}$ для группы разрядов и сигнала переноса E_{k+p} из старшего разряда группы.

счете только от цифр слагаемых в этих разрядах, но не от цифр переноса в данную группу разрядов из предыдущих разрядов. Функция $D^{(1)}$ для данной группы разрядов равна единице, если перенос из старшего разряда группы в следующий равен единице независимо от того, имеется ли перенос в младший разряд группы; функция $R^{(1)}$ равна единице, если наличие переноса из старшего разряда группы определяется наличием переноса в ее младший разряд.

В соответствии с приведенными уравнениями, схемы формирования функций $D^{(1)}_{k+1,k+p}$ и $R^{(1)}_{k+1,\,k+p}$ должны быть такими, как показано на рис. 2-29. На этом же рисунке показано, как по уравнению

$$E_{k+p} + D^{(1)}_{k+1,k+p} + R^{(1)}_{k+1,k+p} e_{k+1}$$

из сигнала e_{k+1} получается сигнал E_{k+P}.

Таким образом, если в данной группе из p разрядов сформированы подготовительные функции первого

*) См. A High-Speed Parallel Adder for Electronic Digital Computers. Techn. News Bull. NBS, 1956, 40, 6, а также W e i n b e r g e r A., S m i t h I. L., A One-microsecond Adder Using One-megacicle Circuitry, IRE Trans. Electronic Comp., 1956.

порядка, то между входным сигналом переноса в младший разряд данной группы e_{k+1} и выходным сигналом переноса из старшего разряда группы E_{k+P} оказывается всего 2 ступени. Если бы подготовительные функции первого порядка не были сформированы заранее, то таких ступеней было бы по 2 на разряд, т. е. всего 2p; наличие этих функций как бы ускоряет процесс распространения сигналов переноса в p раз.

Количество разрядов p, объединяемых в одну группу, определяется тем, какое количество входов допустимо для логических элементов. Как видно из рис. 2-29, в схеме, формирования $R^{(1)}$необходим элемент «и» на p входов, а в схеме формирования $D^{(1)}$ — элементы «и», имеющие до p входов, и элемент «или» на p входов. При использовании обычных логических элементов p может быть порядка 4÷6.

Количество разрядов s, на которое имеет смысл отступить от начала (от младшего разряда) с тем, чтобы начать объединять разряды в группы и формировать для групп подготовительные функции первого порядка, зависит от соотношения между скоростью распространения сигналов переноса и скоростью образования функций $D^{(1)}$ и $R^{(1)}$. Если бы элементы «и» и «или» на p входов срабатывали за то же время, что и элементы «и» и «или» на 2 входа, то следовало бы брать $s = 1$: от момента образования во всех разрядах функций D и R нулевого порядка до появления сигнала переноса на выходе первого разряда проходит время, необходимое на срабатывание двух логических ступеней «и»-«или»; столько же времени требовалось бы для образования функций $D^{(1)}$ и $R^{(1)}$ из D и R в группах последующих разрядов. Практически же при использовании асинхронных элементов целесообразно выбирать $s = 4 \div 6$.

При использовании синхронных элементов — скажем, динамических триггеров с линиями задержки — величина s зависит от того, сколько последовательных разрядов цепи переносов могут сработать от одной фазы синхронизации. В той же фазе синхронизирующие сигналы подаются на схемы формирования функций $D^{(1)}$ и $R^{(1)}$, а логические схемы, стоящие вслед за ними (элементы «и» — «или», формирующие переносы через группы из p разрядов), работают уже от следующей фазы синхронизации.

Через s групп, начиная от первой (т. е. всего через $s + sp$- разрядов от младшего), имеет смысл начать объединять группы разрядов в группы и для каждой такой группы групп формировать *подготовительные функции второго порядка* $D^{(2)}$ и $R^{(2)}$ Пока сигнал переноса проходит через s младших групп разрядов, в старших группах будут сформированы подготовительные функции второго порядка, которые затем еще раз ускорят процесс распространения сигналов переноса.

Подготовительные функции второго порядка формируются из подготовительных функций первого порядка точно так же (по тем же уравнениям и такими же схемами), как подготовительные функции первого порядка формируются из подготовительных функций нулевого порядка. Поэтому и количество г р у п п , которое можно объединять в одну группу для формирования подготовительных функций второго порядка, равно количеству р а з р я дов, которые объединяются в одну группу для формирования подготовительных функций первого порядка *(p)*. Точно так же, как подготовительные функции первого порядка позволяют сформировать в две ступени перенос через группу из p разрядов, подготовительные функции второго порядка позволяют в две ступени сформировать перенос через группу из p групп разрядов (т. е. всего через p^2 разрядов); схема для этого в точности аналогична схеме рис. 2-29. Таким образом, когда сигнал переноса, распространяясь от младших разрядов, достигает тех разрядов, где сформированы подготовительные функции второго порядка, скорость его распространения снова повышается в p раз и становится в p^2 раз выше, чем в самых младших разрядах (где имелись лишь подготовительные функции нулевого порядка).

Отступив затем еще на s групп, объединяющих группы по p разрядов, можно было бы начать группировать группы групп и формировать подготовительные функции третьего порядка и т. д. Однако при обычных соотношениях между величинами s, p и количеством разрядов в складываемых числах необходимости в подготовительных функциях третьего порядка уже не возникает.

Полное построение сверхпараллельного сумматора (младшие разряды) иллюстрируется рис. 2-30. Для сокращения на рисунке принято $s = 1$, $p = 3$. Поэтому вне групп находится только один разряд (первый); группы, по которым формируются $D^{(1)}$ и $R^{(1)}$, объединяют разряды 2-й ÷ 4-й, 5-й ÷ 7-й, 8-й ÷ 10-й, 11-й ÷ 13-й и т. д.; первая группа групп, для которой формируются $D^{(2)}$ и $R^{(2)}$, объединяет группы разрядов 5÷7, 8÷10 и 11÷13, следующая такая группа групп на рисунке уже не поместилась.

Заметим, что во всех разрядах, объединенных в группы, кроме старшего разряда каждой группы, сохранены обычные цепи переноса, построенные в соответствии с рис. 2-25; например, такие цепи имеются во втором и третьем разрядах. Хотя эти цепи не используются для образования сигнала переноса из старшего разряда группы (в данном случае e_5), который формируется более коротким путем, они необходимы для формирования промежуточных сигналов переноса внутри группы *(*e_3, e_4), которые участвуют в образовании цифр суммы в соответствующих разрядах. Точно так же если некоторая группа разрядов входит в группу групп, но не является старшей в ней (например, на рисунке группы 5÷7 и 8÷10), то в ней сохраняются цепи для ускоренного формирования сигнала переноса из старшего разряда группы (e_8 и e_{11}). Хотя эти сигналы не используются для формирования выходного сигнала переноса из группы групп (в данном случае e_{14}), они необходимы для образования промежуточных переносов (e_9, e_{10} и e_{12}, e_{13}).

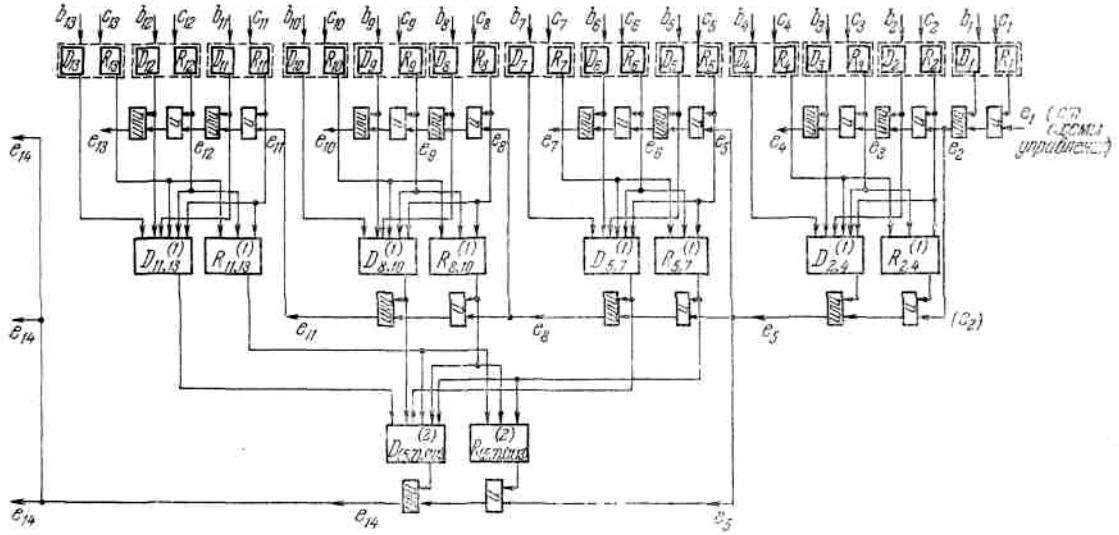


Рис. 2-30. Построение сверхпараллельного сумматора.

При использовании асинхронных элементов наличием указанных цепей удобно воспользоваться для существенного сокращения количества оборудования. Такой сокращенный вариант сверхпараллельного сумматора показан на рис. 2-31. От рассмотренного ранее полного сверхпараллельного сумматора он отличается тем, что в схемах формирования переноса из старшего разряда группы входом элемента «или» является не функция $D^{(1)}$, а сигнал с выхода обычной схемы переносов данного разряда; точно так же при формировании выходного сигнала переноса групп групп разрядов входом элемента «или» является вместо функции $D^{(2)}$ выходной сигнал переноса старшей группы разрядов. Например, при формировании сигнала e_5 на вход оконечного элемента «или» вместо функции $D^{(1)}_{2,4}$ подан сигнал e'_5, представляющий собой выходной сигнал переноса 4-го разряда, сформированный в обычной схеме сумматора (на основе подготовительных функций нулевого порядка).

По смыслу e_5' и e_5 — это одни и те же сигналы. Но в том случае, когда наличие или отсутствие переноса из старшего разряда группы (e_5) определяется тем, имеется ли перенос в ее младший разряд (e_2), т. е. когда $R^{(1)}_{2,4}= 1$, сигнал e_5 формируется быстрее, чем e'_5; скорость распространения сигналов переноса в этом случае получается такой же, как в полной схеме сверхпараллельного сумматора.

Некоторая потеря в быстродействии по сравнению с полной схемой получается в том случае, когда сигнал переноса должен зародиться внутри данной группы разрядов, т. е. когда имели бы $D^{(1)} = 1$. В полной схеме на образование сигнала переноса при этом было бы потрачено время, необходимое для образования функции $D^{(1)}_{2,4}$, т. е. примерно время задержки переноса в одном разряде или несколько больше; в сокращенной схеме на образование e_5' может уйти время, равное сумме задержек переноса по всем разрядам данной группы (с 2-го по 4-й). Таким образом, цепь переносов как бы удлиняется в этом случае на p — 1 разрядов. Однако если и в других группах разрядов имеется аналогичное положение, то дополнительное время на них уже не тратится.

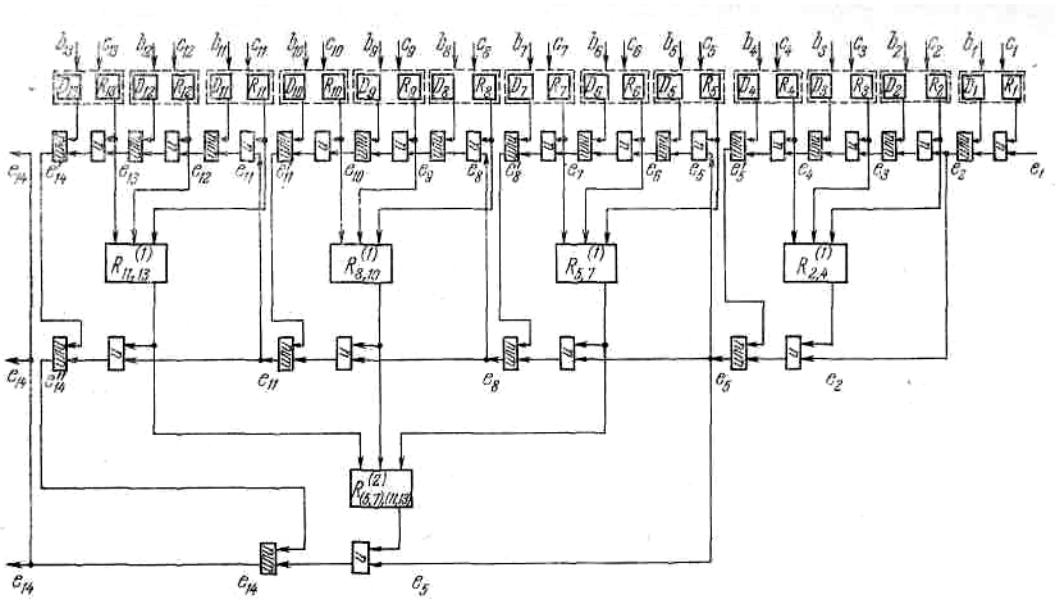


Рис. 2-31. Сокращенный вариант построения сверхпараллельного сумматора.

Добавочная потеря времени такого же порядка возможна только за счет отсутствия цепей формирования $D^{(2)}$ и т. д. Таким образом, если в сверхпараллельном сумматоре используются подготовительные функции до i-го порядка включительно, то потеря времени в сокращенной схеме эквивалентна удлинению цепи переносов на $i\ (p - 1)$ разрядов. Поскольку обычно $i = 1 \div 2$, а $p = 4 \div 6$, эта потеря в быстродействии не очень значительна. Зато сокращение количества оборудования очень велико: как видно из рис. 2-29, именно формирование функций D требует наибольшего оборудования, а функции R, используемые в сокращенной схеме, можно получить сравнительно просто.

Заметим здесь еще, что, хотя в литературе схемы сверхпараллельных сумматоров обсуждаются обычно лишь применительно к двоичной системе счисления, все приведенные построения можно в равной мере применить и, скажем, в десятичном сумматоре, если каждый его разряд построен в соответствии с рис. 2-25, т. е. если в каждом разряде формируются подготовительные функции нулевого порядка D и R (или только R для сокращенной схемы).

2.5.3. Параллельно-параллельные сумматоры.

Внешне к схемам сверхпараллельных сумматоров весьма близки так называемые параллельно-параллельные сумматоры*) (удачна ли эта терминология, мы здесь не обсуждаем). Однако, как мы увидим из дальнейшего, по асимптотическим оценкам сверхпараллельные и параллельно-параллельные схемы различаются между собой весьма значительно.

Параллельно-параллельная схема может быть использована тогда, когда применяются логические элементы с большим числом входов и большой нагрузочной способностью. Например, если бы элементы «и» и «или» обладали этим свойством, то сигналы переноса e_i для всех m разрядов сумматора можно было бы формировать по следующим уравнениям:

$$
\begin{aligned}
e_2 &= D_1 + R_1 e_1, \\
e_3 &= D_2 + R_2 D_1 + R_2 R e_1 \\
e_4 &= D_3 + R_3 D_2 + R_3 R_2 D_1 + R_3 R_2 R_1 e_1, \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
e_m &= D_{m-1} + R_{m-1} D_{m-2} + R_{m-1} R_{m-2}\, D_{m-3} + \ldots \\
&\ldots + R_{m-1} R_{m-2} \ldots R_2 D_1 + R_{m-1} R_{m-2} \ldots R_1 e_1.
\end{aligned}
$$

где R_i, D_i — подготовительные функции нулевого порядка, сформированные в i-м разряде, e_1 — сигнал переноса в младший разряд, поступающий от схемы управления или по цепи кругового переноса (в обычном суммировании $e_1 = 0$), m — количество разрядов в суммируемых числах. Уравнения эти получаются таким же путем, как на стр. 205 нами было получено выражение для переноса из старшего разряда группы $E_{k+p} = e_{k+p+1}$ через подготовительные функции R и D и перенос в младший разряд группы e_{k+1}.

Если бы цепь переносов была построена в соответствии с этими уравнениями, то формирование сигналов переноса по всем разрядам происходило бы одновременно и занимало столько времени, сколько требуется для срабатывания двух логических ступеней «и»-«или». Однако для этого необходимы элементы «и» и элементы «или», имеющие по m входов (см. уравнения для e_m), а выходные элементы в схемах формирования R и D должны выдерживать нагрузку в примерно $(\frac{m}{2})^2$ входов элементов «и» (функция R_1 используется на входах 1 (m—1) элементов «и», функция R_2 — в 2 (m — 2) элементах, функция R_3 — в 3 (m — 3) элементах,..., функция R_{m-1} — в (m — 1)·1 элементах; максимальная нагрузка приходится на средние разряды).

Ясно, что при использовании обычных диодных логических элементов и при обычном количестве двоичных разрядов m. создание таких схем мало реально (хотя, скажем, в десятичной системе, где количество разрядов m порядка 6÷8 является во многих случаях достаточным, построение таких схем не вызывало бы особых затруднений).

Принцип параллельно-параллельного сумматора в действительности был предложен впервые применительно к логическим элементам «ни» («NOR»), каждый из которых, как проектировалось, может иметь до 20÷25 входов и работать на 20÷25 других аналогичных элементов.

Если применяются элементы «ни» («не или», т. е. осуществляющие операцию $\overline{x_1 + x_2 + \ldots + x_k}$), то вместо записанных выше уравнений для $e_2, e_3, \ldots, e_m$ удобнее воспользоваться следующими уравнениями:

*) См. Rowe W. D., A New Approach to High-Speed Logic. Proc. West. Joint Comput. Conf., San Francisco, Calif., 1959, New York, 1959, стр. 277—283.

$$\bar{e}_2 = \overline{D_1 + \overline{(\overline{R_1} + \bar{e}_1)}},$$

$$\bar{e}_3 = \overline{D_2 + \overline{(\overline{R_2} + \overline{D}_1)} + \overline{(\overline{R_2} + \overline{R_1} + \bar{e}_1)}},$$

$$\bar{e}_4 = \overline{D_3 + \overline{(\overline{R_3} + \overline{D}_2)} + \overline{(\overline{R_3} + \overline{R_2} + \overline{D}_1)} + \overline{(\overline{R_3} + \overline{R_2} + \overline{R_1} + \bar{e}_1)}},$$

. . . .

$$\bar{e}_m = \overline{D_m + \overline{(\overline{R_{m-1}} + \overline{D_{m-2}})} + \ldots + \overline{(\overline{R_{m-1}} + \overline{R_{m-2}} + \ldots + \overline{R_2} + \overline{D}_1)} + \overline{(\overline{R_{m-1}} + \ldots + \overline{R_2} + \bar{e}_1)}}.$$

Сигналы $\bar{e}_2$, $\bar{e}_3, \ldots, \bar{e}_m$, которые при этом будут получаться вместо e_2, $e_3, \ldots, e_m$, можно с тем же успехом использовать для образования цифр суммы в каждом разряде (что вытекает из принципа взаимности). В записанные уравнения вместо D_i и D_i удобнее подставить непосредственно их выражение через цифры слагаемых bi и c_i

$$D_i = b_i c_i = \overline{\bar{b}_i + \bar{c}_i},$$
$$\overline{D}_i = \qquad \bar{b}_i + \bar{c}_i.$$

С учетом этой подстановки построение параллельно-параллельного сумматора получится таким, как показано на рис. 2-32.

При всем внешнем сходстве между сверхпараллельными и параллельно-параллельными схемами имеется, как мы говорили выше, принципиальное отличие.

Если количество разрядов m в слагаемых очень велико, то наивысший порядок подготовительных функций, формируемых в сверхпараллельной схеме, равен примерно $k \approx \log_p m$, где p — количество элементов (разрядов, групп), объединяемых в одну группу. При этом приходится формировать m подготовительных функций

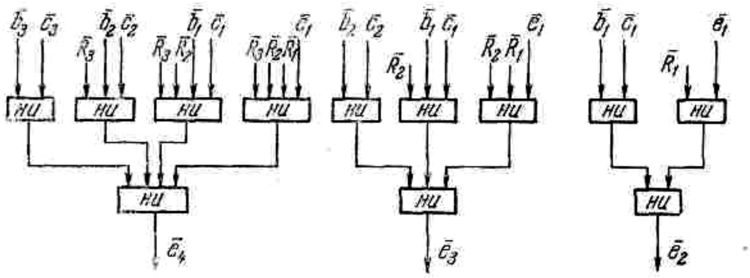


Рис. 2-32. Построение параллельно-параллельного сумматора из элементов «ни» (младшие разряды).

нулевого порядка, примерно m/p подготовительных функций первого порядка, m/p^2 подготовительных функций второгопорядка,..., $m/p^k \approx 1$ подготовительных функций k-го порядка. Полное количество оборудования примерно пропорционально величине

$$m + \frac{m}{p} + \frac{m}{p^2} + \ldots + 1 = m\frac{mp-1}{m(p-1)} \approx m\frac{p}{p-1}.$$

При этом время образования переносов примерно пропорционально количеству ступеней в схеме формирования подготовительных функций наивысшего порядка, т. е. примерно пропорционально $k = \log_p m$.

Таким образом, при большом количестве разрядов m количество оборудования в сверхпараллельном сумматоре растет как первая степень m, а время выполнения суммирования — как $\log m$.

С другой стороны, легко видеть, что в параллельно-параллельном сумматоре при большом количестве разрядов m количество оборудования*) растет как m^2, а время выполнения суммирования не зависит от m. По своим асимптотическим характеристикам сверхпараллельная схема оказывается ближе к обычному параллельному сумматору с учетом статистической структуры переносов (см. 2.5.1), чем к параллельно-параллельной схеме.

2.5.4. Схемы с «мгновенным» переносом.

Схемы с «мгновенным» переносом представляют собой одно интересное техническое решение построения цепей двоичного переноса параллельного сумматора. При этом речь идет не о новых логических приемах построения

*) Мы подсчитываем здесь общее количество логических элементов, а не суммарное количество входов в них.

цепи переносов, а именно об оригинальном техническом решении, основанном на известных логических принципах.

Рассматриваемые ниже схемы с равным успехом могут применяться и в тех случаях, когда регистры, связанные с сумматором, выполнены из асинхронных элементов и когда применяются синхронные элементы. Однако в последних случаях их преимущества более очевидны.

Построение схемы*) иллюстрируется рис. 2-33, где изображено 3 разряда цепи переносов: *(k* — 1)-й (младший из трех), *k*-й и *(k* + 1)-й; величины напряжений проставлены, конечно, только для ориентировки.

Как видно из рисунка, каждый разряд схемы содержит 3 триода. Триоды управляются выходными напряжениями диодных или каких-нибудь других логических схем. Эти схемы построены так, что если выполняется заданное условие (например — для логической схемы, управляющей триодом T_3,— если $b \neq c$), то логическая схема вырабатывает. низкое напряжение (на рис. — 5,5 *в*); если заданное условие не выполняется, то напряжение на выходе логической схемы высокое (на рис. —2,5 *в).* Наличие переноса в какой-либо разряд изображается, как показано на рисунке, напряжением — *3 в,* а отсутствие переноса — напряжением —4,5 *в*; эти уровни можно выбрать и обратными: важно лишь, чтобы оба они находились внутри перепада напряжений, вырабатываемых логическими схемами.

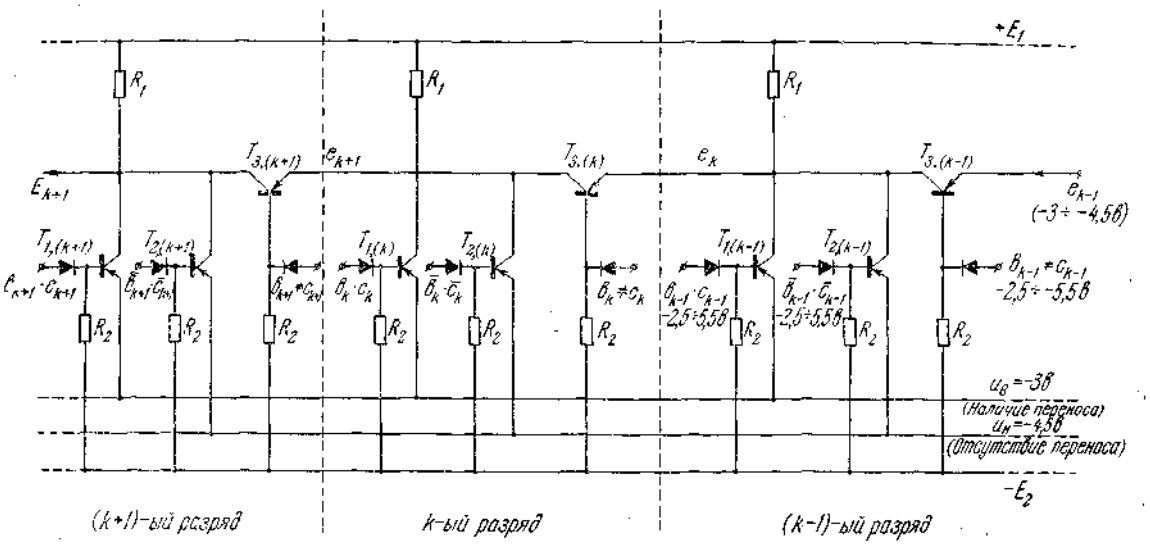


Рис. 2-33. Принцип построения схемы с «мгновенным» переносом.

В общих чертах работа устройства сводится к следующему.

В любой комбинации цифр слагаемых (*b* и с) в некотором разряде низкое напряжение может появиться на выходе лишь одной из трех логических схем. Поскольку это напряжение (—5,5 *в)* ниже, чем низший из возможных уровней напряжения на эмиттерах триодов (—4,5 *в),* то соответствующий триод открыт (насыщен) — независимо от того, каково напряжение на его эмиттере. Два других триода данного разряда заперты, так как высокое напряжение на выходе логической схемы (—2,5 *в)* выше, чем высший из возможных уровней на эмиттерах триодов (—3 *в).*

Напряжение в точке соединения коллекторов примерно равно напряжению на эмиттере насыщенного триода. Если в данном разряде обе цифры слагаемых есть единицы, то открыт триод T_1, и в точке соединения коллекторов напряжение равно —3 *в* (имеется перенос в следующий разряд). Если в данном разряде оба слагаемых содержат нули, то открыт триод T_2, и в точке соединения коллекторов напряжение равно —4,5 в (перенос в следующий разряд отсутствует). Если в данном разряде одна из цифр слагаемых есть единица, а другая нуль, то открыт триод T_3, и напряжение в точке соединения коллекторов равно напряжению в аналогичной точке предыдущего разряда (наличие или отсутствие переноса в следующий разряд определяется тем, имеется ли перенос в данный разряд).

Все логические схемы во всех разрядах устройства срабатывают одновременно — сразу вслед за тем, как в регистры, связанные с сумматором, поступили слагаемые; одновременно по всем разрядам происходит и переключение триодов. Последовательным процессом может быть только распространение волны напряжения от младшего разряда через соединенные последовательно открытые триоды всех остальных разрядов (если комбинация цифр такова, что в каждом разряде $b \neq c$). Однако этот процесс происходит практически со скоростью распространения света.

Схема рис. 2-33 является полной аналогией известной в релейной технике схемы «сумматора Шеннона». Пока в вычислительных машинах применялись электронные лампы, этот принцип нельзя было использовать в полной мере, так как падение напряжения на открытой лампе сравнительно велико, и большое количество ламп нельзя соединить последовательно. Полупроводниковый триод, работающий от запирания до насыщения, по своим свойствам значительно ближе к контактам электромагнитного реле, чем электронная лампа. При правильном

*) См. Ki1burn T. и др., Parallel Addition in Digital Computers: a New Fast Carry Circuit, Proc. IEEE, part B, Sept. 1959, v. 106, No 129.

выборе режимов падение напряжения между коллектором и эмиттером насыщенного германиевого триода может быть порядка 0,03 ÷ 0,05 *в,* так что в схеме рис. 2-33 примерно 10— 12 разрядов могут быть включены последовательно без существенных искажений в перепадах сигналов; через 10÷12 ступеней для восстановления уровней придется ставить специальный усилитель-формирователь.

Рассмотрим теперь схему рис. 2-33 более внимательно. Как мы сейчас увидим, триоды в этой схеме в действительности работают не совсем в обычных режимах.

Представим себе, например, что комбинация цифр слагаемых такова, что в (k — 1)-м разряде триод T_1 открыт, а в *k-м* разряде триод T_3 заперт. При этом оказывается, что в коллектор триода $T_{1,k-1}$ включено только сопротивление R_1 (k — 1)-го разряда, присоединенное к источнику положительного напряжения (токами запертых триодов пренебрегаем). Ток в коллекторной цепи триода $T_{1,\,k-1}$ должен в этом случае протекать в направлении от коллектора к базе, а сам триод превращается фактически в два отдельных открытых диода, включенных навстречу друг другу (рис. 2-34, *а).* База триода $T_{1,\,k-1}$ представляет собой как бы точку соединения катодов трех диодов: двух диодов, эквивалентных транзистору, и диода, включенного между выходом логической схемы и входом транзистора. Этот последний диод, очевидно, заперт. Если пренебречь падениями напряжения на открытых диодах, то напряжение в точке соединения их катодов равно $u_в$ (—3*в*); ток через сопротивление R_2 равен $i_б = \frac{u_в - E_2}{R_2}$, ток коллектора $i_{к\text{I}} = \frac{E_1 - u_в}{R_1}$, а ток эмиттера $i_{э\text{I}} = i_б - i_{к\text{I}}$ (при правильном выборе параметров $i_б > i_{к\text{I}}$ — иначе эмиттерный переход оказался бы запертым). Направление тока коллектора в этом режиме (от коллектора к базе) назовем положительным.

Пусть теперь в *(k* — 1)-м разряде по-прежнему открыт триод T_1, но в *k-м* разряде триод T_3 открыт, а триод T_3 (*k*+ + 1)-го разряда заперт Эквивалентная схема при этом оказывается такой, как показано на рис. 2-34, *б.* Поскольку

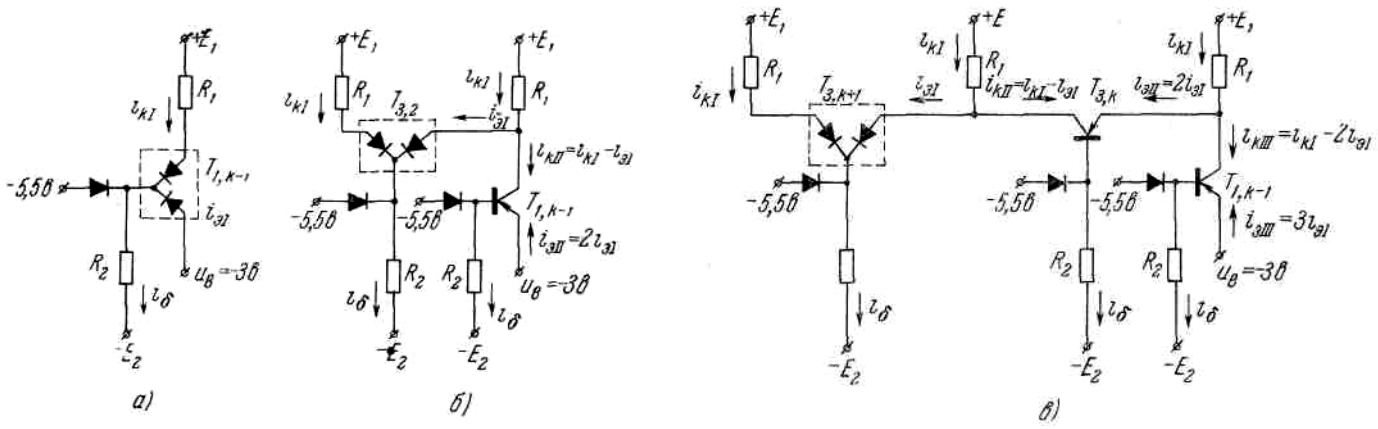


Рис. 2-34. Эквивалентные схемы, поясняющие режимы работы триодов в схеме рис. 2-33: *а)* для случая, когда триод $T_{1,\,k-1}$ открыт, $T_{з,к}$ заперт; б) для случая, когда открыты триоды $T_{1,k-1}$ и $T_{3,k}$, а $T_{3,k+1}$ заперт; в) для случая, когда $T_{1,к-1}$, $T_{3,k}$ и $T_{3,\,k+1}$ открыты, а $T_{3,k+2}$ заперт.

напряжение на эмиттере триода $T_{3,\text{k}}$ равно $u_в$ (—3 *в),* а в коллектор его включено только сопротивление R_1 *k*-го разряда, режим триода $T_{3,K}$ в рассматриваемом случае должен
получиться точно таким же, каким в предыдущем случае был режим триода $T_{1,\,k-1}$. В частности, ток его коллектора равен найденной выше величине $i_{k\,\text{I}}$, а ток эмиттера — величине $i_{э\,\text{I}}$. В то же время ток коллектора триода $T_{1,\,k\_1}$ должен стать равным

$$i_{kII} = i_{kI} — i_{э\text{I}}$$

дело в том, что величина тока через сопротивление R_1 в (k — 1)-м разряде сохраняется такой же, как в предыдущем случае ($i_{k\,\text{I}}$) — поскольку напряжение на коллекторе $T_{1,\,k-1}$ такое же, как прежде,— но часть этого тока ($i_{э\text{I}}$) ответвляется теперь в триод $T_{3,k}$. Поскольку ток через сопротивление R_2 в базовой цепи $T_{1,\,k-1}$ тоже сохраняется прежним, уменьшению коллекторного тока триода $T_{1,\,k-1}$ соответствует увеличение его эмиттерного тока; ток эмиттера становится равным

$$i_{э\text{II}}=2i_{э\text{I}}$$

Заметим, что на рис. 2-34, *б* мы уже не решились изобразить триод $T_{1,k}$-1 в виде двух диодов, так как величина $i_{k\,\text{II}} = i_{k\,\text{I}} - i_{э\,\text{I}}$ может оказаться отрицательной, т. е. коллекторный ток триода $T_{1,k-1}$ может оказаться направленным от базы к коллектору. Указанный триод находился бы при этом в обычном режиме насыщения. Но если даже параметры схемы таковы, что величина $i_{к\,\text{II}}$ еще положительна, то при других комбинациях цифр слагаемых триод $T_{1,k-1}$ все равно перейдет в режим отрицательного тока коллектора.

Действительно. Представим себе теперь, что комбинация цифр слагаемых такова, что в (k — 1)-м и *k-м* разрядах по-прежнему открыты соответственно триоды $T_{1,\,k-1}$ и $T_{3,k}$, но открыт также триод $T_{3,\,k+1}$ в (k+1)-м разряде, а в (k+2)-м

разряде триод T_3 заперт. При этом эквивалентная схема получится такой, как показано на рис. 2-34, *в.* Очевидно, что в этом случае триод $T_{3,\,k+1}$ будет находиться в таком же режиме, как в предыдущем случае был триод $T_{3,k}$ и как в первом из рассмотренных случаев находился триод $T_{1,;k\text{-}1}$; триод $T_{3,k}$ теперь будет находиться в таком же режиме, как в предыдущем случае был триод $T_{1,k\text{-}1}$; для триода же $T_{1,\,k\text{-}1}$ ток коллектора теперь будет равен

$$i_{k\,\mathrm{III}} = i_{k\mathrm{I}} — 2i_{э\mathrm{I}},$$

а ток эмиттера

$$i_{э\,\mathrm{III}} = 3i_{э\mathrm{I}}.$$

Вообще, если в (k — 1)-м разряде включен триод $T_{1,\,k\text{-}1}$, а в p находящихся слева от него разрядах включены триоды T_3, то ток коллектора триода $T_{1,\,k\text{-}1}$ равен

$$i_{k\mathrm{I}} — pi_{э\mathrm{I}},$$

а ток эмиттера $pi_{э\mathrm{I}}$.

Если p достаточно велико, то коллекторный ток отрицателен (т. е. направлен от базы к коллектору) и с увеличением p возрастает по абсолютной величине.

Количество включенных последовательно разрядов не должно быть слишком велико — иначе ток базы $i_б$ триода $T_{1,\,k\text{-}1}$, который во всех рассмотренных режимах остается одинаковым (он задается фактически генератором тока —E_2 — R_2), может оказаться недостаточным для насыщения триода. Чтобы этого не произошло, отрицательный ток коллектора при максимальных значениях p должен быть по абсолютной величине меньше, чем $i_б \cdot \beta$, где β — коэффициент усиления триода по току.

В рассмотренном варианте сигналом наличия или отсутствия переноса являлся тот или иной уровень напряжения. Наряду с этим возможны и *токовые варианты* схемы с мгновенным переносом*). Принцип построения токового варианта иллюстрируется рис. 2-35, на котором схематически показаны три разряда цепи переносов.

Каждый разряд, как и в схеме рис. 2-33, состоит из трех триодов: T_1, T_2 и T_3. Но назначение этих триодов здесь несколько иное.

Триод T_1 вместе с сопротивлением его коллекторной нагрузки представляет собой генератор импульса тока. Технические подробности на рис. 2-35 не показаны; в действительности же T_1 может быть включен по схеме токового ключа или, возможно, является триодом феррито-транзисторной ячейки. Генератор тока дает импульс в том случае, когда оба слагаемых в данном разряде содержат единицы. Если одна из цифр слагаемых в данном разряде есть единица, а другая нуль, то генератор тока (T_1) не работает, а открыт (насыщен) триод T_3; если в его эмиттер подается импульс, то он при этом проходит в коллекторную цепь.

Таким образом, сигналом наличия переноса в рассматриваемой схеме является импульс тока, который либо формируется в некотором разряде безусловно (в случае $bc = 1$),

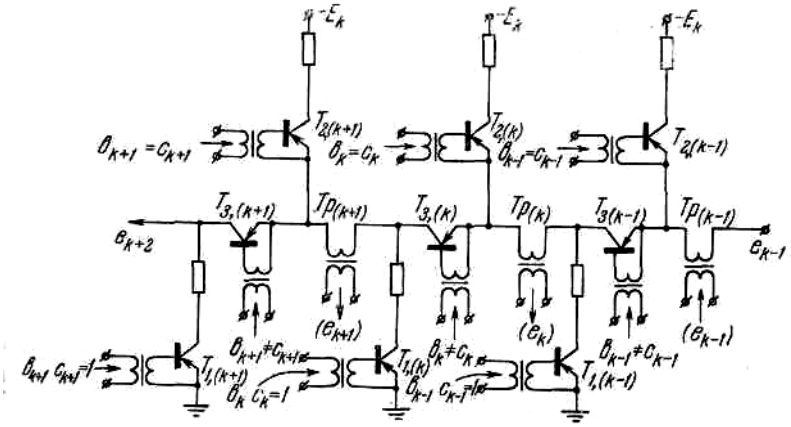


Рис. 2-35. Токовый вариант схемы с «мгновенным» переносом.

либо появляется при условии, что имеется перенос из предыдущего разряда (в случае $b \neq c$). Триод T_2 открыт при условии $b = c$; он служит для того, чтобы замкнуть цепь импульса тока предыдущего разряда в том случае, когда этот импульс не используется в данном разряде, т. е. когда в данном разряде заперт триод T_3.

В схему формирования суммы сигналы переноса поступают через трансформаторы тока *Тр*, которые могут использоваться, например, для запуска феррито-транзисторных ячеек.

Как и в схеме рис. 2-33, переключение всех триодов по всем разрядам происходит одновременно. Последовательным процессом является только, возможно, распространение импульса тока от младших разрядов к старшим по цепи, составленной из открытых триодов T_3 и трансформаторов *Тр*.

Схемы с «мгновенным» переносом удобны в тех случаях, когда регистры, связанные с параллельным сумматором, построены из синхронных элементов. Применение такой схемы позволяет опрашивать все эти элементы одновременно и одновременно же передавать на них цифры суммы по всем разрядам. Отсутствие задержек переноса делает эту

*) Впервые один из них, насколько это известно автору, разработан в Московском энергетическом институте на кафедре математических машин дискретного действия.

схему весьма перспективной и для случая применения в регистрах асинхронных элементов.

На первый взгляд может показаться, что создание схем с «мгновенным» переносом вообще снимает с повестки дня вопросы об оптимальном логическом построении цепи переносов, о методах ускорения суммирования в параллельных сумматорах, о создании сверхпараллельных и параллельно-параллельных схем. В действительности это не так.

После того как в схеме с «мгновенным» переносом по всем разрядам сформированы подготовительные функции (скажем, для схемы рис. 2-35 должны формироваться подготовительные функции $D_k = b_k c_k$, $R_k = b_k \bar{c}_k + \bar{b}_k c_k$, $\bar{D}_k \bar{R}_k = \bar{b}_k \bar{c}_k$), формирование всех сигналов переноса в ней требует столько времени, сколько необходимо для переключения всего лишь одного триода. Очевидно, что ни в какой схеме нельзя выполнить эту операцию с меньшим количеством ступеней и что схема с «мгновенным» переносом является с этой точки зрения идеальной. Но все дело в том, что этой единственной ступенью должен быть обязательно триод, переключающийся из запертого состояния в глубокое насыщение и из насыщенного состояния в запертое. При этом не всякие триоды пригодны для использования в схемах с «мгновенными» переносами; к примеру, в схеме рис. 2-33 нельзя использовать наиболее быстродействующие из серийных отечественных транзисторов — диффузионные, так как возникающие в схеме обратные напряжения на эмиттерных переходах слишком велики.

По этим причинам может оказаться, что схема с «мгновенным» переносом работает медленнее, чем, скажем, параллельно-параллельная схема, где для образования сигналов переноса должны сработать 2 ступени, и даже, возможно, медленнее, чем сверхпараллельная схема или схема, учитывающая статистическую структуру переносов, где количество ступеней, срабатывающих последовательно при формировании сигналов переноса, получается порядка $\log m$ (где m — количество разрядов в слагаемых).

К тому же схема с «мгновенным» переносом требует применения специальных элементов схемы — не таких, какэлементы, используемые в других логических цепях машины, а сами сигналы переноса, вырабатываемые в ней, имеют нестандартный вид, отличающийся от вида сигналов в других цепях. К примеру, в устройстве рис. 2-33 сигналами наличия или отсутствия переносов являются уровни напряжения —3 ÷ —4,5 *в;* в то же время подготовительные функции, управляющие триодами в схеме переносов и формируемые, по-видимому, типовыми для машины логическими цепями, должны иметь вид уровней напряжения—2,5 ÷ —5,5*в*; такое различие не случайно — оно необходимо **в** принципе, иначе схема не могла бы работать. Ясно, что рассматривавшиеся раньше построения цепей переноса свободны от этих недостатков, присущих схемам с «мгновенным» переносом.

Таким образом, неправильно было бы полагать, что схемы с «мгновенным» переносом представляют собой всегда и в любом случае наилучшее решение при построении параллельного сумматора.

2.6. Накапливающие сумматоры. Счетчики

Как говорилось в 2.1.3, основу накапливающего сумматора, работающего в системе счисления с основанием *n,* представляет счетчик импульсов по модулю *n.* Счетчики импульсов вообще широко используются в схемах цифровых вычислительных машин — не только в составе накапливающих сумматоров, но и в цепях управления и других схемах. Мы воспользуемся подходящим случаем, чтобы вначале рассмотреть вообще построение счетчиков, причем проведем это рассмотрение более полно, чем было бы необходимо, если бы речь шла только о накапливающих сумматорах. После этого уже рассматриваются вопросы, касающиеся непосредственно построения накапливающих сумматоров.

2.6.1. Двоичные счетчики импульсно-потенциального типа.

Простейшим счетчиком по модулю 2 является триггер со счетным входом. Каждый входной импульс меняет состояние триггера на противоположное тому, которое имелось в триггере ранее. Если одно состояние триггера соответствует цифре «0», а другое — «1» и если триггер первоначально находился в состоянии «0», то первый импульс переведет его в состояние «1», второй — в состояние «0», третий — снова в состояние «1» и т. д. В любой момент времени число, хранящееся в триггере, равно по модулю 2 количеству импульсов, поступивших на его вход.

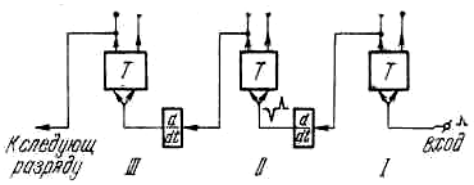


Рис. 2-36. Построение простого многоразрядного двоичного счетчика.

Несколько триггеров со счетными входами можно объединить в простую схему многоразрядного двоичного счетчика. Принцип построения такой схемы иллюстрируется рис. 2-36. На рисунке показаны три младших разряда счетчика.

Счетный вход триггера 1-го разряда (триггера *I*) является входом счетчика. Импульсы, которые должны быть сосчитаны счетчиком, могут поступать на его вход через любые интервалы времени, но не чаще, чем это допускается максимальной частотой переключений триггера.

Сигналы с одного из выходов триггера *I* через дифференцирующую (укорачивающую) цепочку подаются на счетный вход триггера *II*. Точного дифференцирования выходного напряжения, конечно, не требуется: необходимо только, чтобы в момент перехода триггера *I* из состояния «1» в состояние «0» на счетный вход триггера *II* передавался такой импульс, какой необходим для переключения триггера (предполагается, что к импульсам противоположной полярности триггеры нечувствительны). Например, если триггеры должны запускаться по счетному входу положительными импульсами, то дифференцирующая цепочка присоединяется к тому выходу триггера *I*, на котором в положении «1» напряжение низкое, а в положении «0» — высокое.

Аналогичным образом в момент перехода триггера *II* из положения «1» в положение «0» на счетный вход триггера *III* через дифференцирующую цепочку подается импульс такой полярности, как необходимо для запуска триггера *III,* и т. д.

Нетрудно видеть, что в рассматриваемой схеме реализуется, таким образом, правило счета, сформулированное нами в 1.3.2 (стр. 20) для позиционных способов изображения чисел с естественными весами разрядов. Если до прихода первого импульса все триггеры находились в состоянии «0», то в дальнейшем комбинация состояний триггеров счетчика указывает в двоичной системе количество импульсов, поступивших на его вход. Если счетчик содержит *m* триггеров, то счет осуществляется по модулю 2^m.

Для установки начального состояния в счетчике рис. 2-36 можно предусмотреть шину гашения, объединяющую входы установки в «0» всех триггеров. При подаче на нее импульса все триггеры счетчика одновременно устанавливаются в положение «0» — независимо от того, в каких состояниях они находились прежде. Импульс, подаваемый на шину гашения, должен быть достаточно широким, с тем, чтобы импульсы, возникающие на счетных входах некоторых триггеров в момент гашения предыдущих триггеров, не могли бы повлиять на состояния данных триггеров.

Хотя счетчики, выполненные в соответствии с рис. 2-36, используются на практике довольно часто, схема эта обладает рядом существенных недостатков. Первый из них — сравнительно большое запаздывание в установлении состояния счетчика относительно момента поступления входного импульса. Запаздывание объясняется тем, что переброс каждого последующего триггера начинается лишь тогда, когда импульс на его входе достигает определенной величины, т. е. когда на выходе предыдущего триггера напряжение изменилось уже довольно значительно; таким образом, в общее запаздывание частично входят длительности фронтов каждого из триггеров.

На рис. 2-37 показано несколько вариантов двоичных счетчиков с уменьшенным запаздыванием или без запаздывания.

В счетчике рис. 2-37, *а* применена идея обычной цепи переносов — аналогичной цепи переносов параллельного комбинационного сумматора. Правило построения цепи переносов в счетчике очень просто: импульс на следующий разряд должен передаваться в том и только в том случае, когда поступает импульс в данный разряд и когда до его поступления в данном разряде имелась цифра «1». Запаздывание в этой схеме складывается не из фронтов триггеров, а из задержек сигналов в логических элементах цепи переносов. При использовании обычных элементов схема рис. 2-37, *а* получается более быстродействующей, чем схема рис. 2-36.

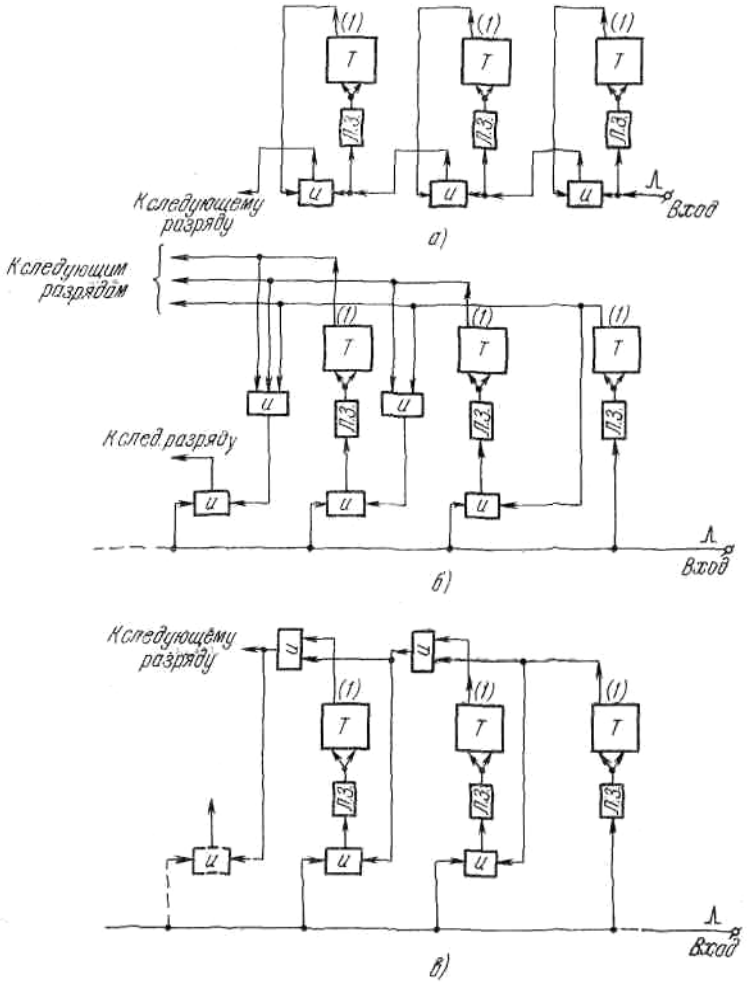


Рис. 2-37. Варианты схемы многоразрядного двоичного счетчика:

а) схема с цепью переносов, работающей на импульсных сигналах;
б) схема без запаздывания;
в) схема с цепью переносов, построенной по потенциальному принципу.

Линии задержки, имеющиеся в каждом разряде рассматриваемой схемы, поставлены для того, чтобы напряжение, поступающее от триггера на один из входов элемента «и», не успело измениться прежде, чем закончится импульс на другом его входе. Линии можно ставить не между выходами вентилей и входами триггеров, а между выходами триггеров и потенциальными входами вентилей. По длительности линия должна обеспечивать задержку на время импульса. В действительности, однако, фронт триггера обычно длиннее, чем запускающий импульс, и линии задержки ставить не обязательно. То же

относится к линиям задержки в схемах рис. 2-37, *б* и 2-37, *в,* а также в схеме рис. 2-38 и некоторых других рассматриваемых ниже схемах.

В счетчике рис. 2-37, *б* запаздывание от разряда к разряду отсутствует вообще; в любом разряде, начиная от второго, импульс, поступающий на счетный вход триггера, представляет собой входной импульс счетчика, прошедший через один вентиль «и». Вентиль второго разряда открыт в том случае, если к моменту прихода импульса в первом разряде имеется единица, вентиль третьего разряда — если имеются единицы в первом и втором разрядах, и т. д. Недостатком этой схемы является необходимость в элементах «и» на большое количество входов — тем большее, чем больше разрядов в счетчике.

Возможно и компромиссное решение. Скажем, 4 младших разряда счетчика можно построить по схеме рис. 2-37, *б*; по этой же схеме формируется и импульс для 5-го разряда, для чего нужен элемент «и» на 4 входа; следующая четверка разрядов по схеме в точности повторяет младшие разряды (причем самый большой в ней элемент «и» имеет 4 входа), но входным импульсом для нее является выходной импульс первой четверки разрядов (сформированный для 5-го разряда) и т. д. В такой схеме все разряды счетчика как бы разбиты на группы: внутри групп связи выполнены в соответствии с рис. 2-37, б, а между группами — в соответствии с рис. 2-37, *а.* Максимальное количество входов, которое необходимо иметь в элементах «и», зависит от количества разрядов в группе, а максимальное запаздывание — от количества групп.

Схема рис. 2-37, *в* по логике напоминает схему рис. 2-37, *б.* Однако в ней к элементу «и» в некотором разряде вместо выходов триггеров предыдущих разрядов подводится выход элемента «и» предыдущего разряда. При этом каждый из элементов имеет всего два входа. Смысл функций, формируемых элементами «и», оказывается в точности таким же, как в схеме рис. 2-37 *б*, и точно так же, как в схеме рис. 2-37 *б*, отсутствует запаздывание в установлении состояния триггеров счетчика друг относительно друга. Зато имеется запаздывание в установлении потенциальных сигналов на входах вентилей: элемент «и» в каждом из разрядов срабатывает только после того, как сработают элементы «и» во всех предыдущих разрядах. Таким образом, если в схемах рис. 2-36 и 2-37, *а* требуется некоторое время, чтобы после входного импульса установились последовательно все триггеры счетчика, то здесь все триггеры счетчика устанавливаются одновременно; необходимо некоторое время в п р о м е ж у т к е между входными импульсами, чтобы последовательно установились сигналы на потенциальных входах вентилей. В общее запаздывание потенциальных сигналов входят также задержки на промежуточных усилителях, которые на рис. 2-37, *в* не показаны, но без которых нельзя соединить последовательно друг за другом большое количество элементов «и».

Как и в предыдущем случае, возможна комбинация схемы рис. 2-37, *в* со схемой рис. 2-37, *а.* Если все разряды счетчика разделены на группы, причем внутри групп соединения выполнены по схеме рис. 2-37, *в,* а между группами — по схеме рис. 2-37, *а,* то запаздывание в формировании подготовительных функций (в промежутках между импульсами) и количество соединенных последовательно ступеней элементов «и» зависят от количества разрядов в группе, а запаздывание в установлении триггеров счетчика относительно входного импульса зависит от количества групп.

В вариантах схемы счетчика, приведенных на рис. 2-37, устранен (или уменьшен) основной недостаток схемы рис. 2-36—большое запаздывание; однако все эти варианты, как и схема рис. 2-36, основаны на использовании счетных входов триггеров. Вообще использование счетного входа вносит в работу триггера некоторый элемент ненадежности; другие типы цифровых элементов (динамические триггеры, магнитные элементы и т. д.) не допускают устройства счетных входов. Схема двоичного счетчика без использования счетных входов триггеров приведена на рис. 2-38.Она рассчитана на триггеры с раздельными входами; однако по тому же принципу можно построить счетчик из других цифровых элементов.

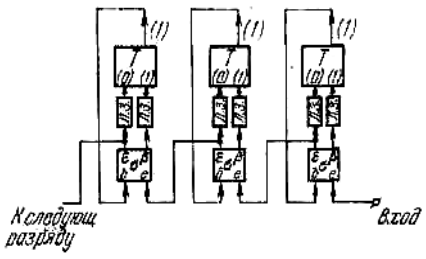


Рис. 2-38. Двоичный счетчик без использования счетных входов триггеров.

Рассматриваемая схема представляет собой фактически регистр, соединенный с параллельным двоичным комбинационным сумматором. В каждом разряде вместо полного одноразрядного сумматора (на 3 входа) имеется полусумматор — так как второе слагаемое отсутствует, и производится лишь сложение цифры исходного числа (из соответствующего триггера регистра) с цифрой переноса из предыдущего разряда. В первом разряде цифра исходного числа складывается с входной единицей счетчика. Для установки триггеров в нулевые состояния использовано то обстоятельство, что изменение состояния триггера некоторого разряда с «1» на «0» возможно лишь при одновременной передаче единицы в следующий разряд.

2.6.2. Двоичные счетчики с потенциальными связями.

Используя только раздельные входы триггеров, можно построить и такую схему счетчика, в которой вообще было бы исключено запоминание на реактивностях.

В триггерах со счетным входом запоминание на реактивностях используется в скрытом виде: когда импульс поступает на счетный вход, то новое состояние триггера зависит непосредственно от того, в каком состоянии он находился прежде, т. е. от того, какие заряды имеются на форсирующих и паразитных емкостях. В схемах

рис. 2-37 и 2-38 имеются, кроме того, явные реактивности — линии задержки, составленные из индуктивностей и емкостей и представляющие собой элементы временного хранения информации; если линии задержки и можно исключить, как говорилось выше, то это значит, что мы рассчитываем на запоминание на паразитных емкостях. При этом ясно, что, имея всего один триггер на разряд счетчика, нельзя вообще обойтись без каких-нибудь дополнительных элементов запоминания. При поступлении входного сигнала в некоторый разряд нужно одновременно и использовать предыдущую информацию, хранившуюся в нем, от которой зависят будущее состояние этого разряда и передача сигнала на следующие разряды, и в то же время цифровой элемент данного разряда должен воспринять новую цифру. Если на реактивности рассчитывать нежелательно, то каждый разряд счетчика должен содержать по меньшей мере два цифровых элемента.

Схема такого' типа представлена на рис. 2-39. Отличительной особенностью ее является возможность использовать во всем счетчике только потенциальные связи.

Вход счетчика рис. 2-39 — парафазный двухпроводный. Наличие сигнала на входе изображается высоким напряжением на одном из проводов и низким на другом, отсутствие сигнала — низким напряжением на первом проводе и высоким напряжением на втором. Единственное требование к форме входных сигналов состоит в том, что при переходных процессах уровень напряжения, соответствующий сигналу «1», не должен появляться одновременно на обоих проводах. Для определенности мы будем полагать, что сигналу «1» соответствует высокое напряжение, сигналу «0» — низкое напряжение. Тот из входных проводов, на котором напряжение высокое, когда на вход счетчика поступает сигнал, обозначен на рисунке цифрой 1.

Из двух триггеров, входящих в каждый разряд счетчика, один (на рисунке — верхний) является основным,

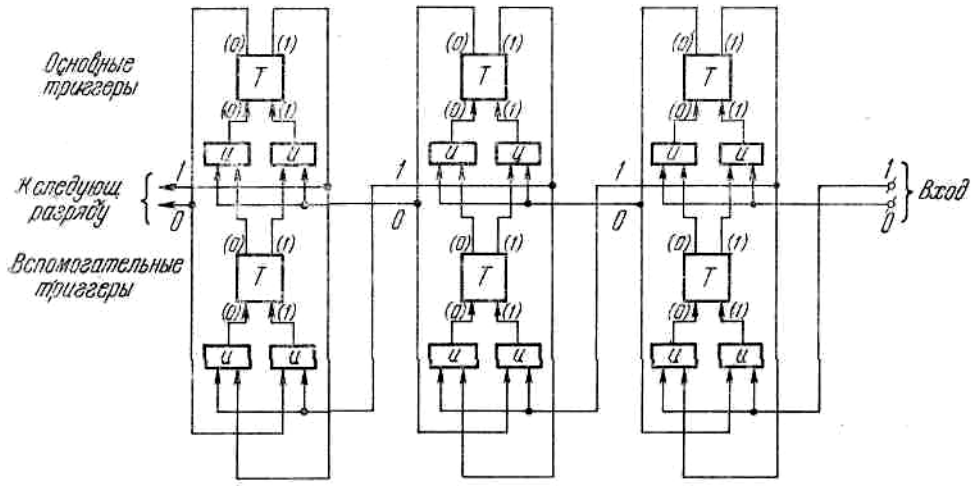


Рис. 2-39. Счетчик с потенциальными связями (три разряда).

другой —вспомогательным. Связи между триггерами в каждом разряде устроены так, что, когда на входе данного разряда (на входе счетчика или в основном триггере предыдущего разряда) имеется единица, вспомогательный триггер принимает состояние, обратное состоянию основного триггера, а когда на входе данного разряда имеется нуль, основной триггер устанавливается в соответствие со вспомогательным. Очевидно, что в любой момент времени в каждом разряде может изменяться состояние лишь одного из двух триггеров, причем новое его состояние зависит только от входного для данного разряда сигнала и от состояния другого триггера. Поэтому никаких дополнительных элементов запоминания не требуется.

Что приведенная схема действительно реализует правило двоичного счета, видно из следующих соображений: пока на входе некоторого разряда имеется «1», в нем происходит подготовка вспомогательного триггера; после того как сигнал на входе сменился с «1» на «0», состояние основного триггера в данном разряде меняется на противоположное тому, которое имелось прежде.

Проследим работу счетчика по нескольким начальным импульсам. Предположим, что вначале все основные и все вспомогательные триггеры находятся в состоянии «0» и сигнал на входе есть «0». Состояние счетчика при этом можно представить в виде

(осн.)... 000	(0)
(всп.)... 000	(вход)

Поскольку на входе имеется «0», то в первом разряде происходит передача цифры «0» из вспомогательного триггера в основной; поскольку первый основной триггер содержит «0», то и во втором разряде цифра «0» передается из вспомогательного триггера в основной и т. д. Фактически для того, чтобы установить такое состояние счетчика, достаточно только в течение некоторого времени задержать вспомогательные триггеры в состоянии «0»; при наличии сигнала «0» на входе основные триггеры сами установятся в состоянии «0».

Пусть теперь на вход подается «1». При этом в первом разряде вспомогательный триггер примет состояние, обратное состоянию основного триггера, т. е. станет в положение «1». Поскольку основной триггер первого разряда пока сохраняет «0», во втором разряде цифра «0» по-прежнему передается из вспомогательного триггера

в основной и т. д. Положение счетчика в момент действия первого сигнала «1» на входе имеет вид

(осн.)... 000	(1)
(всп.)… 001	(вход)

После того как на входе вновь появится «0», в первом разряде цифра «1»из вспомогательного триггера будет передана в основной. Появление цифры «1» в первом основном триггере приведет к тому, что во втором разряде вспомогательный триггер воспримет цифру «1» (обратную цифре второго основного триггера). Так как второй, основной триггер пока по-прежнему хранит «0», в третьем разряде, как и прежде, происходит передача «0» из вспомогательного триггера в основной. Положение счетчика при этом имеет вид

(осн.)... 001	(0)
(всп.)…011	(вход)

Рассуждая точно так же, найдем, что последующие положения счетчика проходятся в следующем порядке:

Во время действия на входе:			После окончания на входе:	
2-го импульса —	(осн.) . . . 001	(1)	. . . 010	(0) — 2-го импульса
	(всп.) . . . 010		. . . 110	
3-го импульса —	(осн.) . . . 010	(1)	. . . 011	(0) — 3-го импульса
	(всп.) . . . 111		. . . 101	
4-го импульса —	(осн.) . . . 011	(1)	. . . 100	(0) — 4-го импульса
	(всп.) . . . 100		. . . 100	
5-го импульса —	(осн.) . . . 100	(1)	. . . 101	(0) — 5-го импульса
	(всп.) . . . 101		. . . 111	
6-го импульса —	(осн.) . . . 101	(1)	. . . 110	(0) — 6-го импульса
	(всп.) . . . 110		. . . 010	
7-го импульса —	(осн.) . . . 110	(1)	. . . 111	(0) — 7-го импульса
	(всп.) . . . 011		. . . 001	
8-го импульса —	(осн.) . . . 111	(1)	. . . 000	(0) — 8-го импульса
	(всп.) . . . 000		. . . 000	

Комбинация, которая получается в основных триггерах счетчика (с 1-го по m-й) после окончания любого импульса на входе, представляет в обычном двоичном коде количество импульсов, сосчитанных счетчиком (по модулю 2^m). В то же время комбинация, содержащаяся во 2-м ÷ (m + 1)-м вспомогательных триггерах, представляет то же число в рефлексном коде (коде I рэя — см. 1.8.2).

Разберемся во временных соотношениях.

Прежде всего обратим внимание на то, каким способом происходит установление состояний основных триггеров. Переключение основного триггера в каком-либо разряде начинается с того момента, как на одном из его входов (установки нуля или установки единицы) появляется высокое напряжение. При этом сначала на соответствующем выходе триггера появляется низкое напряжение, оно передается по внутренней цепи обратной связи на другой вход, в результате чего на другом выходе появляется высокое напряжение. Таким образом, во время переходного процесса может быть такой момент времени, когда на обоих выходах основного триггера имеются низкие напряжения, но невозможно положение, когда на обоих его выходах одновременно появляются высокие напряжения. Выходы основного триггера некоторого разряда являются входными напряжениями для следующего разряда. Аналогичное положение имеется, как мы условились раньше, и на входах первого разряда. Следовательно, невозможно положение, когда в каком-либо разряде одновременно переключаются и основной, и вспомогательный триггеры. Их временные характеристики поэтому особой роли не играют. Важно лишь, чтобы уровень «1» на входе счетчика держался по меньшей мере столько времени, сколько необходимо для переключения одного триггера. Уровень «0» должен держаться по меньшей мере столько времени, сколько необходимо для последовательного переключения m триггеров (m — число разрядов в счетчике): за это время, возможно, последовательно должны будут переключиться из состояния «1» в состояние «0» все основные триггеры.

Если бы вход не был парафазным, то требования к форме входных сигналов оказались бы более жесткими. На рис. 2-40 для примера показано построение одного разряда счетчика с потенциальными связями из элементов типа «и», «или» и «нет». Схема эта в общем соответствует построению рис. 2-39, но вход в ней однопроводный.

Основной триггер образован инверторами «нет-1» и «нет-2». Обратные связи его замкнуты через элементы «и-1» и «и-2». Поэтому запускается этот триггер низкими уровнями напряжения: если на «свободный» (незанятый в цепи обратной связи) вход элемента подать низкие напряжения (соответственно с выхода «или-1» либо с выхода «или-2»), то триггер установится в то или иное состояние независимо от того, в каком состоянии он находился прежде. Во вспомогательном триггере, образованном инверторами «нет-3» и «нет-4», обратные связи замкнуты через элементы «или-3» и «или-4». Поэтому вспомогательный триггер запускается высокими уровнями напряжения; его состояние может измениться тогда, когда на «свободном» входе «или-3» или «или-4» (на выходе элементов «и-3» или «и-4») появляется высокое напряжение.

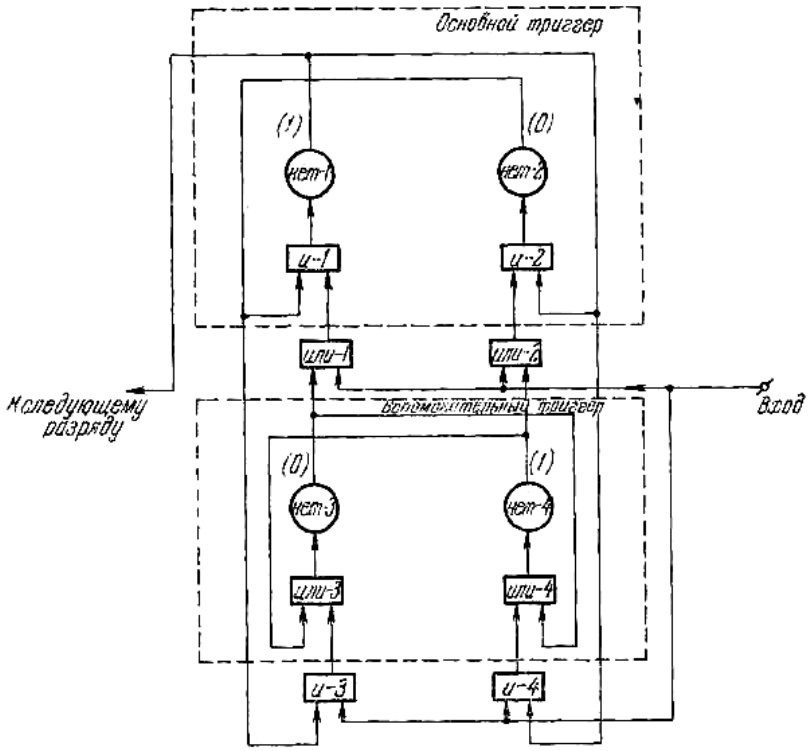


Рис. 2-40. Один разряд двоичного счетчика с потенциальными связями и однопроводным входом.

По рисунку нетрудно проследить, что логика работы этой схемы не отличается от логики работы схемы рис. 2-39: когда на входе имеется «1» (высокое напряжение), состояние вспомогательного триггера становится обратным состоянию основного триггера; когда на входе сигнал отсутствует (низкое напряжение), основной триггер устанавливается в соответствие со вспомогательным. Но теперь нужно специально позаботиться о том, чтобы во время фронта входного сигнала не могли бы срабатывать одновременно и основной, и вспомогательный триггеры (что могло бы привести к лишним просчетам счетчика). Для того чтобы выполнялось это условие, достаточно, чтобы фронт входного напряжения был короче, чем временная задержка в цепи «диодная логическая схема — инвертор».

Действительно. Предположим, что напряжение на входе было низкое и что оно меняется с низкого на высокое. Предположим, кроме того, что оба триггера находились ранее в состоянии «0». Высокое напряжение со входа, поступая на элемент «и-3», позволяет установиться высокому напряжению на его выходе, в результате чего вспомогательный триггер принимает состояние «1». При этом опасность могла бы состоять в том, что напряжение на выходе инвертора «нет-3» станет низким прежде, чем окончательно установится высокое напряжение на входе; в результате на некоторое время могло бы стать низким напряжение на выходе элемента «или-1» и основной триггер перебросился бы в состояние «1». Вслед за этим вспомогательный триггер установился бы не в состояние «1», а в состояние «0»; если бы и в это время еще тянулся фронт входного напряжения, то на некоторое время могло бы стать низким напряжение на выходе «или-2» и т. д. Пара триггеров данного разряда за время фронта входного напряжения могла бы совершить таким образом несколько просчетов. Если, однако, фронт входного напряжения заканчивается раньше, чем изменяется напряжение на выходе инвертора «нет-3», т.е. если фронт входного сигнала короче, чем запаздывание в цепи «и-3» — «или-3» — «нет-3», то в этот момент ошибки в работе схемы не возникнут.

Поскольку входные сигналы получаются обычно от элементов такого же типа, как элементы, из которых построен счетчик, указанное условие выполняется, и схема рис. 2-40 работает достаточно устойчиво. Но нужно помнить, что она все-таки более критична к временным характеристикам элементов и к форме входных сигналов, чем схема рис. 2-39.

В схемах счетчиков с потенциальными связями возможно применение ряда методов ускорения счета; например, ускорение может быть достигнуто на основе идей схемы рис. 2-37,*б*. Одно интересное усовершенствование приведенных схем, позволяющее получить увеличение быстродействия при сокращении количества оборудования, мы рассмотрим в следующем разделе.

2.6.3. Счетчики для других систем счисления.

В течение ряда лет, пока не были известны достаточно надежные и экономичные многопозиционные элементы, счетчики для систем счисления, отличных от двоичной, строились большей частью из двоичных цифровых элементов путем введения некоторых усложнений в схему.

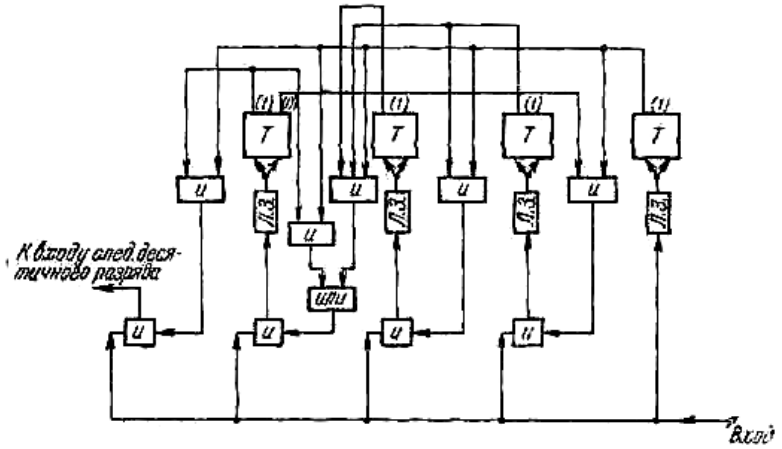


Рис. 2-41. Один разряд десятичного счетчика из двоичных триггеров со счетными входами.

На рис. 2-41 для примера показано построение десятичного счетчика из двоичных триггеров со счетными входами.

Каждый десятичный разряд представляет собой некоторое усложнение 4-разрядного двоичного счетчика рис. 2-37, *б*; устройство связей между десятичными разрядами основано на идеях схемы рис. 2-37, *а.*

Таким образом, предлагаемая схема представляет собой с некоторыми усложнениями двоичный счетчик, в котором двоичные разряды объединены в группы по четыре.

Усложнения, как видно из сравнения рис. 2-37, *б* с рис. 2-41, имеются в двух местах. Во-первых, вентиль «и» второго двоичного разряда пропускает импульс не во всех случаях, когда первый триггер содержит единицу, а только при условии, что четвертый триггер содержит нуль. Во-вторых, вентиль «и» четвертого двоичного разряда пропускает импульс не только в том случае, когда три предыдущих двоичных разряда содержат единицы, но и в том случае, когда имеются единицы в первом и четвертом двоичных разрядах.

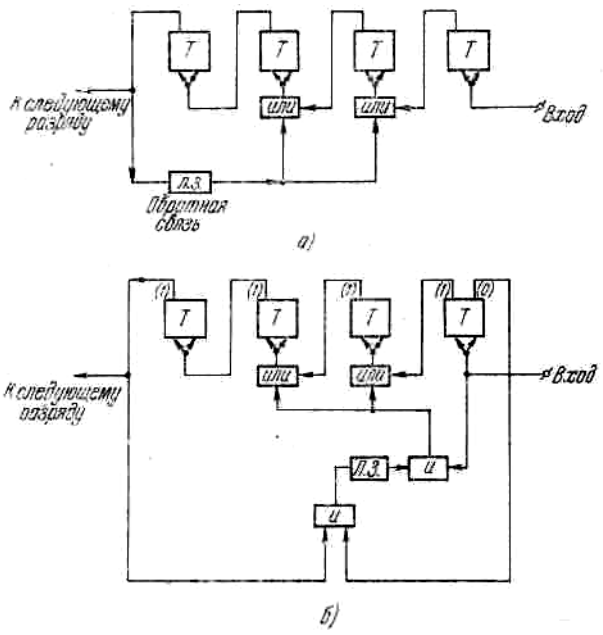


Рис. 2-42. Варианты схемы десятичного счетчика из двоичных элементов (один десятичный разряд): *а*) счетчик с обратной связью; *б*) счетчик с дополнительным насчетом.

Легко видеть, что все положения от 0-го до 9-го включительно (0000, 0001, 0010, ..., 0111, 1000, 1001) проходятся таким счетчиком в нормальном порядке, и имеющиеся в схеме усложнения не играют при этом никакой роли. Только в последнем, 9-м положении (1001) единица, появившаяся в четвертом двоичном разряде, запрещает прохождение следующего входного импульса через вентиль «и» второго разряда и разрешает его прохождение через вентиль четвертого разряда. Поэтому счетчик из 9-го положения переходит не в 10-е (1010), а в нулевое (0000). Одновременно через специальный вентиль «и» (на рисунке — слева) формируется входной импульс для следующего десятичного разряда. Схема формирования этого импульса даже проще, чем она была бы в обычном двоичном счетчике. Все положения счетчика, в которых первый и четвертый двоичный разряды содержат единицы, кроме 9-го положения, обходятся; поэтому элемент «и» в схеме формирования импульса для следующего разряда может содержать всего 2 входа.

На рис. 2-42 приведены два построения десятичного счетчика, тоже основанные на двоичной технике, но использующие несколько иные идеи. Если в счетчике рис. 2-41 один из входных импульсов (десятый) переключался системой вентилей из обычных каналов в некоторый специальный путь, то в счетчике рис. 2-42, *а* применен принцип обратной связи, а в счетчике рис. 2-42, *б* — принцип дополнительного насчета.

Схема, представленная на рис. 2-42, *а,* представляет собой в основе обычный двоичный счетчик, построенный в соответствии с рис. 2-36 или, может быть, в соответствии с рис. 2-37, *а,* или *б,* или *в;* характер связей между двоичными разрядами на рисунке умышленно не показан. Отличие от обычного двоичного счетчика состоит в том, что выходной импульс четвертого двоичного разряда по цепи обратной связи возвращается через линию задержки на входы второго и третьего двоичных разрядов. Поэтому счетчик из последнего положения (1111) переходит не к нулевому состоянию (0000), а сразу к 6-му (0110); таким образом, при счете он проходит не 16, а только 10 состояний (0110, 0111, 1000,..., 1110, 1111).

Схема рис. 2-42, *б* по идее ближе к схеме рис. 2-41. В ней, однако, вместо переключения одного из входных импульсов из одного канала в другой сделано просто включение входного импульса в некоторый дополнительный канал. Специальный вентиль «и», имеющийся в схеме, открывается тогда, когда счетчик находится в 8-м положении (1000). Девятый импульс не только добавляет единицу в первый двоичный разряд, но добавляет единицы также во второй и третий разряды. Таким образом, счетчик как бы насчитывает лишнюю шестерку и из 8-го положения (1000) переходит сразу не в 9-е (1001), а в 15-е положение (1111). Следующий импульс переведет его вновь в нулевое положение и т. д.

Дополнительный насчет шестерки вообще удобно делать после такого состояния счетчика, в котором первый и второй двоичные разряды содержат нули. Иначе следующий импульс попадал бы на входы второго и третьего двоичных разрядов не только через дополнительный вентиль, но и по обычной цепи двоичного счета; импульс от дополнительного вентиля при этом пришлось бы специально задержать по времени, а допустимый интервал между входными импульсами удлинить в расчете на два срабатывания триггеров. Кроме 8-го положения (1000), удобным моментом для насчета лишней шестерки является нулевое положение счетчика (0000), 4-е положение (0100) и 12-е (1100). В частности, если лишний насчет шестерки производился бы после 12-го положения, то счетчик вел бы счет в коде с излишком 3, т. е. проходил бы последовательно состояния 0011, 0100, 0101, ..., 1011, 1100. Видоизменение схемы рис. 2-42, б для работы в коде с излишком 3 показано на рис. 2-43.

Имеется и много других вариантов построения десятичных и вообще *n*-ичных счетчиков из 2-позиционных триггеров.

Однако более интересной представляется возможность использования *n*-триодных (или в более общем случае n-инверторных) колец по модулю *n.*

Рис. 2-44 иллюстрирует построение *n*-ичного счетчика (*n*— четное) из «полусчетных» колец.

Схема выполнена в точности по аналогии с двоичным счетчиком рис. 2-40. Отличие состоит лишь в том, что вместо одного из 2-позиционных триггеров (основного) использовано *n*-позиционное кольцо, а вместо сигналов «0» и «1» с основного триггера к элементам «и» в цепях запуска вспомогательного триггера подведены сигналы кольца «О», или «2», или «4», ..., или *«n* — 2» и соответственно «1», или «3», или «5», ..., или *«n* — 1». Как и основной триггер в схеме рис. 2-40, кольцо запускается низкими уровнями напряжения; вспомогательный триггер в обоих случаях запускается от высоких уровней.

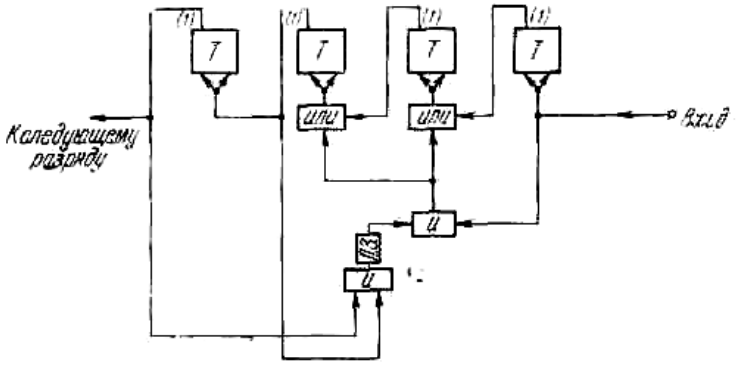


Рис. 2-43. Один десятичный разряд счетчика для работы в коде с излишком 3.

Работа схемы рис. 2-44 очевидна из описания работы счетчика рис. 2-40.*)

Схема полусчетного кольца показана на рис. 2-44, *а.* Для определенности на рисунке представлено четверичное кольцо, однако по аналогии может быть построено кольцо для любого четного модуля *n.* В любом случае кольцо имеет 2 входа, один из которых называется «четным», другой — «нечетным». Для определенности будем полагать, что единице в логических элементах схемы соответствует «высокий» уровень напряжения, нулю — «низкий».

Предположим сначала, что на обоих входах кольца напряжения высокие. Из рассмотрения схемы видно, что кольцо при этом может находиться в одном из *n* устойчивых состояний, причем каждое из устойчивых состояний соответствует наличию высокого напряжения на выходе одного из инверторов (одного из элементов «нет») и низкого напряжения на выходах всех остальных. Занумеровав инверторы 0,1,..., *n*—1, будем полагать далее, что *i*-му состоянию кольца соответствует высокое напряжение на выходе *i*-го инвертора.

Если кольцо находится в любом четном состоянии (0-м, 2-м, ...), то изменения напряжения на его «четном» входе никакого влияния на кольцо не оказывают; если же на «четном» входе оставить высокое напряжение, а на «нечетный» вход подать низкое, то кольцо перейдет из данного четного в очередное нечетное состояние (например, из 0-го — в 1-е, из 2-го — в 3-е, ...).

*) Схемы полусчетных колец и счетчиков из них были предложены впервые автором этой книги — см. «Цифровая техника и вычислительные устройства», сб. 2, АН СССР.

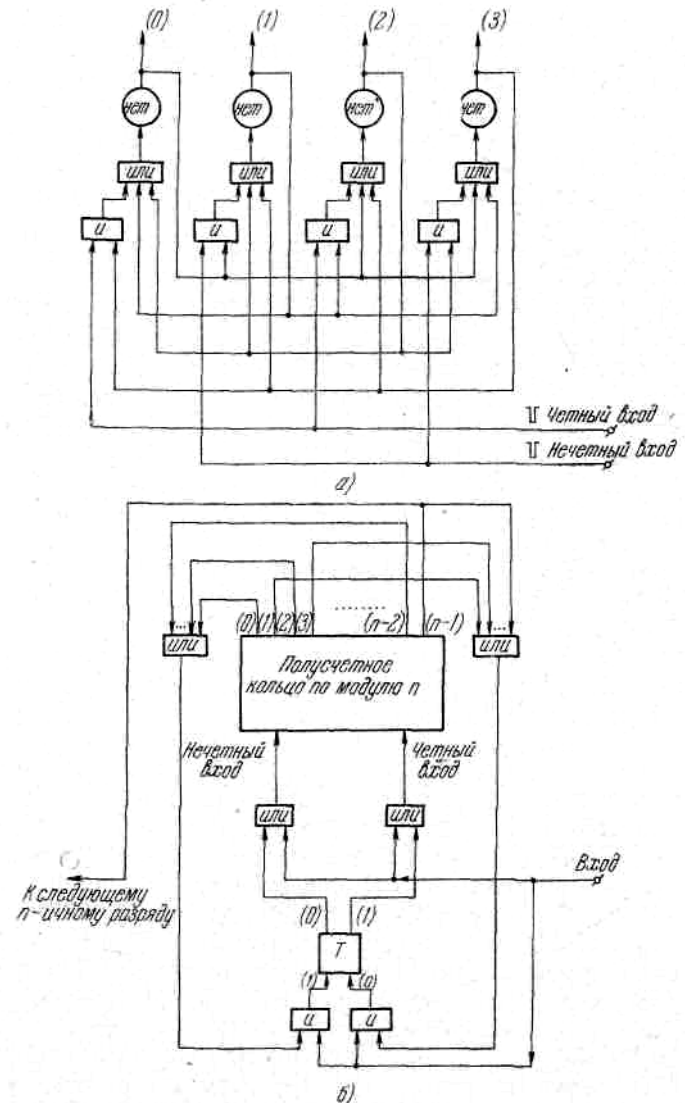


Рис. 2-44. Один разряд *n-ичного* счетчика с потенциальными связями, построенного из полусчетных колец по модулю *n: а*) схема полусчетного кольца по модулю *n* (на рисунке *n*=4); *б*) полная схема одного разряда счетчика.

Далее изменения напряжения на «нечетном» входе не будут влиять на состояние кольца; в частности, напряжение на нем можно вернуть к высокому уровню. Если после этого подать низкое напряжение на «четный» вход, то кольцо перейдет из данного нечетного в очередное четное состояние: из 1-го — во 2-е, ..., из (*n* — 1)-го — в 0-е (в данном случае *n* — 1 =3).

На рис. 2-44, *б* приведена полная схема одного разряда *n*-ичного счетчика.

Интересно, что по оборудованию схема получается явно более экономной, чем двоичная: в двоичном счетчике нам приходилось иметь по одному вспомогательному триггеру на каждый двоичный разряд, здесь же — по одному вспомогательному триггеру (тоже 2-позиционному) на один *n*-ичный разряд. Если учесть, что один 2-позиционный триггер содержит 2 инвертора, а одно *n*-позиционное кольцо — я инверторов, то окажется, что каждый разряд четверичного счетчика содержит 6 инверторов (в то время как заменяющие его 2 разряда двоичного счетчика содержали бы 8 инверторов); один разряд восьмеричного счетчика содержит 10 инверторов вместо 12 инверторов, необходимых для эквивалентных ему трех разрядов двоичного счетчика. Десятичный счетчик, построенный по этой схеме, в каждый разряд которого входит 12 инверторов, оказывается не только более экономным, чем десятичный счетчик с четырьмя двоичными разрядами на каждый десятичный разряд, но и вообще более экономным, чем чисто двоичный счетчик: на каждый десятичный разряд требуется в 3 раза больше инверторов, чем на каждый двоичный разряд (12 вместо 4), в то время как по количеству информации каждый десятичный разряд равен $\log_2 10 = 3{,}32$ двоичного разряда.

Важно отметить, что эта экономия в оборудовании достигается при одновременном увеличении быстродействия. Переключение кольца происходит примерно за то же время, что и переключение 2-позиционного триггера, выполненного из тех же элементов; небольшая разница может быть лишь за счет того, что в кольцо входят логические элементы «или» с большим числом входов, чем в 2-позиционном триггере, и что нагрузки на инверторы в кольце больше. Если пренебречь этим различием, то срабатывание одного разряда *n*-ичного счетчика происходит за то же время, что и срабатывание одного разряда двоичного счетчика; общее же количество разрядов в счетчике тем меньше, чем больше *n.* Например, если вместо 12-разрядного двоичного счетчика сделать 6-разрядный

четверичный, то общее количество инверторов сократится с 48 до 36, а скорость увеличится почти вдвое; если же вместо этого сделать 4-разрядный восьмеричный счетчик, то количество инверторов окажется равным 40, а скорость будет почти втрое выше, чем для двоичного счетчика. Четверичные и восьмеричные счетчики указанного типа выгодно поэтому применять вместо многоразрядных двоичных счетчиков. При этом разные разряды счетчика могут быть построены по разным модулям; в частности, если это необходимо, отдельные разряды могут быть двоичными.

2.6.4. Двоичные накапливающие сумматоры.

Двоичныенакапливающие сумматоры строятся на базе счетчиков по модулю 2, т. е. в простейшем случае 2-позиционных триггеров со счетными входами. На вход триггера в том или ином порядке подаются импульсы, соответствующие цифрам некоторого разряда слагаемых и цифре переноса в данный разряд (единица — наличие импульса, нуль — отсутствие). Если триггер первоначально находился в состоянии«0», то его конечное состояние соответствует цифре суммы данного разряда.

Заметим, что за время выполнения суммирования в одном разряде триггер-счетчик должен переключиться до 3 раз. Четвертое переключение необходимо при подготовке к суммированию: триггер-счетчик необходимо погасить (установить в «0»); но если далее к сумме первых двух чисел нужно добавить третье, четвертое число и т. д., то добавление каждого нового числа потребует максимум двух переключений триггера, потому что к началу каждого такого сложения цифра одного из слагаемых уже находится в счетчике. В связи с этим проигрыш в скорости при выполнении многократных суммирований (скажем, в ходе умножения или деления) по сравнению с комбинационными сумматорами получается не слишком большим; экономия же в количестве оборудования весьма заметна.

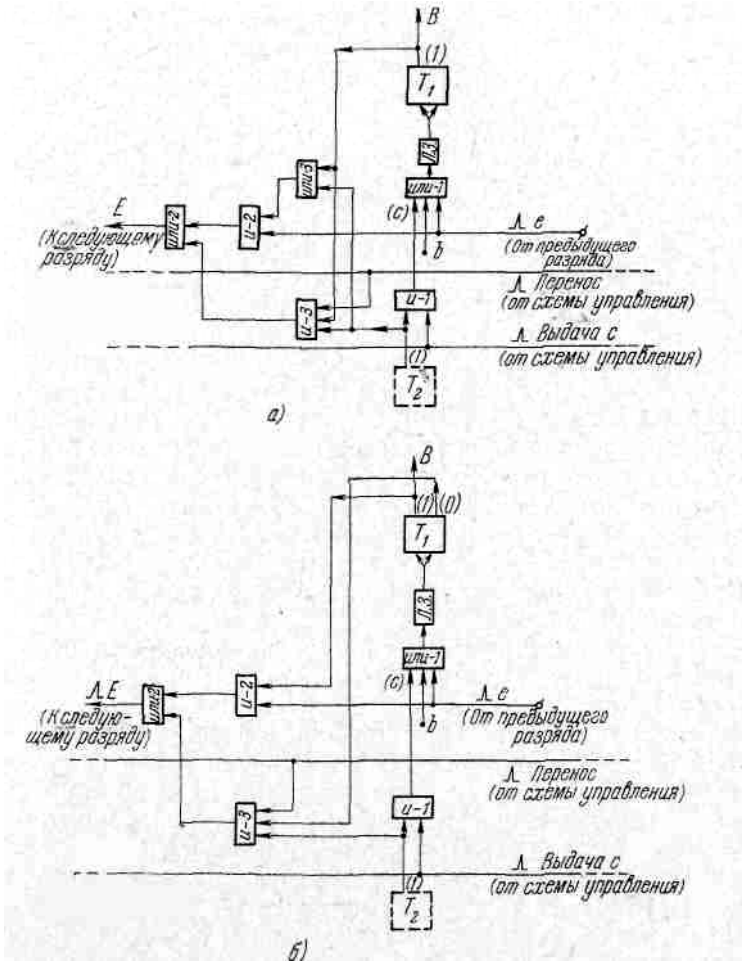


Рис. 2-45. Построение двоичного накапливающего сумматора (один разряд): *а)* вариант, в котором команда «перенос» следует до команды «выдача с»; б) вариант, в котором команда «перенос» следует после команды «выдача *с*».

Последние соображения справедливы, однако, только при использовании параллельной схемы, когда для каждого разряда имеется свой отдельный триггер-счетчик, в котором после выполнения однократного суммирования сохраняется цифра суммы данного разряда. В последовательной или последовательно-параллельной схеме накапливающие сумматоры, как правило, не применяются.

При использовании накапливающего сумматора в параллельной схеме принципиальное отличие по сравнению с комбинационными сумматорами имеются только в участках формирования цифр суммы *(B)*. Участки формирования сигналов переноса *(E)* строятся обычно, как и в комбинационных сумматорах, из логических элементов «и», «или» и «нет» или каких-нибудь других. Впрочем, конкретное построение участка переносов, как мы сейчас увидим, может оказаться проще, чем в комбинационном сумматоре.

Два различных построения двоичных накапливающих сумматоров представлены на рис. 2-45.

Рассмотрим сначала схему рис. 2-45, *а.*

Основной частью одноразрядного накапливающего сумматора является триггер-счетчик T_1. На его счетный вход через элемент «или-1» последовательно подаются импульсные сигналы: *b* (первое слагаемое), *e* (перенос из предыдущего разряда), *c* (второе слагаемое). Конечное состояние триггера T_1 дает цифру суммы *B*. Цепь подачи цифры первого слагаемого (*b*) на вход «или-1» на рисунке не показана. Не показана и цепь гашения счетчика.

Второе слагаемое (c) хранится в триггерном регистре, которому принадлежит триггер T_2. Импульс, соответствующий цифре *c,* формируется элементом «и-1» в момент подачи на все разряды сумматора команды «выдача с».

В промежутке между выдачей на счетчик цифр первого и второго слагаемых схема управления подает на все разряды одновременно импульс-команду «перенос». В это время по всем разрядам происходит формирование и распространение сигналов двоичного переноса.

Участок формирования сигналов переноса на рис. 2-45, *а* построен по аналогии с участком переноса комбинационного сумматора, показанного на рис. 2-8 (стр. 66). Можно было бы применить и какое-нибудь другое построение. Отличия от схемы рис. 2-8 состоят в том, что здесь поменялись местами входы (это, разумеется, никакой роли не играет) и что в элемент «и-3» введен дополнительный импульсный сигнал-команда «перенос». Последнее преследует цель получения сигнала переноса в виде короткого импульса в требуемый момент времени. Что касается сигналов слагаемых *b* и *c*, то они в это время могут быть получены в виде уровней напряжения соответственно с триггеров T_1 и T_2. С выходов триггеров напряжения подведены к выходам схемы формирования переноса. Чтобы состояние триггера Ti не успело измениться под влиянием входного сигнала переноса *е* прежде, чем этот сигнал пройдет или не пройдет через элемент «и-2», в цепи запуска T_1 имеется линия задержки Л. 3. Практически, однако, во многих случаях можно обойтись и без этой линии задержки.

Если рассматриваемую схему накапливающего сумматора сравнивать с теми схемами комбинационных сумматоров, которые приводились в 2.2.1 (рис. 2-4—2-10), то нужно учесть, что на указанных рисунках были опущены триггеры регистров, из которых на комбинационный сумматор поступают слагаемые, и цепи передачи суммы из комбинационного сумматора в триггерный регистр. Если принять это во внимание, то окажется, что почти все элементы, которые в накапливающем сумматоре используются для образования суммы (триггеры T_1 и T_2, элемент «или-1», линия задержки Л. 3.), необходимы и при использовании комбинационного сумматора. Но при его использовании все эти элементы выполняли вспомогательные функции, здесь же они заменяют логические элементы схемы формирования суммы.

Рис. 2-45, *б* иллюстрирует возможность сокращения количества оборудования не только в участке формирования суммы, но и в участке образования сигналов переноса. В схеме рис. 2-45, *а* формирование сигналов переноса выполнялось в тот момент времени, когда в одном из триггерных регистров (триггер T_1) находилось первое слагаемое *b*, в другом (триггер T_2)— второе слагаемое с; эти сигналы вместе с входным сигналом переноса в данный разряд (*е*) подавались на схему формирования сигналов переноса в следующий разряд (*E*); поскольку в отношении входных сигналов положение здесь получалось таким же, как в комбинационных сумматорах, схема формирования сигналов *E* тоже строилась по аналогии с комбинационными сумматорами. В действительности же формирование переноса выгоднее выполнять в тот момент времени, когда в триггере T_4 уже имеется сумма по модулю 2 цифр слагаемых данного разряда *b* и *c*. Если эта сумма равна нулю, а второе слагаемое равно единице, то это означает, что и первое слагаемое равно единице, т. е. что перенос в следующий разряд равен единице независимо от того, имеется ли перенос в данный разряд. Если сумма цифр слагаемых равна единице, то перенос в следующий разряд должен передаваться при условии, что имеется перенос в данный разряд. Именно эти правила реализованы в построении участка переноса в схеме рис. 2-45, *б*. Если бы все элементы участка переносов строились из диодов, то в схеме рис. 2-45, *б* для него потребовалось бы 7 диодов, а в схеме рис. 2-45, *а* — 9 диодов. Однако преимущество схемы рис. 2-45, *б* состоит еще и в том, что ни при каком сочетании цифр слагаемых в ней невозможно появление импульсов сразу на обоих выходах элемента «или-2». В схеме рис. 2-45, *а* в комбинации $b = c = e = 1$ это возможно. При этом импульс *е* может запоздать по времени относительно команды «перенос» (из-за задержек переноса в предыдущих разрядах), в результате чего на выходе *E* появились бы два импульса вместо одного; поступая на счетный вход триггера следующего разряда, лишний импульс мог бы привести к ошибкам в работе сумматора. Поэтому в действительности в схеме рис. 2-45, *а* могут потребоваться при определенных соотношениях между длительностями импульсов, задержками переносов и инерционностью триггеров некоторые усложнения. Указанный недостаток этой схемы можно устранить, например, заменив элемент «или-3» ($b + c$) схемой несовпадения ($bc + bc$); но тогда вместо 9 диодов в участке формирования сигналов переноса оказалось бы 13 диодов — вместо семи в схеме рис. 2-45, *б*. С более общей точки зрения необходимо признать, что накапливающие сумматоры, ориентирующиеся в основном на импульсно-потенциальную технику и на счетные входы триггеров, в настоящее время почти утратили свое значение. В принципе можно использовать для накапливающего сумматора и, скажем, счетчики с потенциальными связями или счетчики, построенные из каких-нибудь других элементов. Но тогда накапливающий сумматор получился бы, вероятно, не проще, а сложнее, чем комбинационный. Особого смысла в этом не видно.

2.6.5. Десятичный накапливающий сумматор.

Десятичный накапливающий сумматор, схема которого приведена на рис. 2-46, в настоящее время представляет в основном исторический интерес. Это—в упрощенном виде построение сумматора первой электронной цифровой вычислительной машины ENIAC.

Построение рис. 2-46 во многом похоже на построение двоичного накапливающего сумматора, приведенное на рис. 2-45, *б*. Главное отличие состоит в том, что вместо 2-позиционного триггера-счетчика T_1 здесь применено 20-триодное десятичное кольцо, являющееся счетчиком импульсов по модулю 10.

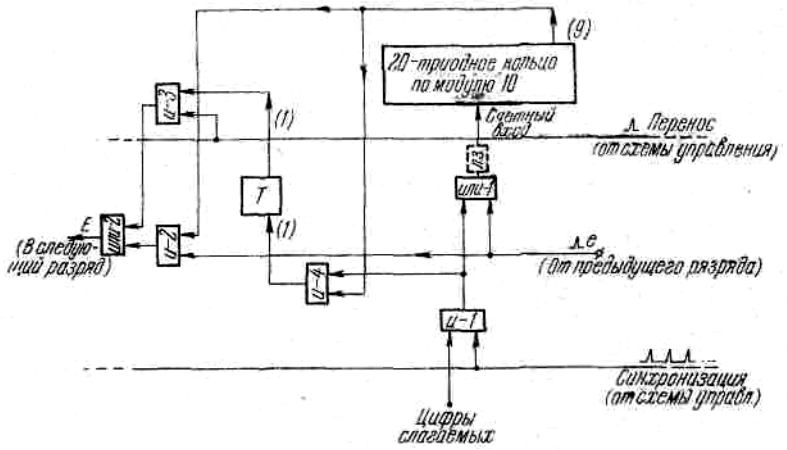


Рис. 2-46. Упрощенная схема десятичного накапливающего сумматора машины ENIAC (один разряд).

Как и в двоичном сумматоре, на счетный вход кольца через элемент «или-1» поступают сначала импульсы, соответствующие цифрам слагаемых, затем импульс переноса *e* из предыдущего разряда. Импульсы синхронизации играют здесь такую же роль, как импульс «выдача с» в схеме рис. 2-45, *б.* Но если в двоичном сумматоре по конечному состоянию триггера-счетчика после подачи на него цифр слагаемых сразу можно было определить, возникал ли в процессе счета перенос в следующий разряд, то здесь это сделать затруднительно. Для запоминания того, что в процессе счета кольцо переходило из последнего (9-го) положения в нулевое, предусматривается специальный триггер *Т.* Триггер устанавливается в положение «1» через элемент «и-4» в том случае, когда кольцо находится в положении «9» и на его счетный вход поступает импульс. Ясно, что при суммировании одной пары десятичных цифр такое положение может возникнуть только один раз.

Команда «перенос» вырабатывается схемой управления после того, как на входы кольца переданы уже все импульсы, соответствующие цифрам слагаемых. Импульс переноса в следующий разряд *Е* возникает либо в том случае, когда в процессе счета цифр слагаемых кольцо переходило из 9-го положения в нулевое (элемент «и-3»), либо в том случае, когда кольцо находится в 9-м положении и приходит импульс переноса из предыдущего разряда (элемент «и-2»). При этом в любой комбинации цифр слагаемых сигнал переноса в следующий разряд может сформироваться только в одном из этих двух каналов: если в процессе суммирования цифр слагаемых кольцо переходило из 9-го положения в нулевое, то к концу оно может находиться максимум в 8-м положении (потому что даже при максимальных цифрах слагаемых имеем 9 + 9=18). Поэтому, как и в схеме рис. 2-45, *б,* сдваивание импульса на входе элемента «или-2» невозможно.

Интересно, что в этой схеме — одной из самых первых схем сумматоров[*]) —имеется уже в каждом разряде специальный триггер для запоминания «местного» переноса, т. е. переноса, возникающего при суммировании цифр слагаемых в данном разряде без учета переносов из младших разрядов. В последующих разработках сумматоров, в том числе и двоичных, тоже по тем или иным соображениям иногда предусматривались такие триггеры. Однако возможности, которые получаются при их наличии, были в полной мере осознаны только значительно позже (см. разделы 4.3, 5.2**).**

2.7. Амплитудные сумматоры и сумматоры промежуточных типов

2.7.1. Амплитудные сумматоры.

Амплитудные сумматоры часто предлагались в первые годы развития цифровой техники. Они описываются во всех книгах по электронным цифровым машинам. Но при этом всегда указывается, что схемы амплитудного типа в принципе значительно менее надежны, чем комбинационные или накапливающие сумматоры и поэтому практического значения не имеют. Однако в технике иногда бывает, что на новом этапе возрождаются те идеи, которые прежде считались непригодными или неосуществимыми.

Имеет смысл поэтому рассмотреть схемы, приводимые на рис. 2-47 и 2-48, хотя схемы эти очень старые. На рис. 2-47 показано одно из возможных построений одноразрядного двоичного сумматора амплитудного типа. К входам *b* и с в виде импульсов подводятся сигналы цифр слагаемых, к входу *e* — импульс переноса из предыдущего разряда. Предполагается, что все эти импульсы положительные, одинаковой амплитуды, длительности и формы и поступают одновременно. При этом цифра «1» изображается наличием импульса, «0» — отсутствием импульса.

Сопротивления R_1, R_2, и R_3, на которые поступают входные сигналы, все одинаковы между собой. Они образуют звезду, причем в точке соединения сопротивлений в момент действия входных сигналов может получиться один из четырех случаев:

[*]) Описанная здесь схема появилась не в первоначальном варианте машины ENIAC, а была введена в нее при модернизации.

— отсутствие импульса (если все три входные цифры нули);
— импульс малой амплитуды $u_{имп.1}$ (если среди входных цифр имеется одна единица и два нуля);
— импульс средней амплитуды $u_{имп.2} > u_{имп.1}$ (если среди входных цифр имеются две единицы и один нуль);
— импульс большой амплитуды $u_{имп.3}$ (если все три входные цифры единицы).

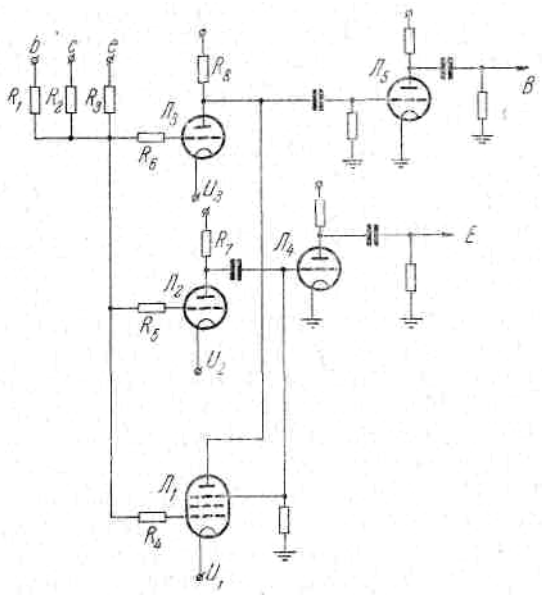


Рис. 2-47. Принцип построения одноразрядного двоичного сумматора амплитудного типа.

Напряжения на катодах ламп $Л_1$, $Л_2$ и $Л_3$ (U_1, U_2 и U_3) подобраны так, что в отсутствие импульса на их сетках все лампы заперты, при наличии импульса малой амплитуды $u_{имп.\ 1}$ отпирается по управляющей сетке лампа $Л_1$, но остаются запертыми лампы $Л_2$ и $Л_3$, при наличии импульса средней амплитуды $u_{имп.\ 2}$ открываются по управляющим сеткам лампы $Л_1$ и $Л_2$, но остается запертой лампа $Л_3$, при наличии на сетках импульса большой амплитуды $u_{имп.\ 3}$ отпираются все три лампы. Для этого необходимо:

$$0 < U_1 < u_{имп.1} < U_2 < u_{имп.\ 2} < U_3 < u_{имп.3}.$$

Причем величина U_1 и разности ($U_2 — u_{имп.1}$) и ($U_3 — u_{имп.\ 2}$) должны превышать напряжение запирания ламп по управляющим сеткам. Сопротивления R_4, R_5 и R_6 поставлены для того, чтобы отпирание одной из ламп по управляющей сетке и появление сеточных токов не привело бы к значительному уменьшению амплитуды импульса в точке соединения сопротивлений R_1, R_2 и R_3 и не помешало бы отпиранию следующей лампы.

Ясно, что отпирание лампы $Л_2$ и появление импульса тока в ее анодной цепи происходит в том и только в том случае, когда должен быть перенос в следующий двоичный разряд (среди входных цифр *b, c* и *e* имеется две или три единицы). При этом на аноде $Л_2$ появляется отрицательный импульс напряжения; лампа $Л_4$ усиливает и изменяет полярность этого импульса и выдает сигнал переноса E в следующий разряд.

Отрицательный импульс с анода лампы $Л_2$ поступает также на антидинатронную сетку $Л_1$. Поэтому, хотя лампа $Л_1$ открывается по управляющей сетке и малым, и средним, и большим импульсами напряжения, импульс тока в ее анодной цепи появляется только в том случае, когда среди входных цифр *b, c, e* имеется одна и только одна единица; если среди входных цифр имеется 2 или 3 единицы, то одновременно с отпиранием лампы $Л_1$ по управляющей сетке происходит запирание ее по антидинатронной сетке.

Так как аноды ламп $Л_1$ и $Л_3$ соединены параллельно, то на общем сопротивлении анодной нагрузки R_8 отрицательный импульс напряжения появляется либо в том случае, когда лампа $Л_1$ дает импульс тока, либо в том случае, когда лампа $Л_3$ дает импульс тока, т. е. когда среди входных цифр имеется нечетное количество единиц (одна или три). Лампа $Л_5$ усиливает и изменяет полярность этого импульса и выдает цифру суммы B для данного разряда.

Амплитудный сумматор, как видим, очень прост и по идее, и в отношении количества оборудования. По скорости он не уступает сумматорам комбинационным и превосходит накапливающие сумматоры. И все же схема рис. 2-47 никогда, по-видимому, не использовались на практике, хотя известна очень давно*). Недостатки ее, вероятно, уже ясны читателю; мы скажем о них несколько позже.

Рассмотрим сначала еще одну схему амплитудного типа. На рис. 2-48 показан участок формирования сигналов переноса в сумматоре одной из первых советских электронных цифровых машин М-1; в участке формирования цифры суммы амплитудный принцип не использовался.

Схема содержит две одинаковые звезды из сопротивлений (R_1 — R_2—R_3 и R_4 — R_5 — R_6). Одна из них

*) См. «Быстродействующие вычислительные машины», перев. с англ. под ред. Д. Ю. Панова, ИЛ, 1952, стр. 252.

соединена с управляющей сеткой левого триода лампы *Л*1; на нее подаются потенциальные сигналы (в виде уровней напряжения) двух цифр слагаемых данного разряда *b* и *c* и цифры переноса в данный разряд. К другой звезде, соединенной с сеткой правого триода $Л_1$ подведены инверсии этих сигналов. Как входные сигналы *b, c* и *e,* так и их инверсии являются выходными напряжениями статических триггеров; предполагается, что высокий уровень напряжения соответствует сигналу «1», а низкий уровень — сигналу «0».

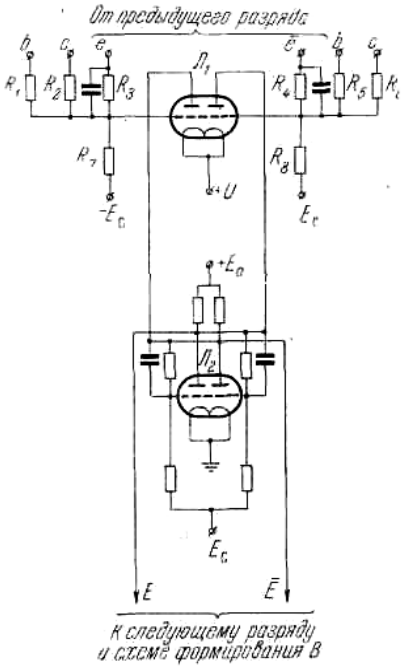


Рис. 2-48. Принцип построения участка переноса в сумматоре машины М-1.

В зависимости от комбинации цифр слагаемых и цифры переноса в данный разряд напряжение на выходе звезды может принимать одно из четырех значений: самое низкое—когда на всех трех входах напряжение низкое, промежуточное низкое — когда на одном из входов напряжение высокое, а на двух других низкие, промежуточное высокое—когда на двух из трех входов напряжения высокие, и самое высокое—когда на всех трех входах напряжения высокие. При этом выходные напряжения звезд парафазны: когда на левой звезде имеем самое низкое напряжение, на правой получается самое высокое напряжение, когда на левой промежуточное низкое — на правой промежуточное высокое и т. д.

Равные между собой сопротивления R_7 и R_8, напряжение смещения $—E_c$ и напряжение $+ U$ на катодах лампы $Л_1$ выбираются с таким расчетом, чтобы при самом низком и промежуточном низком напряжениях соответствующий триод был заперт по управляющей сетке, а при промежуточном высоком и самом высоком напряжении — открыт. При этом оказывается, что левый триод лампы $Л_1$ открыт по управляющей сетке в тех случаях, когда должен быть перенос в следующий разряд, а правый триод — когда переноса в следующий разряд нет. При любой комбинации входных переменных из двух триодов лампы $Л_1$ открыт по управляющей сетке только один.

Аноды триодов $Л_1$ как видно из рисунка, непосредственно соединены с анодами лампы триггера ($Л_2$). Положение триггера жестко связано с комбинацией входных сигналов на входах лампы $Л_1$: триоды $Л_1$ как бы стягивают триггер в определенное положение. Если при этом напряжение $+U$ на катодах триодов $Л_1$ выше, чем низкий уровень напряжения на аноде триггера, то в установившемся режиме оба триода лампы $Л_1$ не проводят: у одного из них напряжение на сетке отрицательно по отношению к катоду, у другого напряжение на аноде ниже, чем на катоде.

В первый момент после изменения комбинации входных сигналов может оказаться, что для одного из триодов лампы $Л_1$ и напряжение на сетке и напряжение на аноде выше, чем на катоде (если на том аноде триггера, с которым связан этот триод, напряжение было высоким). При этом в анодной цепи указанного триода появляется ток. Протекая по сопротивлению анодной нагрузки триггера, ток лампы $Л_1$ вызывает понижение напряжения на аноде триггера. Сам по себе этот ток, конечно, не мог бы понизить напряжение на аноде триггера особенно сильно; во всяком случае оно не могло бы стать ниже +U. Но когда триод лампы $Л_1$ перетянет напряжение на аноде триггера через точку неустойчивого равновесия, вступит в действие положительная обратная связь и триггер перебросится в соответствующее устойчивое состояние. Напряжение на аноде триггера, связанном с открытым по сетке триодом лампы $Л_1$, станет ниже, чем $+U$, и анодный ток этого триода прекратится. До следующего изменения комбинации входных сигналов оба триода лампы $Л_1$ не проводят.

Таким образом, в любом установившемся состоянии положение триггера указывает, имеется ли двоичный перенос в следующий разряд при существующей комбинации цифр слагаемых в данном разряде и переноса из предыдущего разряда. Выходные сигналы переноса *E* и $\bar{E}$ для следующего разряда и для схемы формирования цифры суммы *B* в данном разряде можно получить с выходов триггера.

Форсирующие емкости, которые шунтируют сопротивления R_3 и R_4, предназначены для ускорения распространения сигналов переноса вдоль цепочки разрядов (схема рис. 2-48 являлась частью параллельного сумматора).

Сравнивая рассматриваемую схему со схемой, приведенной на рис. 2-47, мы видим, что здесь приняты весьма радикальные меры к стандартизации выходных уровней напряжения в каждом разряде. Выходные напряжения триггера никак не связаны, конечно, с небольшими отклонениями напряжений, которые могут быть на сетках $Л_1$. Кроме входов триодов $Л_1$, во всей дальнейшей схеме последовательно выдержан двоичный характер: напряжение в любой точке может быть только либо «высоким», либо «низким», лампы могут быть либо полностью открыты, либо заперты и т. д. И все же при реальных разбросах характеристик ламп, при доступной в период проектирования М-1 точности сопротивлений (начальный разброс +10%), при разумных требованиях к стабильности источников питания получить достаточную надежность схемы затруднительно. Тем более трудно было бы обеспечить надежную работу схемы рис. 2-47 или какой-нибудь другой схемы амплитудного типа, спроектированной менее тщательно, чем схема рис. 2-48.

2.7.2. Промежуточные типы сумматоров.

Мы познакомились весьма полно и последовательно с тремя основными типами сумматоров: комбинационными, накапливающими и амплитудными. Вместе с тем имеются и такие сумматоры, относительно которых трудно решить, к какому из этих типов они в точности относятся.

На рис. 2-49 в качестве примера приведено одно из возможных построений двоичного сумматора промежуточного типа*). Построение это выполнено по аналогии со схемой

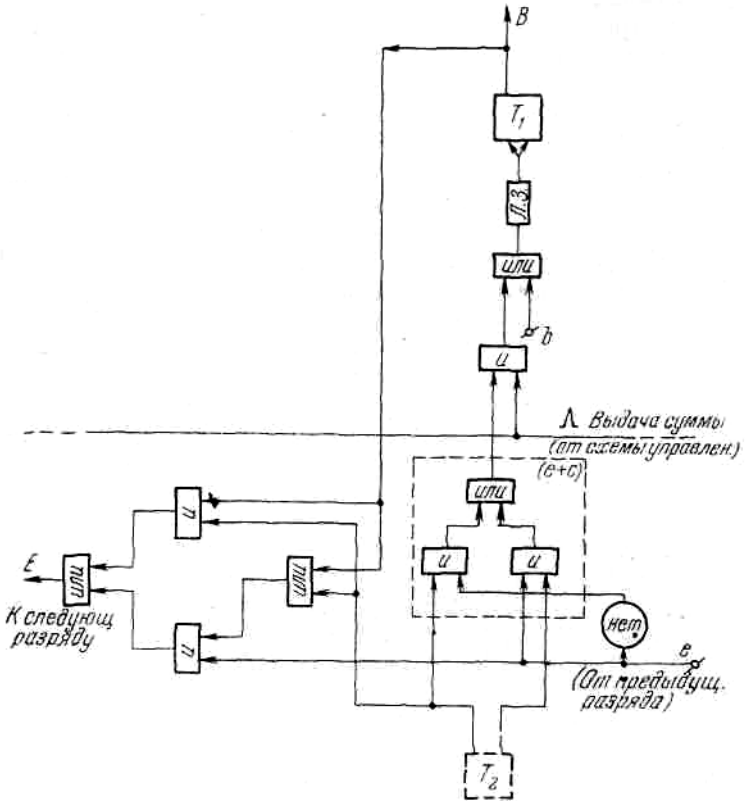


Рис. 2-49. Пример построения двоичного сумматора промежуточного типа (один разряд).

накапливающего сумматора, приведенной на рис. 2-45, *а* (стр. 104), но имеет одно весьма существенное отличие.

Если в схеме рис. 2-45, *а* на вход триггера-счетчика T_1 подавались последовательно импульсы первого слагаемого *b*, затем переноса *e* и затем второго слагаемого *c*, то здесь имеется специальный участок схемы, обведенный на рисунке пунктирной рамкой, который выполняет суммирование по модулю 2 цифр *e* и *c*. Сигналы переноса *e* могут теперь поступать в виде уровней напряжения; сумма по модулю 2 цифр *e и c* в момент поступления от схемы управления команды «выдача суммы» преобразуется в импульсный сигнал и поступает на вход триггера-счетчика T_1 где складывается по модулю 2 с цифрой первого слагаемого *b*.

Теперь за время суммирования триггер T_1 переключается не 3 раза, а всего 2 раза, если считать в том числе и прием в него цифры *b*; если первое слагаемое установлено в регистре триггеров-счетчиков заранее, то за время суммирования каждый такой триггер переключается всего один раз. В этом отношении схема рис. 2-49 ближе к комбинационным сумматорам. Да и вообще почти вся она построена по комбинационному принципу: и участок формирования сигналов переноса (*E*), и участок формирования суммы $e + c$; только для добавления $(e + c)$ к *b* применен принцип счета импульсов, характерный для накапливающих сумматоров.

Если регистры, связанные с параллельным сумматором, выполнены из статических триггеров и если один из этих регистров предназначен и для хранения одного из слагаемых до выполнения суммирования и для приема суммы, то схема сумматора рис. 2-49 удобнее, чем чисто комбинационная или чисто накапливающая схема.

Возможны и разнообразные другие схемы сумматоров промежуточного типа.

2.8. Сдвиг, передачи чисел и другие элементарные операции

2.8.1. Сдвиг.

Пусть имеем регистр из *m* цифровых элементов, предназначенный для хранения одного *m*-разрядного числа. Если цифровые элементы имеют по *n* устойчивых состояний, то каждый цифровой элемент хранит одну *n*-ичную цифру; для дальнейшего существенно, что число представлено однородной системой счисления. Пусть все цифровые элементы занумерованы в соответствии с нумерацией разрядов числа.

*) В общих чертах это построение напоминает схему сумматора вычислительной машины М-2. См. «Быстродействующая вычислительная машина М-2», под ред. И. С. Брука, Гостехиздат, 1957.

Например, если применяется позиционный способ изображения чисел и если все веса меньше единицы, то обычно считают старший разряд (с наибольшим весом) первым, следующий разряд — вторым,…, самый младший разряд (с наименьшим весом) — m-м.

Операция *сдвига вправо* (в сторону младших разрядов) на k разрядов состоит в том, что состояние 1-го цифрового элемента передается в $(k + 1)$-й цифровой элемент, состояние 2-го элемента — в $(k + 2)$-й и т. д. Соответственно при *сдвиге влево* (в сторону старших разрядов) на k разрядов состояние m-го цифрового элемента передается в $(m — k)$-й, $(m — 1)$-го — в $(m — k — 1)$-й и т. д. В некоторых случаях освобождающиеся при сдвиге k крайних разрядов регистра либо сохраняют прежние состояния, либо заполняются нулями, либо принимают новую информацию извне, а информация, выдвигаемая из k разрядов на другом краю регистра, либо теряется, либо передается во внешние цепи. В других случаях регистр может быть замкнут в кольцо, так что информация, выдвигаемая с одного края регистра, принимается в освобождающиеся цифровые элементы на другом краю регистра.

Если числа представлены позиционным способом с естественными весами разрядов, то сдвиг влево (вправо) на k разрядов эквивалентен с точностью до крайних разрядов умножению (делению) числа на n^k, где n — основание системы счисления. При других способах изображения чисел операция сдвига не имеет такого смысла.

Рассматривая ниже схемы для выполнения сдвигов, мы будем говорить только о сдвиге на один разряд. Регистр со схемой сдвигов на k разрядов можно рассматривать как k отдельных регистров, в каждом из которых имеется сдвиг на один разряд. Например, при выполнении сдвига на k разрядов вправо состояние 1-го разряда передается в $(k + 1)$-й, 2-го — в $(k + 2)$-й и т. д.; можно считать, что разряды 1-й, $(k + 1)$-й, $(2k + 1)$-й образуют как бы один частный регистр, разряды 2-й, $(k + 2)$-й, $(2k + 2)$-й, ... — второй и т. д.; выполнение сдвига на $к$ разрядов в исходном регистре эквивалентно выполнению сдвига на один разряд во всех частных регистрах одновременно.

Похожее положение имеет место в том случае, когда основание системы счисления n отлично от 2, но каждая n-ичная цифра кодируется с помощью нескольких двоичных цифр. Если, например, в десятичном регистре для изображения каждой десятичной цифры имеется по 4 двоичных триггера, то операция сдвига на один десятичный разряд эквивалентна сдвигу на 4 двоичных разряда.

Заметим еще, что в настоящем разделе мы говорим о пространственном сдвиге чисел. Рассматривая в последующих разделах выполнение операций в последовательных арифметических устройствах, мы встретимся с элементарной операцией сдвига в виде временного сдвига числа. Представим себе, что код числа передается по одному проводу, причем в определенный момент времени (в определенном такте) передается сигнал, соответствующий цифре младшего разряда, в следующий момент времени — сигнал, соответствующий цифре следующего по старшинству разряда и т. д. Сдвинуть число, скажем, на один разряд влево — это значит задержать его код на один временной интервал между цифрами, так чтобы сигнал, соответствующий младшей цифре, приходил в тот момент времени, который отведен для следующего по старшинству разряда, и т. д. По смыслу временной сдвиг в последовательном устройстве аналогичен пространственному сдвигу в параллельном устройстве, но техника выполнения этих операций различна.

Схемы для выполнения сдвигов различны для разных типов цифровых элементов.

На рис. 2-50 показаны три схемы сдвига в регистре из статических триггеров, построенные по импульсному и импульсно-потенциальному принципам. Во всех трех схемах показаны только цепи сдвига вправо. Однако — по крайней мере относительно схем рис. 2-50, *б* и 2-50, *в* — нетрудно сообразить, как нужно было бы усложнить устройство, чтобы в нем выполнялся и сдвиг влево и, если необходимо, сдвиг на несколько разрядов, и другие операции.

На рис. 2-50, *а* изображена простейшая схема сдвигов.

Для выполнения в ней однократного сдвига на все триггеры регистра одновременно подается импульс-команда «гашение». Если триггер некоторого, скажем i-го, разряда к моменту прихода этого импульса находился в состоянии «1», то команда «гашение» ставит его в положение «0», причем на одном выходе триггера напряжение меняется с высокого на низкое, на другом — с низкого на высокое. Дифференцирующая (укорачивающая) цепочка присоединена к одному из выходов триггера с таким расчетом, чтобы при переходе триггера из единичного состояния в нулевое она вырабатывала импульс той полярности, которая необходима для переключения аналогичного триггера по входу. (Предполагается, что к импульсам обратной полярности триггер не чувствителен.) Пройдя через линию задержки, выходной импульс дифференцирующей цепочки установит триггер следующего $((i + 1)$-го) разряда в положение «1». Если триггер i-го разряда к моменту прихода команды «гашение» находился в состоянии «0», то команда «гашение» не изменяет его состояния; в частности, остается постоянным напряжение на его выходах, и дифференцирующая цепочка никаких импульсов не выдает — следующий триггер остается в состоянии «0».

Таким образом, в схеме рис. 2-50, *а* сдвиг выполняется в 2 такта: сначала все триггеры регистра устанавливаются в «0», затем в тех разрядах, где это необходимо, устанавливаются единицы. Линии задержки, имеющиеся между разрядами схемы, должны обеспечивать задержку импульсов на один такт, с тем, чтобы триггер следующего разряда успел полностью установиться после гашения и мог бы переключиться вторично.

На рис. 2-50, *б* изображена однотактная схема сдвигов. В каждом разряде имеется два импульсно-

потенциальных вентиля «и». Один из них управляется напряжением с единичного выхода триггера предыдущего разряда, другой — напряжением с нулевого выхода. Когда на все элементы «и» подается одновременно по всем разрядам импульс-команда «сдвиг», в каждом разряде этот импульс проходит через один из двух вентилей. Поступая затем на вход триггера, он устанавливает триггер в соответствие с тем состоянием, которое имелось прежде в соседнем слева разряде.

Линии задержки между выходами и входами триггеров предназначены здесь лишь для того, чтобы задержать переброс триггера относительно команды «сдвиг» на длительность импульса, с тем, чтобы изменение напряжений на потенциальных входах элементов «и» не повредило бы правильному прохождению импульсов через них. Однако при реальных соотношениях между длительностью импульсов, паразитными задержками в схемах и длительностью фронтов триггеров линии задержки иногда не нужны.

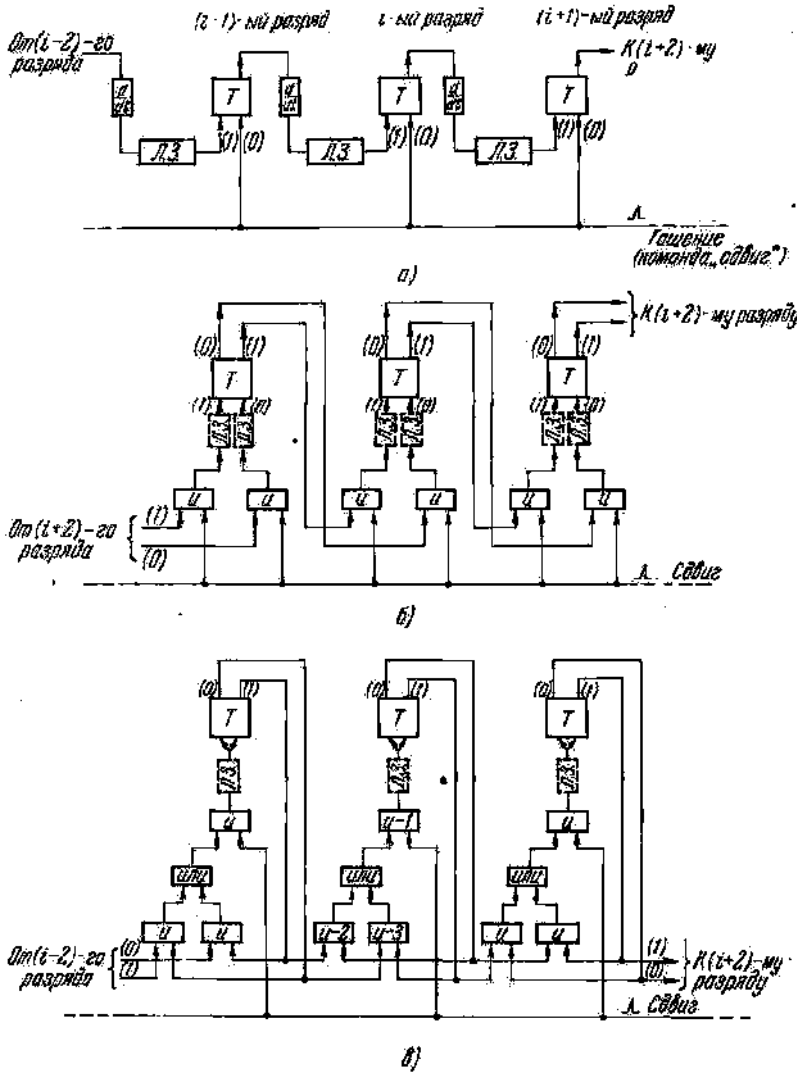


Рис. 2-50. Импульсные и импульсно-потенциальные схемы сдвигающих регистров: *а*) двухтактная схема; *б*) однотактная схема без использования счетных входов; *в*) однотактная схема с использованием счетных входов триггеров.

Схема рис. 2-50, *в* также представляет собой однотактную схему сдвигов, но в ней—в отличие от схемы рис. 2-50, *б* — использованы не раздельные, а счетные входы триггеров. В каждом разряде имеется не два, а только один импульсно-потенциальный вентиль — элемент «и-1». Однако перед его потенциальным входом включена дополнительная логическая схема из элементов потенциального типа «и-2», «и-3» и «или». Дополнительная схема устроена так, что вентиль «и-1» в некотором разряде открывается по потенциальному входу в том и только в том случае, когда состояние триггера в данном разряде не совпадает с состоянием триггера соседнего слева разряда. При этом импульс-команда «сдвиг», пройдя на счетный вход триггера, изменит его состояние на противоположное, т. е. установит триггер в соответствие с прежним состоянием соседнего слева разряда. Если данный триггер до команды «сдвиг» находился в соответствии с триггером соседнего слева разряда, то вентиль «и-1» заперт, и имеющееся состояние сохраняется. Линии задержки выполняют здесь такую же роль, как в схеме рис. 2-50, *б,* и так же могут оказаться излишними. Схема рис. 2-50, *в* применялась в старых машинах, когда потенциальные логические элементы были значительно проще, чем импульсно-потенциальные вентили, а триггеры все равно использовали счетные входы. Импульсно-потенциальные вентили в схеме рис. 2-50, *в* переключаются по потенциальному входу вдвое реже, чем в схеме рис. 2-50, *б.* Если непрерывно выполняются сдвиги, то в схеме

рис. 2-50, *б* напряжения на потенциальных входах вентилей изменяются с высоких на низкие и обратно в худшем случае в каждом такте — когда по регистру продвигается комбинация цифр ... 101010... В схеме рис. 2-50, *в* наиболее трудной с этой точки зрения является комбинация... 11001100..., в которой вентили переключаются один раз за два такта. Наряду с экономией в количестве импульсно-потенциальных вентилей это обстоятельство рассматривалось как дополнительное преимущество схемы 2-50, *в* перед схемой 2-50, *б.*

Главная трудность, с которой мы вообще встречаемся при построении сдвиговых регистров, состоит в том, что при выполнении сдвига каждый разряд должен одновременно и выдавать информацию в следующий разряд и принимать новую информацию из предыдущего разряда. В рассмотренных схемах либо имелись линии задержки для временного хранения информации, либо приходилось рассчитывать на то, что фронты выходных напряжений триггеров несколько растянуты, и, следовательно, изменение выходных напряжений несколько запаздывает относительно входных сигналов. Последнее фактически означает, что мы рассчитываем на кратковременное сохранение информации паразитными емкостями.

Если не пользоваться запоминанием на реактивностях, то для выполнения сдвига в регистре недостаточно иметь по одному триггеру на разряд.

На рис. 2-51 показана схема сдвигающего регистра, в которой каждый разряд содержит два триггера. Один из этих триггеров (на рисунке —верхний) считается основным, другой — вспомогательным. Сдвиг выполняется в 2 такта. Сначала схема управления включает на некоторое время напряжение, соответствующее сигналу «1», на шину «передача на вспомогательный триггер»; затем это напряжение выключается и на некоторое время включается такое же напряжение на шину «сдвиг». В первом такте происходит передача числа из основных триггеров на вспомогательные, во втором то же число возвращается со сдвигом на 1 разряд в основные триггеры.

Схема получается сравнительно сложной. Однако для того, чтобы в том же регистре могли выполняться также сдвиги влево, сдвиги на несколько разрядов и т. д., далее требуются лишь незначительные усложнения в логических схемах на входах основных триггеров.

Идея схемы 2-51 воспроизводится также в построениях сдвиговых регистров на двухтактных магнитных элементах (феррито-диодных, феррито-транзисторных и др.). Сдвиговые регистры из однотактных магнитных ячеек, а также из динамических триггеров с линиями задержки напоминают по идее схему рис. 2-50, *а,* однако элементы задержки входят при этом непосредственно в состав цифрового элемента.

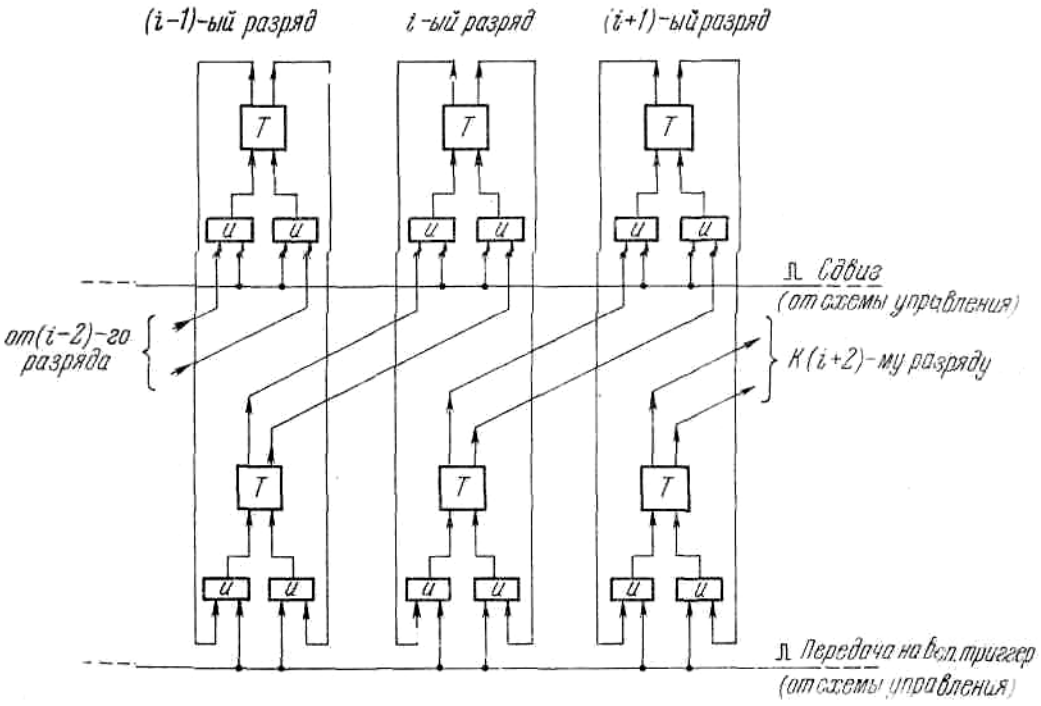


Рис. 2-51. Сдвигающий регистр с потенциальными связями, содержащий по 2 триггера на разряд

Иногда с целью уменьшения количества оборудования по сравнению со схемой рис. 2-51 идут на уменьшение скорости выполнения сдвигов. Можно, например, сократить в $1^1/_2$ раза количество триггеров, но выполнять при этом сдвиг не в 2, а в 3 такта. Устройство такого регистра схематически показано на рис. 2-52. Вспомогательные триггеры имеются не во всех разрядах, а через один — скажем, во всех четных разрядах. Для выполнения сдвига информация сначала передается из четных основных триггеров во вспомогательные, затем из нечетных основных триггеров — в соседние четные триггеры, затем из вспомогательных триггеров — в нечетные триггеры. Цифры у стрелок на рисунке показывают последовательность передач. В результате вся информация в основных триггерах оказывается сдвинутой.

Если каждый триггер содержит по 2 инвертора (элемента «нет»), то всего в схеме рис. 2-52 приходится в

среднем по 3 инвертора на разряд

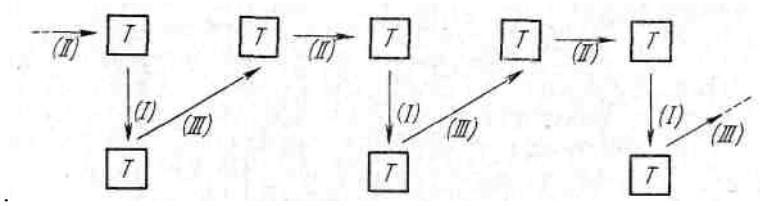


Рис. 2-52. Принцип построения сдвигающего регистра с потенциальными связями, содержащего по одному вспомогательному триггеру на пару разрядов.

С тем же количеством инверторов можно построить более изящную схему сдвигающего регистра — так называемый *регистр с перемещающимися триггерами.* Построение такого регистра показано на рис. 2-53*).

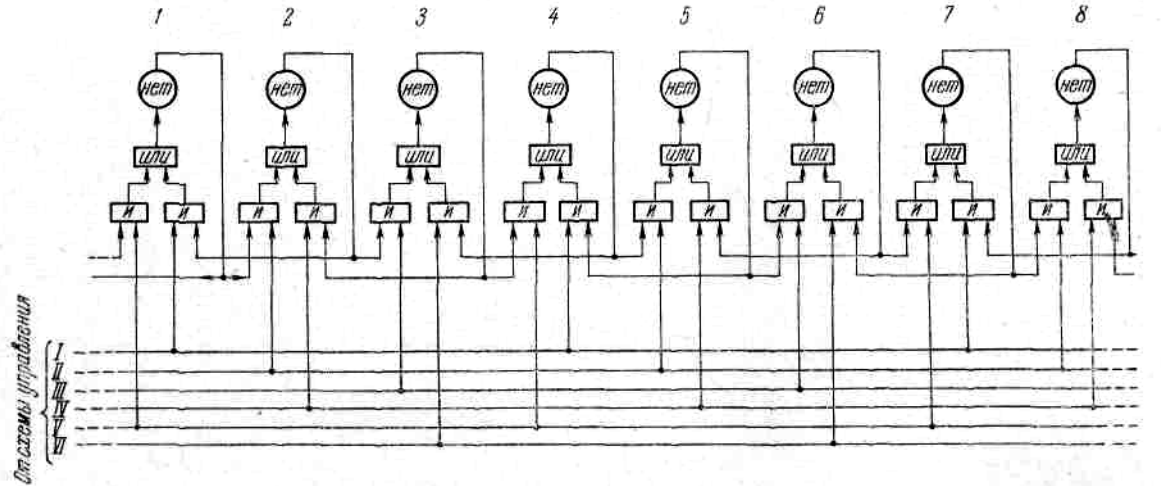


Рис. 2-53. Сдвигающий регистр с перемещающимися триггерами.

Пусть все инверторы занумерованы по порядку: ..., 1, 2, 3, 4, 5, 6, 7, 8, ... Через логические элементы «и» выход каждого *i*-го инвертора соединяется со входами двух соседних инверторов—(*i*— 1)-го и (*i*+l)-го. Но связи эти замыкаются только при наличии определенных сигналов, поступающих от схемы управления по шести шинам (I, II, III, IV, V и VI). Сигналы управления разведены по логическим схемам с периодичностью 3; это значит, что те сигналы, которые управляют связями 1-го инвертора, точно так же управляют связями 4-го, 7-го, 10-го, ... инверторов; те сигналы, которые управляют связями 2-го инвертора, точно так же разведены на 5-й, 8-й, 11-й,... инверторы и т. д.

Допустим, что в начальном положении имеются сигналы управления (высокие уровни напряжения) на шинах I и II. Это означает, что замкнуты перекрестные связи между инверторами..., 1 и 2, 4 и 5, 7 и 8, ...; указанные пары образуют 2-позиционные триггеры, в которых хранится исходное число, а инверторы..., 3, 6, 9... свободны. Начальное положение регистра схематически представим в виде

$$(\mathrm{I,II})\ldots 1 \rightleftarrows 2 \quad 3 \quad 4 \rightleftarrows 5 \quad 6 \quad 7 \rightleftarrows 8 \ldots ,$$

где стрелками указаны замкнутые связи.

Выполнение сдвига вправо начинается с включения высокого напряжения на шину III. При этом замыкаются связи со 2-го инвертора на 3-й, с 5-го на 6-й, с 8-го на 9-й, ...; 3-й инвертор устанавливается в такое же положение, как и 1-й (потому что оба они работают теперь от выходного сигнала 2-го инвертора), 6-й — в такое же положение, как 4-й, и т. д. Положение регистра можно схематически представить в виде

$$(\mathrm{I, II, III})\ldots 1 \rightleftarrows 2 \rightarrow 3 \quad 4 \rightleftarrows 5 \rightarrow 6 \quad 7 \rightleftarrows 8 \rightarrow \ldots$$

Затем включается высокое напряжение на шину IV, которое замыкает связи с 3-го инвертора на 2-й, с 6-го на 5-й, с 9-го на 8-й и т. д. Теперь получаем несимметричные триггеры, в которых одно плечо образовано одним инвертором, другое — соединенными параллельно двумя инверторами (скажем, в одном плече — 2-й инвертор, в другом — 1-й и 3-й и т. д.):

$$(\mathrm{I, II, III, IV})\ldots 1 \rightleftarrows 2 \rightleftarrows 3 \quad 4 \rightleftarrows 5 \rightleftarrows 6 \quad 7 \rightleftarrows 8 \rightleftarrows \ldots$$

Далее выключаются сигналы с шин I и II и включается сигнал на шину V. При этом состояние регистра становится следующим:

$$(\mathrm{III, IV, V})\ldots \rightarrow 1 \quad 2 \rightleftarrows 3 \rightarrow 4 \quad 5 \rightleftarrows 6 \rightarrow 7 \quad 8 \rightleftarrows \ldots$$

*) См. К а р ц е в М. А., Принцип подвижных блокировок при построении схем электронных цифровых машин, ДАН СССР, 1960, 135, № 5, стр. 1064—1067.

Положение теперь аналогично тому, которое имелось в начале выполнения сдвига (когда сигналы имелись на шинах I, II и III), но все триггеры вместе с хранившейся в них информацией сместились на один инвертор вправо. Следующие эволюции выглядят так:

$$(\text{III, IV, V, VI})\ldots \rightleftarrows 1 \quad 2 \rightleftarrows 3 \rightleftarrows 4 \quad 5 \rightleftarrows 6 \rightleftarrows 7 \quad 8 \rightleftarrows \ldots$$
$$(\text{V, VI, I})\ldots \rightleftarrows 1 \rightarrow 2 \quad 3 \rightleftarrows 4 \rightarrow 5 \quad 6 \rightleftarrows 7 \rightarrow 8 \ldots$$
$$(\text{V, VI, I, II})\ldots \rightleftarrows 1 \rightleftarrows 2 \quad 3 \rightleftarrows 4 \rightleftarrows 5 \quad 6 \rightleftarrows 7 \rightleftarrows 8 \ldots$$
$$(\text{I, II})\ldots \rightleftarrows 1 \rightleftarrows 2 \quad 3 \quad 4 \rightleftarrows 5 \quad 6 \quad 7 \rightleftarrows 8 \ldots$$

Последнее положение регистра совпадает с начальным, но вся информация переместилась вместе с триггерами на 3 инвертора (т. е. на один двоичный разряд) вправо.

Если вслед за первым сдвигом предполагается выполнять следующий сдвиг, то на последнем этапе вместе с выключением управляющих сигналов V и VI можно было бы включить сигнал III; тогда на последнем этапе схема оказалась бы следующей:

$$(\text{I, II, III})\ldots 1 \rightleftarrows 2 \rightarrow 3 \quad 4 \rightleftarrows 5 \rightarrow 6 \quad 7 \rightleftarrows 8 \rightarrow \ldots$$

как это и требуется в начале сдвига. Таким образом, однократный сдвиг выполняется здесь в 6 этапов.

Однако по быстродействию эта схема все же не уступает схеме рис. 2-52. В регистре рис. 2-52 длительность каждого такта должна быть равна длительности одного фронта управляющего напряжения плюс две задержки в звене «логическая схема — инвертор» (для того чтобы за время управляющего напряжения успел полностью переключиться триггер, составленный из двух звеньев «логическая схема — инвертор»). Если время фронта обозначить через τ_1, а задержку в одном звене—через τ_2,то полное время выполнения сдвига в регистре рис. 2-51 равно $3\tau_1 + 6\tau_2$. В регистре рис. 2-53 длительность первого этапа (включение управляющего напряжения III и одновременное выключение управляющих напряжений V и VI) должна составлять $1\tau_1 + 1\tau_2$, длительность второго этапа (включение управляющего напряжения IV) $1\tau_1$, третьего этапа (включение V и выключение I и II) $1\tau_1 + 1\tau_2$ и т. д.; полная длительность всех шести этапов оказывается равной $6\tau_1 + 3\tau_2$. Так как обычно $\tau_1 < \tau_2$, регистр рис. 2-53 оказывается более быстродействующим, чем регистр рис. 2-52.

Заметим еще, что для выполнения сдвига влево в регистре рис. 2-53 не потребовалось бы никаких дополнительных усложнений аппаратуры. Для этого достаточно лишь предусмотреть включение тех же управляющих напряжений в другом порядке.

Мы рассмотрели, таким образом, ряд способов выполнения сдвига в регистрах, составленных из 2-позиционных цифровых элементов. Методы выполнения сдвига в регистрах, составленных из многопозиционных колец, не содержат каких-либо новых идей. Если *n*-позиционные кольца, входящие в регистр, имеют по *n* раздельных входов, то можно построить либо импульсно-потенциальную схему сдвига — по аналогии с рис. 2-50, *б,* либо схему с потенциальными связями — по аналогии с рис. 2-51 или 2-52. При использовании колец, в которых внутренние обратные связи замкнуты через логические схемы, возможно построение схемы сдвигов по аналогии с рис. 2-53.

2.8.2. Передачи чисел. Логическое умножение и логическое сложение.

Передача чисел из одного регистра в другой — это операция, во многом похожая на операцию «сдвиг». При сдвиге тоже фактически происходит передача цифр из одних цифровых элементов в другие, но при этом каждый разряд, выдавая информацию в соседний, должен одновременно принять информацию от другого соседа. При передаче чисел этого, как правило, нет, и одни цифровые элементы (одного регистра) только выдают информацию, а другие (элементы другого регистра) — только принимают ее.

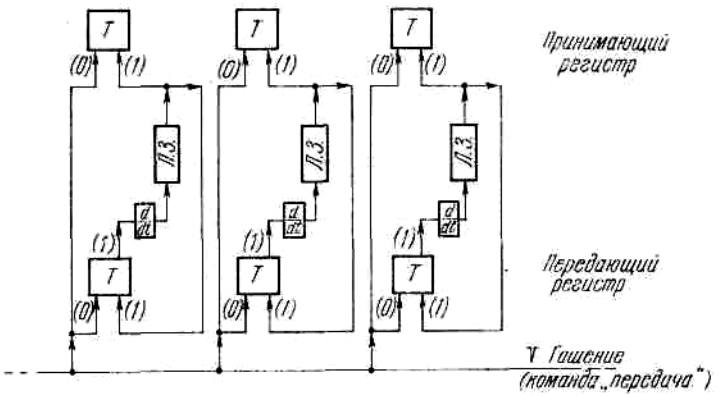


Рис. 2-54. Двухтактная схема передач чисел между регистрами (3 разряда).

Для выполнения передач чисел поэтому подходят все те схемы, которые рассмотрены были нами для операции «сдвиг», однако в ряде случаев в них возможны более или менее существенные упрощения.

В частности, очень простую двухтактную схему для передачи чисел можно построить по аналогии со схемой рис. 2-50, *а.* Недостаток ее состоял бы в том, что для передачи числа передающий регистр нужно гасить, так что хранившаяся в нем информация не сохраняется. Если это необходимо, то указанный недостаток можно устранить, соединив выход линии задержки не только с единичным входом принимающего триггера, но и с единичным входом «своего» (передающего) триггера. Такая схема представлена на рис. 2-54. Принцип ее устройства очевиден из описания рис. 2-50, *а* (стр. 111).

На рис. 2-55 показан другой вариант двухтактной схемы передачи чисел. Для выполнения передачи числа в этой схеме должна быть сначала подана команда «гашение», которая устанавливает все триггеры принимающего регистра в положение «0», затем — команда «передача числа». При поступлении этой последней срабатывают

элементы «и» в тех разрядах, где триггеры передающего регистра содержат единицы, и устанавливаются в положение «1» триггеры соответствующих разрядов принимающего регистра.

Специальное построение триггера позволяет использовать схему рис. 2-55 как однотактную. Пример такого триггера из потенциальных элементов «и», «или» и «нет» имеется на рис. 2-56. Для определенности на рисунке принято, что сигналу «1» соответствует «высокий» уровень напряжения, сигналу «0» — «низкий».

Обратные связи в триггерах замкнуты через элементы «и». Нормально на шине «гашение», присоединенной ко вторым входам этих элементов, напряжение высокое, и сигнал с выхода левого инвертора передается на вход правого.

В момент передачи числа нужно одновременно подать низкое напряжение на шину «гашение» и высокое напряжение — на шину «прием числа». При этом обратные связи в триггерах принимающего регистра разрываются, и сигналы с выходов передающего регистра просто проходят через цепочки из двух последовательно соединенных инверторов; длительность команд «гашение» и «прием числа» должна быть не меньше, чем время прохождения сигнала по каждой из таких цепочек, причем сигнал «гашение» должен заканчиваться несколько раньше, чем сигнал «прием числа» (примерно на длительность фронта). Команды «гашение» и «прием числа» могут быть и более длительными — это никак не повлияет на скорость установления выходных сигналов триггеров принимающего регистра.

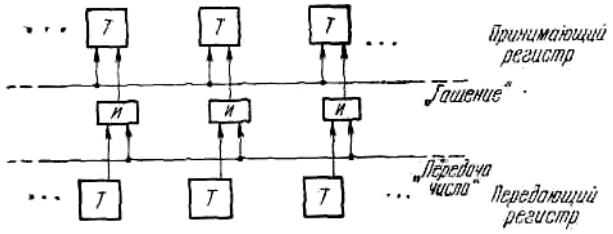


Рис. 2-55. Вариант двухтактной схемы передачи числа.

Более обычные однотактные схемы передачи чисел можно построить по аналогии со схемами сдвигов рис. 2-50, б и 2-50, *в.* При этом ни линий задержки, ни запоминания на паразитных реактивностях не требуется. Схема, построенная по аналогии со сдвигающим регистром рис. 2-50, *б,* в равной мере может использоваться и по импульсно-потенциальному, и по чисто потенциальному принципу. Такие схемы передачи чисел уже применялись нами в качестве части сдвигающего регистра (см. рис. 2-51).

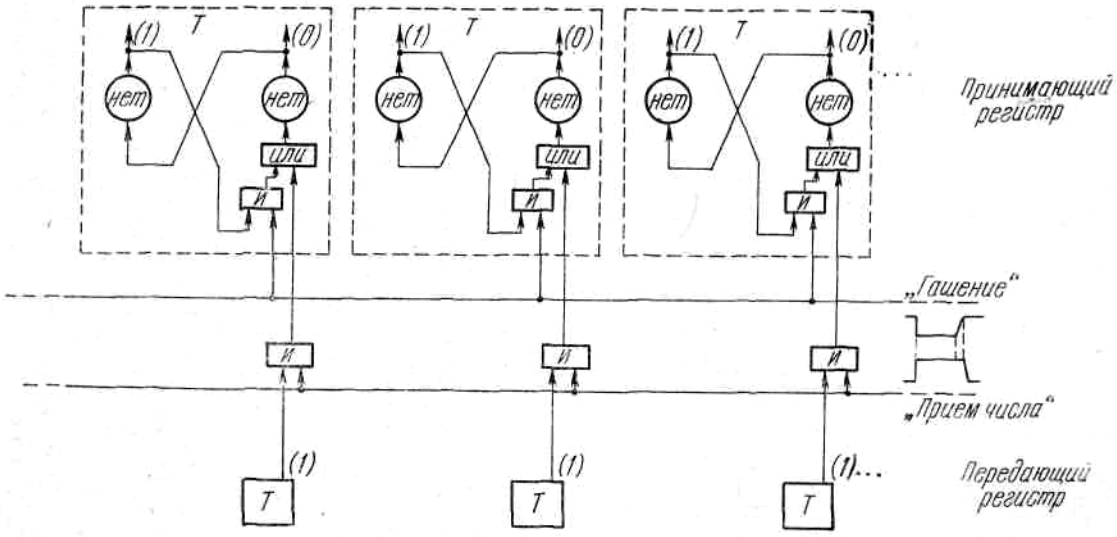


Рис. 2-56. Схема передачи чисел, аналогичная рис. 2-55, но со специальный построением триггеров (однотактная).

Если требуется выполнение перекрестных передач чисел между двумя регистрами, то никаких упрощений по сравнению со схемой сдвигов не получается. Вообще в схеме перекрестных передач каждая пара триггеров соответствующего разряда двух регистров представляет собой как бы 2-разрядный сдвигающий регистр, замкнутый в кольцо. Выполнение перекрестной передачи эквивалентно выполнению сдвига одновременно во всех таких регистрах.

Прежде чем покончить со схемами передачи чисел, обратим внимание на одно любопытное обстоятельство.

На рис. 2-57 представлена однотактная схема передачи чисел из одного регистра в другой, построенная по аналогии со сдвигающим регистром рис. 2-50, *б.* Об этой схеме мы только что говорили, однако на рис. 2-57 в нее внесено небольшое изменение: вместо одной общей шины, объединяющей все элементы «и», на которую должна подаваться команда «передача числа», здесь имеются две различные шины — для элементов «и», связанных с нулевыми входами триггеров, и для элементов, связанных с единичными входами. При передачах чисел из нижнего регистра в верхний теперь нужно будет подавать команды одновременно на обе шины. В связи с этим, может быть, потребуется лишний элемент формирования команды в схеме управления; зато появляется возможность без какого-либо дополнительного оборудования в регистре выполнять операции логического умножения и логического

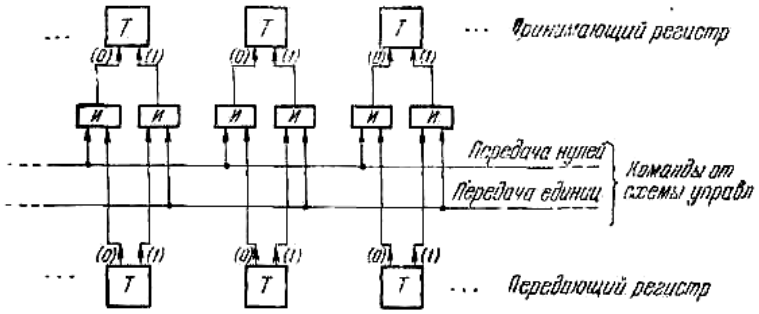


Рис.2-57 . Однотактная схема для передачи чисел, позволяющая выполнять также операции логического умножения и логического сложения (3 разряда).

сложения. Для этого нужно только установить исходные числа в «передающем» и «принимающем» регистрах и затем подать команду либо на шину «передача нулей», либо на шину «передача единиц». В первом случае в «принимающем» регистре получится логическое произведение исходных чисел, во втором случае — логическая сумма. Таким образом, используя ту или другую часть оборудования для передачи чисел, можно выполнять логическое умножение и логическое сложение.

Заметим еще, что мы говорили здесь все время только о параллельных передачах чисел — когда все разряды одного числа передаются одновременно. Если числа представлены последовательным кодом и, например, циркулируют в линиях задержки, то передачи чисел из одной линии задержки в другую осуществляются системой вентилей на входах и на выходах линий; каких-либо особых затруднений осуществление таких передач не вызывает.

2.8.3. Обращение кода числа.

Обращение кода числа состоит в том, что в каждом *n*-ичном разряде числа имеющаяся в нем цифра *a* заменяется на цифру [(*n* — 1) — *a*]. Например, в двоичной системе цифра «0» при обращении кода числа заменяется на «1», а цифра «1» — на «0»; в десятичной системе «0» заменяется на «9», «1» — на «8» и т. д.

В двоичной системе, если регистр составлен из счетчиков по модулю 2, то обращение можно выполнить, поместив в этот регистр исходное число и подав затем на входы всех разрядов по одному импульсу.

Если выходы «0» и «1» каждого двоичного триггера симметричны, то обращенный двоичный код можно получить, попросту переименовав эти выходы, т. е. считая нулевой выход единичным, а единичный выход нулевым.

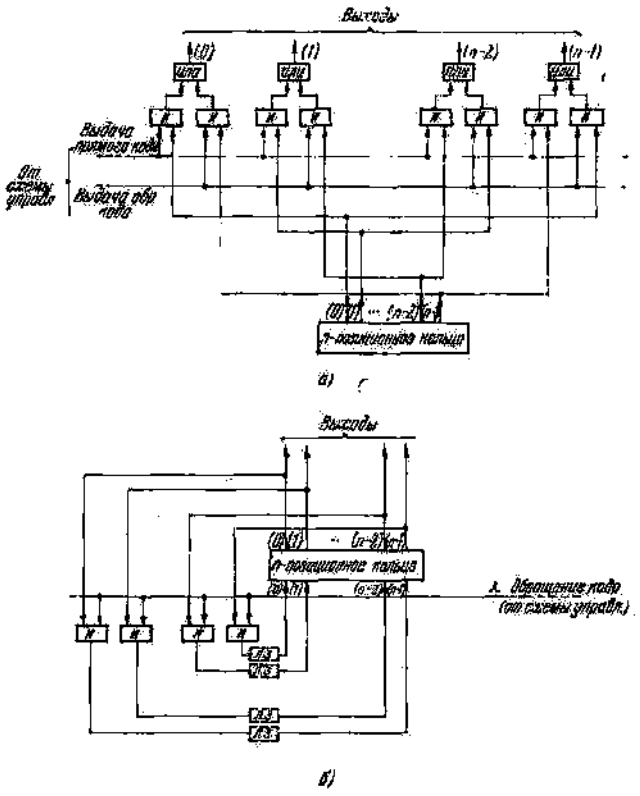


Рис. 2-58. Схемы для обращения кода числа при непосредственных способах изображения *n*-ичных цифр (один разряд): *а*) схема переименования выходов цифрового элемента; *б*) схема возврата обращенного кода в цифровой элемент.

Последний способ годится и для n-ичной системы счисления в том случае, когда применяются n-позиционные кольца. Иллюстрация к его применению имеется на рис. 2-58, *а.*

Если кольца (или соответственно 2-позиционные триггеры) имеют раздельные входы, то можно построить и такую схему, чтобы обращенный код возвращался в исходный регистр и получался бы, таким образом, не на выходах логических схем, а прямо на выходах цифровых элементов вместо прямого кода. Такое устройство показано на рис. 2-58, *б.* Существенным в нем является запоминание на реактивностях — на линиях задержки или, может быть, на паразитных емкостях; без этого выдачу обратного кода в исходный регистр пришлось бы выполнять через вспомогательный регистр.

Обе схемы рис. 2-58, как мы говорили, в равной мере подходят для двоичной и для любой n-ичной ($n \neq 2$) системы счисления, но только в тех случаях, когда применяются непосредственные способы изображения n-ичных цифр (т. е. когда каждой n-ичной цифре соответствует одно из положений n-позиционного цифрового элемента). Сложнее дело обстоит в тех случаях, когда n-ичные цифры кодируются с помощью нескольких двоичных цифр, т. е. когда для изображения каждого n-ичного разряда применяется несколько двоичных триггеров.

На рис. 2-59 в качестве примера показан один десятичный разряд схемы обращения кода для случая, когда каждая десятичная цифра кодируется четырьмя двоичными разрядами с весами 8, 4, 2, 1.

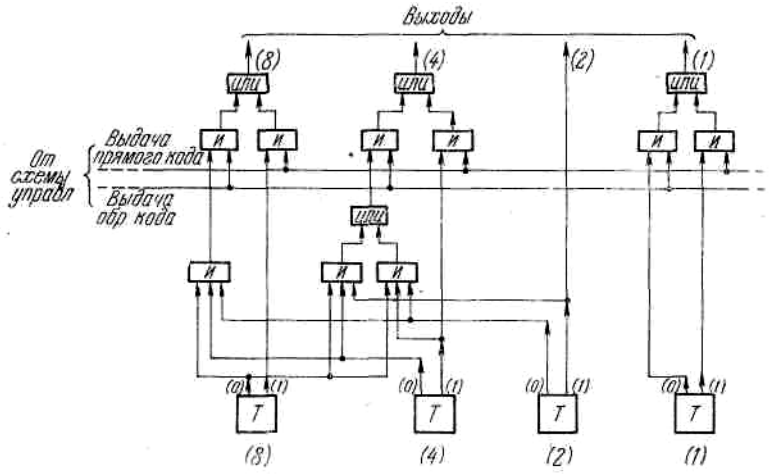


Рис. 2-59. Один разряд схемы для обращения десятичных чисел (для кода «8, 4, 2, 1»).

Построение этой схемы получено на основании простого сопоставления кодов всех десятичных цифр с соответствующими кодами обратных цифр

Прямая цифра	Обратная цифра	Прямая цифра	Обратная цифра
(0) — 0000	(9) — 1001	(5) — 0101	4) — 0100
(1) — 0001	(8) — 1000	(6) — 0110	(3) — 0011
(2) — 0010	(7) — 0111	(7) — 0111	(2) — 0010
(3) — 0011	(6) — 0110	(8) — 1000	(1) — 0001
(4) — 0100	(5) — 0101	(9) — 1001	(0) — 0000

Сравнивая коды прямых и обратных цифр, легко видеть, что цифры первого (младшего) двоичного разряда в них всегда обратны, а цифры второго разряда всегда совпадают между собой. (На рисунке первый двоичный разряд расположен справа, левее от него — второй двоичный разряд и т. д.)

Сложнее всего правило образования третьего двоичного разряда обращенного кода. Этот разряд содержит единицу в тех и только тех случаях, когда в трех старших двоичных разрядах прямого кода содержится либо комбинация 001... (десятичные цифры «2» и «3») либо комбинация 010... (десятичные цифры «4» и «5»).

В четвертом двоичном разряде обращенный код содержит единицу в тех и только тех случаях, когда три старшие двоичные цифры прямого кода являются нулями (десятичные цифры «0» и «1»).

В соответствии с этими правилами построены логические схемы на рис. 2-59. При этом предполагается, что исходное число находится в регистре из статических триггеров, а на выходах логических схем в зависимости от команд схемы управления появляется либо прямой, либо обратный код числа (подобно тому, как это было в схеме рис. 2-58, *а*).

Приведенная схема, очевидно, заметно сложнее, чем схемы обращения кода в двоичной системе или вообще при использовании непосредственных способов изображения n-ичных цифр. Впрочем, в разделе 1 (см. 1.6.2) упоминалось и о таких способах кодирования десятичных цифр, при которых обращение десятичной цифры получается обращением двоичных цифр в ее представлении (код «с излишком 3», код «2, 4, 2, 1»); при использовании таких способов схемы для обращения десятичных чисел ничем не отличаются от схем обращения двоичных чисел.

2.8.4. Непосредственное вычитание.

Непосредственное вычитание сравнительно редко имеется в вычислительных машинах **в** качестве элементарной операции. Большей частью вычитание выполняется с помощью сумматора и устройств обращения кодов чисел; такие методы вычитания описаны в разделе 3. Здесь же мы рассмотрим кратко вычитатели — схемы, которые могли бы осуществлять вычитание непосредственно, подобно тому как в сумматорах непосредственно осуществляется суммирование.

Подобно тому как полный сумматор строится из одноразрядных сумматоров, полный вычитатель должен строиться из одноразрядных вычитателей. *Одноразрядный вычитатель*— это схема, на входы которой поступают цифры соответствующего разряда уменьшаемого (*b*) и вычитаемого (c) и сигнал займа из предыдущего (младшего) разряда (*d)* и которая вырабатывает цифру разности *B* для данного разряда и сигнал займа *D* для следующего (старшего) разряда.

Одноразрядные вычитатели могут объединяться в полную схему вычитателя параллельным, последовательным или последовательно-параллельным способом. Что же касается самой схемы одноразрядного вычитателя, то она может быть построена по комбинационному, накапливающему или амплитудному принципу.

В тех машинах, в которых предусматривалось непосредственное вычитание, схема вычитателя не строилась отдельно, но создавалась на базе типовой схемы сумматора. Вместо сумматора фактически строился сумматор-вычитатель, который при определенной коммутации внешних связей мог использоваться как сумматор, а при другой коммутации — как вычитатель. При этом коммутацию внешних связей можно либо выполнить заранее, постоянно, либо осуществить через логические элементы, которые управлялись бы специальными командами. В первом случае данное устройство могло бы использоваться всегда либо только как сумматор, либо только как вычитатель; во втором случае схема управления указывала бы каждый раз род работы — суммирование или вычитание.

Одно из возможных построений одноразрядного двоичного сумматора-вычитателя иллюстрируется рис. 2-60.

На рис. 2-60, *а* показана схема полусумматора-вычитателя. Она построена в соответствии с таблицей 2-2.

На рис. 2-60, *б* показано, как нужно закоммутировать два полусумматора-вычитателя для того, чтобы получился одноразрядный сумматор на 3 входа. Соединения здесь выполнены так же, как на рис. 2-11 (стр. 68); выходы б полусумматоров-вычитателей не используются.

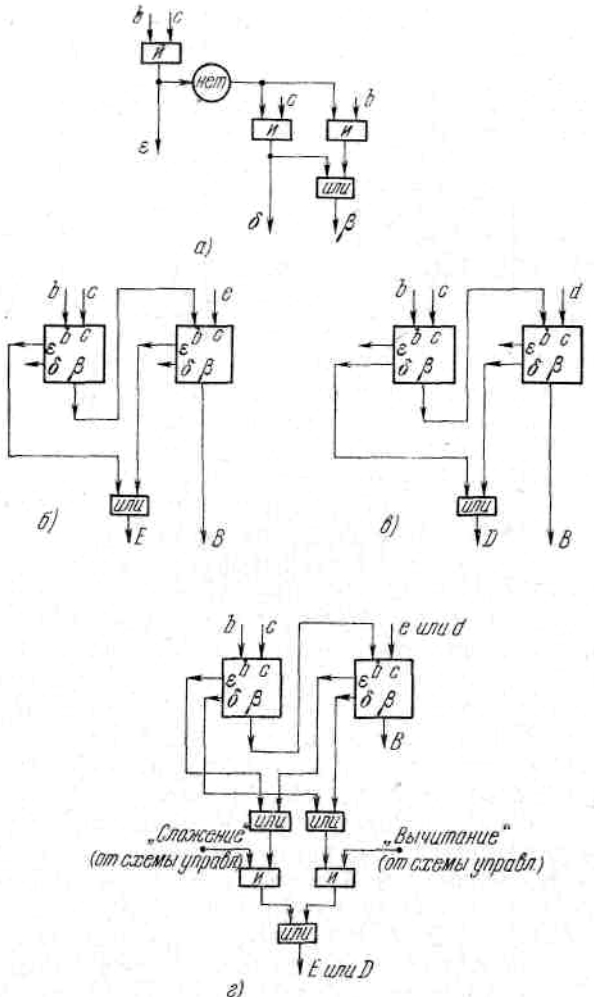


Рис. 2-60. Двоичный одноразрядный сумматор-вычитатель: *а*) построение полусумматора-вычитателя; б) соединение двух полусумматоров-вычитателей в одноразрядный сумматор; в) соединение двух полусумматоров-вычитателей в одноразрядный вычитатель; *г)* одноразрядный сумматор-вычитатель, управляемый командами от схемы управления.

Таблица 2-2

Логика работы двоичного полусумматора-вычитателя

1-е слагаемое (уменьшаемое)	2-е слагаемое (вычитаемое)	Сумма (разность) β	Перенос ε	Заем δ
0	0	0	0	0
0	1	1	0	1
1	0	1	0	0
1	1	0	1	0

На рис-. 2-60, *в* такие же два полусумматора-вычитателя соединены так, чтобы получить полную схему одноразрядного вычитателя. Здесь задействованы выходы б и не используются выходы *е*. Заметим, что сигнал займа в следующий двоичный разряд образуется (как и сигнал переноса при суммировании) либо на одном, либо на другом полувычитателе, но никогда не на обоих полувычитателях вместе. Если, скажем, сигнал займа образовался на первом полувычитателе, то цифра разности на его выходе β есть непременно единица; какова бы ни была цифра d ($d = 0$ или $d = 1$), при вычитании d из β во втором полувычитателе сигнал займа получиться не может. Наоборот, если на первом полувычитателе сигнал займа не получился ($\delta_1 = 0$), то цифра разности на его выходе β может быть и нулем, и единицей; если эта разность есть 0, то при $d = 1$ получается сигнал займа во втором полувычитателе.

Наконец, на рис. 2-60*, г* показана коммутация полусумматоров-вычитателей в полный одноразрядный сумматор-вычитатель, управляемый командами схемы управления. Если одноразрядные сумматоры-вычитатели такого типа соединены в параллельную схему, то команды «сложение» или «вычитание» должны подаваться на все разряды одновременно; если одноразрядный сумматор-вычитатель используется в последовательной схеме, то команда «сложение» или команда «вычитание» должна подаваться в течение всего времени выполнения сложения или соответственно вычитания. При наличии команды «сложение» сигнал, формируемый в схеме рис. 2-60, *г* для следующего разряда, является сигналом двоичного переноса, а при наличии команды «вычитание» — сигналом двоичного займа. Разумеется, схемы рис. 2-60 являются далеко не единственными из возможных построений полусумматора-вычитателя и полного одноразрядного сумматора-вычитателя; последний, конечно, совсем не обязан строиться непременно из двух полусумматоров-вычитателей.

Тот факт, что в двоичной системе сумма двух цифр по модулю 2 равна их разности (см. таблицу 2-2), позволяет строить накапливающие сумматоры-вычитатели по тому же принципу, что и комбинационные; в таком сумматоре-вычитателе имелась бы общая часть образования суммы или разности, представляющая собой триггер-счетчик по модулю 2, и раздельные схемы комбинационного типа для формирования сигналов переноса или займа.

Существенно иным является положение в любой другой системе счисления (с основанием $n \neq 2$), где сумма двух цифр по модулю n вообще не равна их разности. В вычитателе комбинационного типа не только схема образования сигнала займа отличалась бы от схемы образования сигнала переноса в сумматоре, но и цифра разности получалась бы иначе, чем цифра суммы при суммировании. В накапливающем сумматоре вместо суммирующего счетчика нужен был бы так называемый реверсивный счетчик, который мог бы выполнять и добавление единиц к имеющемуся в нем числу, и вычитание единиц; в частности, добавление единиц он должен производить при поступлении на его вход импульсов, соответствующих цифре уменьшаемого, и вычитание единиц — при поступлении импульса займа из предыдущего разряда и импульсов, соответствующих цифре вычитаемого.

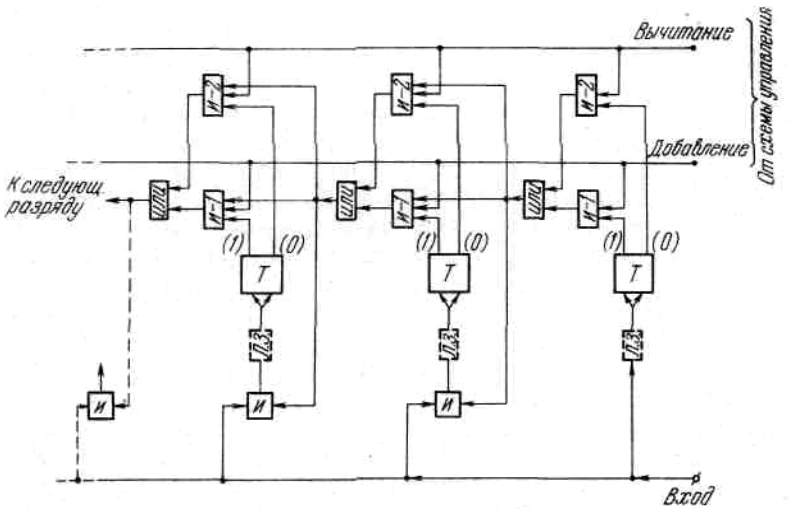


Рис. 2-61. Реверсивный двоичный счетчик, построенный по аналогии с обычным счетчиком рис. 2-37, *е*

Реверсивные счетчики иногда нужны также и в схемах управления. Для того чтобы дать представление об их устройстве, на рис. 2-61 показана схема многоразрядного двоичного реверсивного счетчика, построенная по аналогии со схемой обычного счетчика рис. 2-37, *в* (стр. 96). При наличии команды «добавление» (в виде уровня напряжения от схемы управления) во всех разрядах открыты элементы «и-1» и заперты элементы «и-2»; схема работает точно так же, как обычный счетчик рис. 2-37, *в*. При наличии команды «вычитание» импульс на второй двоичный разряд подается при условии, что к моменту прихода очередного входного импульса в первом разряде содержался «0», импульс на третий разряд — если нули содержались в первом и втором разрядах и т. д.; если,

например, в счетчике первоначально содержалась комбинация ...111, то при наличии команды «вычитание» первый импульс установит комбинацию ...110, второй—...101, третий — ...100, четвертый — ...011 и т. д.

Читателю предоставляется самостоятельно построить схемы реверсивных счетчиков по аналогии с другими схемами обычных счетчиков, приведенными в разделе 2.6.

2.8.5. Общие замечания.

Занимаясь на протяжении всей главы схемами для выполнения тех или иных элементарных операций, мы так и не дали точного определения понятию «элементарная операция» и не высказали никаких соображений о том, какие в точности элементарные операции должны быть предусмотрены в арифметическом устройстве машины. Когда в дальнейшем мы будем рассматривать выполнение арифметических и логических действий, мы будем в разных случаях пользоваться разным набором из перечисленных выше элементарных операций.

Список элементарных операций во многих случаях усложняется из-за крайних разрядов регистров. Например, сдвиг влево на один разряд, который во всех средних разрядах выполняется всегда однотипно, в самом младшем (правом) разряде регистра может выполняться либо так, что в него принимается нуль, либо так, что в нем сохраняется прежняя цифра, либо так, что в него принимается цифра, выдвигаемая из самого старшего (левого) разряда другого регистра, в котором тоже одновременно выполняется сдвиг, либо, наконец, так, что в крайний правый разряд вдвигается некоторая новая информация от схемы управления. При выполнении различных арифметических действий могут потребоваться в разных случаях разные варианты сдвига влево. Иногда в схеме арифметического устройства предусматриваются специальные элементарные операции, касающиеся только крайних разрядов (скажем, изменение на противоположное состояния триггера самого старшего разряда какого-либо двоичного регистра).

Вообще же вопросы о необходимом и достаточном наборе элементарных операций, об оптимальном наборе с точки зрения затрат оборудования в арифметическом устройстве и устройстве управления и с точки зрения быстродействия машины в целом и т. п. ждут еще подробных исследований и разработки.

3. Сложение, вычитание, сравнения

Полная операция сложения или вычитания, выполняемая вычислительной машиной в качестве самостоятельного законченного арифметического действия в соответствии со специальной командой центрального управления, существенно сложнее той элементарной операции суммирования, которая осуществляется сумматором. Элементарное суммирование входит лишь частью в полную операцию сложения или вычитания.

Оставляя в стороне вопросы выполнения сложения и вычитания при использовании других способов изображения чисел, мы рассмотрим здесь методы сложения и вычитания чисел, представленных позиционным способом **с** естественными весами разрядов и с неотрицательными цифрами, или близкими к этому (в смысле 1.3.3) способами. Речь пойдет сначала о сложении и вычитании чисел с фиксированной запятой при произвольных алгебраических знаках, затем — о сложении и вычитании чисел с плавающей запятой. Изложение будет вестись применительно к двоичной системе, хотя описываемые методы легко обобщить для любых других систем счисления; при этом в изображении мантисс чисел **с** плавающей запятой в последующем изложении существенна однородность системы счисления.

Попутно рассматриваются операции сравнения чисел,

3.1. Алгебраическое сложение и вычитание чисел с фиксированной запятой.

3.1.1. Основной метод выполнения алгебраического сложения и вычитания в прямых кодах.

Пусть имеем два двоичных числа с фиксированной запятой, B и C, представленных позиционным способом с естественными весами разрядов и с цифрами 0,1; для определенности будем полагать, что в обоих числах запятая фиксирована перед первым (старшим) разрядом, так что вес этого разряда равен $^1/_2$, а вес младшего *(m*-го) разряда равен $^1/_2{}^m$. Когда мы говорим, что числа представлены прямыми кодами, то тем самым предполагается (см. 1.4.1), что упомянутые m двоичных разрядов изображают *абсолютную величину* числа, а кроме того, в каждом числе имеется специальный разряд *алгебраического знака* с допустимыми символами для него «+» или «—». Одно из двух имеющихся чисел является 1-м слагаемым или уменьшаемым, другое — 2-м слагаемым или вычитаемым. Обычная методика выполнения сложения и вычитания вручную сводится к следующему:

— сравнить алгебраические знаки чисел; если знаки различны, то вместо сложения выполнять дальше вычитание, а вместо вычитания — сложение (при этом условимся, что, когда сложение преобразуется в вычитание, 1-е слагаемое считается в дальнейшем уменьшаемым, 2-е слагаемое — вычитаемым; когда вычитание преобразуется в сложение, то уменьшаемое в дальнейшем считается 1-м слагаемым, а вычитаемое — 2-м слагаемым);

— сложить абсолютные величины чисел или соответственно вычесть из большей абсолютной величины меньшую;

— приписать результату сложения алгебраический знак 1-го слагаемого, а результату вычитания — либо

алгебраический знак уменьшаемого, если оно по абсолютной величине больше, чем вычитаемое, либо противоположный ему знак.

Трудности в осуществлении этой методики в вычислительной машине связаны главным образом с необходимостью выполнить сравнение абсолютных величин чисел перед вычитанием, а также с тем, что в составе арифметического устройства большей частью имеется только сумматор, но нет вычитателя*); о некоторых других усложнениях по сравнению со счетом вручную мы скажем несколько позже.

При отсутствии вычитателя вычисление разности абсолютных величин двух чисел $|C| - |B|$ удобно выполнять путем преобразования величины $|B|$ в дополнительный код.
Поскольку мы считаем, что вес самого старшего разряда равен $^1/_2$, а разряд с весом 1 и старшие разряды отсутствуют, то счет фактически ведется по модулю 1. Найдя дополнение от $|B|$ до единицы, т. е. величину $1 - |B|$, можно затем при помощи суммирования получить величину $|C| + (1 - |B|)$, которая по модулю 1 равна разности $|C| - |B|$.

В процессе выполнения суммирования попутно выясняется, действительно ли абсолютная величина $|C|$ больше $|B|$. Если $|C| \geq |B|$, то $|C| + (1 - |B|) = 1 + |C| - |B| \geq 1$, и при выполнении суммирования получается перенос из старшего разряда (с весом $^1/_2$) в несуществующий разряд слева от него (с весом 1); если $|C|<|B|$, то $1 + |C| - |B| <1$, и такого переноса нет. В последнем случае вместо разности $|C| - |B|$ нужно было на самом деле искать разность $|B| - |C|$. Для получения правильного результата можно вычислить дополнение от величины $|C| - |B|$ до единицы:

$$1 - (|C| - |B|) \equiv |B| - |C| \pmod 1.$$

Дополнительный код некоторой величины, как говорилось в 1.4.2, вообще находится в два приема: отыскание, обратного кода этой величины и добавление единицы младшего разряда. Первая часть (отыскание обратного кода) — это операция поразрядная; что же касается добавления единицы младшего разряда, то, например, в параллельных устройствах на него нужно отводить время, достаточное для распространения сигналов переноса вдоль цепочки разрядов.

В действительности, однако, при отыскании дополнительного кода от величины $|B|$ в начале операции добавление единицы младшего разряда можно совместить с последующим суммированием $|C| + (1 - |B|)$. Для этого в параллельном сумматоре нужно при выполнении суммирования послать единицу на вход переноса самого младшего разряда — как бы перенесенную из несуществующего разряда справа от него; в последовательном сумматоре то же выполняется посылкой единицы на вход переноса в первом такте (в момент суммирования младших разрядов). Именно с такой целью на рис. 2-1 и 2-2, иллюстрирующих построение последовательного и параллельного сумматоров, показана возможность ввода единицы от схемы управления на вход переноса.

При отыскании дополнительного кода от величины $|C| - |B|$, если это потребуется в конце операции, придется полностью отводить время суммирования на добавление единицы.

Итак, с учетом приведенных выше замечаний изложенный метод выполнения алгебраического сложения-вычитания выглядит следующим образом.

Первый метод алгебраического сложения вычитания в прямых кодах ($B \pm C$):

(а) сравнить знаки чисел; если знаки различны, то вместо сложения выполнять вычитание, а вместо вычитания — сложение; если знаки одинаковы, то сложение остается сложением, а вычитание —вычитанием;

(б) если по результатам п. (а) должно выполняться вычитание, то обратить код B (в том числе изменить и знак B на противоположный);

(в) выполнить суммирование основных разрядов (без знаков) B и C; если идет вычитание, то при выполнении суммирования младших разрядов подать единицу на вход переноса; результату приписать знак B;

(г) если идет вычитание и если в п. (в) не был получен перенос из самого старшего разряда, то обратить код результата (опять изменив знак на противоположный);

(д) если выполнялся п. (г), то вслед за ним добавить к результату единицу младшего разряда (e_m).

Пункт (а) здесь такой же, как в правиле выполнения сложения-вычитания вручную. Осуществляется эта часть операции схемой управления, вне собственно арифметических цепей. Заметим, что в последовательном арифметическом устройстве выполнение п. (а) не займет слишком много времени только при условии, что разряд знака поступает перед всеми остальными разрядами числа.

В п. (б) мы действуем, так сказать, наудачу. Хотя заранее неизвестно, какое из чисел больше по абсолютной величине, мы поступаем так, будто $|C| \geq |B|$. Выполняя обращение кода B, мы тем самым подготавливаем выполнение вычитания $|C| - |B|$, а алгебраический знак будущего результата устанавливаем противоположным знаку исходного числа B. Если в действительности окажется, что $|C| < |B|$, то необходимые исправления будут

*) Из дальнейшего как раз и видно, что строить наряду с сумматорами вычитатели, как правило, нецелесообразно.

внесены в пп. (г) и (д).

В п. (в) выполняется наиболее существенная часть операции — собственно сложение или вычитание абсолютных величин чисел. Если идет вычитание, то попутно проверяется, действительно ли $|C| \geq |B|$, как предполагалось в п. (б); об этом мы судим по наличию или отсутствию сигнала переноса из самого старшего разряда. В зависимости от того, каков оказался результат сравнения, далее в ходе вычитания выполняются или не выполняются пп. (г) и (д).

Особо следует обратить внимание на то, что сигнал переноса из самого старшего разряда может возникнуть и при выполнении сложения. Это указывало бы, что сумма абсолютных величин складываемых чисел больше единицы. Вычислительные машины общего назначения с фиксированной запятой строятся большей частью так, что появление этого сигнала приводит к остановке вычислений (результат не помещается в разрядной сетке машины); иногда предусматривается специальная модификация операции сложения, при которой переполнение разрядной сетки приводит вместо остановки вычислений к выработке некоторого признака, который может влиять на ход дальнейших операций. В специализированных машинах, работающих в реальном масштабе времени, часто выгоднее при появлении непредусмотренного переполнения не останавливать работу, но формировать сигнал о том, что при выполнении данного цикла вычислений имелись случаи переполнения.

В п. (г) и (д), как говорилось, исправляется результат вычитания в том случае, когда в действительности абсолютная величина $|C|$ меньше |B|.

Рассмотрим сейчас, каково должно быть оборудование для осуществления сложения-вычитания по указанному методу и каково время выполнения операции. При этом мы будем говорить только об оборудовании арифметических цепей, оставляя в стороне схему управления операциями, а время будем вычислять с момента приема исходных чисел в регистры до момента получения результата в одном из них.

Примерное построение арифметических цепей для выполнения сложения-вычитания иллюстрируется рис. 3-1.

Основу параллельного устройства (рис. 3-1, *а*) составляют два регистра *B* и *C* из цифровых элементов (триггеров) и параллельный сумматор. Если количество основных разрядов в каждом из слагаемых (кроме разряда алгебраического знака) равно *m,* то сумматор должен содержать *m* разрядов, а регистры — по *(m* + 1) разрядов. Основные разряды регистров связаны с сумматором таким образом, что исходные числа из регистров поступают на входы сумматора, а с выхода сумматора результат передается в один из регистров *(B),* где заменяет 1-е слагаемое. В частном случае регистр *B* может состоять из счетчиков по модулю 2, входящих составной частью в накапливающий сумматор. Естественно, что если схема строится по потенциальному принципу и без запоминания на реактивностях, то передача числа с выходов сумматора в регистр *B* возможна только через вспомогательные цифровые элементы. Регистр *B* должен быть оборудован цепями для обращения кода, охватывающими также разряд знака, а в регистре *C* должна быть предусмотрена цепь гашения. Кроме того, в регистрах должны быть устройства для приема исходных чисел и для выдачи результата; эти цепи на рис. 3-1 не показаны. Возможно, что цепь гашения *C* используется также и при приеме чисел в *C*; тогда она должна охватывать также и разряд знака *C*, что при выполнении сложения-вычитания роли не играет.

Обозначим через $\tau_р$ время расшифровки знаков в схеме управления, через τ_0 — время обращения кода, через τ_Σ — время суммирования двух чисел параллельным сумматором. Временем $\tau_р$ обычно можно пренебречь по сравнению с другими временами; τ_0 — это во многих случаях время такого же порядка, как минимальный такт срабатывания цифровых элементов, а τ_Σ обычно в 2—3 раза больше, чем τ_0 (τ_Σ можно представить в виде $\tau_\Sigma = m\tau_E + \tau_B$, где *m* — количество разрядов, τ_E — задержка переноса на разряд, τ_B — время образования суммы). Без учета $\tau_р$ максимальное время выполнения сложения-вычитания в параллельном устройстве составит

$$2\tau_0 + 2\tau_\Sigma.$$

Среднее время выполнения операции заметно меньше этой максимальной величины. Если бы все числа от +1 до —1 были равновероятны, то первое обращение кода (п. (б)) пришлось бы выполнять в половине всех случаев. Однако в действительности часть комбинаций чисел — те, в которых получилось бы переполнение разрядной сетки,— заранее исключаются при программировании. Из рис. 3-2, иллюстрирующегосоотношение вероятностей различных случаев, ясно, что первое обращение кода (п. (б)) выполняется в $^2/_3$ всех случаев. По этому же рисунку видно, что второе обращение кода (п. (в)) и второе суммирование (п. (д)) приходится выполнять в $^1/_3$ всех случаев. Таким образом, среднее время выполнения сложения-вычитания в параллельном устройстве равно

$$\frac{2}{3}\tau_0 + \tau_\Sigma + \frac{1}{3}(\tau_0 + \tau_\Sigma) = \tau_0 + \frac{4}{3}\tau_\Sigma.$$

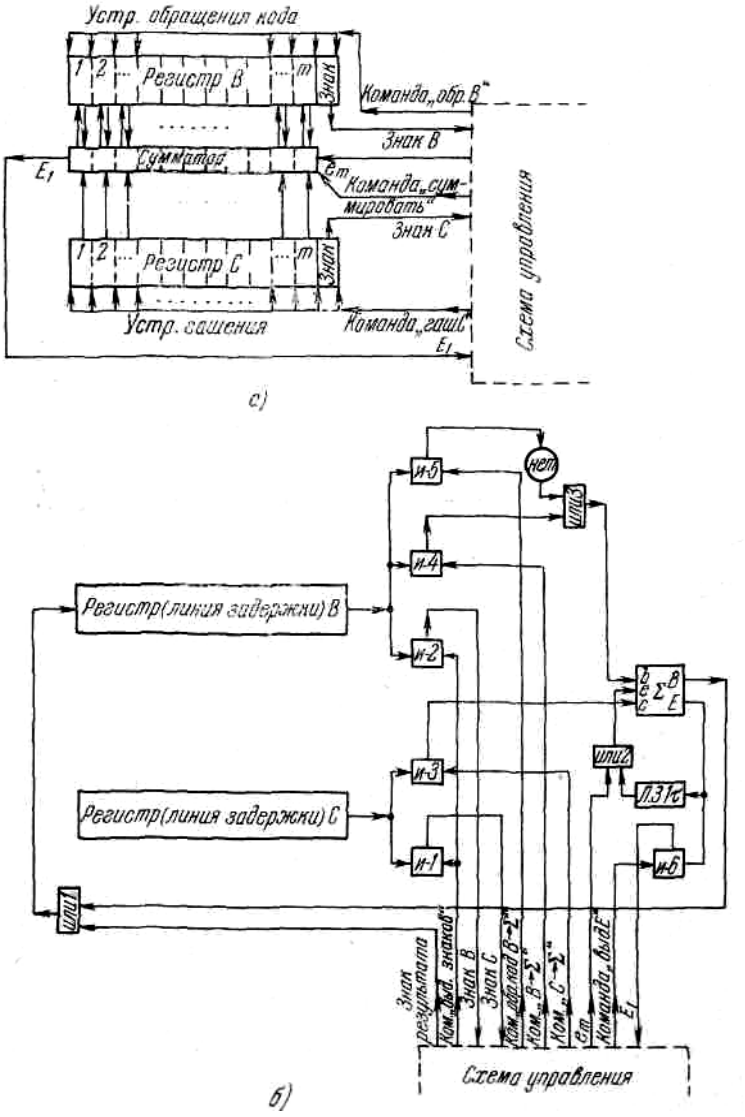


Рис. 3-1. Примерное построение арифметических цепей для выполнения сложения-вычитания:
а) в параллельном варианте; б) в последовательном варианте.

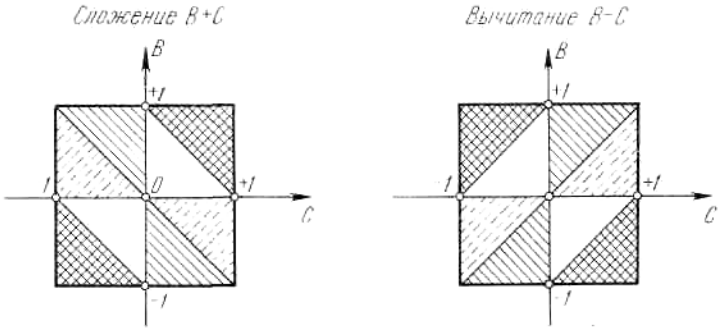


Рис. 3-2. Соотношение вероятностей различных случаев сложения, вычитания.

— случаи, в которых требуется только одно суммирование;

— случаи, в которых требуется одно обращение кода и одно суммирование;

— случаи, в которых требуются два обращения кода и два суммирования;

— случаи, приводящие к переполнению разрядной сетки (исключаются при программировании).

В последовательном устройстве рис. 3-1, б регистры *B* и *C* — последовательного типа. Каждый из них может

представлять собой обычный регистр из цифровых элементов с цепями сдвига вправо и, возможно, с цепями для параллельного приема чисел. Но это могут быть и, скажем, линии задержки на $m + 1$ разрядов каждая с последовательным вводом и выводом чисел. Цепи ввода и вывода чисел как и детали устройства регистров на рис. 3-1, *б* не показаны. Зато устройство остальной части схемы сложения-вычитания изображено Здесь более подробно, чем на рис. 3-1, *а,* потому, что выполнение элементарных операций в последовательных устройствах мы почти не рассматривали раньше.

Вентили (элементы «и») «и-1» — «и-6» управляются командами от схемы управления. На входы этих вентилей цифра за цифрой поступают числа из регистров (оба одновременно). В регистрах числа расположены так, что на их выходах появляются сначала алгебраические знаки, затем младшие разряды чисел, затем следующие разряды в порядке возрастания весов.

В первом такте, когда из регистров поступают алгебраические знаки чисел, схема управления открывает вентили «и1» и «и2». Знаки поступают в схему управления, где производится их расшифровка. В зависимости от ее результатов в следующем такте на вход регистра *B* через элемент «или-1» подается либо тот же знак, который был выдвинут из *B,* либо противоположный ему знак, через элемент «или-2» на вход *e* сумматора подается или не подается единица, а из двух вентилей на входе *b* сумматора открывается «и4» или «и5». Этот вентиль будет открыт в течение всех *m* тактов, когда из регистра *B* поступают основные разряды числа. Одновременно на выходе регистра *C* открыт вентиль «и3». Таким образом, абсолютная величина $|C|$ и прямой или обратный код абсолютной величины $|B|$ цифра за цифрой поступают на входы одноразрядного сумматора 2, который вместе с линией задержки на 1 такт для сигнала переноса образует последовательный сумматор. С выхода сумматора цифры суммы возвращаются на вход регистра *B*.

В тот момент, когда через сумматор проходят последние (старшие) разряды слагаемых, схема управления открывает вентиль «и6» и получает через него сигнал переноса E_1. Если при выполнении вычитания оказывается, что $E_1 = 1$, то дальше из регистра *B* имеющееся там число снова пропускается через вентиль «и5» (т. е. обратным кодом) на сумматор. Выходные клапаны регистра *C* при этом закрыты, так что на вход *c* поступают нули. В момент прохождения через сумматор младшего разряда обратного кода *B* на вход *e* от схемы управления снова поступает сигнал $e_m = 1$. Таким образом, на выходе сумматора появляется дополнение от предыдущего результата (с противоположным знаком). Число с выхода сумматора вновь возвращается в регистр *B.*

Как видим, окончательный результат сложения-вычитания образуется в регистре *B* либо в конце первого периода числа (т. е. за $m + 1$ тактов), либо в конце второго периода числа (т. е. всего за 2 $(m + 1)$ тактов). Поскольку взятие дополнения от результата первого суммирования требуется лишь в $^1/_3$ случаев сложения-вычитания, среднее время выполнения операции в последовательном варианте оказывается равным $^4/_3$ периода числа, т. е. $^4/_3$ $(m + 1)$ тактов (периодов цифр).

Читателю предлагается убедиться самостоятельно, что применение сумматоров-вычитателей вместо простых сумматоров дает лишь незначительную экономию по времени в параллельных устройствах и не приносит совсем никакой пользы в последовательных устройствах — разве что упрощает несколько функции схемы управления.

Несколько примеров выполнения сложения-вычитания по описанному методу имеется в конце следующего подраздела.

3.1.2. Другие методы выполнения сложения-вычитания в прямых кодах.

Описываемые ниже методы позволяют получить некоторый выигрыш во времени по сравнению с основным методом. Они удобны для применения в параллельных устройствах.

Сравнительно небольшое увеличение скорости можно получить следующим путем. Представим себе, что устройство параллельного сумматора и организация его связей с регистрами позволяют отделить процесс образования двоичных переносов от процесса формирования цифр суммы. После того как числа, которые должны суммироваться, установлены в регистрах *B* и C, происходит установление сигналов двоичного переноса по всем разрядам. Однако для образования цифр суммы и передачи их в регистр *B* требуется отдельная команда от схемы управления; пока этой команды нет, содержимое регистров *B* и C не изменяется. Такое положение часто имеет место при использовании комбинационных сумматоров, а также, например, в сумматоре промежуточного типа рис. 2-49 (стр. 109).

Если сумматор удовлетворяет указанному условию, то сложение и вычитание можно выполнять следующим образом (второй метод).

Вначале — так же как в основном методе (см. стр. 121) —производится сравнение знаков чисел *B* и *C* (п. (а)); если оказывается, что далее должно выполняться сложение, то обычным образом производится суммирование (п. (в)). Если оказывается, что должно производиться вычитание, то выполняем п. (б) основного метода — обращение кода *B,* а далее поступаем несколько иначе:

(в') образовать сигналы двоичного переноса по всем разрядам, послав при этом от схемы управления единицу на вход переноса младшего разряда;

(г') если в п. (в') не был получен сигнал переноса из старшего двоичного разряда, то вновь обратить код *B,* одновременно изменив и его знак, и одновременно обратить код *C*; затем снова образовать сигналы двоичного переноса по всем разрядам, послав при этом от схемы управления единицу на вход переноса

младшего разряда;

(д') закончить выполнение суммирования: сформировать во всех разрядах цифры суммы.

Как видим, отличие от основного алгоритма здесь имеется только в выполнении вычитания. Произведя наугад обращение кода *B,* мы в п. (в') прежде всего проверяем, действительно ли $|C| \geq |B|$; для этого, очевидно, нет нужды производить полное вычитание, но достаточно только образовать сигналы двоичного переноса. Если оказалось, что на самом деле $|C| < |B|$, т. е. нужно было обращать код $|C|$, а не код B, то в п. (г') восстанавливается снова прямой код B и производится обращение кода $|C|$. Только после этого в п. (д') выполняется до конца суммирование, которое теперь во всех случаях дает сразу правильный результат. Экономия во времени выполнения сложения-вычитания по сравнению с основным методом получается за счет того, что одно из суммирований производится не полностью. На него затрачивается вместо полного времени суммирования $\tau_\Sigma = m\tau_E + \tau_B$ только время $m\tau_E$ (где m — количество разрядов, τ_E — задержка переноса на разряд, τ_B — время образования суммы). Максимальное время выполнения сложения-вычитания в параллельном устройстве теперь оказывается равным $2\tau_0 + 2m\tau_E + \tau_B$ (где τ_0 — время обращения кода) вместо $2\tau_0 + 2\tau_\Sigma = 2\tau_0 + 2m\tau_E + 2\tau_B$ в основном методе; среднее время равно теперь $\tau_0 + \frac{4}{3} m\tau_E + \tau_B$ вместо $\tau_0 + \frac{4}{3}\tau_\Sigma = \tau_0 + \frac{4}{3} m\tau_E + \frac{4}{3}\tau_B$ для основного метода. Лишнее оборудование, которое необходимо по сравнению со случаем использования основного метода (см-рис. 3-1, *а),* — это устройство для обращения кода в регистре C; однако из дальнейшего видно, что такое устройство все равно необходимо и для выполнения других операций (например, деления);

В последовательных устройствах применение такого метода, очевидно, невыгодно, так как для полного суммирования двух чисел требуется такое же время, как для образования сигналов двоичного переноса (один период числа). Более того, если первый период числа мы потратили бы на образование сигналов переноса без полного суммирования, то тогда во всех случаях вычитания (а не только в части из них) пришлось бы тратить еще один период числа на окончательное суммирование, так что среднее время выполнения операции даже возросло бы по сравнению с основным вариантом.

Но и для параллельных устройств описанный метод не универсален, так как предъявляет некоторые специальные требования к сумматору. Выигрыш по скорости, который он дает, не очень значителен (особенно в среднем).

Более универсальным (хотя тоже только для параллельных устройств) и более быстрым является метод, предусматривающий выполнение *вычитания через обратные коды* (в отличие от предыдущих методов, в которых выполнялось *вычитание через дополнительные коды).* Идея этого метода состоит в том, что если бы счет в машине шел не по модулю 1, а по модулю $1—2^{-m}$, где 2^{-m} — вес младшего разряда (т. е. в двоичной системе по модулю 0.111...11), то в процессе вычитания нам нужно было бы вычислять не дополнительные коды, а всегда только обратные коды чисел[*]). При этом в конце вычитания нам не пришлось бы тратить лишнее время на добавление единицы младшего разряда к обратному коду первоначального результата (п. (д) основного алгоритма — см. стр. 121).

Когда счет идет по модулю 1, то целое количество единиц автоматически вычитается из результата в тех случаях, когда результат больше или равен единице; это происходит просто за счет того, что разряды целых в машине отсутствуют. Если мы решим вести счет по модулю $1—2^{-m}$, то нам придется в определенных случаях отбрасывать от результата суммирования эту величину. Заметим, однако, что теперь сама величина модуля $(1—2^{-m})$ помещается в разрядной сетке машины (в то время как модуль 1 в разрядной сетке не помещался); поэтому вычитание модуля из результата суммирования нужно теперь производить не при условии, что результат больше или равен модулю, а только при условии, что результат строго больше модуля. Так как вес младшего разряда есть 2^{-m}, то ближайшее большее число к величине $1—2^{-m}$ есть 1; если некоторое число строго больше величины $1—2^{-m}$, то это значит, что оно больше или равно 1. Таким образом, выполняя суммирование по модулю $1—2^{-m}$, мы должны отбрасывать величину этого модуля в тех и только тех случаях, когда результат больше или равен 1, т. е. когда получается единица переноса из старшего разряда сумматора влево — в несуществующий разряд с весом 1. Условие, при котором из результата суммирования следует вычитать модуль, оказывается точно таким же, как при счете по модулю 1.

Но при счете по модулю 1 игнорирование единицы переноса из старшего разряда сумматора как раз и было эквивалентно вычитанию модуля. Теперь же потеря единицы переноса из старшего разряда приводила бы к вычитанию из результата единицы, а не $1—2^{-m}$, как требуется. Исправить это положение можно введением *цепи кругового (циклического) переноса.* Если сумматор устроить так, чтобы сигнал переноса из старшего разряда поступал на вход переноса младшего разряда сумматора, то каждый раз, когда результат суммирования больше или равен 1, к результату одновременно с вычитанием 1 будет добавляться величина 2^{-m} (единица младшего разряда), так что в итоге результат окажется уменьшенным на величину $1—2^{-m}$, как это и требуется.

О том, что параллельные сумматоры часто строятся с цепью кругового переноса, мы уже упоминали в 2.1, а на рис. 2-1 даже показали пунктиром эту цепь.

Ясно, что в последовательном сумматоре осуществить круговой перенос невозможно. В параллельном же

[*]) Напомним, что обратный код числа a как раз и представляет собой дополнение от a до числа 0,111... 11, т. е. величину $(1 —2^{-m}) — a$.

сумматоре при наличии цепи кругового переноса установление сигналов переноса по всем разрядам занимает, как правило, время не больше $m\tau_E$, где m — количество разрядов, τ_E — задержка переноса на разряд,— столько же, сколько в обычном параллельном сумматоре. Это видно из следующего примера:

Распространение сигналов переноса начинается от того разряда, в котором среди цифр слагаемых имеются две единицы. Далее пример составлен так, что во всех старших разрядах и вслед за этим во всех младших разрядах цифры переноса устанавливаются последовательно. Однако когда сигнал переноса доходит до того разряда, откуда началось распространение переносов, то дальше уже ничего не изменяется: если в данном разряде обе цифры слагаемых есть единицы, то в следующий разряд переносится единица независимо от того, имеется ли перенос в данный разряд.

Исключение составляет такая схема параллельного сумматора с парафазной цепью переносов, в которой каждый разряд вырабатывает сигнал отсутствия или наличия переноса только после получения сигнала отсутствия или наличия переноса от предыдущего разряда (см. рис. 2-27 на стр. 85); распространение переносов в сумматоре такого типа начинается лишь с момента поступления сигнала «отсутствие переноса» или «наличие переноса» от схемы управления на младший разряд.

Ясно, что при указанном построении сумматора наличие цепи кругового переноса удлинило бы время установления цифр переноса до $2m\tau_E$.

Необходимо учитывать также, что наличие цепи кругового переноса иногда приводит к повышению требований к техническому уровню проектирования. Представим себе, например, что параллельный сумматор с круговым переносом строится из потенциальных элементов и что при этом допускается некоторая неодновременность в установке сигналов слагаемых на входах. В результате могло бы получиться, что в некотором разряде при установлении на входе комбинации цифр слагаемых 0,1 или 1,0 на короткое время появилась бы комбинация 1,1, и в данном разряде сумматора появился бы кратковременный сигнал переноса в следующий разряд; если и во всех остальных разрядах имеются комбинации цифр слагаемых 0, 1 или 1, 0, то этот сигнал стал бы непрерывно циркулировать по цепи кругового переноса с периодом $m\tau_E$. Если несинфазность в установлении сигналов слагаемых больше $m\tau_E$ или если один из одноразрядных сумматоров дает задержку больше, чем другие одноразрядные сумматоры, то при наличии во всех разрядах цифр слагаемых 0, 1 или 1,0 в цепи кругового переноса может даже установиться ложный потенциальный сигнал «1» по всем разрядам. По сути дела, при появлении таких слагаемых цепь кругового переноса превращается в триггер (усилитель постоянного тока с коэффициентом усиления 1, выход которого замкнут со входом). Триггер может иметь 2 устойчивых состояния: 0 (нули на входах и выходах переноса во всех разрядах) и 1 (единицы на входах и выходах переноса во всех разрядах). Аналогичные неприятности могут возникать в импульсных и импульсно-потенциальных системах.

Если отвлечься от указанных тонкостей, то метод выполнения вычитания через обратные коды применим в любой схеме параллельного сумматора; он, следовательно, более универсален, чем второй метод.

Подводя итоги, мы можем сформулировать его следующим образом (т р е т и й м е т о д) .

Вначале — так же как в основном методе (см. стр. 121) — производится сравнение знаков чисел B и C (*п.* (а)). Если оказывается, что далее должно выполняться сложение, то обычным образом производится суммирование (п. (в)). При этом наличие цепи кругового переноса в сумматоре никак не мешает выполнению суммирования: при сложении с фиксированной запятой сумма абсолютных величин слагаемых должна быть меньше единицы, перенос из старшего разряда отсутствует и, следовательно, на вход переноса младшего разряда поступает «0» — как и должно быть при сложении. Если оказывается, что должно производиться вычитание, то выполняем п. (б) основного метода — обращение кода B, а далее поступаем несколько иначе:

(в") выполнить суммирование основных разрядов (без знаков) B и C; при этом от схемы управления никакие сигналы на вход переноса младшего разряда не поступают; если необходимо, то сигнал «1» может прийти на этот вход по цепи кругового переноса;

(г") если в предыдущем пункте не было единицы в цепи кругового переноса, то обратить код результата, изменив одновременно его алгебраический знак на противоположный; п. (д) при этом отсутствует.

По оборудованию схема для осуществления этого метода отличается от схемы рис. 3-1, а только отсутствием в схеме управления цепей для выработки сигналов e_m и наличием вместо этого цепи кругового переноса в сумматоре.

Максимальная длительность выполнения сложения-вычитания по третьему методу оказывается равной $2\tau_0+\tau_\Sigma$, где τ_0 — время обращения кода, τ_Σ — время суммирования (вместо $2\tau_0 + 2\tau_\Sigma$ в основном методе); средняя

длительность выполнения сложения-вычитания по третьему методу равна $\tau_0 + \tau_\Sigma$ (вместо $\tau_0 + \frac{4}{3}\tau_\Sigma$ для основного метода).

Пример. Пусть требуется произвести сложение чисел $(+^{11}/_{16}) + (—^{5}/_{16})$. Числа представлены в двоичной системе, позиционным способом с естественными весами разрядов (количество основных разрядов m = 4), прямыми кодами.

Каким бы методом ни выполнялось сложение-вычитание, оно начинается с приема 1-го слагаемого в регистр *B* и 2-го слагаемого в регистр *C:*

(*B*) .1011 +
(C) .0101 —

Далее производится сравнение алгебраических знаков и выясняется, что вместо сложения нужно выполнять вычитание (п. (а)).

Поэтому дальше в любом методе выполняется п. (б) — обращение кода *B:*

(B) .0100 —
(C) .0101 —

Затем при использовании разных методов действуем различно.

По первому методу:

(в) выполняем суммирование основных разрядов *B* и *C*, посылая при этом единицу на вход переноса младшего разряда:

(исходные числа)	(*B*) ·0 1 0 0 —
	(*C*) ·0 1 0 1 —
(цифры переноса)	(0) ·(1) (0) (1) ①
(сумма)	.1 0 . 1 0 —

(г) так как в предыдущем пункте не было переноса из старшего разряда, производим обращение кода суммы

.0101 +

(д) добавляем единицу переноса в младший разряд

	. 0 1 0 1 + 1
(цифры переноса)	(0). (0) (0)(1)1
(сумма)	. 0 1 1 0 +

По второму методу:

(в') формируем цифры переноса, посылая при этом единицу на вход переноса младшего разряда

	(*B*) . 0 1 0 0 —
	(*C*) . 0 1 0 1 —
(цифры переноса)	(0).(1) (0)(1)①

(г') поскольку в предыдущем пункте не было переноса из старшего разряда, выполняем одновременно обращение кодов в регистрах *B* и *C* и вновь формируем сигналы переноса, посылая при этом единицу на вход переноса младшего разряда

	(*B*) .1 0 1 1 +
	(*C*) .1 0 1 0 +
(цифры переноса)	(1).(0)(1)(1)①

(д') заканчиваем суммирование

(исходные числа)	(*B*) .1 0 1 1+
	(*C*) .1 0 1 0+
(цифры переноса)	(1).(0)(1)(1)①
(сумма)	*. 0 1 1 0+

(слева от запятой сумма тоже должна была бы получиться единицей; однако этот разряд в регистрах отсутствует).

По третьему методу:

(в") выполняем суммирование содержимого основных разрядов регистров *B* и *C* с круговым переносом (исходные числа)

	(*B*).0 1 0 0—
	(C).0 1 0 1—
(цифры переноса)	(0).(1)(0)(0)(0)
(сумма)	.1001 —

(г") так как в предыдущем пункте по цепи кругового переноса передавалась цифра «0», производим обращение кода суммы, одновременно изменяя ее знак

.0110+

Число, которое в любом случае мы получали в конечном итоге (.0110+), —это +6/16, результат операции.

Читателю предлагается самостоятельно повторить этот пример для случая, когда исходные числа поступают в обратном порядке: (—5/16) + (+11/16). При этом в начале операции число —5/16 должно поступить в регистр *B,* а число +11/16 — в регистр *C;* результат должен получиться, конечно, таким же, как в рассмотренном ранее случае.

3.1.3. Алгебраическое сложение-вычитание в дополнительных кодах.

В дополнительных кодах сложение и вычитание выполняются, как правило, проще, чем в прямых кодах. Ради этого, собственно, и используются вообще дополнительные коды.

Для определенности будем полагать, что дополнительный код построен по первому варианту (см. 1.4.2), т. е. что положительные числа содержат в разряде алгебраического знака «0», а отрицательные — цифру «1». Будем считать также, что разряд алгебраического знака является первым разрядом в целой части числа и потому вес его равен —1 (вместо естественного веса +1), а *m* основных разрядов числа находятся все в дробной части числа[*]).

При этом все положительные числа будут представлены **в** двоичной системе чисто позиционным способом, с естественными весами разрядов, с цифрами 0, 1 и с запятой, фиксированной перед первым (старшим) из основных разрядов; отличие веса знакового разряда от естественного не играет роли для положительных чисел, так как цифра данного разряда есть «0». Так как для отрицательных чисел в знаковом разряде содержится «1», то в способе изображения отрицательных чисел формально имеется небольшое отступление от естественных весов разрядов.

При использовании дополнительных кодов алгебраическое сложение-вычитание $B \pm C$ может выполняться следующим методом:

(а) если выполняется вычитание, то обратить код числа *C* (в том числе изменить и знак *C* на противоположный);

(б) выполнить суммирование чисел *B* и *C,* обращаясьпри этом с разрядом алгебраического знака точно так же, как с остальными (младшими) разрядами чисел; если идетвычитание, то при выполнении суммирования подать единицу на вход переноса младшего разряда.

Никакой предварительной расшифровки знаков не требуется: правило выполнения сложения, как и правило выполнения вычитания, всегда одно и то же, независимо от знаков чисел; каковы бы ни были алгебраические знаки чисел, сложение всегда остается сложением, а вычитание вычитанием.

Проверим правильность приведенного алгоритма для случая выполнения с л о ж е н и я . Фактически выполнение сложения сводится к простому суммированию содержимого регистров *B* и *C* (п. (а) пропускается). При этом важно только проследить, чтобы при суммировании двух положительных чисел не было переноса в разряд знака, а при суммировании двух отрицательных чисел перенос в разряд знака непременно был (невыполнение этих условий означает переполнение разрядной сетки); при суммировании чисел разных знаков перенос в разряд алгебраического знака может быть, а может и не быть.

Пусть сначала имеем два п о л о ж и т е л ь н ы х числа. При этом, как говорилось, в основных разрядах содержатся их абсолютные величины (| *B* | и | С |), в разрядах знака — нули. При суммировании получим в основных разрядах величину | *B* | + | *C* |; если эта величина меньше единицы (как это и требуется — иначе было бы переполнение), то перенос в разряд знака отсутствует, и при суммировании в этом разряде сохранится нуль (0 + 0 + 0 = 0). Таким образом, результат суммирования представляет собой правильный результат сложения двух положительных чисел.

Пусть теперь имеем два о т р и ц а т е л ь н ы х числа. В основных разрядах при этом содержатся дополнения до единицы от их абсолютных величин (1 — | В |) и (1 — | *C* |), в разрядах алгебраических знаков — единицы. При суммировании основных разрядов в них образуется величина-

$$(1—|B|) + (1—|C|) = 1 + 1—(|B| + |C|).$$

Если бы эта величина была меньше единицы, то это означало бы, что | *B* | + | *C* | > 1, т. е. имели бы переполнение разрядной сетки.- Нормально, когда $|B| + |C| < 1$, результат суммирования в основных разрядах больше единицы; поэтому в основных разрядах остается величина 1—(| *B* | +| *C* |)
и возникает перенос в разряд знака. С учетом этого переноса суммирование в разряде знака дает

$$1 + 1 + 1 = 1 \pmod 2,$$

т. е. сохраняет знак «—». Полученный результат (знак «—» и величина 1 — (| *B* | + | *C* |) в основных разрядах) представляет собой сумму двух отрицательных чисел *B* и *C,* записанную, как это и требуется, дополнительным кодом.

Предположим, наконец, что складываются два числа *разных знаков,* скажем $B \geq 0$ и $C < 0$. Для числа *B* основные разряды содержат величину |*B*|, а разряд знака — «0», для *C* в основных разрядах содержится 1 — |*C* |, в разряде знака — «1». Суммирование в основных разрядах дает величину

$$|B| + (1—|C|) = 1 + |B|—|C|.$$

Если $|B| \geq |C|$, то эта величина больше единицы; при этом в основных разрядах остается |*B* | — |*C* |, а единица

[*]) В последовательном представлении чисел разряд знака теперь будет следовать после всех остальных разрядов числа (так как он является старшим).

переносится в разряд знака, где суммирование даст

$$0+1+1=0 \pmod 2,$$

т. е. знак «+». Если $|B|<|C|$, то $1+|B|-|C|<1$, в основных разрядах остается вся эта величина, а перенос в разряд знака отсутствует; в разряде знака суммирование теперь даст $0+1+0=1$, т. е. знак «—». В первом случае ($|B| \geq |C|$) получаем положительный результат сложения (знак «+» и величина $|B|-|C|$ в основных разрядах), во втором случае ($|B|<|C|$) — отрицательный результат сложения, выраженный, как и требуется, дополнительным кодом (знак «—» и величина $1-(|C|-|B|)$ в основных разрядах).

Выполнение вычитания, как видно из приведенного алгоритма, отличается тем, что перед суммированием производится обращение кода вычитаемого, а при суммировании добавляется лишняя единица младшего разряда. Таким образом, фактически код вычитаемого заменяется дополнительным кодом, т. е. числом с обратным знаком, которое и складывается обычным порядком с уменьшаемым.

Примеры

1) $\left(+\frac{5}{16}\right)+\left(+\frac{6}{16}\right)$:

(исходные числа)	(B)	0.	0	1	0	1
	+					
	(C)	0.	0	1	1	0
(цифры переноса)		(0).	(1)	(0)	(0)	⓪
(результат суммирования)		0.	1	0	1	1 $\left(+\frac{11}{16}\right)$

2) $\left(-\frac{5}{16}\right)+\left(-\frac{6}{16}\right)$:

(исходные числа)	(B)	1.	1	0	1	1
	+					
	(C)	1.	1	0	1	0
(цифры переноса)		(1).	(0)	(1)	(0)	⓪
(результат суммирования)		1.	0	1	0	1 $\left(-\frac{11}{16}\right)$

3) $\left(-\frac{5}{16}\right)+\left(+\frac{6}{16}\right)$:

(исходные числа)	(B)	1.	1	0	1	1
	+					
	(C)	0.	0	1	1	0
(цифры переноса)		(1).	(1)	(1)	(0)	⓪
(результат суммирования)		0.	0	0	0	1 $\left(+\frac{1}{16}\right)$

4) $\left(+\frac{5}{16}\right)+\left(-\frac{6}{16}\right)$:

(исходные числа)	(B)	0.	0	1	0	1
	+					
	(C)	1.	1	0	1	0
(цифры переноса)		(0).	(0)	(0)	(0)	⓪
(результат суммирования)		1.	1	1	1	1 $\left(-\frac{1}{16}\right)$

5) $\left(+\frac{5}{16}\right)-\left(-\frac{6}{16}\right)$:

(исходные числа)	(B)	0.0101				
	−					
	(C)	1.1010				
(обращение кода C)	(B)	0.	0	1	0	1
	+					
	(C)	0.	0	1	0	1
(цифры переноса)		(0).	(1)	(0)	(1)	①
(результат суммирования)		0.	1	0	1	1 $\left(+\frac{11}{16}\right)$

6) $\left(-\frac{5}{16}\right)-\left(+\frac{6}{16}\right)$:

(исходные числа)	(B)	1.1011
	$+$	
	(C)	0.0110
(обращение кода C)	(B)	1. 1 0 1 1
	$+$	
	(C)	1. 1 0 0 1
(цифры переноса)		(1).(0)(1)(1)①
(результат суммирования)		1. 0 1 0 1 $\left(-\frac{11}{16}\right)$

Примеры выполнения вычитания при одинаковых знаках уменьшаемого и вычитаемого читателю предлагается проделать самостоятельно. Желательно рассмотреть также примеры, в которых получается переполнение разрядной сетки.

Как видим, применение дополнительных кодов позволяет построить более простой и систематический алгоритм сложения-вычитания, чем при использовании прямых кодов. Оборудование собственно арифметических цепей, правда, остается примерно таким же, как при использовании прямых кодов; зато заметно упрощается схема управления.

Кроме того, применение дополнительных кодов дает значительную экономию по времени. Очевидно, что в параллельном устройстве сложение в дополнительных кодах занимает время τ_Σ, а вычитание $\tau_0 + \tau_\Sigma$ (где τ_0 — время обращения кода, τ_Σ — время суммирования); если полагать, что эти операции равновероятны, то среднее время выполнения сложения-вычитания равно $\frac{1}{2}\tau_0 + \tau_\Sigma$. В последовательном устройстве сложение или вычитание в дополнительных кодах в любом случае можно выполнить за время одного периода числа. Нетрудно проверить, что приведенный алгоритм сложения-вычитания без каких-либо изменений пригоден и в тех случаях, когда дополнительный код строится по второму варианту (см. 1.4.2).

В параллельных устройствах с цепью циклического переноса можно было бы построить аналогичный алгоритм сложения-вычитания *в обратных кодах.*

Вместе с тем, как мы увидим из дальнейшего, применение дополнительных или обратных кодов создает некоторые затруднения при выполнении других операций (скажем, умножения и деления). Во многих случаях в цикле машины выполнение сложения или вычитания перекрывается по времени с работой запоминания устройства, а умножение и деление не укладываются в этот интервал времени. Поэтому, несмотря на неоспоримые преимущества дополнительных кодов при выполнении сложения-вычитания, в машинах все же чаще используются прямые коды*).

3.1.4. Об использовании добавочных сумматоров.

До сих пор, рассматривая методы выполнения алгебраического сложения-вычитания, мы рассчитывали на тот примерно объем оборудования арифметических цепей, который показан на рис. 3-1. Между тем иногда незначительное увеличение общего объема оборудования может очень существенно повлиять на скорость выполнения операции.

Предположим, например, что в схему последовательного устройства рис. 3-1, *б* (стр. 123) введен дополнительно еще один полусумматор с линией задержки на 1 такт. Включение добавочного полусумматора в схему показано сокращенно (без вспомогательных элементов) на рис. 3-3. Как мы сейчас увидим, также незначительное усовершенствование позволит вдвое сократить максимальное время выполнения сложения-вычитания в прямых кодах; фактически оно уравнивает длительность сложения-вычитания в прямых кодах с длительностью сложения-вычитания в дополнительных кодах.

Выполнение сложения-вычитания в схеме рис. 3-3 идет в общем по основному алгоритму (стр. 121). Отклонение состоит в том, что при вычитании одновременно с выполнением основного суммирования (п. (в)) на добавочный полусумматор подается обратный код получающегося результата. Добавочный полусумматор складывает его с единицей младшего разряда, т. е. вычисляет дополнительный код от результата суммирования.

*) Последнее не означает, что можно не заботиться о скорости выполнения элементарного суммирования: элементарное суммирование приходится многократно производить в процессе выполнения умножения и деления.

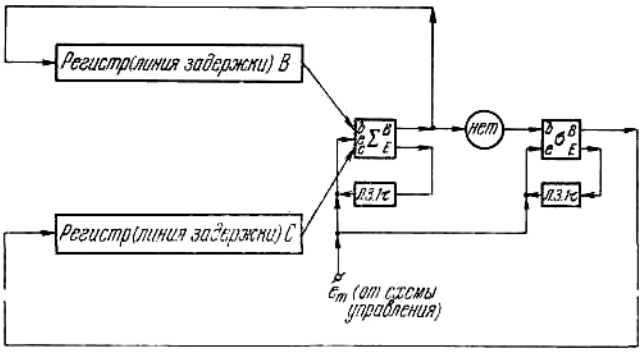


Рис. 3-3. Построение арифметических цепей для выполнения сложения-вычитания (последовательный вариант) с включением дополнительного полусумматора. (Вспомогательные элементы и большинство связей со схемой управления не показаны — ср. с рис. 3-1, *б* на стр. 123.)

Результат суммирования передается по-прежнему в регистр B (с выхода основного сумматора), а дополнение от него одновременно записывается в регистр C (с выхода добавочного полусумматора). Таким образом, попутно с выполнением п. (в) мы на всякий случай выполняем также пп. (г) и (д). К концу суммирования выясняется, действительно ли нужно было выполнять пп. (г) и (д). Если это так (если в последнем такте суммирования на выходе основного сумматора не получился сигнал переноса E_1), то окончательный результат операции прочтем из регистра C. Если в последнем такте суммирования на выходе переноса основного сумматора получили $E_1 = 1$, то окончательный результат операции нужно прочесть из регистра B. Таким образом, сложение или вычитание в прямых кодах в устройстве рис. 3-3 выполняется всегда в течение одного периода числа.

Разумеется, вместо добавочного полусумматора можно было бы иметь добавочный полный, одноразрядный сумматор, на один из входов которого при выполнении сложения-вычитания всегда подавался бы нуль. Как видно из раздела 4.6, такой добавочный сумматор очень полезен при выполнении умножения; он позволяет примерно вдвое увеличить скорость выполнения умножения.

Можно придумать, вероятно, и другие схемы для сокращения времени сложения-вычитания.

3.2. Сложение и вычитание с плавающей запятой

3.2.1. Общий метод выполнения операций.

Пусть необходимо сложить или вычесть два числа, β и γ, записанные с плавающей запятой. Для определенности будем полагать, что используется двоичная система счисления, хотя все изложенное ниже легко обобщить для любого основания системы счисления n.

В соответствии с 1.5.2 при представлении чисел с плавающей запятой имеем

$$\beta = 2^b B,$$
$$\gamma = 2^c C,$$

где b и c — целые числа (они называются порядками), а B и C удовлетворяют условию

$$1 > |C| \geq {}^1/_2,$$
$$1 > |C| \geq {}^1/_2;$$

величины B и C называются мантиссами чисел β и γ. Выполнение последних неравенств указывает на то, что числа нормализованы; иногда допускается наряду с нормализованными числами запись нуля в ненормализованной форме— с мантиссой, равной нулю. Если применяются только нормализованные числа, то старший разряд мантиссы можно не записывать в запоминающее устройство, так как информация в нем известна заранее. Но при этом для нуля нужно установить какое-нибудь условное изображение: скажем, наибольший по абсолютной величине отрицательный порядок и мантисса, равная ${}^1/_2$; при построении операций сложения-вычитания с плавающей запятой важно проследить, чтобы добавление нуля к некоторому числу или вычитание из данного числа нуля не изменяло бы этого числа.

Разрядная сетка машины разделена на две части; в одной из них хранятся порядки чисел («разряды порядка»), в другой — мантиссы чисел («разряды мантиссы»). Например, при размещении числа β в каком-либо регистре в разрядах порядка записывается величина b (со своим алгебраическим знаком), а в разрядах мантиссы — величина B (тоже со своим знаком, который является знаком всего числа β).

Поскольку применяется двоичная система счисления, как порядок, так и мантисса записываются по двоичной системе, позиционным способом, с естественными весами разрядов (небольшое отступление может быть сделано из-за отрицательных величин: могут применяться прямые, или дополнительные, или обратные коды). При этом в изображении порядка запятая фиксирована обычно справа от младшего разряда (все порядки целые), а в изображении мантиссы — слева от старшего из основных разрядов (все мантиссы дробные; слева от запятой может быть разряд знака); для нормализованных чисел старший из основных разрядов мантиссы (с весом ${}^1/_2$) всегда

содержит цифру «1».

Таким образом, изображение числа с плавающей запятой состоит из двух чисел с фиксированной запятой: порядка и мантиссы. Если по ходу дела нам придется производить отдельно сложение или вычитание порядков или мантисс, то это можно сделать по обычным правилам сложения-вычитания чисел с фиксированной запятой, изложенным в 3.1 (о некоторых особенностях сказано ниже).

В дальнейшем мы будем называть тот регистр арифметического устройства, в который первоначально принимается число β и в котором в конце образуется результат, регистром β, а тот регистр, в который первоначально принимается число γ,— регистром γ. Разряды порядков регистров β и γ мы будем иногда называть *регистрами порядковb* и *c*, а разряды мантисс регистров β и γ — соответственно *регистрами мантисс B* и *C*. Соответственно будут применяться термины *сумматор порядков* и *сумматор мантисс.*

Выполнение сложения или вычитания чисел с плавающей запятой должно начинаться с *выравнивания порядков.* Старший разряд мантиссы числа β, которому в изображении мантиссы приписывается вес $^1/_2$, в действительности — если учитывать порядок числа — имеет вес $^1/_2 \cdot 2^b$; старший разряд мантиссы числа γ в действительности имеет вес $^1/_2 \cdot 2^c$ и т. д. Прежде чем складывать или вычитать мантиссы, нужно поставить разряды с одинаковыми весами друг против друга. В этом и состоит смысл выравнивания порядков.

Для того чтобы выравнять порядки двух чисел, нужно найти разность порядков и затем сдвинуть мантиссу числа с меньшим порядком на столько разрядов вправо, сколько единиц содержится в абсолютной величине разности порядков. Сдвигать влево мантиссу числа с большим порядком, конечно, нельзя, так при этом за пределы регистра уходили бы старшие разряды; потеря младших разрядов при сдвиге вправо не играет такой роли и может быть учтена при округлении (см. 3.2.5).

После того как порядки выравнены, можно произвести сложение или вычитание мантисс.

Результату этой операции должен быть приписан порядок того числа, которое не сдвигалось, при выравнивании порядков. Если при вычислении разности порядков результат вычисления был передан, как обычно это делается, в разряды порядка одного из регистров, где он заменил находившийся там ранее порядок исходного числа, то теперь задача состоит в том, чтобы восстановить старший порядок.

Последний этап сложения-вычитания с плавающей запятой — это *нормализация результата.* Поскольку перед сложением или вычитанием мантисс абсолютная величина одной из них была заключена в интервале (1, $^1/_2$], а абсолютная величина другой (в результате выравнивания порядков) — в интервале (1, 0], то результат сложения-вычитания по абсолютной величине может находиться в интервале (2, 0]. Нормализовать результат — это значит ввести абсолютную величину его мантиссы в интервал (1, $^1/_2$] одновременно изменив порядок так, чтобы общая величина результата не изменилась. Если мантисса результата до нормализации находилась в интервале (2, 1], то нормализация результата состоит в сдвиге мантиссы вправо на один разряд и одновременном добавлении единицы к порядку; если до нормализации мантисса результата меньше $^1/_2$, то ее придется один или, может быть, несколько раз сдвигать влево, каждый раз вычитая по единице из порядка — до тех пор, пока цифра «1» не появится в старшем из основных разрядов мантиссы (если мантисса изображается прямым кодом).

Итак, выполнение сложения-вычитания с плавающей запятой состоит фактически из четырех этапов:

(а) выравнивание порядков;
(б) сложение-вычитание мантисс;
(в) восстановление старшего порядка;
(г) нормализация результата.

Пункты (б) и (в) могут выполняться одновременно или вообще в любой последовательности, так как один из них касается только мантисс, другой — только порядков.

Каждый из этих четырех этапов включает выполнение ряда более мелких операций. Скажем, выравнивание порядков включает вычитание двух чисел с фиксированной запятой, которое необходимо для вычисления разности порядков, выяснение знака полученной разности (для того чтобы установить, какой из порядков больше), сдвиг мантиссы числа с меньшим порядком и т. д. Ниже каждый из этапов рассматривается более подробно.

3.2.2. Детали выполнения выравнивания порядков.

При практическом осуществлении сложения и вычитания с плавающей запятой возникает много различных трудностей; известны разные способы их преодоления. Ниже кратко рассматриваются некоторые из относящихся сюда вопросов — не столько для того, чтобы дать решение всех проблем, которые могут появиться в каждом конкретном случае, сколько для того, чтобы показать читателю, какого рода это проблемы и какими способами приходится их решать.

Начнем с выравнивания порядков.

Прежде всего на этом этапе должна вычисляться разность порядков.

Вообще на всем протяжении операции сложения или вычитания чисел с плавающей запятой в разрядах порядков приходится выполнять только обычные сложения и вычитания или их частные случаи — добавление или вычитание единицы младшего разряда (прямой или обратный счет). Точно так же при выполнении умножения с плавающей запятой в разрядах порядков производится только сложение, а при выполнении деления с плавающей запятой — только вычитание (и тоже прямой или обратный счет при нормализации результата). Поскольку это так,

то для изображения отрицательных порядков имеет смысл применять дополнительные коды, в которых именно операции сложения и вычитания, а также счет выполняются наиболее удобным способом. Если в порядках применяются дополнительные коды, то вычисление разности порядков производится весьма просто — в соответствии с алгоритмом раздела 3.1.3.

Заметим, однако, что когда порядки исходных чисел имеют разные знаки, то разность порядков может не поместиться в разрядной сетке порядков. При этом, конечно, нет причин для остановки вычислений, так как это совсем не значит, что результат полной операции сложения или вычитания с плавающей запятой не сможет поместиться в разрядной сетке машины.

Проще всего это затруднение можно обойти, введя по одному лишнему разряду в сумматоре порядков и в регистрах, связанных с ним. В последовательном варианте вместо добавления лишнего разряда в сумматоре вводится лишний такт при суммировании порядков.

Если при изображении порядков принят дополнительный код (скажем, для определенности, по первому варианту, когда «+» изображается цифрой «0», а «—» — цифрой «1»), то добавочный разряд помещается слева от разряда знака. При приеме чисел из памяти (конечно, без добавочного разряда) в добавочном разряде устанавливается такая же цифра, какая принята в знаковый разряд. При этом можно считать, что знаковый разряд теперь имеет свой естественный вес $+2^p$, где p — количество основных разрядов порядка (вместо веса — 2^p, как предполагалось раньше), а добавочный разряд — вес — $2P^{+1}$. Что от этого ничего не меняется в положительных числах, очевидно сразу: оба эти разряда содержат нули, и

$$0\bullet\ (-2^{P+1}) + 0\bullet (+2^P) = 0 \bullet (+2^P) = 0.$$

Но и для отрицательных чисел присоединение добавочного разряда не меняет величину числа, так как

$$1\bullet (-2^{P+1}) + 1 \bullet (2^p) = 1 \bullet (-2^P) = -2^p.$$

Зато в разрядах порядков как бы добавляется один лишний основной разряд — с весом 2^P, а роль разряда знака выполняет теперь новый, добавочный разряд.

Далее перед началом сдвигов мантиссы необходимо проверить, каков алгебраический знак разности порядков. Если выполнялось вычитание $b - c$ и разность положительна, то это означает, что сдвигать вправо нужно будет C (мантиссу числа γ), потому что $b >$ c; если $b—c < 0$, то сдвигать вправо нужно будет B (мантиссу числа β).

Имеет смысл также проверить заранее, не слишком ли велика разность порядков. Если разность порядков по абсолютной величине превышает количество разрядов в мантиссе, то выполнение сдвигов в разрядах мантиссы было бы просто лишней потерей времени: мантиссу числа с меньшим порядком нужно сразу погасить (установить «0»); при сдвигах все цифры мантиссы все равно ушли бы за пределы регистра.

Выяснение этого обстоятельства не вызывает трудностей, если количество разрядов в мантиссе является целой степенью двойки без единицы. Например, если мантисса содержит 31 разряд ($2^5 - 1$), то признаком превышения разностью порядков количества разрядов в мантиссе является наличие единицы в 6-м разряде разности порядков (его вес равен 2^5) или единиц слева от него. Если применяются дополнительные коды, то для отрицательной разности порядков нужно было бы, наоборот, проверять наличие нулей в 6-м разряде или слева от него; последнее, однако, не совсем точно, так как наименьшее по абсолютной величине отрицательное число с нулем в 6-м разряде есть — 33, а не —32.

Но и когда количество разрядов в мантиссе является каким-нибудь другим числом, отличным от 2^k—1, можно пользоваться аналогичным упрощенным критерием. Например, в вычислительной машине М-2 мантиссы содержали 26 основных разрядов, но сдвиги мантиссы заменялись гашением только при условии, что разность порядков по абсолютной величине больше или равна 32. При этом в сравнительно редких случаях (когда разность порядков равна 27, 28, 29, 30 или 31) тратится лишнее время на выполнение сдвигов; зато отпадает необходимость в сравнении разности порядков с числом 27, что нужно было бы делать каждый раз и что занимало бы по длительности примерно столько же, сколько одно суммирование порядков.

В действительности, как мы увидим из 3.2.5, может оказаться, что для упрощения схемы округления блокирование сдвигов и гашение мантиссы выгодно вводить только в том случае, когда абсолютная величина разности порядков превышает количество разрядов в мантиссе хотя бы на 2 (т. е., например, при 26 разрядах в мантиссе и при разности порядков 27 еще нужно производить сдвиги, а при разности порядков 28 уже не нужно).

Заметим еще, что в случаях, когда разность порядков слишком велика, можно не только избежать сдвигов при выравнивании порядков, но и существенно сократить всю дальнейшую операцию. В этих случаях фактически выравнивание порядков обращает в нуль число с меньшим порядком. Поэтому результат полной операции сложения или вычитания либо просто равен исходному числу с большим порядком, либо равен этому же числу с обратным знаком (если оно является вычитаемым). Однако реализовать такое сокращение можно лишь при наличии специальных схем в устройстве управления.

Ясно, что разность порядков вообще может оказаться слишком большой по абсолютной величине по сравнению с количеством разрядов мантиссы только при условии, что

$$\log_2 s \le p,$$

где s — количество основных разрядов в разрядной сетке мантиссы, p— количество основных разрядов в разрядной

сетке порядков. Однако это условие почти всегда выполняется при проектировании машин с плавающей запятой в дальнейшем тоже рассматриваются только те случаи, когда указанное соотношение между s и p справедливо. Сам по себе сдвиг мантиссы вправо не вызывает особых трудностей. Ясно, что при выполнении сдвига освобождающиеся слева основные разряды регистра мантиссы должны заполняться нулями, если абсолютная величина мантиссы представлена прямым кодом, и единицами — если дополнительным или обратным кодом. Разряд знака в сдвигах участвовать не должен. Младшие разряды, уходящие при сдвигах за пределы регистра, воспринимаются, возможно, схемой округления (см. 3.2.5).

Некоторые трудности представляет отсчет количества сдвигов и определение конца сдвигов. Если бы в регистрах мантиссы имелись цепи для сдвига на 1, 2, 4, 8 и т. д. разрядов (речь идет о параллельных регистрах), то тогда первый (младший) разряд разности порядков управлял бы цепью сдвига мантиссы на 1 разряд, второй разряд разности порядков — цепью сдвига на 2 разряда, третий — цепью сдвига на 4 разряда и т. д. Максимальное количество тактов сдвига равнялось бы примерно $\log_2 s$, где s — количество основных разрядов в мантиссе. Однако во многих случаях в регистрах мантиссы имеются только цепи сдвига на 1 разряд. При этом для отсчета количества сдвигов можно поступить, например, следующим образом.

Пусть в схеме управления имеется двоичный счетчик, содержащий такое количество разрядов q, чтобы 2^q было больше максимально допустимого количества сдвигов в мантиссе. Например, если количество основных разрядов в мантиссе s равно 26, а сдвиги блокируются при условии, что разность порядков превышает 32, т. е. при наличии единицы в 6-м разряде разности порядков, то счетчик должен содержать 5 двоичных разрядов. Заметим, что количество основных разрядов в порядках p больше или равно количеству разрядов счетчика.

Примем в счетчик q младших разрядов абсолютной величины разности порядков, выраженной дополнительным кодом (а если $p = q$ — то все разряды разности порядков). При этом, когда разность порядков отрицательна, ее абсолютная величина сразу выражена дополнительным кодом, и прием в счетчик осуществляется прямо. Если же разность порядков положительна, то прием в счетчик нужно выполнить с одновременным обращением кода (т. е. вместо нулей принять единицы, а вместо единиц — нули) и перед началом отсчета сдвигов добавить один раз единицу в счетчик.

Затем одновременно с каждым сдвигом мантиссы вправо на один разряд будем добавлять по единице в счетчик. В тот момент, когда будет выполнен последний из необходимых сдвигов, счетчик перейдет из последнего состояния в нулевое. При этом на выходе его старшего разряда появится сигнал для запуска следующего (несуществующего) разряда. Этот сигнал и будет восприниматься как команда прекращения дальнейших сдвигов.

Обратим внимание на случай, когда разность порядков равна нулю. Если для изображения порядков применяются дополнительные коды, то нуль изображается в виде положительного числа: в его выражении имеется знак «+» (цифра «0» или «1» в разряде знака — в зависимости от того, по какому варианту построен дополнительный код) и нули во всех основных разрядах. Поскольку это так, прием в счетчик из основных разрядов идет с обращением кода; в основные разряды принимаются все единицы, т. е. счетчик уже при приеме устанавливается в последнее состояние. Далее перед началом сдвигов должна добавляться единица. Она переведет счетчик из последнего состояния в нулевое, причем появится команда «прекращение сдвигов». Следовательно, фактически в этом случае не будет выполнено ни одного сдвига — как это и должно быть, когда разность порядков равна нулю.

Вместо того чтобы устраивать две цепи приема в счетчик — прямого или с обращением кода,— можно было бы воспользоваться реверсивным счетчиком (см. раздел 2.8.4).

Можно также воспользоваться вместо счетчика регистрами и сумматором порядков, добавляя или вычитая единицы из имеющейся разности порядков. Однако для этого нужно сначала погасить разряды порядка с, а к концу отсчета сдвигов мы потеряли бы также информацию из разрядов порядка b (имевшуюся там ранее разность порядков $b — c$); после этого восстановить старший порядок непосредственно уже невозможно. Поэтому если счетчик сдвигов в схеме управления отсутствует и мы хотим вместо него воспользоваться регистрами и сумматором порядков, то необходимо предусмотреть специальный регистр, в котором хранился бы старший порядок. Передача в этот регистр осуществляется довольно просто в том случае, когда сумматор порядков параллельный и когда выполнение двоичных переносов в нем может быть отделено от процесса формирования цифр суммы (то же условие, что в разделе 3.1.2). При этом передачу старшего порядка удобно произвести в тот момент времени, когда в ходе вычисления разности порядков образованы цифры двоичных переносов при суммировании, но еще не выдана сумма: в этот момент времени уже известно, какой из порядков больше (об этом можно судить по сигналу переноса из старшего разряда сумматора), но сами коды порядков b и c еще не подвергались изменениям (только обращен код c).

Могут быть, конечно, и другие варианты отсчета количества сдвигов с сохранением возможности восстановить старший порядок.

3.2.3. Детали выполнения сложения-вычитания мантисс.

В отличие от порядков, при изображении которых большей частью применяются дополнительные коды, для изображения мантисс используют большей частью прямые коды. Связано это как с тем, что при выполнении умножения и деления мантиссы чисел приходится перемножать или делить, так и с тем, что применение здесь

прямых кодов не ведет к такому удлинению операций сложения и вычитания, как для чисел с фиксированной запятой.

Прежде всего ясно, что само по себе выполнение сложения или вычитания мантисс, которое в ходе полной операции сложения-вычитания чисел с плавающей запятой приходится выполнять один раз, не играет существенной роли в общем балансе времени.

Во-вторых, при выполнении вычитания мантисс в прямых кодах почти всегда заранее известно, какая из мантисс больше по абсолютной величине. Поскольку исходные числа нормализованы, абсолютные величины их мантисс в начале операции заключены в интервале $(1, {}^1/_2]$; если при выравнивании порядков одна из мантисс была сдвинута хотя бы на разряд вправо, то она стала по абсолютной величине меньше, чем ${}^1/_2$, следовательно, наверняка меньше, чем мантисса другого числа. Поэтому выполнение вычитания мантисс нужно всегда начинать с обращения кода той мантиссы, которая сдвигалась вправо при выравнивании порядков, т. е. с обращения кода мантиссы числа с меньшим порядком. После этого останется лишь выполнить суммирование (с посылкой единицы на вход переноса младшего разряда), а вычисление дополнительного кода от результата наверняка не потребуется.

Только в сравнительно редких случаях равенства порядков нельзя сказать заранее, какая из мантисс больше. Но и при этом второй такт обращения кодов и второй такт суммирования (или второй период числа, если речь идет о последовательных устройствах) потребуются не в ${}^1/_3$ всех случаев (как мы имеем при операциях с фиксированной запятой), а только в среднем в ${}^1/_4$ случаев, потому что комбинации чисел, в которых сумма мантисс превышает по абсолютной величине единицу, не исключаются заранее при программировании (см. рис. 3-2 на стр. 123).

Заметим, кстати, что переполнение разрядной сетки мантисс не должно приводить к остановке вычислений, как при операциях с фиксированной запятой. Оно является лишь признаком того, что при нормализации мантиссу результата нужно будет сдвинуть вправо. Как и для порядков, здесь нужно будет предусмотреть на этот случай лишний разряд в регистрах мантиссы и в сумматоре.

Отличие сложения-вычитания мантисс от сложения-вычитания чисел с фиксированной запятой может состоять еще и в том, что посылкой сигналов на вход переноса при суммировании младших разрядов заведует не только устройство управления, но и схема округления (см. 3.2.5). В интересах упрощения схемы округления иногда обращение кода мантиссы числа с меньшим порядком производится до сдвигов вправо; таким образом, когда выполняется вычитание, то при выравнивании порядков сдвигается уже обратный код мантиссы.

3.2.4. Детали выполнения нормализации результата сложения-вычитания.

Как уже говорилось, результат сложения-вычитания мантисс по абсолютной величине заключен в интервале $(2, 0]$.

Если мантисса результата до нормализации больше единицы (но при этом всегда меньше двух), то нормализация состоит в сдвиге мантиссы на один разряд вправо.

Одновременно нужно добавить единицу к порядку результата. При этом важно проследить, чтобы добавление единицы не вызвало переполнения разрядной сетки порядков; добавочный разряд, который мы ввели для вычисления разности порядков исходных чисел, здесь не должен, конечно, приниматься во внимание.

Признаком переполнения является наличие знака «+» у порядка до суммирования и появление переноса единицы в знаковый разряд при выполнении суммирования. Иначе о переполнении разрядной сетки порядков можно судить по тому, что после добавления единицы порядок результата содержит «0» в добавочном разряде (слева от знакового) и «1» в знаковом разряде (как и выше, предполагается, что применены дополнительные коды, построенные по 1-му варианту).

При обнаружении переполнения разрядов порядка в этом случае дальнейшие вычисления нужно остановить: результат полной операции сложения-вычитания с плавающей запятой не помещается в разрядной сетке машины. Вместо остановки вычислений могут быть предусмотрены другие варианты поведения машины в случае переполнения; об этом говорилось в 3.1.1. (стр. 122).

Рассмотрим теперь случай, когда мантисса результата до нормализации меньше ${}^1/_2$ абсолютной величине. Так может получиться, если в ходе операции выполнялось вычитание мантисс.

Как мы уже говорили, нормализация результата в этом случае состоит в сдвигах мантиссы влево до тех пор, пока не появится единица в старшем разряде мантиссы, т. е. до тех пор, пока не окажется, что абсолютная величина мантиссы больше или равна ${}^1/_2$ (предполагается, что в изображении мантисс применяются прямые коды); одновременно с каждым сдвигом мантиссы влево нужно вычитать по единице из порядка результата.

Особое внимание следует обратить при этом на случай получения малого по абсолютной величине результата. Может быть так, что сдвиги мантиссы влево еще не привели к появлению единицы в старшем разряде мантиссы, а порядок в результате вычитания единиц достиг уже минимально возможной величины (т. е. стал наибольшим по абсолютной величине отрицательным числом, которое может разместиться в разрядной сетке порядков — без учета, разумеется, добавочного разряда). Можно сказать, что здесь имеет место отрицательное переполнение разрядной сетки порядков.

Признаком отрицательного переполнения может являться то, что до вычитания порядок отрицателен, а при выполнении суммирования с кодом 111... 11 (дополнительный код числа —1) отсутствует перенос в разряд знака;

это означает, что перед суммированием код порядка был 100...00, т. е. максимальным по абсолютной величине отрицательным числом. Иначе об отрицательном переполнении порядков можно судить по тому, что после суммирования добавочный разряд (слева от знакового) сохраняет цифру 1, а в знаковом разряде порядка появилась цифра 0.

При появлении отрицательного переполнения в порядках можно поступать различно.

Иногда в этом случае прерывают нормализацию и разрешают результату участвовать в последующих операциях в ненормализованном виде.

Большей частью, однако, в случае отрицательного переполнения разрядной сетки порядков считают результат операции равным нулю и искусственно передают в регистр результата принятое условное изображение нуля в системе с плавающей запятой. При этом нуль оказывается единственным ненормализованным числом, что существенно упрощает многие операции[*]).

Частным случаем, когда может появиться отрицательное переполнение порядков, является случай вычитания равных между собой чисел с плавающей запятой. При этом выравнивания порядков нет (потому что порядки равны), а при вычитании мантисс получаются нули во всех разрядах мантиссы результата. Ясно, что сколько бы раз затем мантисса ни сдвигалась влево, единица в ее старшем разряде не появится. Рано или поздно процесс нормализации приведет при этом к отрицательному переполнению в порядках. Но если порядки исходных чисел велики, то до получения отрицательного переполнения порядков нужно много раз выполнить сдвиг мантиссы и вычитание единицы в порядках. Фактически же если количество сдвигов уже превышает количество разрядов в мантиссе *s,* а единица в старшем разряде мантиссы еще не появилась, то это значит, что единица там никогда и не появится, и в результате можно сразу установить нуль, не дожидаясь отрицательного переполнения порядков.

Поэтому, выполняя при нормализации сдвиги мантиссы влево, имеет смысл подсчитывать их количество тем же счетчиком, которым при выравнивании порядков подсчитывалось количество сдвигов вправо (если такой счетчик имеется в схеме управления — см. 3.2.2). Начав счет от нуля (к концу выравнивания порядков счетчик в таком положении и находится), в момент перехода счетчика из последнего состояния в нулевое получим от него команду прекращения нормализации.

Конечно, если бы все числа были равновероятны, то случай вычитания двух равных чисел встречался бы очень редко, и не стоило бы специально усложнять схему, чтобы сэкономить время именно в этом случае. Однако в реальных задачах вычитание равных между собой чисел выполняется значительно чаще, чем можно было бы ожидать. Кроме того, часто существенным является не среднее, а максимальное время выполнения операции, и тогда этот случай играет важную роль.

Для подсчетов полного времени выполнения сложения-вычитания с плавающей запятой заметим еще, что если при выравнивании порядков было выполнено хотя бы два сдвига вправо, то при нормализации результата потребуется **не** более одного сдвига влево. Скажем, если при выравнивании порядков мантисса C была сдвинута на 2 разряда вправо, то $|C| < {}^1/_4$; так как при этом $|B| \geq {}^1/_2$, то $|B| — |C| > {}^1/_4$.

3.2.5. Погрешности сложения и вычитания с плавающей запятой. Округление результата.

Сложение и вычитание с фиксированной запятой являются вполне точными операциями. Конечно, если исходные числа не были точными, то при выполнении сложения или вычитания с фиксированной запятой погрешность результата может оказаться равной сумме погрешностей исходных чисел. Однако никаких новых ошибок сложение или вычитание с фиксированной запятой не вносит.

Иное дело — сложение-вычитание с плавающей запятой. Даже если исходные числа были вполне точными, результат сложения или вычитания может оказаться приближенным.

Источниками погрешностей при выполнении сложения или вычитания с плавающей запятой являются сдвиг вправо мантиссы одного из исходных чисел при выравнивании порядков, сдвиг вправо мантиссы результата при нормализации, а также, может быть, искусственная установка нуля в регистре результата в случае отрицательного переполнения порядков (см. 3.2.4).

Когда при выравнивании порядков мантисса одного из чисел сдвигается вправо, за пределы регистра уходят младшие разряды мантиссы. После выполнения нескольких сдвигов, если каждый раз выдвигалась цифра «1», за пределами регистра может оказаться величина, равная примерно единице младшего разряда. Если эту величину просто не принимать во внимание, то результат сложения или вычитания мантисс имел бы погрешность порядка единицы младшего разряда. При этом в результате сложения мантисс погрешность абсолютной величины результата всегда отрицательна, а в результате вычитания — положительна (так как при вычитании мантисс то число, которое сдвигалось вправо, всегда является вычитаемым).

Полагая, что случаи сложения и вычитания мантисс равновероятны, найдем, что среднее значение ошибки в абсолютной величине результата равно нулю.

---

[*]) Во всем предыдущем изложении мы совсем не касаемся вопроса об операциях с условным (нормализованным) изображением нуля. Если бы, скажем, как говорилось в 3.2.1, нуль изображался бы числом $2^{-2^p} \cdot \frac{1}{2}$ (максимальный отрицательный порядок и мантисса ${}^1/_2$). То нужно было бы принимать специальные меры, чтобы добавление или вычитание нуля не изменило бы исходное число.

Построение схемы округления может преследовать цель уменьшения максимальных ошибок. Для этого можно поступить следующим образом.

Пусть при сдвиге мантиссы вправо последняя цифра, ушедшая за пределы регистра, т. е. старшая из выдвинутых цифр, запоминается специальным триггером. Если далее должно выполняться вычитание мантисс, то сдвиг производится после обращения кода; иначе говоря, в добавочном триггере будет храниться старшая из выдвинутых цифр обратного кода. Далее при выполнении суммирования на вход переноса младшего разряда будет подаваться та цифра, которая хранится в этом специальном триггере.

Если выполняется сложение, то нормально при суммировании на вход переноса младшего разряда подавался бы нуль. Следовательно, когда в нашей схеме мы подаем на этот вход единицу (в случае, когда старшая из отброшенных цифр есть 1), происходит, по сути дела лишнее добавление единицы младшего разряда. Но это как раз тот случай, когда отброшенные разряды составляют все вместе величину, равную по меньшей мере половине младшего разряда.

Следовательно, поступая описанным образом, мы получим при этом в абсолютной величине результата положительную погрешность, не превышающую половины младшего разряда. В случае, когда старшая из отброшенных цифр есть 0, на вход переноса младшего разряда подается нуль и, следовательно, ничего лишнего при суммировании не добавляется; абсолютная величина результата может иметь при этом отрицательную погрешность, но эта погрешность будет наверняка меньше половины младшего разряда.

Аналогичное положение получится и при вычитании.

Таким образом, мы снова получаем симметричное распределение погрешностей в абсолютной величине результата (со средним значением нуль), но максимальные ошибки теперь не превышают половины младшего разряда.

Заметим, что, вообще говоря, и распределение ошибок теперь более благоприятно. Хотя мы сочли, что в схеме без округления средняя ошибка равна нулю, но это было справедливо только при условии, что знаки «+» и «—» у чисел равновероятны. Если бы, скажем, вычислялась сумма ряда положительных чисел, то при отсутствии округления наверняка накопилась бы значительная отрицательная погрешность. При наличии округления, выполняемого указанным методом, этого нет.

Если при нормализации результата вновь производится сдвиг мантиссы результата вправо, то осуществить вслед за этим округление аналогичным способом было бы затруднительно. Дело в том, что сдвиг вправо при нормализации не сопровождается суммированием мантисс. Следовательно, для добавления лишней единицы младшего разряда нужно было бы специально произвести суммирование в разрядах мантисс. Но самое интересное состоит в том, что никакой пользы это не принесло бы.

Поскольку мантиссы, как мы условились, изображаются всегда прямыми кодами, сдвиг мантиссы на 1 разряд вправо при выполнении нормализации вносит в ее абсолютную величину либо нулевую погрешность (если выдвигается 0), либо погрешность, равную $-1/2$ младшего разряда (если выдвинута цифра 1). Если бы взамен выдвинутой за пределы регистра единицы мы всегда добавляли лишнюю единицу младшего разряда, то в итоге погрешность всегда была бы либо 0, либо $+1/2$ младшего разряда, что ничем не лучше, чем 0 или $-1/2$.

Схему округления после нормализации нужно построить так, чтобы в том случае, когда при сдвиге вправо за пределы регистра выдвинута цифра 1, происходила бы принудительная установка единицы в младшем разряде мантиссы результата. Если младший разряд и без того содержал единицу, то в нем, очевидно, ничего не изменится, и погрешность останется равной $-1/2$ младшего разряда. Если же младший разряд результата содержал нуль, то установка в нем цифры 1 эквивалентна добавлению единицы младшего разряда, и погрешность становится равной $+1/2$ младшего разряда. Как видим, указанный метод округления не уменьшает возможной абсолютной величины ошибки, но делает диапазон ошибок симметричным, а математическое ожидание ошибки сводит к нулю.

Необходимо заметить еще, что когда при нормализации выполняется сдвиг мантиссы результата вправо на 1 разряд, то та погрешность, которая имелась в мантиссе ранее (до $\pm 1/2$ младшего разряда, если после выравнивания порядков производилось округление), при сдвиге уменьшается вдвое. На нее накладывается дополнительно новая погрешность в $\pm 1/2$ младшего разряда, возникающая при нормализации. Таким образом, суммарная погрешность в этом случае становится равной $\pm 3/4$ младшего разряда.

Разберем теперь случай, когда при нормализации результата выполняется сдвиг мантиссы влево. Никаких новых погрешностей сдвиг влево, конечно не вносит. Однако тут возможны неприятности другого рода.

Если порядки исходных чисел были равны между собой, то при нормализации результата может потребоваться большое количество сдвигов влево. Однако при этом выравнивания порядков не было, никаких погрешностей в мантиссе результата, следовательно, нет, и сдвиг ее влево можно производить без всяких опасений.

Если порядки исходных чисел отличались на 2 или более, то при нормализации результата может потребоваться не больше одного сдвига влево (см. 3.2.4). При этом та погрешность, которая имелась в мантиссе результата ранее (до $\pm 1/2$ младшего разряда, если после выравнивания порядков производилось округление), при сдвиге влево удваивается и может оказаться равной ± 1 младшего разряда.

Наибольшую опасность представляет случай, когда порядки исходных чисел отличались на единицу. При этом, с одной стороны, мантисса результата может содержать некоторую погрешность (потому что при выравнивании порядков одна из мантисс сдвигалась вправо), с другой стороны, при нормализации результата

может потребоваться большое количество сдвигов влево. Если не принять специальные меры, то сдвиги влево могут вывести погрешность мантиссы результата, имевшуюся в ней ранее, чуть ли не в старшие разряды мантиссы. К счастью, в этом случае при выравнивании порядков был произведен всего один сдвиг вправо, и выдвинутая за пределы регистра цифра еще хранится в специальном триггере схемы округления.

Этой информацией можно воспользоваться, чтобы при первом же сдвиге мантиссы результата влево полностью исправить имеющуюся в ней погрешность; после этого, если необходимо, можно выполнять все остальные сдвиги. Однако для этого, конечно, требуется специальное усложнение схемы.

Что касается, наконец, погрешностей, возникающих при искусственной установке нуля в случае получения малого результата (см. 3.2.4), то эти погрешности направлены всегда в сторону уменьшения абсолютной величины мантиссы. Сделать тут что-нибудь схемным путем затруднительно[*]). Просто программист должен учитывать, что для достижения достаточной точности нужно несколько отойти от нижнего предела допустимых чисел.

3.3. Операции сравнения

3.3.1. Содержание операций сравнения.

Операции сравнения — наиболее важные из логических операций, выполняемых вычислительной машиной.

В литературе по вычислительной технике имеется некоторый разнобой в использовании термина «сравнение». Иногда сравнением называют поразрядную операцию (в двоичной системе счисления), дающую в любом из разрядов результат нуль, если в двух исходных числах цифры данного разряда не совпадают между собой, и единицу — если совпадают. Мы же будем относить термин «сравнение» к операции, выполняемой над полными числами, а не над их отдельными разрядами; при этом ответ сравнения (результат операции) имеет вид одноразрядного двоичного или, может быть, троичного числа. Смысл операций сравнения, как мы их понимаем, состоит в том, чтобы ответить, какое из двух заданных чисел больше, а какое меньше. Получив два исходных числа (скажем, B и C), арифметическое устройство должно, например, дать ответ одного вида, если $B > C$, и ответ другого вида, если $B \leq C$. Иногда ответ второго вида дается только в случаях $B < C$, но может быть сформирован еще и ответ третьего вида — в случае $B = C$; иногда же требуется, чтобы ответ одного вида выходил в случае $B \neq C$, а ответ другого вида— в случае $B = C$ («сравнение на равенство»). Модификацией этих операций является сравнение только абсолютных величин чисел — без учета их алгебраических знаков. Если в машине предусмотрено и полное сравнение чисел, с учетом знаков, и сравнение абсолютных величин, то первое
иногда называют алгебраическим сравнением, а второе — сравнением по модулю.

Ответ сравнения может вырабатываться в арифметическом устройстве в виде состояния некоторого триггера. Скажем, если этот триггер к концу выполнения сравнения находится в положении «0», то это значит, что $B > C$, а если в положении «1» — то $B \leq C$. Иначе ответ может даваться в виде электрического импульса, появляющегося на одном из двух проводов или каким-нибудь другим способом. Если сравнение может дать один из трех результатов (скажем, $B > C$, или $B < C$, или $B = C$), то его ответ изображается в виде комбинации состояний двух триггеров, или в виде состояния 3-позиционного кольца, или в виде импульса на одном из трех проводов и т. д.

После того как в арифметическом устройстве получен тот или иной ответ сравнения, вторая часть этой операции доделывается устройством управления машины. В зависимости от результата сравнения дальше выбирается та или иная команда, т. е. последующие вычисления направляются по одному или по другому пути (условный переход).

Операции сравнения применяются для отсчета выполненных циклов вычислений, для определения конца итерационного процесса по достижению заданной точности и во многих других случаях. При этом сравнение по модулю необходимо в общем чаще, чем алгебраическое сравнение[**]). В некоторых машинах вместо операций сравнения предусматривается операция условного перехода по алгебраическому знаку предыдущего результата. Если перед условным переходом выполнялось вычитание двух чисел, то это дает примерно то же, что и операция сравнения. Однако такое решение усложняет программу и требует большего времени (вместо одной команды сравнения требуются две команды — вычитание и условный переход)[***]); с другой стороны, как мы увидим из 3.3.2., условный переход по знаку разности не вполне эквивалентен операции сравнения и не полностью ее заменяет.

3.3.2 Выполнение алгебраического сравнения чисел с фиксированной запятой.

Пусть необходимо выполнить сравнение двух чисел, B и C, каждое из которых представлено с фиксированной запятой. Для определенности будем полагать, что числа записаны в двоичной системе цифрами 0, 1 позиционным способом, с естественными весами разрядов, и что запятая фиксирована перед первым из основных разрядов числа;

---

[*]) Если процесс нормализации не приводит к получению нормализованного числа, то перед искусственной установкой нуля можно попытаться добавить к мантиссе результата $^1/_4$ (единицу второго разряда); если и при этом не появится единица в первом (старшем) разряде мантиссы, то тогда уже наверняка нужно установить в результате «0». Это дало бы более симметричный диапазон ошибок.

[**]) Например, при определении конца итерационного процесса в общем случае нужно применять сравнения по модулю; конец обнаруживается по тому, что результаты двух последовательных итераций мало отличаются друг от друга по абсолютной величине.

[***]) По крайней мере это справедливо для трехадресных машин.

возможное отступление от позиционного способа изображения чисел или от естественного порядка весов может быть связано с применением дополнительных кодов для отрицательных чисел. Впрочем, все сказанное ниже легко обобщить на случаи аналогичного изображения чисел в любой *n*-ичной системе и для запятой, фиксированной в любом другом месте.

Сравнения обычно выполняются теми же арифметическими цепями, которые предназначены для выполнения сложения-вычитания, может быть — с добавлением нескольких специальных элементов.

Говоря в дальнейшем о выполнении сравнения, мы будем иметь в виду те операции, когда ответ одного вида должен получаться при $B \leq C$, ответ другого вида — при $B > C$. Модификации, в которых требуется особый ответ в случае равенства чисел ($B = C$), встречаются сравнительно редко.

На первый взгляд наиболее естественным представляется тот путь выполнения сравнения, который упоминался в 3.3.1: произвести вычитание двух заданных чисел $B — C$ и прочесть ответ сравнения по знаку результата. Если результат отрицателен, то $B < C$, если положителен — то $B > C$. При этом, однако, нужно принять во внимание два существенных затруднения.

Первое из них состоит в том, что вычитание исходных чисел может привести — если алгебраические знаки их различны — к переполнению разрядной сетки машины. Если бы на самом деле выполнялась операция вычитания, то в этом случае нужно было бы остановить вычисления или по меньшей мере дать специальный сигнал ошибки, свидетельствующий, что результат вычислений неверен. При сравнении чисел не нужно делать ни того ни другого, так как нас интересует только знак разности. Однако необходимо учитывать, что, когда операции ведутся в дополнительных кодах при переполнении разрядной сетки, может получиться неправильным и знак разности (см. метод сложения-вычитания в разделе 3.1.3).

Вторая трудность проявляется только при использовании прямых кодов для изображения чисел. Она связана с неоднозначностью представления нуля. Если в прямых кодах выполняется вычитание двух равных чисел, то результатом может быть и + 0, и —0 в зависимости от алгебраических знаков исходных чисел. Например, выполняя вычитание равных чисел по методу 3.1.1, получим, что разность двух положительных чисел равна — 0, а разность двух отрицательных чисел + 0. Еще хуже, если оба исходных числа равны нулю; действуя указанным методом, получим

$$(+0) — (+0) = — 0; \quad (—0) — (—0) = +0;$$

$$(+0) — (—0) = — 0; \quad (—0) — (+0) = +0.$$

Хотя числа во всех случаях равны между собой, знак разности получается то «+», то «—»; если бы не были приняты специальные меры, то и ответ сравнения получался бы каждый раз разный. Между тем большей частью желательно, чтобы ответ сравнения был определенным: скажем, либо $B \leq C$, либо $B > C$.

Рассмотрим пути преодоления указанных трудностей.

Возможность переполнения разрядной сетки при вычитании не представляет особой опасности, если операции выполняются в прямых кодах. Можно просто заблокировать цепи останова дальнейших вычислений или формирования сигнала ошибки (при появлении переполнения), а знак разности все равно получится правильным (по крайней мере если сложение-вычитание выполняется одним из методов разделов 3.1.1 или 3.1.2.); следовательно, правильным будет и ответ сравнения. Правда, тот факт, что в ходе вычитания получалось переполнение, нужно все же запомнить. Эта информация может потребоваться, когда дальше мы попытаемся выяснить, не является ли результат вычитания нулем, т. е. не равны ли между собой исходные числа.

Если операции выполняются в дополнительных кодах, то с возможностью переполнения справиться сложнее. Так или иначе нужно, конечно, заблокировать цепи остановки вычислений или формирования сигнала ошибки. Но важно еще получить и правильный знак разности.

Применяя дополнительные коды, построенные по 1-му варианту («0» соответствует знаку « + », «1» — «—»), можно поступить следующим образом. Добавим по одному лишнему разряду в регистрах B и C и в сумматоре — слева от знакового разряда; в последовательном устройстве вместо добавления лишнего разряда в сумматоре нужно добавить, лишний такт суммирования. Если при приеме исходных чисел установить в добавочных разрядах такие же цифры, как в разрядах знака, то дальше можно считать, что бывший знаковый разряд является обычным разрядом целых со своим естественным весом +1, а добавочный разряд — это новый разряд знака с весом —2. При этом в любом случае, в том числе и при переполнении прежней разрядной сетки (без добавочного разряда), в добавочном разряде получим правильный знак разности. Точно таким же образом мы поступали в 3.2.2, при вычислении разности порядков чисел с плавающей запятой.

Интересна и другая возможность. Переполнение разрядной сетки при вычитании возможно лишь в том случае, если знаки исходных чисел B и C различны. Но при этом знак разности $B — C$ всегда совпадает со знаком исходного числа B. Следовательно, в начале сравнения можно проверить, каковы знаки сравниваемых чисел, и если они различны, то вообще не делать никакого вычитания; о результате сравнения можно судить сразу по знаку B[*]).

При последовательном способе операций этот метод неудобен тем, что знаки следуют в конце, после всех

[*]) В прямых кодах тоже можно сократить операцию таким образом. Но нужно учитывать, что исходными числами разных знаков могут быть, в частности, +0 и —0.

основных разрядов чисел. Чтобы не тратить зря лишний период числа на выяснение знаков, имеет смысл на всякий случай, пока проходят основные разряды чисел, начать выполнение вычитания, а в конце периода числа, если окажется, что знаки чисел различны, выполнение вычитания прервать.

Теперь относительно другой трудности, о которой мы говорили выше.

Обнаружение равенства исходных чисел при операциях в прямых кодах можно осуществить, например, следующим путем.

После того как закончено вычитание, подадим на один из входов сумматора разность B — C, на другой — число «111...11» и выполним суммирование. (Число 111...11 можно получить, например, погасив один из входных регистров, т. е. установив в нем код 000...00, и затем выполнив в нем обращение кода).

При выполнении суммирования (B — C) + (.111... 11) обратим внимание на сигнал переноса из самого старшего разряда. Если среди цифр, изображающих величину B — С, имеется хотя бы одна единица, то появится единица переноса из старшего разряда.

Отсутствие сигнала переноса на выходе старшего разряда указывает, что все основные разряды регистра B содержат нули.

Это, правда, еще не значит, что разность B — C наверняка равна нулю. Если, например $B = +^1/_2$, а $C = —^1/_2$, то B — C = 1, и все основные разряды регистра B (дробная часть величины B — C) тоже содержат нули. Но в подобных случаях имелось переполнение разрядной сетки в ходе вычитания.

Итак, признаком равенства исходных чисел ($B = C$; B — C = 0) является одновременно отсутствие переполнения разрядной сетки при вычитании и отсутствие переноса единицы из старшего разряда при дополнительном суммировании.

При выявлении равенства сравниваемых чисел можно искусственно установить, скажем, знак «—» в результате вычитания, так что к концу операции сравнения «—» в разряде знака результата означал бы $B \leq C$, *а* «+» $B > C$. Если необходимо, то можно сформировать специальный ответ сравнения в случае $B = C$.

Обнаружение равенства можно выполнить иными, может быть, более удобными методами, если в параллельном сумматоре возможно отделить формирование цифр переноса от формирования цифр суммы (такой сумматор рассматривался нами в связи со вторым методом сложения-вычитания в прямых кодах — см. 3.1.2). При этом можно действовать, например, следующим образом.

Равные числа, если они появятся, должны иметь, между прочим, и одинаковые знаки*). Поэтому, когда в ходе выполнения сравнения будет производиться их вычитание— скажем, по первому методу (3.1.1),— то перед суммированием будет произведено обращение кода B, а во время суммирования от схемы управления будет послана единица на вход переноса младшего разряда сумматора. При этом наличие единицы на выходе переноса старшего разряда свидетельствовало бы о том, что $|B| \leq |C|$ (так как (1— $|B|$) + $|C| \geq 1$), а нуль на выходе переноса старшего разряда означал бы, что $|B| > |C|$ (так как (1 — $|B|$) + $|C| < 1$). Если образование цифр переноса обнаруживает случай $|B| \leq |C|$, то можно вместо формирования цифр суммы вновь повторить процесс образования переносов, но при этом не подавать единицу на вход переноса младшего разряда; вместо суммирования (1 — $|B|$) + $|C|$ будет как бы выполняться суммирование (1—2^{-m} — $|B|$) + $|C|$, где 2^{-m}— вес младшего (m-го) разряда, а 1—2^{-m} — $|B|$ — обратный код величины $|B|$. Ясно, что в случае $|B| < |C|$ при таком суммировании тоже возник бы перенос из старшего разряда, так как минимальная разница между $|C|$ и $|B|$ есть 2^{-m}; если же $|B| = |C|$, то (1— 2^{-m} — $|B|$) + $|C| < 1$, и без единицы на входе переноса младшего разряда перенос на выходе старшего разряда не возникает. Таким способом при одинаковых знаках B и C можно обнаружить равенство их абсолютных величин; это будет свидетельствовать о том, что числа B и C вообще равны между собой.

Имея сумматор указанного типа, обнаружить равенство абсолютных величин сравниваемых чисел можно еще и иначе. Предположим, что мы сначала обратили код $|B|$ и сформировали сигналы переноса по всем разрядам, а потом вновь обратили код $|B|$ (т. е. вернули его к исходному состоянию), одновременно обратили код C и опять сформировали сигналы двоичного переноса. Зафиксируем, какие сигналы переноса появились на выходе старшего разряда в одном и в другом случае. Если каждый раз, формируя сигналы переноса по всем разрядам, мы посылали единицу на вход переноса младшего разряда, то признаком равенства $|B| = |C|$ является появление в обоих случаях единицы на выходе переноса старшего разряда сумматора. Действительно, при равенстве $|B| = |C|$ и только при равенстве $|B| = |C|$ получим

$$(1—|B|)+ |C|\geq 1$$

и

$$(1—|C|)+ |B|\geq 1.$$

Если бы при формировании цифр переноса по всем разрядам на вход переноса младшего разряда каждый раз посылался бы нуль, то признаком равенства $|B| = |C|$ было бы, наоборот, отсутствие единицы на выходе переноса старшего разряда в обоих случаях.

Вообще сумматор, в котором формирование цифр переноса может быть отделено от формирования цифр суммы, очень удобен при выполнении сравнений. Мы в этом убедимся еще раз в 3.3.3, когда речь будет идти о сравнении по модулю. Там же, кстати, излагается и еще один метод выполнения алгебраического сравнения.

*) Исключение составляют только + 0 или — 0; но этот случай в дальнейших рассуждениях опущен.

Все тонкости, о которых здесь шла речь, можно уловить, конечно, только в том случае, когда сравнение выделено в отдельную операцию. Если вместо сравнения предполагается выполнять просто вычитание чисел и затем условный переход по знаку разности, то все эти тонкости будут опущены. Как видим, это совсем не эквивалентная замена.

3.3.3. Выполнение сравнения по модулю чисел с фиксированной запятой.

Сравнение по модулю производится очень просто в прямых кодах и достаточно сложно в дополнительных кодах. Мы не приводим здесь алгоритмов выполнения сравнения по модулю в дополнительных кодах; в качестве полезного упражнения читатель может попытаться составить их самостоятельно. Ниже речь пойдет только о сравнении по модулю чисел, представленных в прямых кодах.

Один из способов выполнения этой операции состоит в том, что сравнение по модулю сводится к алгебраическому сравнению. Если после приема исходных чисел принудительно установить у всех чисел знак «+», то тем самым будут найдены абсолютные величины чисел. Далее их можно сравнивать обычным путем, как выполнялось бы алгебраическое сравнение. В арифметических цепях и в схеме управления при этом потребуются минимальные усложнения. Но в действительности сравнение по модулю можно выполнить значительно более коротким путем.

Сделаем, скажем, так.

Приняв исходные числа в регистры B и C, не обращая внимания на знаки, обратим код одного из чисел (например, B) и выполним двоичные переносы по всем разрядам. Если устройство сумматора не позволяет отделить процесс формирования цифр переноса от образования суммы, то это в обгцем не играет роли (только ведет к лишней потере времени), потому что нас будет далее интересовать лишь сигнал переноса из старшего разряда и больше ничего. Относительно входа переноса младшего разряда можно поступить любым способом: либо всегда подавать на него «1», либо всегда подавать «0». В первом случае наличие единицы на выходе переноса старшего разряда будет соответствовать неравенству $|B| \leq |C|$ (так как эта единица указывает, что $(1 - |B|) + |C| \geq 1$), а отсутствие единицы — неравенству $|B| > |C|$. Во втором случае наличие единицы на выходе переноса старшего разряда будет соответствовать неравенству $|B| < |C|$, а отсутствие единицы — неравенству $|B| \geq |C|$ (появление единицы переноса из старшего разряда указывает теперь, что $(1 - 2^{-m} - |B|) + |C| \geq 1$, а отсутствие единицы — что $(1 - 2^{-m} - |B|) + |C| < 1$, где 2^{-m} — вес младшего разряда, $(1 - 2^{-m} - |B|)$ — обратный код B).

Таким образом, сигнал переноса из старшего разряда является по существу готовым ответом сравнения по модулю.

Интересно, что аналогичный способ можно применить для выполнения алгебраического сравнения в дополнительных кодах. Напомним (см. 1.4.2), что при построении дополнительного кода по второму варианту — когда знак «+» соответствует цифре «1» в знаковом разряде — изображение любого числа x представляет собой фактически величину $1 + x$*), записанную прямым кодом в разрядной сетке, включая и разряд знака. Каково бы ни было x, его изображение $1 + x$ всегда положительно. Между тем ясно, что сравнение чисел B и C дало бы такой же ответ, как сравнение величин $1 + B$ и $1 + C$. Последнее можно делать по тем же правилам, по которым делается сравнение по модулю чисел, записанных прямым кодом.

Дополнительный код, построенный по первому варианту, всегда легко превратить в дополнительный код, построенный по второму варианту. Для этого достаточно, например, изменить на противоположную цифру в разряде знака.

Таким образом, алгебраическое сравнение в дополнительных кодах выполняется столь же просто, как сравнение по модулю в прямых кодах. Но это скорее недостаток, а не достоинство дополнительных кодов; как говорилось в 3.3.1, сравнение по модулю является более нужной операцией, чем алгебраическое сравнение, и важно обеспечить простые и быстрые способы выполнения именно для него.

3.3.4. Сравнение чисел с плавающей запятой.

Алгебраическое сравнение чисел с плавающей запятой, как и в случае фиксированной запятой, может быть сведено к вычитанию исходных чисел и индикации знака разности. Переполнение разрядной сетки при вычитании с плавающей запятой — случай настолько редкий, что для него не стоит предусматривать какие-нибудь специальные схемные решения: просто программист должен учитывать, что слишком большие по абсолютной величине числа разных знаков нельзя сравнивать, так же как нельзя вычитать их одно из другого. Получение нуля при вычитании с плавающей запятой всегда связано с определенными отличиями в процессе нормализации результата (см. 3.2.4); при этом изображение нуля устанавливается в регистре результата искусственным путем, вследствие чего оно всегда однозначно. Следовательно, и с этой стороны трудностей не предвидится. Таким образом, алгебраическое сравнение с плавающей запятой фактически ничем не отличается от вычитания с плавающей запятой с последующим переходом по знаку результата.

Поскольку мантиссы чисел с плавающей запятой изображаются обычно прямыми кодами, изменение

*) Как и выше, мы здесь полагаем, что запятая фиксирована между разрядом знака и первым из основных разрядов числа. В выражении $1 + x$ единица есть вес знакового разряда.

алгебраического знака числа заключается в изменении в нем только разряда знака мантиссы. Поэтому выполнение сравнения по модулю легко свести к выполнению алгебраического сравнения.

Казалось бы, все обстоит настолько просто, что не имеет смысла искать другие пути для выполнения сравнения чисел с плавающей запятой. Однако мы сейчас рассмотрим алгоритм сравнения, обладающий весьма существенными преимуществами по сравнению с изложенным выше[*]).

Пусть имеем два числа, β и γ, записанные с плавающей запятой. Для определенности предположим, что мантиссы чисел (соответственно B и C) записываются, как обычно, прямыми кодами, порядки (b и c) — дополнительными кодами. Существенно, что числа β и γ нормализованы.

Условимся, кроме того, что дополнительные коды для представления порядков построены по второму варианту, т. е. что знак «+» изображается цифрой «1» в знаковом разряде, а знак «—» — цифрой «0». Напомним (см. 1.3.2), что при этом изображением любого числа x является фактически величина $\delta_0 + x$, где δ_0 — вес знакового разряда; этим обстоятельством мы только что пользовались в 3.3.3. Для любых x величины $\delta_0 + x$ положительны.

Теперь поступим следующим образом. Запишем подряд, слева направо, сначала изображение порядка b числа β (в том числе, слева от других разрядов, и алгебраический знак порядка), затем основные разряды его мантиссы B и затем где-нибудь отдельно алгебраический знак мантиссы. Будем считать, что таким образом мы получим изображение в прямом коде некоторого числа с фиксированной запятой, β'; число β' содержит $1 + p + s$ основных разрядов (разряд знака порядка, p — количество основных разрядов в изображении порядка числа β, s — количество основных разрядов в изображении мантиссы), а разряд знака мантиссы числа β является также разрядом знака для β'.

Возьмем для примера какое-нибудь число β; скажем, $\beta = -{}^3/_{16} = 2^{-2}$ $(-{}^3/_4)$. Предположим для простоты, что разрядная сетка порядков содержит 3 основных разряда и разряд знака, а разрядная сетка мантисс — четыре основных разряда и разряд знака. Применяя дополнительный код, построенный по второму варианту, получим изображение порядка ($b = -2$) в виде 0110, слева находится разряд знака порядка с весом $\delta_0 = 8$; изображение порядка $b = -2$ является фактически числом $\delta_0 + b = 8 + (-2) = 6$. Мантисса ($B = {}^3/_4$) в прямом коде изображается в виде . 1100. Итак, число β представлено в системе с плавающей запятой в виде

$$\underbrace{0110}_{b}\ \underbrace{.1100}_{B}$$

Соответствующее число β' есть. 01101100— (т. е. $\beta' = -{}^{27}/_{64}$). Точно таким же образом поступим с числом y, причем получим число с фиксированной запятой γ'.

Легко видеть, что *сравнение чисел* β и γ, *представленных с плавающей запятой*, *можно заменить сравнением чисел с фиксированной запятой* β' и γ', *составленных указанным образом.*

Действительно, знаки у чисел β' и γ' те же, что и у чисел β и γ. Следовательно, нам остается только проверить, что равенству $|\beta| = |\gamma|$ соответствует равенство $|\beta'| = |\gamma'|$, соотношению $|\beta| > |\gamma|$ — соотношение $|\beta'| > |\gamma'|$ и соотношению $|\beta| < |\gamma|$ — соотношение $|\beta'| < |\gamma'|$.

Случай равенства очевиден; если $\beta = \gamma$, т. е. если $b = c$ и $B = C$, то совпадают все разряды величин β' и γ′ и, следовательно, $\beta' = \gamma'$.

Неравенство $|\beta| > |\gamma|$ возможно либо при условии, что $b > c$ (поскольку числа нормализованы, то мантиссы их при этом не играют роли), либо при условии, что $b = c$, но $|B| > |C|$. В первом случае в $(1 + p)$ старших разрядах числа $|\beta'|$ содержится большая величина, чем в соответствующих разрядах числа $|\gamma'|$; при этом $|\beta'| > |\gamma'|$ независимо от содержимого младших разрядов. Во втором случае старшие разряды чисел $|\beta'|$ и $|\gamma'|$ совпадают между собой, но в s младших разрядах числа $|\beta'|$ содержится большая величина, чем в соответствующих разрядах $|\gamma'|$; следовательно, снова получаем $|\beta'| > |\gamma'|$

Точно так же доказывается, что неравенству $|\beta| < |\gamma|$ всегда соответствует неравенство $|\beta'| < |\gamma'|$.

Подмена сравнения с плавающей запятой сравнением с фиксированной запятой не требует почти никаких усложнений в арифметических цепях. Скажем, если идет речь о параллельных устройствах, единственное, что требуется для этого — это обеспечить передачу сигнала переноса с выхода старшего разряда сумматора мантисс на вход младшего разряда сумматора порядков. С полученным таким образом $(1 + s + p)$-разрядным параллельным сумматором, как и с регистрами β и γ, нужно будет обращаться, конечно, как с единым целым; например, если выполняется суммирование, то оно выполняется по всем разрядам одновременно, если обращается код β то обращение производится одновременно и в разрядах порядка b, и в разрядах мантиссы B.

Таким образом, простейшими средствами нам удалось свести выполнение сравнения с плавающей запятой к выполнению сравнения с фиксированной запятой. Его можно выполнять одним из методов сравнения с фиксированной запятой в прямых кодах, описанных выше. При этом речь может идти и об алгебраическом сравнении, и о сравнении по модулю.

Основное преимущество этого метода состоит в значительном выигрыше в скорости. Мы избавляемся сразу и от

[*]) Этот метод, предложенный автором настоящей книги, был впервые реализован в вычислительной машине М-2 (см.«Быстродействующая вычислительная машина М-2», под ред. И. С. Брука, Гостехиздат, 1957, стр. 56).

выравнивания порядков, и от нормализации результата, и от всех других атрибутов вычитания с плавающей запятой.

Дополнительное преимущество состоит в том, что сравнение с плавающей запятой становится вполне точной операцией. Если бы мы действовали через вычитание с плавающей запятой, то при сравнении близких между собой, но неравных чисел вычитание могло бы дать ответ «0» (а сравнение, следовательно,— ответ «равенство») просто потому, что результат вычитания не может быть представлен нормализованным числом в разрядной сетке машины (см. разделы 3.2.4 и 3.2.5).

С той точностью, с какой выполняется вычитание, числа действительно можно считать равными, но на самом деле они все же неодинаковы. Сравнение, выполняемое путем перехода к фиксированной запятой, заметит эту разницу.

3.3.5. О сравнении на равенство.

В предыдущих разделах мы неоднократно встречались с обнаружением случаев равенства сравниваемых чисел. Если это необходимо, то в случае обнаружения равенства можно было бы сформировать специальный ответ сравнения.

Но по крайней мере с теоретической точки зрения важно отметить, что между сравнением на равенство и выяснением того, какое из чисел больше, а какое меньше, имеется принципиальное отличие. Когда нас интересует, какое из двух данных чисел больше, то существенную роль играют веса разрядов в изображениях этих чисел; сравнение выполняется как бы последовательно: если цифры в старшем по весу разряде сравниваемых чисел совпадают, то проверяется следующий по весу разряд и т. д. Отображением такого последовательного сравнения является процесс распространения сигнала двоичного переноса вдоль цепочки разрядов, который существенным образом входил в рассмотренные выше алгоритмы[*]).

Если целью является только обнаружение равенства или неравенства двух чисел, то веса их разрядов никакой роли не играют, как не играет роли и вообще позиционный характер способа записи чисел. Сравнение на равенство можно выполнять не только последовательно, разряд за разрядом, но и одновременно по всем разрядам.

Простейшая схема такого вида показана на рис. 3-4. Исходные числа размещены в двух регистрах, составленных из триггеров. Выходы каждой пары соответствующих друг другу триггеров объединяются в двух элементах «и»; если цифры в этих триггерах одинаковы (0 и 0 или 1 и 1), то оба эти элемента вырабатывают сигнал «0», если цифры разные (0 и 1 или 1 и 0), то один из элементов «и» дает сигнал «1». Выходные сигналы от всех элементов «и» поступают в общую схему «или»; если хотя бы в одном двоичном разряде сравниваемых чисел имеется несовпадение цифр, то элемент «или» дает сигнал «1»; только при полном равенстве чисел на выходе «или» получится сигнал «0».

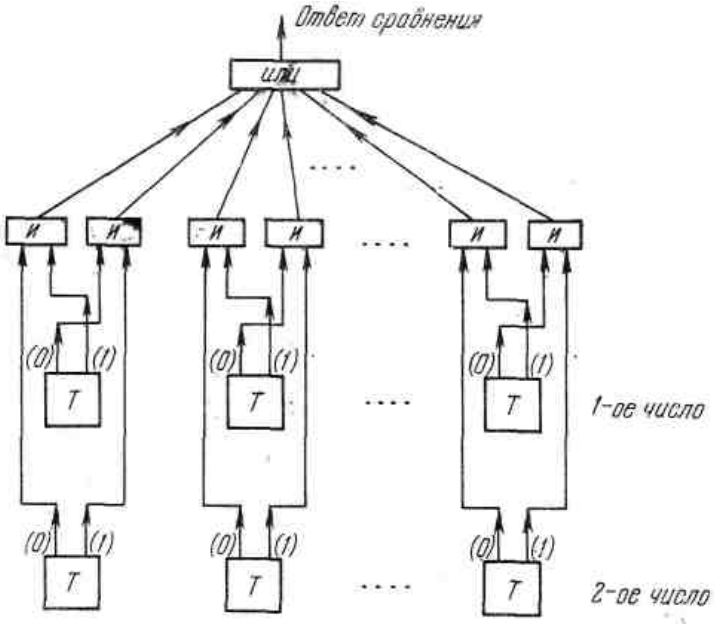


Рис. 3-4. Простейшая схема выполнения сравнения на равенство.

[*]) Хотя сигналы переноса распространяются от младших разрядов к старшим, цифра переноса на выходе самого старшего разряда зависит в первую очередь от старших разрядов и во вторую очередь от младших. Если при суммировании двух чисел в каком-либо разряде имеются два нуля или две единицы, то данный разряд сразу выдает сигнал переноса «0» или соответственно «1»; только при наличии комбинаций 0, 1 или 1, 0 выходной сигнал данного разряда зависит от сигнала переноса из предыдущих младших разрядов. При выполнении сравнения к началу распространения переносов одно из чисел обычно записано в прямом коде, другое — в обратном; поэтому комбинации 0, 0 и 1, 1 в некотором разряде соответствуют несовпадению цифр данного разряда в сравниваемых числах, а комбинации 0, 1 и 1, 0 — совпадению этих цифр.

Такая схема, полностью игнорирующая смысл разрядов сравниваемых чисел, неудобна только в тех случаях, когда допускается неоднозначность в изображении некоторых чисел. Например, +0 и — 0 воспринимались бы такой схемой, как неравные числа.

Как мы уже говорили, сравнение на равенство редко фигурирует в списке операций вычислительной машины в качестве самостоятельной операции. Но как элементарная операция, выполняемая, например, специальной схемой рис. 3-4, сравнение на равенство может оказаться весьма полезным.

4. Умножение

В программах решения различных задач операции умножения в общем встречаются реже, чем сложение, вычитание и сравнение, вместе взятые. Тем не менее во многих задачах оказывается, что главную часть времени машина занята выполнением умножений, так как одно умножение требует, как правило, больше времени, чем одно сложение, вычитание или сравнение.

Поэтому методам выполнения умножения, способам его ускорения и рациональному построению множительных устройств всегда уделялось значительное внимание в разработках и в теоретических исследованиях по цифровой технике. Мы попытаемся осветить эти вопросы возможно полнее.

Основную часть настоящей главы займут вопросы выполнения умножения чисел с фиксированной запятой, представленных позиционным способом с естественными весами разрядов в однородной n-ичной системе счисления цифрами 0, 1, 2, ..., (n — 1). Такой способ представления чисел будет предполагаться везде, где нет специальных оговорок. Фактически здесь речь будет идти о перемножении абсолютных величин чисел с фиксированной запятой, представленных прямыми кодами, или о перемножении абсолютных величин мантисс чисел с плавающей запятой (тоже в прямых кодах). Умножение чисел разных знаков и умножение с плавающей запятой рассматриваются в последних разделах главы.

4.1. Простые методы выполнения умножения

4.1.1. Основные идеи.

Основная идея обычных методов выполнения умножения рассмотрена бегло в разделе 1.2.3.

Дополним арифметические цепи, предназначенные для выполнения сложения и вычитания (см., например, рис. 3-1 на стр. 123), еще одним числовым регистром, который мыназовем регистром *A,* и, может быть, некоторыми вспомогательными элементами. Разряды знака пока исключим из рассмотрения.

Такое построение без каких-либо технических подробностей показано на рис. 4-1. Сумматор, включенный между регистрами *B* и *C*, может быть и последовательным, и параллельным. В последовательном устройстве вместо регистров могут использоваться линии задержки, в параллельном же устройстве регистры должны иметь ряд вспомогательных цепей (например, цепи сдвига). Однако эти и другие детали, как и вопрос о количестве разрядов в регистрах, *мы пока отложим.* Заметим только, что регистры *B* и *C* с сумматором могут использоваться также для выполнения сложения-вычитания; при этом в них необходимы цепи для обращения кодов чисел, которые в простых методах выполнения умножения не нужны, но часто полезны для ускорения умножения.

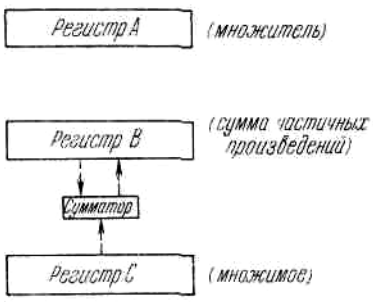


Рис. 4-1. Основная идея построения множительного устройства при использовании простых методов

Поместим в регистр *C* множимое, в регистр *A* — множитель, а регистр *B* к началу операции очистим (установим в нем число «0», т. е. выполним гашение *B).*

Выполним затем суммирование $B + C$ столько раз, сколько единиц содержится в цифре a_m младшего разряда регистра *A* (в двоичной системе — либо ни одного раза, либо один раз). Так как результат суммирования каждый раз направляется в регистр *B*, то в итоге в регистре *B* получится частичное произведение множимого *C* на младшую цифру множителя a_m.

Сдвинем затем содержимое регистра *C* на один разряд влево, т. е. увеличим множимое в n раз, где n — основание системы счисления. После этого снова выполним суммирование $B + C$ — теперь столько раз, сколько единиц содержится во второй справа цифре множителя a_{m-1} и т. д.

В результате m таких циклов, каждый из которых состоит из суммирований и сдвигов, в регистре *B* получается произведение *AC.*

Если речь идет о параллельном устройстве (полученном, скажем, путем усовершенствования схемы рис. 3-1, *а),* то для осуществления описанного процесса нужно дополнительно предусмотреть в регистре *B* цепь гашения, а в регистре *C* — схему сдвига влево на 1 разряд; кроме того необходимо предусмотреть цепь сдвига вправо на 1 разряд в новом регистре *A.* Сдвиг в регистре *A* будет выполняться всегда одновременно со сдвигом в регистре С; при этом в его крайнем справа разряде каждый раз будет находиться как раз та цифра множителя, которая в данный момент определяет количество суммирований; именно этот разряд и будет связан со схемой управления.

В последовательном устройстве, если в качестве регистров применяются линии задержки, сдвигу влево на 1 разряд соответствует удлинение задержки в цепи рециркуляции на 1 такт. (Для этого нужно, скажем, открыть

специальный вентиль, который направил бы импульсы через лишний 1-тактный элемент задержки в цепи рециркуляции.) Когда цепь рециркуляции удлиняется указанным образом, то оказывается, что в том такте, в котором нормально должна поступать цифра второго справа разряда числа, вместо нее выходит еще только цифра младшего разряда и т. д.— весь код оказывается сдвинутым по времени **в** сторону старших разрядов. Точно так же сдвигу вправо на 1 разряд соответствует укорочение задержки в цепи рециркуляции на 1 такт.

4.1.2. Четыре варианта осуществления основного метода выполнения умножения.

Процесс умножения, описанный в 4.1.1**,** за исключением некоторых несущественных деталей, совпадает с тем методом, которым умножение выполняется обычно вручную.

Однако не видно причин, почему нельзя, например, вычислить сначала частичное произведение множимого на старшую цифру множителя (Ca_1), затем сдвинуть влево это частичное произведение, после чего добавить к нему Ca_2 и т. д. Всего же, как мы сейчас увидим, имеется четыре различных варианта выполнения умножения**.**

Варианты эти отличаются один от другого необходимым количеством оборудования (в основном числом разрядов в регистрах и в сумматоре), отчасти скоростью выполнения умножения и некоторыми другими особенностями. Сравнение различных вариантов мы проведем применительно к параллельным устройствам. Иллюстрацией к этому является рис. 4-2.

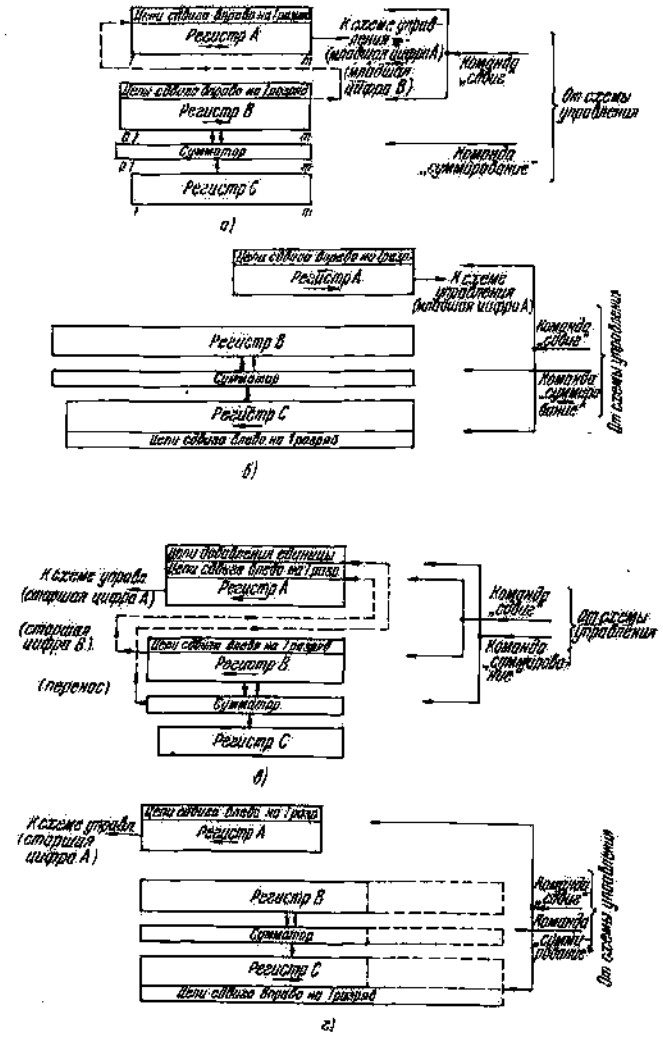


Рис. 4-2. Построение арифметического устройства параллельного типа при использовании различных вариантов выполнения умножения: *а*) 1-й вариант; *б*) 2-й вариант; *в*) 3-й вариант; *г*) 4-й вариант.

1°. *Первый вариант* (рис. 4-2, *а)* применяется при построении параллельных устройств чаще всего[*]). Как и в 4.1.1, умножение начинается здесь от младших разрядов множителя. Поэтому в регистре множителя *А* в каждом цикле умножения должен выполняться сдвиг вправо. Однако в отличие от 4.1.1 одновременно с этим вместо сдвига множимого влево производится сдвиг суммы предыдущих частичных произведений вправо (в регистре *В).*

[*]) Автор, по-видимому, отчасти повинен в том, что в литературе первый вариант называется иногда вариантом «а», а второй, третий и четвертый варианты — соответственно вариантами «б», «в», и «г»: описав подряд эти четыре варианта в своей предыдущей книге (см. предисловие), он не дал им никаких наименований, но поместил рисунок, аналогичный рис. 4-2, где иллюстрации к этим вариантам так же были отмечены буквами**.**

Основное достоинство этого варианта состоит в том, что для получения полного $2m$-разрядного произведения двух m-разрядных сомножителей нет необходимости в $2m$-разрядном регистре *B:* каждый раз, как при сдвиге вправо за пределы основных m разрядов регистра *B* уходит очередная из младших цифр произведения, для нее освобождается разряд в регистре *A*. Поэтому вместо продолжения регистра *B* на рис. 4-2, *а* изображена связь цепей сдвига регистров *B* и *A*, показанная стрелками; цифры, выдвигаемые справа из регистра *B,* попадают в старшие разряды регистра *A*. В итоге в регистре *A* получаются m младших цифр произведения, а в регистре *B* — его m старших цифр. Регистр *B* и сумматор должны содержать всего по $(m+1)$ разрядов. Один дополнительный разряд слева необходим для запоминания цифры переполнения, которая может получаться в процессе суммирования; при последующем сдвиге эта цифра уйдет в старший из основных разрядов регистра *B*, так что в конечном результате переполнения не будет, и цифра в дополнительном разряде — всегда
нуль (если оба сомножителя меньше единицы, то и их произведение меньше единицы).

2°. *Второй вариант* (рис. 4-2, *б*) выполнен в точности по описанию процесса умножения, имеющемуся в 4.1.1. и в разделе 1 (см. 1.2.3.). Умножение здесь тоже начинается от младших разрядов множителя, так что в регистре *A* каждый раз производятся сдвиги вправо. Но одновременно с этим производится сдвиг множимого *C* влево (к началу умножения множимое располагается в младших m разрядах регистра *C*).

Вариант этот не используется в параллельных устройствах потому, что он оказывается очень не экономным по количеству необходимого оборудования. Для его осуществления, очевидно, необходимо иметь $(2m - 1)$ разрядов в регистре *C* и по $2m$ разрядов в регистре *B* и в сумматоре, где m — число разрядов в сомножителях. При этом, например, разряды сумматора используются плохо: в начальных циклах умножения старшие разряды заняты все время «суммированием» нулей, в конце умножения на младшие разряды поступают из регистра *C* нули, т. е. никаких полезных операций они фактически не производят.

В то же время описанный вариант мог бы оказаться выгодным с двух точек зрения.

Во-первых, в этом варианте к началу умножения можно установить в регистре *B* вместо нуля какое-нибудь другое число — *B* (скажем, результат предыдущего умножения). Тогда в результате умножения мы получили бы в регистре *B* вместо произведения *AC* величину $B + AC$; это позволило бы легко организовать накопление суммы произведений пар чисел $(A_1C_1 + A_2C_2 + A_3C_3 + ...)$. В первом варианте этого сделать нельзя, так как там в процессе умножения первоначальное содержимое регистра *B* сдвигалось бы m раз вправо; если бы в регистре *B* первоначально размещалось какое-нибудь число *B*, то в итоге выполнения умножения мы получили бы вместо $B + AC$ величину $Bn^{-m} + AC,$ где n — основание системы счисления, m — количество разрядов.

Во-вторых, преимуществом рассматриваемого варианта могло бы явиться отсутствие сдвигов в регистре *B*. При определенных построениях схем это позволило бы совместить во времени прием последней для данного цикла умножения суммы в регистр *B* со сдвигом в регистрах *A* и *C* (в двоичной системе в каждом цикле умножения может быть только одно суммирование, которое, таким образом, всегда совмещалось бы во времени с тактом сдвига).

Однако, как мы увидим из дальнейшего, те же преимущества могут быть реализованы и более простыми средствами — без таких больших затрат оборудования, как в рассмотренном варианте.

3°. *Третий вариант* (рис. 4-2, *в)* отличается тем, что умножение начинается от старших разрядов множителя. Поэтому в каждом цикле умножения множитель в регистре *A* нужно будет сдвигать влево на 1 разряд. Сдвиг влево будем производить также и в регистре *B* (вариант, о котором говорилось в самом начале настоящего раздела).

Поскольку сдвиг в обоих регистрах *A* и *B* выполняется в одну и ту же сторону, здесь можно, как и в первом варианте, передавать в освобождающиеся разряды регистра *A* те цифры произведения, которые выходят слева за пределы регистра *B*. Однако в первом варианте за пределы регистра *B* при сдвигах уходили готовые младшие разряды произведения. Здесь же за пределы регистра *B* уходят старшие разряды предыдущей суммы частичных произведений; при последующих суммированиях с младшими частичными произведениями возможен перенос единиц в старшие разряды, находящиеся уже в регистре Л. Поэтому в те моменты времени, когда выполняются суммирования $B + C$, регистр *A* должен быть включен счетчиком, с тем чтобы единица переноса с выхода старшего разряда сумматора могла бы добавляться к содержимому регистра *A*. Например, если речь идет о двоичной системе, то регистр *A* может быть выполнен по схеме рис. 2-38 (стр. 97); наряду с цепью полных одноразрядных сумматоров, образующих основной параллельный сумматор, в устройство будет входить дополнительно еще цепь из полусумматоров, связанная с регистром *A*. Регистр *B* и связанный с ним параллельный сумматор должны содержать по $(m + 1)$ разрядов — тогда добавление единицы в регистре *A* никогда не приведет к изменениям в его старших разрядах — тех, где еще хранятся неиспользованные разряды множителя; добавление единицы будет затрагивать лишь те разряды регистра *A,* где находятся цифры, выдвинутые из *B*.

Таким образом, добавление цепей, превращающих регистр *A* в счетчик (скажем, добавление m полусумматоров),— это единственное усложнение в рассматриваемом устройстве по сравнению с первым вариантом. Однако это усложнение может компенсироваться тем, что цепи сдвига здесь такие, какие необходимы для выполнения деления (см. раздел 5); если бы умножение строилось по первому варианту, то для него нужно было бы предусматривать цепи сдвига вправо в регистрах *A и B, а* для деления — специальные цепи сдвига влево

в тех же регистрах.

Преимуществом рассматриваемого варианта является возможность организовать накопление сумм парных произведений: если перед началом умножения в регистр *B* поместить какое-нибудь число *B,* затем сдвинуть его на 1 разряд влево и начать процесс умножения *AC,* то к концу его получим величину $B + AC$[*]).

В рассматриваемом варианте — по сравнению с первым вариантом — выгодно и то, что старшие разряды произведения получаются в регистре *A*, младшие — в регистре *B,* а не наоборот. Как мы увидим из раздела 5, в таком же порядке в регистрах *A* и *B* могут быть получены в результате деления частное и остаток. Это удобно, когда для машины программируются операции с 2*m*-разрядными числами (с двойной точностью). В программе сложения 2*m*-разрядных чисел, кстати сказать, весьма полезна и возможность переноса единицы из старшего разряда сумматора в регистр *A*.

4°. *Четвертый вариант* выполнения умножения иллюстрируется рис. 4-2, *г.* Как и в третьем варианте, умножение начинается здесь от старших разрядов множителя, но вместо сдвига влево суммы частичных произведений в каждом цикле умножения производится сдвиг вправо множимого.

Поскольку сумма частичных произведений неподвижна, преимущества четвертого варианта те же, что и второго варианта: возможность совмещения по времени тактов сдвига и тактов суммирования и возможность организовать накопление суммы парных произведений. На первый взгляд кажется, что и оборудование требуется такое же: *m*-разрядный регистр *A,* (2*m* — 1)-разрядный регистр *C* и 2*m*-разрядные регистр *B* и сумматор. Однако во втором варианте лишние *m* разрядов, скажем, регистра *C* требовались для того, чтобы сдвигать в них старшие разряды множимого; здесь множимое сдвигается вправо, в сторону младших разрядов. В 4.1.4 мы увидим, что в тех случаях, когда не требуется получения полного 2*m*-раз-рядного произведения, по ходу дела можно производить округление множимого; для получения произведения с точностью до *m* разрядов при этом достаточно иметь всего 3—4 дополнительных разряда (сверх *m* основных разрядов) в регистрах *B* и *C* и в сумматоре. Если это так, то четвертый вариант почти не отличается по количеству оборудования от первого варианта, но сохраняет те преимущества, о которых говорилось выше. Те разряды регистров *B* и *C* и сумматора, которые необходимы только для получения полного 2*m*-разрядного произведения, на рис. 4-2, *г* показаны пунктиром.

Заметим, кроме того, что схема сдвига влево, необходимая для осуществления умножения по четвертому варианту, может быть использована при выполнении деления.

4.1.3. Округление результата умножения в конце операции.

Произведение двух *m*-разрядных сомножителей содержит вообще *2m* значащих цифр. При выполнении умножения по первому или по третьему варианту получение всех *2m* разрядов произведения не составляет труда; идя на известные затраты оборудования, их можно получить также и при использовании второго или четвертого варианта. Однако все дело в том, что большей частью такое количество разрядов не нужно, и результат умножения должен быть представлен с той же точностью, с какой были заданы исходные числа (до *m*-го разряда). Только в редких случаях — при программировании операций с удвоенной точностью — старшая и младшая части 2*m*-разрядного произведения записываются в раздельные ячейки оперативной памяти (или сохраняются в регистрах арифметического устройства) и используются в дальнейших операциях.

Если от 2*m*-разрядного произведения просто отбросить *m* младших разрядов, то старшие *m* разрядов представляли бы произведение с погрешностью от 0 до почти $-n^{-m}$, где n^{-m}—вес младшего разряда (*n* — основание системы счисления, *m* — количество разрядов).

1°. Наиболее очевидный способ округления состоит в том, чтобы после окончания умножения добавить лишнюю единицу младшего (*m*-го) разряда к старшей части произведения в том случае, если *(m* + 1)-й разряд (старший из отбрасываемых разрядов) содержит цифру $^{n}/_{2}$ или большую цифру. По существу это тот же метод округления, который постоянно применяется при ручном счете. Он, очевидно, уменьшает абсолютную величину возможной погрешности вдвое — до $n^{-m}/2$, а математическое ожидание величины погрешности сводит к нулю.

Добавление единицы к старшей части произведения может быть произведено либо через сумматор — если умножение выполнялось по первому варианту и старшая часть результата находится в *B*,— либо через счетные цепи регистра *A* — если умножение выполнялось по третьему варианту. Однако в обоих случаях для округления требуется время, равное примерно одному такту суммирования.

2°. Более простой способ округления состоит в том, что цифра младшего (*m*-го) разряда старшей части произведения произвольно увеличивается на единицу, если она меньше $^{n}/_{2}$, и оставляется неизменной, если больше или равна $^{n}/_{2}$. Например, в десятичной системе вместо цифр 0, 1, 2, 3, 4 в младшем разряде нужно было бы ставить соответственно 1, 2, 3, 4, 5, а цифры 5, 6, 7, 8, 9 не подвергались бы изменениям. В двоичной системе этот метод сводится просто к принудительной установке единицы в младшем разряде («0» заменяется на «1», цифра «1» остается неизменной, т. е. всегда получается «1»). Хотя абсолютная величина возможной ошибки при таком округлении не уменьшается — она остается равной n^{-m} — диапазон" возможных ошибок становится

[*]) Заметим, однако, что в отличие от второго варианта число *B* здесь может содержать только *m* разрядов. Если *B* — это результат предыдущего умножения, то его нужно до начала нового умножения округлить до *m* разрядов. Об округлении —см. 4.1.3.

симметричным относительно нуля, а математическое ожидание ошибки становится равным нулю (по крайней мере при четных, а также при любых достаточно больших n).

Схема округления при использовании второго способа проще, чем для первого способа, а времени тоже требуется меньше (так как при округлении отсутствуют переносы вдоль цепочки разрядов).

Интересно, что при использовании второго способа округления цифры, находящиеся в младших (отброшенных) разрядах произведения, не играют совсем никакой роли и никак не принимаются во внимание. Вариантами второго способа выполнения округления для системы счисления с четным основанием n являются либо увеличение на единицу цифры младшего (m-го) разряда в том случае, когда она четна, и оставление ее неизменной, когда она нечетна, либо принудительная установка цифры ${}^{n}l_2$ в младшем разряде. Если $n \neq 2$, то в последнем варианте абсолютная величина возможной погрешности получается больше, чем в двух предыдущих вариантах; при n=2 все три варианта эквивалентны. Общим недостатком всех этих вариантов округления является то, что значения младшей цифры результата перестают быть равновероятными; например, если в десятичной системе применяется увеличение на единицу всех цифр, меньших ${}^{n}/_{2}$ (в данном случае ${}^{n}/_{2}$ =5), то появление цифры 0 в младшем разряде результата становится невозможным, цифра 5 появляется с вероятностью 0,2, а каждая из остальных цифр (1, 2, 3, 4, 6, 7, 8, 9) — с вероятностью 0,1.

3°. Необходимо отметить, что наличие простых методов округления результатов умножения для чисел с фиксированной запятой или для мантисс чисел с плавающей запятой, является одним из наиболее серьезных достижений позиционного способа изображения чисел.

Дело в том, что при любом способе изображения чисел в машине совокупность чисел, с которыми оперирует машина, не образует группы относительно операции умножения (впрочем, и относительно некоторых других операций*). Причина этого состоит в том, что сомножители, содержащие по m значащих цифр, в общем случае дают произведение, содержащее $2m$ значащих цифр; при этом не имеет значения, что в машине могут быть предусмотрены и операции над $2m$-разрядными числами: произведение двух $2m$-разрядных сомножителей дает вообще $4m$-разрядное произведение. Внешнее выражение этого факта состоит в том, что при попытке организовать операции над целыми числами мы при выполнении цепочки умножений получали бы все время возрастающие числа, а при операциях над дробями — все больше и больше знаков в изображении дробной части числа.

Типовой способ преодоления этой трудности состоит в том, что в качестве объектов для операций выбираются правильные дроби (числа с фиксированной запятой, мантиссы чисел с плавающей запятой), а результат каждого умножения округляется до m значащих цифр — хотя бы путем отбрасывания младших m цифр из $2m$-разрядного произведения или другим способом. Таким образом, умножение фактически заменяется некоторой близкой к нему операцией («умножение с округлением»), относительно которой совокупность чисел, с которыми оперирует машина, образует группу.

Простота средств, с помощью которых в позиционной системе удается получить этот результат, не должна скрывать от читателя его важность.

Достаточно сказать, что один из наиболее известных и привлекательных символических способов изображения чисел — запись чисел в остатках (см. 1.3.4) — не получил распространения главным образом потому, что в нем не удается достаточно просто построить такой приближенный эквивалент умножения, относительно которого совокупность машинных чисел представляла бы собой группу.

4.1.4. Округление в процессе умножения.

Округление в процессе умножения применяется в том случае, когда умножение строится по четвертому варианту (см. 4.1.2, п. 4°), но произведение требуется получить лишь с точностью до m-го разряда, а особенно увеличивать количество разрядов в регистрах B и C и в сумматоре нежелательно.

Рассмотрим этот случай применительно к двоичной системе.

Представим себе, что в двоичном множительном устройстве, построенном по четвертому варианту (рис. 4-2, *г),* регистры B и C и сумматор содержат $m+r$ двоичных разрядов, где m — количество разрядов в сомножителях, а r— количество добавочных разрядов, $r < m$. Необходимую величину r мы попытаемся вычислить в данном разделе.

Пусть сначала округление отсутствует, так что разряды множимого, выходящие за пределы $(m+r)$-разрядного регистра C, просто пропадают. Найдем, какова может быть при этом максимальная погрешность $(m + r)$-разрядного произведения. Очевидно, что в течение первых r циклов умножения — пока множимое не начало выходить за пределы регистра C, никаких погрешностей в произведение не вносится. В $(r + 1)$-м цикле, если очередная $((r + 1)$-я) цифра множителя есть единица и младшая цифра множимого единица, будет внесена погрешность $-2^{-(m+r+1)}$) . В $(r + 2)$-м цикле погрешность вносится при условии, что $(r + 2)$-я цифра множителя есть 1; максимальной эта погрешность будет, если две младшие цифры множимого единицы (при этом погрешность будет равна $-2^{-(m+r+1)} - -2^{-(m+r+2)}$) и т. д. Суммарная погрешность произведения оказывается максимальной при перемножении чисел

*) Иными словами, результат выполнения умножения (либо соответственно другой операции) над двумя числами, каждое из которых может быть представлено в разрядной сетке машины, оказывается выходящим из разрядной сетки.

$$(\dots\underbrace{111\dots11}_{m-r\ \text{разрядов}})\times(\dots\underbrace{111\dots11}_{m-r\ \text{разрядов}});$$

она равна

$$\delta = -2^{-(m+r+1)}-2^{-(m+r+1)}-2^{-(m+r+2)}-2^{-(m+r+1)}-2^{-(m+r+2)}-2^{-(m+r+3)}-$$
$$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$$
$$-2^{-(m+r+1)}-2^{-(m+r+2)}-2^{-(m+r+3)}-\dots-2^{-2m}=-2^{-(m+r)}[m-r-1+2^{-(m-r)}]\approx$$
$$\approx -2^{-(m+r)}(m-r-1).$$

Потребуем теперь, чтобы эта погрешность была по абсолютной величине не больше половины младшего из полезных разрядов произведения (т. е. половины m-го разряда):

$$2^{-(m+r)}(m-r-1)\leq 2^{-(m+1)}.$$

Отсюда вытекает условие

$$2^{r-1}\geq m-r-1,$$

по которому можно определить необходимое количество дополнительных разрядов r. В частности,

$$\text{если } 13 < m \leq 22, \text{ то } r = 5,$$
$$\text{если } 23 \leq m \leq 39, \text{ то } r = 6,$$
$$\text{если } 40 \leq m \leq 72, \text{ то } r = 7.$$

Количество добавочных разрядов можно существенно сократить, введя округление в каждом цикле умножения.

Обозначим через r_0 количество дополнительных разрядов, необходимых при наличии округления. Схему округления, построим так, чтобы каждый раз, когда за пределы регистра множимого (С) выдвигается единица, в ближайшем такте суммирования (если, это суммирование есть, т. е. если очередная цифра множителя единица) добавлялась бы одна лишняя единица $(m + r_0)$-го разряда. Добавление лишней единицы можно осуществить, как обычно, через вход переноса младшего разряда сумматора.

При наличии такой схемы округления каждый цикл умножения, если в нем есть суммирование, может внести в результат положительную или отрицательную погрешность.

Положительная погрешность вносится в том случае, если непосредственно перед суммированием из множимого была выдвинута единица. Эта погрешность максимальна в том случае, когда ушедшая за пределы регистра единица является единственной, и во всех предыдущих циклах за пределы регистра множимого уходили нули. При этом за пределами регистра находится величина

..1000...,

а вместо нее добавляется величина

1..0000...

(двумя точками обозначена граница регистра). Погрешность в данном цикле умножения оказывается равной половине младшего разряда, т. е. $+2^{-(m+r_0+1)}$.

Отрицательная погрешность вносится в том случае, когда непосредственно перед суммированием из множимого был выдвинут нуль, но в предыдущих циклах умножения за пределы регистра множимого уходили единицы. Максимальной отрицательная погрешность данного суммирования получится в том случае, когда непосредственно перед суммированием из регистра C был выдвинут нуль, но в о в с е х предыдущих циклах выдвигались единицы; за пределами регистра при этом находится величина

..01111...,

а взамен нее при суммировании ничего не добавляется. Максимальная по абсолютной величине отрицательная погрешность одного цикла умножения тоже составляет около половины младшего разряда, т. е. около $-2^{-(m+r_0+1)}$.

Ясно, однако, что в первых r_0 циклах умножения в сумму частичных произведений не вносится вообще никаких погрешностей, а из последующих $m - r_0$, циклов только один может дать максимальную погрешность; погрешности всех остальных циклов непременно меньше, чем максимально возможные. Максимум суммарной погрешности в общем случае, очевидно, достигается в условиях, когда ни один из циклов не вносит максимальной для о д н о г о цикла погрешности.

Как показал В. М. Храпченко*), максимальная положительная погрешность получается в том случае, когда

$$c_{r_0+1} = c_m = 1,$$

а $c_{r_0+2}, c_{r_0+3}, \dots, c_{m-1}$ образуют последовательность из чередующихся нулей и единиц, причем c_{r_0+2} может быть как нулем, так и единицей, и при этом

*) Сб. «Проблемы кибернетики», вып. 10, 1963, стр. 165—177.

$$a_i = c_{m+r_0-i+1}, \quad i = r_0+1, \quad r_0+2,..., m,$$

где c_i — цифра i-го разряда множимого, a_i — цифра i-го разряда множителя. Иначе говоря, максимальная положительная погрешность получается при перемножении следующих чисел:

если $m - r_0$ нечетно —

$$(\ldots \underbrace{10101\ldots01}_{m-r_0 \text{ разрядов}}) \times (\ldots \underbrace{10\ldots10101}_{m-r_0 \text{ разрядов}})$$

и

$$(\ldots \underbrace{11010\ldots11}_{m-r_0 \text{ разрядов}}) \times (\ldots \underbrace{11\ldots01011}_{m-r_0 \text{ разрядов}})$$

если $m - r_0$ четно —

$$(\ldots \underbrace{10101\ldots11}_{m-r_0 \text{ разрядов}}) \times (\ldots \underbrace{11\ldots10101}_{m-r_0 \text{ разрядов}})$$

и

$$(\ldots \underbrace{11010\ldots01}_{m-r_0 \text{ разрядов}}) \times (\ldots \underbrace{10\ldots01011}_{m-r_0 \text{ разрядов}})$$

При этом величина этой погрешности равна

$$\delta^+_{\max} = \left(\frac{m-r_0}{6} + \frac{7}{18} + \frac{1}{9}\left(-\frac{1}{2}\right)^{m-r_0}\right) 2^{-(m+r_0)}.$$

Аналогичным образом, если $r_0 < m + 1$, то максимальная по абсолютной величине отрицательная погрешность получается в тех случаях, когда

$$c_{r_0+1} = 0, \qquad c_m = 1,$$

а $c_{r_0+2}, c_{r_0+3}, \ldots, c_{m-1}$ образуют последовательность из чередующихся нулей и единиц, причем c_{r_0+2} может быть как нулем, так и единицей, и при этом

$$a_i = 1 - c_{m+r_0-i+1}, \quad i = r_0+1, \quad r_0+2,\ldots,m,$$

где обозначения c_i, a_i те же, что и прежде. Иначе говоря, максимальная по абсолютной величине отрицательная погрешность получается при перемножении следующих чисел:

если $m - r_0$ нечетно —

$$(\ldots \underbrace{01010\ldots11}_{m-r_0 \text{ разрядов}}) \times (\ldots \underbrace{00\ldots10101}_{m-r_0 \text{ разрядов}})$$

и

$$(\ldots \underbrace{00101\ldots01}_{m-r_0 \text{ разрядов}}) \times (\ldots \underbrace{01\ldots01011}_{m-r_0 \text{ разрядов}}),$$

если $m - r_0$ четно —

$$(\ldots \underbrace{01010\ldots01}_{m-r_0 \text{ разрядов}}) \times (\ldots \underbrace{01\ldots10101}_{m-r_0 \text{ разрядов}})$$

и

$$(\ldots \underbrace{00101\ldots11}_{m-r_0 \text{ разрядов}}) \times (\ldots \underbrace{00\ldots01011}_{m-r_0 \text{ разрядов}})$$

При этом величина указанной погрешности равна

$$\delta^-_{\max} = -\left(\frac{m-r_0}{6} - \frac{1}{9} + \frac{1}{9}\left(-\frac{1}{2}\right)^{m-r_0}\right) 2^{-(m+r_0)}.$$

Максимальная положительная погрешность всегда на $2^{-(m+r_0+1)}$ больше по абсолютной величине максимальной отрицательной погрешности:

$$\delta^+_{\max} - \left|\delta^-_{\max}\right| = \frac{1}{2} \cdot 2^{-(m+r_0)}.$$

Если теперь потребовать, как мы делали прежде, чтобы погрешность не превышала половины младшего из полезных разрядов (т. е. половины m-го разряда), то получим условие:

если $12 \le m \le 24$, то $r_0 = 3$,
если $25 \le m \le 49$, то $r_0 = 4$,

и т. д.

В заключение $(m + r)$-разрядное или соответственно $(m + r_0)$-разрядное произведение нужно округлить до m-го разряда. Последнее выполняется по обычным правилам (см. 4.1.3).

Перед заключительным округлением можно несколько улучшить $(m + r)$-разрядное или $(m + r_0)$-разрядное произведение, вычтя из него среднее арифметическое максимальной и минимальной возможных погрешностей (максимум и минимум в алгебраическом смысле). Например, если промежуточные округления не делаются, то ошибки могут быть заключены в пределах от $-2^{-(m+r)}$ $(m - r - 1)$ до 0. Добавляя к $(m + r)$ -разрядному произведению величину

$$\frac{1}{2}\cdot 2^{-(m+r)}(m-r-1),$$

получили бы погрешность в пределах от

$$+-\frac{1}{2}\cdot 2^{-(m+r)}(m-r-1) \quad \text{до} \quad +\frac{1}{2}\cdot 2^{-(m+r)}(m-r-1).$$

Точно так же при наличии промежуточных округлений из $(m + r_0)$-разрядного произведения нужно было бы всегда вычитать величину

$$\frac{\delta^{+}_{\max} + \delta^{-}_{\max}}{2} = \frac{1}{4}\cdot 2^{-(m+r_0)}.$$

Не известно, однако, симметрично ли распределение погрешностей $(m + r)$-разрядного (или $(m + r_0)$-разрядного) произведения относительно среднего арифметического от крайних ошибок; неясно поэтому, удастся ли внесением такой поправки свести к нулю математическое ожидание ошибки.

4.2. Логические методы ускорения умножения

4.2.1.Определения.

Известные к настоящему времени методы ускорения умножения весьма разнообразны. Чтобы в них разобраться, необходимо иметь хоть какую-нибудь классификацию и хотя бы грубые критерии для сравнительной оценки.

Все методы ускорения умножения мы разделим на две большие группы: логические и аппаратные.

Логическими методами ускорения умножения мы будем называть те методы, в которых сохраняется без каких-либо изменений основная структура арифметических цепей множительного устройства, описанная в 4.1.1 (рис. 4-1), а ускорение достигается только за счет усложнения схемы управления. Применительно к параллельным устройствам признаком того, что мы имеем дело с логическим методом ускорения умножения, является независимость количества дополнительного оборудования (по сравнению со схемой рис. 4-1) от количества разрядов в сомножителях *m.* В последовательных устройствах применение логических методов ускорения умножения не приводит ни к изменению суммарного объема регистров, ни к изменению количества сумматоров (один).

Аппаратные методы ускорения умножения требуют для своего осуществления введения дополнительного оборудования в основные арифметические цепи множительного устройства.

Говоря о параллельных устройствах, мы будем различать аппаратные методы первого порядка и второго порядка. Для аппаратных методов первого порядка характерна линейная зависимость количества дополнительного оборудования от количества разрядов в сомножителях m; для осуществления методов второго порядка количество необходимого дополнительного оборудования пропорционально m^2. Применительно к последовательным устройствам такое разделение проводиться не будет.

4.2.2. Логические методы ускорения в двоичной системе.

1°. Простейшим из логических методов ускорения умножения является *пропуск тактов суммирования* в тех случаях, когда очередная цифра множителя есть нуль. Схему управления вообще проще построить так, чтобы вслед за тактом сдвига каждый раз отводилось время на суммирование, но суммирование выполнялось бы в зависимости от цифры множителя. Небольшое усложнение схемы управления, позволяющее отводить время на суммирование только тогда, когда оно действительно необходимо, сокращает число тактов суммирования в среднем вдвое. На каждый разряд множителя при выполнении умножения будет приходиться в среднем 0,5 такта суммирования. Этот метод ускорения в равной мере подходит для тех случаев, когда умножение начинается от старших разрядов множителя, и для случаев, когда умножение начинается от младших разрядов. Более сложные логические методы ускорения нам придется рассматривать большей частью отдельно для разных случаев.

2°. Рассмотрим теперь метод *группировки разрядов* множителя. Начнем со случая, когда умножение начинается от младших разрядов, т. е. выполняется по первому или по второму варианту (4.1.2, п.п. 1°, 2°).

Пусть в схему управления поступает не одна очередная цифра, а сразу пара разрядов множителя. Воспользовавшись наличием цепей для обращения кодов в регистрах *B* и *C,* составим такую программу суммирований и вычитаний на ближайшие два цикла умножения, чтобы количество тактов суммирования-вычитания было минимальным.

Рассмотрим сначала младшую пару разрядов множителя. Если она содержит комбинацию 00, то это значит, что в течение двух ближайших циклов умножения не нужно будет выполнять ни сложений, ни вычитаний; при

наличии комбинации 01 нужно будет выполнить одно суммирование в первом из двух ближайших циклов умножения, а при комбинации 10 — во втором. При всех этих комбинациях количество тактов суммирования получается таким же, каким оно было бы при отсутствии группировки—если бы каждый разряд множителя рассматривался отдельно и просто пропускались бы такты суммирования в тех случаях, когда очередная цифра множителя есть нуль.

Выигрыш получается только за счет комбинации 11. Нормально здесь требовалось бы суммирование в обоих ближайших циклах умножения. Мы же вместо этого произведем одно вычитание в первом цикле и запомним единицу, которую нужно будет добавить к следующей паре разрядов множителя. Иными словами, комбинацию ..11 (две точки означают границу пары разрядов) мы представляем в виде

$$..11 = 1..00\text{—}..01.$$

С учетом единицы, переданной из предыдущей пары разрядов, в следующей паре разрядов может встретиться одна из пяти комбинаций:

00 — если данная пара разрядов содержит цифры 00 и из предыдущей пары единица не передается;

01 — если данная пара содержит цифры 01 и из предыдущей пары единица не передается или если данная пара разрядов содержит 00 и передается единица из предыдущей пары разрядов;

10 — если данная пара содержит 10 и нет единицы из предыдущей пары или если данная пара содержит 01 и есть единица из предыдущей пары разрядов;

11 — если данная пара разрядов содержит 11 и нет единицы из предыдущей пары или если данная пара содержит 10 и есть единица из предыдущей пары;

1..00 — если данная пара разрядов содержит цифры 11 и единица передается из предыдущей пары разрядов.

В первых четырех случаях (комбинации 00, 01, 10, 11) поступим аналогично предыдущему; в последнем случае (комбинация 1..00) в течение ближайших двух циклов умножения не будем выполнять ни суммирований, ни вычитаний, но запомним единицу, передаваемую в следующую пару разрядов.

Очевидно, что и при расшифровке всех последующих пар разрядов множителя положение будет аналогичным тому, которое имелось при расшифровке второй пары.

Операции, которые мы здесь выполняем, заключают в себе преобразование множителя в такую форму, когда наряду с цифрами 0, 1 допускается также цифра —1 (ее мы будем обозначать $\overline{1}$). Цифра 0 в множителе соответствует отсутствию суммирований или вычитаний, цифра 1 — выполнению суммирования, цифра $\overline{1}$ — выполнению вычитания. О некоторых формальных соображениях по этому поводу сказано в конце настоящего раздела.

Изображения чисел в этой форме неоднозначны. Фактически все логические методы ускорения двоичного умножения, которые мы будем рассматривать дальше, сводятся к различным способам преобразования множителя в указанную форму. Тот или иной способ тем более эффективен, чем больше в преобразованном множителе среднее количество нулей и меньше среднее количество положительных и отрицательных единиц, т. е. чем меньше в среднем требуется тактов суммирования или вычитания; важно также, каково максимальное количество тактов суммирования вычитания, которое может встретиться при выполнении умножения.

Для оценки эффективности описанного выше метода группировки разрядов по парам заметим прежде всего, что при любой комбинации цифр слагаемых в течение пары циклов умножения может быть не более одного суммирования или вычитания. Таким образом, теперь в худшем случае приходится половина такта суммирования или вычитания на каждый разряд множителя — вдвое меньше, чем при отсутствии группировки пар разрядов (теперь «худшими» являются случаи умножения на 01...010101 или на 10...101010, в то время как при отсутствии группировки разрядов по парам «худшим» был случай умножения на 11...111111).

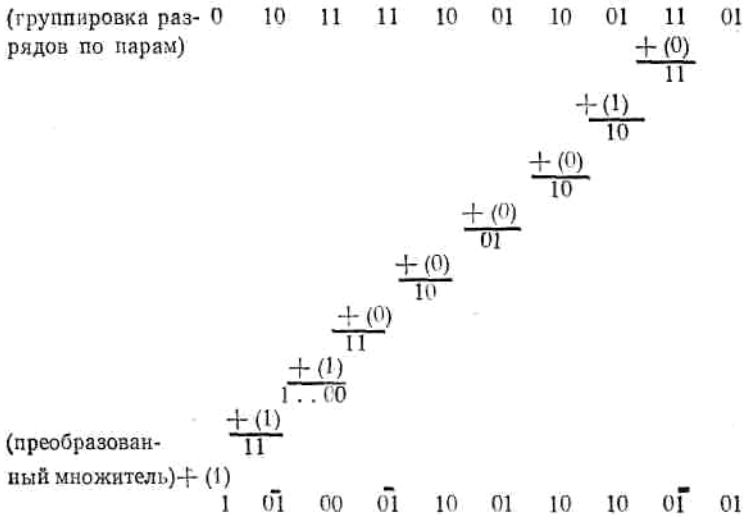
Что касается среднего количества тактов суммирований или вычитаний, то для его подсчета нужно обратить внимание на то, что для первой пары разрядов комбинации 00,01, 10 и 11 равновероятны (вероятность появления любой из них равна $^1/_4$). Для любой последующей пары разрядов вероятность появления каждой из комбинаций 01,10 и 11 равна $^1/_4$, для комбинаций же 00 и 1..00 их вероятности в сумме равны $^1/_4$. (Если вероятность передачи единицы в данную пару разрядов обозначить через p, то вероятность комбинации 00 равна $(1-p)\cdot\frac{1}{4}$, а вероятность комбинации $1..00 - p\cdot\frac{1}{4}$, что в сумме дает $^1/_4$; вероятность появления любой другой из комбинаций — скажем, комбинации 01 — равна $(1-p)\cdot\frac{1}{4} + p\cdot\frac{1}{4} = \frac{1}{4}.$)

Поскольку в комбинациях 01, 10 и 11 требуется по одному такту суммирования или вычитания в течение ближайшей пары циклов умножения, а в комбинациях 00 и 1..00— ни одного такта, то в среднем на пару циклов умножения будет приходиться по $^3/_4$ такта суммирования-вычитания. На каждый разряд множителя будет приходиться, таким образом, в среднем $^3/_8$ (или 0,375) такта суммирования-вычитания вместо 0,5 такта, которые мы имели без группировки.

Пример. Пусть множитель первоначально имеет вид

$$0.101111100110011101.$$

Действуя описанным методом, получим (в круглых скобках указаны цифры, переносимые из одной пары разрядов в другую)



Вместо 12 единиц в исходном представлении множителя получаем 9 положительных и отрицательных единиц в преобразованном множителе.

Для дальнейшего важно еще и следующее обстоятельство. Комбинацию цифр множителя 10 мы расшифровали так, что суммирование производилось во втором из двух ближайших циклов умножения, а в следующую пару разрядов единица не передавалась. Вместо этого/можно было бы во втором из двух ближайших циклов умножения выполнить вычитание и передать единицу в следующую пару разрядов, т. е. представить комбинацию 10 в виде $..10 = 1..\bar{1}0$. Пока в схему управления поступают только пары разрядов множителя, такое изменение никакой роли не играет. Но если бы при расшифровке данной пары разрядов было сразу известно, какова комбинация цифр в следующей паре разрядов, для комбинации «10» можно было бы подбирать каждый раз свой, наиболее выгодный способ расшифровки. Например, если в следующей паре разрядов имеется комбинация «00», то из данной пары разрядов в нее лучше не переносить единицу, если же в ней имеется комбинация «11», то лучше перенести.

Это дает основание надеяться, что при увеличении количества разрядов, расшифровываемых единовременно, можно получить дальнейшее сокращение среднего количества тактов суммирования-вычитания.

Однако непосредственным подсчетом (аналогичным проведенному выше для пар разрядов) нетрудно убедиться, что при группировке разрядов множителя в тройки получили бы в среднем тоже 0,375 такта суммирования-вычитания на 1 разряд множителя, при группировке по четверкам— 0,359 такта, при группировке по пятеркам — 0,356 такта и т. д. С увеличением количества разрядов в группе среднее количество тактов суммирования-вычитания, приходящееся на 1 разряд множителя, убывает очень медленно; в 4.2.4 мы покажем, что оно в пределе стремится к 0,333... Однако тот же результат может быть достигнут более простым путем — методом последовательного преобразования множителя, о котором говорится ниже. Группировать разряды более чем по два не имеет, вероятно, смысла.

Теперь рассмотрим применение метода группировки разрядов множителя в тех случаях, когда умножение начинается от его старших разрядов. Вернувшись несколько назад, разрешим при группировке разрядов по парам возможность двоякого представления комбинации 10 (в виде ..10 или $1..\bar{1}0$). Тогда правило расшифровки пар разрядов, которым мы пользовались выше, для умножения от младших разрядов, можно сформулировать следующим образом:[*])

— если старшая цифра данной пары разрядов множителя есть единица, то вычесть из имеющейся комбинации величину 1..00;

— если старшая цифра соседней справа пары разрядов есть единица, то добавить к имеющейся в данной паре разрядов комбинации величину ..01.

Тождественность предлагаемых преобразований множителя очевидна: величина 1..00 для соседней справа пары разрядов равна величине ..01 для данной пары. В то же время, как нетрудно проверить простым перебором всех комбинаций, в результате указанного преобразования в любой паре циклов умножения может быть не более одного такта сложения или вычитания, а в среднем на каждый разряд множителя придется 0,375 тактов сложения-вычитания. В схеме управления, конечно, добавление величины ..01 или вычитание 1 ..00 фактически не производится: это преобразование выполняется дешифратором, в который должны поступать теперь не 2 разряда множителя, как прежде, а три: кроме данной пары разрядов, необходима также цифра старшего разряда

[*]) Заимствовано из работы Кухарчука А. Т. и Проценко Н. М., «Методы ускоренного умножения со старших разрядов множителя» (сб. «Труды конференции «Новые разработки по вычислительной математике и вычислительной технике»», Киев, 1960), где приводимая формулировка предлагается для умножения от старших разрядов.

следующей пары. Но это ведет не к усложнению, а, возможно, даже к некоторому упрощению схемы управления, так как отпадает необходимость в формировании, запоминании и расшифровке цифры, переносимой из предыдущей пары разрядов.

Однако наиболее важное преимущество, которое достигается при таком истолковании правила расшифровки пар разрядов, состоит в том, что способ этот с равным успехом можно применять и при умножении, начинающемся от младших разрядов множителя, и при умножении от старших разрядов.

Разбивка разрядов по парам в любом случае должна быть такой, чтобы в старшую пару входили несуществующий разряд целых и первый (старший) из значащие разрядов числа. Поскольку разряд целых содержит всегда цифру «0», невозможна такая комбинация, при которой из старшей пары разрядов вычиталась бы величина 1..00 (допускать этого нельзя, так как вычитание из старшей пары разрядов величины 1..00 не могло бы быть скомпенсировано добавлением величины .. 01 к разрядам, находящимся слева от нее,— умножение на эти разряды уже не выполняется).

Может оказаться, что при преобразовании старшей пары разрядов появляется единица в ее левом разряде, т. е. в разряде целых. Применительно, скажем, к первому варианту выполнения умножения (см. 4.1.2) это указывает на необходимость добавить множимое к сумме частичных произведений после последнего сдвига вправо суммы частичных произведений; применительно, например, к четвертому варианту выполнения умножения такая единица указывает на необходимость выполнения суммирования до первого сдвига множимого вправо.

П р и м е р . Пусть, как и в примере на стр. 152, множитель первоначально имеет вид

(0). 101111100110011101 (0).

Действуя описанным способом, получим:

(группировка разрядов по парам)

01 01 11 11 00 11 00 11 10 10

(преобразование пар разрядов в любом порядке)

+0 +1 +1 +0 +1 +0 +1 +1 +1

—0 — 0 —1..00 — 1..00 —0 —1..00 —0 — 1..00 —1..00—1..00

(преобразованный множитель)

0.1 10 00 $0\bar{1}$ 01 $0\bar{1}$ 01 00 $0\bar{1}$ $\bar{1}0$

Здесь мы получили 8 положительных и отрицательных единиц в преобразованном множителе вместо 12 единиц в исходном и 9 единиц в примере на стр. 153. Однако последнее различие носит случайный характер: при других комбинациях цифр способ расшифровки пар, которым мы пользовались на стр. 153, мог бы дать лучший результат, чем данный способ, а в среднем они одинаковы.

Группировать разряды более чем по два при умножении от старших разрядов также не имеет смысла, как при умножении от младших разрядов.

3°. Наиболее эффективным из логических методов ускорения является *последовательное преобразование цифр множителя*[*]).

Идея метода состоит в том, что в к а ж д о м ц и к л е умножения осматриваются, кроме очередного разряда множителя, также его соседние разряды, и в зависимости от имеющихся в них цифр выбирается та или иная цифра для одного данного разряда преобразованного множителя. Если при группировке разрядов, например, по парам мы производили преобразование сразу двух разрядов множителя и таким образом вырабатывали программу суммирований-вычитаний на два цикла умножения вперед, то здесь схема преобразования множителя (которую нужно иметь в составе устройства управления) должна будет работать в каждом цикле умножения (а не один раз за 2 цикла), и каждый раз будет вырабатываться программа суммирований или вычитаний только для одного ближайшего цикла умножения.

Правило преобразования множителя сформулируем одинаково как для умножения от младших разрядов, так и для умножения от старших разрядов:

— если данная цифра н е п р е о б р а з о в а н н о г о множителя не совпадает с соседней справа его цифрой, соседняя слева цифра есть 0 и предыдущая цифра п р е о б р а з о в а н н о г о множителя есть 0, то данный разряд преобразованного множителя должен содержать +1;

— если данная цифра н е п р е о б р а з о в а н н о г о множителя не совпадает с соседней справа его цифрой, соседняя слева цифра есть 1 и предыдущая цифра п р е о б р а з о в а н н о г о множителя есть 0, то данный разряд преобразованного множителя должен содержать —1;

— если данная цифра н е п р е о б р а з о в а н н о г о множителя совпадает с соседней справа его цифрой или

[*]) В литературе этот метод известен также под названием *усовершенствованного метода Booth и Booth.* Впервые он опубликован в работе L e h m a n M., Short-cut Multiplication and Division in Automatic Digital Computers, sept. 1957, v. EC-6, No 3, справедливее было бы, вероятно, называть его методом Lehman. В настоящей книге метод обобщается на случай умножения от старших разрядов. В несколько иной формулировке этот метод имеется также в работе А. Т. Кухарчука и Н. М. Проценко (см. сноску на стр. 153).

если предыдущая цифра преобразованного множителя не является нулем, то данный разряд в преобразованном множителе есть 0.

При применении этого правила необходимо учитывать, что старшая значащая цифра преобразованного множителя может находиться в разряде целых — там, где непреобразованный множитель содержит всегда нуль; справа и слева от значащих разрядов преобразованного множителя всегда предполагаются нули. Когда в приведенном правиле говорится о «предыдущей» цифре преобразованного множителя, то применительно к умножению от младших разрядов это относится к предыдущей младшей цифре преобразованного множителя, а применительно к умножению от старших разрядов — к предыдущей старшей цифре.

В справедливости указанного правила легко убедиться непосредственным перебором различных вариантов образования цифр преобразованного множителя.

Действительно. Предположим, что умножение ведется от младших разрядов. Справа от младшего разряда множителя всегда предполагается нуль. Если далее младшие разряды множителя тоже содержат серию нулей, то в соответствующих разрядах преобразованного множителя будут получаться нули:

исходный множитель ... 00 ... 0 [0]
преобразованный множитель ... 00 ... 0 ,

потому что каждый раз цифра данного разряда совпадает с цифрой соседнего справа разряда.

Первая ненулевая цифра в преобразованном множителе появится в том разряде, где в исходном множителе имеется первая единица (в частности, это может быть и самый младший разряд). При этом возможны 2 случая: либо за этой первой единицей снова пойдут нули (или хотя бы один нуль), либо эта единица является началом серии единиц. В первом случае по нашему правилу получим:

исходный множитель ... 0100 ...
преобразованный множитель ... 0100 ...

Во втором случае:

исходный множитель ... 1100 ...
преобразованный множитель ... 0100 ...

Первый случай тривиален: преобразованный множитель пока не отличается от исходного; дальнейшие преобразования пойдут так же, как и в начале. Во втором случае необходимо посмотреть, что получится в преобразованном множителе, когда серия единиц исходного множителя прервется нулями. Здесь тоже могут быть два случая: либо серия единиц прерывается одиночным нулем, за которым снова идут единицы (или хотя бы одна единица), либо серия единиц прерывается серией нулей (не меньше двух). Оба эти случая встречаются в следующей паре строчек:

исходный множитель ... 0011 ... 11011 ... 110...
преобразованный множитель ... 0100 ... 00100 ... 010 ...

Из их рассмотрения видно, что после того, как в исходном множителе появляется хотя бы пара нулей подряд, восстанавливается равенство между величиной преобразованного и исходного множителя, и дальнейшее преобразование пойдет опять так же, как в начале операции.

Может оказаться, что серия единиц в исходном множителе продолжается до его старшего разряда. Однако вслед за этим в нем всегда предполагается пара нулей: в разряде целых и в несуществующем разряде слева от него. Поэтому в окончательном виде величина преобразованного множителя всегда равна величине исходного множителя.

Читателю предлагается самостоятельно убедиться, что аналогичный результат получается и при умножении от старших разрядов.

Возвращаясь к примеру, рассматривавшемуся на стр. 152, 153 и 154, найдем, что преобразование по указанному правилу исходного множителя

$$[0]\,[0].\ 101111100110011101\quad [0]$$

дало бы

$$[0]\cdot 1\,.\,0\,\bar{1}\,0000\,\bar{1}\,010\,\bar{1}\,01001\,\bar{0}\,1\cdot[0],$$

независимо от того, откуда начинается преобразование — от старших или от младших разрядов.

Количество положительных и отрицательных единиц в преобразованном множителе здесь получилось равным восьми — таким же, как в примере на стр. 154. Однако в среднем этот метод, как мы сейчас увидим, более эффективен.

Прежде всего заметим, что сама формулировка правила преобразования такова, что из двух соседних цифр преобразованного множителя по меньшей мере одна должна быть нулем. Поэтому максимальное количество тактов суммирования-вычитания не превышает половины такта на один разряд множителя.

Для подсчета среднего количества тактов суммирования-вычитания обозначим через p_i вероятность того, что i-й разряд преобразованного множителя содержит цифру +1 или —1; нумерация разрядов здесь начинается от младшего или от старшего из значащих разрядов — в зависимости от того, от какого разряда начинается преобразование множителя, т. е. от какого разряда начинается процесс выполнения умножения. Очевидно, что $p_i = {}^1\!/_2$, так как предыдущая цифра преобразованного множителя не существует (всегда «0»), а вероятность

несовпадения двух соседних цифр непреобразованного множителя равна $^1/_2$. Поскольку вероятность того, что (i — 1)-я цифра преобразованного множителя есть 0, равна 1— p_{i-1}, то

$$p_i = \frac{1}{2}(1 - p_{i-1}).$$

В частности, $p_2 = {}^1/_4$, $p_3 = {}^3/_8$, $p_4 = {}^5/_{16}$, … В пределе

$$\lim_{i\to\infty} p_i = \frac{1}{2}(1 - \lim_{i\to\infty} p_i),$$

откуда

$$\lim_{i\to\infty} p_i = {}^1/_3.$$

Таким образом, при достаточно большом количестве разрядов в множителе среднее количество суммирований, или вычитаний, приходящееся на один разряд множителя, равно примерно $^1/_3$ =0,333... Как мы увидим в 4.2.4, это — *наилучший результат, который может быть получен при использовании логических методов ускорения умножения.*

В осуществлении метод последовательных преобразований ненамного сложнее, чем метод группировки разрядов множителя, эффективность же его выше. Однако метод группировки разрядов, рассмотренный в п. 2°, представляет определенный интерес как с теоретической точки зрения (см. 4.2.4), так и в качестве основы для некоторых аппаратных методов ускорения умножения (см. 4.3).

Как мы уже говорили, все логические методы ускорения умножения в двоичной системе фактически сводятся к преобразованию множителя в такую форму, когда для каждого разряда допустимыми цифрами являются —1, 0 и +1. Если следовать формально первому определению основания системы счисления, приведенному в 1.2.1 (стр. 8), то мы должны были бы утверждать, что множитель преобразуется из двоичной системы счисления в троичную, поскольку для каждого разряда теперь допустимы не два, а три различных символа. Здесь мы впервые сталкиваемся с ситуацией, когда общепринятое определение основания системы счисления оказывается неудобным и когда лучше пользоваться вторым определением раздела 1.2.1 (стр. 9). С точки зрения второго определения, поскольку цифры —1, 0, +1 неравновероятны и поскольку общее количество информации, содержащееся в m разрядах преобразованного множителя, такое же, как и в m двоичных разрядах непреобразованного множителя, форма представления множителя остается по-прежнему двоичной.

Аналогичное положение возникает также при использовании логических методов ускорения умножения в системе счисления с основанием $n > 2$ (раздел 4.2.3), некоторых аппаратных методов ускорения умножения (раздел 4.3), использовании избыточных цифр частного для ускорения деления (раздел 5.1.3) и т. д.

4.2.3. О логических методах ускорения умножения в системе счисления с основанием n ($n > 2$).

В системе счисления **с** произвольным основанием можно, как и в двоичной системе, отводить для умножения на каждый из разрядов множителя равно столько тактов суммирования, сколько единиц содержится в цифре данного разряда; при этом среднее количество суммирований было бы примерно вдвое меньше, чем максимально возможное.

Однако при $n > 2$ имеется весьма простой метод, для которого нет аналога в чисто двоичной системе, позволяющий сократить среднее количество суммирований еще примерно вдвое. Им является метод *перехода к симметричному диапазону цифр.*

Суть этого метода состоит в том, что в ходе умножения множитель преобразуется из обычной формы записи, в которой применяются цифры

$$0, 1, 2, \ldots, (n-1),$$

в такую форму, где используются:

для нечетного n цифры

$$-\frac{n-1}{2},\ -\frac{n-3}{2},\ldots,-1,0,+1,\ldots,\frac{n-3}{2},\frac{n-1}{2},$$

для четного n цифры

$$-\frac{n-2}{2},-\frac{n-4}{2},\ldots,-1,0,+1,\ldots,\frac{n}{2}$$

или

$$-\frac{n}{2},-\frac{n-2}{2},-\frac{n-4}{2},\ldots,-1,0,\ldots+1,\ldots,\frac{n-2}{2}.$$

Для четного n иногда допускают и цифру — $^n/_2$ и цифру $^n/_2$; если это так, то преобразование множителя становится неоднозначным.

Допустив указанную возможность, правило получения преобразованной цифры множителя для некоторого разряда при четном n можно сформулировать следующим образом:

(а) если данная цифра множителя больше или равна $^n/_2$, то вычесть из нее величину n;

(б) если соседняя справа цифра множителя больше или равна $^{n}/_{2}$, то добавить единицу к величине, полученной в результате (а).

В такой формулировке правило это одинаково применимо и в случае, когда умножение начинается от старших разрядов, и в случае, когда умножение начинается от младших разрядов. Нужно только учесть, что для преобразованного множителя значащим является также и первый разряд в целой части числа: в нем может оказаться цифра 1; в непреобразованном множителе этот разряд не существует (его цифра всегда 0).

П р и м е р . Множитель, записанный по обычной десятичной системе в виде

$$[0]\,.\,8470251964,$$

в соответствии с приведенным правилом преобразуется в форму

$$1.\,\bar{2}\,5\,\bar{3}\,0\,3\,\bar{5}\,2\,0\,\bar{4}\,4.$$

При умножении на непреобразованный множитель потребовалось бы

$$8+4+7+0+2+5+1+9+6+4=46$$

тактов суммирования; при умножении на преобразованный множитель требуется

$$1+\left|\bar{2}\right|+5+\left|\bar{3}\right|+0+3+\left|\bar{5}\right|+2+0+\left|\bar{4}\right|+4=29$$

тактов суммирования-вычитания.

Тождественность выполняемых здесь преобразований очевидна: добавление единицы к цифре некоторого разряда компенсируется вычитанием величины n из соседнего справа разряда. По сути говоря, это правило преобразования n-ичного множителя полностью аналогично правилу, сформулированному в 4.2.2 на стр. 153: группируя двоичные разряды по парам, мы фактически переходим от двоичной системы к двоично-кодированной четверичной системе счисления; затем каждая пара двоичных разрядов, т. е. каждый четверичный разряд, преобразовывается точно так же, как указано здесь для любого четного n.

Если при четном n нежелательно допускать и цифру $-\,^{n}/_{2}$, и цифру $+\,^{n}/_{2}$ или если n нечетно, то преобразование множителя удобнее выполнять, начиная от младших разрядов. При этом правило преобразования формулируется следующим образом:

— если цифра данного разряда больше $^{n}/_{2}$, то вычесть из нее n и запомнить единицу, передаваемую в следующий (старший) разряд;

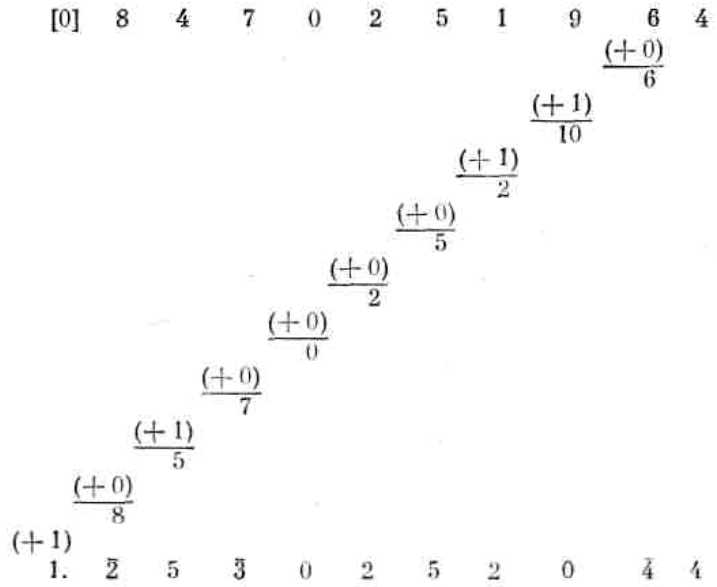
— если из предыдущего (младшего) разряда в данный разряд передана единица, то перед сравнением цифры данного разряда с $^{n}/_{2}$ и, может быть, вычитанием n добавить к ней единицу.

Таким правилом преобразования мы тоже пользовались в начале 4.2.2 (стр. 152, 153) для четверичной системы, которая получалась при группировке по парам разрядов двоичного множителя.

Вернувшись к предыдущему примеру, десятичный множитель

$$[0]\,.\,8\;4\;7\;0\;2\;5\;1\;9\;6\;4$$

преобразуем по указанному правилу следующим образом (где в круглых скобках указаны цифры, передаваемые из данного десятичного разряда в следующий старший разряд):



Количество необходимых тактов сложения-вычитания теперь оказывается равным

$$1+\left|\bar{2}\right|+5+\left|\bar{3}\right|+0+2+5+2+0+\left|\bar{4}\right|+4=28$$

(на один меньше, чем при применении предыдущего правила преобразования). Это, однако, случайное расхождение: при других комбинациях цифр мог бы получиться и обратный результат, а в среднем оба правила

преобразования равноценны.

Нетрудно подсчитать, что для четного n среднее количество тактов суммирований-вычитаний, приходящееся на один разряд множителя, оказывается равным $n/4$, для нечетного — величине

$$\frac{n-1/n}{4}$$

Сразу ясно, почему описанный метод не может ничего дать для чисто двоичной системы: в двоичной системе нет таких цифр, которые были бы больше $n/2$ (максимальная цифра — единица — в точности равна $n/2$); между тем именно на преобразовании таких цифр мы и получали экономию в количестве тактов суммирования-вычитания. Подходя к делу формально, можно сказать, что среднее количество тактов суммирования-вычитания $n/4$, которое получается в этом методе для любого четного n, при n=2 оказывается равным $1/2$— таким же, как в случае, когда просто для умножения на каждый разряд отводится столько тактов суммирования, сколько единиц содержится в очередной цифре множителя (0 или 1).

Максимальное количество тактов суммирования-вычитания, приходящееся на один разряд множителя, для четного n равно $n/2$, для нечетного — величине $\frac{n-1}{2}$ вместо n — 1 для случая, когда симметрирование диапазона цифр не применяется. (При n =2 снова получаем, что метод симметрирования диапазона цифр ничего не дает, так как $n/2 = n$ — 1.)

Очень существенным для дальнейшего является одно обстоятельство, о котором мы вскользь уже упоминали. Обратим еще раз внимание на ту принципиальную разницу, которая имеется между четным и нечетным основанием системы счисления n.

Если n четно, то среди всех возможных цифр имеется одна, а именно $n/2$, с которой при преобразовании можно обращаться—двояко: представляя ее либо в виде $(0)\frac{n}{2}$, либо в виде $(1)\frac{\overline{n}}{2}$; при этом количество тактов суммирования или вычитания при умножении на данный разряд множителя получается одинаковым. Это позволило нам на стр. 156 сформулировать такое правило преобразования множителя для четного n, которое удобно было применять и при умножении от старших разрядов, и при умножении от младших разрядов. В соответствии с указанным правилом цифра в некотором разряде преобразованного множителя зависела только от цифры данного разряда и от одной соседней справа цифры непреобразованного множителя и не зависела от всех других цифр. Если, в частности, i-й разряд (нумерация разрядов — слева направо) содержал до преобразования цифру $n/2$, то к $(i$ — 1)-му разряду при его преобразовании добавляется единица, а в i-м разряде получится (в зависимости от $(i$ +1)-го разряда) либо — $n/2$, либо $-\frac{n-2}{2}$; если же i-й разряд до преобразования содержал цифру $n-2/2$, то к $(i$ — 1)-му разряду единица не добавляется, а в i-м разряде может получиться (в зависимости от $(i + 1)$-го разряда) либо $n/2$, либо $n-2/2$.

С другой стороны, это позволяет надеяться на получение дальнейшей экономии в количестве суммирований-вычитаний при использовании более сложных логических методов ускорения умножения.

Пусть, например, умножение начинается от младших разрядов, и пусть в некотором разряде с учетом единицы, переданной, возможно, из младшего разряда, получилась цифра $n/2$; ее выгодно оставить без изменения, если цифра следующего (старшего) разряда меньше $n/2$, и преобразовать к виду $(1)\frac{\overline{n}}{2}$, если цифра следующего разряда больше $n/2$; умножение на данный разряд в любом случае потребует $n/2$ тактов суммирования-вычитания, зато при умножении на следующий разряд в половине случаев один такт вычитания выигрывается (когда исходная цифра этого разряда больше, чем $n/2$). Используя эту идею, для системы счисления с четным основанием n можно построить методы ускорения, аналогичные методам группировки разрядов и методу последовательного преобразования множителя, изложенным в 4.2.2 для двоичной системы.

При нечетном n нет цифр, допускающих двоякую расшифровку, и указанные возможности отсутствуют.

Впрочем, и для четного n вряд ли имеет смысл применять какие-либо сложные логические методы ускорения умножения — вроде группировки разрядов или последовательного преобразования множителя. Не рассматривая эти методы по существу, мы оценим их возможности в 4.2.4. Однако заранее ясно, что чем выше n, тем меньше могут дать эти методы: если при n =2 вероятность цифры «1», допускающей двоякую расшифровку, равна $1/2$ то, скажем, для десятичной системы (n = 10) вероятность цифры «5», допускающей двоякую расшифровку, равна всего $1/10$.

4.2.4. Предельные возможности логических методов ускорения умножения.

По определению (см. 4.2.1) применение логических методов ускорения умножения не затрагивает основной структуры арифметических цепей, описанных в 4.1.2. Следовательно, каковы бы ни были эти методы, умножение все равно должно состоять из элементарных операций суммирования, вычитания (или суммирования с

дополнительным или обратным кодом) и сдвига на 1 разряд; при этом, поскольку в схеме устройства имеется один сумматор, одновременно может производиться только одно суммирование или вычитание. Таким образом, логические методы ускорения умножения в любом случае сводятся к преобразованиям множителя в такую форму, чтобы количество суммирований или вычитаний, необходимых при умножении на любое число, было возможно меньшим. Точнее говоря, должно быть сведено к минимуму математическое ожидание количества суммирований-вычитаний, необходимых при выполнении одного умножения, в предположении, что все возможные числа равновероятны. Поставим теперь задачу следующим образом.

Пусть каким-либо образом любое m-разрядное n-ичное число α преобразуется к виду

$$\alpha = \sum_{i=0}^{m} a_i n^{-i},$$

где количество разрядов m обратно пропорционально $\log n$ (так что при разных основаниях системы счисления сохраняется одинаковая точность вычислений), a_i — произвольные целые числа, положительные или отрицательные. Пусть известно, что такое преобразование выполнено наилучшим из возможных способов — в том смысле, что сумма $\sum_{i=0}^{m} |a_i|$ минимальна, т. е. что минимальным является количество тактов суммирования или вычитания, необходимых при умножении на α.

Определим:

1) каково математическое ожидание величины $\sum_{i=0}^{m} |a_i|$ для всевозможных α при данных m и n, в предположении, что все значения α равновероятны;
2) как зависит математическое ожидание величины $\sum_{i=0}^{m} |a_i|$ от основания системы счисления n (при указанном выше условии для m) и при каком основании системы счисления $n_{\text{опт}}$ оно минимально;
3) каково математическое ожидание величины $\sum_{i=0}^{m} |a_i|$ при использовании системы счисления с основанием $n_{\text{опт}}$.

Решение первого вопроса даст нам ответ о предельном ускорении умножения, которое возможно при использовании логических методов, при условии, что основание системы счисления n задано заранее. Второй вопрос фактически должен расширить наши представления о влиянии величины n на скорость выполнения операций (см. 1.2.3); величина $n_{\text{опт}}$, которую мы хотим определить,— это то оптимальное основание системы счисления, использование которого позволяет логическими методами достигнуть максимальной скорости выполнения умножения с заданной точностью. Наконец, ответ на третий вопрос должен дать представление об абсолютном пределе скорости выполнения умножения, которая может быть получена применением логических методов ускорения — когда и основание системы счисления выбрано оптимальным именно с этой точки зрения и примененный метод ускорения является наиболее сильным из возможных логических методов для данной системы счисления[*]).

Как говорилось в 4.2.3, при нечетном основании системы счисления n преобразование множителя к симметричному диапазону цифр выполняется однозначно. Минимальное количество суммирований-вычитаний, приходящееся на один разряд множителя, оказывается равным $\frac{1}{4}(n - 1/n)$ и никаких возможностей для дальнейшего сокращения умножения нечетное n не предоставляет.

Займемся поэтому подробнее случаем, когда основание системы счисления n четно.

В качестве отправной точки предположим, что умножение начинается от младших разрядов множителя и для его ускорения применяется метод группировки разрядов. Дальнейший план наших действий будет таков.

Для случая, когда в каждую группу входит по одному разряду (т. е. фактически когда никакой группировки нет), минимальное количество тактов суммирования-вычитания, необходимое в среднем для умножения на такую «группу» разрядов, известно нам из предыдущего. Оно равно

$$A_1 = n/4.$$

Непосредственным перебором всех комбинаций цифр найдем также, каково в среднем минимальное количество суммирований-вычитаний при умножении на пару разрядов, если разряды множителя группируются по два. Эту

[*]) Для частного случая $n = 2$ эта задача была поставлена и решена автором настоящей книги в 1957 г. (см. диссертацию «Основы построения арифметических устройств и инженерное решение арифметического узла вычислительной машины М-2», 1957, а также книгу «Арифметические устройства электронных цифровых машин», Физматгиз, 1958, стр. 116—122). Судя по ссылкам, имеющимся в литературе, в то же примерно время такой же результат был получен д-ром Reitwissner (США). Постановка и решение полной задачи имеются в работе автора «Логические методы ускорения умножения в цифровых вычислительных машинах» (сб. «Проблемы кибернетики», вып. 4, 1960, стр. 111—120).

величину обозначим A_2.

Далее предположим, что всевозможные группы из (p — 1) разрядов множителя преобразованы оптимальным способом и что нам известно, каково в среднем минимальное количество суммирований-вычитаний A_{p-1}, необходимое при умножении на группу из (p — 1) разрядов; найдем, на сколько увеличится это количество, если разряды будут теперь группироваться не по (p — 1), а по p, т. е. найдем разность $A_p - A_{p-1}$. По этим данным вычислим A_q — среднее значение минимального количества суммирований-вычитаний, приходящееся на группу из q разрядов множителя при группировке разрядов по q, где q — произвольное целое число, $q > 1$; вычислим также величину A_q/q, т. е. среднее значение минимального количества суммирований-вычитаний, приходящееся на 1 разряд множителя, при условии, что разряды множителя собираются в группы по q, и каждая такая группа преобразуется оптимальным образом.

Далее можно будет найти и $\lim\limits_{q\to\infty} \frac{A_q}{q}$. Поскольку, как мы увидим, величина A_q/q весьма близка к пределу уже при сравнительно небольших значениях q, можно полагать, что для m-разрядного множителя (m обычно достаточно велико) минимальное количество суммирований-вычитаний при оптимальном преобразовании множителя (если его рассматривать как одну «группу» из m разрядов) равно $m \lim\limits_{q\to\infty} \frac{A_q}{q}$. Этот результат будет иметь уже более общее значение, так как при этом неважно, каким в точности способом было получено оптимальное представление множителя — группировкой разрядов или как-нибудь иначе; не зависит он и от того, от каких разрядов множителя начинается умножение — от младших или от старших.

Приступим к осуществлению этого плана.

Итак, нам известно, что $A_1 = {}^n/_4$.

Пусть теперь разряды группируются по два ($q = 2$).

Пока в каждую «группу» входил один разряд множителя, при наличии в нем цифры ${}^n/_2$ нам было безразлично, как вести ближайший цикл умножения: добавлять ли множимое ${}^n/_2$ раз, ничего не передавая в следующий разряд множителя, или вычитать множимое ${}^n/_2$ раз и передавать единицу в следующий разряд. Если теперь разряды множителя группируются по два, то в тех комбинациях, когда правая цифра равна ${}^n/_2$, а левая меньше ${}^n/_2$, выгодно в первом цикле производить ${}^n/_2$ суммирований, когда же левая цифра равна или больше ${}^n/_2$ — то ${}^n/_2$ вычитаний. В первом случае в левый разряд ничего не передается, и количество суммирований-вычитаний такое же, как при преобразовании каждого разряда в отдельности; во втором случае увеличение на единицу левой цифры пары разрядов приводит к уменьшению на единицу количества вычитаний при выполнении второго из пары циклов умножения.

Например, в десятичной системе при преобразовании каждого разряда в отдельности комбинация ..4..5 сохраняется без изменений, а комбинация ..6..5 представляется в виде (1)..$\overline{4}$..5 (две точки означают границы групп); в обоих случаях получаем по 9 суммирований-вычитаний при выполнении двух циклов умножения (для комбинации ..4..5..5 суммирований в первом цикле и 4 суммирования во втором; для комбинации ..6..5 5 суммирований в первом цикле и 4 вычитания во втором). Если же разряды группируются по парам, то комбинацию ...45 выгодно по-прежнему оставлять без изменения, а комбинацию ..65 представить в виде (1)..$\overline{3}\,\overline{5}$, в результате чего количество суммирований-вычитаний при выполнении пары циклов умножения уменьшится для этой комбинации на единицу (5 вычитаний в первом цикле и 3 вычитания во втором).

Так как суммарная вероятность комбинаций, в которых при группировке разрядов в пары можно сэкономить по одному суммированию-вычитанию, равна $\frac{1}{2}\cdot\frac{1}{n}$, то

$$A_2 = 2A_1 - \frac{1}{2}\cdot\frac{1}{n} = \frac{n}{2} - \frac{1}{2n}.$$

Пусть теперь нам известна величина A_{p-1}; найдем разность $A_p - A_{p-1}$.

Будем полагать, что преобразуемая группа из p цифр получена в результате того, что к соответствующей группе из p — 1 цифр была приписана одна цифра слева. Кроме того, мы предполагаем, что для соответствующей группы из p — 1 цифр было известно ее оптимальное преобразование. Теперь подсчитаем для различных случаев, сколько лишних суммирований-вычитаний потребуется при умножении на данную группу из p разрядов множителя по сравнению с соответствующей группой из p — 1 разрядов.

Рассмотрим сначала те комбинации из p цифр, в которых левая цифра есть 0.

Прежде всего очевидно, что во всех тех случаях, когда следующая (вторая слева) цифра меньше, чем ${}^n/_2$, количество суммирований-вычитаний при умножении на p цифр будет при этом таким же, как при умножении на соответствующую группу из p — 1 цифр.

Легко убедиться также, что если вторая цифра больше ${}^n/_2$, то количество суммирований-вычитаний при переходе от группы из p — 1 разрядов к группе из p разрядов должно увеличиться на 1. Действительно, если

старшая цифра в группе из p — 1 разрядов больше $n/2$, то, каковы бы ни были остальные разряды этой группы, в ее оптимальном представлении должна обязательно присутствовать единица, передаваемая в следующий по весу разряд. Если в группы объединяется по (p — 1) разрядов, то эта единица добавляется к следующей группе из (p — 1) разрядов. При этом вероятности всех комбинаций в этой следующей группе, кроме комбинации 00...0, сохраняются неизменными, комбинация 00...0 становится невозможной, но зато с той же вероятностью может появиться комбинация 1..00...0 (для которой также не требуется ни одного суммирования или вычитания); поэтому математическое ожидание количества суммирований-вычитаний при умножении на следующую группу разрядов не изменяется от того, что в нее передается единица из предыдущей группы. Однако когда мы, переходя от группы из p — 1 разрядов к группе из p разрядов, приписываем к ней слева один разряд и ставим в этот разряд цифру «0», то передаваемая в этот разряд из младших разрядов единица потребует лишнего суммирования при умножении на группу из p разрядов (в последнем из p циклов умножения). Попытка изменить способ преобразования младших разрядов с тем, чтобы не было передачи единицы в старший (p-й) разряд, привела бы к тому, что в (p — 1)-м цикле умножения количество суммирований-вычитаний возросло бы по крайней мере на 1.

Итак, если левая из p цифр есть 0, то в $n/2$ случаях (в комбинациях 00..., 01..., 02..., $0\left(\frac{n-2}{2}\right)$ …) количество суммирований-вычитаний такое же, как при умножении на соответствующую группу из p — 1 цифр; в $n/2$ — 1 случаях (в комбинациях $0\left(\frac{n+2}{2}\right)$..., $0\left(\frac{n+4}{2}\right)$...,..., $0\,(n-1)$...) оно на 1 больше, чем для соответствующих групп из p—1 цифр; последний из возможных случаев — когда левая из p цифр есть 0, а следующая за ней цифра в точности равна $n/2$ (т. е. когда имеем комбинацию $0\frac{n}{2}$...) — требует особого рассмотрения.

Если p = 2, то при умножении на группу из p цифр « $0\frac{n}{2}$» требуется такое же количество суммирований-вычитаний, как при умножении на соответствующую группу из p — 1 разрядов (при этом p — 1 = 1; соответствующая группа из p — 1 разрядов представляет собой просто 1 разряд, в котором, записана цифра $n/2$ и при умножении на который требуется $n/2$ суммирований; при умножении на группу из p = 2 цифр « $0\frac{n}{2}$ » тоже требуется $n/2$ суммирований). Если p >2, то изменение количества суммирований-вычитаний при переходе от группы из p — 1 разрядов вида $n/2$ … к группе из p разрядов вида $0\frac{n}{2}$… зависит от следующей цифры. Аналогично предыдущему можно показать, что в $n/2$ — 1 случаях (в комбинациях $0\frac{n}{2}\,0...$, $0\frac{n}{2}\,1$ …, $0\frac{n}{2}\frac{n-4}{2}$...) количество суммировании при умножении на группу из p разрядов остается таким же, как при умножении на соответствующую группу из p — 1 разрядов, в $n/2$ случаях (в комбинациях $0\frac{n}{2}\frac{n}{2}$..., $0\frac{n}{2}\frac{n+2}{2}$...,..., $0\frac{n}{2}(n-1)$...) она возрастает на 1, а в одном случае — для комбинации $0\frac{n}{2}\frac{n-2}{2}$... — требуется особое рассмотрение. В частности, при p = 3 умножение на группу разрядов «$0\frac{n}{2}\frac{n-2}{2}$» требует столько же суммирований-вычитаний, как умножение на соответствующую группу из p—1 разрядов «$\frac{n}{2}\frac{n-2}{2}$»; при p>3 изменение в количестве суммирований-вычитаний для комбинаций вида $0\frac{n}{2}\frac{n-2}{2}$... зависит от четвертой цифры. Приведенное рассуждение можно продолжить и дальше.

Теперь выпишем все те комбинации, начинающиеся с 0, для которых количество суммирований-вычитаний больше на 1, чем для соответствующих комбинаций из p — 1 разрядов. Сюда войдут:

— комбинации вида $0\frac{n+2}{2}$..., $0\frac{n+4}{2}$..., ..., $0\,(n-1)$ ...; их суммарная вероятность равна $\left(\frac{n}{2}-1\right)\frac{1}{n^2}$;

— комбинации .вида $0\frac{n}{2}\frac{n}{2}$..., $0\frac{n}{2}\frac{n+2}{2}$...,..., $0\frac{n}{2}(n-1)$... ; их суммарная вероятность $\frac{n}{2}\cdot\frac{1}{n^3}$;

— комбинации вида $0\frac{n}{2}\frac{n-2}{2}\frac{n}{2}$..., $0\frac{n}{2}\frac{n-2}{2}\frac{n+2}{2}$..., ..., $0\frac{n}{2}\frac{n-2}{2}(n-1)$...; их суммарная вероятность

$\left(\frac{n}{2}-1\right)\frac{1}{n^4}$; и т. д.

Таким образом, комбинации, начинающиеся с 0, в общей сложности внесут в разность $A_p - A_{p-1}$ следующую величину:

при p четном

$$\Delta^{(0)}_{\text{чет}} = (\frac{n}{2}-1)\frac{1}{n^2} + \frac{n}{2}\cdot\frac{1}{n^3} + (\frac{n}{2}-1)\frac{1}{n^4} + \ldots + (\frac{n}{2}-1)\frac{1}{n^p};$$

при p нечетном

$$\Delta^{(0)}_{\text{неч}} = (\frac{n}{2}-1)\frac{1}{n^2} + \frac{n}{2}\cdot\frac{1}{n^3} + (\frac{n}{2}-1)\frac{1}{n^4} + \ldots + \frac{n}{2}\cdot\frac{1}{n^p};$$

Отсюда для любого $p > 2$ можно записать

$$\Delta^{(0)} = \frac{n^{p+1} - n^p - n^2 + 1 - (-1)^p(n-1)}{2(n^2-1)n^p}.$$

Рассмотрим далее комбинации, начинающиеся с цифры 1. Нетрудно показать, что в тех случаях, когда в комбинациях, начинающихся с цифры 0, количество суммирований-вычитаний при переходе от групп по p — 1 разрядов к группам по p разрядов не возрастало, здесь будем иметь увеличение на 1; в тех случаях, когда в комбинациях, начинающихся с «0», мы имели увеличение количества суммирований-вычитаний на 1, в комбинациях, начинающихся с «1», увеличение количества суммирований-вычитаний составит 2. Учитывая, кроме того, что суммарная вероятность комбинаций, начинающихся с «1», равна $\frac{1}{n}$, найдем величину, которая вносится в разность $A_p - A_{p-1}$ всеми этими комбинациями:

$$\Delta^{(1)} = \Delta^{(0)} + \frac{1}{n}.$$

Точно так же

$$\Delta^{(2)} = \Delta^{(0)} + \frac{2}{n},$$

$$\Delta^{(3)} = \Delta^{(0)} + \frac{3}{n},$$

$$\ldots\ldots\ldots\ldots\ldots$$

$$\Delta^{\left(\frac{n-2}{2}\right)} = \Delta^{(0)} + \frac{\frac{n-2}{2}}{n}.$$

Для комбинаций, начинающихся с цифры $\frac{n}{2}$, можно провести рассмотрение, аналогичное тому, которое было проведено для комбинаций, начинающихся с цифры 0. Опуская его здесь, запишем прямо конечный результат:

$$\Delta^{\left(\frac{n}{2}\right)} = \Delta^{(0)} + \frac{\frac{n-2}{2}}{n} + \frac{1}{n^p}.$$

Затем, аналогично предыдущему,

$$\Delta^{\left(\frac{n+2}{2}\right)} = \Delta^{\left(\frac{n}{2}\right)} - \frac{1}{n} = \Delta^{(0)} + \frac{\frac{n-4}{2}}{n} + \frac{1}{n^p},$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\Delta^{(i-1)} = \Delta^{\left(\frac{n}{2}\right)} - \frac{\frac{n-2}{2}}{n} = \Delta^{(0)} + \frac{1}{n^p}.$$

Теперь можем получить (для $p > 2$)

$$A_p - A_{p-1} = \sum_{i=0}^{n-1}\Delta^{(i)} = \frac{n}{4} + \frac{(-n)^{1-p}-1}{2(n+1)}.$$

Наконец,

$$A_q = A_2 + \sum_{p=3}^{q} (A_p - A_{p-1}) .$$

Подставляя вычисленные выше значения A_2 и A_p—A_{p-1}, после некоторых преобразований найдем, что для любого четного n

$$A_q = \frac{1}{2(n+1)} \left\{ \frac{n}{n+1} \left[1-(\_ n)^{-q} \right] + \frac{n^2+n-2}{2} q \right\}.$$

Для нечетных n, как мы говорили, при любых q

$$A_q = qA_1 = \frac{q}{4}(n - \frac{1}{n}).$$

Объединяя эти результаты, получим (для любых n и q)

$$\frac{A_q}{q} = \frac{1}{4(n+1)} \left\{ n^2 + n - 2 + \frac{n}{q(n+1)} \left[1 + (-1)^n \right] \times \left[1 - (-n)^{-q} \right] + \frac{n-1}{2n} \left[1 - (-1)^n \right] \right\}.$$

Отсюда

$$\lim_{q\to\infty} \frac{A_q}{q} = \frac{1}{4(n+1)} \left\{ n^2 + n - 2 + \frac{n-1}{2n} \left[1 - (-1)^n \right] \right\}.$$

В полученном выражении содержится ответ на первый из поставленных в начале настоящего раздела вопросов. Если основание системы счисления n заранее известно, то при достаточно большом количестве разрядов m минимальное количество суммирований-вычитаний, которое потребуется в ходе выполнения умножения (при условии использования наилучших для данного n логических методов ускорения умножения), в среднем равно примерно $m \lim\limits_{q\to\infty} \frac{A_q}{q}$.

Иначе говоря, на каждый n-ичный разряд множителя в среднем придется по ($\lim\limits_{q\to\infty} \frac{A_q}{q}$) суммирований-вычитаний; величина $m \lim\limits_{q\to\infty} \frac{A_q}{q}$ равна примерно математическому ожиданию суммы цифр $\sum\limits_{i=0}^{m} |a_i|$ в оптимальном представлении множителя. Выражения эти тем точнее, чем больше m. Однако, имея приведенное строчкой выше выражение для A_q / q, нетрудно подсчитать, что даже при небольших значениях $q = m$ отношение A_m / m весьма близко к предельному значению.

В частности, для двоичной системы, подставляя $n = 2$, найдем $(\lim\limits_{q\to\infty} \frac{A_q}{q})_{n=2} = \frac{1}{3}$, для троичной системы $(\lim\limits_{q\to\infty} \frac{A_q}{q})_{n=3} = \frac{2}{3}$, для четверичной системы $(\lim\limits_{q\to\infty} \frac{A_q}{q})_{n=4} = \frac{9}{10}$, …, для десятичной $(\lim\limits_{q\to\infty} \frac{A_q}{q})_{n=10} = \frac{27}{11}$.

Таким образом, лучший результат, который может быть достигнут применением логических методов ускорения умножения в двоичной системе — это $^1/_3$ суммирования или вычитания на каждый двоичный разряд множителя,— как раз то, что дает метод последовательных преобразований множителя (см. 4.2.2); дальнейшие усовершенствования, может быть, позволят получить более удобное правило преобразования и в связи с этим упростить схему управления, но большего быстродействия получить не удастся.

Аналогичным образом, скажем, в десятичной системе наиболее сильные логические методы ускорения могут дать $^{27}/_{11} \approx 2{,}45$ суммирований-вычитаний на десятичный разряд — примерно столько же, сколько дает сравнительно простой метод перехода к симметричному диапазону цифр (2,5 суммирований-вычитаний на разряд); впрочем, в 4.2.3, где описывался этот метод, мы упоминали уже о том, что при большом я дальнейшее усложнение логических методов ускорения умножения не может дать значительного эффекта.

Чтобы эти результаты для разных я можно было сопоставить, нужно учесть, что при равной точности количество разрядов обратно пропорционально log n. Примем, как мы это делали в 1.2, что количество разных чисел, с которыми должна оперировать машина, равно N; тогда необходимое количество n-ичных разрядов равно $\log_n N$. Функция

$$f_\infty(n) = \frac{\left(\lim\limits_{q\to\infty} \frac{A_q}{q} \right)_n \cdot \log_n N}{\left(\lim\limits_{q\to\infty} \frac{A_q}{q} \right)_2 \cdot \log_2 N} = \frac{3}{\log_2 n} \left(\lim_{q\to\infty} \frac{A_q}{q} \right)_n$$

показывает, во сколько раз среднее количество суммирований-вычитаний, необходимых при выполнении умножения в позиционной n-ичной системе с использованием наиболее сильных логических методов ускорения, больше, чем аналогичная величина в двоичной системе, если точность вычислений одинакова. Значения $f_\infty(n)$ для нескольких начальных значений n приведены в таблице 4-1.

Таблица 4-1

Значения функции

n	2	3	4	5	6	7	8	9	10	
$f_\infty(n)$	1,00	1,26	1,35	1,55	1,66	1,84	1,95	2,10	2,22	...

Интересно, что соотношения здесь примерно такие же, как в таблице 1-3 (стр. 29) для функции $f_0(n)$; таким образом, и при использовании логических методов ускорения двоичная система сохраняет свои преимущества перед другими системами с точки зрения скорости выполнения умножения.

Вообще нетрудно показать, что для $n \geq 2$ наименьшее значение $f_\infty(n)$ есть $f_\infty(2) = 1$; с ростом n функция $f_\infty(n)$ монотонно возрастает. Таким образом, двоичная система является оптимальной с точки зрения получения наибольшей скорости выполнения умножения при использовании логических методов ускорения, так что $n_{\text{опт}} = 2$; результаты, достигаемые в двоичной системе, есть абсолютный предел возможностей логических методов ускорения умножения.

4.3. Аппаратные методы 1-го порядка ускорения умножения в параллельных устройствах (для двоичной системы)

4.3.1. Методы, основанные на добавлении сумматоров и цепей сдвига.

В предыдущем разделе, говоря о скорости выполнения умножения, мы учитывали только количество суммирований-вычитаний, необходимых в процессе выполнения умножения, но совсем опускали вопрос о времени для выполнения сдвигов. Между тем **в** параллельных устройствах времена выполнения одного суммирования или вычитания и выполнения однократного сдвига близки между собой. В 2.4 указывалось, что параллельный сумматор имеет смысл строить в том случае, когда общая задержка переносов по всем m разрядам, равная $m\tau_E$ (где τ_E—задержка переноса в одном разряде), соответствует по длительности примерно одному такту сдвига τ_c. Если учесть, что время сложения τ_Σ складывается из времени установления переносов $m\tau_E$ и времени образования суммы τ_B:

$$\tau_\Sigma = m\tau_E + \tau_B,$$

то станет ясным, почему во многих параллельных устройствах такт сдвига τ_c оказывается равным примерно половине такта суммирования $\left(\frac{1}{2}\tau_\Sigma\right)$.

При использовании логических методов ускорения умножения количество тактов суммирования-вычитания сокращается до $^1/_3$ на двоичный разряд, количество же тактов сдвига остается неизменным (один сдвиг на разряд).

Если принять, что $\tau_c = \frac{1}{2}\tau_\Sigma$, то при выполнении умножения по первому или по третьему варианту (см. 4.1.2), когда такты сдвига не перекрываются с тактами суммирования, применение наиболее сильных логических методов ускорения позволит получить время умножения равным

$$\frac{1}{3}\tau_\Sigma + \tau_c = \frac{5}{6}\tau_\Sigma$$

на каждый разряд множителя; при тех же условиях выполнение умножения по второму или по четвертому варианту, когда такты сдвига перекрываются с тактами суммирования, потребовало бы времени

$$\frac{1}{3}\tau_\Sigma + \frac{2}{3}\tau_c = \frac{2}{3}\tau_\Sigma$$

на каждый разряд множителя. Таким образом, в рассмотренных случаях сдвиги занимают столько же или даже больше времени, чем суммирования-вычитания.

Первые два из рассматриваемых ниже аппаратных методов ускорения — введение дополнительных цепей сдвига и введение сдвинутого сумматора — имеют целью именно сокращение времени, затрачиваемого на сдвиги.

1 °. Рассмотрим сначала введение дополнительных цепей сдвига. Для определенности представим себе, что умножение выполняется по первому варианту и что арифметическое устройство в связи с этим построено в соответствии с рис. 4-2, *а* на стр. 145. Однако для ускорения умножения в регистрах A и B в дополнение к цепям сдвига вправо на 1 разряд устроены также цепи сдвига вправо на 2 разряда (рис. 4-3).

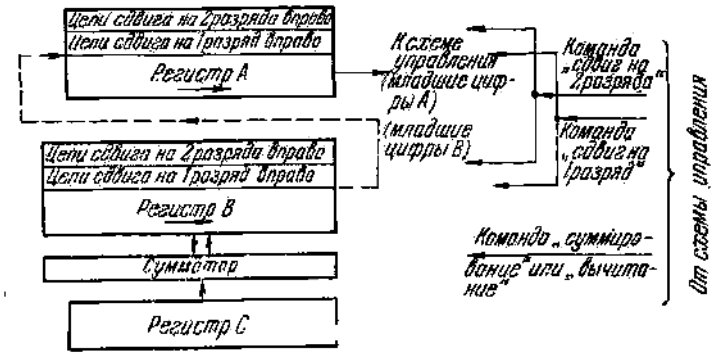


Рис. 4-3. Построение арифметического устройства при использовании для ускорения умножения дополнительных цепей сдвига.

Дополнительными цепями сдвига можно распорядиться по-разному. Но наибольший эффект они дают в комбинации с методом последовательного преобразования множителя, изложенным в 4.2.2*).

В схему преобразования по-прежнему будут поступать каждый раз три разряда непреобразованного множителя: очередной разряд, соседний справа (младший) и соседний слева (старший); предыдущий разряд п р е о б р а з о в а н ного множителя, который использовался в правиле преобразования, приведенном на стр. 154, здесь нам не потребуется. Правило преобразования по существу останется точно таким же, как, прежде, однако мы дополним его одной важной деталью: если очередная цифра преобразованного множителя есть + 1 или — 1 или если очередная цифра непреобразованного множителя совпадает с соседней слева его цифрой, то сдвиг вправо в регистрах *A* и *B* будем производить сразу на два разряда. В обоих этих случаях следующая цифра преобразованного множителя должна получиться нулем; следовательно, ни суммирования, ни вычитания в следующем цикле умножения все равно не было бы, и этот цикл умножения можно просто опустить. Это мы и делаем, производя сдвиг множителя и суммы частичных произведений на 2 разряда. Зато когда после очередного сдвига (на 1 или на 2 разряда) в схему преобразования поступает очередная тройка разрядов множителя, то уже заранее известно, что предыдущая цифра преобразованного множителя была нулем; поэтому, собственно, и можно не запоминать предыдущую цифру преобразованного множителя и не принимать ее во внимание при преобразовании очередного разряда.

С учетом указанных дополнений правило умножения можно свести в таблицу 4-2.

Для того чтобы выяснить, какие скорости обеспечивает указанный метод, нужно прежде всего подсчитать вероятности комбинаций цифр, которые перечислены в левой колонке таблицы 4-2. Дело в том, что при использовании рассматриваемого метода комбинации эти не равновероятны. В первом цикле умножения (когда справа от очередного разряда множителя находится цифра несуществующего разряда) вероятность каждой из

Т а б л и ц а 4-2

Правило умножения при использовании дополнительных цепей сдвига в комбинации с методом последовательного преобразования множителя

Комбинация цифр в непреобразованном множителе (очередная цифра — жирным шрифтом)	Очередная цифра преобразованного множителя	Сдвиг
0 **0** 0	0	на 2 разряда
0 **0** 1	+1	» 2 »
0 **1** 0	+1	» 2 »
0 **1** 1	0	на 1 разряд
1 **0** 0	0	» 1 »
1 **0** 1	—1	на 2 разряда
1 **1** 0	—1	» 2 »
1 **1** 1	0	» 2 »

комбинаций 0**0**0, 0**1**0, 1**0**0 и 1**1**0 равна $^1/_4$, вероятности остальных комбинаций равны нулю. Затем от цикла к циклу вероятности постепенно изменяются. Обозначим через p_0 предел, к которому стремится вероятность комбинации 0**0**0, через p_1 — предел вероятности комбинации 0**0**1 и т. д.

Теперь обратим внимание на то, что после комбинаций 0**0**0, 0**0**1 и 0**1**0 в результате очередного сдвига на краю регистра с равными вероятностями могут получаться комбинации 0**0**0, 0**1**0, 1**0**0 и 1**1**0; после комбинации 0**1**1 с равными вероятностями получаются либо 1**0**1, либо 0**0**1; после комбинации 1**0**0 с равными вероятностями

*) Описываемый ниже метод в литературе, по-видимому, не был описан. Преимущества его перед другими способами использования дополнительных цепей сдвига весьма значительны — ср. со способом, описанным в предыдущей работе автора («Арифметические устройства электронных цифровых машин», Физматгиз, 1958, стр. 128—130).

получаются либо 0**1**0, либо 1**1**0; после комбинаций 1**0**1, 1**1**0 и 1**1**1 с равными вероятностями появляются комбинации 0**0**1, либо 0**1**1, либо 1**0**1, либо 1**1**1. Поэтому

$$p_0 = \frac{1}{4}(p_0 + p_1 + p_2),$$

$$p_1 = \frac{1}{2}p_3 + \frac{1}{4}(p_5 + p_6 + p_7),'$$

$$p_2 = \frac{1}{4}(p_0 + p_1 + p_2) + \frac{1}{2}p_4,$$

$$p_3 = \frac{1}{4}(p_5 + p_6 + p_7),$$

$$p_4 = \frac{1}{4}(p_0 + p_1 + p_2),$$

$$p_5 = \frac{1}{2}p_3 + \frac{1}{4}(p_5 + p_6 + p_7),$$

$$p_6 = \frac{1}{4}(p_0 + p_1 + p_2) + \frac{1}{2}p_4,$$

$$p_7 = \frac{1}{4}(p_5 + p_6 + p_7).$$

Кроме того, конечно,

$$p_0 + p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 = 1.$$

Решая совместно эти уравнения, найдем

$$p_3 = p_4 = p_7 = {}^1/_{10},$$
$$p_0 = p_1 = p_2 = p_5 = p_6 = {}^3/_{20}.$$

Теперь можно найти, на сколько разрядов в среднем производится сдвиг. Поскольку сдвиг на 1 разряд выполняется в комбинациях 0**1**1 и 1**0**0, суммарная вероятность которых равна $p_3 + p_4 = {}^1/_5$, а сдвиг на 2 разряда — во всех остальных комбинациях с вероятностью ${}^4/_5$, то в среднем сдвиг выполняется на $\frac{1}{5}\cdot 1 + \frac{4}{5}\cdot 2 = \frac{9}{5}$ разряда. Таким образом, на каждый разряд множителя приходится ${}^5/_9$ тактов сдвига и столько же циклов умножения (каждый из которых наряду со сдвигом включает преобразование очередной цифры множителя и, может быть, суммирование или вычитание).

Далее можно заметить, что в некотором цикле умножения суммирование или вычитание выполняется только в комбинациях 0**0**1, 0**1**0, 1**0**1 и 1**1**0, суммарная вероятность которых равна $p_1 + p_2 + p_5 + p_6 = {}^3/_5$Так как на каждый разряд множителя приходится в среднем ${}^5/_9$ цикла умножения, то среднее количество суммирований-вычитаний, приходящееся на 1 разряд множителя, равно $\frac{3}{5}\cdot\frac{5}{9} = \frac{1}{3}$. Как и следовало ожидать, введение дополнительных цепей сдвига не изменило среднего количества тактов суммирования-вычитания, но почти вдвое уменьшило количество тактов сдвига.

Если по-прежнему считать, что такт суммирования τ_Σ вдвое длиннее, чем такт сдвига τ_c, то среднее времяумножения получится равным

$$\frac{1}{3}\tau_\Sigma + \frac{5}{9}\tau_c = \frac{11}{18}\tau_\Sigma$$

на каждый разряд множителя — почти на 40% меньше, чем при отсутствии дополнительных цепей сдвига (для этого случая мы имеем $\frac{5}{6}\tau_\Sigma$ на разряд — см. стр. 164).

Максимальное время умножения подсчитать здесь несколько сложнее. Ясно, что наибольшее количество тактов суммирования-вычитания — ${}^1/_2$ такта на разряд — получится в том случае, когда множитель имеет вид... 101010...; однако при этом и количество сдвигов составляет всего ${}^1/_2$ на разряд. Наибольшее количество сдвигов — ${}^2/_3$ на разряд — получится, когда множитель имеет вид... 1001001...; но при этом на каждый разряд множителя придется всего ${}^1/_3$ такта суммирования-вычитания. Комбинация, в которой получается максимум по времени, и величина этого максимума зависят от соотношения между τ_Σ и τ_c. Во всяком случае, максимальное время умножения всегда меньше, чем $\frac{1}{2}\tau_\Sigma + \frac{2}{3}\tau_c$ на разряд.

2°. Рассмотрим теперь в в е д е н и е с д в и н у т о г о с у м м а т о р а. Предположим, как и прежде, что умножение выполняется по первому варианту (см. 4.1.2) и что поэтому построение множительного устройства в общих чертах соответствует рис. 4-2, *а* на стр. 145. В этом построении, однако, цепи сдвига вправо на 1 разряд, имевшиеся в регистрах *A и B,* заменены цепями сдвига на 2 разряда, а между регистрами *B* и *C*, наряду с имеющимся обычно сумматором, включен дополнительно еще один сумматор; последний позволяет добавлять к содержимому регистра *B* удвоенное число из регистра *C* (рис. 4-4, *а).* Оба сумматора устроены одинаково, но ко входам второго из них выходные сигналы регистра *C* присоединены со сдвигом влево на 1 разряд.

Вместо того чтобы устраивать дополнительный сумматор, можно было бы предусмотреть два ряда элементов «и» и далее ряд элементов «или», с тем чтобы число из регистра *C* передавалось на входы сумматора либо через один ряд элементов «и» (прямо), либо через другой ряд (со сдвигом на один разряд). Однако такое решение, показанное на рис. 4-4,б, далеко не всегда проще осуществить, чем вариант с лишним сумматором.

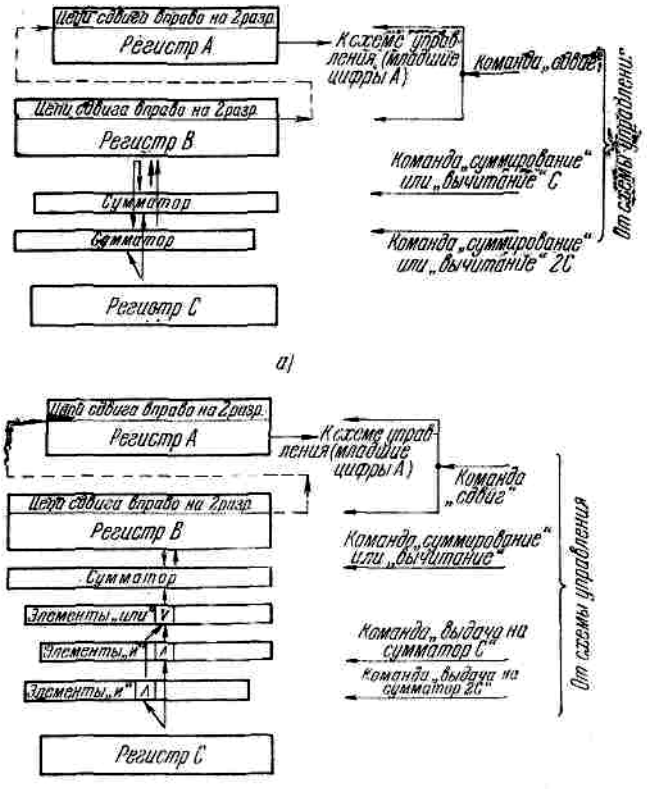


Рис. 4-4. Построение арифметического устройства при использовании для ускорения умножения сдвинутого сумматора: а) с дополнительным сумматором; *б)* с дополнительными логическими элементами для передачи множимого на сумматор прямо или со сдвигом.

Дело в том, что в каждом разряде сумматора входной сигнал одного из слагаемых поступает обычно на несколько логических элементов; поэтому на выходах элементов «или» могут потребоваться усилители-формирователи сигналов; кроме того, в сумматоре иногда используются прямые и инверсные выходы триггеров регистра *C,* что потребовало бы удвоения количества элементов «и» и «или». Оценивая в дальнейшем рассматриваемый метод ускорения умножения, мы будем считать, что для его осуществления требуется лишний ряд сумматоров, т. е. что схема строится в соответствии с рис. 4-4, *а.*

Дополнительный сумматор, как и дополнительные цепи сдвига, можно использовать по-разному. По литературе*) известно применение сдвинутого сумматора в комбинации с логическим методом группировки разрядов по парам (см. 4.2.2). При этом поступают следующим образом. Если при расшифровке очередной пары разрядов оказывается, что суммирование или вычитание должно производиться в первом из двух ближайших циклов умножения, то это выполняется с помощью основного сумматора, если суммирование или вычитание требуется во втором из пары циклов умножения, то соответствующие операции выполняются через дополнительный сумматор (без такта сдвига суммы частичных произведений вправо); если в течение ближайших двух циклов умножения не требуется производить ни сложений, ни вычитаний, то команды не подаются ни на один из сумматоров. В любом случае далее следует сдвиг множителя и суммы частичных произведений сразу на 2 разряда, а затем расшифровка следующей пары разрядов множителя.

Ясно, что количество суммирований или вычитаний при этом получается точно таким же, как и при отсутствии сдвинутого сумматора — в среднем $^3/_8$, а в худшем случае $^1/_2$ суммирования или вычитания на разряд. Количество сдвигов всегда равно $^1/_2$ на разряд множителя. Если полагать, как мы это делали раньше, что такт суммирования

*) Впервые, вероятно, этот метод описан в докладе Л е б е д е в а С. А. «Электронные вычислительные машины», Сессия АН СССР по научным проблемам автоматизации производства, Пленарные заседания, АН СССР, 1957.

τ_Σ равен двум тактам сдвига τ_c, то среднее время умножения окажется равным

$$\frac{3}{8}\tau_\Sigma \cdots + \frac{1}{2}\tau_C = \frac{5}{8}\tau_\Sigma$$

на каждый разряд множителя, т. е. несколько хуже, чем для описанного выше метода применения дополнительных цепей сдвига. При более рациональном способе преобразования множителя введение дополнительного сумматора могло бы дать и несколько лучшие результаты, чем введение дополнительных цепей сдвига. В любом случае, однако, эти методы близки между собой с точки зрения достигаемых скоростей.

3°. В заключение рассмотрим еще один аппаратный метод 1-го порядка ускорения умножения — на этот раз такой, который позволит сократить не только количество тактов сдвига, но и количество суммирований-вычитаний. Мы назовем его методом *разделения множителя.* Для определенности снова будем полагать, что выполнение умножения осуществляется по первому варианту (см.4.1.2). Кроме того, будем считать, что задача получения полного 2*m*-разрядного произведения двух *m*-разрядных чисел не ставится и что в результате умножения необходима такая же точность, как в исходных числах — до *m-го* разряда. Если прежде получение 2*m*-разрядного произведения доставалось нам фактически «бесплатно», то здесь это потребовало бы довольно значительных дополнительных затрат оборудования.

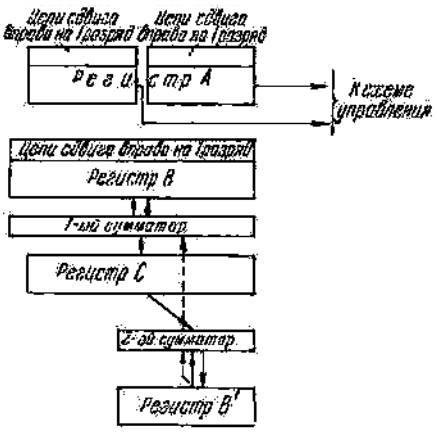


Рис. 4-5. Построение арифметического устройства при использовании для ускорения умножения метода

Идея метода состоит в том, что множитель разделяется, скажем, на 2 части, и умножение на старшую и младшую части выполняется в одно и то же время независимо одно от другого; полученные результаты затем суммируются с необходимым сдвигом.

На рис. 4-5 показано, каким должно быть построение арифметического устройства для осуществления указанного метода ускорения умножения. Регистр множителя *A* разделен на 2 части; сдвиги в них могут выполняться, если необходимо, в разное время. В схему управления поступают каждый раз младшие разряды старшей части множителя и младшие разряды младшей части. В зависимости от имеющихся в них цифр устройство управления вырабатывает команды суммирования или вычитания для 1-го (основного) сумматора, связанного с регистром *B,* и для 2-го (дополнительного) сумматора, связанного с регистром *B′,* а также команды «сдвиг» для регистров *B* и *B′* и соответствующих частей регистра *A.* В регистре *B* при этом накапливается сумма частичных произведений множимого на старшие разряды множителя, а в регистре *B′* — сумма частичных произведений множимого на младшие разряды. Поскольку, как мы условились, произведение должно быть получено лишь с *m* знаками, регистр *B′* и дополнительный сумматор имеют половинное количество разрядов, а цифры, выдвигаемые из регистров *B и B′* при сдвигах вправо, теряются. В заключение умножения через 1-й сумматор производится суммирование содержимого регистров *B* и *B′,* причем результат передается в *B.* При выполнении округления необходимо учитывать, что погрешность полученного результата может находиться в пределах от нуля до минус двух единиц младшего разряда; если бы регистры *B* и *B′* и 1-й сумматор содержали по одному лишнему разряду справа, эта погрешность была бы вдвое меньше.

Ясно, что при выполнении умножения отдельно на старшую и отдельно на младшую части множителя могут применяться любые из описанных выше логических или аппаратных методов ускорения. В любом случае разделение множителя должно примерно вдвое сокращать длительность умножения. Однако при использовании различных логических способов выигрыш в среднем времени будет получаться менее значительным, чем обычно, так как умножение заканчивается лишь после того, как оно выполнено и в старшей, и младшей части; длительность его определяется менее благоприятной комбинацией цифр из имеющихся в одной и в другой половине множителя. Чем меньше разрядов содержит множитель, тем заметнее этот эффект. Поэтому при обычном количестве разрядов разделять множитель больше чем на 2 части вряд ли имеет смысл.

4.3.2. Методы, основанные на запоминании цифр переносов.

Те методы, которые рассматривались до сих пор, имели целью сокращение количества тактов суммирования или тактов сдвига, необходимых в ходе выполнения умножения. Рассматриваемые ниже методы направлены вместо этого (или в дополнение к этому) на сокращение длительности самого такта суммирования. Сокращение оказывается возможным потому, что в процессе умножения требуется не однократное суммирование, а длинная цепь суммирований.

Длительность такта суммирования τ_Σ равна

$$\tau_\Sigma = m\tau_E + \tau_B,$$

где m — количество разрядов в суммируемых числах, τ_E — задержка сигнала переноса в одном разряде сумматора, τ_B — время формирования сигналов суммы. Сокращение длительности τ_Σ мы попытаемся получить за счет сокращения первого слагаемого ($m\tau_E$); чем больше $m\tau_E$ по сравнению с τ_B, тем эффективнее излагаемые ниже методы.

На первых порах мы будем предполагать, что умножение выполняется либо по второму, либо по четвертому варианту (см. 1.2), т. е. так, что сумма частичных произведений (регистр *B)* никуда не сдвигается. Позже это ограничение будет снято.

Представим себе, что цепь переносов в параллельном сумматоре разорвана посередине — скажем, на входе *i*-го разряда. Выходной сигнал переноса *(i* — 1)-го разряда вместо того, чтобы поступать на вход переноса *i*-го разряда, запоминается каким-нибудь специальным элементом, например триггером; на вход переноса *i*-го разряда сигналы будут поступать с выхода этого триггера (рис. 4-6).

К началу умножения одновременно с гашением регистра частичных произведений (регистра *B)* гасится **и** дополнительный триггер. Поэтому при первом суммировании на вход переноса *i*-го разряда поступает нуль. Результат первого суммирования получается фактически в виде двух чисел: числа, находящегося в регистре *B,* и числа, представленного цифрой в дополнительном триггере.

При каждом из последующих суммирований цифра из дополнительного триггера будет поступать на вход переноса *i*-го разряда, так что при этом, по сути дела, будут суммироваться три числа: предыдущее число из регистра *B,* число, представленное цифрой в дополнительном триггере, и новое частичное произведение. При этом выходной сигнал переноса *(i*— 1)-го разряда вновь поступает не на вход *i*-го разряда, а на вход дополнительного триггера, так что результат снова получается в виде двух чисел.

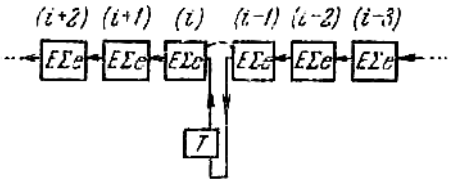


Рис. 4-6. Разрыв цепи переносов параллельного сумматора у *i*-го разряда.

Так продолжается до самого конца; в заключение умножения потребуется отдельный такт суммирования, чтобы добавить цифру из дополнительного триггера к содержимому регистра *B* и получить, таким образом, окончательный результат. Зато в каждом из промежуточных суммирований на выполнение переносов можно будет отводить вдвое меньше времени, чем обычно,—примерно $\frac{m}{2}\tau_E$, и длительность каждого такта суммирования будет равна

$$\tau_\Sigma \approx \frac{m}{2}\tau_E + \tau_B.$$

Разделив таким же образом сумматор не на две, а на три части, можно получить длительность такта суммирования

$$\tau_\Sigma \approx \frac{m}{3}\tau_E + \tau_B.$$

и т. д.

Нужно учесть, правда, что когда сумматор делится на 2 части, то в заключительном такте в младшей части добавляется нуль, и длина пробега сигналов переноса равна максимум *m/2* разрядов, т. е. заключительный такт по длительности не отличается от всех других тактов; если же цепь переносов делится на большое число *(k)* частей, то в заключительном такте переносы проходят через m — $m/k \approx m$ разрядов, так что длительность заключительного такта суммирования должна быть почти $m\tau_E + \tau_B$, как при обычном сложении. Однако в общем балансе времени умножения заключительное суммирование особой роли не играет. Сокращение же всех основных тактов суммирования может дать весьма значительный эффект.

Разделение сумматора на 2 или, скажем, на 3 части требует добавления всего одного или двух триггеров (независимо от количества разрядов в перемножаемых числах). Поэтому с формальной точки зрения метод этот по своим свойствам ближе к логическим, чем к аппаратным методам ускорения. Однако большей частью задача ставится так, что количество частей, на которые делится сумматор, должно обеспечить максимальную длину пробега не выше заданной величины. При этом, конечно, количество дополнительных триггеров пропорционально количеству разрядов в перемножаемых числах, и мы имеем дело с аппаратным методом ускорения умножения первого порядка.

Наиболее известен предельный случай — когда дополнительные триггеры имеются во всех разрядах сумматора и во всех тактах суммирования, кроме заключительного, пробег сигналов переноса от разряда к разряду отсутствует вообще[*]). Такт суммирования сокращается до величины τ_B, которая обычно равна такту сдвига τ_C.

[*]) Многие авторы предлагали этот метод независимо друг от друга. Впервые он изложен, вероятно, в работе B u r k s A., G o l d s t i n e H., von N e u m a n n J., 1947—см. русский перевод: «Кибернетический сборник», вып. 9, 1964. С несколько иным объяснением (см. 4.6) метод неоднократно использовался на практике. Именно по этой причине приводимое здесь описание было опущено в предыдущей книге автора (см. ссылку на стр. 6). Сейчас, однако, мы решили изложить и то и другое объяснения, так как они приводят к несколько различным идеям в дальнейшем развитии.

Любые аппаратные или логические методы ускорения, описанные в предыдущих разделах, могут применяться совместно с разобранным методом. Однако не всегда это имеет смысл. Например, если такты суммирования перекрываются по времени с тактами сдвига (что возможно, когда умножение выполняется по второму или по четвертому варианту) и если по длительности эти такты равны, то не имеет, очевидно, смысла принимать какие-либо меры для сокращения количества суммирований. С другой стороны, имеет прямой смысл, например, разделить множитель на 2 части, как было описано в 4.3.1, и вести умножение раздельно на младшую и старшую части; другие методы ускорения тоже могут быть полезны.

Если дополнительные триггеры для запоминания переносов имеются во всех разрядах, то они образуют как бы четвертый регистр арифметического устройства — мы будем называть его регистром *E.* В этом регистре можно устроить цепи сдвигов, или можно его выходы присоединить ко

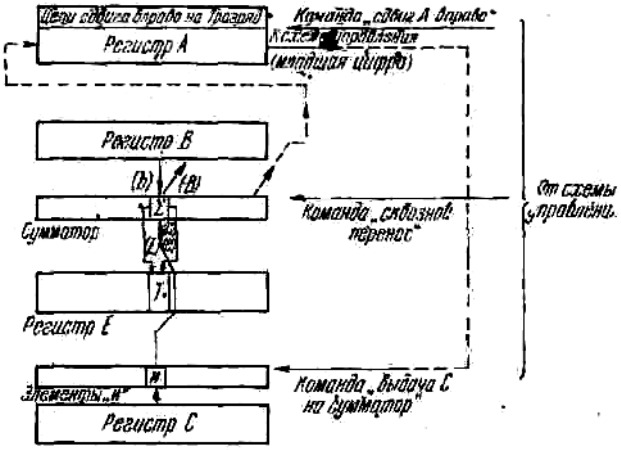


Рис. 4-7. Построение арифметического устройства при использовании для ускорения умножения запоминания переносов.

входам сумматора с определенным сдвигом; тогда не обязательно применять те варианты выполнения умножения, в которых частичные произведения не сдвигаются (второй или четвертый), но можно использовать и любой другой вариант выполнения умножения.

На рис. 4-7 в качестве примера показано одно из возможных построений арифметического устройства, в котором имеется полноразрядный регистр запоминания переносов *E* (т. е. дополнительные триггеры для запоминания переносов во всех разрядах), а умножение выполняется в соответствии с первым вариантом (4.1.2).

В течение всего времени умножения в каждом такте (кроме заключительного) выполняется сдвиг в регистре *A* и одновременно суммирование без сквозных переносов. При этом цифры с выходов *E* одноразрядных сумматоров поступают на входы соответствующих триггеров регистра *E,* откуда они через такт возвращаются на входы *e* в те же разряды сумматора; цифры с выходов *B* одноразрядных сумматоров поступают на входы триггеров регистра *B* со сдвигом на 1 разряд вправо, так что в следующем такте они возвращаются на входы *b* соседних справа разрядов сумматора. Таким образом, здесь каждый из сигналов переноса вместо того, чтобы передаваться в следующий разряд, запоминается в своем разряде, но в следующем такте к нему пододвигается соответствующая цифра суммы частичных произведений.

Пропускать такты суммирования при этом, конечно, нельзя. Если бы мы выполнили сдвиг в регистре *B* без суммирования, то в следующем такте цифры из регистра *E* складывались бы не с теми разрядами *B,* к которым они относятся, а со сдвинутыми на 1 разряд. Поэтому, собственно и не предусмотрены отдельные цепи сдвига в регистре *B,* а сдвиг выполняется определенной разводкой выходов сумматора. Однако, производя суммирование в каждом такте, мы либо передаем на сумматор множимое (*C*), либо не передаем — в зависимости от очередной цифры множителя, таким образом, предыдущая сумма частичных произведений, представленная в виде двух чисел (регистр *B* и регистр *E),* либо суммируется с множимым, либо суммируется с нулем.

Заметим, что цифры *E* в каждом такте имеются лишь для *m* старших разрядов суммы частичных произведений; младшие разряды в готовом виде появляются один за другим на выходе суммы («*B*») младшего разряда сумматора, откуда их можно вдвигать в освобождающиеся слева разряды регистра *A*. В. заключительном такте в старших разрядах должно быть выполнено суммирование чисел из регистров *B* и *E* (со всеми сквозными переносами).

Если полагать, что суммирование без сквозных переносов может быть произведено за время однократного сдвига τ_c, то длительность умножения оказывается равной

$$m\tau_c + \tau_\Sigma.$$

П р и м е р . Пусть множимое равно

$$C = .110111,$$

а множитель —

$$A = .101011.$$

Множитель помещается в начале умножения в регистр *A,* множимое — в регистр *C,* а регистры *B* и *E* гасятся.

При первом суммировании, поскольку младшая цифра *A* есть единица, число *C* передается на сумматор, так что идет суммирование нуля из регистра *B,* нулей из регистра *E* и числа *C.* При этом переносы из разряда в разряд не возникают, так что в *E* остаются нули, а результат суммирования, равный C, передается со сдвигом на 1 разряд вправо в B; младшая цифра результата передвигается в регистр *A* (ниже цифры, передаваемые в *A,* выделяются курсивом). В следующих тактах таким же образом суммируются числа из регистров *B, E* и, может быть, *C.*

Итак,

начальное состояние	(*B*) .000000	
	(*E*) .000000	
	(*C*) .110111	
результат первого такта	(*B*) .(0)11011	*1*
	(*E*) . 000000	
	(*C*) 110111	
результат второго такта (суммирования *B, E* и *C)*	(*B*) .(0)10110	*01*
	(*E*) . 0 10011	
	(*C*) (.1 10111)	
результат третьего такта (суммирования *B* и *E* без *C*, так как третья цифра *A* есть 0)	*(B)* . (0) 00010	*101*
	(*E*) . 0 10010	
	(C) . 1 10111	
результат четвертого такта (суммирования *B, E* и C)	(*B*) .(0) 10011	*1101*
	(*E*) . 0 10010	
	(*C*) (.1 10111)	
результат пятого такта (суммирования *B* и *E* без *C)*	(*B*) . (0) 00000	*11101*
	(*E*) . 0 10010	
	(*C*) . 1 10111	
результат шестого такта (суммирования *B, E* и *C*)	(*B*) .(0)10010	*111101*
	(*E*) . 0 10010	

Далее выполняется заключительное суммирование чисел *B* и *E* со сквозными переносами, в результате чего получим

.100100*111101*

что и является окончательным результатом умножения.

Интересно отметить, что в рассмотренном построении арифметического устройства во всех тактах умножения, кроме последнего, параллельный сумматор фактически распадается на *m* отдельных последовательных сумматоров, каждый из которых состоит, как обычно, из одного одноразрядного сумматора и устройства запоминания на один такт цифры двоичного переноса. К этому обстоятельству мы вернемся еще в 4.6.

4.3.3. Сравнительные оценки аппаратных методов ускорения умножения.

Уже из того, что было изложено выше, видно, что способы выполнения умножения весьма разнообразны. Ряд методов обсуждается еще и в следующих разделах.

Выбирая то или иное из возможных решений, конструктор волен руководствоваться различными соображениями. В некоторых случаях важно получить максимальную скорость — чего бы это ни стоило в отношении количества оборудования; иногда, наоборот, устройство должно быть возможно более экономным, а скорость выполнения умножения решающей роли не играет. Во многих случаях для сравнительной оценки различных методов удобно пользоваться величиной *эффективности метода Q.*

Пусть какой-либо метод умножения выбран в качестве опорного (эффективность этого метода *Q* будем считать равной единице); обозначим время выполнения умножения по этому методу через τ, а количество оборудования, необходимого для его реализации в арифметическом устройстве, через *N. Эффективностью* некоторого метода *x* назовем отношение

$$Q_x = \frac{\tau\ N}{\tau_x N_x},$$

где τ_x и N_x— соответственно время выполнения умножения и количество оборудования при использовании метода *x.* Отношение это показывает, во сколько раз сокращение времени при использовании рассматриваемого метода превосходит увеличение количества оборудования; чем больше величина *Q*, тем более эффективным является метод выполнения умножения.

Чтобы оценка эффективности того или иного аппаратного метода ускорения умножения была более объективной, она должна отражать также возможность использования логических методов повышения

быстродействия при использовании данного аппаратного метода. Во всех случаях, когда применение логических методов возможно, мы будем при расчете времени выполнения умножения полагать, что одновременно с аппаратным методом ускорения умножения используется наиболее сильный из возможных логических методов.

Соотношение между величинами Q_x не зависит, конечно, от того, какой из методов выбран в качестве опорного. Зато оно сильно зависит от того, из каких физических элементов предполагается строить множительное устройство, каковы соотношения в сложности оборудования для этих элементов и каковы соотношения в длительности выполнения элементарных операций. Поэтому в каждом конкретном случае требуется специальный расчет эффективности различных методов.

Покажем на практическом примере, как выполняется такой подсчет.

Предположим, что двоичное арифметическое устройство мы намерены строить из триггерных регистров, сумматоров и цепей сдвига; элементарными операциями, которые будут выполняться в нем, могут являться суммирование и сдвиг.

В качестве опорного метода выберем первый из основных вариантов выполнения умножения, описанный в 4.1.2 (соответствующее построение арифметического устройства показано на рис. 4-2, *а,* стр. 145). При этом на каждый разряд перемножаемых чисел приходится: по оборудованию — 3 триггера (в регистрах *A*, *B* и С), 1 одноразрядный сумматор, 2 одноразрядные схемы сдвигов (по одной в регистрах *A* и *B),* а по времени $^1/_3$ такта суммирования (при применении наиболее сильных логических методов) и 1 такт сдвига Если речь дальше пойдет об аппаратных методах ускорения 1-го порядка, то выражение для Q_x удобно преобразовать к виду

$$Q_x = \frac{\tau\ N}{\tau_x N_x} = \frac{1}{(l_\Sigma z_\Sigma + l_\text{р} z_\text{р} + l_\text{с} z_\text{с})\,(k_\Sigma y_\Sigma + k_\text{с} y_\text{с})},$$

где l_Σ — необходимое в оцениваемом методе количество одноразрядных сумматоров, приходящихся на один разряд перемножаемых чисел, $l_\text{р}$ — соответственно необходимое количество триггеров, $l_\text{с}$ — необходимое количество одноразрядных схем сдвига; k_Σ и $k_\text{с}$— количество суммирований и количество сдвигов, приходящееся на 1 разряд перемножаемых чисел. Соотношение $z_\Sigma : z_\text{р} : z_\text{с}$ характеризует относительную сложность соответственно одноразрядного сумматора, одного триггера и одноразрядной схемы сдвига, причем коэффициенты z_Σ, $z_\text{р}$ и $z_\text{с}$ пронормированы так, что $z_\Sigma + 3z_\text{р} + 2z_\text{с} = 1$ (исходя из соотношений в количестве оборудования для опорного метода). Коэффициенты y_Σ и $y_\text{с}$ находятся из соотношения $y_\Sigma : y_\text{с} = \tau_\Sigma : \tau_\text{с} = r$; они пронормированы так, что

$$\frac{1}{3} y_\Sigma + y_C = 1, \quad \text{т.е. } y_\Sigma = \frac{r}{\frac{1}{3} r + 1}, \qquad y_c = \frac{1}{\frac{1}{3} r + 1}.$$

При определении коэффициентов z_Σ, $z_\text{р}$ и $z_\text{с}$ должны учитываться стоимость, срок службы, габариты, потребляемая мощность и другие характеристики применяемых элементов. Соотношением этих коэффициентов нужно задаться из тех или иных соображений. Например, при проектировании вычислительной машины М-2 было принято, что одна электронная лампа эквивалентна 20 полупроводниковым диодам, а другие детали схемы — сопротивления, конденсаторы, импульсные трансформаторы и т. д.— в расчете количества оборудования не учитывались. Так как в типовых элементах М-2 один одноразрядный сумматор содержал 3 лампы и 20 диодов, один триггер (вместе с цепями запуска) — 3 лампы и 3 диода, а одноразрядная схема сдвигов — 8 диодов, то

$$z_\Sigma : z_p : z_\text{с} = 80 : 63{:}8,$$

откуда $z_\Sigma = 0{,}281$, $z_\text{р} = 0{,}221$, $z_\text{с} = 0{,}028$ (так что $z_\Sigma + 3z_\text{р} + +2z_\text{с} = 1$). Кроме того, для схем, принятых в М-2, $\tau_\Sigma : \tau_\text{с} = r = 2$, откуда $y_\Sigma = 1{,}2$, $y_\text{с} = 0{,}6$.

Попробуем определить с помощью приведенных соотношений, выгодно ли, например, для ускорения умножения заменить первый вариант его выполнения (см. рис. 4-2, *а)* четвертым вариантом (см. рис. 4-2, *г*). Как известно, в четвертом варианте возможно перекрытие по времени тактов сдвига и тактов суммирования, так что в среднем на каждый разряд перемножаемых чисел приходится $^1/_3$ такта суммирования и $^2/_3$ такта сдвига. Если произведение должно получаться с тем же числом знаков, с каким заданы исходные числа, то применение этого варианта безусловно выгодно, так как сокращение времени достигается почти без дополнительных затрат оборудования. Однако если произведение нужно получать с полным числом знаков, то придется удвоить количество разрядов в сумматоре, количество триггеров в регистрах *B* и *C* и количество одноразрядных схем сдвига в регистре *C* (ср. рис. 4-2, *г* с рис. 4-2,*а*). Таким образом, в этом случае $l_\Sigma = 2$, $l_\text{р} = 5$, $l_\text{с} = 3$; $k_\Sigma = 1/3$, $k_\text{с}=2/3$.

Подставляя эти значения в выражение для Q, найдем

$$Q = \frac{1}{(2 \cdot 0{,}281 + 5 \cdot 0{,}221 + 3 \cdot 0{,}028)\,(0{,}333 \cdot 1{,}2 + 0{,}667 \cdot 0{,}6)} = \frac{1}{1{,}796 \cdot 0{,}8} = 0{,}69.$$

Здесь величина 1,796 характеризует увеличение количества оборудования, а 0,8 — сокращение времени;

эффективность метода при данных соотношениях коэффициентов весьма низкая.

При тех же условиях сравнительно высокой эффективностью обладает, например, метод введения дополнительных цепей сдвига. Как видно из сравнения рис. 4-3 (стр. 165) с рис. 4-2,*а*, добавочное оборудование составляют лишь две одноразрядные схемы сдвига на каждый разряд перемножаемых чисел (по одной в регистрах *A и B*), так что для этого метода $l_\Sigma = 1$, $l_\text{Р} = 3$, $l_\text{с} = 4$; при этом (см. 4.3.1) количество тактов суммирования, приходящееся на 1 рязряд перемножаемых чисел, остается равным $k_\Sigma = 1/3 \approx 0{,}333$, а количество тактов сдвига сокращается до $k_c = \frac{5}{9} \approx 0{,}556$. После подстановки найдем

$$Q = \frac{1}{(1 \cdot 0{,}281 + 3 \cdot 0{,}221 + 4 \cdot 0{,}028)\ (0{,}333 \cdot 1{,}2 + 0{,}556 \cdot 0{,}6)} = \frac{1}{1{,}056 \cdot 0{,}733} = 1{,}3.$$

Аналогичные оценки можно найти и для других способов выполнения умножения. Естественно, что величины Q_x для различных методов очень сильно зависят от принятых предположений о том, какие элементы схемы будут использоваться в устройствах.

Оценка различных методов умножения с помощью величины Q_x не учитывает, насколько полно используется при выполнении других действий оборудование, которое затрачено для выполнения умножения. Фактически вместо величины Q_x, зависящей от $\tau_x N_x$ (τ_x — среднее время выполнения умножения, N_x — количество оборудования, занятого при выполнении умножения), следовало бы пользоваться аналогичной величиной, зависящей от $\tau_0 N_0$, где τ_0 — среднее время выполнения всех операций (с учетом их относительной частоты), а N_0 — полное количество оборудования арифметического устройства.

4.4. Аппаратные методы 1-го порядка ускорения умножения в параллельных устройствах (для систем счисления с основанием n≠2). Предельные оценки

Некоторые из описанных в разделе 4.3 аппаратных методов ускорения умножения в двоичной системе могут быть с равным успехом использованы и при любом другом основании системы счисления *n.* Без каких-нибудь изменений, например, можно использовать метод разделения множителя на части (4.3.1) и метод запоминания переносов (4.3.2). Дополнительные цепи сдвига при $n > 2$ мало эффективны, так как при большом *n* относительно редко получается «0» в очередном разряде преобразованного множителя (т. е. относительно редко встречаются случаи, когда эти цепи полезны).

Рассмотрим, однако, те методы ускорения умножения для систем счисления с основанием $n > 2$, которые содержат принципиально новые идеи по сравнению с методами ускорения в двоичной системе.

4.4.1. Метод предварительной подготовки чисел, кратных множимому.

Представим себе, что наряду с обычным регистром множимого *C* (см., например, рис. 4-1 на стр. 144) в состав устройства входит еще один регистр—*C′*, предназначенный для хранения удвоенного множимого. После того как в начале операции в регистре *C* установлено множимое, а регистр *B* погашен, выполним дважды суммирование *B* + *C* с передачей результата в *B*. В результате в регистре *B* получится удвоенное множимое. Передадим его затем в регистр *C′*, вновь погасим регистр *B* и только потом начнем обычный процесс умножения. При этом, однако, выполняя умножение на какой-либо разряд множителя, мы будем в основном добавлять к сумме частичных произведений или вычитать из нее число, находящееся в регистре *C′*, столько раз, сколько двоек содержится в цифре множителя и только один раз (если цифра множителя нечетна) добавим или вычтем число, находящееся в регистре *C*; количество двоек, содержащееся в некоторой цифре множителя, легко определить, если *n*-ичные цифры кодируются двоичным кодом. В тех случаях, когда в процессе умножения должны выполняться сдвиги в регистре *C,* одновременно нужно будет сдвигать и число в регистре *C′* (второй и четвертый варианты выполнения умножения — см. 4.1.2).

Ясно, что тем же способом, каким в регистре *C′* мы заранее заготовили удвоенное множимое (2*C*), в нем можно было бы получить и любое другое число, кратное множимому — *kC,* где *k* — целое. При кодировании *n*-ичных цифр множителя двоичным кодом удобно, чтобы *k* представляло собой целую степень двойки; однако и применение других *k* не привело бы к значительным усложнениям в схеме управления. Для каждого конкретного основания системы счисления *n* можно подсчитать, при какой величине *k* время умножения получается минимальным. В таблице 4-3 для примера показан такой подсчет для десятичной системы. При этом предполагается, что одновременно с описываемым методом применяется логический метод ускорения, состоящий в переходе к симметричному диапазону цифр (см. 4.2.3), в результате чего цифрами преобразованного множителя являются — 4, —3, —2, —1, 0, 1,2, 3, 4, 5, каждая с вероятностью $^1/_{10}$. Из таблицы видно, что выгоднее всего заготовить в регистре *C′* число 4*C*, что позволяет получить сред нее количество суммирований-вычитаний на разряд равным 1,4 (в этом подсчете, правда, не учтены начальные такты суммирования). Для сравнения напомним, что наиболее сильные логические методы ускорения для десятичной системы дают $^{27}/_{11} \approx 2{,}45$ суммирований-вычитаний на разряд (см. 4.2.4).

Таблица 4-3

Использование чисел, кратных множимому (для десятичной системы счисления)

В регистре C' хранится:	$2C$		$3C$		$4C$	
цифры преобразованного множителя	программа суммирований-вычитаний	количество суммирований-вычитаний	программа суммирований-вычитаний	количество суммирований-вычитаний	программа суммирований-вычитаний	количество суммирований-вычитаний
— 4 — 3 — 2 — 1 0 1 2 3 4 5	—(2c)— (2c) —(2c) — (c) —(2c) —c — +c + (2c) + (2c) + c + (2c) + (2c) +(2c) + + (2c) + c	2 2 1 1 0 1 1 2 2 3	—(3c) — c —(3c) — c— c —c — + c + c+c + (3c) + (3c)+c + (3c) + c + + c	2 1 2 1 0 1 2 1 2 3	—(4c) — (4c) + c — c— c — c — + c + c +c + (4c)—c + (4c) + (4c) + c	1 2 2 1 0 1 2 2 1 2
Среднее количество суммирований-вычитаний на 1 разряд множителя	1,5		1,5		1,4	

Наряду с регистром C' в состав устройства можно ввести также регистр C'', в котором хранилось бы другое кратное множимого. Например, если бы в десятичной системе в регистре C' хранилось удвоенное множимое, а в регистре C''— учетверенное, то в среднем на каждый разряд множителя приходилось бы всего 1,2 суммирования-вычитания. Однако вообще увеличение количества добавочных регистров имело бы смысл при больших основаниях системы счисления n. В троичной системе ($n = 3$) даже и один добавочный регистр C' не может принести пользы.

К рассмотренному методу по идее близок описываемый ниже метод использования особых множителей.

4.4.2. Особые множители.

Особым множителем мы будем называть число, представляющее собой *произведение любых целых степеней простых делителей основания системы счисления n.* Если

$$n = n_1^{r_1} n_2^{r_2} \ldots n_k^{r_k},$$

где $n_1, n_2, \ldots, n_k$ — простые числа, а $r_1, r_2, \ldots, r_k$ — целые положительные числа, то особым множителем является любое число вида

$$q = n_1^{s_1} n_2^{s_2} \ldots n_k^{s_k},$$

где каждый из показателей степени s_i — целое число (положительное, отрицательное или нуль). Например, для десятичной системы $n = 10 = 2^1 \cdot 5^1$; особыми множителями являются числа $2 = 2^1 \cdot 5^0$, $4 = 2^2\, 5^0$, $5 = 2^0\, 5^1$, $8 = 2^3\, 5^0$, $10 = 2^1 \cdot 5^1$, $16 = 2^4 \cdot 5^0$, $20 = 2^2 \cdot 5^1$, $25 = 2^0 \cdot 5^2$,..., числа ${}^1/_2 = 2^{-1} \cdot 5^0$, ${}^1/_4 = 2^{-2}\, 5^0$, ${}^1/_5 = 2^0 \cdot 5^{-1}$,..., а также такие числа, как ${}^2/_5 = 2^1\, 5^{-1}$, ${}^4/_5 = 2^2 \cdot 5^{-1}$, ${}^8/_5 = 2^3\, 5^{-1}$, ${}^5/_2 = 2^{-1} \cdot 5^1$, ${}^5/_4 = 2^{-2}\, 5^1$, ${}^{25}/_8 = 2^{-3} \cdot 5^2$ и др.

Мы докажем сейчас, что при умножении любого числа, представленного позиционной n-ичной системой с естественными весами разрядов и неотрицательными цифрами, на особый множитель $q = n_1^{s_1} n_2^{s_2} \ldots n_k^{s_k}$, каждая цифра результата зависит не более чем от $1 + |s_1| + |s_2| + \ldots + |s_k|$ цифр исходного числа и не зависит от всех других его цифр. При этом на цифру данного разряда результата могут влиять цифры того же разряда исходного числа и цифры $|s_1| + |s_2| + \ldots + |s_k|$соседних разрядов; если все s_i положительны, то речь идет о соседних справа разрядах, если все s_i отрицательны — то соседних слева; если $s_{l_1}, s_{l_2}, \ldots s_{l_v}$ положительны, а $s_{m_1}, s_{m_2}, \ldots, s_{m_u}$ отрицательны, то влияние могут оказать $s_{l_1} + s_{l_2} + \ldots + s_{l_v}$ соседних справа и $|s_{m_1} + s_{m_2} + \ldots + s_{m_u}|$ соседних слева разрядов*).

*) Материал разделов 4.4.2, 4.4.3,4.4.4 был впервые опубликован в докладе Карцева М. А. и Гливенко Е. В., «Дробные основания системы счисления и их использование для ускорения операций в цифровых машинах», Colloquium on the Foundations of Mathematics, Mathematical Machines and their Applications, Будапешт, 1965, стр. 215—226.

Для доказательства рассмотрим сначала случай умножения *n*-ичного числа на особые множители вида $n_1^0\, n_2^0 \ldots n_{i-1}^0 n_i^1 n_{i+1}^0 \ldots n_k^0 = n_i$. Для десятичной системы, например, имеется два особых множителя указанного вида: 2 и 5.

Поскольку, как мы условились, мы пользуемся *n*-ичной системой счисления с неотрицательными цифрами, в любом из разрядов исходного числа цифра *a* может принимать одно из *n* значений: 0, 1, 2, . . ., (*n* — 1).

Рассмотрим прежде всего все возможные произведения вида an_i. Поскольку n_i — простой делитель я, то наверняка $n_i < n$ (тривиальный случай, когда *n* — простое число и его простые делители равны $n_1 = 1$, $n_2 = n$, не принимается во внимание); отсюда видно, что произведение an_i наверняка меньше, чем n^2, и в *n*-ичной системе представляет собой однозначное или двузначное число.

Левая цифра в его изображении показывает, сколько раз *n* содержится в произведении an_i; так как максимальное произведение вида an_i есть $(n — 1)n_i = n\, n_i — n_i$, то левая цифра в любом произведении an_i не превышает $n_i — 1$.

Тем же свойством обладали бы левые цифры произведений an_i и в том случае, если бы n_i не было простым делителем *n* (но $n_i < n$).

Правая цифра произведения an_i есть остаток от деления на *n* числа an_i. Поскольку *a* — целое, а n_i представляет собой делитель *n* (здесь это уже существенно), то среди любых чисел вида an_i может быть только n/n_i чисел, различных между собой по модулю *n*. Все эти n/n_i; различных по модулю я чисел получатся при умножении на n_i первых n/n_i чисел натурального ряда $0, 1, 2, \ldots (\frac{n}{n_i} - 1)$.

Правые цифры указанных произведений равны соответственно 0, n_i, $2n_i$, ..., $n — n_i$. При умножении на n_i следующего числа из натурального ряда (числа n/n_i) получим в качестве произведения число *n*, для которого правая цифра *n*-ичного представления есть снова 0; при умножении следующего числа $\left(\frac{n}{n_i} + 1\right)$ получим $(n+n_i)$, где правая цифра в *n*-ичном представлении есть опять n_i, и т. д.

Итак, при умножении любой *n*-ичной цифры *a* на особый множитель n_i получаем одно- или двухзначное число, в котором левая цифра не превышает $n_i — 1$, а правая цифра не превышает $n — n_i$. Например, при умножении десятичных цифр 0,1, 2, . . ., 9 на 2 левая цифра произведения не превышает 1, а правая цифра не превышает 8; при умножении тех же цифр на 5 левая цифра произведения не превышает 4, а правая цифра не превышает 5.

Пусть теперь на n_i умножается многоразрядное *n*-ичное число *A*. Умножив каждую из его цифр на n_i, выпишем отдельно число, состоящее из правых цифр этих произведений, и отдельно число, состоящее из левых цифр произведений; в первом из них каждый разряд определяется соответствующим разрядом исходного числа, во втором — соседним справа разрядом исходного числа. Для того чтобы получить полное произведение An_i, эти два числа нужно между собой сложить. При этом, поскольку любая из цифр первого числа не превышает $n — n_i$, а любая из цифр второго числа не превышает $n — 1$, сумма цифр в любом разряде не превышает $n — 1$, т. е. сложение выполняется без переносов из разряда в разряд. Следовательно, любая цифра в произведении $A\, n_i$ зависит только от цифры соответствующего разряда и цифры одного соседнего справа разряда исходного числа, но не зависит от цифр всех остальных его разрядов. Пусть, например, десятичное число 875 039 466 умножается на 2:

	8	7	5	0	3	9	4	6	6
									×2
(произведения отдельных цифр на 2)	16	14	10	00	06	18	08	12	12
(число, составленное из правых цифр)	6	4	0	0	6	8	8	2	2
(число, составленное из левых цифр)	11	1	0	0	1	0	1	1	
(полное произведение)	17	5	0	0	7	8	9	3	2

Как бы мы ни изменили любую цифру в исходном числе, в конечном результате может измениться не более двух цифр. Иное положение при умножении десятичных чисел, скажем, на число 3, не являющееся особым множителем. Например

333 333 × 3 = 999 999, а 333 334 × 3 = 1 000 002;

изменение одной младшей цифры в исходном числе привело к изменению всех цифр произведения.

Точно так же можно доказать, что в произведении произвольного *n*-ичного числа на n_i^{-1} цифра любого разряда зависит только от цифры данного разряда и цифры соседнего слева разряда исходного числа.

Действительно. Умножение числа на $n^{-1}{}_i$ эквивалентно его делению на n_i. Предположим, что деление выполняется обычным «школьным» методом — углом. Обозначим цифру некоторого разряда исходного числа через *a,* а остаток от деления на n_i предыдущих разрядов через *r.* Отыскивая очередную цифру частного, мы должны выполнить деление на n_i числа $r\, n + a$. Очередная цифра частного, которую мы при этом получим, зависит, конечно, и от величины предыдущего остатка *r* и от очередной цифры делимого *a*; однако новый остаток r_1

зависит только от a; и не зависит от r, так как, каково бы ни было r, число $r\,n$ делится на n_i нацело (потому что n_i есть делитель числа n). Таким образом, каждая цифра частного зависит только от цифры того же разряда и цифры одного соседнего слева разряда делимого и не зависит от всех остальных его цифр.

Теперь уже нетрудно доказать сформулированное в начале раздела положение и для произвольного особого множителя $q = n_1^{s1}\, n_2^{s2} \ldots n_k^{sk}$. Произведение любого n-ичного числа A на q можно рассматривать как результат s_i последовательных умножений на n_1 (или, если s_1 отрицательно, $|\, s_1\,|$ последовательных делений на n_1), затем s_2 последовательных умножений на n_2 (или может быть, $|\, s_2\,|$ последовательных делений на n_2, если $s_2 < 0$) и т. д. Отсюда и вытекают приведенные выше утверждения.

Указанные свойства особых множителей позволяют получить произведение любого числа на особый множитель сравнительно простыми средствами.

Представим себе, например, что основание системы счисления четно — скажем, $n = 10$. Число 2 является особым множителем, причем таким, что в произведении любого числа на 2 цифра некоторого разряда зависит только от цифры данного разряда и цифры одного соседнего справа разряда исходного числа. Пусть исходное число помещается в триггерный регистр. Присоединив к выходам каждой пары разрядов регистра (1-го и 2-го, 2-го и 3-го, 3-го и 4-го и т. д.) некоторую специальным образом построенную логическую схему, можно на выходах этих схем получить произведение исходного числа, помещенного в регистр, на 2. При этом логические схемы в каждой паре разрядов одинаковы между собой, суммарное количество оборудования пропорционально общему количеству разрядов в регистре, а образование удвоенного числа может происходить в течение одного такта (потому что все логические схемы в каждой из пар разрядов срабатывают одновременно — независимо одна от другой).

Аналогичным образом легко устроить цепи, формирующие произведение числа, находящегося в регистре, на 4, на 5, на 8 или вообще на любой особый множитель.

Ясно, что столь простыми средствами нельзя сформировать произведение десятичного числа, скажем, на 3, так как 3 не является особым множителем. Для получения утроенного числа нужно было бы либо усложнять логические цепи от разряда к разряду (потому что каждый разряд в произведении некоторого числа на 3 зависит вообще от всех младших разрядов исходного числа), либо строить цепи, включающие сумматор или аналогичные элементы, с переносами из разряда в разряд (например, сумматор мог бы складывать удвоенное исходное число с самим исходным числом, что давало бы произведение на 3); в первом случае суммарное количество оборудования оказалось бы пропорциональным квадрату количества разрядов в исходном числе, во втором случае формирование утроенного числа не укладывалось бы по времени в один такт.

Особые множители можно использовать аналогично тому, как использовались подготовленные заранее числа, кратные множимому (см. 4.4.1). Вместо того чтобы в схеме устройства наряду с регистром множимого C предусматривать дополнительно, скажем, регистры C' и C'' для хранения удвоенного и учетверенного множимого (полученных заранее с помощью сумматора), можно присоединить к регистру C цепи удвоения и учетверения и предусмотреть возможность передачи на сумматор чисел с их выходов. Эффект при этом получится точно такой же, как при наличии регистров C' и C'', а в схемном отношении такое решение может оказаться более простым. Оно возможно, однако, только при условии, что основание системы счисления четно (например, $n = 10$), вследствие чего 2 и 4 являются особыми множителями; заготовить заранее с помощью сумматора удвоенное и учетверенное множимое в регистрах C' и C'' можно было при любом n.

4.4.3. Понижение основания системы счисления.

В системах счисления, для которых основание n не является простым числом, возможен еще один метод ускорения умножения, использующий особые множители. Мы назовем его методом *понижения основания системы счисления.* Хотя один из частных случаев этого метода известен по литературе давно (под названием «метода удвоения и деления пополам»), в реальных устройствах он, по-видимому, не применялся. Однако с теоретической точки зрения этот метод представляет значительный интерес. Впрочем, не видно причин, почему бы не использовать его и на практике.

Идея метода состоит в следующем.

Представим себе для определенности, что речь идет об умножении десятичных чисел, изображенных позиционным способом с естественными весами разрядов и с запятой, фиксированной перед старшим разрядом числа, и что умножение предполагается выполнять в соответствии с четвертым вариантом основного алгоритма (см. 4.1.2); в связи с этим общее построение множительного устройства соответствует рис. 4-2, *г* на стр. 145. Однако и в способ выполнения умножения, и в общее построение устройства мы внесем некоторые изменения.

Изменения в построении множительного устройства сведутся к тому, что в регистре множимого (C) цепи сдвига вправо на один десятичный разряд мы заменим цепями деления пополам, а в регистре множителя (A) цепи сдвига влево на один десятичный разряд заменим цепями удвоения (рис. 4-8). Цепи эти устроим так, чтобы каждый раз, как на регистр A, скажем, подается команда «удвоение», число с выходов цепей удвоения возвращалось бы в регистр A, заменяя в нем исходное число; то же должно происходить и в регистре C при делении пополам.

Изменения в порядке выполнения операции сводятся к следующему. Нормально каждый цикл умножения

состоит из сдвига влево в регистре A, сдвига вправо в регистре C (т. е. фактически из умножения A на 10 и деления C на 10) и суммирований; количество суммирований соответствует количеству единиц в той цифре, которая выходит за пределы регистра A при сдвиге влево. Теперь вместо сдвигов в каждый цикл войдет удвоение числа в регистре A и деление пополам числа в регистре C; вслед за тактом удвоения и деления пополам либо выполняется одно суммирование, если при удвоении числа за пределы регистра A выходит единица, либо суммирование не производится, если за пределы регистра A выходит нуль.

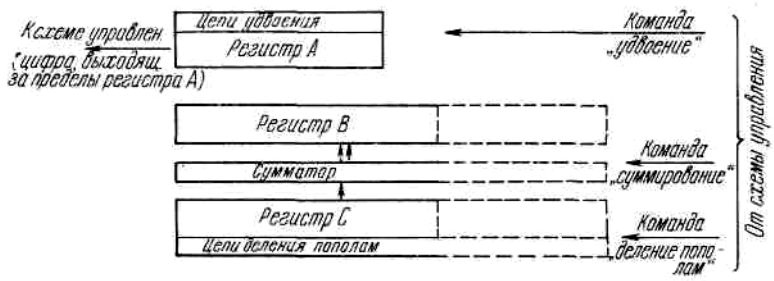


Рис. 4-8. Построение арифметического устройства при выполнении умножения методом удвоения и деления пополам.

Для того чтобы понять, каким образом при этом происходит умножение, нужно прежде всего разобраться, что делается в регистре множителя A, когда в нем производятся указанные операции. Как мы сейчас увидим, в описанном процессе умножения по ходу дела фактически выполняется преобразование множителя в двоичную систему.

Действительно. Когда в первом цикле умножения мы производим удвоение множителя Л и фиксируем цифру, которая получается слева от запятой (за пределами регистра A), то тем самым проверяется, больше ли множитель, чем $^1/_2$, или меньше; если множитель больше или равен $^1/_2$, то после его удвоения в разряде целых получим 1, если множитель меньше $^1/_2$, то после его удвоения цифра в разряде целых есть 0. Цифра, которая оказывается за пределами регистра A после первого удвоения множителя (1 или 0), является старшей цифрой в двоичном представлении множителя; она относится к разряду с весом $^1/_2$. В регистре A при этом остается величина 2 $(A - a_1)$, где A — исходное значение множителя, a_1 — число, представленное его первой двоичной цифрой (если первая двоичная цифра есть 1, то $a_1 = {}^1/_2$, если первая двоичная цифра есть 0, то $a_1 = 0$).

Для того чтобы найти следующую двоичную цифру (относящуюся к разряду с весом $^1/_4$), нужно узнать, больше ли величина $A - a_1$ чем $^1/_4$ или меньше. С этой целью мы производим новое удвоение в регистре A, т. е. находим величину A $(A - a_1)$ и проверяем, получилась ли эта величина больше единицы. Если при втором удвоении за пределы регистра A вышла единица, то это значит, что 4 $(A - - a_1) \geq 1$ (т. е. $A - a_1 \geq {}^1/_4$), и, следовательно, вторая цифра в двоичном представлении множителя есть 1; если после второго удвоения в разряде целых оказывается нуль, то это значит, что $4(A - a_1) < 1$ (т. е. $A - a_1 < {}^1/_4$), и вторая цифра в двоичном представлении множителя есть 0. В регистре A при этом остается величина 4 $(A - a_2)$, где a_2 — число, представленное п е р в ы м и д в у м я двоичными цифрами. Удваивая затем эту величину, найдем третью двоичную цифру (относящуюся к разряду с весом $^1/_8$) и т. д.

Итак, в процессе умножения шаг за шагом осуществляется перевод множителя в двоичную систему, причем очередные двоичные цифры в каждом цикле получаются слева от запятой в регистре A.

В тот момент, когда за пределами регистра A находится первая цифра двоичного представления множителя (относящаяся к разряду с весом $^1/_2$), в регистре множимого C находится результат первого деления множимого пополам— величина $^1/_2$ C. Если первая двоичная цифра множителя есть 1, то величина $^1/_2$ C добавляется к регистру B, если 0 — не добавляется. Во втором цикле умножения, когда за пределами регистра A находится вторая двоичная цифра множителя (относящаяся к разряду с весом $^1/_4$), в регистре множимого находится результат второго деления пополам — величина $^1/_4$ C; в зависимости от второй двоичной цифры множителя она либо добавляется, либо не добавляется к содержимому регистра B и т. д. Таким образом, умножение выполняется фактически как бы в двоичной системе счисления; однако при этом сами суммирования производятся в десятичной системе, и результат в регистре C получается десятичным.

П р и м е р . Пусть требуется произвести умножение 0,738 × 0,476.

Точный результат должен быть равен 0,351288. Однако в дальнейшем мы получим его с некоторой погрешностью. В принципе некоторая ошибка могла бы возникнуть в результате округления множимого (так как количество разрядов в регистре множимого ограничено и при делении множимого пополам за пределы регистра уходят вправо младшие цифры — ср. с 4.1.4). Однако этой ошибки можно избежать, выписывая каждый раз все цифры очередного деления множимого пополам и все цифры суммы частичных произведений (в действительности, конечно, все цифры сохранять не нужно); цифры, находящиеся за пределами регистров, мы будем записывать в круглых скобках. Дополнительная погрешность получается за счет того, что множитель не переводится в двоичную систему точно. Трем десятичным знакам множителя должны соответствовать по точности 10—11 двоичных разрядов, т. е. в нашем случае 10—11 циклов умножения; младшими двоичными разрядами мы просто

пренебрежем, оборвав умножение на 11-м цикле.

Итак, начальное расположение чисел:

множитель *(A)* .476
множимое (C) .738
(*B*) .000.

В первом цикле удваиваем число в регистре *A* и делим пополам число в регистре C. Поскольку за пределы регистра *A* влево выходит нуль (первая цифра в двоичном представлении множителя есть 0), суммирование в первом цикле не производится, и в регистре *B* сохраняются нули. После первого цикла получим

(*A*) (0).952
(*C*) .369
(*B*) .000.

Во втором цикле удвоение числа, находящегося в регистре *A*, дает единицу слева от запятой; поэтому выполняется суммирование *B*. В результате получим

(*A*) (1).904
(*C*) .184(5)
[+ .000]
(*B*) .184(5).

В третьем цикле удвоение числа, представленного основными разрядами регистра *A*, вновь дает единицу слева от запятой, вследствие чего вновь выполняется суммирование *B* + *C*. После третьего цикла получаем

(*A*) (1).808
(C) .092 (25)
[+ .184(5)]
(B) .276(75).

Аналогично поступаем и в следующих циклах. При этом получается:

после четвертого цикла

(*A*) (1).616
(*C*) .046(1 2 5)
[+ .276(7 5)]
(*B*) .322(8 7 5),

после пятого цикла

(*A*) (1).232
(C) .023 (0625)
[+ .322(875)]
(*B*) .345(9375),

после шестого цикла

(*A*) (0).464
(*C*) .011(53125)
(*B*) .345(9375),

после седьмого цикла

(*A*) (0).928
(*C*) .005(765625)
(*B*) .345(9375) ,

после восьмого цикла

(*A*) (1).856
(*C*) .002(8828125)
[+.345(9375)]
(*B*) .348(8203125),

после девятого цикла

(*A*)(1).712
(*C*) .001(44140625)
[+ .348(8203125)]
(*B*) .350(26171875),

после десятого цикла

(*A*)(1).424
(*C*) .000(720703125)
[+ .350 (26171875)]

(*B*) .350(982421875),

после одиннадцатого цикла

(*A*) (0).848
(C) .000(3603515625)

(*B*) .350(982421875).

Если произведение должно быть получено с той же точностью, что и исходные числа, с тремя десятичными знаками, то продолжать дальше процесс умножения не имеет смысла: в основных разрядах регистра *C* остались нули; величина, находящаяся за пределами регистра (сохраненная, может быть, частично в дополнительных разрядах округления), тоже достаточно мала. Произведя округление в третьем десятичном знаке, получим в регистре *B* результат с необходимой точностью (0,351).

В течение всего процесса умножения нам пришлось выполнить здесь всего 7 суммирований. Если бы умножение на 0,476 выполнялось по десятичной системе обычным способом (с пропуском пустых тактов суммирования), то потребовалось бы 4 + 7 + 6 = 17 тактов суммирования, а если бы мы перешли к симметричному диапазону цифр (см. 4.2.3), то количество необходимых тактов суммирования вычитания было бы равно 11 (потому что $0{,}476 = 0{,}52\overline{4}$; при этом $5 + |\overline{2}| + |\overline{4}| = 11$). Вообще же при использовании указанного метода справедливы все те соотношения для количества тактов суммирования-вычитания, которые приводились нами для двоичной системы, и могут быть использованы любые описанные ранее методы ускорения умножения в двоичной системе.

Если при переходе к двоичной системе никакие логические методы ускорения (кроме пропуска «пустых» тактов суммирования) не применяются, то на 10—11 двоичных разрядов в среднем для различных чисел придется 5—$5^1/_2$ суммирований. При умножении обычным способом на 3-разрядное десятичное число при прочих равных условиях потребовалось бы в среднем $13^1/_2$ суммирований.

Ясно, что аналогичным образом умножение в *n*-ичной системе можно свести к умножению в системе счисления с любым основанием n_0 при условии, что n_0 является особым множителем для данного *n.* Например, умножение в десятичной системе *(n* = 10) можно свести к умножению по четверичной системе, пятеричной системе, восьмеричной и т. д. (потому что при *n* — 10 числа 4, 5, 8 и т. д. являются особыми множителями).

На первый взгляд может показаться, что если умножение в *n*-ичной системе удается свести к двоичному умножению, то ни о каких других основаниях уже не имеет смысла говорить: в разделах 1.2.3 и 4.2.4 было доказано, что с точки зрения скорости выполнения умножения двоичная система является наиболее выгодной. В действительности, однако, этот вывод справедлив только для целых оснований системы счисления *n.* Как функция $f_0(n)$, так и функция $f_\infty(n)$, которыми мы пользовались ранее для оценки скорости выполнения умножения, монотонно убывают с уменьшением *n.* Для целочисленных *n* эти функции имеют наименьшие значения при *n* — 2; но брать $n < 2$ мы не пытались. Мы покажем сейчас, как можно использовать системы счисления с дробными основаниями; при этом очевидно, наибольший интерес будут представлять случаи, когда $1 < n < 2$.

4.4.4. Применение дробных оснований систем счисления. Предельная оценка для методов 1-го порядка.

Допустим, что в исходном представлении чисел основание системы счисления *n (n* — целое, $n > 1$) не является простым числом, или целой степенью простого числа. При этом всегда имеется ряд особых множителей, находящихся в интервале от 1 до 2. Например, в десятичной системе имеются особые множители 1,25 (или $2^{-2}\,5^1$); 1,6 (или $2^3 \cdot 5^{-1}$) и др.

С целью упрощения дальнейшего изложения будем вести его применительно к десятичной системе, а в качестве особого множителя, заключенного в интервале от 1 до 2, выберем для определенности множитель 1,6.

Заменим в схеме десятичного множительного устройства, изображенной на рис. 4-8 (стр. 175), цепи удвоения в регистре *A* цепями умножения на 1,6 , а цепи деления пополам в регистре *C* — цепями деления на 1,6.

Далее будем действовать в точности по аналогии с разделом 4.4.3. Произведя первое умножение на 1,6в регистре *A*, мы узнаем, превосходит ли исходный множитель величину $(1{,}6)^{-1}$, т. е. найдем первый коэффициент в разложении множителя по степеням числа 1,6. В зависимости от того, каков этот коэффициент — 1 или 0,— мы либо произведем суммирование *B* + *C,* либо не произведем (в регистре *C* в это время находится результат первого деления множимого на 1,6, т. е. число $(1{,}6)^{-1}$ *C*). Далее вновь произведем умножение на 1,6 в регистре *A* и деление на 1,6 в регистре *C*; разряд целых регистра *A* даст при этом второй коэффициент в разложении множителя по степеням числа 1,6 (при члене $(1{,}6)^{-2}$), а в регистре *C* получится произведение множимого *C* на $(1{,}6)^{-2}$, которое либо будет суммироваться с регистром *B,* либо не будет и т. д.

Количество циклов умножения $m_{(1,6)}$, которые нам придется выполнить, определяется соотношением

$$m_{(1,6)} = m_{(10)}/\lg 1{,}6,$$

где $m_{(10)}$ — количество десятичных разрядов в множителе. Действительно. Отыскивая коэффициенты полинома по степеням 1,6, мы каждый раз умножаем на 1,6 число, оставшееся в регистре A от предыдущего цикла. Непосредственно по смыслу выполняемых преобразований очевидно, что величина, остающаяся в регистре A после $m_{(1,6)}$-го цикла, заключена в интервале от 0 до 1 (потому что в регистре A остается дробная часть предыдущего результата) и является последним остатком, умноженным на $1{,}6^{(m_{1,6})}$ (так как в каждом из предыдущих циклов очередной остаток умножался на 1,6). Таким образом, погрешность представления числа с помощью $m_{(1,6)}$ членов полинома по основанию 1,6 меньше, чем $1{,}6^{-(m_{1,6})}$. Для получения той же точности, что и при изображении числа десятичными разрядами, необходимо, чтобы

$$1{,}6^{-m_{(1,6)}} = 10^{-m_{(10)}},$$

откуда и следует приведенное выше соотношение для $m_{(1,6)}$. Например, точности в три десятичных разряда соответствует, примерно 15 членов в разложении по степеням числа 1,6; следовательно, количество циклов умножения в рассматриваемом выше примере должно было бы стать примерно равным 15.

Труднее подсчитать, каково в среднем количество суммирований, которые придется при этом выполнить. Хотя каждый из коэффициентов в разложении по степеням 1,6 может принимать одно из двух возможных значений — 1 или 0 (так же, как в двоичной системе), цифры эти — в отличие от двоичной системы — не равновероятны.

Предположим, что исходное значение множителя может быть с равной вероятностью любым числом из интервала 0 ÷ 1. Тогда вероятность равенства единице первого коэффициента в разложении по степеням 1,6 есть 0,375: для того чтобы результат первого умножения множителя на 1,6 был больше или равен единице, исходная величина множителя должна быть заключена в интервале 0,625 ÷ 1; если исходная величина множителя заключена в интервале от 0 до 0,625, то в результате умножения на 1,6 получим число меньше единицы.

Если первый коэффициент оказался равным 0, то в регистре A после первого умножения на 1,6 с равной вероятностью может оказаться любое число из интервала 0 ÷ 1; если же первый коэффициент оказался равным 1, то в регистре A остается одно из чисел из интервала 0 ÷ 0,6 (A·1,6 — 1, где A —исходное число, $0{,}625 \le A < 1$). Поэтому вероятность равенства единице второго коэффициента равна произведению 0,375 на вероятность того, что первый коэффициент равен нулю, т. е. 0,375 × 0,625 = = 0,234.

Число, остающееся в регистре A после второго умножения на 1,6, с вероятностью $0{,}625^2$ заключено в интервале 0 ÷ 1, с вероятностью 0,375—в интервале от 0 до 0,6 × ×1,6 = 0,96, с вероятностью 0,625·0,375—в интервале 0 ÷ 0,6. Поэтому вероятность получить единицу после третьего умножения на 1,6 равна

$$0{,}625^2 \cdot 0{,}375 + 0{,}375 \cdot \frac{0{,}96 - 0{,}625}{0{,}96} \approx 0{,}278.$$

Рассуждая аналогичным образом, найдем, что вероятность равенства единице четвертого коэффициента в разложении по степеням 1,6 есть 0,264, пятого коэффициента— 0,254 и т. д.

Полагая — с некоторым запасом,— что вероятность равенства единице каждого из коэффициентов в разложении по степеням 1,6 не превышает в среднем 0,3, найдем, что при умножении, скажем, 3-значных десятичных чисел указанным способом среднее количество суммирований не превысит 0,3·15, т. е. 4,5 (15 — это количество членов разложения по степеням 1,6). Для сравнения напомним, что при переходе к двоичной системе среднее количество суммирований для этого случая оказалось равным 5 ÷ 5,5, а в десятичной системе оно было еще больше.

Читателю предлагается самостоятельно повторить данным методом пример, выполненный в 4.4.3 методом удвоений и делений пополам: умножить 0,738 × 0,476, пользуясь умножениями и делениями на 1,6; количество суммирований, которые придется при этом произвести за 15 циклов умножения, окажется равным всего 5 (против 7 в том же примере при использовании метода удвоения и деления пополам).

По сути дела, в данном разделе мы не предложили никакого нового метода по сравнению с 4.4.3, но понижение основания системы счисления стали производить не к целому n, $n \ge 2$, а к дробному n, $1 < n < 2$. При этом об основании системы счисления мы говорим теперь в смысле второго определения, приведенного в 1.2.1 (стр. 9), которым мы до сих пор на протяжении почти всей книги пользовались мало.

Однако приведенное здесь обобщение метода понижения основания системы счисления имеет принципиальное значение, так как позволяет оценить предельные возможности аппаратных методов 1-го порядка в части уменьшения количества тактов суммирования, необходимых в процессе выполнения умножения.

Как мы увидим из дальнейшего, для каждой заданной точности представления исходных чисел имеется некоторое оптимальное значение основания системы счисления n, т. е. такое основание, при переходе к которому количество суммирований, необходимых в ходе выполнения умножения, минимально.

Пусть точность операций задана количеством десятичных разрядов $m_{(10)}$. Для любого n необходимое количество разрядов $m_{(n)}$ определятся аналогично предыдущему по соотношению

$$m_{(n)} = \frac{m_{(10)}}{\lg n}.$$

Введем для данного n, $1 < n < 2$, еще одну численную характеристику k_n определяемую по соотношению

$$n^{k_n}(n-1) = 1$$

или

$$k_n = \frac{-\lg(n-1)}{\lg n}.$$

По смыслу k_n есть минимальное расстояние между двумя n -ичными разрядами, которые могут содержать единицы в изображении любого числа A. Это следует непосредственно из рассмотрения алгоритма перевода некоторого числа в n-ичную систему счисления. В самом деле, к началу любого цикла перевода (преобразования множителя A в ходе умножения в n-ичную систему) остаток от предыдущего цикла, умноженный соответствующее количество раз на n, может оказаться по величине заключенным в интервале $0 \div 1$. Допустим, что в данном i-м цикле очередная n-ичная цифра получается равной единице, т. е. что произведение предыдущего остатка на n в данном случае получается больше или равным единице. Но произведение предыдущего остатка на « не может быть больше, чем « (так как предыдущий остаток не больше единицы). Очередным остатком будет при этом дробная часть указанного произведения, т.е. величина, не превышающая n—1. В следующем i+1)-м цикле этот остаток умножается на n. Если $n\ (n - 1) < 1$, то следующая n-ичная цифра будет непременно нулем. Затем в $(i + 2)$-м цикле величина $n(n - 1)$ умножается на n; если $n^2\ (n - 1) < 1$, то и $(i + 2)$-я цифра будет непременно нулем. Следующая единица может получиться, очевидно, не раньше $(i + k)$ -го цикла, где

$$k > k_n = \frac{-\lg(n-1)}{\lg n}.$$

Интересно подсчитать, при каких значениях n получаются целые значения k_n. Непосредственно из выражения для k_n можно найти:

$k_n = 1$	при	$n \approx 1{,}62$,	$k_n = 4$	при	$n \approx 1{,}32$,
$k_n = 2$	при	$n \approx 1{,}47$,	$k_n = 5$	при	$n \approx 1{,}28$,
$k_n = 3$	при	$n \approx 1{,}38$,	$k_n = 6$	при	$n \approx 1{,}25$

(см. также таблицу 4-4).

Это значит, что, когда основание системы счисления n заключено в интервале от 1,62 до 2, в n-ичном изображении любого числа две соседние цифры могут быть единицами; когда $1{,}47 < n \leq 1{,}62$, между двумя единицами в n-ичном представлений любого числа должен находиться по меньшей мере один разряд, содержащий «0»; когда $1{,}38 < n \leq 1{,}47$, между двумя единицами в n-ичном представлении любого числа находятся по меньшей мере 2 разряда, содержащие нули, и т. д.

Сравнивая выражения для $m_{(n)}$ и для $k_{(n)}$, нетрудно убедиться, что, каково бы ни было число $m_{(10)}$, всегда можно найти такое значение n, достаточно близкое к единице, чтобы выполнялось соотношение

$$m_{(n)} \leq k_n.$$

Это означает, что при заданной точности $m_{(10)}$ в n-ичном изображении любого числа будет иметься не более одного разряда, содержащего цифру «1», а все остальные ($m_{(n)}$ — 1) разрядов непременно содержат нули. Фактически при этом в изображении числа «0» все n-ичные цифры будут нулями, а в изображении любого другого числа будет содержаться одна единица в каком-то из $m_{(n)}$ разрядов. При умножении в n-ичной системе счисления максимальное количество суммирований, необходимых в ходе выполнения умножения, будет при этом равно единице. Среднее количество суммирований, приходящееся на одно умножение, тоже равно примерно единице, поскольку только при умножении на нуль не требуется ни одного суммирования.

Ясно, что ни при каком значении n количество суммирований не может быть меньше единицы. Поэтому с точки зрения количества суммирований, необходимых в ходе выполнения умножения, основание системы счисления n, удовлетворяющее условию

$$m_{(n)} \leq k_n,$$

является оптимальным. Иначе говоря, при заданной точности (при заданном количестве десятичных разрядов $m_{(10)}$) оптимальное основание системы счисления n_{opt} находится по соотношению

$$m_{(10)} \leq\!— \lg (n_{\text{opt}} — 1)$$

или

$$n_{\text{opt}} \le 1 + 10^{-m_{(10)}}.$$

Принципиальное значение полученного результата состоит в следующем. Если n_{opt} является особым множителем для некоторой исходной системы счисления с основанием n, то умножение в системе счисления с основанием я можно свести к умножению в системе счисления с основанием n_{opt}, причем количество оборудования для этого будет пропорционально количеству разрядов в исходной системе счисления. Таким образом, *применение аппаратных методов* 1-*го порядка для ускорения умножения позволяет в принципе уменьшить до единицы среднее и максимальное количество суммирований, необходимых в ходе выполнения умножения—при любом количестве разрядов в исходных числах',* это является вообще *предельным возможным сокращением количества тактов суммирования,* необходимых для выполнения умножения.

При этом не следует забывать, что, отыскивая оптимальное значение основания системы счисления, мы принимали во внимание только количество тактов суммирования, необходимых в ходе выполнения умножения, но совсем не считали, сколько тактов сдвига (умножения-деления на n) при этом потребуется. Между тем для реальных точностей величина $n = n_{\text{opt}}$ получается очень близкой к единице и количество n-ичных разрядов $m_{(n)}$ непомерно велико.

Количество тактов сдвига (умножения-деления на n) в ходе выполнения умножения в n-ичной системе равно вообще $m_{(n)}$. Исключение составляет только $n = n_{\text{opt}}$ при использовании которого умножение можно прервать сразу вслед за тем, как будет произведено одно суммирование (потому что заранее известно, что изображение множителя при этом содержит только одну единицу); количество тактов сдвига в этом случае в среднем равно $\frac{m_{(n)}}{2}$. Но и эта величина практически слишком велика. Оптимальное значение n с точки зрения общего времени выполнения умножения зависит на самом деле от соотношения длительностей такта суммирования и такта сдвига.

Не пытаясь решать задачу в общем виде, приведем некоторые -характеристики различных оснований системы счисления n из интервала $1 < n < 2$, вычисленные с помощью вычислительной машины. Результаты вычислений сведены в таблицу 4-4, в которой обозначения n, $m_{(n)}$, k_n прежние, вместо величины $m_{(10)}$ используется величина m_2 (количество двоичных разрядов, соответствующее заданной точности), а через $P(n, m_2)$ обозначено математическое ожидание количества тактов суммирования, необходимых в ходе выполнения умножения в системе счисления с основанием n при точности, соответствующей m_2 двоичным разрядам. Величины n приведены в таблице с четырьмя десятичными знаками после запятой, величины $m_{(n)}$ и $P(n, m_2)$ —с двумя знаками; с ростом m_2 зависимости m_n и $P(n, m_2)$ от m_2 приближаются к линейным.

Таблица 4-4

n	1,6177		1,4651		1,3794		1,3236		1,2840		1,2535		1,2392	
k_n	1		2		3		4		5		6		7	
m_2	$m_{(n)}$	$P(n,m_2)$	$m_{(n)}$	$P(n,m_2)$	$m_{(n)}$	$P(n,m_2)$	$m_{(n)}$	$P(n,m_2)$	$m_{(n)}$	$P(n,m_2)$	$m_{(n)}$	$P(n,m_2)$	$m_{(n)}$	$P(n,m_2)$
1	1,4	0,5	1,8	0,5	2,2	0,5	2,5	0,5	2,8	0,5	3,1	0,5	3,3	0,5
2	2,9	0,9	3,6	0,9	4,3	0,8	4,9	0,7	5,5	0,7	6,1	0,7	6,7	0,7
3	5,4	1,2	6,5	1,1	7,4	1,1	8,3	1,1	9,2	1,0	10	1,0	10	0,9
4	5,7	1,7	7,3	1,6	8,6	1,5	9,9	1,4	11	1,4	12	1,4	13	1,3
5	7,2	2,3	9,0	2,0	10	1,8	12	1,7	14	1,6	15	1,6	16	1,6
6	8,6	2,5	10	2,2	13	2,1	15	2,0	17	1,9	18	1,9	20	1,9
7	10	3,1	13	2,6	15	2,5	17	2,4	19	2,3	21	2,2	23	2,1
8	11	3,4	14	3,0	17	2,8	20	2,6	22	2,6	24	2,4	26	2,4
9	13	3,6	16	3,4	19	3,1	22	3,0	25	2,8	27	2,7	30	2,7
10	14	4,2	18	3,8	21	3,5	24	3,2	27	3,1	30	3,0	33	2,9
11	16	4,5	20	4,0	24	3,7	27	3,6	30	3,4	34	3,3	36	3,1
12	17	5,0	21	4,4	26	4,0	29	3,8	33	3,7	36	3,5	40	3,5
13	19	5,3	24	4,8	28	4,5	32	4,3	36	4,0	40	3,8	43	3,7
14	20	5,9	25	5,2	30	4,8	35	4,5	39	4,3	43	4,1	46	4,0
15	22	6,1	27	5,5	32	5,1	37	4,8	42	4,6	46	4,5	50	4,3
16	23	6,7	29	5,9	34	5,4	39	5,1	44	4,9	49	4,6	53	4,5
17	24	7,0	30	6,1	36	5,7	42	5,5	47	5,2	52	5,0	57	4,8

Таблица 4-4 (продолжение)

n	1,2107		1,1952		1,1815		1,1697		1,1594		1,1512		1,1432		1,1360	
k_n	8		9		10		11		12		1		14		15	
m_2	$m(n)$	$P(n,m_2)$	$m(n)$	$P(n,m_2)$	$m(n)$	$P(n,m_2)$	$m(n)$	$P(n,m_2)$	$m(n)$	$P(n,m_2)$	$m(n)$	$P(n,m_2)$	$m(n)$	$P(n,m_2)$	$m(n)$	$P(n,m_2)$
1	3,6	0,5	3,9	0,5	4,2	0,5	4,4	0,5	4,7	0,5	4,9	0,5	5,2	0,5	5,4	0,5
2	7,3	0,7	7,8	0,7	8,3	0,7	8,8	0,7	9,4	0,7	9,8	0,7	10	0,7	11	0,7
3	11	0,9	12	0,9	13	0,9	14	0,9	14	0,9	16	0,9	16	0,9	17	0,9
4	14	1,3	15	1,3	16	1,2	17	1,2	18	1,2	20	1,2	21	1,1	22	1,1
5	18	1,6	19	1,5	20	1,5	22	1,5	23	1,5	25	1,4	26	1,4	27	1,4
6	21	1,8	23	1,8	25	1,7	26	1,7	28	1,7	29	1,7	31	1,7	32	1,6
7	25	2,1	27	2,0	29	2,0	30	1,9	33	1,9	34	1,9	36	1,9	38	1,9
8	29	2,4	31	2,3	33	2,3	35	2,2	37	2,2	39	2,1	41	2,1	43	2,1
9	33	2,6	35	2,5	37	2,5	39	2,4	42	2,4	44	2,4	46	2,4	49	2,3
10	36	2,9	38	2,7	41	2,7	44	2,7	46	2,6	49	2,6	51	2,5	54	2,5
11	39	3,1	43	3,0	46	3,0	49	3,0	51	2,9	54	2,9	57	2,8	59	2,8
12	44	3,4	47	3,3	49	3,2	53	3,2	56	3,1	59	3,1	62	3,0	65	3,0
13	47	3,7	50	3,6	54	3,5	57	3,4	61	3,3	64	3,3	67	3,2	70	3,2
14	50	3,9	54	3,8	53	3,7	62	3,7	66	3,6	68	3,5	72	3,5	76	3,5
15	54	4,2	58	4,1	62	4,0	66	3,9	70	3,9	74	3,8	78	3,7	81	3,7
16	58	4,5	62	4,3	66	4,2	70	4,1	75	4,0	78	3,9	83	3,8	87	3,8
17	61	4,7	66	4,7	70	4,5	75	4,4	79	4,3	83	4,2	88	4,2	92	4,1

4.5. Аппаратные методы второго порядка ускорения умножения в параллельных устройствах

Излагаемые ниже методы можно обобщить для любых оснований системы счисления. Однако мы рассмотрим их лишь применительно к двоичной системе.

4.5.1. Основная идея.

Аппаратные методы второго порядка — это такие методы ускорения умножения, для осуществления которых необходимое количество дополнительного оборудования пропорционально примерно m^2 — квадрату количества разрядов в перемножаемых числах.

Принцип построения множительного устройства, использующего метод ускорения умножения 2-го порядка, иллюстрируется рис. 4-9.

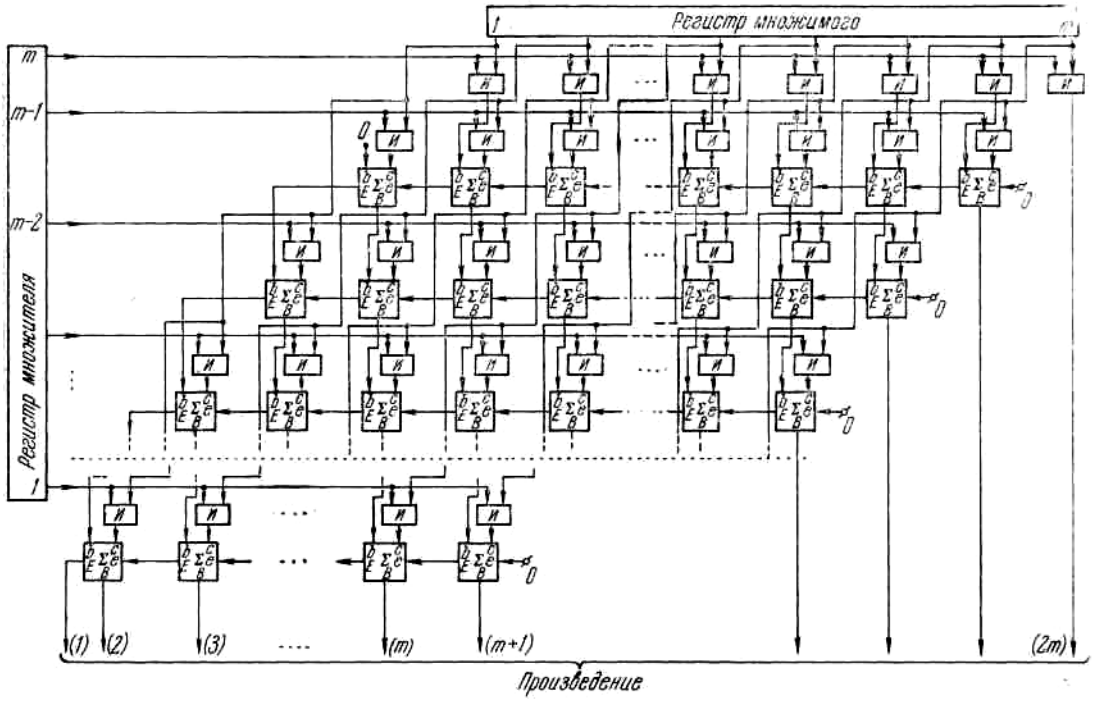


Рис. 4-9. Принцип построения множительного устройства, в котором использован метод ускорения умножения 2-го порядка.

Устройство состоит из m — 1 параллельных сумматоров (каждый — на m разрядов) и m рядов логических элементов «и», по m элементов в ряду. Параллельные m-разрядные сумматоры изображены на рисунке в виде

рядов одноразрядных сумматоров, по m в ряду.

Каждый из рядов элементов «и» управляется одним каким-нибудь разрядом множителя; например, во все элементы «и» первого ряда (на рисунке — верхний ряд) вводится сигнал, соответствующий младшей цифре множителя, во все элементы «и» второго ряда —сигнал, соответствующий (m — 1)-й цифре множителя, и т. д. На другие входы элементов «и» поданы сигналы, соответствующие цифрам множимого. Если, к примеру, m-я цифра множителя есть «1», то комбинация сигналов на выходах элементов «и» первого ряда соответствует комбинации сигналов на выходе регистра множимого, т. е. изображает множимое; если же m-я цифра множителя есть «0», то и сигналы на выходах всех элементов «и» первого ряда являются нулями. Таким образом, первый ряд элементов «и» отрабатывает частичное произведение множимого на младшую цифру множителя.

Точно так же второй ряд элементов «и» отрабатывает частичное произведение множимого на (m — 1)-ю цифру множителя и т. д.

Параллельные сумматоры, имеющиеся в составе устройства, выполняют суммирование этих частичных произведений с необходимым сдвигом друг относительно друга. В первом из сумматоров (на рисунке—верхний ряд одноразрядных сумматоров) суммируются частичные произведения множимого на m-ю и на (m — 1)-ю цифры множителя, причем второе сдвинуто относительно первого на один разряд влево; во втором параллельном сумматоре сумма предыдущих частичных произведений суммируется с частичным произведением множимого на (m — 2)-ю цифру множителя, причем последнее сдвинуто относительно предыдущей суммы частичных произведений на один разряд влево; и т. д. На выходах последнего параллельного сумматора (на рисунке — нижний ряд одноразрядных сумматоров) получается окончательный результат — точнее, его (m + 1) старших разрядов; (m — 1) младших разрядов произведения получаются с выходов младших одноразрядных сумматоров предыдущих рядов.

В тех случаях, когда задача получения $2m$-разрядного произведения не ставится, количество оборудования можно существенно сократить. Сумматоры и элементы «и» образуют на рис. 4-9 как бы ромб; крайний правый одноразрядный сумматор нижнего ряда находится на малой диагонали этого ромба. Если произведение должно получаться с точностью только до m-го разряда, то в схеме нужно сохранить только ту часть оборудования, которая находится слева от этой диагонали, а также несколько вертикальных колонок (несколько разрядов) справа от диагонали; количество этих дополнительных разрядов, необходимых для правильного округления произведения, можно определить в соответствии с результатами раздела 4.1.4. В полной схеме количество логических элементов «и» равно m^2, а количество одноразрядных сумматоров — m (m — 1) (при обычных значениях m, когда $m = 30 \div 40$, m (m — 1) $\approx m^2$); в сокращенной схеме оборудования почти вдвое меньше. В любом случае, однако, количество оборудования весьма велико по сравнению с тем, что требовалось для осуществления рассмотренных ранее методов.

Заметим еще, что для передачи сигналов с выходов одних сумматоров на входы других и т. д. могут потребоваться промежуточные усилители постоянного тока. Такие усилители во многих случаях дают одновременно с усилением инвертирование сигнала; ставить же 2 каскада усиления невыгодно как с точки зрения количества оборудования, так и точки зрения быстродействия. В этой ситуации весьма полезной оказывается самодвойственность функций B и E, рассмотренная в 2.2.3 и используемая аналогично тому, как это предлагалось в указанном разделе применительно к параллельным сумматорам.

Какой же выигрыш по быстродействию может быть получен в данной схеме?

Пусть сигналы, соответствующие цифрам сомножителей, поданы на входы схемы все одновременно; это должны быть сигналы потенциального типа — скажем, в виде уровней напряжения с выходов статических триггеров регистров множителя и множимого. Сразу вслед за этим срабатывают все логические элементы «и», имеющиеся в составе устройства; сигналы на их выходах появляются с небольшой задержкой относительно момента появления входных сигналов, на всех элементах «и» одновременно. Основное время требуется, конечно, не для работы элементов «и», а для работы сумматоров. Первым срабатывает одноразрядный сумматор, находящийся в правом верхнем углу ромба (младший одноразрядный сумматор верхнего ряда). Через промежуток времени τ_E сигнал с его выхода переноса E поступает на вход переноса e второго справа одноразрядного сумматора верхнего ряда и начинается срабатывание второго одноразрядного сумматора; через промежуток времени $2\tau_E$ начнется срабатывание третьего одноразрядного сумматора верхнего ряда и т. д. Обозначим через τ_B время, необходимое для получения в одноразрядном сумматоре сигнала суммы B, а через τ_E —время получения сигнала переноса E после того, как на его входы поступил последний из входных сигналов. При этом окажется, что второй сверху ряд сумматоров начинает работу со сдвигом по времени $\tau_E + \tau_B$ относительно первого ряда; срабатывание крайнего правого одноразрядного сумматора второго ряда начинается с того момента времени, когда на его вход b поступает выходной сигнал B из второго одноразрядного сумматора верхнего ряда; далее через время τ_E (т. е. всего через $2\tau_E + \tau_B$ от начала работы сумматоров) одновременно поступают сигналы на входы b и e второго одноразрядного сумматора второго ряда и т. д. «Волна» установления уровней сигналов, начинаясь в правом верхнем углу ромба, распространяется одновременно (хотя и с разными скоростями) и в горизонтальном, и в вертикальном направлениях. Последним в схеме срабатывает, очевидно, одноразрядный сумматор в левом нижнем углу ромба; установлением напряжения на его выходе B заканчивается образование сигналов произведения на выходах устройства. Нетрудно подсчитать, что время формирования произведения равно примерно $2m\tau_E + m\tau_B$.

Для сравнения напомним, что, например, в множительном устройстве с регистром запоминания переносов,

описанном в 4.3.2 (см. рис. 4-7 на стр. 170), время образования произведения было равно $m\tau_c + \tau_\Sigma$, где τ_c —такт сдвига, τ_E — время однократного суммирования m-разрядных чисел; так как $\tau_\Sigma = m\tau_E + \tau_B$, то $m\tau_c + \tau_\Sigma \approx m\tau_E + m\tau_c$.

На первый взгляд кажется, что это времена одного порядка и что устройство рис. 4-9 не дает никакого или почти никакого выигрыша в быстродействии. В действительности это не так.

Заметим, что условия работы одноразрядных сумматоров в схеме рис. 4-9 заметно отличаются от обычных условий работы в схеме параллельного сумматора.. Если обычно можно считать, что сигналы цифр слагаемых b и с установлены на входах заранее и необходимо обеспечить лишь минимальную задержку между поступлением входного сигнала переноса ей образованием выходного сигнала переноса E, то здесь во многих случаях входной сигнал переноса e и сигнал одного из слагаемых (на входе b) появляются одновременно, и необходимо обеспечить минимальное запаздывание между временем их появления и образованием сигналов на выходах B и E одноразрядного сумматора. Поэтому здесь необходимо строить весь одноразрядный сумматор из быстродействующих логических элементов; не имеет смысла выделять участок формирования переносов, а в участках формирования суммы и формирования подготовительных функций применять более медленные элементы, как это предлагалось в 2.4.1.

При использовании во всем сумматоре одинаковых логических элементов с временем срабатывания τ_1 на ступень для многих построений сумматора оказывается, что $\tau_E \approx \tau_1$, $\tau_B \approx 2\tau_1$. В этом случае время выполнения умножения равно примерно $4m\tau_1$.

В современных потенциальных элементах схем, в которых многократно применяются ограничители для стандартизации уровней сигналов, подавляющее большинство неисправностей проявляется в виде уменьшения скорости установления выходного сигнала.

В множительном устройстве рис. 4-7 и вообще в устройствах, рассматривавшихся в предыдущих разделах, схема управления нарезала полное время умножения на ряд «тактов», в течение каждого из которых должны были одновременно сработать во всех разрядах сравнительно короткие цепочки из нескольких потенциальных элементов. Например, в схеме рис. 4-7 (стр. 170) в течение каждого из тактов, кроме заключительного, во всех m разрядах срабатывают по 2 цепочки:

—элемент «и» на выходе регистра C — одноразрядный сумматор (выход E) — триггер E;

—элемент «и» на выходе регистра C — одноразрядный сумматор (выход B) —триггер регистра B.

При использовании обычных потенциальных элементов эти цепочки содержат примерно по 4 элемента каждая, так что $\tau_c \approx 4\tau_1$.

В отличие от этого в множительном устройстве рис. 4-9 схема управления отводит один, сравнительно большой отрезок времени на всю операцию, в течение которого должна сработать длинная цепочка из примерно $4m$ элементов и одновременно с ней несколько более коротких цепочек.

Время, отводимое на срабатывание цепочки потенциальных элементов, выбирается обычно равным сумме времен срабатываний всех элементов, входящих в цепочку, плюс некоторый запас. При этом ясно, что чем длиннее цепочка элементов, на которую отводится единое время для срабатывания (без нарезки на более мелкие такты), тем меньше может повлиять снижение быстродействия одного какого-нибудь элемента на общее время срабатывания цепочки.

Зная закон распределения времени срабатывания для отдельных элементов (с учетом технологического разброса, старения и т. д.), можно подсчитать выигрыш в быстродействии, который при этом получается. Пусть, например, время срабатывания τ_1 отдельных элементов распределено по нормальному закону с дисперсией D_1 и соответственно среднеквадратичным отклонением $\sigma_1 = \sqrt{D_1}$, а на срабатывание цепочки из k элементов отводится время $k\tau_{1ср}$+ +3σ_k, где $\tau_{1ср}$ — среднее значение τ_1, а σ_k — среднеквадратичное отклонение времени срабатывания цепочки из k элементов. (Время срабатывания цепочек из k элементов в этом случае тоже распределено по нормальному закону, причем его среднее значение равно $k\tau_{1ср}$, дисперсия $D_k = kD_1$, а среднеквадратичное отклонение $\sigma_k = \sqrt{Dk} = \sqrt{k}\,\sigma_1$.) Таким образом, время, отводимое на срабатывание цепочки из 4 элементов, должно быть равно $4\tau_{1ср}$ + 3 2 σ_1, и в выражении для времени срабатывания схемы рис. 4-7 член $m\tau_c$ в действительности равен примерно $4m\tau_{1ср}$ + $6m\sigma_1$ а член $m\tau_E$ — величине $m\,\tau_{1ср} + 3\sqrt{m}\,\sigma_1$, так что полное время умножения в схеме рис. 4-7должно быть равно $5m\,\tau_{1\,ср} + (6m + 3\sqrt{m})\,\sigma_1$. Для схемы рис. 4-9 при равных требованиях к надежности на выполнение умножения должно отводиться время

$$4m\,\tau_{1ср} + 3\sqrt{4m\,\sigma_1} = 4m\,\tau_{1ср} + 6\sqrt{m\,\sigma_1}$$

Чем большую надежность требуется получить, тем большие величины отклонений σ_1 должны быть допущены (с учетом старения и других дестабилизирующих факторов) и тем, следовательно, больший выигрыш по быстродействию дает схема рис. 4-9.

Дополнительный выигрыш по быстродействию получается в ней за счет перекрытия фронтов срабатывания отдельных элементов и за счет отсутствия фронтов сигналов управления (кроме двух — в начале и в конце операции).

Нельзя сбрасывать со счетов и другие преимущества рассмотренного устройства. К ним относится в первую

очередь крайне простая схема управления множительным устройством. Если обычно схема управления должна чередовать команды суммирования и сдвига, выполнять счет циклов умножения и производить ряд других операций, то здесь достаточно лишь выждать определенное (всегда одно и то же) время от момента установления сомножителей.

Еще одно преимущество рассмотренной схемы состоит в простоте ее настройки и выявления дефектов. В большинстве случаев вся проверка схемы может быть произведена в статике с помощью обычного вольтметра: набирая в регистрах те или иные комбинации чисел и измеряя напряжения в различных точках схемы, можно установить, правильно ли работает каждый из ее элементов.

4.5.2. Варианты основной схемы.

Схема, представленная на рис. 4-9, является далеко не единственным и не всегда лучшим решением, позволяющим реализовать метод ускорения умножения 2-го порядка. Прежде всего очевидно, что ромб из логических элементов «и» и сумматоров можно повернуть в другую сторону — так, чтобы умножение начиналось не от младших, а от старших разрядов множителя. Такое построение показано на рис. 4-10. Здесь первые два ряда логических элементов «и» (на рисунке — верхние) формируют частичные произведения множимого на две старшие цифры множителя — 1-ю и 2-ю. Первый ряд одно разрядных сумматоров суммирует их, причем второе частичное произведение сдвинуто на 1 разряд вправо относительно первого. В третьем ряду одноразрядных сумматоров к предыдущей сумме частичных произведений добавляется частичное произведение множимого на третью цифру множителя, причем последнее вновь сдвинуто на 1 разряд вправо и т. д.

Обратим внимание на то обстоятельство, что в каждом ряду параллельный сумматор образуется не из всех *m,* а только из *m* — 1 одноразрядных сумматоров. Последний (крайний левый) одноразрядный сумматор из каждого ряда включается в отдельный дополнительный сумматор, расположенный вдоль левой стороны ромба (снизу вверх); этот дополнительный сумматор на рисунке обведен пунктиром. Дополнительный сумматор выполняет суммирование цифр, которые появляются на выходах *B* и *E* старших разрядов основных сумматоров.

По количеству оборудования схема рис. 4-10 полностью эквивалентна схеме рис. 4-9. Как и в предыдущей схеме, здесь возможно сокращение оборудования почти вдвое в том случае, когда произведение должно получаться только с *m* верными знаками.

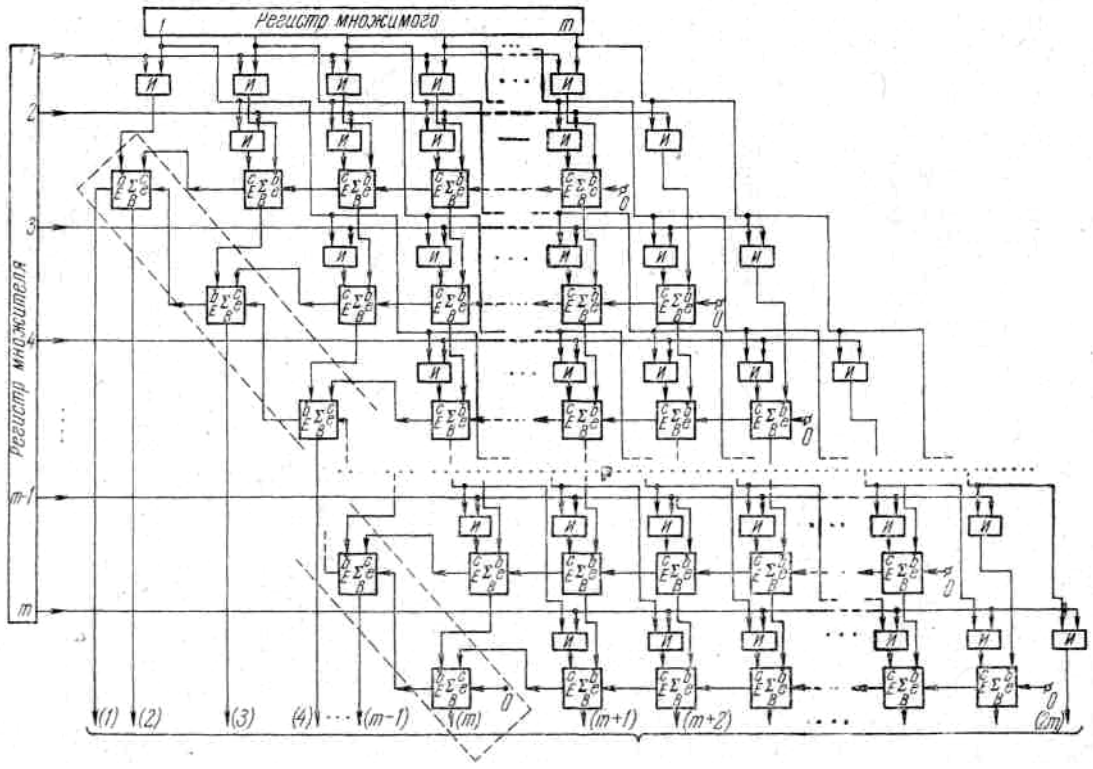


Рис. 4-10. Вариант множительного устройства с использованием метода ускорения умножения 2-го порядка (умножение начинается от старших разрядов).

По быстродействию схема рис. 4-10 несколько лучше, чем схема рис. 4-9. «Волна» установления сигналов начинается в ней от правой (наклонной) стороны ромба: младшие одноразрядные сумматоры во всех рядах получают входные сигналы все одновременно. Далее фронт «волны» перемещается влево параллельно самому себе, однако у верхней границы ромба скорость распространения волны несколько выше, и поэтому по мере продвижения волны влево ее фронт у верхнего края все больше выдвигается вперед. Скажем, вторые справа одноразрядные сумматоры всех горизонтальных рядов срабатывают с задержкой $1\tau_B$ относительно крайних правых сумматоров (они срабатывают после того, как на вход *b* каждого из них поступает цифра с выхода *B* крайнего

правого одноразрядного сумматора предыдущего ряда); только в самом верхнем ряду второй справа одноразрядный сумматор срабатывает несколько раньше — с задержкой $1\tau_E$ относительно первого*). Третьи справа одноразрядные сумматоры всех рядов срабатывают с задержкой $2\tau_B$ относительно первых, но верхний и второй сверху ряды представляют в этом смысле исключение: в верхнем ряду задержка для третьего справа одноразрядного сумматора составляет всего $2\tau_E$, а во втором сверху ряду $1\tau_E + 1\tau_B$ и т. д. Левой стороны ромба волна достигает не одновременно: раньше других срабатывает старший (второй слева) одноразрядный сумматор верхнего ряда, затем старший разряд второго сумматора и т. д., наконец — с запаздыванием $(m - 2)\tau_B$ относительно начала умножения — старший (второй слева) одноразрядный сумматор в нижнем ряду. После этого идет распространение сигналов переноса вдоль дополнительного сумматора (по крайним левым одноразрядным сумматорам всех рядов) — из нижнего левого угла ромба к верхнему левому углу; запаздывание от разряда к разряду здесь составляет $1\tau_E$. Итого, время выполнения умножения оказывается равным примерно $m\tau_E + m\tau_B$ вместо $2m\tau_E + m\tau_B$ в схеме рис. 4-9.

В части быстродействия у схемы рис. 4-10 имеется еще одна интересная особенность по сравнению со схемой рис. 4-9. Дело в том, что в ней в течение всей заключительной части умножения (длительностью в $m\ \tau_E$) идет установление сигналов только вдоль дополнительного сумматора (по левой стороне ромба), а во всей остальной части схемы сигналы остаются неизменными. Ускорив один только этот сумматор — например, одним из способов, изложенных в 2.5.2, или 2.5.3, или 2.5.4,— можно получить дополнительно существенное сокращение времени выполнения умножения. В схеме рис. 4-9 для достижения аналогичного результата пришлось бы применять схемы ускорения практически во всех сумматорах устройства.

Некоторыми новыми свойствами обладает схема рис. 4-11.

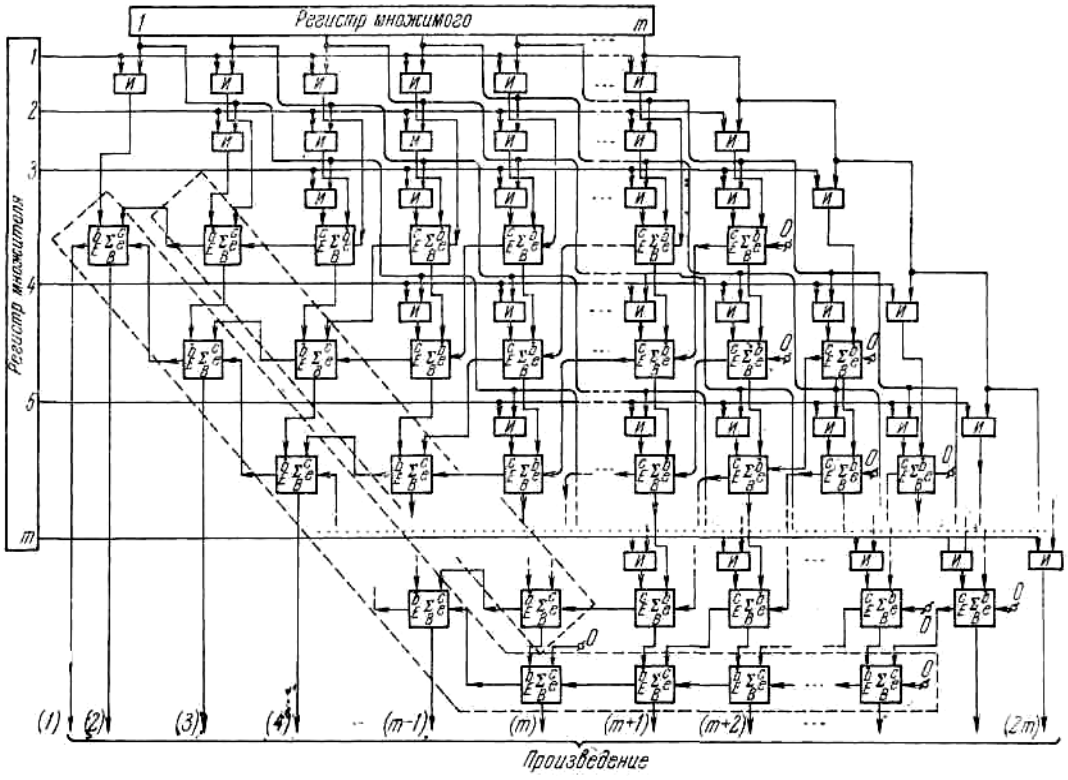


Рис.4-11. Вариант множительного устройства с использованием метода ускорения умножения 2-го порядка (переносы распространяются по столбцам).

Расположение логических элементов «и» и частичных произведений в ней такое же, как в схеме рис. 4-10. Основное отличие — в коммутации одноразрядных сумматоров: выход переноса (E) каждого из основных одноразрядных сумматоров соединяется, как и должно быть, со входом e соседнего слева одноразрядного сумматора, но не в своем ряду, а в следующем (нижнем) ряду.

Из-за этого несколько изменено и расположение одноразрядных сумматоров. Первый (верхний) ряд одноразрядных сумматоров размещен не после второго, а после третьего ряда логических элементов «и». Он выполняет суммирование трех старших частичных произведений, причем их сумма получается на выходах этого ряда в виде двух чисел: числа, составленного из цифр на выходах B, и числа, составленного из цифр на выходах E. Следующий ряд одноразрядных сумматоров снова суммирует три числа, два из которых представляют предыдущую сумму частичных произведений, а третьим является новое частичное произведение, причем результат суммирования снова получается в виде двух чисел (числа на выходах B и числа на выходах E) и т. д. Таким

*) Мы полагаем, как и в 4.5.1, что время образования сигнала переноса в одноразрядном сумматоре τ_E меньше, чем τ_B — время образования сигнала суммы.

образом, функции каждого из основных рядов сумматоров сводятся к тому, что он сворачивает тройку чисел в пару чисел.

Аналогичные функции выполняет и дополнительный ряд сумматоров, расположенный на рисунке вдоль левой стороны ромба (наискось) и обведенный пунктирной рамкой. Для каждого из старших разрядов произведения в основной части устройства вырабатываются по три цифры: цифра на выходе *B* крайнего левого одноразрядного сумматора данного ряда, цифра на выходе *E* второго слева одноразрядного сумматора этого же ряда и цифра на выходе *E* крайнего левого сумматора следующего ряда. Ряд сумматоров, обведенный пунктирной рамкой, свертывает тройку чисел, составленных из этих цифр, в пару чисел.

После этого для каждого разряда произведения остаются уже по 2 цифры; в старшей части — это цифры на выходах *B* и *E* двух соседних одноразрядных сумматоров дополнительного ряда, в младшей части — это цифры на выходах *B* и *E* двух соседних одноразрядных сумматоров младшего (самого нижнего) из основных рядов. Два числа, составленные из указанных цифр, суммируются специальным параллельным сумматором, который на рис. 4-11 тоже обведен пунктиром и размещен левее и ниже первого из дополнительных рядов. На его выходах получается окончательное произведение.

Главное достоинство рассматриваемого построения по сравнению с описанными ранее состоит в том, что в нем легко достигнуть одновременности срабатывания всех одноразрядных сумматоров любого из основных рядов. В той схеме, которая приведена на рис. 4-11, этого непосредственно не получается: в первом (верхнем) ряду все одноразрядные сумматоры срабатывают одновременно, но в тот же момент времени срабатывают и крайние правые сумматоры всех остальных рядов; дальше «волна» установления напряжений распространяется сверху вниз и справа налево. Однако, введя элементы задержки перед входами крайних правых сумматоров, легко уравнять времена для одноразрядных сумматоров в каждом из горизонтальных рядов. Если же получения 2*m*-разрядного произведения не требуется и из схемы можно исключить правую часть ромба (как это было описано применительно к рис. 4-9), то одновременность срабатывания одноразрядных сумматоров в каждом, из горизонтальных рядов получается сама собой.

Эта особенность построения, приведенного на рис. 4-11, важна с двух точек зрения. Во-первых, ею можно воспользоваться для того, чтобы применить в схеме синхронные элементы импульсного типа. Для каждого ряда сумматоров при этом пришлось бы подбирать свою фазу сигналов синхронизации. Конечно, это связано с определенными затруднениями; но в схемах рис. 4-9 и 4-10 применить синхронные элементы еще труднее. Во-вторых,— и это главное —указанное свойство схемы рис. 4-11 позволяет использовать имеющееся в ней оборудование не только для ускорения умножения, но и для ускорения деления (см. раздел 5.2). Ясно, что при тех больших затратах, которые необходимы для осуществления методов 2-го порядка ускорения умножения, очень желательно возможно полнее и эффективнее загрузить все устанавливаемое оборудование. Построения рис. 4-9 и 4-10 такой возможности не дают.

Приведенными схемами не исчерпывается многообразие различных построений, использующих методы ускорения умножения 2-го порядка. Во многих случаях они отличаются одно от другого некоторыми техническими деталями, иногда быстродействием, возможностью или невозможностью применения дополнительного ускорения и т. д., но являются в общем модификациями основного построения, описанного в 4.5.1. Не имея возможности рассмотреть все известные варианты, мы остановимся в 4.5.3 и 4.5.4 на тех из них, которые содержат некоторые новые идеи.

4.5.3. Сокращение количества оборудования и повышение быстродействия путем усложнения логики.

В настоящем разделе мы встретимся со сравнительно редким в технике случаем, когда повышение быстродействия сопряжено не с увеличением, а с уменьшением количества необходимого оборудования.

Идея предлагаемого метода иллюстрируется рис. 4-12. Схема, которая приведена здесь, построена по аналогии с рис. 4-9, причем на рисунке изображен только левый верхний угол ромба; остальную часть схемы читатель без труда может достроить самостоятельно.

Верхний ряд одноразрядных сумматоров является вспомогательным. В нем выполняется суммирование множимого с удвоенным множимым (сдвинутым на 1 разряд влево) и получается, таким образом, утроенное множимое. Множимое, удвоенное множимое и утроенное множимое поступают на входы первого ряда логических элементов, затем — со сдвигом на 2 разряда влево — на входы второго ряда логических элементов и т. д.

Каждый ряд логических элементов управляется выходными сигналами дешифратора Д. На входы младшего(верхнего) из дешифраторов поступают последняя (младшая) и предпоследняя цифры множителя. Если эти два разряда содержат комбинацию «00», то сигналов нет ни на одном из трех выходов дешифратора, при наличии комбинации «01» дешифратор вырабатывает сигнал «+1», при наличии комбинации «10» — сигнал «+2», а при наличии комбинации «11» — сигнал «+3». Соответственно в верхнем ряду элементов «и» либо заперты все элементы, либо открыты правые или средние или левые элементы «и» из каждой тройки; на выходах первого ряда элементов «или» при этом появляются либо нуль, либо множимое, либо удвоенное множимое, либо утроенное множимое. Таким образом, первый ряд логических элементов отрабатывает частичное произведение множимого на младшую пару разрядов множителя. Точно так же второй ряд логических элементов отрабатывает частичное

произведение множимого на вторую пару цифр множителя и т. д.

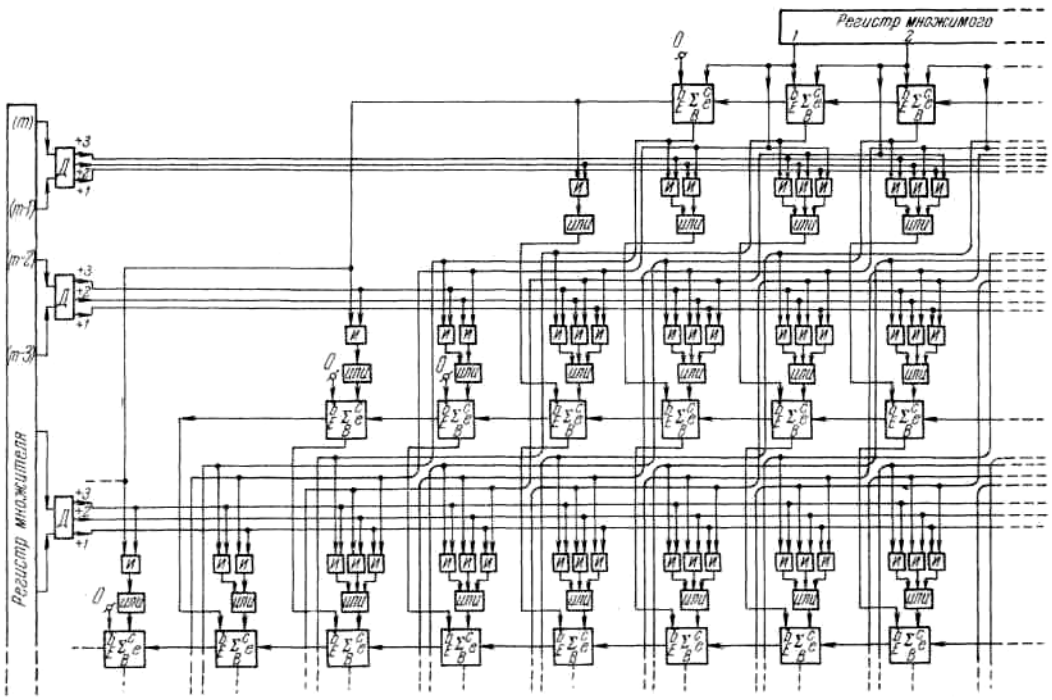


Рис. 4-12. Вариант множительного устройства с использованием метода ускорения умножения 2-го порядка (схема с усложненной логикой).

Первый из основных рядов одноразрядных сумматоров выполняет суммирование первого частичного произведения со вторым, причем второе сдвинуто относительно первого на 2 разряда влево. С помощью следующего ряда одноразрядных сумматоров полученная сумма складывается затем с частичным произведением множимого на следующую пару разрядов множителя, причем последнее вновь сдвинуто на 2 разряда влево и т. д.

Ясно, что в схеме рис. 4-12 сумматоров содержится примерно вдвое меньше, чем в схеме рис. 4-9. Несколько больше в ней логических элементов. Из каждой тройки элементов «и», относящихся к входу c некоторого одноразрядного сумматора, два имеются и в схеме рис. 4-9: элемент «и» для цифры множимого и элемент «и» для цифры удвоенного множимого; последний в схеме рис. 4-9 относится к следующему ряду одноразрядных сумматоров. Третьих же элементов «и» (для цифр утроенного множимого), а также элементов «или» в схеме рис. 4-9 нет. Таким образом, можно считать, что каждый одноразрядный сумматор, исключенный из схемы рис. 4-9, заменен в схеме рис. 4-12 одним элементом «и» и одним элементом «или». Преимущество схемы рис. 4-12 в смысле экономии оборудования очевидно.

Нетрудно убедиться, что и по быстродействию схема рис. 4-12 выгоднее схемы рис. 4-9. Если в схеме рис. 4-9 время формирования произведения было равно примерно $2m\tau_E + m\tau_B$, то здесь оно получится порядка $2m\tau_E + \frac{m}{2}\tau_B$. Полагая, как и прежде, что $\tau_B \approx 2\tau_E$, найдем, что по быстродействию схема рис. 4-12 примерно на 25% лучше, чем схема рис. 4-9. В тех случаях, когда задача получения полноразрядного произведения не ставится и из схемы удаляется правая часть ромба, выигрыш по скорости еще более значителен ($m\tau_E$ +— τ_B для схемы рис. 4-12 вместо $m\tau_E + m\tau_B$ для схемы типа рис. 4-9, т.е. — при $\tau_B \approx 2\tau_E$ — примерно в полтора раза быстрее).

Подобно тому как схема рис. 4-12 построена по аналогии со схемой рис. 4-9, можно построить схемы с усложненной логикой по аналогии со схемами рис. 4-10 и 4-11. Если количество разрядов в перемножаемых числах (m) велико, то может оказаться выгодным сформировать с помощью вспомогательных рядов сумматоров не только утроенное множимое 3c, но также 5c, 6c и 7c и умножение вести не по парам, а по тройкам разрядов множителя.

4.5.4. Многослойные построения.

Во всех рассмотренных выше схемах количество параллельных сумматоров было пропорционально количеству разрядов в перемножаемых числах m (количество одноразрядных сумматоров пропорционально m^2). Однако при этом сумматоры срабатывали последовательно, один за другим, потому что выходные сигналы одного сумматора должны были передаваться на входы следующего; в общее время выполнения умножения входил член, пропорциональный $m\tau_B$, где τ_B — время образования цифры суммы в одноразрядном сумматоре.

Многослойные построения позволяют при том же количестве сумматоров исключить из общего времени умножения слагаемое вида $m\tau_B$; вместо него войдет член вида $\log m\tau_B$.

Идея многослойного построения состоит в следующем.

Пусть, как и прежде, имеем m рядов элементов «и», отрабатывающих m частичных произведений множимого на

каждую из цифр множителя. Такие ряды логических элементов имеются во всех рассмотренных ранее схемах (см., например, рис. 4-9, 4-10, 4-11). Логические элементы составят как бы нулевой «слой» множительного устройства; все они срабатывают одновременно. Первый «слой» построения образуем из $m/2$ параллельных сумматоров, которые будут выполнять попарное суммирование частичных произведений. На выходах этого слоя получим $m/2$ чисел, которые будут передаваться на входы следующего слоя. Во втором слое эти числа складываются попарно с помощью $m/4$ параллельных сумматоров. На выходах сумматоров второго слоя получается $m/4$ чисел, которые передаются на третий слой, и т. д.

Общее количество слоев оказывается равным примерно $\log_2 m$ (потому что каждый слой сокращает количество чисел вдвое, а всего мы должны из m чисел получить одно число), а параллельных сумматоров всего получается примерно $m/2 + m/4 + m/8 + ... \approx m$, т. е. примерно столько же, сколько в рассмотренных ранее построениях (рис. 4-9, 4-10, 4-11). Все сумматоры в каждом слое, как мы говорили, начинают срабатывание одновременно. Начало срабатывания каждого последующего слоя запаздывает относительно начала срабатывания данного слоя на время τ_B. Таким образом, общее запаздывание между началом срабатывания последнего слоя и началом срабатывания первого слоя будет равно примерно $(\log_2 m)\tau_B$.

Возможное видоизменение этой идеи может быть следующим.

Пусть в первом слое содержится не $m/2$, а $m/3$ рядов одноразрядных сумматоров. В отличие от предыдущего внутри каждого ряда выходы цифр переноса одноразрядных сумматоров не соединяются со входами переноса соседних (старших) одноразрядных сумматоров, так что ряд одноразрядных сумматоров по существу не является единым параллельным сумматором; в этом отношении положение аналогично тому, которое имеется на рис. 4-11. К каждому ряду одноразрядных сумматоров подводится три частичных произведения множимого на цифры множителя. Одно из них поступает на входы b одноразрядных сумматоров, другое — на входы c и третье — на входы e; все три входа при этом равноправны. Ряд одноразрядных сумматоров производит как бы суммирование этих трех чисел, но при этом результат получается в виде двух чисел — числа, составленного из цифр на выходах B, и числа, составленного из цифр на выходах E одноразрядных сумматоров; оба эти выхода тоже равноправны. Таким образом, первый слой сумматоров как бы сворачивает m слагаемых, которые должны войти в окончательное произведение, в $\frac{2}{3}m$ слагаемых.

Второй слой будет содержать $\frac{\frac{2}{3}m}{3}$, т. е. $\frac{2}{9}m$ рядов одноразрядных сумматоров. Устроен он точно так же, как и первый слой. На каждый ряд одноразрядных сумматоров подадим 3 числа из выходных чисел первого слоя; на выходах этого ряда получим вместо тройки пару чисел. Таким образом, во втором слое $\frac{2}{3}m$ чисел свертываются в $\frac{2}{3}(\frac{2}{3}m)$, т. е. в $\frac{4}{9}m$ — m чисел.

Далее в третьем слое поставим $\frac{\frac{4}{9}m}{3}$, т. е $\frac{4}{27}m$ рядов одноразрядных сумматоров, которые снова уменьшат количество имеющихся чисел в 1,5 раза и т. д. Только в самом конце, когда из m исходных чисел получится уже всего 2 числа, произведем суммирование этой пары чисел со всеми сквозными переносами; в результате и получим окончательный результат умножения.

Общее количество рядов одноразрядных сумматоров, $\frac{1}{3}m + \frac{2}{9}m + \frac{4}{27}m + ... \approx m$, т. е. то же, что и в предыдущем многослойном построении. Однако количество слоев здесь несколько больше — $\log_{\frac{3}{2}} m$ (вместо $\log_2 m$ в предыдущем построении). Поэтому и время умножения несколько больше: вместо слагаемого $\log_2 m\ \tau_B$ в него входит слагаемое $\log_{\frac{3}{2}} m\ \tau_B$. Зато преимущество состоит в том, что в каждом слое (кроме последнего) все одноразрядные сумматоры срабатывают одновременно; распространение сигналов переноса от младших разрядов к старшим внутри каждого слоя отсутствует. Это дает возможность строить схему из синхронных, элементов, причем на каждый слой сумматоров нужно будет подавать сигналы синхронизации в своей фазе: на первый слой — в одной фазе, на второй — с некоторым запаздыванием, на третий — с еще большим запаздыванием и т. д. Если к какому-то слою суммарное запаздывание составит целый период, так что на k-й слой сигналы синхронизации нужно будет подавать в той же фазе, что и на 1-й слой, то k-й и все последующие слои можно не строить; вместо этого выходные сигналы $(k$ — 1)-го слоя можно подать на входы первого слоя (причем из сумматоров первого слоя будет использована только часть) и т. д.

В многослойных построениях, так же как в построениях разделов 4.5.1 и 4.5.2, сокращение количества оборудования и некоторое увеличение быстродействия могут быть достигнуты путем усложнения логики (см. 4.5.3).

4.6. Об умножении в последовательных устройствах

Многие из аппаратных методов ускорения умножения, рассмотренных нами ранее применительно к параллельным устройствам, могут быть в том или ином виде применены и при построении последовательных устройств. При этом, однако, большей частью сохраняется только общая идея; по форме же устройство получается совсем другим.

Ниже рассматриваются различные построения ускоренных множительных устройств последовательного типа. В части из них использованы те же идеи, которые применялись для ускорения умножения в параллельных устройствах; другие основаны на отличных принципах.

Все изложение ведется применительно к двоичной системе, хотя многие из рассматриваемых построений можно непосредственно обобщить на любые другие основания системы счисления.

4.6.1. Использование чисел, кратных множимому.

Использование чисел, кратных множимому, в последовательных устройствах по существу аналогично методу, описанному в разделе 4.4.1 применительно к параллельным устройствам. Конкретные решения могут быть весьма разнообразными. Одно из них — в качестве иллюстрации — приведено на рис. 4-13, причем технические подробности на рисунке опущены.

По сравнению с обычными множительными устройствами последовательного типа новым элементом здесь является схема формирования чисел, кратных множимому; на рисунке она обведена пунктирной рамкой. Непосредственно с выхода регистра множимого в последовательном коде

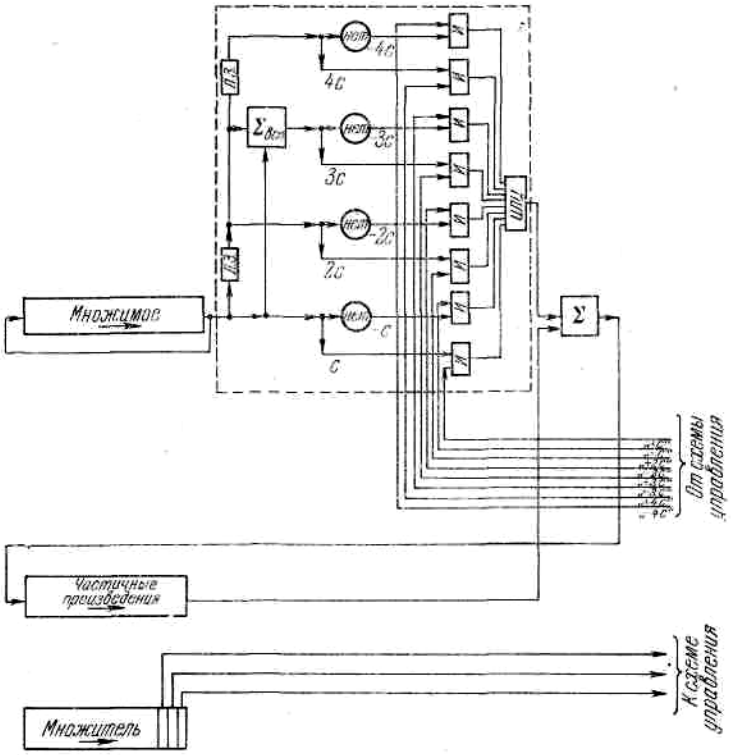


Рис. 4-13. Принцип построения последовательного множительного устройства, в котором для ускорения умножения используются числа, кратные множимому.

получается само множимое (*c*); инвертор («нет»), присоединенный к выходу регистра, позволяет одновременно получать обратный код множимого, т. е. величину — *c* (с учетом того, что в первом такте суммирования может быть добавлена единица младшего разряда для получения дополнительного кода). Линия задержки на 1 такт (Л. 3.), присоединенная к выходу регистра, дает возможность получать удвоенное множимое, а следующая линия задержки (еще на 1 такт) — учетверенное множимое. Кроме того, в схему формирования кратных входит вспомогательный сумматор $\Sigma_{всп}$, который путем суммирования $c + 2c$ формирует утроенное множимое. На каждом из выходов удвоенного, утроенного и учетверенного множимого имеются инверторы («нет»), выполняющие обращение соответствующего кода. Таким образом, в рассматриваемой схеме формируются 8 чисел, кратных множимому *c*: — 4*c*, —3*c*, —2*c*, —*c*, *c*, +2*c*, +3*c* и +4c.

Далее имеется 8 элементов «и», соединенных с входами одного элемента «или». Схема управления открывает на время одного цикла умножения один какой-нибудь элемент «и», либо не открывает ни одного из этих элементов. В результате на вход основного сумматора (2) множительного устройства поступает либо одно из восьми чисел, кратных множимому, либо нуль.

Сигналы для отпирания элементов «и» вырабатываются схемой управления в соответствии с той комбинацией

цифр, которая имеется в трех очередных разрядах множителя. Получая каждый раз тройку двоичных разрядов, схема управления фактически ведет умножение как бы в восьмеричной системе; по ходу дела множитель преобразуется ею из обычной восьмеричной записи — с цифрами «О» (двоичная комбинация 000),«1» (двоичная комбинация 001),..., «7» (двоичная комбинация 111) — в форму записи цифрами —4, —3, —2, —1, 0, 1, 2, 3, 4. Поскольку все соответствующие кратные множимому заготавливаются заранее, умножение на тройку двоичных разрядов (т. е. на один восьмеричный разряд) требует всего одного суммирования. Ускорение по сравнению с обычным умножением (без схемы формирования чисел, кратных множимому) получается примерно в 3 раза.

4.6.2. Введение дополнительных сумматоров.

На рис-4-14 имеется иллюстрация к другому методу ускорения умножения в последовательных устройствах — методу введения дополнительных сумматоров.

В отличие от обычных множительных устройств множимое в последовательном коде, подаваемое из регистра множимого, поступает не на один, а одновременно на 2 элемента «и». Эти два элемента управляются двумя очередными цифрами множителя: элемент «и-1» — старшей из этой пары цифр, элемент «и-2» — младшей. Если, например, старшая из очередных цифр множителя есть 1, то в течение соответствующего цикла умножения элемент «и-1» открыт и на его выход поступает последовательный код множимого; если старшая из очередных цифр множителя есть «0», то элемент «и-1» заперт, и на его выход поступает нуль. Аналогичным образом элемент «и-2» отрабатывает частичное произведение множимого на младшую из очередной пары цифр множителя. Последовательный сумматор Σ_1 складывает предыдущую сумму частичных произведений, поступающую из регистра, с частичным произведением, отрабатываемым элементом «и-1». Прежде чем вернуться в регистр частичных произведений, результат с выхода сумматора $\sum_1$ через линию задержки на 1 такт (Л. 3.) поступает на вход второго сумматора Σ_2, в котором происходит суммирование с частичным произведением, отрабатываемым элементом «и-2».

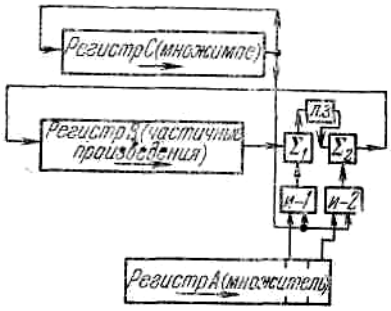


Рис. 4-14. Последовательное множительное устройство с двумя сумматорами.

Таким образом, в течение одного цикла умножения к предыдущей сумме частичных произведений добавляется не одно, как обычно, а одновременно два очередных частичных произведения. Умножение при этом выполняется примерно вдвое быстрее, чем в обычных множительных устройствах последовательного типа (с одним сумматором).

Точно так же при наличии трех последовательных сумматоров длительность умножения сократилась бы примерно втрое, при наличии четырех сумматоров — вчетверо и т. д.

Мы рассмотрим сейчас предельный случай — когда количество сумматоров равно количеству разрядов в перемножаемых числах (m). Казалось бы, время умножения должно при этом сократиться в m раз — до одного периода числа, т. е. до m тактов. В действительности, однако, при наличии двух сумматоров время умножения равно $m/2$ периодам числа плюс 1 такт (на прохождение результата через линию задержки Л.3.; см. рис. 4-14), при наличии трех сумматоров — $m/3$, периодам числа плюс 2 такта и т. д.; при наличии m сумматоров оно равно одному периоду числа ($m/m = 1$) плюс (m — 1) тактов, т. е. примерно двум периодам числа.

Устройство, о котором идет речь, изображено на рис. 4-15. В схему входит m элементов «и», каждый из которых управляется своим разрядом множителя. В течение всего времени умножения множитель в регистре A неподвижен. Из регистра C на элементы «и» последовательным кодом подается множимое. Как и в схеме рис. 4-14, элементы «и» отрабатывают частичные произведения множимого на цифры множителя; например, на выходе k-го элемента «и» получается в последовательном коде величина $a_k c$, которая либо равна нулю (если k-я цифра множителя есть 0, $a_k = 0$), либо равна c (если $a_k = 1$).

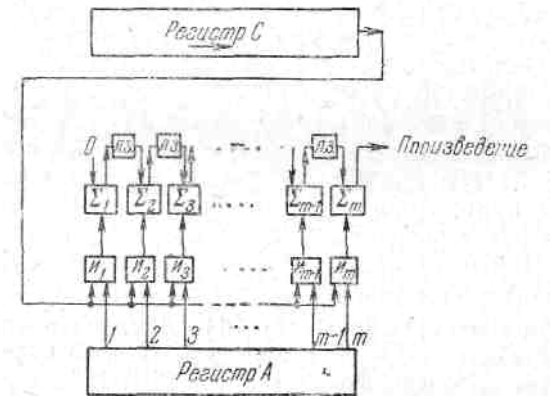


Рис. 4-15. Предельный случай применения дополнительных сумматоров — последовательное множительное устройство, содержащее m сумматоров.

Сумматоры выполняют суммирование этих частичных произведений. На выходе 1-го сумматора (Σ_1) получаем величину $a_1 c$, где a_1 — старшая цифра множителя. Сдвиг на 1 такт, выполняемый первой линией задержки (Л. 3.), эквивалентен удвоению этой величины. Далее во втором сумматоре (Σ_2) происходит сложение с $a_2 c$, причем получается $2a_1 c + a_2 c$. После второй линии задержки на 1 такт получим величину $4a_1 c + 2a_2 c$, а после третьего сумматора (Σ_3) — величину $4a_1 c + 2a_2 c + a_3 c$ и т. д. На выходе m-го сумматора получается величина

$$2^{m-1}a_1 c + 2^{m-2}a_2 c + \ldots + 2a_{m-1}c + a_m c.$$

Передвинув мысленно запятую в этом числе так, чтобы она находилась перед первым (старшим) разрядом, мы

уменьшим его в 2^m раз. При этом оно будет равно

$$2^{-1}a_1c + 2^{-2}a_2c + \ldots + 2^{-(m-1)}a_{m\_1}c + 2^{-m}a_mc,$$

т. е. искомому произведению $a \times c$ (поскольку a_1, a_2,... — это двоичные цифры множителя a, и $a = 2^{-1}a_1 + 2^{-2}a_2 + \ldots + 2^{-m}a_m$).

Младшая цифра произведения появляется на выходе последнего сумматора в том же такте, в котором из регистра *C* выходит младшая цифра множимого. Так как всего произведение содержит $2m$ разрядов, полное время умножения равно $2m$ — 1 тактов. При этом в течение первых m тактов из регистра *C* выдаются цифры множимого, а в течение последующих тактов — нули*); когда с выхода последнего сумматора выходят старшие m разрядов произведения, ни множимое, ни множитель уже не играют фактически роли: вся необходимая информация к этому времени имеется в линиях задержки Л. 3., а также в тех элементах запоминания, которые входят в состав последовательных сумматоров и предназначены для хранения цифр переноса.

Даже с первого взгляда ясно, что по количеству оборудования и по скорости выполнения умножения схема рис. 4-15 весьма близка к множительным устройствам параллельного типа. Однако при более внимательном рассмотрении оказывается, что она попросту в точности совпадает со схемой множительного устройства параллельного типа, описанной в разделе 4.3.2 и изображенной на рис. 4-7 (стр. 170).

Действительно. Работу схемы рис. 4-15 можно объяснить следующим образом. Предположим, что в регистре *A* находится множимое (на рис. 4-7 этот регистр назван регистром *C)*, а в регистре *C* — множитель (на рис. 4-7 — это регистр *A).* Множимое из регистра *A* схемы рис. 4-15 параллельным кодом выдается на систему элементов «и», которые в каждом такте управляются очередной цифрой множителя, поступившей из регистра *C.* На выходах элементов «и» в каждом такте в параллельном коде получается очередное частичное произведение множимого (a) на соответствующую цифру множимого (c_i): если $c_i = 0$, то c_ia — = 0, если $c_i = 1$, то $c_ia = a$. Ряд последовательных сумматоров $\Sigma_1 \div \Sigma_m$, имеющихся на рис. 4-15, можно рассматривать как единый параллельный сумматор, в котором разорваны цепи сквозных переносов; элементы запоминания цифр переноса, которые входят в состав каждого из последовательных сумматоров, на рис. 4-7 изображены отдельно, в виде триггеров регистра *E*; значком «Σ» на рис. 4-7 обозначен одноразрядный двоичный сумматор, который в соединении с триггером *T* из регистра *E* образует один последовательный сумматор — то, что обозначено как Σ_i на рис. 4-15. В сумматоре происходит сложение частичных произведений, причем результат каждого суммирования получается в виде 2 чисел; одно из них — состоящее из цифр на выходах *E* одноразрядных сумматоров — запоминается внутри последовательных сумматоров (на рис. 4-7 в триггерах регистра *E)* и в следующем такте поступает на входы *e* одноразрядных сумматоров; другое число — состоящее из цифр на выходах *B* одноразрядных сумматоров — передается в ряд линий задержки (Л. 3.) (на рис. 4-7 — регистр *B)* и в следующем такте возвращается со сдвигом вправо на 1 разряд на входы *b* одноразрядных сумматоров.

Разница между схемами рис. 4-15 и 4-7 состоит, как мы видим, по существу лишь в том, каким способом заканчивается умножение, а также в какой форме и откуда воспринимается его результат. В схеме рис. 4-7 в течение первых m тактов умножения младшие цифры произведения, которые получаются последовательно, одна за другой, на выходе младшего разряда сумматора, передаются в освобождающиеся разряды регистра множителя (регистр *A* на рис. 4-7); затем в сумматоре открываются цепи сквозного переноса, число из регистра *E* добавляется к числу из регистра *B* и результат умножения прочитывается параллельным кодом: младшие m разрядов — из регистра *A,* старшие m разрядов — из регистра *B.* В схеме рис. 4-15 произведение прочитывается в последовательной форме прямо с выхода младшего из сумматоров, причем после первых m тактов тратится еще столько же времени на то, чтобы цифры, хранившиеся в элементах запоминания последовательных сумматоров, просуммировались бы с цифрами, имеющимися в линиях задержки (Л. 3.); за это время на выходе последнего сумматора как раз и появляются старшие цифры произведения; времени на это тратится больше, чем в схеме рис. 4-7, зато нет необходимости в цепях сквозного переноса, и набор из m последовательных сумматоров никогда не должен превращаться в настоящий m-разрядный параллельный сумматор.

Мы упоминали уже в 4.3.2, что многие авторы независимо друг от друга предлагали схему рис. 4-7 или ее разновидности, не зная о том, что схема эта была впервые описана в одной из самых ранних работ по вычислительной технике (см. сноску на стр. 169), и не замечая, что она в точности идентична широко известной схеме рис. 4-15. Вместе с тем не стоит пренебрегать и тем фактом, что схема рис. 4-7 была получена как предельный случай применения в параллельных множительных устройствах методов запоминания переносов, а схема рис. 4-15 — как предельный случай применения в последовательных устройствах метода введения дополнительных сумматоров. Если в 4.3.2 мы обсуждали вопрос о том, как многие из других методов ускорения умножения, известных для параллельных устройств, можно применять в комбинации с идеей схемы рис. 4-7, то здесь имело бы смысл посмотреть, как идеи схемы рис. 4-15 можно сочетать с другими идеями ускорения умножения в последовательных устройствах.

*) Как мы увидим в 4.7, при выполнении умножения с учетом алгебраических знаков в течение последних m тактов из регистра множимого в некоторых случаях должны выдаваться единицы.

Для иллюстрации возможностей, которые возникают при этом, на рис. 4-16 приведен принцип построения множительного устройства 2-й вычислительной машины (Mark-II) Манчестерского университета. В устройстве этом наряду с принципами, заложенными в схеме рис. 4-15, использована идея формирования чисел, кратных множимому (рис. 4-13 на стр. 191).

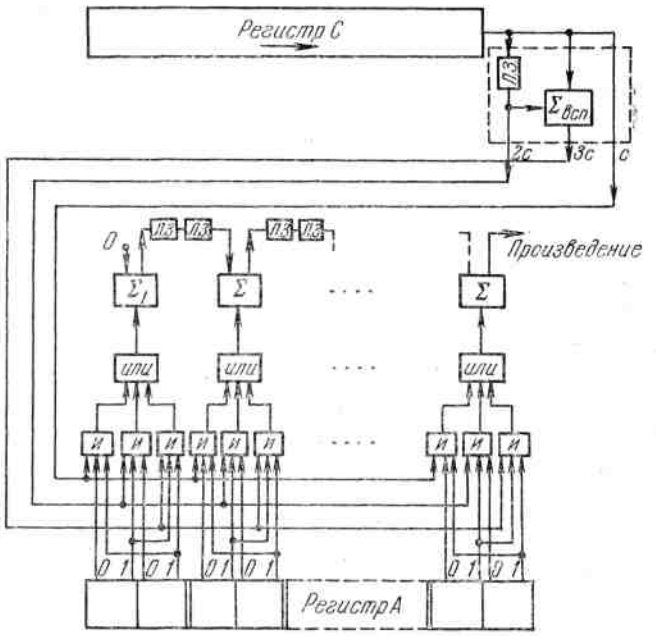


Рис. 4-16. Множительное устройство, аналогичное по принципу построения схеме рис. 4-15, но с применением чисел, кратных множимому.

Часть схемы, формирующая числа, кратные множимому, обведена на рис. 4-16 пунктирной рамкой. Линия задержки на 1 такт, через которую проходит последовательный код множимого, дает удвоенное множимое ($2c$), а вспомогательный последовательный сумматор $\Sigma_{всп}$, выполняющий суммирование $c + 2c$, формирует утроенное множимое ($3c$).

Величины c, $2c$ и 3с поступают на систему логических элементов, управляемых цифрами множителя. С каждой парой разрядов множителя соединена логическая схема «и-или», устроенная так, что, когда в данной паре разрядов содержится комбинация цифр 00, на выходе элемента «или» никаких сигналов нет (0), когда в данной паре разрядов содержится комбинация 01 (где младшая цифра справа), на выход «или» проходит код множимого c, при наличии комбинации 10 — код $2c$, а при наличии комбинации 11 — код $3c$. Таким образом, каждая такая логическая схема отрабатывает частичное произведение множимого на пару разрядов множителя.

Полученные частичные произведения складываются затем при помощи сумматоров, точно так же как в схеме рис. 4-15. Однако сумматоров теперь вдвое меньше, так как вдвое меньше и частичных произведений. Между сумматорами на рис. 4-16 показано условно по 2 линии задержки (Л. 3.), для того чтобы подчеркнуть, что временная задержка должна быть здесь равна двум тактам.

Ясно, что по количеству оборудования схема рис. 4-16 значительно экономнее, чем схема рис. 4-15. На каждую пару разрядов множителя в ней приходится 3 элемента «и», 1 элемент «или» и 1 последовательный сумматор вместо 2 элементов «и» и 2 последовательных сумматоров в схеме рис. 4-15; следовательно, один из каждой пары последовательных сумматоров, имеющихся в схеме рис. 4-15, как бы заменен здесь одним элементом «и» и одним элементом «или». В отношении скорости выполнения умножения схемы рис. 4-15 и 4-16 равноценны.

Вряд ли нам удалось бы прийти к схеме рис. 4-16, если бы мы удовлетворились только тем объяснением работы предыдущей схемы, которое приводилось в 4.3.2.

На рис. 4-17 показан еще один из возможных вариантов построения множительного устройства, содержащего m последовательных сумматоров[*]). Ни по количеству оборудования, ни по быстродействию схема рис. 4-17 почти не отличается от схемы рис. 4-15. Разница состоит лишь в том, что частичные произведения множимого на цифры множителя суммируются в ней в другом порядке: сначала попарно, затем вновь попарно и т. д. Вариант этот, вероятно, менее популярен, чем предыдущий, потому что построение схемы получается менее стройным, особенно если количество разрядов m не является целой степенью двойки.

[*])По этому типу (но для троичной системы) построено множительное устройство вычислительной машины «Сетунь», созданной Вычислительным центром МГУ.

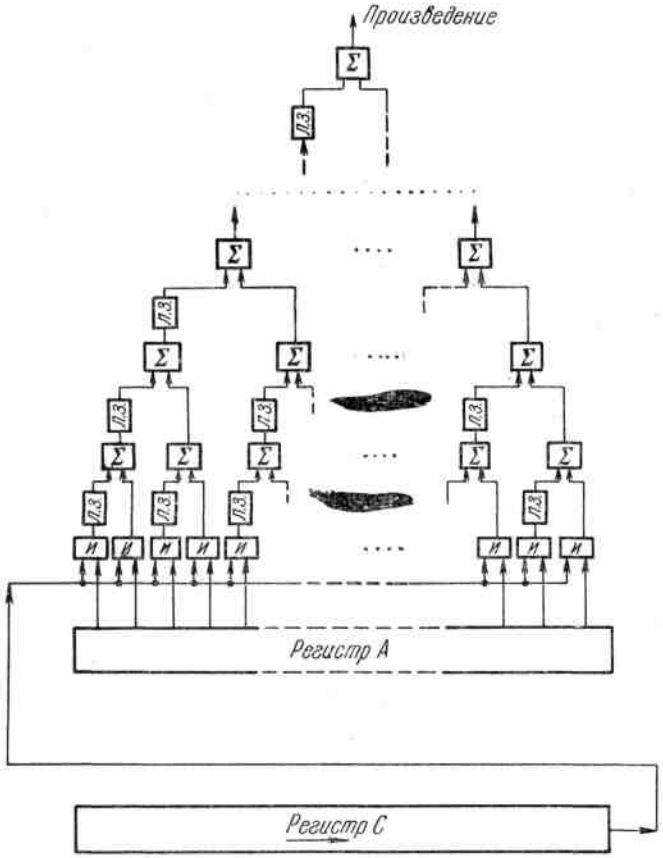


Рис. 4-17. Вариант множительного устройства, содержащего примерно m последовательных сумматоров (построение типа пирамиды).

4.6.3. Упрощенное множительное устройство последовательного типа.

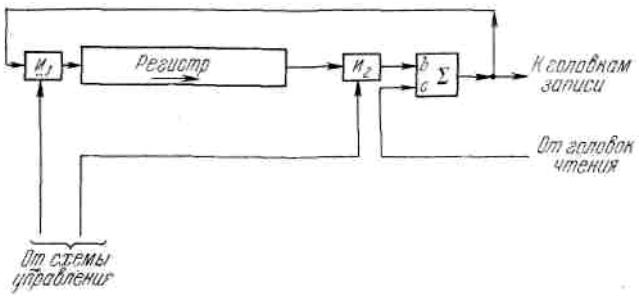


Рис. 4-18. Упрощенное арифметическое устройство для последовательной машины с магнитным барабаном.

Мы рассмотрели сравнительно кратко некоторые аппаратные методы ускорения умножения в последовательных устройствах, уделив им значительно меньше места, чем способам ускорения умножения в устройствах параллельного типа. С более общей точки зрения очевидно, что в тех случаях, когда необходимо любой ценой достигнуть увеличения быстродействия, а вопрос о количестве оборудования играет подчиненную роль, незачем строить схемы последовательного типа. Последние применяются как раз тогда, когда устройство должно быть возможно более экономичным. Иногда при этом ставится обратная задача: достигнуть максимальной экономии в количестве оборудования и лишь по возможности сохранить быстродействие схемы.

В качестве иллюстрации того, что может быть достигнуто на этом пути, на рис. 4-18 показан принцип построения упрощенного множительного устройства для последовательных машин с запоминающим устройством в виде магнитного барабана[*]). Для осуществления описываемого ниже устройства требуется, чтобы количество чисел, размещаемых на одной дорожке барабана, было не меньше, чем количество разрядов в каждом числе (m).

Как видно из рисунка, арифметическое устройство состоит всего из одного регистра с цепями для сдвига вправо, одного последовательного сумматора и нескольких вентилей (элементов «и»). Если задача получения $2m$-разрядных произведений не ставится, то регистр должен содержать только (m + 1) разрядов. Выход регистра через вентиль «$и_2$» соединен с одним из входов сумматора; другой вход сумматора через выходные вентили запоминающего устройства может соединяться с читающими головками магнитного барабана. Выходные сигналы сумматора через вентиль «$и_1$» передаются на вход регистра и через входные вентили запоминающего устройства — на записывающие головки барабана. (В запоминающем устройстве могут быть применены и универсальные головки

[*]) Впервые принцип построения такого устройства был опубликован автором в книге «Арифметические устройства электронных цифровых машин», Физматгиз, 1958.

чтения-записи: для арифметического устройства это несущественно; входные и выходные вентили запоминающего устройства на рисунке не показаны.)

Способ выполнения в рассматриваемом устройстве операции сложения или вычитания очевиден. Прежде всего в регистр нужно поместить первое слагаемое (уменьшаемое). Для этого во время прохождения требуемого числа под головкой чтения в арифметическом устройстве открывается вентиль «$и_1$», запирается вентиль «$и_2$» и выполняются сдвиги вправо в регистре; одновременно открыт соответствующий выходной вентиль запоминающего устройства. Число, прочитываемое головкой чтения, поступает на вход *c* сумматора, складывается с нулем (поскольку вентиль «$и_2$» заперт) и через вентиль «$и_1$» цифра за цифрой вдвигается в регистр. В регистре оно хранится до тех пор, пока к головкам чтения не подойдет второе слагаемое (вычитаемое). Во время прохождения второго числа в арифметическом устройстве вновь открывается вентиль «$и_1$», открывается также вентиль «$и_2$» и выполняются сдвиги вправо в регистре; на вход *c* сумматора через соответствующий выходной вентиль запоминающего устройства подается второе слагаемое или дополнительный код вычитаемого. (Цепи для получения дополнительного кода — инвертор, выполняющий обращение, и вентиль, через который вводится добавочная единица младшего разряда в сумматор,— на рисунке не показаны.) Сумматор производит сложение числа, находящегося в регистре, которое поступает через вентиль «$и_2$», с числом, поступающим с барабана. Через вентиль «$и_1$» результат возвращается в регистр, где он может сохраняться для следующей операции либо для записи на барабан. При выполнении записи из регистра в арифметическом устройстве открыт вентиль «$и_2$» и выполняются сдвиги в регистре, а выходные вентили запоминающего устройства должны быть заперты; число из регистра проходит на вход *b* сумматора, складывается с нулем и далее через соответствующий входной вентиль запоминающего устройства попадает на нужную головку записи; одновременно через вентиль «$и_1$» оно может возвращаться в регистр, если это необходимо.

Таким образом, выполнение сложения и вычитания не требует никакого специального времени: эти операции выполняются одновременно с чтением второго числа с магнитного барабана.

Выполнение умножения, как мы сейчас увидим, потребует времени, равного по длительности трем оборотам барабана. Кроме того, на время выполнения умножения нам придется занять одну дорожку барабана; при выполнении других операций эта дорожка может использоваться как-нибудь иначе.

В течение первого оборота барабана в регистре должно находиться множимое; оно расписывается по окружности барабана столько раз, сколько имеется разрядов в числе (m). Затем в регистр помещается множитель. В течение второго оборота цифры множителя управляют стиранием (перезаписью на «0») расписанного на рабочей дорожке множимого: если младшая цифра множителя есть «0», то первая запись множимого стирается, если «1» — сохраняется; в зависимости от второй цифры стирается или не стирается вторая запись множимого и т. д. В течение третьего оборота в регистре накапливается сумма чисел, сохранившихся на рабочей дорожке барабана после второго оборота; при этом после каждого периода числа в регистре производится один дополнительный сдвиг вправо. К концу этого оборота в нем образуется произведение.

В разделе 5.1.6 мы покажем, как при небольшом усложнении описанного устройства его можно использовать также для выполнения деления с такой же скоростью, как выполняется умножение. Конечно, 3 оборота барабана — это сравнительно большое время для выполнения умножения или деления; однако если вспомнить, что чтение с барабана команды и двух исходных чисел и запись результата займут по времени в среднем 2 оборота барабана (по $^1/_2$ оборота на каждое обращение — при случайном расположении адресов), то это и не такой уж плохой результат. Зато арифметическое устройство отличается едва ли не предельной экономичностью: трудно себе представить, что можно было бы сделать с меньшим количеством оборудования.

4.7. Умножение чисел с учетом алгебраических знаков и чисел с плавающей запятой

На протяжении всех предыдущих разделов настоящей главы мы говорили только о перемножении абсолютных величин чисел, представленных прямым кодом, позиционной системой счисления с естественным порядком весов и **с** фиксированной запятой. Усложнения, с которыми чаще всего приходится встречаться на практике, состоят в том, что исходные числа могут быть и положительными, и отрицательными (причем в некоторых случаях для изображения отрицательных чисел используется дополнительный код), а также в том, что для изображения чисел может применяться система с плавающей запятой. Именно этими случаями мы и займемся в настоящем разделе. При этом все изложение будет вестись применительно к двоичной системе счисления; впрочем, рассматриваемые методы выполнения операций можно без труда обобщить для произвольного основания системы счисления n.

4.7.1. Об умножении чисел с учетом алгебраических знаков.

Если для изображения отрицательных чисел используется прямой код, то учет алгебраических знаков при умножении не связан с какими-либо существенными трудностями. В этом случае обычно с абсолютными величинами чисел операции производятся отдельно, по тем правилам, которые были изложены ранее, и отдельно по общим правилам формируется знак произведения. Затруднения с выполнением умножения возникают тогда, когда для изображения отрицательных чисел применяются обратный или дополнительный коды. Этим вопросом мы сейчас и займемся.

Наиболее естественным на первый взгляд представляется следующий путь: если один или оба сомножителя отрицательны, то до начала умножения нужно превратить изображения их абсолютных величин в прямой код, перемножить затем обычным способом абсолютные величины чисел, отдельно сформировать знак произведения и, если произведение отрицательно, вновь превратить изображение его абсолютной величины в обратный или в дополнительный код. Практически, однако, такой путь может оказаться слишком сложным, особенно при использовании дополнительных кодов.

Представим себе, например, что множительное устройство параллельного типа построено в соответствии с рис. 4-2, *а* (стр. 144), а умножение осуществляется без применения каких-либо методов ускорения в соответствии с 1-м вариантом, описанным в 4.1.2. До начала умножения сомножители размещаются в регистрах *А* и *С*, а регистр *В* погашен. Обращение кода множимого (в регистре *С*) до начала умножения, если это необходимо, можно выполнить без особого труда, поскольку цепи для обращения кодов в регистре *С* имеются так или иначе (для выполнения вычитания, сравнения и других операций); для выполнения аналогичной операции с множителем в регистре *А* такие цепи пришлось бы предусмотреть специально. Если применяются не обратные, а дополнительные коды, то для получения прямого кода нужно произвести не только обращение, но и добавление единицы младшего разряда. С этой целью нужно либо предусматривать специальные цепи в регистрах*А* и *С*, либо передавать по очереди сомножители в регистр *В,* выполнять добавление единицы через сумматор и затем возвращать полученные коды в свои регистры. Еще сложнее будет найти дополнительный код от произведения. Произведение, как мы помним, содержит вдвое больше разрядов, чем каждый из сомножителей, и размещается в двух регистрах: младшая часть — в регистре *А*, старшая часть — в регистре *В.* Нахождение дополнительного кода от него должно было бы состоять из обращения кодов в регистрах *А* и *В* и добавления затем единицы младшего разряда в регистр *А*; при этом возможен случай, когда при добавлении единицы возникнет перенос из старшего разряда регистра *А* в младший разряд регистра *В.* Для выполнения таких операций нужно либо иметь довольно сложные специальные цепи в регистрах *А* и *В,* либо, может быть, выполнить ряд передач между регистрами, использовать для добавления единицы имеющийся сумматор и предусмотреть специальное устройство, которое запомнило бы возможный перенос из старшего разряда младшей части произведения и выполнило бы затем добавление единицы в младший разряд старшей части произведения. Конечно, такое решение слишком громоздко.

Трудности аналогичного рода могут встретиться на этом пути и при использовании других множительных устройств и других методов умножения из числа описанных ранее. Поэтому умножение с учетом алгебраических знаков при использовании дополнительных кодов выполняется обычно одним из искусственных приемов, которые мы рассмотрим ниже. Аналогичные приемы — в несколько упрощенной форме — могут применяться и при использовании обратных кодов для изображения отрицательных чисел, но последующее изложение будет вестись применительно к дополнительным кодам. Для определенности будем полагать, что дополнительный код строится по первому варианту (см. 1.4.2), т. е. что положительные числа содержат **в** разряде знака «0», а отрицательные — «1»; как **и** прежде, будем считать, что запятая фиксирована перед старшим из основных разрядов числа (таким образом, все числа дробные, вес знакового разряда равен — 1 вместо естественного для этого места веса +1**).**

4.7.2. Умножение в дополнительных кодах с двумя корректирующими шагами.

Представим себе для определенности, что множительное устройство построено **в** соответствии с рис. 4-2, *а* (стр. 144) и что умножение абсолютных величин выполняется обычным методом **в** первом варианте (см. 4.1.2). Это предположение делается для удобства последующего изложения, но вообще описываемые ниже способы годятся и при других построениях множительных устройств. Обозначим сомножители через *а* и *с*, а их абсолютные величины—через $|a|$ и $|c|$. Если некоторый сомножитель, скажем *а,* положителен, то в его основных разрядах (без разряда знака) содержится величина $|a|$, если же он отрицателен, то в его основных разрядах содержится величина $1 - |a|$ (дополнение от величины $|a|$ до единицы).

Перемножим, не обращая внимания на алгебраические знаки, числа, содержащиеся в основных разрядах сомножителей; результат этого умножения назовем *псевдопроизведением.* Затем выполним коррекцию таким образом, чтобы из псевдопроизведения получить изображение абсолютной величины искомого произведения. При этом могут встретиться следующие 4 случая:

1) если $a \geq 0$ и $c \geq 0$, то псевдопроизведение равно $|a|\ |c|$, т. е. сразу, без всякой коррекции, равно абсолютной величине произведения;

2) если $a \geq 0$, а $c < 0$, то псевдопроизведение равно

$$|a|\ (1 - |c|) = |a| - |a|\ |c|;$$

настоящее произведение по абсолютной величине равно $|a|\ |c|$, но, поскольку оно в данном случае отрицательно, его основные разряды должны содержать дополнение от $|a|\ |c|$ до единицы, т. е. $1 - |a|\ |c|$; для получения этой величины к псевдопроизведению нужно добавить $1 - |a|$;

3) если $a < 0$, а $c \geq 0$, то псевдопроизведение равно

$$(1 - |a|)\ |c| = |c| - |a|\ |c|;$$

для получения правильного представления абсолютной величины произведения (величины $1 - |a|\ |c|$) к псевдопроизведению нужно добавить $1 - |c|$;

4) если $a < 0$ и $c < 0$, то псевдопроизведение равно

$$(1 - |a|)(1 - |c|) = 1 - |a| - |c| + |a|\,|c|;$$

произведение при этом положительно, и в изображении его абсолютной величины мы должны получить $|a|\,|c|$; для этого к псевдопроизведению нужно добавить $|a|$ и $|c|$, в результате чего получим величину $1 + |a|\,|c|$; так как эта величина больше единицы, а счет идет по модулю 1, в основных разрядах регистра результата останется в действительности величина $|a|\,|c|$.

Итак, можно сформулировать следующее правило выполнения коррекции:

1) если $a < 0$, то добавить к псевдопроизведению дополнительный код от изображения числа c (когда $c \geq 0$, его изображение представляет собой величину $|c|$, а дополнительный код от изображения — величину $1 - |c|$; когда $c < 0$, изображение представляет собой величину $1 - |c|$, а дополнительный код от изображения — величину $|c|$;

2) если $c < 0$, то добавить к псевдопроизведению дополнительный код от изображения числа a.

Таким образом, в конце умножения может потребоваться максимум два, а в среднем один корректирующий шаг.

Примеры.

1) (—7/16) × 11/16 = —77/256,
—7/16 = 1.1001 (дополнительный код),
+ 11/16 = 0.1011,
—77/256= 1.10110011 (дополнительный код).

Перемножим сначала величины, представленные основными разрядами сомножителей, не обращая внимания на знаки чисел:

```
  .1001
× .1011
--------
  1001
 1001
1001
--------
.01100011  (псевдопроизведение).
```

Поскольку первый сомножитель отрицателен, добавим дополнительный код от изображения второго сомножителя:

```
  .01100011
+ .0100      — обратный код от .1011
       1     — единица для получения дополнительного
                                              кода
-----------
  .10110011
```

В результате получили правильное изображение абсолютной величины произведения.

2) (—7/16) × (—11/16) =+77/256,
—7/16 = 1.1001 (дополнительный код),
—11/16 = 1.0101 (дополнительный код),
+77/256 = 0.01001101.

Перемножим величины, представленные основными разрядами в изображениях сомножителей:

```
  .1001
× .0101
--------
  1001
 1001
--------
.00101101 (псевдопроизведение).
```

Поскольку первый сомножитель отрицателен, добавим дополнительный код от изображения второго сомножителя:

```
  .00101101
+ .1010      — обратный код от .0101
       1     — единица для получения допол-
               нительного кода
-----------
  .11011101
```

Поскольку второй сомножитель отрицателен, добавим дополнительный код от изображения первого сомножителя:

```
  .11011101
+ .0110      — обратный код от .1001
       1     — единица для получения допол-
               нительного кода
-----------
  .01001101
```

(единица переполнения не принимается во внимание, так как счет идет по модулю 1). В результате получено правильное изображение абсолютной величины произведения.

Обратим внимание на следующую важную деталь. К концу умножения псевдопроизведение занимает 2 регистра арифметического устройства —*A* и *B* (см. рис. 4-2, *а* на стр. 144). Младшие его разряды, находящиеся в регистре *A,* никакой корректировки не требуют: поскольку исходные числа содержат вдвое меньше разрядов, чем псевдопроизведение, при добавлении дополнительного кода от изображения какого-либо сомножителя младшие разряды псевдопроизведения не могут измениться; последнее обстоятельство легко усмотреть и в рассмотренных примерах, где младшие 4 разряда в псевдопроизведении получаются сразу правильными. Поскольку старшие разряды псевдопроизведения находятся в регистре *B,* а множимое — в регистре *C,* добавление дополнительного кода от изображения *c* (в том случае, когда $a < 0$) выполняется очень легко. Труднее выполнить (при $c < 0$) добавление к псевдопроизведению дополнительного кода от множителя *a:* к концу умножения величина множителя вообще не сохраняется, а регистр *A* занят младшими разрядами псевдопроизведения.

В связи с этим последний корректирующий шаг — добавление дополнительного кода от изображения множителя в том случае, когда множимое отрицательно, выполняется отчасти в ходе умножения. Делается это следующим образом. В обычном процессе умножения множимое (его абсолютная величина $|c|$) добавляется к предыдущей сумме частичных произведений каждый раз, как очередная цифра множителя есть 1; если очередная цифра множителя есть 0, то к предыдущей сумме частичных произведений не добавляется ничего. Пусть теперь в последнем случае вместо того, чтобы ничего не добавлять, будем добавлять знаковый разряд множимого. Этот знаковый разряд каждый раз попадает в знаковый разряд регистра *B* (слева от запятой), а потом участвует в сдвигах в регистре *B* на общих основаниях. В конце операции (после последнего сдвига) знак *B* нужно будет установить по общему правилу.

Если множимое положительно, то его знаковый разряд содержит цифру «0», и ничего лишнего к сумме частичных произведений фактически не добавляется. Если же множимое отрицательно и его знаковый разряд содержит цифру «1», то в результате описанного процесса окажется, что к произведению добавлен обратный код от изображения множителя.

Действительно. Если, например, *m*-й (младший) разряд изображения множителя содержит «0», то в первом цикле умножения будет добавлена лишняя единица в знаковый разряд регистра *B*; после *m* циклов умножения, в каждом из которых выполняется один сдвиг вправо в регистре *B,* эта единица передвинется в *m*-й разряд произведения. Вообще если «0» содержится в *i*-м разряде изображения множителя, то лишняя единица добавляется в знаковый разряд регистра *B* в (*m*—*i*)-м цикле умножения; после этого до конца умножения остается еще *i* сдвигов, в результате которых эта единица передвинется в *i*-й разряд произведения. Лишние единицы будут, таким образом, добавлены во все те разряды произведения, в которых изображение множителя содержит нули; иными словами, к произведению будет добавлен обратный код от изображения множителя.

Далее — в конце умножения — остается только добавить к псевдопроизведению еще единицу *m*-го разряда (в том случае, когда множимое отрицательно). В результате к псевдопроизведению будет добавлен дополнительный код изображения множителя, что и требуется, когда множимое отрицательно.

Повторим с этим видоизменением второй из примеров, рассмотренных на стр. 198.

Пример. (—7/16) X (—11/16) = + 77/256,
— 7/16 = 1.1001,
— 11/16 = 1.0101,
77/256 = 0.01001101.

Вместе с перемножением величин, представленных основными разрядами сомножителей, одновременно частично выполняется коррекция:

```
  1.1001   — множимое
×  .0101   ← основные разряды множителя
---------
   1001
  10000
  1001
10000
---------
11001101   — псевдопроизведение с добавленным обратным кодом от изображения множителя
```

Поскольку множимое отрицательно, добавляем еще одну единицу *m*-го разряда (чтобы к псевдопроизведению был добавлен не обратный, а дополнительный код от изображения множителя):

```
  .11001101
+        1
-----------
  .11011101
```

Второй корректирующий шаг выполняется точно так же, как и прежде, и дает тот же результат:

```
  .11011101
+ .0110    — обратный код изображения множимого (.1001)
       1   — единица для получения дополнительного код а
  --------
  .01001101
```

4.7.3. Вариант умножения в дополнительных кодах с одним корректирующим шагом.

Как и в 4.7**.2,** мы будем для определенности полагать, что множительное устройство построено в соответствии с рис. 4-2, *а* на стр. 144, хотя описываемый ниже способ умножения в дополнительных кодах годится и в других случаях.

Основная идея состоит в том, что в главной части процесса умножения по-прежнему будут использоваться только разряды, представляющие абсолютную величину множителя (без разряда алгебраического знака), но во всех суммированиях, выполняемых по ходу умножения, множимое *с* будет участвовать полностью, так что при любом суммировании учитываются знаки чисел, находящихся в регистрах *В* и *С.* Так как основные разряды в изображении множителя *а* представляют либо величину $|a|$ (если $a \geq 0$), либо величину $1 - |a|$ (если $a < 0$), то результат этого процесса (псевдопроизведение) равен либо

$$c\,|a| = ca \quad (\text{если } a \geq 0)$$

либо

$$c\,(1 - |a|) = c - c\,|a| = c + ca \ (\text{если } a < 0).$$

В первом случае получаем сразу готовое произведение, во втором случае (если $a < 0$) нужно выполнить один корректирующий шаг — вычитание из псевдопроизведения множимого *с.* При выполнении вычитания обычным образом учитываются алгебраические знаки псевдопроизведения и множимого. Таким образом, в конце умножения теперь требуется максимум один корректирующий шаг**,** а в среднем $^1/_2$ корректирующего шага — вдвое меньше, чем в том способе, который рассмотрен в 4.7.2.

Трудность, которая возникает при осуществлении этого алгоритма, состоит в следующем. Как окончательное произведение, так и псевдопроизведение, получаемое по ходу дела, по своим абсолютным величинам непременно меньше единицы. Для случая $a \geq 0$ это очевидно сразу: поскольку $c < 1$ и $a < 1$, то и псевдопроизведение, равное непосредственно окончательному произведению *ас,* меньше единицы.

Для случая $a < 0$ псевдопроизведение равно $c + ca,$ но, так как $a < 0$, знаки слагаемых *с* и *са* противоположны; кроме того (так как $|a| < 1$), должно выполняться неравенство $|c| > |ca|$; следовательно, псевдопроизведение имеет тот же знак, что и *с,* а по абсолютной величине меньше $|c|$, т. е. и подавно меньше единицы. Однако по ходу умножения, когда в некотором цикле множимое с складывается с предыдущей суммой частичных произведений в *В,* результат может оказаться по абсолютной величине больше единицы.

При этом цифра в знаковом разряде регистра *В* не будет соответствовать алгебраическому знаку числа, а является просто его старшей значащей цифрой. Вслед за суммированием всегда выполняется сдвиг вправо, в результате которого число в регистре *В* становится по абсолютной величине меньше единицы. Но если в прежних способах умножения знак *В* не играл роли (потому что все равно мы оперировали только с абсолютными величинами чисел), то теперь при сдвиге необходимо установить правильный знак в регистре *В.*

Затруднение это преодолевается сравнительно легко. В начале умножения регистр *В* гасится, т. е. устанавливается в нуль. Число, которое затем появляется в нем, является результатом многократного добавления множимого *с* (с определенными сдвигами). Следовательно, в ходе формирования псевдопроизведения, пока не выполнен последний корректирующий шаг (если он необходим), число в регистре *В* либо равно нулю (до первого суммирования), либо совпадает по знаку с множимым *с.* Поэтому при сдвиге *В* вправо нужно каждый раз устанавливать разряд знака *В* в соответствии с разрядом знака регистра С, а до первой выдачи суммы в регистр *В* (пока в регистре *В* находится нуль) сдвиги *В* не производить.

Попробуем выполнить указанным методом второй из примеров, разобранных на стр. 198.

П р и м е р . $(-7/16) \times (-11/16) = +77/256,$
$c = -7/16 = 1.1001,$
$a = -11/16 = 1.0101,$
$+77/256 = 0.01001101.$

Основная часть умножения:

```
  1.1001     — множимое c
× .0101      — основные разряды множителя a
-------
  0.0000     — начальное состояние B
+ 1.1001     — частичное произведение c на младшую цифру a
-------
  1.1001
-------
  1.11001    — результат первого сдвига B
-------
  1.111001   — результат второго сдвига B
+ 1.1001     — частичное произведение c на третью справа цифру a
-------
  1.011101
-------
  1.1011101  — результат третьего сдвига B
-------
 1.11011101 — результат последнего сдвига B (псевдопроизведение)
```

Поскольку множитель отрицателен, необходимо выполнить коррекцию — вычесть из псевдопроизведения множимое с:

```
  — 1.11011101
    1.1001
-------------
    0.01001101
```

Читателю предлагается самостоятельно разобрать несколько примеров на применение этого метода при разных сочетаниях знаков.

На смысл корректирующего шага, выполняемого в конце умножения при использовании описанного метода, интересно взглянуть еще и с другой точки зрения. При построении дополнительного кода (см. первый вариант в 1.4.2) мы сохранили позиционный характер системы изображения чисел, но разряду алгебраического знака, расположенному на месте первого разряда в целой части числа, приписали вместо естественного веса «+1» вес «—1». Положительные числа содержат в этом разряде цифру «0». В частности, если множитель положителен, то частичное произведение множимого c на знаковый разряд множителя есть 0. В отрицательных числах знаковый разряд содержит 1; поэтому частичное, произведение множимого c на знаковый разряд отрицательного множителя равно $—c$. Вычитание из псевдопроизведения величины с, выполняемое в том случае, когда множитель отрицателен, представляет собой фактически добавление частичного произведения множимого на знаковый разряд множителя к предыдущей сумме частичных произведений. В результате и получается правильное произведение. Такое толкование смысла корректирующего шага будет полезно в последующем изложении.

4.7.4. Умножение в дополнительных кодах путем последовательного преобразования множителя*).

Как и в способе, изложенном в 4.7.3, множимое будет участвовать в операции целиком, включая разряд алгебраического знака, а количество циклов умножения равно $m + 1$, где m — количество основных разрядов множителя (в предыдущем способе $(m + 1)$-м циклом умножения можно было считать корректирующий шаг). Однако правило умножения будет сформулировано таким образом, чтобы оно было одинаковым для всех циклов умножения, в том числе и для последнего, $(m + 1)$-го цикла, который не выделяется в качестве корректирующего шага, выполняемого по специальным правилам и требующего некоторых специальных устройств в схеме управления.

В ходе умножения множитель преобразуется в форму записи цифрами —1, 0, +1 (ср. с 4.2.2); соответственно либо производится вычитание множимого из суммы частичных произведений, либо не производится ни сложения, ни вычитания, либо множимое добавляется к сумме частичных произведений. При этом правило преобразования должно выглядеть так:

1) если данная цифра непреобразованного множителя совпадает с соседней справа его цифрой, то цифра преобразованного множителя есть 0;

2) если данная цифра непреобразованного множителя есть 0, а соседняя справа его цифра есть 1, то соответствующая цифра преобразованного множителя есть 1;

3) если данная цифра непреобразованного множителя есть 1, а соседняя справа его цифра есть 0, то соответствующая цифра преобразованного множителя есть —1.

Обозначим цифры непреобразованного множителя через $a_0, a_1,..., a_{m-1}, a_m$, где a_0 — цифра в разряде алгебраического знака. Как известно (см. 1.4.2), при построении дополнительного кода по первому варианту число a может быть записано в виде

*) Этот способ известен в литературе под названием *метода Booth и Booth* — см., например, БутЭ. и Бут К., Автоматические цифровые машины, Физматгиз, 1959, стр. 64—65.

$$a = — 1 + a_0 + 2^{-1}a_1 + 2^{-2}a_2 + \ldots + 2^{-(m-1)}a_{m-1} + 2^{-m}a_m.$$

Если обозначить цифры преобразованного множителя через α_0, α_1, α_2,..., α_{m_1}, α_m, то приведенное выше правило преобразования можно свести к равенству

$$\alpha_i = a_{i+1} — a_i.$$

Отсюда видно, что число

$$+ 1 \cdot \alpha_0 + 2^{-1}\alpha_1 + 2^{-2}\alpha_2 + \ldots + 2^{-(m-1)}\alpha_{m-1} + 2^{-m}\alpha_m$$

на самом деле равно

$$+ 1 \cdot (a_1 — a_0) + 2^{-1}(a_2 — a_1) + 2^{-2}(a_3 — a_2) + \ldots + 2^{-(m-1)} \times (a_m — a_{m-1}) + 2^{-m}(0 — a_m)$$

(справа от самой младшей цифры a_m всегда находится 0, т. е, $a_{m+1} = 0$). Раскрывая скобки, убедимся в том, что

$$+ 1 \cdot \alpha_0 + 2^{-1}\alpha_1 + 2^{-2}\alpha_2 + \ldots + 2^{-m}\alpha_m = a.$$

Таким образом, в преобразованном множителе все $m+1$ разрядов имеют естественные веса ($+1$, $+2^{-1}$, $+2^{-2}$,... ...,$+2^{-(m-1)}$,$+2^{-m}$); со знаковым разрядом можно обращаться точно так же, как со всеми остальными разрядами.

При практическом осуществлении описанного метода трудность может состоять в том, что множимое то добавляется к сумме частичных произведений, то вычитается из нее, и поэтому алгебраический знак суммы частичных произведений не обязательно совпадает со знаком множимого. В любой момент времени он определяется старшей из уже использованных значащих цифр преобразованного множителя: если эта цифра есть +1,то знак суммы частичных произведений совпадает со знаком множимого, если —1, то противоположен. Если, как и прежде, умножение выполняется по схеме рис. 4-2, *а,* то при сдвиге *B* вправо цифра, которая должна вдвигаться в разряд знака, может определяться по указанному правилу. Более простое решение может состоять в том, чтобы иметь в регистре *B* и в сумматоре один дополнительный разряд, который являлся бы разрядом знака.

Хотя здесь, как и при использовании логических методов ускорения умножения, множитель преобразуется к представлению цифрами + 1, 0, —1, однако никакого выигрыша в среднем или максимальном количестве суммирований-вычитаний по сравнению с обычным пропуском тактов суммирования при равенстве нулю очередной цифры множителя не получается. В частности, максимальное количество суммирований-вычитаний, равное количеству разрядов в множителе $m + 1$, получится в том случае, когда множитель имеет вид ....01010101.

Попробуем выполнить описанным способом один из примеров, рассмотренных ранее.

П р и м е р . (—11/16) × (—7/16) = + 77/256, *c* = —11/16 = 1.0101, *a* = —7/16 = 1.1001, + 77/256 = 0.01001101.

Преобразование множителя (выполняемое шаг за шагом в каждом цикле умножения) дает

$$\tilde{a} = 0.1\bar{0}\bar{1}1.$$

Далее следует обычный процесс умножения, в ходе которого со всеми цифрами $\tilde{a}$ мы обращаемся одинаково. В скобках указаны те цифры, которые при сдвигах *B* вправо вдвигаются в его знаковый разряд; эти цифры определяются по описанному выше правилу: поскольку знак *c* есть 1 (минус), единица в указанный разряд вдвигается в том случае, когда последняя из использованных значащих цифр $\tilde{a}$ есть 1 (т. е. когда последним выполнялось сложение), а нуль вдвигается в том случае, когда последняя из использованных значащих цифр $\tilde{a}$ есть 1 (т. е. когда последним выполнялось вычитание)

```
  1.0101      — множимое c
× 0 1̄011̄     — преобразованный множитель ã
-----------
  0 0000      — начальное состояние B
— 1 0101      — частичное произведение c на младшую цифру ã
              (на — 1)
-----------
  0 1011
-----------
  (0)01011    — результат первого сдвига B вправо
+ 1 0101      — частичное произведение c на вторую справа цифру ã
              (на + 1)
-----------
  1 10101
-----------
  (1)110101   — результат второго сдвига B вправо
-----------
  (1)1110101  — результат третьего сдвига B вправо
— 1 0101      — частичное произведение c на четвертую справа
              цифру ã (на — 1)
-----------
  0 1001101
-----------
  (0)01001101 — результат четвертого сдвига B вправо
```

Полученный результат, как видим, не требует никакой дополнительной коррекции.

Читателю предлагается самостоятельно убедиться в том, что так называемый усовершенствованный метод Booth и Booth (см. сноску на стр. 154), который рассматривался нами в 4.2.2 в качестве наиболее сильного логического метода ускорения умножения, сохраняет все достоинства описанного здесь метода в части умножения в дополнительных кодах.

4.7.5. Способ умножения в дополнительных кодах, основанный на расширении разрядной сетки сомножителей.

Рассматривая примеры, приведенные в 4.7.2, нетрудно заметить, что при любом сочетании знаков младшие четыре цифры произведения всегда получаются сразу правильными, и никакой дополнительной коррекции в них не требуется. Старшие же разряды получаются правильными только в том случае, когда оба сомножителя положительны; если хотя бы один из сомножителей отрицателен, то требуется дополнительная коррекция.

Это наводит на следующую мысль.

Если сомножители содержат разряды с весами от 2^{-1} до 2^{-m}, то произведение содержит разряды с весами от 2^{-1} до 2^{-2m}. При этом m младших разрядов произведения с весами от $2^{-(m+1)}$ до 2^{-2m} зависят только от цифр сомножителей, имеющих веса от 2^{-1} до $2^{-m;}$ если бы в сомножителях появились какие-нибудь новые старшие цифры с весами 2^0, 2^1,..., то от этого младшие разряды произведения не изменились бы. Иное дело m старших разрядов произведения с весами от 2^{-1} до 2^{-m}; если бы в одном из сомножителей появилась бы, скажем, цифра с весом 2^0, то при умножении на цифру другого сомножителя с весом 2^{-m} получилась бы величина с весом $2^0 2^{-m} = 2^{-m}$, которая изменила бы соответствующий разряд произведения, и т. д. До тех пор, пока мы имеем дело только с положительными числами, разряды с весами 2°, 2^1, ... всегда содержат нули; не принимая их во внимание мы никакой ошибки не вносим. Если же появляются отрицательные сомножители, представленные дополнительным кодом, то источником ошибок в псевдопроизведении является, может быть, именно то обстоятельство, что мы пренебрегаем цифрами старших разрядов.

Попробуем поэтому поступить следующим образом. Будем считать, что знаковый разряд имеет свой естественный вес $+ 2^\circ = 1$; слева от него припишем к каждому из сомножителей еще m разрядов с естественными весами 2^1, 2^2, ..., 2^m, а относительно разряда знака будем полагать, что он находится еще левее, чем старший из добавочных разрядов и имеет вес $— 2^{m+1}$. Ясно, что поскольку сомножители по абсолютной величине все-таки меньше единицы, то в положительных числах все добавочные разряды будут содержать нули, а в отрицательных числах — единицы. Произведение таких $(2m + 1)$-разрядных чисел содержит вообще $4m + 2$ разрядов, из которых младшие $2m + 1$ разрядов должны получиться сразу правильными без всякой дополнительной коррекции; только эти $2m + 1$ разрядов нас в сущности и интересуют.

Примеры.

1) $(—7/16) \times 11/16 = —77/256$,
$— 7/16 = 1.1001$,
$+ 11/16 = 0.1011$,
$— 77/256 = 1.10110011$.

Дополним каждый из сомножителей четырьмя (поскольку $m = 4$) цифрами слева: отрицательный—четырьмя единицами, положительный — четырьмя нулями — и выполним умножение обычным способом:

```
          11111.1001
        × 00000.1011
        ------------
          11111.1001
         111111 001
        11111100 1
      ---------------
      0000010101101 10011
```

Младшие разряды произведения получились сразу правильными (они выделены жирным шрифтом). Старшие разряды нас не интересуют.

2) $11/16 \times (—7/16) = —77/256$

```
          00000.1011
        × 11111.1001
        ------------
          00000 1011
         00000101 1
        000001011
       000001011
      000001011
     000001011
    000001011
   ---------------
   000001010110110011
```

Результат тот же, что прежде.

3) $(—7/16) \times (—11/16) = +77/256$,
$—7/16 = 1.1001$,
$—11/16 = 1.0101$,
$+77/256 = 0.01001101$,

Поступая аналогично предыдущему, получим

```
          11111.1001
        × 11111.0101
        ------------
            11111 1001
           1111110 01
          111111001
         111111001
        111111001
       111111001
      111111001
      ----------------------
      11110111000100 1101
```

Правильность приведенного алгоритма можно проверить и более строго. Предположим, что знаковый разряд имеет свой естественный вес +1. При этом изображение некоторого сомножителя, скажем *a,* содержит величину $|a|$, если $a \geq 0$, и величину $2 - |c|$, если $a < 0$ (дополнение от величины $|a|$ до единицы плюс единица в знаковом разряде: $(1 - |a|) + 1 = 2 - |a|$). Когда изображение отрицательного сомножителя, скажем *a,* дополняется *m* единицами слева (в разрядах с весами от 2^1 до 2^m), то к нему тем самым добавляется величина $2^{m+1} - 2$, в результате чего получается величина $2^{m+1} - 2 + (2 - |a|) = 2^{m+1} - |a|$ Таким образом, если оба сомножителя *(a* и с) отрицательны, то по описанному выше методу выполняется умножение

$$(2^{m+1} - |a|) \times (2^{m+1} - |c|),$$

если же, например, $a < 0$, $c \geq 0$, то выполняется умножение

$$(2^{m+1} - |a|) \times |c|.$$

В первом случае в результате получим

$$2^{2m+2} - 2^{m+1}(|a| + |c|) + |a|\,|c|.$$

Поскольку $|a|$ и $|c|$ содержат разряды с весами не меньше 2^{-m}, величина 2^{m+1} $(|a| + |c|)$ есть целое четное число. Отбрасывая в произведении все разряды слева от знакового (с весами 2^1 и старше), мы фактически ведем счет по модулю 2.

Поэтому полученная величина равна просто $|a|\,|c|$; так как $|a| < 1$ и $|c| < 1$, то и $|a|\,|c| < 1$, слева от запятой находится 0, как это и требуется для положительного числа.

Во втором случае ($a < 0$, $c \geq 0$) в результате умножения получим

$$2^{m+1}|c| - |a|\,|c| \equiv 2 - |a|\,|c| \pmod 2.$$

(Подобно предыдущему, член $2^{m+1}|c|$ представляет собой целое четное число.) Величина $2 - |a|\,|c|$ есть изображение отрицательного произведения *ac*; так как $|a| < 1$ и $|c| < 1$, то $|a|\,|c| < 1$, а $2 - |a|\,|c| > 1$, и слева от запятой (в знаковом разряде) произведения получается цифра 1. Ясно, что непосредственное применение описанного метода в большинстве случаев привело бы к значительному увеличению времени выполнения умножения, а в параллельных устройствах потребовало бы еще и значительного увеличения количества оборудования. (Удвоение параллельного сумматора и регистра *B.)* Иногда, однако, описанный способ оказывается весьма удобным. Например, его удобно частично использовать в множительном устройстве, описанном в 4.6.2 (см. рис. 4-15 на стр. 192). В 4.6.2 рассматривалось только умножение абсолютных величин чисел, представленных прямым кодом. В действительности, поскольку необходимо и умножение в дополнительных кодах, начало цепочки сумматоров должно быть устроено несколько сложнее — так, как это изображено на рис. 4-19.

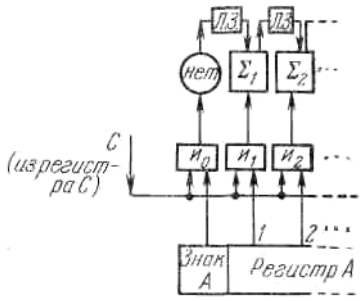


Рис. 4-19. Усложнение схемы рис, 4-15 (стр. 460) для работы с дополнительным кодами.

Элемент «и $_0$», управляемый триггером разряда знака множителя, и следующий за ним инвертор (элемент «нет») вырабатывают частичное произведение множимого с на знаковый разряд множителя; благодаря наличию инвертора учитывается, что вес знакового разряда есть—1:если в разряде знака регистра *A* стоит единица, то сумматор Σ_1 производит фактически не добавление, а вычитание множимого из суммы остальных частичных произведений. Таким образом, алгебраический знак множителя учитывается так, как было описано в 4.7.3. Что же касается алгебраического знака множимого с, то его удобно учесть путем расширения разрядной сетки множимого. С этой целью в течение последних *m* тактов умножения из регистра *C* должны выдаваться либо все время нули (если $c \geq 0$), либо все время единицы (если $c < 0$). Напомним, что в разделе 4.6.2, когда мы имели дело только с прямыми кодами, мы предполагали в течение *m* последних тактов всегда выдавать из регистра *C* нули.

Возможно, что описанные здесь идеи могут оказаться полезными и в других случаях.

4.7.6. Об умножении чисел с плавающей запятой.

Пусть имеем два числа с плавающей запятой:

$$\alpha = 2^a A$$

и

$$\gamma = 2^c C,$$

где a и c — порядки чисел, A и C — мантиссы.

Если числа нормализованы, то $\frac{1}{2} \le |A| < 1$ и $\frac{1}{2} \le |C| < 1$.

Произведение этих чисел равно

$$\alpha\gamma = 2^{a+c} AC,$$

откуда видно, что порядок произведения равен сумме порядков сомножителей, а мантисса произведения — произведению их мантисс. Правило это легко реализуется и не требует каких-либо дополнительных пояснений.

Некоторые тонкости связаны с нормализацией результата. Если | A | и | C | заключены в указанных пределах, то мантисса произведения находится, очевидно, в пределах $\frac{1}{4} \le |AC| < 1$. Так как произведение тоже должно быть получено в нормальной форме, то в случаях, когда $|AC| < \frac{1}{2}$, необходимо удвоить мантиссу результата (сдвинуть ее влево на 1 разряд) и одновременно вычесть единицу из порядка результата. Во многих случаях осуществление сдвига влево мантиссы результата представляет определенные трудности. Если, например, умножение мантисс выполняется по схеме рис. 4-2,*а*, то младшая часть результата получается в регистре A, старшая —в регистре B; для выполнения сдвига влево нужны были бы специальные цепи сдвига в регистрах A и B и цепь передачи старшей цифры из регистра A в младший разряд регистра B. Проще в подобных случаях вместо сдвига влево мантиссы результата блокировать сдвиг вправо в последнем цикле умножения мантисс.

Некоторые тонкости связаны также с суммированием порядков. При суммировании порядков может получиться отрицательный или положительный результат, превышающий по абсолютной величине максимально допустимую величину порядка. В первом случае результат умножения считается равным нулю, во втором случае необходимо либо остановить вычисления, либо выработать специальный сигнал переполнения. Нужно, однако, учесть, что, когда при нормализации результата выполняется вычитание единицы в порядках, могут, с одной стороны, выявиться новые случаи получения нуля, с другой стороны, может оказаться, что переполнение, получившееся при сложении порядков сомножителей, на самом деле не имеет места. В интересах упрощения схемы управления последний случай иногда выпускается из рассмотрения, так что остановка вычислений или выработка сигнала переполнения в некоторых случаях происходят и при отсутствии действительного переполнения; это приводит лишь к незначительному сужению диапазона допустимых чисел.

5. Деление, извлечение квадратного корня и другие операции

Операции, рассматриваемые в настоящей главе, не играют обычно той решающей роли, которая при проектировании арифметического устройства принадлежит сложению, вычитанию и умножению. Иногда даже универсальные вычислительные машины строятся вообще без деления. Извлечение квадратного корня предусматривается в качестве отдельной операции большей частью только в специализированных машинах, например в машинах, с помощью которых предполагается выполнять преобразование пространственных координат из одной системы в другую.

Решение вопроса о целесообразности наличия той или иной операции в универсальной машине во многих случаях затруднительно. Например, в отношении операции деления можно было бы рассуждать так. Если деление не предусмотрено в качестве отдельной операции, то оно может осуществляться путем итерационного процесса, включающего сложения, вычитания и умножения, по специальной подпрограмме; такая подпрограмма содержит, может быть, 15—20 операций (что зависит от необходимой точности, системы команд и других обстоятельств). Полагая, что деление будет встречаться в среднем один раз на 20 операций (5% делений из общего числа операций), найдем, что отсутствие деления снизило бы эффективную скорость машины примерно вдвое: вместо каждых 20 операций пришлось бы выполнять 35—40 операций. С другой стороны, программисты, долго работающие на машинах без деления, утверждают, что деление в действительности необходимо значительно реже. Точно так же программисты, работающие на специализированной машине, в которой в качестве отдельной операции было предусмотрено извлечение квадратного корня, высказывали мнение, что эта операция и в универсальных машинах оказалась бы полезной значительно чаще, чем это представляется сейчас.

Как правило, при осуществлении в качестве отдельных операций деления, извлечения квадратного корня и т. п. ставятся в основном следующие задачи:

1) добиться того, чтобы данная операция выполнялась существенно быстрее, чем по подпрограмме;

2) возможно полнее использовать оборудование, предназначенное для выполнения сложения, вычитания и умножения и, насколько удастся, обойтись минимальным количеством каких-либо специальных устройств.

Вопрос о выборе списка операций рассматривается также в разделе 5.2.4.

5.1. Деление

5.1.1. Два основных метода выполнения деления в двоичной системе и два варианта их осуществления.

Мы рассмотрим сейчас методы выполнения деления чисел с фиксированной запятой или мантисс нормализованных чисел с плавающей запятой. Будем считать пока, что в любом случае все числа представлены

прямыми кодами.

1°. Если обычный способ деления «углом», хорошо известный для десятичной системы, изложить применительно к двоичной системе, то получится следующее правило выполнения деления *(первый способ выполнения деления, первый вариант):*

(а) сравнить делимое с делителем; если делимое больше делителя или равно ему, то старшая цифра частного, относящаяся к разряду целых, равна единице, а первый остаток равен разности делимого и делителя; если делимое меньше делителя, то старшая цифра частного есть нуль, а первый остаток равен делимому. (Для дальнейшего важно, чтобы первый остаток, полученный в результате данного шага, не был больше делителя. Если речь идет о мантиссах нормализованных чисел с плавающей запятой, то это условие выполняется всегда, поскольку мантиссы делимого и делителя заключены в интервале $\left[\frac{1}{2}, 1\right)$ и, следовательно, отличаются друг от друга менее чем вдвое. Если числа ненормализованы, то в случае, когда делимое превышает делитель более чем вдвое, первый остаток получится больше делителя; но для чисел с фиксированной запятой вообще любое превышение делимого над делителем соответствует случаю переполнения, когда результат деления должен получиться больше единицы, и продолжать деление не имеет смысла, деление же ненормализованных чисел с плавающей запятой мы не рассматриваем.)

(б) удвоить предыдущий остаток;

(в) сравнить удвоенный предыдущий остаток с делителем; образовать очередную цифру частного и новый остаток по правилам, аналогичным (а);

(г) повторять (б) и (в) до тех пор, пока не будут получены все необходимые цифры частного.

П р и м е р . 15/32 : 3/4 = 5/8 (все числа с пятью двоичными знаками),

15/32 = .01111,
3/4 = 24/32 = .11000,
5/8 = 20/32 = .10100.

Удвоение 1-го остатка	.01111 — .11110 .11000	.11000 0.10100...
2-й остаток	.00110	
удвоение 2-го остатка (3-й остаток)	.01100	
удвоение 3-го остатка	— .11000 .11000	
4-й остаток	.00000	
удвоение 4-го остатка (5-й остаток)	.00000	
удвоение 5-го остатка (6-й остаток)	.00000	
. . . . .		

Описанный алгоритм можно реализовать, например, в схеме, изображенной на рис. 5-1,*а*.

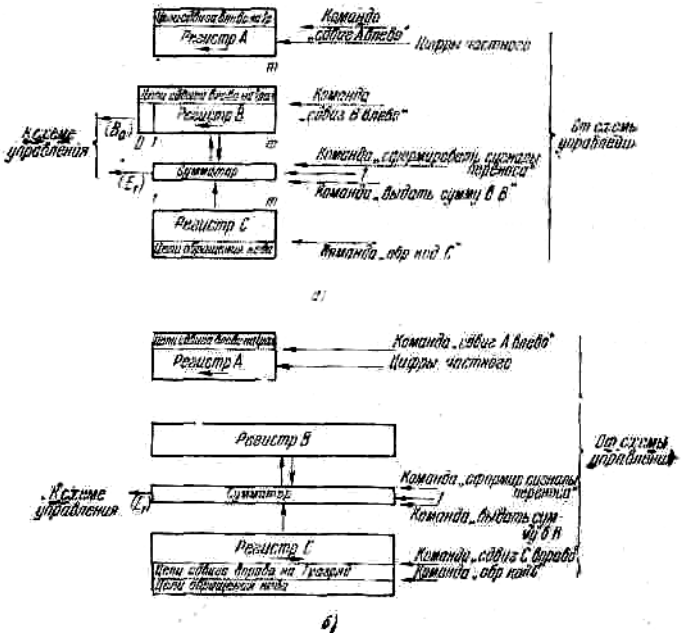


Рис. 5-1. Два варианта осуществления деления: *а)* 1-й вариант;

б) 2-й вариант.

Устройство состоит из трех регистров (*A*, *B* и *C*) и параллельного сумматора, включенного между регистрами *B* и *C* так, что входы сумматора соединены с регистрами *B* и *C*, а с выхода сумматора число передается в регистр *B,* где замещает одно из слагаемых; в регистрах *B* и *C* имеются цепи для сдвига влево на 1 разряд, а в регистре *C* — также цепи для обращения кодов. Почти все оборудование, необходимое для выполнения деления, используется и при выполнении умножения: регистр *A* является регистром множителя, регистр *C* — регистром множимого, регистр *B* — регистром частичных произведений; используется и сумматор. Регистры *B* и *C,* сумматор и цепи обращения кода в *C* нужны также для сложения и вычитания. Одни лишь цепи сдвига влево годятся, может быть, только для выполнения деления (последнее зависит от того, в каком варианте выполняется умножение — ср. с множительными устройствами, рассматриваемыми в 4.1.2).

Когда производится деление, схема рис. 5-1,*а* должна работать следующим образом. До начала операции делимое (обозначим его через *B)* принимается в регистр *B,* делитель (который мы обозначаем буквой *C*) —в регистр *C*; в регистре *C* выполняется обращение кода делителя. Последующие действия удобно выполнять в том случае, если устройство сумматора позволяет отделить процесс формирования сигналов двоичного переноса от выдачи суммы в регистр *B* (такие сумматоры рассматривались в разделе 2). Если это так, то деление начинается с формирования сигналов двоичного переноса, причем на вход переноса младшего разряда сумматора, подается единица. Благодаря наличию этой единицы сигналы переноса образуются так, как будто делимое *B* складывается не с обратным, а с дополнительным кодом делителя *C*, т. е. с величиной 1 —*C*. Далее возможны два случая:

если на выходе старшего разряда сумматора сигнал переноса отсутствует (в разряд целых переносится нуль), то это означает, что $B + (1 — C) < 1$, т. е. $B < C$;

если на выходе старшего разряда сумматора появляется единица переноса, то это значит, что $B + (1—C) \geq 1$ и, следовательно, $B \geq C$ (подобный метод сравнения чисел рассматривался уже в 3.3). В зависимости от исхода сравнения образуется старшая цифра частного (0 или 1), которая заносится в м л а д ш и й разряд регистра *A,* причем содержи» мое регистра *A* сдвигается на 1 разряд влево. Также в зависимости от исхода сравнения либо прерывается, либо доводится до конца суммирование $B + (1 — C)$, так что в регистре *B* соответственно либо остается прежнее число — (делимое *B),* либо получается разность $B — C$; в результате в *B* образуется первый остаток B_1. Затем содержимое регистра *B* сдвигается влево на 1 разряд (удвоение первого остатка), выполняется формирование сигналов двоичного переноса, что позволяет сравнить $2B_1$ с *C* и определить вторую цифру частного и т. д. Цифры частного по одной вдвигаются в регистр *A* со стороны младших разрядов.

Мы говорили уже о том, что первый остаток всегда меньше делителя ($B_1 < C$). После удвоения первого остатка получим $2B_1 < 2C$; если при этом $2B_1 \geq C,$ то вторая цифра частного есть единица, а второй остаток равен $B_2 = 2B_1 — — C < C$; точно так же и все последующие остатки (B_i) меньше *C.* Так как $C < 1$, то и любой из очередных остатков меньше единицы и, следовательно, может разместиться в основных разрядах регистра *B* (хотя в *B* каждый раз и производится сдвиг влево).

Правда, непосредственно после сдвига очередного остатка (B_i) влево может оказаться, что $2B_i \geq 1$. Поэтому в регистре *B* нужно все же иметь один лишний разряд слева от запятой. Если в этот разряд выдвинута единица, то это значит, что $2B_i >$C (потому что $C < 1$). Очередная цифра частного равна единице либо в том случае, когда при формировании сигналов переноса получилась единица на выходе переноса старшего разряда сумматора, либо в том случае, когда после сдвига предыдущего остатка влево выдвинулась единица в дополнительный разряд регистра *B.* Производить суммирование в дополнительном разряде *B* не имеет смысла, так как заранее известно, что следующий остаток будет меньше единицы, а цифра, имеющаяся в дополнительном разряде, при последующем сдвиге влево все равно уходит за пределы регистра *B*.

2°. Возможен и *второй вариант* осуществления описанного алгоритма деления. Этот вариант иллюстрируется, рис. 5-1,б. Отличается он тем, что вместо сдвига *B* влево в каждом цикле деления выполняется сдвиг *C* вправо на 1 разряд; при этом в старшие разряды *C* должны, конечно, вдвигаться единицы, так как в *C* находится обратный код делителя. Преимуществом этого варианта является возможность совмещения по времени тактов суммирования и сдвига (поскольку выдача суммы производится в регистр *B,* а сдвиг должен выполняться в регистре С). Недостаток его — необходимость иметь удвоенное количество разрядов в регистрах *B, C* и сумматоре.

Впрочем, в некоторых случаях удвоения длины регистров и сумматора можно избежать. Соответствующая модификация схемы рис. 5-1,б показана на рис. 5-2[*]). Регистры *B, C* и сумматор содержат здесь всего по $m + 1$ разрядов. Сумматор выполнен с цепью циклического переноса, так что сравнение и вычитание должны производиться не через дополнительные, а через обратные коды. На результате деления это может сказаться только в одном месте: если на некотором шаге очередной остаток оказался в точности равным делителю, взятому с соответствующим весом, то при работе с дополнительными кодами очередная цифра частного получилась бы равной единице, а все последующие цифры были бы нулями, при работе же с обратными кодами очередная цифра частного в этом случае получается нулем (потому что перенос из старшего разряда сумматора отсутствует), а все последующие цифры — единицами; возможная погрешность при этом равна единице младшего разряда. Основная идея схемы рис. 5-2 базируется на том, что очередной остаток в любом цикле деления получается меньше, чем

[*]) Идея этой схемы принадлежит Н. Я. Матюхину, с разрешения которого она была опубликована автором в книге «Арифметические устройства электронных цифровых машин», Физматгиз, 1958, стр. 147—148.

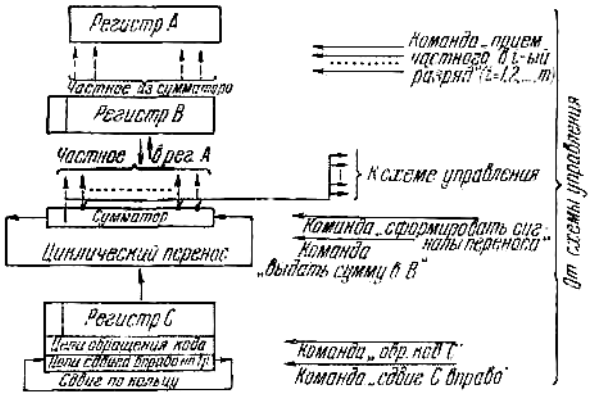


Рис. 5-2. Модификация схемы рис.5-1, *б.*

сдвинутый делитель, или, может быть, равным ему (поскольку мы работаем с обратными кодами) и поэтому содержит впереди не меньшее количество нулей, чем сдвинутый делитель; после того как делитель будет сдвинут вправо еще на один разряд, предыдущий остаток может оказаться больше делителя, но при этом он остается меньше, чем удвоенный делитель, и содержит, может быть, только на один нуль впереди меньше, чем изображение делителя (если бы делитель был в прямом коде). В связи с этим при сдвиге делителя вправо его младшие разряды можно по кольцу вдвигать в освобождающиеся старшие разряды регистра С; младшие разряды очередного остатка будут при этом автоматически занимать свободные старшие разряды регистра *В.* Что же касается самого процесса вычитания, то, он будет идти правильно, несмотря на то, что в регистрах *В* и *С* младшие разряды чисел поставлены впереди старших: цепь переносов все равно замкнута в кольцо (циклический перенос). Единственное неудобство состоит в том, что в прежней схеме об исходе сравнения мы могли судить всегда по сигналу из одного и того же места (с выхода переноса старшего разряда сумматора), теперь же в каждом цикле старший разряд передвигается по сумматору на одну позицию вправо. В схеме управления поэтому необходимо предусмотреть специальное устройство, которое отсчитывало бы циклы деления и в зависимости от номера цикла выбирало бы сигнал с нужного разряда цепи переносов. Зато можно отказаться от цепей сдвига в регистре *А* и цифры частного передавать каждый раз прямо в соответствующий разряд *А*.

Заметим, что, хотя умножение и деление являются взаимно обратными операциями, при реализации деления возможны только два варианта (а не четыре, как для умножения). Дело в том, что получение частного можно начинать только от старших его разрядов, в то время как умножение может начинаться и от старших, и от младших разрядов множителя. Поэтому первый и второй варианты выполнения умножения (см. 4.1.2) не имеют аналогов среди вариантов выполнения деления.

3°. Описанный алгоритм выполнения деления, независимо от того в каком варианте он реализуется, удобен только для параллельных устройств и только в том случае, когда построение сумматора позволяет отделить процесс формирования сигналов переноса от выдачи суммы в регистр *В.*

Если такой возможности нет, то для сравнения очередного остатка с делителем нужно было бы выполнить до конца вычитания; затем, если бы оказалось, что разность меньше нуля (т. е. остаток меньше делителя и, значит, вычитания производить не следовало), пришлось бы перед выполнением сдвига восстановить остаток, снова добавив делитель.

Потери времени на восстановление остатков можно избежать, применяя *второй способ* выполнения деления, который называется *делением без восстановления.* Реализация этого способа также возможна в двух вариантах, аналогично тому как реализуется первый способ (см. рис. 5-1,*а* и 5-1,б), но модернизацию второго варианта, показанную на схеме рис. 5-2, осуществить затруднительно. Правило выполнения деления сформулируем применительно к первому варианту его реализации:

(а) вычесть из делимого делитель; если результат отрицателен, то старшая цифра частного, относящаяся к разряду целых, есть нуль, если результат положителен или равен нулю, то старшая цифра частного есть единица; полученная разность в любом случае представляет собой первый остаток;

(б) удвоить предыдущий остаток;

(в) если предыдущий остаток отрицателен, то добавить делитель, если положителен или равен нулю — вычесть делитель; при этом образуется очередной остаток, знак которого определяет очередную цифру частного по правилам, аналогичным (а);

(г) повторять (б) и (в) до тех пор, пока не будут получены все цифры частного.

В качестве обоснования этого метода заметим, что если, например, в первом шаге разность $B - C$ (где B — делимое, C —делитель) оказалась отрицательной, то это значит, что $B < C$ и, следовательно, первая цифра частного есть нуль; после удвоения этой разности получим $1B - 2C$, а добавление делителя дает $2B - 2C + C = 2B - C$; если и эта величина отрицательна, то $2B < C$, и вторая цифра частного тоже нуль; после очередного удвоения получим величину $4B - 2C$, а последующее добавление C даст $4B - 2C + + C = 4B - C$ и т. д. Если,

скажем, $4B - C \geq 0$, то третья цифра частного есть единица, а полученный остаток в точности совпадает с тем, который получился бы при выполнении деления первым способом; вообще каждый раз, когда остаток неотрицателен, он получается в точности таким же, как соответствующий остаток при выполнении деления первым способом.

Пример (ср. с примером на стр. 206).

15/32 : 3/4 = 5/8,

15/32 = .01111,

3/4 = .11000,

5/8 = .10100.

Вычитание делителя	— .01111 .11000	11000 0.10100...
1-й остаток	(— .01001)	
удвоение 1-го остатка добавление делителя	+ (— .10010) .11000	
2-й остаток	(+ .00110)	
удвоение 2-го остатка вычитание делителя	— (+ .01100) .11000	
3-й остаток	(— .01100)	
удвоение 3-го остатка добавление делителя	+ (— .11000) .11000	
4-й остаток	(+ .00000)	
удвоение 4-го остатка вычитание делителя	— (+ .00000) .11000	
5-й остаток	(— .11000)	
удвоение 5-го остатка добавление делителя	+ (—1.10000) .11000	
6-й остаток	(— .11000)	
. . . . .		

Как мы уже говорили, второй способ выполнения деления более универсален, чем первый: его можно применять и в последовательных устройствах, и в параллельных, при любом построении сумматора. Однако там, где это возможно, большей частью выгоднее применить первый способ, так как он позволит получить некоторый выигрыш по времени (за счет того, что такт сравнения короче, чем полный такт суммирования или вычитания).

Методы ускорения деления разработаны вообще слабее, чем методы ускорения умножения. Однако если для ускорения умножения применяется какой-либо очень эффективный метод, то приходится заботиться и об ускорении деления; иначе оказалось бы, что деление выполняется во много раз медленнее, чем умножение, и наличие деления в качестве самостоятельной операции потеряло бы смысл: с той же скоростью его можно было бы выполнять и по подпрограмме.

5.1.2. Логический метод ускорения двоичного деления.

Один из немногих известных логических методов ускорения деления [*]) основывается на следующих соображениях.

Если очередной остаток до сдвига влево (или до сдвига делителя вправо) очень мал по сравнению **с** делителем, так что в изображении остатка прямым кодом количество нулей перед первой значащей цифрой на k превышает количество нулей перед первой значащей цифрой в изображении делителя, то по меньшей мере k очередных цифр частного будут нулями. Следовательно, ближайшие k сдвигов остатка влево или делителя вправо можно производить без пропуска тактов для сравнения или сложения-вычитания.

Точно так же в случае, когда очередной, i-й остаток B_i очень велик (близок к числу C_i находящемуся в регистре делителя), так что разность $B_i - C_i$ содержит перед первой значащей цифрой на k нулей больше, чем число C_i, по крайней мере k очередных цифр частного будут единицами. При этом $(i + k)$-й остаток можно получить, сдвинув разность на k разрядов влево (или делитель на k разрядов вправо) и добавив делитель к разности. Действительно, если бы мы, например, в каждом цикле деления сдвигали остаток влево, то $(i + k)$-й остаток (после получения k единиц в частном) был бы равен

$$2(...(2(2B_i - C) - C). . .) - C = 2^k B_i - 2^{k-1}C - 2^{k-2} C - . . . - 2C - C.$$

Сдвинув на k разрядов влево разность $B_i - C$ и добавив к ней делитель C, получим ту же величину:

$$2^k(B_i - C) + C = 2^k B_i - (2^k - 1)C = 2^k B_i - 2^{k-1}C - 2^{k-2}C - . . . - 2C - C.$$

[*]) См. Клямко Э. И., Монахов Г. Д., Метод ускоренного двоичного деления в цифровых вычислительных машинах, «Приборостроение», 1957, № 2.

Следовательно, и в этом случае k очередных сдвигов остатка влево или делителя вправо можно выполнять без пропуска тактов для сравнений или сложений-вычитаний.

Ясно, что метод, основанный на этих соображениях, легко реализуется лишь при условии, что делитель нормализован (т. е. содержит единицу в старшем разряде), а деление выполняется по первому варианту (т. е. со сдвигами остатков влево, а не делителя вправо). Можно подсчитать, что применение такого метода сократило бы количество тактов сравнения или сложения-вычитания примерно на 40%.

5.1.3. Использование избыточных цифр частного (применительно к двоичной системе).

Ряд аппаратных методов ускорения деления базируется на идее использования избыточных цифр частного*). Прежде чем перейти к ее изложению, выскажем некоторые общие соображения.

Для простоты предположим, что речь идет о делении нормализованных чисел и что в процессе деления производятся сдвиги делителя вправо, а не сдвиги остатков влево. Ограничения эти вообще несущественны. Обозначим делимое и делитель соответственно через B и С, остатки — через Bi (i = 0, 1, 2, 3, . . . ; B_0 — B), а величины, получающиеся при последовательных сдвигах делителя вправо — через C_i (при этом $C_0 = C$, $C_1 = 2^{-1}$ C,..., $C_i = 2^{-i}$ C,...).

В самом общем виде процесс двоичного деления нормализованных чисел можно описать следующим образом. Если на некотором шаге предыдущий остаток есть B_i, а величина, полученная в результате сдвигов делителя вправо, равна C_i и если будет найдено, что очередная цифра частного равна a_i, то следующий остаток находится по соотношению

$$B_{i+1} = B_i - a_i C_i.$$

Определив каким-либо способом цифры частного $a_0, a_1, \ldots ...,a_m$, мы в конечном итоге получим

$$\begin{aligned} B_{m+1} &= B_m - a_m C_m = \\ &= B_{m-1} - a_{m-1}C_{m-1} - a_m C_m = \\ &\quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &= B - a_0 C_0 - a_1 C_1 - \ldots - a_m C_m. \end{aligned}$$

Учитывая, что $C_i = 2^{-i}$C, перепишем это равенство в виде

$$B = C(a_0 - 2^{-1}a_1 - 2^{-2}a_2 - \ldots - 2^{-m}a_m) + B_{m+1}.$$

Отсюда видно, что единственное условие, которое следует соблюсти при выборе очередных цифр частного, состоит в том, чтобы остатки B_i от шага к шагу (с ростом номера i) убывали: если последний остаток B_m окажется достаточно малой величиной, то двоичное число Л, записанное цифрами $a_0, a_1, ..., a_m$ ($A - a_0 + 2^{-1}a_1 + \ldots + 2^{-m}a_m$), будет с достаточной степенью точности представлять частное от деления $B : C$ (так как $A = \frac{B}{C} - \frac{B_{m+1}}{C}$). Для того чтобы погрешность частного не превышала единицы младшего разряда, необходимо, чтобы выполнялось условие

$$\frac{|B_{m+1}|}{C} \le 2^{-m} \quad \text{или} \quad |B_{m+1}| \le 2^{-m} C = C_m.$$

Рассмотрим с этой точки зрения первый из методов двоичного деления, изложенный в 5.1.1. Возможными цифрами частного на каждом шаге являются 0 и 1. Если бы на некотором шаге оказалось, что $B_{i-1} > 2C_{i\_1}$, то, выбрав максимально возможную цифру частного — единицу (с тем, чтобы следующий остаток был возможно меньше), мы все равно получили бы

$$B_i = B_{i\_1} - 1 \cdot C_{i-1} > C_{i\_1} = 2C_i.$$

Такое же положение сохранялось бы и для всех последующих шагов, так что в конечном итоге оказалось бы, что

$$B_{m+1} > C_m.$$

Следовательно, получить частное с заданной точностью не удалось бы. Точно так же деление не дало бы правильного результата, если бы на каком-либо шаге оказалось, что $B_{i-1} < 0$: если бы мы дальше выбрали в качестве цифры частного $a_{i-1} = 1$, то следующий остаток снова получился бы отрицательным и возрос бы по абсолютной величине, если бы в качестве всех дальнейших цифр частного выбирались нули, то величина остатка не убывала бы. Таким образом, выбирая каждый раз очередную цифру частного a_i мы должны заботиться о том, чтобы следующий остаток $B_{i+1} = B_i - a_i C_i$ был заключен в пределах

$$0 \le B_{i+1} \le 2C_{i+1} = C_i;$$

---

*) Впервые, вероятно, эта идея высказана М. Надлером — см. Acta Techn., 1956, 1, № 6, стр. 464—478 (Чехословакия). Она разрабатывалась также в нескольких работах автора настоящей книги (напр., «Об одном методе двоичного деления», сб. «Труды конференции «Новые разработки по вычислительной математике и вычислительной технике», Киев, 1960). Сходные идеи, но относящиеся к системам счисления с более высокими основаниями, излагались также в докладе проф. Робертсона Дж. И. (США), сделанном в Москве в 1958 г.

тогда в конце окажется, как это и требуется, что $|B_{m+1}| \le C_m$*).

Поскольку исходные числа нормализованы, т. е.

$$^1/_2 \le B < 1 \quad \text{и} \quad ^1/_2 \le C < 1.$$

то на первом шаге ($B_0 = B$ и $C_0 = C$) указанное условие выполняется. Чтобы оно выполнялось и дальше, цифры частного на каждом шаге должны выбираться по правилу:

$$\text{если } B_i < C_i, \text{ то } a_i = 0$$
$$\text{если } B_i > C_i, \text{ то } a_i = 1.$$

Если $B_i = C_i$ то очередную цифру частного можно принять равной либо 0, либо 1; в первом случае следующий остаток равен $B_{i+1} = B_i = C_i = 2C_{i+1}$, и все последующие цифры частного окажутся единицами; во втором случае следующий остаток B_{i+1} получится равным нулю, и все последующие цифры частного будут нулями. (Возможность различно выбирать очередную цифру частного в случае $B_i = C_i$ отмечалась в 5.1.1 в связи с вопросом о том, как производить сравнение делителя с делимым — через дополнительные или через обратные коды.)

Приведенное здесь объяснение можно считать общим обоснованием первого метода деления, изложенного в 5.1.1.

Теперь рассмотрим с этой точки зрения приведенный в 5.1.1 второй метод выполнения деления.

Возможными цифрами частного в нем являются фактически не 0 и 1, а —1 и +1. Этому соответствует то обстоятельство, что на каждом шаге очередной остаток вычисляется либо добавлением, либо вычитанием величины C_i из предыдущего остатка ($B_{i+1} = B_i - a_i C_i$, где $a_i = -1$ или $a_i = +1$). Когда в 5.1.1 мы предполагали записывать в регистр частного цифры 0 или 1, то при этом по ходу дела в сущности производилось преобразование частного из формы записи цифрами —1, +1 в форму записи цифрами 0, 1.

Приведенное правило деления включало это преобразование. Когда, например, в соответствии с указанным правилом на первом шаге, производится вычитание делителя из делимого, то тем самым уже предопределено, что в форме записи цифрами —1, 1 старшая цифра частного есть 1. Однако эту цифру мы не спешили вносить в регистр частного; вместо этого мы дожидались получения следующей цифры. Если первый остаток получался положительным, то при этом на следующем шаге вновь выполняется вычитание, т. е. следующий разряд частного в форме с цифрами — 1, 1 вновь содержит 1; тогда и в форме записи цифрами 0, 1 первый разряд содержит единицу, и эту единицу уже можно заносить в регистр частного. Если же первый остаток отрицателен, то в форме записи цифрами —1, 1 следующий разряд частного содержит —1 (так как на втором шаге производится добавление сдвинутого делителя), а при преобразовании в форму записи цифрами 0,1 в первом разряде окажется 0 (потому что $(+1)\cdot 1 + (-1)\cdot\frac{1}{2} + \ldots = (0)\cdot 1 + (1)\cdot\frac{1}{2} + \ldots$, или, иначе, $1.\overline{1}\ldots = 0.1\ldots$). Точно так же вторую цифру частного мы не заносили в регистр результата до получения знака второго остатка, определяющего следующую цифру частного, и т. д.

Если принять таким образом, что цифрами частного при использовании второго метода деления могут быть — 1 и + 1, то обоснование этого метода из общих соображений аналогично обоснованию первого метода. Нетрудно видеть, что, когда возможными цифрами частного являются —1 и + 1, для получения правильного результата деления необходимо и достаточно, чтобы на каждом шаге выполнялось условие

$$-2C_i \le B_i \le +2C_i$$

(оно аналогично условию $0 \le B_i \le 2C_i$ для первого метода). На первом шаге ($B_0 = B$, $C_0 = C$, где B и C — делимое и делитель) это условие выполняется. Чтобы оно выполнялось и дальше, цифры частного на каждом шаге должны выбираться по правилу:

$$\text{если } B_i < 0, \text{ то } a_i = -1;$$
$$\text{если } B_i > 0, \text{ то } a_i = +1.$$

В случае $B_i = 0$ очередную цифру частного можно выбрать произвольно; приняв, в частности, $a_i = +1$, получим

$$B_{i+1} = B_i - a_i C_i = -C_i,$$

в результате чего все последующие цифры будут —1; приняв $a_i = -1$, получим

$$B_{i+1} = B_i - a_i C_i = +C_i,$$

так что все последующие цифры частного будут равны +1.

Перейдем теперь непосредственно к изложению идеи использования избыточных цифр частного. Идея эта состоит в том, что для каждого разряда частного допустимыми считаются не две возможные цифры (скажем, 0 и 1, или —1 и +1), а три или больше различных цифр. При этом мы рассчитываем на то, что за счет избыточных возможностей мы получим несколько большую свободу в выборе очередных цифр частного. При наличии всего двух допустимых цифр можно было выбирать любую из них лишь при одном определенном значении остатка ($B_i = C_i$ для первого метода или $B_i = 0$ для второго метода) и не более чем один раз за весь процесс деления; при наличии большего количества допустимых цифр такая возможность встретится чаще.

При рассмотрении логических методов ускорения умножения мы уже встречались с формой записи двоичного

*) Более строгое обоснование этого положения имеется на стр. 211—212.

числа (множителя), когда для каждого разряда допускались не две, а три различные цифры:—1, 0, +1 (см. 4.2.2). Преобразование чисел в указанную форму оказывалось неоднозначным; например, число $^3/_4$ можно представить либо в виде $0\cdot 1+1\cdot\frac{1}{2}+1\cdot\frac{1}{4}$ (т. е. 0.11), либо в виде $1\cdot 1+0\cdot\frac{1}{2}+(-1)\cdot\frac{1}{4}$ (т.е. $1.0\,\bar{1}$), либо в виде $1\cdot 1+(-1)\cdot\frac{1}{2}+1\cdot\frac{1}{4}$ (т.е. $1.\bar{1}\,1$) .В сущности эта неоднозначность записи и позволит получить некоторую свободу в выборе цифр частного.

Итак, пусть для любого разряда допустимыми цифрами частного являются $\alpha_1, \alpha_2, \ldots, \alpha_k$, причем α_1 — наименьшая из них, а α_k — наибольшая (в алгебраическом смысле).

Рассмотрим некоторый *j*-й шаг. Для того чтобы во всех последующих шагах ($i > j$) остатки B_i убывали с той же скоростью, с какой убывают величины C_i необходимо и достаточно, чтобы на *j*-м шаге выполнялись условия

$$2\alpha_1 C_j \le B_j \le 2\alpha_k C_j.$$

Достаточность этих условий вытекает из того, что их выполнение на *j*-м шаге позволяет выбрать очередную цифру частного a_j так, чтобы аналогичные условия выполнялись и для (j + 1)-го шага. Например, если B_j минимально, $B_j = 2\alpha_1 C_j$, то, выбрав минимально возможную цифру частного $a_j = \alpha_1$ (с тем, чтобы следующий остаток уменьшился возможно слабее), получили бы

$$B_{j+1} = B_j - a_j C_j = 2\alpha_1 C_j - \alpha_1 C_j = \alpha_1 C_j = 2\alpha_1 C_{j+1}.$$

Точно так же если B_j максимально, $B_j = 2\alpha_k C_j$, то для a_j нужно выбрать максимально возможное значение $a_j = a_k$, в результате чего получится $B_{j+1} = 2\alpha_k C_{j+1}$.

Для доказательства необходимости указанных условий предположим, что на *j*-м шаге они не выполняются; пусть, скажем, $B_j = (2\alpha_k + \varepsilon_j)Cj$, где $\varepsilon_j > 0$. Выбрав для этого случая максимальную цифру частного $a_j = \alpha_k$ (с тем, чтобы возможно сильнее уменьшить следующий остаток), мы тем не менее получили бы

$$B_{j+1} = B_j - a_j C_j = (2\alpha_k + \varepsilon_j)\, C_j - \alpha_k C_j = (\alpha_k + \varepsilon_j) C_j = (2\alpha_k + 2\varepsilon_j) C_{j+1},$$

так что $\varepsilon_{j+1} = 2\varepsilon_j$ и вообще для всех последующих шагов $\varepsilon_{i+1} = 2\varepsilon_i$ ($i \ge j$).

На первом шаге деления нормализованных чисел, когда $B_0 = B$, $C_0 = C$, имеет место соотношение $0 \le B_0 \le 2C_0$. Чтобы оно было эквивалентно указанному условию, необходимо, чтобы наименьшая из возможных цифр частного (α_1) была не больше 0, а наибольшая из возможных цифр частного (α_k) —не меньше 1. Это — единственное ограничение, накладываемое на набор возможных цифр частного $\alpha_1, \alpha_2, \ldots, \alpha_k$. (Отсюда, в частности, видно, что количество возможных цифр должно быть не меньше двух; например, годятся наборы цифр $\alpha_1 = 0$, $\alpha_2 = 1$ — как в первом из описанных ранее методов деления, или $\alpha_1 = -1$, $\alpha_2 = 1$ — как во втором методе.) Далее каждую очередную цифру частного a_i следует выбирать из имеющегося набора допустимых цифр $\alpha_1, \alpha_2, \ldots, \alpha_k$ по правилу:

если $2\alpha_1 C_i \le B_i \le (\alpha_k + \alpha_1)\, C_i$, то $a_i = \alpha_1$,
если $(\alpha_1 + \alpha_2)\, C_i \le B_i \le (\alpha_k + \alpha_2)\, C_i$, то $a_i = \alpha_2$,
если $(\alpha_1 + \alpha_3)\, C_i \le B_i \le (\alpha_k + \alpha_3)\, C_i$, то $a_i = \alpha_3$,
. . . . . .
если $(\alpha_1 + \alpha_k)\, C_i \le B_i \le 2\alpha_k C_i$, то $a_i = \alpha_k$.

Так как для следующего шага $B_{i+1} = B_i - a_i C_i$, а $C_{i+1} = \frac{1}{2} C_i$, то соблюдение этого правила на *i*-м шаге обеспечит для следующего (i + 1)-го шага выполнение аналогичного условия ($2\alpha_1 C_{i+1} \le B_{i+1} \le 2\alpha_k C_{i+1}$), как это и требуется.

В рассматривавшихся ранее случаях набор возможных цифр частного состоял всего из двух цифр α_1 и α_2; при этом правило выбора очередной цифры частного формулировалось следующим образом:

если $2\alpha_1 C_i \le B_i \le (\alpha_1 + \alpha_2)\, C_i$, то $\alpha_i = \alpha_1$;
если $(\alpha_1 + \alpha_2) C_i \le B_i \le 2\alpha_2 C_i$, то $a_i = \alpha_2$;

свобода в выборе очередной цифры частного существовала только в одном частном случае, когда $B_i = (\alpha_1 + \alpha_2)\, C_i$. Например, если $\alpha_1 = 0$, $\alpha_2 = 1$, то при $0 \le B_i \le C_i$ выбирается $a_i = 0$, при $C_i \le Bi \le 2C_i$ —$a_i = 1$; если $\alpha_1 = = -1, \alpha_2 = 1$, то при $-2C_i \le B_i \le 0$ выбирается $a_i = -1$, а при $0 \le B_i \le 2C_i$ — $a_i = 1$). Допустив существование большего количества возможных цифр частного, получим большую свободу в выборе очередной цифры; например, имея набор из трех возможных цифр, $\alpha_1 = -1$, $\alpha_2 = 0$, $\alpha_3 = 1$, получим следующее правило:

если $-2C_i \le B_i \le 0$, то $a_i = -1$,
если $-Ci \le B_i \le C_i$, то $a_i = 0$,
если $0 \le B \le 2C_i$, то $a_i = 1$;

в интервалах значений B_i от $-2C_i$; до $-C_i$, и от + C_i до + $2C_i$ очередная цифра частного выбирается всегда однозначно; в интервале от — C_i до 0 в качестве a_i; можно выбрать и —1, и 0, а в интервале от 0 до + C_i — либо 0, либо + 1. Чем больше разных возможностей допускается для цифр частного, тем шире интервалы значений B_i, в которых возможен неоднозначный выбор значения a_i.

5.1.4. Аппаратный метод 1-го порядка для ускорения двоичного деления.

Применим идею использования избыточных цифр частного к построению делительного устройства, в котором все суммирования и вычитания, необходимые в ходе деления, выполнялись бы без сквозных переносов. Метод *ускорения деления путем запоминания переносов* аналогичен методам ускорения умножения, описанным в 4.3.2.

Как и умножение, деление представляет собой цепь последовательных суммирований (вычитаний) некоторого числа с содержимым регистра-накопителя. Ясно, что каждое такое суммирование (вычитание) можно выполнять без сквозных переносов, запоминая местные переносы с тем, чтобы учесть их при последующем суммировании. Однако при этом результат (в данном случае очередной остаток) получается каждый раз в виде пары чисел: числа, составленного из цифр суммы, и числа, составленного из цифр переноса. Определить очередную цифру частного, имея остаток, который записан в такой форме, без использования избыточных цифр частного было бы затруднительно.

Возможное построение устройства иллюстрируется рис. 5-3. В общих чертах оно аналогично схеме рис. 5-1,*а*, приведенной на стр. 206. Главное отличие состоит в том, что основной сумматор выполнен без сквозных переносов; вместо этого в каждом разряде имеется специальный триггер для запоминания сигнала переноса, поступающего из предыдущего разряда. Все вместе эти триггеры образуют четвертый, вспомогательный, регистр устройства, который на рисунке назван регистром *E.* В регистре *E*, как и в регистре *B*, имеются цепи для сдвига влево на 1 разряд; когда по ходу выполнения деления очередной остаток должен удваиваться, то для этого, очевидно, нужно сдвинуть влево одновременно и содержимое регистра *B*, и содержимое регистра *E.*

В старших разрядах регистров *B* и *E*, однако, вместо цепей сдвига влево имеется вспомогательный сумматор. Одновременно с тем, как в младших разрядах регистров *B* и *E* выполняется сдвиг влево, в старших разрядах суммируются цифры *B* и *E* и результат суммирования передается в старшие разряды регистра *B* со сдвигом влево на 1 разряд. Вспомогательный регистр охватывает 4 разряда регистров *B* и *E:* два разряда слева от запятой и два разряда справа от запятой, т. е. разряды с весами 2, 1, $^1/_2$ и $^1/_4$. Результат суммирования, который выдается со сдвигом на 1 разряд влево, попадает соответственно в разряды 3-й (знаковый), 2-й и 1-й слева от запятой и 1-й справа от

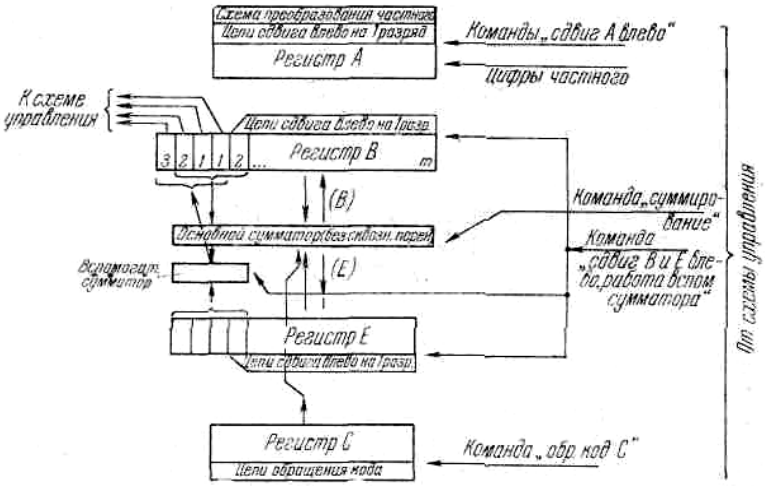


Рис. 5-3. Устройство для выполнения деления без сквозных переносов при суммированиях-вычитаниях.

запятой, т. е. в разряды с весами 4, 2, 1 и $^1/_2$ (о том, почему регистр *B* должен иметь три разряда слева от запятой, сказано ниже).

Пусть, например, в результате суммирования очередной остаток получился в виде

* 10.0110011110... (цифры поразрядной суммы в регистре *B*)
* 00.1111000111 ... (цифры поразрядных переносов в регистре *E*)

Звездочками обозначены цифры 3-го разряда слева от запятой, которые до сдвига влево не играют роли. Так как до сдвига влево знаковым является второй разряд слева от запятой, то полученный остаток в действительности отрицателен и по абсолютной величине меньше единицы. Сдвиг влево в младших разрядах и суммирование со сдвигом в старших разрядах дадут:

Суммирование со сдвигом влево	Сдвиг влево	
*10.01	10011110...	
*00.11	11000111...	
110.0	10011110...	(регистр *B*)
	11000111...	(регистр *E*)

В результате получим удвоенный очередной остаток; он записан, однако, в несколько иной форме, чем прежде: в

старших разрядах значащие цифры имеются только в регистре *B,* а цифры *E* являются нулями. Поскольку во вспомогательном сумматоре сигналы переноса распространяются всего через 3 разряда, суммирование в нем выполняется достаточно быстро и укладывается примерно в то же время, что и выполнение сдвига в младших разрядах.

Рассмотрим теперь непосредственно работу устройства при выполнении деления.

Будем полагать, что для любого разряда частного допустимы цифры —1, 0, 1, а делитель нормализован, т. е. $^1/_2 \le C < 1$. В соответствии с общим правилом цифры частного будут выбираться с таким расчетом, чтобы очередной удвоенный остаток B_{i+1} был заключен в пределах

$$-2C \le B_{i+1} \le 2C.$$

Добавление и вычитание делителя (в случаях, когда очередная цифра частного есть —1 или соответственно + 1) будут выполняться путем добавления прямого или дополнительного кода делителя; поэтому и остаток, если он отрицателен, будет получаться в дополнительном коде. Казалось бы, поскольку удвоенный остаток по абсолютной величине меньше чем 2, знаковым разрядом мог бы служить второй разряд слева от запятой в регистре *B*. Однако удвоенный остаток состоит не только из числа в регистре *B,* но и из положительной добавки, находящейся в регистре *E*; по абсолютной величине эта добавка меньше $^1/_2$ (ее старшая значащая цифра находится в разряде с весом $^1/_4$); следовательно, величина, находящаяся в регистре *B,* по модулю меньше чем $2^1/_2$ и знаковым разрядом должен быть третий слева от запятой разряд регистра *B*.

При выборе очередной цифры частного мы будем принимать во внимание только 4 старших разряда регистра *B* (3-й, 2-й и 1-й разряды слева от запятой и 1-й разряд справа от запятой). Тем самым очередной остаток уменьшен как бы на величину, представленную младшими разрядами регистра *B,* и на величину, имеющуюся в регистре *E*. Обе эти величины неотрицательны, причем каждая из них меньше чем $^1/_2$. В сумме они дают неотрицательную величину, меньшую чем 1. Если в четырех старших разрядах регистра *B* имеется нуль или некоторая положительная величина, то это значит, что весь остаток неотрицателен. Если в четырех старших разрядах *B* имеется отрицательная величина, неравная $-^1/_2$ (она может быть равна —1,—$1^1/_2$, — 2,...), то и весь остаток наверняка отрицателен. Наконец, если в старших разрядах содержится величина $-^1/_2$ (т. е. 111.1, где первая слева цифра принадлежит разряду знака с весом — 4), то весь остаток может оказаться и положительным, и отрицательным, но по абсолютной величине не превышает $^1/_2$ (т. е. наверняка не превышает делитель *C*).

В соответствии с правилом выбора цифр частного, имеющимся на стр. 212, в первом случае очередная цифра частного будет считаться равной + 1, во втором случае —1, в третьем случае 0. Схема образования цифр частного чрезвычайно проста: если третий слева от запятой разряд *B* содержит 0, то цифра частного есть + 1; если третий слева от запятой разряд *B* содержит 1, а в трех последующих разрядах *B* имеется хотя бы один 0, то цифра частного есть —1; если во всех четырех старших разрядах *B* содержатся единицы, то очередная цифра частного есть 0.

Таким образом, очередную цифру частного можно определить по 4 старшим разрядам *B,* не принимая во внимание возможные переносы из младших разрядов. Сокращение времени деления получается за счет того, что все суммирования и вычитания, необходимые по ходу дела, выполняются без сквозных переносов.

Последняя трудность, которую нужно при этом разрешить, состоит в том, чтобы получить частное в обычной форме записи, использующей цифры 0, 1, но не —1.

Наиболее очевидное решение состоит в том, чтобы вместо одного регистра *A* иметь 2 регистра: *A′* и *A*". В первом из них в ходе деления запоминались бы положительные единицы частного, во втором — отрицательные. Для получения

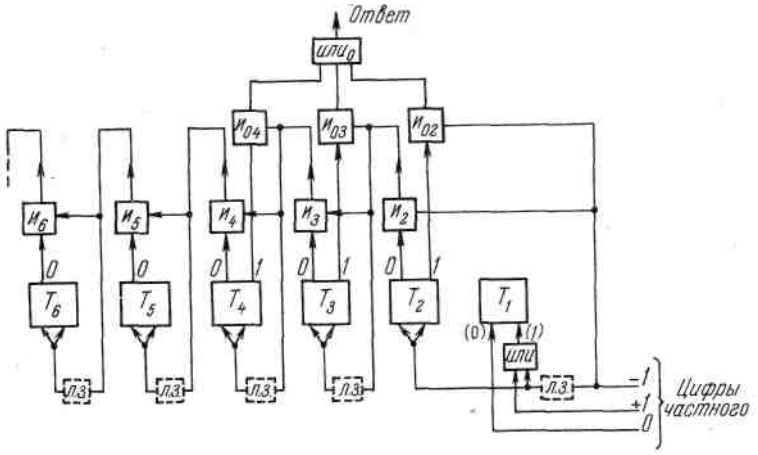


Рис. 5-4. Одно из возможных построений схемы преобразования частного.

частного в обычной форме нужно в конце выполнения деления произвести вычитание содержимого регистра *A″* из содержимого регистра *A′*.

Другое решение могло бы состоять в создании специальной схемы преобразования частного, например, такой, как показано на рис. 5-4. Триггеры, изображенные на этом рисунке, принадлежат регистру *A*. Цепи сдвига влево,

имеющиеся в этом регистре, на рисунке не показаны; сдвиги влево в регистре A и преобразования частного выполняются вообще в разное время: сдвиги — одновременно со сдвигами в регистрах B и E, преобразования частного — одновременно с суммированиями в основном сумматоре. К тому времени, Когда в младший разряд регистра A (триггер T_1) поступает очередная цифра частного, предыдущие цифры уже сдвинуты на 1 разряд влево, так что триггер T_1 свободен. Если очередная цифра частного есть 0 или +1, то триггер T_1 просто устанавливается в соответствующее состояние, и никаких дополнительных операций производить не нужно. Если же очередная цифра есть—1,то триггер T_1 устанавливается в положение +1 и одновременно преобразуются предыдущие цифры. При этом цепь преобразования частного, образованная вентилями «$и_2$», «$и_3$», «$и_4$», ... и линиями задержки, представляет собой попросту схему двоичного займа: если на T_1 поступил сигнал «—1», то одновременно с установкой T_1 в положение «+ 1» изменяется на противоположное состояние T_2; если T_2 до этого хранил 0, то одновременно подается сигнал займа на T_3 и т. д.; распространение сигналов займа обрывается на первом из разрядов, в котором имеется единица. На рис. 5-4 цепь двоичного займа построена по аналогии с цепью двоичного переноса, имеющейся в счетчике рис. 2-37,*а* на стр. 96; ее можно построить и как-нибудь иначе.

На первый взгляд кажется, что необходимость в распространении сигналов двоичного займа вдоль цепочки разрядов регистра A сводит на нет то преимущество, что в основном сумматоре отсутствует распространение сигналов двоичного переноса. Цепь распространения сигналов двоичного переноса в сумматоре можно было бы построить из таких же элементов, из которых строится цепь двоичного займа, и процесс этот происходил бы с той же скоростью.

Заметим, однако, что сигналы двоичного займа в цепи преобразования частного возникают лишь тогда, когда очередная цифра частного есть —1; вероятность этих случаев несколько меньше половины (очередной остаток меньше нуля в половине всех случаев; но если при этом по абсолютной величине он не превышает $^1/_2$, то очередная цифра частного есть 0, а не —1). Кроме того, в отличие от сигналов переноса в сумматоре, которые могут возникнуть в любом разряде, сигналы двоичного займа в цепи преобразования частного зарождаются всегда в самом младшем разряде и распространяются только до первого из разрядов, который до преобразования содержал 1. Так как цифры «0» и «1» во всех разрядах частного равновероятны, вероятность длинной цепочки нулей в младших разрядах невелика. На рис. 5-4 вентили «$и_{02}$», «$и_{03}$» и «$и_{04}$» и элемент «$или_0$» образуют схему формирования ответа. Если пробег сигналов займа оборвется в одном из трех младших разрядов, то эта схема сформирует импульс-ответ, обозначающий, что специального времени для преобразования частного можно не отводить (мы полагаем, что распространение сигналов займа через 2 разряда происходит достаточно быстро и по времени укладывается в такт сдвига). Вероятность же того, что все три младшие разряда до преобразования содержали нули, равна всего $^1/_8$. Таким образом, общая вероятность того, что в схеме рис. 5-4 в некотором шаге деления потребуется специальное время для выполнения преобразования частного, меньше чем $\frac{1}{2}\cdot\frac{1}{8}$, т. е. меньше $^1/_{16}$. Скажем, при выполнении операций с 30 двоичными разрядами, такие случаи встречаются в среднем в одном-двух (из тридцати) циклах деления.

О некоторых дальнейших возможностях развития этого метода ускорения деления говорится в 5.1.8.

5.1.5. Аппаратный метод 2-го порядка для ускорения двоичного деления.

Если для ускорения умножения применяется метод второго порядка, то оборудование множительного устройства желательно использовать и для ускорения деления. При этом весьма полезной оказывается идея использования избыточных цифр частного. Одно из возможных решений иллюстрируется рис. 5-5[*]). Устройство, показанное на этом рисунке, представляет собой несколько усложненную схему множительного устройства рис. 4-11 (стр. 187), описанного в разделе 4.5.2. На рис. 5-5 показан только левый верхний угол ромба.

Принцип выполнения деления в приведенной схеме состоит в следующем.

На входах каждого горизонтального ряда одноразрядных сумматоров имеются логические элементы «и — или», которые позволяют передать на входы сумматоров либо прямой, либо обратный код делителя; при этом данный ряд сумматоров будет выполнять либо добавление, либо вычитание делителя, сдвинутого соответствующим образом.

Работой логических элементов управляют дешифраторы сигналов управления («Д»), имеющиеся в каждом ряду. На входы дешифраторов сигналы подаются с выходов старших разрядов предыдущего ряда сумматоров. В зависимости от комбинации цифр на входах, т. е. в зависимости от старших разрядов предыдущего остатка, в дешифраторе формируется очередная цифра частного: + 1, —1 или 0. Если цифра частного + 1, то дешифратор вырабатывает сигнал «—», и в следующем ряду сумматоров сдвинутый делитель вычитается из предыдущего остатка; если цифра частного есть—1, то дешифратор вырабатывает сигнал «+», и в следующем ряду сумматоров сдвинутый делитель добавляется к предыдущему остатку; если цифра частного есть 0, то дешифратор не вырабатывает никаких сигналов, и следующий ряд сумматоров не производит ни сложения, ни вычитания. В зависимости от цифр, которые получатся на выходе следующего ряда сумматоров, та или иная цифра частного сформируется в следующем дешифраторе сигналов управления и т. д.

[*]) К а р ц е в М. А., Авт. св. СССР, кл. 42т, 14, с приоритетом от 4.09.1959 г., № 129390.

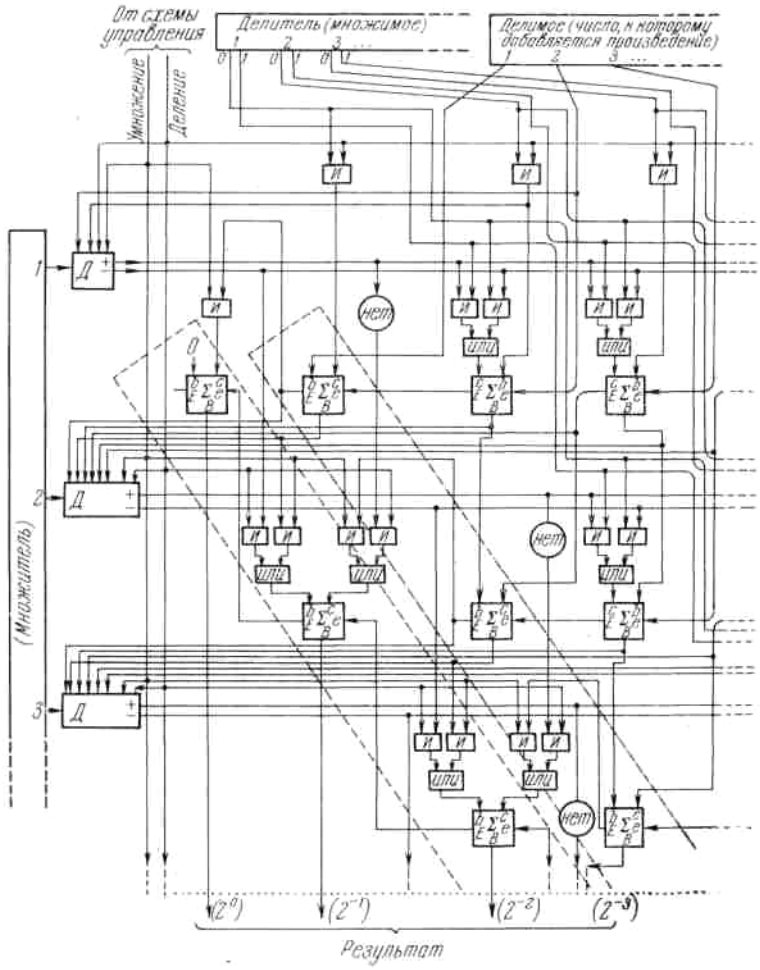


Рис. 5-5. Устройство для выполнения умножения и деления с использованием методов ускорения 2-го порядка.

Существенным здесь является то, что сигналы переноса распространяются не вдоль горизонтальных рядов одноразрядных сумматоров (как было, например, в множительном устройстве рис. 4-10 на стр. 186), а по вертикали (так же, как на рис. 4-11). Благодаря этому все сумматорынекоторого горизонтального ряда срабатывают одновременно. Сразу вслед за этим начинается срабатывание дешифратора сигналов управления следующего ряда. Если бы переносы распространялись по горизонтали, то послеподачи сигнала «+» или «—» на данный ряд сумматоров нужно было бы выждать, пока сигналы переноса пройдут вдоль этого ряда от младших разрядов к старшим, и только затем мог бы начать работу следующий дешифратор сигналов управления; деление при этом выполнялось бы слишком медленно.

Однако в рассматриваемой схеме каждый очередной остаток получается в виде двух чисел: числа, составленного из цифр на выходах *B* предыдущего ряда сумматоров, и числа, составленного из цифр на выходах *E*. Для определения очередной цифры частного приходится пользоваться теми же приемами, которые рассматривались в 5.1.3 и 5.1.4. Во всех рядах, кроме первого, в дешифратор сигналов управления входит по 6 цифр из предыдущего остатка; если старший разряд сдвинутого делителя, подаваемого на следующий ряд сумматоров, имеет вес 2^{-i} (веса разрядов проставлены на рис. 5-5 внизу), то в дешифратор сигналов управления должны входить из предыдущего ряда сумматоров: цифра с весом 2^{-i+3} (с выхода *E* левого, дополнительного, одноразрядного сумматора, который на рис. 5-5 находится в пунктирной рамке), цифра с весом 2^{-i+2} (с выхода *B* того же сумматора), две цифры с весами 2^{-i+1} (с выхода *B* старшего из основных одноразрядных сумматоров и с выхода *E* второго из основных одноразрядных сумматоров) и две цифры с весами 2^{-i} (с выхода *B* второго из основных сумматоров и с выхода *E* третьего). Правила, по которым образуется очередная цифра частного, сходны с описанными ранее (см. 5.1.3 и 5.1.4). Что касается самого первого дешифратора, то, поскольку первый остаток является разностью двух нормализованных чисел (эта разность — результат вычитания делителя из делимого — в схеме рис. 5-5 нигде в явном виде не присутствует), то, как нетрудно проверить, в него достаточно подать только вторую цифру делителя; дешифратор этот значительно проще, чем все остальные.

Второй дополнительный параллельный сумматор, тоже обведенный на рис. 5-5 пунктиром и расположенный левее первого, при выполнении деления служит для получения частного в обычной форме. На одни его входы (*b*) подается число, составленное из цифр частного «+1», на другие входы (*c*) — обратный код числа, составленного из цифр частного «—1» (обратный код образуется инверторами — элементами «нет»); сумматор фактически производит вычитание этих чисел, в результате чего получается частное, записанное цифрами 0, + 1, как это обычно и требуется.

При выполнении умножения этот параллельный сумматор служит для получения произведения в виде одного числа. Вообще при выполнении умножения схема рис. 5-5 работает в общем так же, как схема рис. 4-11. Дешифраторы сигналов управления вырабатывают сигналы «+» или не вырабатывают никаких сигналов в зависимости от соответствующих цифр множителя. Наличие регистра делимого и «лишнего» по сравнению рис. 4-11 ряда одноразрядных сумматоров (верхний горизонтальный ряд) позволяет выполнять не только простое умножение, но и умножение со сложением, когда к числу, помещенному в регистр делимого, добавляется произведение чисел, находящихся в регистре множимого и множителя.

Усложнения, сделанные специально для деления, невелики: логические схемы на входах горизонтальных рядов сумматоров и на входах дополнительного параллельного сумматора, дешифраторы сигналов управления. Зато оборудование, затраченное для ускорения умножения, позволяет почти с такой же скоростью выполнять деление. Разница в скоростях получается только за счет того, что при умножении дешифраторы сигналов управления подготавливаются все одновременно (сразу вслед за приемом в свой регистр множителя), а при делении каждый из дешифраторов сигналов управления срабатывает после того, как сработает предыдущий ряд сумматоров.

Из раздела 5.1.8 видно, что имеется также возможность выполнения деления в устройствах с усложненной логикой (см. 4.5.3), дающих увеличение быстродействия при одновременном сокращении количества оборудования.

5.1.6. Выполнение двоичного деления в упрощенном арифметическом устройстве последовательного типа.

Во всех предыдущих разделах мы неоднократно подчеркивали, что для выполнения деления необходимо максимально использовать оборудование, предназначенное для выполнения других операций, и возможно меньше применять специальное оборудование. Ясно, что если, скажем, в последовательной машине применяется какое-либо очень простое арифметическое устройство для выполнения сложения, вычитания и умножения, то желательно и операцию деления реализовать в той же схеме.

В качестве примера покажем, как можно реализовать деление в упрощенном арифметическом устройстве для последовательной машины с магнитным барабаном, приведенным на рис. 4-18 (стр. 195) и описанным в 4.6.3. Напомним, что все арифметическое устройство это состоит из одного последовательного сумматора и одного регистра с цепями сдвига вправо, а также нескольких вспомогательных вентилей и инверторов. Для выполнения умножения дополнительно занималась одна дорожка магнитного барабана, причем умножение выполнялось в течение 3 оборотов барабана.

При выполнении деления также нужно будет занять одну дорожку магнитного барабана. По времени оно займет столько же, сколько и умножение,— 3 оборота барабана. Выполняется деление в следующем порядке.

В течение первого оборота барабана в регистре находится делитель; он расписывается на рабочей дорожке барабана столько раз, сколько разрядов содержится в исходных числах.

Затем в регистр помещается делимое. В начале второго оборота барабана из делимого вычитается делитель, причем разность возвращается в регистр через элемент задержки на один такт, т. е. сдвинутой влево на 1 разряд. Знак разности записывается на рабочую дорожку барабана в промежутке между первой и второй записью делителя или, скажем, вместо младшей цифры в первой записи делителя. В зависимости от того, положительной или отрицательной получилась первая разность, далее производится соответственно вычитание или прибавление делителя; результат снова через однотактный элемент задержки возвращается в регистр, а его знак выводится на рабочую дорожку барабана между второй и третьей записью делителя и т. д.

В течение третьего оборота барабана в регистре собирается частное. Частное фактически уже имеется на рабочей дорожке барабана. Трудность состоит только в том, что его цифры расположены на дорожке, начиная от старшей, а в регистре отсутствуют цепи для сдвига влево. Так как цифры частного записаны на барабане с большими промежутками, то можно поступить следующим образом: каждую очередную цифру частного будем вносить в старший разряд регистра; в промежутке времени до появления следующей цифры в регистре будут выполняться сдвиги вправо, причем цифры, выходящие из младшего разряда регистра, будут по кольцу вдвигаться в его младший разряд, а количество сдвигов каждый раз должно быть на единицу меньше количества разрядов в регистре. Нетрудно видеть, что при этом фактически получается сдвиг влево на один разряд. В итоге к концу третьего оборота в регистре в правильном порядке оказываются все цифры частного.

Таким образом, весьма простое арифметическое устройство, показанное на рис. 4-18, может обеспечить достаточную (для машины с магнитным барабаном) скорость работы при выполнении всех четырех арифметических действий.

5.1.7. Выполнение деления с учетом алгебраических знаков и деления чисел с плавающей запятой (применительно к двоичной системе).

1°. Как и при выполнении умножения, деление с учетом алгебраических знаков не вызывает никаких трудностей в том случае, когда для изображения отрицательных чисел используются прямые коды. Интересно, однако, отметить, что наиболее часто применяемый алгоритм деления — деление без восстановления (по терминологии, принятой в 5.1.1, «второй способ») является по существу аналогом метода выполнения умножения

путем последовательного преобразования множителя, который был описан в 4.7.4 (метод Booth и Booth). Поэтому и в дополнительных кодах выполнение деления не вызывает особых затруднений.

Правило выполнения деления по второму методу, сформулированное в 5.1.1 на стр. 208, можно обобщить для работы с числами разных знаков при использовании дополнительных кодов следующим образом:

(а) сравнить знаки делимого и делителя; если они одинаковы, то вписать в знак частного (во второй разряд слева от запятой) «+», если различны — «—»; произвести в первом случае вычитание делителя из делимого, во втором — добавление делителя к делимому; если знак результата не совпадает со знаком делителя, то старшая значащая цифра частного есть нуль, в противном случае — единица; полученный результат в любом случае представляет собой первый остаток;

(б) удвоить предыдущий остаток;

(в) если знак предыдущего остатка не совпадает сознаком делителя, ,то добавить делитель, если совпадает — вычесть делитель; при этом образуется очередной остаток, знак которого определяет очередную цифру частного по правилам, аналогичным (а);

(г) повторять (б) и (в) до тех пор, пока не будут получены все цифры частного.

Естественно, что сложение и вычитание должны выполняться здесь полностью, с учетом алгебраических знаков делимого, делителя и остатков.

Пример. + 15/32: (—3/4) = —5/8,

$$\left.\begin{array}{l} 15/32 = 0.01111, \\ -3/4 = 1.01000, \\ -5/8 = 1.01100 \end{array}\right\} \text{(дополнительные коды).}$$

Так как знаки делимого и делителя различны, знак результата «—» (1*.*****...). По этой же причине первая операция — сложение

$$\begin{array}{r} 0.01111 \\ +1.01000 \\ \hline 1.10111 \end{array}$$ —1-й остаток.

Знаки 1-го остатка и делителя одинаковы. Поэтому старшая значащая цифра результата «1» (11.*****...), а следующая операция — вычитание. Эти и последующие операции запишем, как принято обычно, «углом», причем вместо вычитания делителя будем каждый раз производить сложение с его дополнительным кодом (т. е. с величиной 0.11000 = + 3/4):

добавление делителя	+ 0.01111 1.01000	1.01000 11.01011...
1-й остаток	1.10111	
удвоение 1-го остатка	11.01110	
вычитание делителя	+ 0.11000	
2-й остаток	0.00110	
удвоение 2-го остатка	00.01100	
добавление делителя	+ 1.01000	
3-остаток	1.10100	
удвоение 3-го остатка	11.01000	
вычитание делителя	+ 0.11000	
4-й остаток	0.00000	
удвоение 4-го остатка	00.00000	
добавление делителя	+ 1.01000	
5-й остаток	1.01000	
удвоение 5-го остатка	10.10000	
вычитание делителя	+ 0.11000	
6-й остаток	1.01000	
	.....	

Заметим, что результат получился у нас с погрешностью в единицу младшего разряда.

Читателю предлагается самостоятельно рассмотреть примеры с другими комбинациями алгебраических знаков, а также примеры, в которых результат деления получился бы по абсолютной величине больше единицы (но меньше двух).

2°. Выполнение деления чисел с плавающей запятой, как и выполнение умножения с плавающей запятой, не вызывает особых трудностей.

Если

$$\beta = 2^b B,$$
$$\gamma = 2^c C,$$

то

$$\beta : \gamma = 2^{b-c}(B : C).$$

Таким образом, порядок частного равен разности порядков делимого и делителя, а мантисса частного — частному от деления мантисс делимого и делителя.

Если исходные числа нормализованы, так что

$$^1/_2 \le |B| < 1 \text{ и } 1/2 \le |C| < 1,$$

то

$$^1/_2 < |B:C| < 2.$$

Для нормализации результата может потребоваться сдвиг мантиссы результата на 1 разряд вправо (т. е. деление ее пополам); одновременно нужно, конечно, добавить единицу к порядку результата. Во многих случаях удобно вместо выполнения сдвига мантиссы результата вправо заблокировать последний сдвиг мантиссы влево (если по ходу деления в каждом цикле частное сдвигается на 1 разряд влево— см., например, схемы рис. 5-1, а и *б*).

При вычислении разности порядков может получиться отрицательный или положительный результат, превышающий по абсолютной величине максимально допустимую величину порядка. Первый случай соответствует получению нулевого результата (с той точностью, с которой оперирует машина), второй случай свидетельствует о переполнении разрядной сетки. В обоих случаях должна быть, разумеется, принята во внимание и та единица, которая, если необходимо, добавляется к разности порядков при нормализации результата.

5.1.8. Выполнение деления в системе счисления с основанием п >2.

Способы выполнения деления в системе счисления с основанием *n* не содержат каких-либо существенно новых идей по сравнению с рассмотренными ранее способами деления в двоичной системе.

Последующее изложение будем вести применительно к делению *n*-ичных чисел с фиксированной запятой, представленных прямым кодом.

При выполнении деления *первым способом* (см. 5.1.1), когда определение очередной цифры частного и следующего остатка выполняется с помощью сравнений и вычитаний, отличие от двоичной системы состоит только в количестве сравнений и вычитаний, входящих в каждый цикл деления.

В двоичной системе счисления каждый цикл деления включал всегда одно сравнение; если при сравнении предыдущего остатка с делителем после соответствующих сдвигов оказывалось, что делитель больше остатка, то это означало, что очередная цифра частного есть 0, вычитания производить не нужно; если делитель оказывался меньше остатка, то очередная цифра частного есть 1, и нужно было произвести одно вычитание делителя из остатка.

В системе счисления с основанием я первое сравнение делителя с остатком выяснит только, не равна ли нулю очередная цифра частного: если делитель больше остатка, то очередная цифра частного есть 0, и вычитания производить не нужно; если делитель меньше остатка, то очередная цифра частного не равна нулю (но еще не известно, какова она), и нужно произвести первое вычитание делителя из остатка. Полученный результат вычитания (первый промежуточный остаток) далее снова сравнивается с делителем. Это второе сравнение выясняет, не равна ли очередная цифра частного единице: если делитель больше, чем первый промежуточный остаток, то это значит, что очередная цифра частного есть 1, и вычитания больше производить не нужно; в противном случае очередная цифра частного не равна единице (но еще не известно, какова она), нужно произвести второе вычитание делителя и перейти к следующему сравнению. Третье сравнение выяснит, не есть ли очередная цифра частного 2 и т. д. Указанный порядок нарушается только в конце (если до этого доходит дело): если (n — 1)-е сравнение выяснит, что очередная цифра частного не есть n — 2, то это значит, что она есть n — 1 (других возможностей не остается), и n-е сравнение производить уже не имеет смысла.

Таким образом, каждый цикл выполнения деления в системе счисления с основанием *n* содержит от одного до (n — 1) сравнений и от нуля до n — 1 вычитаний (вычитаний обычно на одно меньше, чем сравнений, кроме случая, когда очередная цифра частного оказывается равной n — 1). Подобно тому как *n*-ичное умножение сводится к серии сдвигов и суммирований (до n — 1 суммирований в каждом цикле), так *n*-ичное деление сводится к серии сдвигов и сравнений-вычитаний (до n — 1 сравнений-вычитаний в каждом цикле).

Как и в двоичной системе, указанный способ деления удобен только в параллельных устройствах и только при условии, что устройство сумматора позволяет отделить процесс формирования сигналов переноса от выдачи суммы в регистр-накопитель. Только при этих условиях легко реализовать сравнение как часть операции вычитания, которая либо доводится до конца, если это необходимо, либо прерывается на этапе образования сигналов переноса (см. 5.1.1). Если таких возможностей нет, то лучше воспользоваться *вторым способом* выполнения деления.

В полном соответствии с тем объяснением второго способа деления, которое приведено в 5.1.3 (стр. 211-212), этот способ применительно к *n*-ичной системе счисления должен быть изложен следующим образом. Если к началу некоторого цикла деления предыдущий остаток положителен, то в данном цикле должна производиться серия вычитаний делителя из остатка; вычитание нужно производить до тех пор, пока не получится отрицательный результат (но не более n—1 раз, хотя бы результат и остался положительным). Если к началу некоторого цикла деления предыдущий остаток отрицателен, то в данном цикле деления должна производиться серия сложений делителя с остатком; сложение нужно выполнять до тех пор, пока не получится положительный результат, но не более n — 1 раз (хотя бы результат и остался отрицательным). Количество вычитаний, произведенных в данном цикле деления, есть количество положительных единиц, содержащихся в очередной цифре частного, количество сложений — количество отрицательных единиц; результат сложении или вычитаний в любом случае представляет собой очередной остаток, который перед следующим циклом должен быть сдвинут на 1 разряд влево (или делитель нужно сдвинуть на разряд вправо). Частное получается в форме записи цифрами —(n—1), —(n—2), . . ., —2, —1, +1, +2, . . ., + (n — 2), + (n — 1); впрочем, нетрудно предусмотреть преобразование частного в обычную форму

записи (с цифрами 0, 1, 2, . . ., (n— 1)) по ходу выполнения деления — так, как в 5.1.1 это сделано для двоичной системы.

При осуществлении этого способа в машине существенно, что либо операция ведется с фиксированной запятой, так что результат, больший единицы, не должен получаться, либо она выполняется над нормализованными числами. В противном случае потребовались бы существенные усложнения процесса, чтобы получить правильные цифры в целой части частного (кроме первого разряда слева от запятой, который получается автоматически).

П р и м е р ы (в десятичной системе счисления).

1.

	0,025 | 0,625	
1-й цикл (старшая цифра +1)	0,625 | 1,961999...	
	(1-й остаток)	−0,600
	(сдвиг влево)	−6,000
		+0,625
		−5,375
		+0,625
		−4,750
		+0,625
		−4,125
		+0,625
2-й цикл (вторая цифра — 9)		−3,500
		+0,625
		−2,875
		+0,625
		−2,250
		+0,625
		−1,625
		+0,625
		−1,000
		+0,625
	(2-й остаток)	−0,375
	(сдвиг влево)	−3,750
		+0,625
		−3,125
		+0,625
		−2,500
		+0,625
3-й цикл (третья цифра — 6)		−1,875
		+0,625
		−1,250
		+0,625
		−0,625
		+0,625
	(3-й остаток)	+0,000
	(сдвиг влево)	+0,000
4-й цикл (четвертая цифра +1)		−0,625
	(4-й остаток)	−0,625
	(сдвиг влево)	−6,250
		+0,625
		−5,625
		+0,625
		−5,000
		+0,625
		−4,375
		+0,625
		−3,750
5-й цикл (пятая цифра — 9)		+0,625
		−3,125
		+0,625
		−2,500
		+0,625
		−1,875
		+0,625
		−1,250
		+0,625
	(5-й остаток).	−0,625
	(сдвиг влево)	−6,250
		.....

Полученный результат $(1.\overline{961999}...)$ с точностью до единицы младшего разряда равен 0,04, как и требуется.

2.

		0,625	0,500
		−0,500	2,86999...
1-й цикл (первая цифра частного + 2)		0,125	
		−0,500	
	(1-й остаток)	−0,375	
	(сдвиг влево)	−3,750	
		+0,500	
		−3,250	
		+0,500	
		−2,750	
		+0,500	
		−2,250	
		+0,500	
2-й цикл (вторая цифра частного — 8)		−1,750	
		+0,500	
		−1,250	
		+0,500	
		−0,750	
		+0,500	
		−0,250	
		+0,500	
	(2-й остаток)	+0,250	
	(сдвиг влево)	+2,500	
		−0,500	
		+2,000	
		−0,500	
		+1,500	
		−0,500	
3-й цикл (третья цифра частного + 6)		+1,000	
		−0,500	
		+0,500	
		−0,500	
		+0,000	
		−0,500	
	(3-й остаток)	−0,500	
	(сдвиг влево)	−5,000	
		.....	

(Во всех последующих циклах цифры частного будут получаться равными — 9, остатки — такие же, как третий остаток.) С точностью до единицы младшего разряда полученный результат $2.\overline{8}6\overline{999}...$ равен 1,25, как это и требуется.

Во многих случаях оборудование, предназначенное для ускорения умножения в n-ичной системе, удается использовать и для ускорения деления. Таково, например, оборудование цепей для получения чисел, кратных множимому (при делений — кратных делителю), цепи для умножения множимого (делителя) на особые множители и др. Способы ускорения деления, описанные выше применительно к двоичной системе, в той или иной степени пригодны и в системе счисления с любым другим основанием a.

Интересные возможности дает использование избыточных цифр частного при выполнении деления в системе счисления с основанием $n > 2$. Допустив существование избыточных цифр, можно вести определение очередной цифры частного на основе приблизительной оценки соотношения остатка и делителя, рассматривая только их старшие разряды*).

Например, если в десятичной системе допустить, что цифрами частного могут быть — 9, — 8, — 7, . . ., — 1, 0, + 1, . . ., + 7, + 8, + 9, то очередная цифра частного c_i должна выбираться по правилу:

если $-10C_i \le B_i \le -8C_i$, то $a_i = -9$,
если $-9C_i \le B_i \le -7C_i$, то $a_i = -8$,
.....
если $-2C_i \le B_i \le 0$, то $a_i = -1$,
если $-C_i \le B_i \le +C_i$, то $a_i = 0$,
если $0 \le B_i \le +2C_i$, то $a_i = +1$,
.....
если $7C_i \le B_i \le 9C_i$, то $a_i = +8$,
если $8C_i \le B_i \le 10C_i$, то $a_i = +9$

*) С этой целью, собственно, идея избыточных цифр частного и предлагалась проф. Дж. Робертсоном — см. сноску на стр. 210.

(где B_i — очередной остаток, C_i—делитель, взятый с соответствующим весом).

Это правило получено из того условия, чтобы следующий остаток B_{i+1}, равный $B_i - a_iC_i$, удовлетворял бы неравенствам:

$$\frac{n}{n-1}\alpha_1 C_i \le B_i \le \frac{n}{n-1}\alpha_k C_i,$$

которые являются обобщением неравенств, приведенных к стр. 212 для $n = 2$ (здесь n — основание системы счисления, α_1 — наименьшая допустимая цифра, α_k — наибольшая допустимая цифра частного; в данном случае $n = 10$, $\alpha_1 = -9$, $\alpha_k = +9$). Приведенные правила дают довольно широкую свободу для выбора цифр частного: в интервале $-9C_i \le B_i \le -8C_i$ в качестве цифры частного можно выбирать либо — 9, либо — 8, в интервале $-8C_i \le B_i \le -7C_i$ —либо—8, либо —7 и т. д. Нетрудно проверить, что когда делитель нормализован ($0{,}1 \le C \le 1$), а в каждом цикле деления производится сдвиг остатка влево (вместо сдвига делителя вправо), то для определения очередной цифры частного по указанным правилам достаточно рассмотреть только 2 старших разряда остатка и 2 старших разряда делителя. Иначе говоря, очередной остаток, взятый с соответствующим весом, и делитель достаточно знать с точностью только до 0,01. Следовательно, для определения очередной цифры частного нет нужды производить последовательные добавления или вычитания делителя: ее можно определить в один такт — непосредственной расшифровкой пары старших разрядов остатка и делителя.

Правда, схема расшифровки получилась бы довольно сложной, потому что пара разрядов остатка может содержать 100 различных комбинаций цифр, и 90 различных комбинаций цифр может содержаться в паре старших разрядов делителя. Кроме того, отыскание очередной цифрычастного—это еще половина дела; дальше нужно найти следующий остаток, а для выполнения этой операции водин такт нужно было бы заготовить заранее все девять кратных делителя (числа $1C$, $2C$,…, $9C$).

Однако описанный способ можно комбинировать с обычным способом деления. Например, если заранее заготовлены два числа, кратных делителю, — $2C$ и $5C$ (произведения делителя на особые множители), то при первоначальной расшифровке старших разрядов остатка и делителя желательно выяснить только, в каком интервале лежит очередная цифра частного: от — 9 до — 7, от — 7 до — 5, от — 5 до — 2, от — 2 до 0, от 0 до + 2, от + 2 до + 5, от +5 до + 7, от + 7 до + 9. Затем производится соответственно добавление или вычитание величины $2C$, или $5C$ или обеих этих величин (если цифра частного находится в интервалах от — 9 до — 7 или от + 7 до + 9), а окончательное уточнение цифры частного и величины очередного остатка выполняется последовательными добавлениями или вычитаниями делителя C. Схема предварительной расшифровки старших разрядов остатка и делителя получается при этом сравнительно простой, а максимальное количество сложений или вычитаний, приходящихся на 1 цикл деления, сокращается с 9 для обычного способа до 3.

Аналогичные идеи, развиваемые для системы счисления с основанием n=4, дают фактически возможность ускорить двоичное деление. В четверичной системе избыточным является уже набор цифр частного —2, — 1, 0, + 1, + 2, содержащий 5 различных цифр. Если каждую пару двоичных разрядов рассматривать как один четверичный разряд, то расшифровка нескольких старших двоичных разрядов остатка и делителя позволит в один такт определить одну четверичную цифру, т. е. сразу пару двоичных цифр частного. Все числа, кратные делителю ($0{,}1C$ и $2C$), которые необходимы для отыскания остатка в один такт, также легко доступны ($2C$ —делитель, сдвинутый на 1 разряд влево). Этот способ может быть применен для сокращения вдвое длительности деления в схеме рис. 5-3 (о чем упоминалось в 5.1.4), а также для сокращения количества оборудования и увеличения быстродействия схемы рис. 5-5 по аналогии с тем, как это делалось при выполнении умножения (см. 4.5.3).

Аналогичным образом могли бы быть использованы системы счисления с основаниями 8, 16 и т. д. Как было указано В. А. Бриком, при этом по ряду соображений может оказаться выгодным выбирать в качестве допустимых цифр частного неравноотстоящие и даже не обязательно целые числа. Например, для шестнадцатеричной системы вместо набора допустимых цифр ± 15, ± 14, . . ., ± 1, 0, построенного по аналогии с теми наборами, которые предлагались выше для двоичной, десятичной и четверичной систем, им рекомендуется набор цифр

$$\pm 15, \pm 14\frac{1}{2}, \pm 14, \pm 13\frac{1}{2}, \pm 13, \pm 12, \pm 11, \pm 9\frac{1}{2}, \pm 8, \pm 6\frac{1}{2}, \pm 5, \pm 4, \pm 2\frac{1}{2}, \pm 1,$$

позволяющий получить некоторые упрощения как в схеме выбора очередной цифры частного, так и в схеме преобразования частного в обычную двоичную систему.

Ускорение двоичного деления при этом может быть достигнуто весьма значительное.

5.1.9. Итеративные методы выполнения деления.

В разделах 5.1.2—5.1.5 были последовательно рассмотрены логический метод ускорения деления и аппаратные методы 1-го и 2-го порядка, которые рекомендовалось применять совместно с аналогичными методами ускорения умножения.

Однако, например, аппаратный метод 2-го порядка, рассмотренный в 5.1.5, обладает тем недостатком, что он может применяться только на основе множительного устройства рис. 4-11 и не позволяет использовать более быстродействующие многослойные построения множительных устройств (см. раздел 4.5.4); таким образом

оказывается, что ради получения высокой скорости выполнения деления (примерно такой же, как скорость выполнения умножения), нам приходится снижать скорость умножения.

В некоторых современных разработках арифметических устройств такой путь считается нецелесообразным. Вместо этого предпочитают получить максимально возможную скорость выполнения умножения, не связывая конструкцию множительного устройства с возможностью выполнения деления, а деление выполнять итеративным путем, через умножение. Если при этом для ускорения умножения затрачено большое количество аппаратуры, то дополнительное оборудование для выполнения деления сравнительно невелико. Скорость деления получается в 3—5 раз меньше скорости умножения.

По существу при этом применяются те же итеративные методы, которые используются для построения подпрограмм деления в машинах, где деление в качестве самостоятельной операции отсутствует. Однако операция деления в этомслучае выполняется без дополнительных обращений к запоминающему устройству и поэтому быстрее, чем по подпрограмме. Промежуточные результаты хранятся в регистрах арифметического устройства, а последовательность операций задается схемой управления арифметического устройства.

Наиболее известны итеративные формулы, позволяющие вычислить обратную величину от делителя; процесс деления завершается умножением обратной величины делителя на делимое, в результате чего находится частное.

Обозначим, как обычно, делитель через С. Предположим, что имеется некоторое число x_i, представляющее обратную величину делителя $1/C$ с относительной погрешностью δ_i,

$$\delta_i = \frac{\frac{1}{C} - x_i}{1/C} = 1 - Cx_i.$$

Отсюда легко найти искомую величину 1/С*:*

$$\frac{1}{C} = \frac{x_i}{1-\delta_i} = \frac{x_i}{1-(1-Cx_i}.$$

Если полагать, что погрешность δ_i достаточно мала, то

$$\frac{x_i}{1-\delta_i} \approx x_i(1+\delta_i) = (2 - Cx_i).$$

Примем эту величину за следующее приближение к искомой величине $1/C$*:*

$$x_{i+1} = x_i(2—Cx_i). \qquad (*)$$

Это и есть основная формула итерации.

Нетрудно вычислить относительную погрешность, с которой x_{i+1} выражает величину $1/C$. Если

$$x_i = \frac{1}{C}(1-\delta_i),$$

то

$$x_{i+1} = \frac{1}{C}(1-\delta_i)[2 - C\frac{1}{C}(1-\delta_i)] = \frac{1}{C}(1-\delta_i^2).$$

Таким образом, если относительная погрешность i-го приближения есть δ_i, то относительная погрешность $(i + 1)$-го приближения будет $\delta_{i+1} = \delta_i^2$, $(i + 2)$-го — $\delta_{i+2} = \delta_i^4$ и т. д.

Ясно, что условием сходимости иттераций является такой выбор начального приближения x_0, для которого относительная ошибка δ_0 по абсолютной величине меньше единицы:

$$|\delta_0| = |1—Cx_0| <1,$$

откуда

$$0 < Cx_0 < 2$$

или

$$0<x_0<2/C.$$

Если C — нормализованное положительное двоичное число, $1/2 \le C < 1$, то требуется

$$0 < x_0 < 2.$$

Например, можно принять $x_0 = 1$ для всех чисел C из указанного диапазона. Однако при этом потребовалось бы очень много итераций для достижения необходимой точности.

Если δ_i достаточно мало по сравнению с единицей, то, грубо говоря, количество верных знаков удваивается при каждой итерации.

В составе оборудования арифметического устройства для выполнения деления по этому методу нужно иметь таблицу для нахождения начального приближения (x_0), один или, может быть, два дополнительных регистра (для хранения в течение всего процесса деления величины делимого и для хранения промежуточных результатов Cx_i и затем 2 — Cx_i в каждой итерации), а также, разумеется, соответствующие цепи в схеме управления. Желательно, кроме того, предусмотреть возможность получать результат умножения в дополнительном коде, что позволит в результате умножения $C \times x_i$ получать сразу величину 2 — Cx_i.

Длительность выполнения деления при этом зависит от точности начального приближения. Если, например, мантиссы нормализованных чисел, с которыми оперирует машина, содержат по 32 двоичных разряда и такова же должна быть точность деления, а первое приближение находится по таблице с 8 верными двоичными знаками, то процесс деления должен включать . две итерации; поскольку каждая итерация состоит из двух умножений, а в конце деления требуется еще одно умножение (обратной величины делителя на делимое), то всего деление будет выполняться за время 5 умножений.

Поскольку количество оборудования, необходимого для получения начального приближения, растет примерно как kn^k, где n — основание системы счисления, k — количество верных знаков в начальном приближении, то иногда выгоднее для ускорения деления использовать более сложные формулы итераций. Например, если вместо приближенного равенства

$$\frac{x_i}{1-\delta_i} \approx x_i(1+\delta_i)$$

воспользоваться выражениями

$$\frac{x_i}{1-\delta_i} \approx x_i(1+\delta_i+\delta_i^2)$$

или

$$\frac{x_i}{1-\delta_i} \approx x_i(1+\delta_i+\delta_i^2+\delta_i^3),$$

то, подставляя $\delta_i = 1 - Cx_i$, получим вместо итерационной формулы (*) итерационные формулы

$$x_{i+1} = x_i[3(1-Cx_i) + (Cx_i)^2] \qquad (**)$$

и

$$x_{i+1} = x_i[6(4 - Cx_i) + (Cx_i)^2(4 - Cx_i) - 20]. \qquad (***)$$

Формулу (***) можно получить также из (*), подставляя в выражение x_{i+2} через x_{i+1} выражение x_{i+1} через x_i и затем заменяя индекс $i + 2$ на $i + 1$:

из

$$x_{i+2} = x_{i+1}(2 - Cx_{i+1})$$

и

$$x_{i+1} = x_i(2 - Cx_i)$$

получим

$$x_{i+2} = x_i(2 - Cx_i)[2 - Cx_i(2 - Cx_i)],$$

откуда и следует (***).

Аналогично предыдущему нетрудно убедиться, что если, «относительная погрешность величины x_i есть δ_i, то относительная погрешность величины x_{i+1}, вычисленной по формуле (**),есть $\delta_{i+1} = \delta_i^3$: из $x_i = \frac{1}{C}(1-\delta_i)$ и формулы (**) следует

$$x_{i+1} = \frac{1}{C}(1-\delta_i)\left\{3[1 - C\frac{1}{C}(1-\delta_i)] + [C\frac{1}{C}(1-\delta_i)]^2\right\} = \frac{1}{C}(1-\delta_i^3).$$

Точно так же относительная погрешность величины x_{i+1}, вычисленной по формуле (***), есть $\delta_{i+1} = \delta_i^4$.

Таким образом, каждая итерация по формуле (**), грубо говоря, утраивает количество верных знаков, а по формуле (***) — учетверяет.

При более аккуратном определении погрешности деления следует, конечно, принять во внимание также погрешности умножений, которые вносятся при вычислениях по (формулам итераций.

Существуют и другие способы выполнения деления через умножения.

5.2. Редкие операции

5.2.1. Извлечение квадратного корня.

Извлечение квадратного корня, как и другие операции, рассматриваемые в настоящем разделе, сравнительно редко реализуется в вычислительных машинах аппаратным путем. Большей частью извлечение квадратного корня осуществляется по подпрограммам, состоящим из других операций (умножений, делений, вычитаний), итерационным или другим способом.

Не ставя себе задачей дать сколько-нибудь полный обзор возможных аппаратных способов выполнения извлечения

корня, мы приведем лишь один способ и одно конкретное двоичное устройство, служащее этой цели*). Основная идея, которую должно подтвердить сказанное ниже,— это возможность выполнения операции извлечения корня с по для выполнения умножения и деления, почти без каких-либо специальных цепей.

Итак, пусть арифметическое устройство построено в соответствии со схемой рис. 5-1, *а* (см. стр. 206). Усложнения, которые нам придется ввести в схему, состоят в следующем:

— в регистре *С* вводятся цепи сдвига влево;

— в регистре *В* и в сумматоре вводится дополнительно по 2 младших разряда.

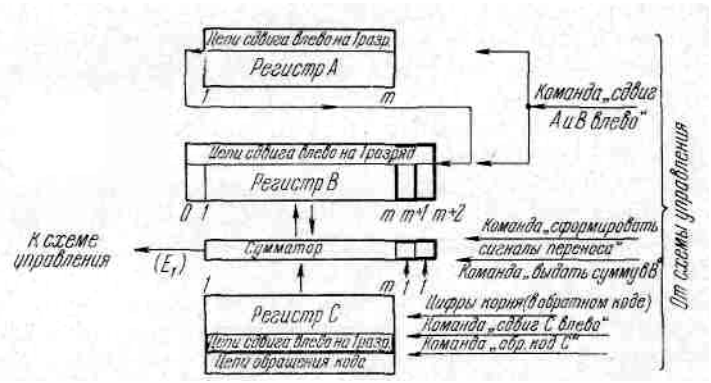


Рис. 5-6. Устройство для извлечения квадратного корня в двоичной системе. (Специальные цепи, отсутствующие в схеме рис. 5-1, *а,* обведены жирными линиями.)

Дополнительные разряды регистра *В* связаны цепями сдвига влево с регистром *А* и основными разрядами регистра *В*; когда в регистрах *А* и *В* производится одновременно сдвиг влево, старшая цифра из *А* передвигается в младший дополнительный разряд регистра *В,* а цифра из старшего дополнительного разряда передвигается в младший основной разряд регистра *В.* На входы «с» дополнительных разрядов сумматора постоянно подаются сигналы «11», ввиду чего дополнительные разряды сумматора могут быть по схеме проще, чем основные разряды.

Полная схема устройства приведена на рис. 5-6, причем цепи, введенные специально для извлечения квадратного корня, выделены жирными линиями. Видно, что количество дополнительного оборудования в схеме арифметического устройства незначительно.

Кроме того, некоторые специальные цепи потребуются в устройстве управления: цепи, задающие временную последовательность элементарных операций для извлечения корня, и сравнительно простая, как мы сейчас увидим, схема формирования цифр корня.

Работает устройство при извлечении корня следующим образом.

До начала операции подкоренное выражение устанавливается в регистре *А,* регистры *В* и *С* гасятся (устанавливаются в положение «0»), после чего в регистре *С* производится обращение кода (причем в нем устанавливается код 111...11).

Далее операция выполняется в точном соответствии с тем алгоритмом извлечения квадратного корня, который читатель, должно быть, помнит из курса средней школы.

Сначала необходимо снести 2 старшие цифры подкоренного выражения и извлечь из них квадратный корень, причем должна получиться первая цифра результата. С этой целью в начале операции производим два сдвига влево в регистрах *А* и *В*, в результате чего 2 старшие цифры подкоренного выражения перемещаются в младшие (дополнительные) разряды регистра *В*, и даем команду «сформировать единицы двоичного переноса».

Старшие разряды подкоренного выражения могут содержать одну из 4 комбинаций: 00, 01, 10 или 11. В первом случае (комбинация 00) корень из величины, представленной этими разрядами, равен нулю, в трех остальных случаях — единице. Поскольку в регистре *С* в это время находится величина **.**111...1'11, то на выходе старшего разряда сумматора не получится сигнала переноса только при наличии всех нулей в регистре *В.* Если в регистре *В* содержится комбинация .000...0'01 или .000...0'10 или .000...0'11 (где штрихом отделены младшие, дополнительные, разряды), то на выходе старшего разряда сумматора появится единица переноса.

Таким образом, результат образования единиц переноса определяет первую цифру результата a_1: если $E_1 = 0$, то первая цифра результата равна нулю, если $E_1 = 1$, то она равна единице.

Далее следует возвести в квадрат первую цифру результата и вычесть ее из величины, представленной старшими разрядами подкоренного выражения. В рассматриваемом устройстве для выполнения этих операций мы либо заканчиваем первый цикл на формировании единиц двоичного переноса (если $E_1 = 0$), либо доделываем до конца суммирование, выполняя выдачу суммы в регистр *В* (если $E_1 = 1$). В первом случае из величины, находящейся в *В,* мы фактически не вычли ничего, или, иными словами, вычли 0^2 ($a_1 = 0$). Во втором случае добавление .111...1'11 эквивалентно вычитанию .000...0'01 (так как .111...1' 11 есть дополнительный код от величины .000...0'01); таким образом, и в этом случае фактически вычитается квадрат первой цифры результата

*) Принцип построения такого устройства был предложен автором настоящей книги в 1958 г.

($a_1^2 = 1^2$ — 1). В любом случае в регистре B образуется первый остаток (B_1).

Для отыскания второй цифры корня в соответствии со «школьным» алгоритмом нужно выполнить следующие операции. К первому остатку (B_1) нужно снести (т. е. приписать справа) две очередные цифры подкоренного выражения; величину, которая при этом образуется, обозначим B'_1. Найденную ранее цифру корня нужно удвоить и к ней приписать справа максимально возможную цифру—такую, чтобы при умножении величины, которая получится при этом, на приписанную цифру произведение было меньше, чем B'_1. Эта цифра и будет второй цифрой корня.

В двоичной системе удвоение первой цифры корня можно осуществить, приписав к ней справа 0. Вторая цифра корня может быть либо нулем, либо единицей. Если вторая цифра корня есть 0, то величина, образованная путем приписки нуля справа от удвоенной первой цифры корня, имеет вид «$a_1$00»; если вторая цифра корня есть 1, то аналогичная величина имеет вид «$a_1$0l», а при умножении этой величины на приписанную справа цифру (т. е. на 1) получим ее же. Таким образом, в двоичной системе для определения второй цифры корня достаточно сравнить величину «$a_1$0l» с B'_1: если $a_1$01 больше B'_1, то вторая цифра корня (a_2) есть 0, в противном случае $a_2 = 1$.

В рассматриваемом устройстве указанные операции выполняются в следующем порядке. В регистрах A и B производятся два сдвига влево; при этом первый остаток передвигается в основные разряды регистра B, а справа к нему цифры подкоренного выражения; таким образом, в регистре B образуется величина B'_1. В регистре С производится один сдвиг влево, причем в младший разряд вдвигается в обратном коде первая цифра корня ($\bar{a}_1$); поскольку во всех старших разрядах регистра C сохраняются единицы, а в несуществующих дополнительных разрядах C как бы закреплена комбинация «11» (эти цифры поданы на входы с дополнительных разрядов сумматора), то в регистре C образуется величина $11\ldots1\bar{a}_1'11$, которая является дополнительным кодом величины $00...0a_1'0l$. Далее выполняется формирование сигналов двоичного переноса. Если величина $a_1$01 больше B'_1, то $E_1 = 0$; вторая цифра корня при этом 0, а в регистре B сразу находится второй остаток ($B_2 = B'_1$).

Если $a_1$01 меньше B'_1, то $E_1 = 1$; вторая цифра корня при этом 1, а для получения второго остатка в регистре B необходимо доделать до конца суммирование, произведя выдачу суммы в регистр B; при этом фактически произойдет вычитание величины $a_1$01 из B_1'.

Для получения третьей цифры корня нужно снова произвести два сдвига влево в регистрах A и B, один сдвиг влево в регистре C (причем в младший разряд C вдвигается $\bar{a}_2$ — вторая цифра корня в обратном коде) и затем сформировать сигналы двоичного переноса и т. д.

Таким образом, процесс извлечения квадратного корня сводится к повторению однообразных циклов, каждый из которых включает 2 сдвига в регистрах A и B, один сдвиг в регистре C, сравнение содержимого регистров B и C и, если необходимо, вычитание (точнее, суммирование с обратным кодом).

Первый цикл формально тоже не отличается от всех последующих, хотя смысл операций в нем иной. Для получения m цифр корня нужно выполнить m таких циклов; при этом в течение последних $m/2$ циклов из регистра A в младшие разряды регистра B будут передвигаться нули. Результат операции в обратном коде получается в регистре C, куда мы каждый раз вдвигаем в обратном коде очередные цифры корня. Произведя в конце обращение кода C, можно получить значение корня в прямом коде.

П р и м е р . Извлечь корень из числа 25/64 (результат должен быть 5/8).

$$\frac{25}{64} = .011001$$

$$\frac{5}{8} = .101000$$

Для примера предположим, что регистры содержат по 6 разрядов (не считая 2 дополнительных разрядов справа); для получения 6 цифр корня нужно выполнить б циклов, описанных выше:

1-й цикл

после 2 сдвигов A и B:
в регистре B .000000'01
в регистре C .111111(11)

$E_1 = 1$ $(a_1 = 1)$

после выдачи суммы в B:
в регистре B — величина B_1 .000000'00

2-й цикл

после 2 сдвигов A и B:
в регистре B — величина B_1' .000000'10
после сдвига в C .111110 (11) (первая цифра корня в обратном коде)

$E_1 = 0$ $(a_2 = 0)$

3-й цикл

после 2 сдвигов A и B:
в регистре B — величина B_2' .000010'01
после сдвига в C .111101 (11) (вторая цифра корня в обратном коде)

$E_1 = 1$ $(a_3 = 1)$

после выдачи суммы в B:
в регистре B — величина B_3 .000000'00

4-й цикл

после 2 сдвигов A и B:
в регистре B — величина B_3' .000000'00
после сдвига в C .111010 (11) (третья цифра корня в обратном коде)

$E_1 = 0$ $(a_4 = 0)$

5-й цикл

после 2 сдвигов A и B:
в регистре B — величина B_4' .000000'00
после сдвига в C .110101 (11) (четвертая цифра корня в обратном коде)

$E_1 = 0$ $(a_5 = 0)$

6-й цикл

после 2 сдвигов A и B:
в регистре B — величина B_5' .000000'00
после сдвига в C .101011 (11) (пятая цифра корня в обратном коде)

$E_1 = 0$ $(a_6 = 0)$

Вдвигаем 6-ю цифру корня в регистр C, после чего производим обращение кода C:

после сдвига в C .010111 (11) (шестая цифра корня в обратном коде)
после обращения кода .101000 — результат.

Процесс извлечения квадратного корня очень похож на процесс выполнения деления, и во многих случаях, вероятно, методы и средства ускорения деления могли бы пригодиться и для ускорения этой операции. В частности, можно воспользоваться идеей применения избыточных цифр частного — в данном случае избыточных цифр корня*). Впрочем, вопросы ускорения операции извлечения квадратного корня, как и вообще вопросы ее выполнения, разработаны к настоящему времени мало.

5.2.2. Преобразование чисел из одной системы счисления в другую.

Как и извлечение квадратного корня, эта операция выполняется большей частью не аппаратным, а программным путем. В отдельных случаях в составе вычислительной машины предусматривается специальное устройство для перевода чисел из одной системы счисления в другую; еще реже преобразование чисел строится как специальная операция основного арифметического устройства.

На рис. 5-7 показано одно из возможных построений специального устройства для преобразования чисел из десятичной системы счисления в двоичную и из двоичной в десятичную.

Устройство содержит два триггерных регистра — двоичный и десятичный. Первый из них, как показывает название, предназначен для хранения двоичных чисел; во втором каждая четверка двоичных разрядов («тетрада») предназначается для хранения одной десятичной цифры, а десятичный регистр в целом предназначен для хранения десятичных чисел. Количество двоичных элементов различно: если двоичный регистр содержит m

Рис. 5-7. Построение устройства для преобразования чисел из десятичной системы в двоичную и из двоичной в десятичную.

*) Работа М. Надлера — см. сноску на стр. 210

двоичных разрядов, то для сохранения той же точности в десятичном числе десятичный регистр должен содержать примерно $m\lg 2$ тетрад или $4m \lg 2$ двоичных элементов (см. 1.2.2).

В обоих регистрах имеются цепи сдвига вправо и влево на 1 двоичный разряд, которые замыкают регистры в кольцо: при выполнении сдвига влево старшая двоичная цифра, выдвигаемая из десятичного регистра, передвигается в младший разряд двоичного регистра, а старшая цифра из двоичного регистра — в младший двоичный разряд десятичного регистра; при выполнении сдвига вправо младшая двоичная цифра из десятичного регистра передвигается в старший разряд двоичного регистра, а младшая цифра из двоичного регистра — в старший двоичный разряд десятичного регистра.

Десятичный регистр оборудован, кроме того, цепями коррекции, которые позволяют производить в каждой тетраде добавление или вычитание числа 3. Цепи коррекции устроены так, что выполнение одного сдвига вправо и одного корректирующего шага эквивалентно делению пополам числа, находящегося в десятичном регистре, а выполнение одного сдвига влево и одного корректирующего шага эквивалентно удвоению числа в десятичном регистре.

Рассмотрим сначала устройство цепей коррекции.

Представим себе, что в десятичном регистре находится некоторое число, представленное простейшим двоично-десятичным кодом с весами 8,4,2, 1 (см. 1.6.1). Если это число сдвинуть вправо на 1 двоичный разряд, то десятичная цифра, имевшаяся в каждой тетраде, окажется разделенной пополам. При этом если исходная цифра была нечетной, то единица из младшего двоичного разряда данной тетрады передвигается в соседний справа десятичный разряд; иными словами, если при делении пополам данной десятичной цифры получается результат с дробью (с половиной), то дробная часть в данном десятичном разряде теряется и переносится в следующий, младший разряд. Все это пока так и должно быть при делении десятичного числа пополам. Однако когда из соседнего старшего разряда в некоторую тетраду переносится единица, то она попадает здесь в двоичный разряде весом 8, т. е. увеличивает значение десятичной цифры в данной тетраде на 8 единиц; в действительности же половина предыдущего разряда составляет не 8, а 5 единиц данного десятичного разряда.

Следовательно:

корректирующий шаг после сдвига вправо должен состоять в вычитании числа 3 *в тех тетрадах, в старшие двоичные разряды которых при сдвиге вправо попали единицы.*

П р и м е р . Пусть в десятичном регистре находится число .3678.

В двоично-десятичной форме оно имеет вид

.0011 0110 0111 1000.

Сдвиг вправо на один разряд дает

.0001 1011 0011 1100.

При выполнении корректирующего шага получим

.**0**001	**1**011	**0**011	**1**100
—	—3	—	—3
.0001	1000	0011	1001

что является изображением числа .1839 — половины исходного числа. (Жирным шрифтом выделены цифры, определяющие характер коррекции в данной тетраде.)

Удвоение чисел в десятичном регистре удобнее производить, если десятичные числа изображаются двоичным кодом с излишком 3 (см. 1.6.2). Для перехода от обычного двоичного кода (8, 4, 2, 1) к коду с излишком 3 достаточно произвести добавление троек по всем тетрадам, а для возврата к обычному двоичному коду —вычитание троек. Как мы увидим из дальнейшего, такой переход придется выполнять не более одного раза за всю операцию преобразования числа из одной системы счисления в другую. Зато цепи коррекции при использовании кода с излишком 3 получаются значительно проще, чем в коде «8,4,2,1».

Итак, пусть в десятичном регистре находится некоторое число, представленное двоично-десятичным кодом с излишком 3. При сдвиге влево на 1 двоичный разряд в каждой тетраде происходит удвоение находившейся в ней ранее величины. Если десятичная цифра в данной тетраде была больше или равна 5, то в изображении кодом с излишком 3 она была больше или равна 8, т. е. содержала единицу в старшем двоичном разряде; при сдвиге влево эта единица передвинется в младший двоичный разряд соседней слева тетрады и увеличит на единицу находящуюся в ней величину, как это и требуется при удвоении десятичного числа. Однако результат сдвига влево нужно еще подправить, чтобы получить удвоенное исходное число. Прежде всего ясно, что по всем тетрадам следует вычесть тройки, так как при сдвиге влево мы получили вместо излишка 3 излишек 6. Кроме того, в тех тетрадах, в которых исходная цифра была больше или равна 5, следовало после передачи единицы в соседний слева десятичный разряд уменьшить величину в данной тетраде на 10; в действительности же мы отбросили единицу двоичного разряда с весом 16; следовательно, в этих тетрадах нужно еще произвести добавление шестерки; добавление шестерки и вычитание тройки эквивалентно, очевидно, добавлению тройки. Поэтому окончательно правило коррекции после сдвига влево выглядит следующим образом:

корректирующий шаг после сдвига влево должен состоять из вычитания числа 3 *в тех тетрадах, из которых при*

сдвиге влево не была выдвинута единица в соседний (старший) десятичный разряд, и добавления числа 3 *в тех тетрадах, из которых при сдвиге влево выдвигалась единица в соседний десятичный разряд.*

П р и м е р . Пусть в десятичном регистре находится число .1839. В двоично-десятичной форме (в коде с излишком 3) оно имеет вид

.0100 1011 0110 1100,

сдвиг влево на один разряд дает

.1001 0110 1101 1000.

При выполнении корректирующего шага получим

(**0**) .100**1** 011**0** 110**1** 1000
—3 +3 —3 +3
.0110 1001 1010 1011

что является изображением в коде с излишком 3 числа .3678 — удвоенного исходного числа. (Жирным шрифтом выделены цифры, определяющие характер коррекции в соседней справа тетраде.)

Теперь перейдем непосредственно к рассмотрению того, как в описываемом устройстве выполняются операции по преобразованию чисел из одной системы счисления в другую. В том виде, как устройство изображено на рис. 5-7, оно пригодно для операций с числами с фиксированной запятой — либо с дробными с запятой, фиксированной перед старшим разрядом, либо с целыми с запятой, фиксированной после младшего разряда. Операции над целыми и дробными числами будут идти по-разному. Алгоритмы преобразования, используемые в рассматриваемом устройстве, в точности те же, что и в 1.2.4 рекомендовались для выполнения преобразования вручную.

Для преобразования дробного числа из десятичной системы счисления в двоичную исходное число помещается в десятичный регистр. Если в исходной форме число было представлено простым двоичным кодом (кодом «8,4,2,1»), то через цепи коррекции выполняется добавление троек по всем тетрадам одновременно, в результате чего оно преобразуется в код с излишком 3. Двоичный регистр в исходном состоянии погашен. Затем m раз подряд повторяется цикл, состоящий из сдвига влево на 1 двоичный разряд в десятичном и двоичном регистрах и одного корректирующего шага в десятичном регистре (где m — количество разрядов в д в о и ч н о м регистре). Каждый сдвиг влево с последующим корректирующим шагом эквивалентен удвоению числа в десятичном регистре. При этом за пределы десятичного регистра (в несуществующий разряд, находящийся слева от запятой) выходят по очереди цифры двоичного представления числа: при первом удвоении исходного числа — цифра, относящаяся к разряду с весом 1/2, при следующем удвоении содержимого десятичного регистра — цифра, относящаяся к разряду с весом 1/4, и т. д. Поскольку сдвиги в двоичном и десятичном регистрах выполняются всегда одновременно, а цепи сдвига замкнуты в кольцо, эти цифры вдвигаются в двоичный регистр со стороны младших разрядов. После m-го цикла первая двоичная цифра займет свое место в старшем разряде двоичного регистра, за ней в двоичном регистре будут расположены все остальные разряды двоичного представления числа.

Для *преобразования дробного числа из двоичной системы в десятичную* исходное число помещается в двоичный регистр. Десятичный регистр в исходном состоянии погашен. Затем m раз повторяется цикл, состоящий из сдвига вправо на 1 разряд в двоичном и десятичном регистрах и одного корректирующего шага в десятичном регистре (где m — количество разрядов в двоичном регистре). Если младшая цифра (a_m) в исходном (двоичном) представлении числа есть нуль, то после выполнения 1-го цикла в десятичном регистре сохранится нуль. Если же $a_m = 1$, то после первого сдвига вправо в старшей тетраде десятичного регистра появится комбинация 1000, а после коррекции (вычитания тройки) получится комбинация 0101, соответствующая цифре «5»; так как вес первой тетрады (старшего десятичного разряда) есть 1/10, то в этом случае после первого цикла в десятичном регистре получится число 0,5. Таким образом, в результате первого цикла в десятичном регистре получится величина $a_m \cdot 2^{-1}$. При выполнении второго цикла эта величина в десятичном регистре делится пополам и к ней аналогично предыдущему добавляется величина $a_{m-1} \cdot 2^{-1}$, так что в десятичном регистре получится сумма $a_{m-1} \cdot 2^{-1} + a_m \cdot 2^{-2}$. После m циклов в нем образуется величина $a_1 \cdot 2^{-1} + a_2 \cdot 2^{-2} + \ldots + a_m \cdot 2^{-m}$, т.е. исходное число, но в десятичном представлении, в коде «8, 4, 2, 1». Если его необходимо получить в коде с излишком 3, то в заключение операции производится добавление троек по всем тетрадам (через цепи коррекции).

Хотя смысл операций, выполняемых при преобразовании из десятичной системы в двоичную и при преобразовании из двоичной системы в десятичную, совершенно различен, внешне эти преобразования выглядят как обратные операции, чего в общем и следовало ожидать заранее.

При работе с целыми числами будем считать, что запятая и в двоичном, и в десятичном регистрах расположена справа от младшего разряда. Для перевода *целого числа из десятичной системы в двоичную* нужно исходное число в коде «8, 4, 2, 1» поместить в десятичный регистр, двоичный регистр предварительно погасить, а затем действовать точно так же, как при первом переводе дробного числа из двоичной системы в десятичную. Если исходное число было представлено кодом с излишком 3, то предварительно нужно через цепи коррекции произвести вычитание троек по всем тетрадам.

Наоборот, для перевода *целого числа из двоичной системы в десятичную* нужно исходное число поместить в двоичный регистр, десятичный регистр предварительно погасить, а затем действовать так же, как при переводе дробного числа из десятичной системы в двоичную; результат в десятичном регистре получится в коде с излишком 3, и для перехода к простой двоично-десятичной записи (к коду «8, 4, 2, 1») придется в конце операции произвести вычитание троек по всем тетрадам.

Читателю предлагается самостоятельно провести сопоставление тех операций, которые производятся при этом в рассматриваемом устройстве, с теми алгоритмами перевода целых чисел из одной системы счисления в другую, которые были приведены в 1.2.4. Желательно также повторить самостоятельно—применительно к операциям в рассматриваемом устройстве — те примеры, которые приводились в 1.2.4.

С некоторыми дополнениями устройство рис. 5-7 можно приспособить и для перевода чисел *с плавающей запятой* из одной системы счисления в другую. Та часть, которая изображена на рис. 5-7, может быть непосредственно использована для преобразования мантисс чисел с плавающей запятой. Однако нужно учесть, что с мантиссами придется производить еще некоторые дополнительные операции. Дело в том, что число α, записанное в нормальной форме, в двоичной системе имеет вид

$$\alpha = 2^{a_2} A_2,$$

где a_2 —двоичный порядок (целое число), A_2 —двоичная мантисса, $1/2 \le |A_2| < 1$, а в десятичной системе

$$\alpha = 10^{a_{10}} A_{10},$$

где a_{10} —десятичный порядок (целое число), A_{10} —десятичная мантисса, $1/10 \le |A_{10}| < 1$. Ясно, что в общем случае $A_2 \neq A_{10}$.

Для выполнения дополнительных операций над мантиссами нужно будет увеличить десятичный регистр на 2 дополнительные тетрады — одну слева от старшего разряда, другую —справа от младшего разряда; кроме того, желательно (но не обязательно) оборудовать десятичный регистр цепями сдвига влево и вправо на 4 двоичных разряда. Для выполнения операций над порядками нам потребуются, кроме того, два реверсивных счетчика — один десятичный и один двоичный.

Пусть задано десятичное число с плавающей запятой $\alpha = 10^{a_{10}} A_{10}$; как порядок ($a_{10}$), так и мантисса ($A_{10}$) заданы в десятичной системе. Преобразование числа в двоичную систему состоит в вычислении двоичного порядка a_2, который сам должен быть представлен в двоичной форме, вычислении новой мантиссы A_2 (удовлетворяющей условиям $2^{a_2} A_2 = 10^{a_{10}} A^{10}$, $\frac{1}{2} \le |A_2| < 1$) и преобразовании найденной величины A_2 в двоичную систему счисления.

Поместим a_{10} в десятичный счетчик, A_{10} — в десятичный регистр (имеющийся на рис. 5-7), а двоичный счетчик и двоичный регистр погасим.

Если $a_{10} > 0$, то будем дальше действовать следующим образом. Умножим величину A_{10} на 10, т. е. произведем в десятичном регистре сдвиг на 4 двоичных разряда влево (для этого мы, собственно, и ввели в десятичном регистре дополнительную тетраду слева от старшего разряда); одновременно вычтем единицу в десятичном счетчике. Затем произведем несколько делений пополам в десятичном регистре (каждое деление пополам — это сдвиг вправо на 1 двоичный разряд и коррекция). Деление пополам будем делать до тех пор, пока число в десятичном регистре не станет снова меньше единицы; для этого может потребоваться от одного до четырех делений пополам, так как результат умножения A_{10} на 10 заключен в интервале $1 \le 10\, A_{10} < 10$; при каждом делении пополам будем добавлять по единице в двоичный счетчик. Затем снова умножим на 10 число, имеющееся в десятичном регистре, уменьшив при этом на единицу число в десятичном счетчике, и снова произведем ряд делений пополам в десятичном регистре, добавляя каждый раз по единице к двоичному счетчику, и т.д.— до тех пор, пока в десятичном счетчике не останется нуль. В результате в двоичном счетчике получим величину двоичного порядка a_2, а в десятичном регистре — величину двоичной мантиссы A_2, выраженную пока в десятичной системе. Затем остается только в обычном порядке преобразовать A_2 в двоичную форму.

Если $a_{10} < 0$, то вместо умножений на 10 и делений пополам нужно будет выполнять деления на 10 (каждое из которых представляет собой сдвиг в десятичном регистре на 4 двоичных разряда вправо) и удвоения числа.

Аналогичные операции выполняются и при преобразовании чисел из двоичной системы в десятичную. Но при этом операции над порядками удобнее производить после преобразования двоичной мантиссы в десятичную форму, так как именно в десятичном регистре легко выполняются и умножения и деления на 10, и умножения и деления на 2.

Конечно, этот путь преобразования порядков слишком длинный. Количество циклов удвоения или деления пополам оказывается равным количеству единиц в величине двоичного порядка, да еще требуется столько тактов сдвига на 4 разряда, сколько единиц содержится в десятичном порядке (либо в 4 раза больше тактов сдвига на 1 разряд, если цепи сдвига на 4 разряда не предусмотрены). Можно, вероятно, без труда найти более короткие способы преобразования порядков, но большей частью это и не требуется.

Дело в том, что описанное устройство по идее должно быть связано с внешними устройствами машины.

Предположим, например, что на изображение абсолютной величины двоичного порядка отводится 5 двоичных разрядов; максимальный по абсолютной величине двоичный порядок равен при этом 31, а максимальный десятичный порядок — десяти (потому что $10^{10} \approx 2^{31}$). Считая, что сдвиг и коррекция занимают по одному такту, найдем, что для преобразования порядков требуется максимум 2 × 31 + 10 = 72 такта (если бы в десятичном регистре не было цепей сдвига на 4 двоичных разряда и вместо сдвига на 4 разряда нужно было бы выполнять 4 сдвига на 1 разряд, то количество тактов оказалось бы равным 2 × 31 +4 ×10 = 102). Кроме того, 2 *m* тактов требуется на преобразование мантисс; если, например, количество двоичных разрядов в мантиссе *m* равно 32, то 2*m* — 64, и общее количество тактов на преобразование числа равно 72 + 64 = 136. При тактовой частоте, скажем, 200 *кгц* (длительность такта 5 *мксек)* преобразование одного числа займет меньше 700 *мксек.* При использовании обычных внешних устройств это время меньше, чем, к примеру, время прохождения 1 ряда отверстий на перфоленте через фотоэлектрическое входное устройство, или меньше, чем время прохождения пустого сектора на барабане цифро-печатающего выходного устройства (синхропринта). Если учесть, что десятичный регистр все равно потребовался бы в схеме внешних устройств, то количество дополнительного оборудования, необходимого для преобразования чисел из одной системы счисления в другую с приемлемой скоростью, оказывается не весьма значительным.

Пользуясь аналогичными приемами, можно осуществить преобразование чисел не в специальном устройстве, а в виде особой операции арифметического устройства — наряду с умножением, делением и другими операциями. Если, например, арифметическое устройство построено в соответствии с рис. 4-2,*а* (стр. 144), а для выполнения деления в нем предусмотрены некоторые дополнительные цепи в соответствии со схемой рис. 5-1, *а* (стр. 206), то при реализации преобразования чисел регистр *A* удобно использовать в качестве двоичного регистра, а регистр *B* — в качестве десятичного регистра (ср. с рис. 5-7). Необходимые цепи сдвига в этих регистрах уже имеются. Разбив регистр *B* на тетрады, можно в соответствующих разрядах регистра *C* установить комбинации 0011 (тройки); тогда при небольшом усложнении схемы регистр *C* вместе с сумматором можно будет использовать в качестве цепей коррекции для регистра *B.* Длительность выполнения преобразования числа с фиксированной запятой получится при этом равной длительности умножения или деления, либо, возможно, несколько меньше (за счет того, что такт коррекции может быть короче, чем такт суммирования или сравнения-вычитания, так как при выполнении коррекции двоичные переносы распространяются в сумматоре только через 4 разряда). Некоторое неудобство по сравнению со специальным устройством состоит в том, что регистры *A* и *B* содержат одинаковое количество двоичных разрядов, из-за чего ухудшается точность преобразования.

5.2.3. Специальные операции.

Специальные операции, которые необходимы иногда в тех или иных специализированных цифровых вычислительных машинах, могут оказаться весьма разнообразными и порой неожиданными как по смыслу, так и по способам выполнения.

Например, в одной из машин, предназначенных для решения в основном логических, а не вычислительных задач, очень полезной оказалась операция «логическое умножение со сборкой». Операция эта состоит в том, что из исходного двоичного числа извлекаются цифры тех разрядов, в которых в другом из исходных чисел («извлекателе») содержатся единицы; выбранные цифры затем располагаются одна за другой в старших разрядах результата, а младшие разряды результата заполняются нулями. Например, если исходное число имеет вид

1011000110101101,

а извлекатель задан в виде

1001111011000100,

то результат извлечения необходимых разрядов получился бы

1——1000—10———1——,

а после сборки этих цифр окончательный результат имел бы вид

1100010100000000.

В арифметическом устройстве, построенном в соответствии с рис. 4-2, а на стр. 144 (дополненном, однако, цепями сдвига вправо в регистре *C),* эта операция выполнялась бы следующим образом. Исходное число помещается в регистр *B,* извлекатель — в регистр *C*; регистр *A* первоначально гасится (устанавливается в положение «0»).. Затем в регистрах *B* и *C* одновременно выполняются сдвиги вправо; каждый раз, когда в младшем разряде регистра *C* оказывается единица, одновременно выполняется также сдвиг вправо в регистре *A*, причем младшая цифра из *B* передвигается в старший разряд регистра *A.* После *m* тактов, где *m* — количество двоичных разрядов в исходных числах, в регистре *A* окажется, очевидно, результат «логического умножения со сборкой».

Операция эта первоначально предназначалась для формирования адреса оперативной памяти по определенным группам признаков, содержащихся в исходном числе, но при пробном программировании выяснилось, что она весьма удобна и в ряде других случаев.

Аналогичное положение создалось и в связи с операцией «двойное сравнение», которая была введена в список операций другой специализированной машины.

Смысл операции «двойное сравнение» состоит в следующем. В качестве исходных в арифметическое устройство посылаются 3 числа: *A, B* и *C*; выполняется проверка, удовлетворяются ли одновременно оба неравенства:

$$0 \leq A - B < |C|,$$

и в зависимости от этого вычисления направляются по одному или другому пути.

По первоначальному замыслу эта операция предназначалась для выяснения того, входит ли число *A* в заданный интервал $[B, |C|)$. Однако уже при пробном программировании она оказалась чрезвычайно удобной и полезной и в ряде других случаев. Если принять, что *C* — максимально возможное в машине число, то двойное сравнение превращается в простое алгебраическое сравнение чисел *A* и *B*; если *C* — минимально возможное не равное нулю число, то двойное сравнение дает положительный ответ только при точном равенстве $A = B$, а во всех остальных случаях — отрицательный ответ; если во все три регистра заслать одно и то же число *A* ($B = C = A$), то двойное сравнение $0 \leq A - A < |A|$ дает отрицательный ответ только при условии $A = 0$, а во всех других случаях — положительный ответ и т. д. Возможно, что двойное сравнение было бы полезно не только в специализированных, но и в вычислительных машинах общего назначения.

Разумеется, мы привели здесь только отдельные примеры специализированных операций. Вряд ли можно перечислить все имеющиеся здесь возможности.

5.2.4. Формирование списка операций. Использование табличных методов выполнения операций.

В начале настоящего раздела, в связи с рассмотрением операции деления, мы уже пытались найти ответ на вопрос о том, целесообразно ли включение некоторой конкретной операции (в частности, деления) в список операций арифметического устройства цифровой вычислительной машины.

Сейчас, познакомившись с рядом других, более экзотических, чем деление, операций, уместно поставить вопрос шире: как вообще формируется список операций арифметического устройства? На основе каких критериев та или иная конкретная операция вносится в этот список или исключается из него?

Прежде всего, конечно, список операций должен обладать свойством функциональной полноты.

Пусть имеется некая произвольная функция $f(x_1, x_2, ..., x_n)$, где независимые переменные $x_1, x_2, \ldots, x_n$ и сама функция *f* принимают значения из множества допустимых чисел, с которыми оперирует машина. В самом общем виде функция $f(x_1, x_2, \ldots, x_n)$ может быть задана таблицей, в левой части которой перечислены все возможные комбинации $x_1, x_2, \ldots, x_n$, а в правой части стоят соответствующие значения *f*. Если машина оперирует с *N* разными числами, то такая таблица содержит N^n строк, а количество различных функций от *n* переменных равно N^{N_n}.

Требование функциональной полноты списка операций состоит в том, чтобы любая функция, заданная указанным образом, могла быть вычислена с помощью конечного количества операций, принадлежащих к имеющемуся списку.

Условия, при которых достигается функциональная полнота, рассматриваются в работах по *k*-значной логике*). Можно, в частности, показать, что функциональной полнотой обладал бы список всего из двух арифметических операций: *сравнение,* позволяющее выбрать большее из двух заданных чисел, и *счет,* т. е. получение для любого заданного числа ближайшего большего (если заданное число есть максимально допустимое, то в результате выполнения операции «счет» должно получаться минимально допустимое число). Эти две операции можно даже заменить одной, состоящей в том, что из двух заданных чисел выбирается большее и дальше находится ближайшее большее от него (с той же оговоркой, что и для операции «счет»); если бы машина оперировала с *N* целыми неотрицательными числами 0, 1, 2, ..., $N - 1$, то эта операция записывалась бы в виде

$$\max(x_1, x_2) + 1 \pmod N$$

(«штрих Шеффера»).

Требование функциональной полноты обычно заведомо удовлетворяется (даже если при составлении списка операций этой стороне дела не уделялось особого внимания), и при формировании списка операций на первом плане находятся соображения другого рода. Чаще всего наиболее существенными из них считаются, во-первых, возможность использования традиционных и хорошо исследованных алгоритмов, для чего в списке операций должны быть такие операции, как сложение, вычитание, умножение и т. д., и, во-вторых, достижение высокой эффективности машины, т. е. возможность получить при не слишком больших затратах оборудования возможно большую скорость решения определенных задач (для специализированных машин) или определенных классов задач (для универсальных машин).

В одной из разработок специализированной машины, осуществлявшихся автором, для решения вопроса о списке операций был применен следующий прием.

Первоначально был вчерне составлен список операций, включавший в основном операции, обычные для универсальных машин. В соответствии с ним вчерне была составлена и схема арифметического устройства. При пробном программировании тех классов задач, для которых предназначалась машина, программисты и инженеры, изучавшие программы, выдвигали предложения о введении тех или иных специальных операций. Эти

*) См., например, работы Я б л о н с к о г о С. В., (сб. «Труды математического института им. Стеклова», т. LI.).

предложения принимались при условии, что специальная операция давала экономию в общем времени решения задачи не менее 10% и что необходимое усложнение арифметического устройства не превосходило (в процентах к общему количеству оборудования) достигаемого выигрыша по быстродействию. По этим критериям были проверены и операции из первоначального списка. Таким образом, были отвергнуты, например, такие операции, как логическое умножение и логическое сложение, и введены такие операции, как извлечение квадратного корня и «двойное сравнение» (вместо простых сравнений, описанных в 3.3).

Конечно, эти критерии являются только примером того, как можно подойти к составлению списка операций, но не предлагаются нами ни в качестве образца, ни тем более в качестве правила для решения этого вопроса.

В некоторых случаях могут быть иными и основные соображения, влияющие на формирование списка операций. Например, Н. Я. Матюхин указывает, что для вычислительной машины, работающей не в вычислительном центре, а в системе автоматического управления, когда быстродействие требуется умеренное, а надежность должна быть очень большая и когда не приходится рассчитывать на особенно высокую квалификацию обслуживающего персонала, наиболее существенным является упрощение логической структуры машины. С этой целью желательно по возможности сократить список операций, ограничив его, может быть, только сложением, вычитанием и сдвигом вправо на 1 разряд (сдвиг влево заменяется сложением числа самого с собой, сравнение — вычитанием с условным переходом по знаку разности). Умножение, деление и другие операции могли бы при этом выполняться по подпрограммам.

Для случая, когда количество разрядов невелико, а для хранения программы и констант в машине все равно имеется постоянная память, Н. Я. Матюхин предлагает использовать таблицы для ускорения операций. Имея, например, таблицу квадратов чисел, можно было бы умножение $x \cdot y$ выполнить по формуле

$$xy = \frac{1}{2}\left[(x+y)^2 - x^2 - y^2\right]$$

или

$$xy = \frac{1}{4}\left[(x+y)^2 - (x-y)^2\right].$$

Вычисление по первой из них включает одно сложение, три обращения к таблице, два вычитания, один сдвиг вправо, по второй — сложение, два вычитания, два обращения к таблице и два сдвига вправо. Возможно, что во многих случаях выполнение такой подпрограммы заняло бы не больше времени, чем выполнение умножения аппаратным путем, если бы таковое было реализовано без применения методов ускорения.

Вообще-то таблица квадратов всех чисел, с которыми оперирует машина, при более или менее реальной разрядности чисел получается слишком громоздкой. Скажем, если машина оперирует с 16-разрядными двоичными числами, то таблица должна была бы иметь объем $2^{16} \approx 65\,000$ чисел, при 20 разрядах — $2^{20} \approx 1\,000\,000$ чисел. Выход из положения мог бы состоять в том, чтобы аргумент таблицы задавался вдвое меньшим количеством разрядов, чем для соответствующего табличного значения функции. Умножение полноразрядных чисел при этом могло бы выполняться по формулам умножения с двойной точностью

$$xy = \left(x_1 + 2^{\frac{-m}{2}} x_2\right)\left(y_1 + 2^{\frac{-m}{2}} y_2\right) \approx x_1 y_1 + 2^{\frac{-m}{2}}\left(x_2 y_1 + x_1 y_2\right),$$

где x_1, y_1 — старшие части соответственно x и y, x_2, y_2 — младшие части. При количестве разрядов $m = 16$ объем таблицы при этом составлял бы всего $2^{\frac{m}{2}} = 256$ чисел, при $m = 20$ — 1024 числа.

Вместо таблицы квадратов в ряде случаев удобнее пользоваться таблицами некоторых других функций, в частности таблицей

$$f(x) = \cos\frac{\pi}{2}x.$$

В любом случае, однако, табличные методы применимы тогда, когда количество разрядов в машине невелико и когда в составе машины так или иначе должна иметься постоянная память.

* *
*

Эта книга была задумана в свое время как попытка дать моментальную фотографию современного состояния теории и практики построения арифметических устройств. Однако развитие в этой области все еще идет достаточно быстро, а «экспозиция» по понятным причинам не может быть слишком короткой.

Иногда кажется, что поток новых идей, относящихся к методам выполнения арифметических операций и построению арифметических устройств цифровых вычислительных машин, постепенно иссякает и что уже через несколько лет арифметика цифровых машин превратится в классическую дисциплину с установившимися канонами и структурой, в которой существенно новые результаты будут появляться чрезвычайно редко.

Во всяком случае, в последние годы наиболее существенное продвижение наблюдалось только в методах выполнения деления, схемах осуществления элементарных операций (в связи с появлением новых элементов схем и, в частности, связи с успехами микроэлектроники), а также отчасти в способах изображения чисел. Некоторые из

полученных в этих направлениях результатов еще не нашли достаточного отражения в этой книге.

Скорее всего, однако, возможности прогресса и в других направлениях далеко не исчерпаны, и появление некоторых новых идей общего характера могло бы привести к новой волне быстрого продвижения по всему фронту.

Библиография

Авалиани Ю. Е., АлексеевЮ. Н., Глухов Ю. Н., Дорохова Н. А., Танетов Г. И., Арифметическое устройство специализированной машины. В сб. «Цифровая техн.. и вычислительные устройства», № 3, АН СССР.

Акушский И. Я., Юдицкий Д. И., Машинная арифметика в остаточных классах, «Советское радио», 1968.

Альперович Л. З., Шигин А. Г., О контроле по модулю арифметических операций, «Изв. АН СССР, Техн. кибернетика», 1964, № 3, стр. 69—71.

Анисимов Б. В., Четвериков В. Н., Основы теории и проектирования цифровых вычислительных машин, Машгиз, 1962.

Афанасьев В. А. и др., Специализированное арифметическое устройство, «Автоматика и приборостр. Информ. научно-техн. сб.», 1965, № 2 (22), стр. 17—19.

Балашов Е. П., Смирнов В. Б., Логический элемент на двухотверстном трансфлюксоре, Авт. св. СССР, кл. 21 a^1, 36/18 (G 06f), № 187832.

Берулава Р. Г., Быстродействующее суммирующее устройство с независимым переносом на динамических элементах, «Тр. ВЦ АН ГрузССР», 1965, 6, № 1, стр. 59—72.

Бородин Н. Н., Арифметическое устройство, Авт. св. СССР, кл. 42m, 14 (G06d), № 122948.

Брик В. А., Борисов Ю. М., Танетов Г. И., Пирамида сумматоров для арифметического устройства ЦВМ. В сб. «Элементы и устройства управл. машин», 1966, 159—165.

Бруевич Н. Г., О системе управления арифметического устройства электронной цифровой математической машины, «Изв. АН СССР, Отд. техн. наук Энерг. и автоматика», 1961, № 5, 143—153.

Брусенцов Н. П., Маслов С. П. и др., Малая цифровая вычислительная машина «Сетунь», Изд-во МГУ, 1965.

Букреев И. Н. и др., Устройство различения чисел по модулю в быстродействующих электронных сумматорах, Тр. Ин-та кибернетики АН ГрузССР, 1963, 1, стр. 105—110.

Буоджюс И. Р., Григас А. Г.. Устройство умножения, Авт. св. СССР, кл. 42m, 14, № 146604.

Бутаев Г. М., Ромашкин В. С, Цифровые устройства извлечения квадратного корня, «Автоматика и приборостроение. Информ научно-технич. сб.», 1964, № 3 (19), стр. 26—28.

Вавилов Е. Н., Портной Г. П., Синтез схем электронных цифровых машин, «Сов. Радио», 1963.

Венкстерн А. С, Турчин Д. Г., Контроль по модулю арифметических операций ЭЦВМ. В сб. «Цифр. выч. техн. и программир.», вып. 1, «Сов. радио», 1966, стр. 22—27.

Видин Б. В., Орфеев Ю. В., Об одной модификации алгоритма деления ЭВМ, «Приборостроение», 1964, 7, № 4, стр. 90—93.

Вишневский А. П. и др., Накопительный десятичный сумматор, Авт. св. СССР, кл. 42m, 14/03 (G06f), № 170212.

Вольский В. С, Зданкевич В. Л., Компаратор двоичных чисел. В сб. «Автоматика и выч. техн.», № 12, Рига, «Зинатне», 1966, стр. 247—251.

Галкин Ф. Я., Об арифметических возможностях кода «2 из 5», «Изв. высш. учебн. заведений. Приборостроение», 1965, 8, № 6, стр. 73—76.

Генкин В. Л., Смирнов В. Б., Сумматор по модулю 2 на флюксоре, Авт. св. СССР, кл. 42m, 14/02 (G06f), № 178165.

Гиттис Э. И., Бергельсон М. Н., Арифметическое устройство для выполнения операций сложения и вычитания в самоконтролирующемся циклическом коде, Авт. св. СССР, кл. 42m, 14 (Q06f), № 183484.

Глушков В. М., Синтез цифровых автоматов, Физматгиз, 1962.

Голубев Л. А., Способ выполнения операции деления Авт. св. СССР, кл. 42m, 14, № 145069.

Голышев Л. К., Арифметическое устройство на динамических элементах с задержками. В сб. «Вычисл. матем. и техн.», АН, УССР, 1962, стр. 68—77.

Григорьев Ю. П., Морозов А. А., Устройство для определения старшего разряда разности двух двоичных чисел, Авт. св. СССР, кл. 42m, 14 (G06d), № 134071.

Дашевский Л. Н., Погребинский С. Б., Шкабора Е. Д., Вычислительная машина «Киев», Киев, «Техника», 1964.

Дейнеко В. Н., Сумматор-вычитатель параллельного действия на трехтактных феррит-диодных ячейках с магнитным преобразователем кода переноса (заема). В сб. «Магнитные элементы автоматики, телемеханики, измерит. и выч. техники», Киев, 1964, стр. 474—480.

Доброчаева Э. А., Программный контроль арифметического узла с помощью диагностических таблиц и контрольной операции. В сб. «Диагностика неисправностей вычислительных машин», «Наука», 1965, стр. 53—59.

Долидзе Т.В., Гогоберидзе М. Г., Джибути М.С, Однотактный импульсно-потенциальный параллельный сумматор, «Труды ТНИИСА», 1966, № 7, стр. 208—214.

Долидзе Т. В. и др., Параллельный сумматор накапливающего типа, Авт. св. СССР, кл. 42m, 14/03 (G06d), № 169885.

Долидзе Т.В., Устройство ускоренного двоичного умножения, «Труды ТНИИСА», 1966, № 7, стр. 162—172.

Долкарт В. М. и др., Арифметическое устройство управляющей вычислительной машины ВНИИЭМ-1, «Электротехника», 1966, №6, стр. 47—51.

Дроздов Е. А., Прохоров В. И., Пятибратов А. П., Основы вычислительной техники, Воениздат, 1961.

Евреинов Э. В., Косарев Ю. Г., Однородные универсальные вычислительные системы высокой производительности, Новосибирск, «Наука», СО, 1966.

Жилко Э. И., МитропольскийЮ. И., Некоторые вопросы построения арифметических устройств управляющих вычислительных машин. В сб. «Автоматизация технол. процессов в черной металлургии», вып. 3, Металлургиздат, 1963, стр. 2—81.

Жук В. И., Цифровое арифметическое устройство, Авт. св. СССР, кл. 42m, 14/03 (G06d), № 170213.

Замрий В. Н., Об одном способе ускоренной обработки цифр в устройствах двоично-десятичного преобразования чисел, «Тр. 6-й Научно-техн. конференции по ядерн. радиоэлектрон. Т. 3. 4.2», Атом-издат, 1965, стр. 162—170.

Зимин В. А., Электронные вычислительные машины. Основы теории расчета и применения, Машгиз, 1962.

Зонис В. С, Танетов Г. И., Комбинационный сумматор для двоичной и десятичной систем счисления, «Приборы и системы

упр.», 1967, № 16, стр. 17—19.

Зуев А. Ф. и др., Арифметическое устройство на однотипных пороговых элементах. В сб. «Полупроводниковые элементы в вычисл. техн.», ОНТИПрибор, 1965, стр. 66—76.

Иванов Г. П., Шамашкин А. М., Никитин В. Н., Способ суммирования в цифровых машинах, Авт. св. СССР, кл. 42m, 14 (G06f), № 188753.

Иловайский И. В., Фет Я. И., Об одном способе организации быстродействующего арифметического устройства, Изв. СО АН СССР, 1966, № 6, Сер. техн. н., вып. 2, стр. 60—68.

Канторович Л. В. и др., Сумматор для одновременного сложения нескольких двоичных слагаемых, Авт. св. СССР, кл. 42m, 14 (G06f), № 188151.

Карташев В. И., Синтез управления арифметического устройства вычислительной машины. В сб. «Кибернет. и вычисл. техн.», Киев, «Наук. думка», 1964, стр. 18—25.

Карцев М. А., Арифметические устройства электронных цифровых машин, Физматгиз, 1958.

Карцев М. А., Гливенко Е. В., Дробные основания системы счисления и их использование для ускорения операций в цифровых машинах. В сб. «Colloquium on the Foundations of Mathematics, Mathematical Machines and Their Applications», Будапешт, 1965, стр. 215—226.

Карцев М. А., РогачевЮ. В., Потенциальные полупроводниковые элементы и построение из них логических схем. В сб. «Цифровая техника и вычислительные устройства». АН СССР, 1962, № 2.

Кац А. М., Пурец Н. Я., Способ прекращения операции деления, Авт. св. СССР, кл. 42m, 14/03 (G06f), № 174436.

Кладов Г. К. и др., Контроль арифметических операций в коде вычетов, «Кибернетика», 1965, № 4, стр. 43—44.

Кобринский Н. Е., Трахтенброт Б. А., Введение в теорию конечных автоматов, Физматгиз, 1962.

Ковачич Ю. В. и др., Цифровое интегрирующее и логическое устройство, Авт. св. СССР, кл. 42m, 14/04 (G06f), № 181881.

Комаров А. Н., Логика построения арифметических устройств. (Многоступенчатый сумматор с полным параллельным переносом.) «Тр. Всес. заочн. энерг. ин-та», вып. 29, 1965, 16—35.

Костелянский В. М., Устройство для формирования и хранения вычетов чисел по модулю 3, Авт. св. СССР, кл. 42m, 14/03 (G06f), № 163806.

Кузин Е. С, Арифметическое устройство цифровой вычислительной машины последовательного действия. В сб. «Вычисл. техника», вып. 3, Оборонгиз, 1963, стр. 206—217.

Лебедев И. В. и др., Метод извлечения квадратного корня в цифровой вычислительной машине. В сб. «Вопр. вычисл. математики и вычисл. техн.», Машгиз, 1963, стр. 175—178.

Лебедев О. В., Кольцевая параллельная пересчетная схема, Авт. св. СССР, кл. 42m, 14/03, 21 a^1, 36/18 (G06f, H03k), № 159334.

Лебедев С. А., Мельников В. А., Общее описание БЭСМ и методика выполнения операций, Физматгиз, 1959.

Майоров Ф. В., Электронные цифровые интегрирующие машины, Машгиз, 1962.

Мамчиц В. А., ВолковицкийК. Е., Параллельный феррит-диодный вычитатель, Авт. св. СССР, кл. 42m, 14/03 (G06d), № 172560.

Манукян Ю. С. и др., Реверсивный счетчик в коде Грея, Авт. св. СССР, кл. 42m, 14/03 (G06d), № 167365.

Манукян Ю. С. и др., Устройство для сравнения чисел, Авт. св. СССР, кл. 42m, 14/02 (G06f), № 163426.

Махмудов Ю. А., Дагкесаманская Р. Ю., Схемы последовательного двоичного умножения и деления на ферритовых элементах, «Тр. ВЦ АН АзербССР», 1962, 1, стр. 83—95.

Меерович Л.А., Конев К. В., Сулин Л. И., Логическая схема НЕ-ИЛИ на базе элемента магнитного бездиодного сдвигающего регистра, Авт. св. СССР, кл. 42m, 14 (G06f), № 188141.

Михелев В. М., Потенциальный сумматор с временем суммирования 0,5 *мксек.* В сб. «Вопр. вычисл. матем. и вычисл. техн.», Машгиз, 1963, стр. 143—149.

Михелев В. М., Способ выполнения операции умножения двоичных чисел в ЦВМ, Авт. св. СССР, кл. 42m, 14/03, № 157839.

Мнацаканов Р. Б. и др., Способ сложения двоичных чисел, Авт. св. СССР, кл. 42m, 14/03 (G06f), № 176724.

Мороз І. Г., Про один метод прискоренного ділення. Веб. «Обчислювальна математика і техніка», Киев, 1963, стр. 126 —129.

Мямлин А. Н. и др., Арифметическое устройство универсальной электронной вычислительной машины с контролем и совмещением операций. В сб. «Вопр. вычисл. матем. и вычисл. техн.», Машгиз, 1963, стр. 136—142.

Папернов А. А., Логические основы цифровых машин и программирования, «Наука», 1968.

Папернов А. А., Петрова Г. Н., К вопросу об округлении произведения при умножении со старших разрядов множителя. В сб. «Цифр. вычисл. техн. и программир.», вып. 1, «Сов. радио», 1966, стр.. 5—11.

Параметроны, ИЛ, 1962.

Пискунов С. В., Последовательный десятичный сумматор, Авт. св. СССР, кл. 42m, 14/03 (G06f), № 169891.

Питкевич А. А., Процкевич А. И., Устройство для сравнения чисел, Авт. св. СССР, кл. 42m, 14 (G06f), № 183483.

Поснов Н. Н., Метод десятичного умножения с последовательной заготовкой кратных множимого, «Весці АН БССР. Сер. Фіз.-тэхн. н. Изв. АН БССР, Сер. физ.-техн. н.», 1964, № 2, стр. 12—15.

Прессман А. И., Расчет и проектирование схем на полупроводниковых приборах для цифровых вычислительных машин, ИЛ, 1963.

Рабинович З. Л. и др., Арифметическое устройство с повышенной скоростью выполнения операций. В сб. «Вопр. вычисл. матем. и вычисл. техн.», Машгиз, 1963, стр. 165—170.

Рабинович З. Л., Элементарные операции в вычислительных машинах, Киев, «Техніка», 1966.

Рабинович З. Л., Михновский С. Д. и др., Анализ методов миоготактного умножения и деления в ЦВМ, «Автоматика и приборостроение», 1962, № 2, стр. 38—43.

Ричардс Р. К., Элементы и схемы цифровых вычислительных машин, ИЛ, 1961.

Ричардс Р. К., Арифметические операции в цифровых вычислительных машинах, ИЛ, 1957.

Самохвалов Е. А., Арифметическое устройство ЦВМ на параметронах, «Автоматика и приборостроение. Информ. научно-техн. сб.», 1965, № 4 (24), стр. 20—22.

Скрипкин А. Я., Поздняков В. А., Устройство для сравнения двоичных чисел, Авт. св. СССР, кл. 42m, 14/03 (G06f), № 174442.

Смирнов А. Ф., Быстродействующая схема умножения на полупроводниковых элементах. В сб. «Вычисл. и информ. техника», 1962, стр. 82—90.

Степановский Д. И., Двоичный счетчик, Авт. св. СССР, кл. 42m, 14,21 a^1, 37/48 (G06f), № 190061.

Сумароков Л. Н. и др., Параллельный сумматор, Авт. св. СССР, кл. 42 m, 14/03 (G06f), № 169886.

Терешков Д. С. и др., Десятичный счетчик с переменным модулем пересчета, Авт. св. СССР, кл. 42m, 14/03 (G06f) № 169884.

ТорговЮ. И., Арифметическое устройство на динамических элементах, ВЦ АН СССР, 1963, стр. 86.

Фридман Г. Ф., Логические схемы *n*-входных полусумматоров и их применение для преобразования рефлексного двоичного кода в прямой, «Изв. АН СССР, Техн. кибернетика», 1963, № 5, стр 43—53.

Хетагуров Я. А., Арифметические устройства вычислительных машин дискретного действия, Госатомиздат, 1961.

Хетагуров Я. А., О проектировании арифметических устройств с учетом надежности работы. В сб. «Вычисл. техника», вып. 3, Госатомиздат, 1962, стр. 52—60.

Хетагуров Я. А. и др., Матричное устройство перемножения. В сб. «Вопр. вычисл. матем. и вычисл. техн.», Машгиз, 1963, стр. 157— 184.

Храпченко В. М., Об оценке погрешности двоичного умножения. В сб. «Проблемы кибернетики», вып. 10, 1963, стр. 165—177.

Храпченко В. М., Об одном способе преобразования многорядного кода в однорядный, ДАН СССР, 1963, 148, № 2, стр. 296, 299.

Чистяков Б. В. и др., Реверсивный двоичный феррит-транзисторный счетчик, Авт. св. СССР, кл. 42m, 14/03 (G06f), № 167372.

Чугаев Ю. Г., Плиско В. А., Электронные цифровые вычислительные машины, Воениздат, 1962.

Шапиро Г. С, ЗлотникД. Л., К математической теории кодов с исправлением ошибок, «Кибернетический сборник», ИЛ, 1962, № 5.

Шауман А. М., Класс сумматоров с повышенной надежностью. В сб. «Вычисл. техн. и вопр. программир.», вып. 4, Ленинград. ун-т, 1965, стр. 104—109.

Шварц С., Полупроводниковые схемы, ИЛ, 1962.

Шеннон К., Работы по теории информации и кибернетике, ИЛ, 1963.

Щербина П. И., Асимметрические коды, исправляющие многократные ошибки при арифметических операциях. В сб. «Вычисл. техника», Минск, «Наука и техника», 1965, стр. 20—30.

Юнчик А. М., Устройство для сравнения модулей двух двоично-кодированных чисел, «Автоматика и приборостр. Информ. научно-техн. сб.», 1965, № 2 (22), стр. 22—23.

Adams L. R., Ring circuit, пат. США, кл. 340-174, № 3160862.

Aleksander I., Array networks for a parallel adder and its control, «IEEE Trans. Electron. Comput.», 1967, 16, N 2, стр. 226— 229.

Amano R., Logic circuits using Esaki diodes, «Electronics and Communications in Japan», 1964, 47, стр. 122—132.

Ambrosino F., Improvements in or relating to adders, англ. пат. кл. G 4A (G06f), № 1017314.

Amdahl G. M., Structure of System/360—3. Processing Unit Design Considerations, «IBM Systems J.», 1964, 3, N 2, стр. 144—164.

Amdahl L. D. et al., Simultaneous digital multiply-add, multiply-subtract circuit., пат. США, кл. 235-164, № 3202805.

Amdahl L. D. et al., Stored logic computer., пат. США, кл. 340-172. 5, № 3246303.

Amodei J. J., High-speed adders and comparators using transistors and tunnel diodes, «IEEE Trans. Electron. Comput.», 1964,13, N 5, стр. 563—575.

Anderson D. W., SparacioF. J., Tomasulo R. M., The IBM System 360 Model 91. Machine philosophy and instruction handling, «IBM J. Rec. and Developm.», 1967, 11, N 1, стр. 8—24.

Anderson S. F. et al., The IBM System/360 Model 91. Floatingpoint execution unit, «IBM J. Res. and Developm.», 1967, 11, N 1, стр. 34—53.

Andrews C. A., Core adder, пат. США, 235-175, № 3185826.

Arango H., Santos J., A fast carry-propagation circuit for base 3 signed non redundant arithmetic, «IEEE Trans. Electron. Comput.», 1965, 15, N 2, стр. 254—255.

Armgarth D., Schaltungen zur Realisierung von logischen Funktionen in DTL und RTL, «Nachrichtentechnik», 1967, 17, N 6, стр. 240—246.

Arya M.C, Binary magnitude comparator, пат. США, кл. 235-177, № 3091392.

Ashenhurst R. L., The Maniac 111 arithmetic system, «AFIPS Proc. Spring Joint Comput. Conf., San Francisco, Calif. 1962, vol. 21», Palo Alto, Calif., 1962, стр. 195—202.

Auguste J., Deloffre J., Dispositif calculateur associé á une additionneuse, фр. пат. кл. G06f, № 1269264.

Avizienis A., Kimble D. M., A general building block for digital arithmetic, Proc. Nat. Sympos. Impact. Batch Fabric. Future Comput., 1965, стр. 173—180.

Baer R. M., Redlich M. G., Multiple — Precision Arithmetic and Exact Calculation of 3-j, 6-j and 9-j Symbols, «Comm. of the ACM», 1964, 7, N 11, стр. 657—659.

Bartee T. C, Digital computer fundamentals, 2nd ed. New-York, McGraw-Hill Book Co., 1966.

Bartee T. C, Chapman D. J., High-speed accumulator for real-time computer, «IEEE Internat. Convent. Rec», 1965, 13, N 3, стр. 242—247.

Bartee T. C, Chapman D. J., Design of an accumulator for a general purpose computer, «IEEE Trans. Electron. Comput.», 1965, 14, N 4, стр. 570—574.

Bartee T. C, Lebow I.L, Reed I. S., Theory and design of digital machines, London, McGraw-Hill, 1962.

Batcher K. E., On number of stable states in NOR network, «IEEE Trans, on Electron. Comput.», 1965, vol EC-14, N 6, стр. 931 — 932.

Bental L. J., Floating point arithmetic unit, «Electronic Eng.», 1962, 34, N 409, стр. 144—147.

Bernstein A. J., Kim W. H., Single- and double adjacent, error-correcting codes for arithmetic units, «IEEE Trans. Inform. Theory», 1963, 9, N 3, стр. 209—210.

Blaauw G. A., Structure of System 360—5, Multisystem organisation, «IBM Systems J.», 1964, 3, N 2, стр. 181—185.

Blaauw A., Brooks F. J., Jr., Structure of System 360—1. Outline of logical structure, «IBM Systems J.», 1964, 3, N 2, стр. 119—135.

Blum R. J., Binary code translating device, пат. США, кл. 340— 347, № 3138794.

Bodo R. et al., Schaltungsanordnung zum serienmäβigen Vergle-ich zweier Informationswcjte, пат. ФРГ, кл. 42m, 14 (G06f), № 1179028.

Booher R. K., Division apparatus, пат. США, кл. 235-167, № 3222505.

Booher R. K., Multiplication apparatus, пат. США, кл. 235-167, № 3237000.

Booher R. K., Carry assimilating system, пат. США, кл. 235-175, № 3249747.

Borne J., Ribes M., Perfectionnements aux machines arithmetiques calculant des racines carrés, фр. пат., кл. G06f, № 1387911.

Brčič I., «Ideally fast» decimal counters with bistables, «IEEE Trans. Electron. Comput.», 1965, 14, № 5, стр. 733—737.

Brett M., Schiff A., Overflow indicator пат. США. кл. 235-153, № 3016193.

Broce T. C, Universal radix adder., пат. США, кл. 235-169, № 3159740.

Brown N., Improvements relating to paraller digital adders, англ. пат., кл. G4 A (G06f), № 1040241.

Brown T.G., Jr., Self checking digital computer system, пат. США, кл. 340-146, 1, № 3098994.

Buelow F.K., Hilsenrath M., Full adder, пат. США, кл. 235-172, № 3248529.

Carol1 W. N. et al., Transfer circuit for controlling data transfer from adder to accumulator, пат. США, кл. 307-885, № 3152262.

Carrol W. N., Pulse train sensing circuitry, пат. США, кл. 235-164, № 3157780.
Carrol W. N., Tilton C.J., Asynchronous division apparatus, пат. США, кл. 235—164, № 3064896.
Chao S. C., High speed encoding with resistor-transistor logic circuits, «Electronics», 1962, 35, N 6, стр. 48—51.
Chiapuzio A., Jr., Digital comparator, пат. США, кл. 235-177, № 3137789.
Cifariello M., Ferrari D., Un circuito per la determi-nazione rapida dell'ordine di grandezza di un numero binario «Alta frequenza», 1967, 36, N 6, стр. 565—568.
Clark G. A., LangC. A., A new counting method for performing digital arithmetic, «Electron. Eng», 1963, 35, N 428, стр. 670—675, 703, 710.
Coates C L., Lewis P. M., II. Bistable device employing threshold gate circuits, пат. США, кл. 307-88. 5, № 3275849.
Cochran L. L. et al., Fast multiply system, пат. США, кл. 235-164, № 3192366.
Cochrane H. W., Binary matrix multiplier utilizing coincident inputs and sequential readout, пат. США, кл. 235-164, №3104317.
Cochrane H. W., Core matrix coded decimal parallel adder utilizing propagated carries, пат. США, кл. 235-175, № 3166669.
Computer arithmetic— 1 (Sympos. Summary), «Proc. IFIP Congr., New York City, 1965, vol. 2», Washington, D. C, Spartan Books, 1966, стр. 440—446.
Connet J., Improvements relating to digital computers, англ. пат., кл. 106 (1), № 909441.
Conroy E. D., Data timed mutiplication system, пат. США, кл. 235-164, № 3242304.
Cooke-Yarborough E. H., Serial adder for bits occuring in reverse time sequence, «Inst. Elec. Ingrs-EIectronic Letters», 1967, 3, N 2, стр. 77—78.
Coopper R. A., Parallel adder, пат. США, кл. 235-175, № 3031140.
Cromleigh R. G., Serial binary adder, пат. США, кл. 235-176, № 3264458.
Croy J. E., Electronic multiplier for a variable field length computer, пат. США, кл. 235-160, № 3161764.
Dadda L, Some schemes for parallel multipliers, «Alta frequen-za», 1965, 34, N 5, стр. 349—356.
Daly W. G., KruyJ. F., A high speed arithmetic unit using tunnel diodes, «IEEE Trans. Electron. Comput.», 1963, 12, N 5, стр. 503—511.
Davis C. M., De Veer J. A., Divider utilising multiples of a divisor, пат. США, кл. 235-156, № 3234366.
Davis R. A., A checking arithmetic unit, «AFIPS Conf. Proc. vol. 27, Part. 1», Washington D. C, Spartan Books, 1965, стр. 705—713.
Deerfield A. J., Fast multiply apparatus, пат. США, кл. 235-164, № 3159739.
De Regt M. P., Negative radix arithmetic. Part 1. Introduction to negative radix number systems, «Comput. Design», 1967, 6, N 5, стр. 52—55, 57, 60, 62, 63.
Dickinson A. H., Electronic multipliers, пат. США, кл. 235-160, № 3015442.
Digital integrated micro circuits, their characteristic, prices and sources, «Electronic Products», 1965, 7, N 8, стр. 87—103.
Dunwell S. W. et al., Electronic calculator, пат. США, кл. 340-172. 5, № 3202969.
Earnshaw J. B., Fenwick P M., Design for a parallel binary adder, «Electron. Eng», 1966, 38, N 466, стр. 794—796, 831, 838.
Electronic multiplier hints use of modular arithmetic in computers «Electronic Desing», 1963, v 11, N 19, стр. 12.
Elmore W. B., Minkow M., Arithmetic device, пат. США, кл. 235-92, № 2995298.
Emmons D. L., High speed comparator, пат. США, кл. 340-146. 2, № 3237159.
Erickson G. J., Segmented arithmetic device, пат. США, кл. 235-164, № 3234370.
Flores J., The logic of computer arithmetic, London, Prentice-Hall, Int., 1963.
Ferrari D., A. division method using a parallel multiplier, «IEEE Trans. Electron. Comput.», 1967, 16, N 2, стр. 224—226.
Ferris A. L, Full adder, пат. США, кл. 235-176, № 2998918.
Fialkowski K., Pseudosystematyczne zapisy dwójkowe, «Arch, electrotechn.», 1965, 14, N 3, стр. 617—626.
Fischler M. A., Poe E. A., Threshold realization of arithmetic circuits, «IRE Tran. Electronic Comput.», 1962, 11, N2, стр. 287—288.
Flynn M. J., Low P. R., The IBM System 360 Model 91. Some remarks on system development, «IBM J. Res. and Developm», 1967, 11, N 1, стр. 2—7.
Goldberg J., Antiparallel control logic, «IEEE Trans. Electron. Comput.», 1965,14, N 3, стр. 383—393.
Gottschalk H., Electronische Bausteinsysteme der Digital-technik, Berlin, Technik, 1965, стр. 68.
Gregory R. T., On the design of the arithmetic unit of a fixed-word-length computer from the standpoint of computational accuracy, «IEEE Trans. Electron. Comput.», 1966, 15 N 2, стр. 255—257.
Grondbeginselen van het elekronisch tellen en de daarvoor benodig-de schakelingen. Vooral voor gecompliceerde apparaten verdienen synchrone tellers de voorkeur, «Electronica», 1967, 20, N 490, стр. 438—443.
Hamming R. W., Mamme1 W. L., A note on the location of the binary point in a computing machine, «IEEE Trans. Electron. Comput.», 1965, 14, N 2, стр. 260—261.
Händler W., Rechenwerk einer digitalen Rechenmaschine, пат. ФРГ, кл. 42m, 24 (G06f), № 1157009.
HarndenG. W., Estimating number of integrated circuits for systems, «Electronic Products», 1964, 7, N 7, стр. 30, 69.
Hayden L. E., Tunnel diode majority logic serial binary adder/ substractor, пат. США, кл. 235-172, № 3260841.
Heijn H. J., Binary parallel adder utilising sequential and simultaneous carry generation, пат. США, кл. 235-175, № 3105897.
Helbig W., Woldrich R, J., High speed binary adder-substractor with carry ripple, пат. США, кл. 235-175, № 3198939.
Hendrickson H. C., Fast high-accuracy binary parallel addition, «IRE Trans, on Electron. Comput.», 1960, vol EC-9, N4, стр. 465—469.
Henle R. A., Hill L. L., Integrated computer circuits past, present and future, «IEEE Proc», 1966, 54, N 12, стр. 1849—1860.
Hertz T. M., YoungF. H., Binary square root mechanisation, пат. США, кл. 235-158, № 3049296.
Heywood J. E., Arithmetic circuitry, пат. США, кл. 235-170, № 3112396.
Hoberg G. G. et al., Electronic adder using two decode counters alternately, пат. США, кл. 235-176, № 3161765.
Homan M. E., 4-Megacycle 24-bit checked binary adder, «AIEE Trans.», «Communication and Electronic», 1961, 80, part 1, N 56, стр. 443—450.
Hotz G., Zweistufiges Rechenwerk, пат. ФРГ, кл. 42m, 14 (G06f), № 1184125.
Howe W. H., High speed logic circuit considerations, «AFIPS Conf. Proc, 27, Part 1», Washington, D. C, Spartan Books, 1965, p. 05—510.
Hurst S. L., An asynchronous binary-coded decimal circuit with gated feedback, «Electron. Eng», 1967, 39, N 467, стр. 25—28, 63, 70.
Jackson R. C. et al., A built-in table lookup arithmetic unit., «Proc. West Joint Computer Conf.», 17, S. 1, 1960, стр. 239—250.
Jeeves T. A., NOR element parallel dynamic adder-subtractor, пат. США, кл. 235-175, № 3125076.
Johnson E. C, Lu Chi Ho., Digital multiplier, пат. США, кл. 235-165, № 3006550.
Karp J., Use four-phase MOS IC logic when power and chip area at a premium, «Electron. Design», 1967, 15, N 7, стр. 62—66.

Katell E., A bounded carry inspection adder for fast parallel arithmetic, «AFIPS Conf. Proc», 27, part 1 Washington, D. C, Spartan Books, 1965, стр. 689—694.
Keir R. A., A division system, пат. США, кл. 235-164, № 3182180.
Kilburn T. et al., Improvements in or relating to multiplying arrangements for digital computing and like purposes, англ. пат., кл. G4A (G06f), № 976620.
King L. E., Variable mode arithmetic circuits with carry select, пат. США, кл. 235-169, № 3260840.
Kintner P. M., A simple method of designing NOR logic, «Control Engineering», 1963, 10, N 2, стр. 77—79.
Klinger F., Arithmetique pour calculateurs electroniques, les operations dans les systemes binaire, ternaire, octal, etc. dans les divers codes, le tout suivi d'exercices avec leurs solutions, Paris, Chiron, 1963, стр. 176.
Kregness G. R., Scale factor device for normalizing a binary number, пат. США, кл. 235-159, № 3234368.
Kruy J.R, Duben F. T., Integrated CONDITIONED OR and INHIBITED OR logic circuits, IEEE — J. Solid State Circuits, 1966, 1, N 2, стр. 81—85.
Kuntzmann J., Naslin P., Algebre de Boole et machines logiques, Sous la direct. Paris, Dunod, 1967, XVI, стр. 313.
Kuroyanagi N., Improvements of high-speed shift register, «Electronics and Communications in Japan», 1963, 46, N 10, стр. 1—12.
Kuroyanagi N., High speed arithmetic system for multiplication and floating point system calculation, «Rev. Electr. Commun. Lab.», 1962, 10, N 3—4, стр. 207—229.
Kuroyanagi N., Forty nonoseconds high speed shifter, «Tokyo. Electr. Communication Laboratory-Rev.», 1966, 14, N 5—6, стр. 381—399.
Kuttner P., Full adder using thin magnetic films, пат. США, кл. 235-176, № 3234372.
La Manna R. . J., Binary integer divider, пат. США, кл. 235-165, № 3018047.
Lee E. S., III, Binary full adder and «or» circuit, пат. США, кл. 307-88.5, №3215857.
Lehman M., Lee J., Serial arithmetic techniques, «AFIPS Conf. Proc, vol 27, part 1», Washington, D. C, Spartan Books, 1965, стр. 712—725.
Levien R. E., Determining best ordering of variables in cascade switching circuits, «Switching Circuits Theory and Logical Design»,4th Annual Symposium — Special Publ. S—156, oct. 1963, стр. 83—104.
Ling H., High speed binary paraller adder, «IEEE Trans. Electron. Comput.», 1966, 15, N 5, стр. 799—802.
Link H. Jr., Binary divider, пат. США, кл. 235-164, № 3229079.
Lissos D., Copper white Q. W., Design of minimal NOR/NAND logical circuits, «Electron. Eng.», 1965, 37. N 451, стр. 592—597.
Liu C. N., A state variable assignment method for asinchronous sequental switching circuits, «Journ. Assoc. Comput. Mach.», 1963, 10, N 2, стр. 209—216.
Loeli M., Rosenfeld G., Logical design of ternary switching-circuits, «IEEE Trans. Electron. Comput.», 1965, 14, N 1, стр. 19—29.
Loomis H. H., Jr., Wyman R. H., Jr., On complete sets of logic primitives, «IEEE Trans. Electron. Comput.», 1965, 14, N 2, стр. 173—174.
Mac Sorley O.L, Binary divider, пат. США, кл. 235-164, № 3192364.
Majerski S., On determination of optimal distributions of carry skips in adders, «IEEE Trans. Electron. Comput.», 1967, 16, N 1, стр. 45—58.
Majerski S., Wiweger M., NOR-gate binary adder with carry completion detection, «IEEE Trans. Electron. Comput.», 1967, 16, N 1, стр. 90—92.
Maley G. A., Full adder and substractor using NOR logic, пат. США, кл. 235-176, № 3074640.
Mano M. M., Converting to NOR and NAND logic, «Electro-Technology», 1965, 75, N 4, стр. 34—37.
Martin Y., Voitel K., Rechenanordnung zur Multiplikati-on zweierpositiver Zahlen, пат. ФРГ, кл. 42m, 14 (Q06f), № 115873.
Martin A. R., Rosenstein A. B., Shiftrix for high-speed multiplication, «IEEE Trans, on Electron. Comput.», 1965, vol. EC — 14, N 4.
McCluskey E. J., Logical design theory of NOR gate networks with no completented inputs, «IEEE—Switching circuit Theory and Logical Design», 4th Annual Symposium — Special Publ. S-156, oct. 1963, стр. 137—148.
McLane G. F., Multioperture plate half adder., пат. США, кл. 307-88, № 3243599.
McLane G. F., Multioperture plate logic, пат. США, кл. 340-174, №3253268.
Meggit J. E., Pseudo division and pseudo multiplication processes, «IBM Y. Res. and Developm.», 1962, 6, N 2, стр. 210—226.
Mendelson M. J., Apparatus for performing arithmetic operations, пат. США, кл. 235—155, № 3018955.
Merner J. N., Digital divider for integer and remainder division operations, пат. США, кл. 235—160, № 3254204.
Merriam Ann S., Asynchronous binary counter circuits, пат. США, кл. 328-41 № 3238461.
Meo A. R., Sulla sintesi direti NAND NOR a molti livelli, «Calcolo», 1965, 2, N 1, стр. 1—82.
Merrill R. D., Jr., Improving digital computer performance using residue number theory, «IEEE Trans. Electron. Comput.», 1964, 13, N 2, стр. 93—101.
Metz G. F., Digital comparator, пат. США, кл. 340-146.2, № 3217293.
Metze G. F., Class of binary divisions yielding minimally represented quotients, «IRE Trans. Electronic Comput.», 1962, 11, N6, стр. 761—764.
Meyer B. W., Arithmetic circuits, пат. США, кл. 235-176, №3147372.
Miller W. S., Electronic gang switching system, пат. США, кл. 235-164, № Re 25724.
Mines H. W., Binary counter circuit, пат. США, кл. 235-179, №3141966.
Morelli S., Stefanelli R., Alcuni circuiti contatori di tipo parallelo, «Alta frequenza», 1965, 34, N 12, стр. 836—846.
Morris D. J. . Alexander W., Binary multiplication and division using ferrite cores, «Electron. Eng», 1962, 34, N 414, стр. 549—552, 579, 586.
Mukhopadhyay A., Symmetric ternary switching functions, «IEEE Trans. Electron. Comput.», 1966, 15, N 5, стр. 731-739.
Naff G. W., Yourke H. S., Tunnel diode shift registers, пат. США, кл. 307-88.5, № 3209158.
NAND logic operation in serial adder circuits, «Syst. Design.», 1964, 8, N 4, стр. 8—9.
Nandi S. K., Krishnamurthy E. V., A simple technique for digital division, «Communs. ACM», 1967, 10, N 5, стр. 299— 301.
Nashelsky L., Digital computer theory. S. 1, John Wiley and Sons, 1966, стр. 321.
Newborn M. M., Propagating logic structures, Nat. Electronics Conference — Proc. 1966, 22, paper 2263, стр. 731—736.
Newman E. A., Stringer Y. B., Electrical digital computing engines, англ. пат., кл. 106 (1), № 907381.
Newton Y. D., Core counter, пат. США, кл. 340-174, № 3181130.
Nutall Y., Computer process for conversion of binary-coded decimal numbers to pure binary form, «Radio and Electronic Engr.», 1965,

30, N 5, стр. 317—318.
O m a n R . M., Fast multiply system, пат. США, кл. 235-164, №3192367.
Opérateur arithmétique pour calculateur numérique, фр. пат., кл. G06b, № 1376559.
O s o f s k y H . , Parallel adder circuit with impoved carry circuitry, пат. США, кл. 235-175, № 3234371.
O t t a w a y G . e t al., Quotient guess divider, пат. США, кл. 235-156, № 3234367.
O w e n C . E ., Improvements in data storage or code conversion arrays, англ. пат. кл. G4A (G06f), № 976169.
Parallel and concurrent computer systems (Sympos. Summary), «Proc. FIP Congr., New York City, 1965, vol 2», Washington, D. C, Spartan Books, 1966, стр. 319—322.
P a r k e r B . D ., Improvements in or relating to digital registers, англ. пат., кл. G4 C (G06f), № 981296.
P a r t o n K . C ., The digital computer, Oxford-London, Pergamon Press, 1966.
P a u l G . T . et al., High-speed multiplier, пат. США, кл. 235-164, № 3115574.
P a u l G . T ., D i r a c J . F ., A m d a h l G . M ., Large scale shifter, пат. США, кл. 340-172. 5, № 3274556.
P i l a t o M . , Logique à trois ètats stables, «Electronique», 1966, N 57, стр. 3—8.
P l a t z e r G . E ., Using transistors circuits to multiply and divide, «Electronics», 1966, 39, N 7, стр. 109—115.
P o d e g s A . , Structure of systems 360—4. Channel design considerations, «IBM Systems J.», 1964, 3, № 2, стр. 165—180.
P o m e r e n e J . H . , C o c k e J ., Asynchronous adder-substractor system, пат. США, кл. 235-153, № 3051387.
P o m e r e n e J . H ., Asynchronous add-substract system, пат. США, кл. 235-153, № 3058656.
P o t o c k i A . , Dispositif et procédé pour effectuer des opérations arithmétiaues, фр. пат., кл. G06d, № 80453.
P u g m i r e J . M ., Improvements in adders, англ. пат., кл. G4A (G06f), № 1038770.
Q u a s t s e J . T . , K e i r R ., A paraller accumulator a general-purpose computer, «IEEE Trans. Electron. Comput.», 1967, 16, №2, стр. 165—171.
R e a c h R . W . , J r . , K a n n W . N ., Data processing apparatus, пат. США, кл. 235-175, № 3003695.
R e i j n s G . L . , High-speed binary parallel adder, «Tijdschr. Ne-derl. radiogen.», 1962, 27, № 1, стр. 1 — 15.
R i c e R ., Photologic arithmetic circuits, пат. США, кл. 235-176, № 3138704.
R i c h a r d s R . K ., Electronic digital systems, New York-London-Sydney, John Wiley and sons, inc., 1966.
R i c h a r d s o n A . , F o s s R . C , New binary counter circuit, «Electron. Letters», 1965, I, № 10, стр. 273.
R i l e y W . B . , Illiac 4, world's fastest computer, won't be slowed by criticism, «Electronics», 1967, 40, N 10, стр. 141—144.
R i o r d a n R . H . S . , M o r t o n R . R . A., The use of analog techniques in binary arithmetic units, «IEEE Trans. Electron. Comput.», 1965, 14, N 1, стр. 29—35.
R o t h H . H . , Linear binary shift register circuits utilising minimum number of Mod-2 adders, «IEEE Transactions on Information Theory», 1965, 11, N 2, стр. 215—220.
R o t h R . I . , F l e i s h e r H ., Mixed code calculator, пат. США, кл. 235-175, № 3191014.
R o t h R . I . , F l e i s h e r H ., Square root device employing converging approximation, пат. США, 235-164, № 3234369.
R o t h R . I . , F l e i s h e r H ., Photoconductive combinational multiplier, пат. США, кл. 235-164, № 3163749.
R o z i e r C . P . , Decimal-to-binary conversion using octal radix arithmetic, «IRE Trans. Electron. Comput.», 1962, 11, N 5, стр. 708—709,
R u s s o R . L . , Synthesis of error-tolerant counters using minimum distance three state assignements, «IEEE Trans. Electron. Comput.», 1965, 14, N 3, стр. 359—366.
S a n t o s J . , A r a n g o H ., Base, 3 vs base 2 synchronous arithmetic units, «IEEE Trans. Electron. Comput.», 1964, 13, N 5, стр. 608—609.
S a s a k i A ., Addition and substraction in residue number systems, «IEEE Trans. Electron. Comput.», 1967, vol. EC-16, N 2, стр. 157—164.
S c a n l o n J . M . , T o r n g H . C , The design of binary adders with a flexible criterion function, «IEEE Internat. Convent. Rec», 1966, 14, N 3, стр. 44—60.
S c h a e f e r D . H . , Full binary adder, пат. США, кл. 235-176, № 3196261.
S c h m i t t E . I . , S m i h J . G ., Digital computing systems, пат. США, кл. 235-176, № 3023963.
S c h m i t t W . F . , Parallel adder with fast carry network, пат. США, кл, 235-175, № 3192369.
S c h m o o k l e r M . S ., Microelectronics opens the gate to faster digital computers, «Electron. Design», 1966, 14, N 16, стр. 52—57.
S c h m o o k l e r M . S . , Perform binary division fast, «Electron Design», 1966, 14, N 17, стр. 61—65.
S c h ö n h a g e A . , Multiplikation groβer zahlen, «Computing», 1966, 1, N 3, стр. 182—196.
S c h ü t z e D . , S c h ü t z e L . , Elektronische Zäahlvorricht-ung mit Vor-Rückwärtszählung, пат. ГДР, кл. 21 a[1], 36/22 (H03k), № 45108.
S c o l l a r I . , A reversible decade counter with simmetrical 1-2-4-2 coding, «Electron. Eng», 1965, 37, N 449, стр. 468—470, 493, 500.
S e e d s R . B . et al., Integrated complementary transistor nanosecond logic, «IEEE Proc», 1964, 52, N 12, стр. 1584—1590.
S i e r r a H . M ., Division system, пат. США, кл. 235-160, №3028086.
S i e r r a H . M ., Multiplier system, пат. США, кл. 235-160, №3033455.
S i e r r a H . M ., Matrix arithmetic system with input and output error checking circuits, пат. США, кл. 235-153, № 3063636.
S i l v e r b e r g M . , Logic circuits, пат. США, кл. 307-88.5, № 3274398.
S i m s J . C , Jr., High speed asynchronous computer, пат. США, кл. 307-88, № 3290511.
S k l a r S . , M a c D o n a l d R ., High speed binary to decimal conversion, «EDN Electron. Engr's Design Mag.», 1967, 12, N 5, стр. 54—57.
S l a d e A . E . , B u c k D . A ., Cryotron translators, пат. США, кл. 235-164, № 3019978.
S n y d e r Y . B . , Digital counter configurations, «Data Syst. Eng», 1963, 18, N 10, 16—18.
S q u i r e J . S . , An 11-cryotron full adder, «IRE Trans. Electron. Conput.», 1962, 11, № 5, стр. 710—711.
S t a f f o r d R . A ., Apparatus for performing high speed division, пат. США, кл. 235-164, № 3023961.
S t e i n M . L ., Divide-and-Correct Methods for Multiple Procession Division, «Comm. of the ACM», 1964, 7, N 8, стр. 472— 478.
S t e v e n s W . I . , Structure of system 360—2 System Implementations, «IBM System J.», 1964, 3, N 2, стр. 136—143.
S t e w a r t J . R ., Highspeed binary divider, пат. США, кл. 235-164, № 3192365.
S t o n e H . S ., One-pass compilation of arithmetic expressions for a parallel processor, «Communs ACM», 1967,10, N 4, стр., 220—223.
S v o b o d a A . , V a l a c h M . , Arithmetickỳ obvod operační jednotsky, чехосл. пат., кл. 42m, 14. № 99069.
S w e e n y D . W ., Divider device for skipping a string of zeros or radix-minusone digits, пат. США, кл. 235-164, № 3145296.
T a n a k a R . I . , Some options in the design of a residue arithmetic computer, «Proc. Nat. Electron. Conf., Chicago, I11., 1963», Chicago, 111., 1963, стр. 123—130.
T h o r n t o n J . E . , Parallel Operation in Control Data 6600, «AFIPS-Joint Comp. Conf.- Proc.», 26, part 2, 1964, стр. 33—40.
T i l t o n C . J ., Asynchronous multiplier, пат. США, кл. 235-165, № 3085747.

T o m a s u l o R . M ., An efficient algorithm for exploiting multiple arithmetic units, «IBM J. Res. and Developm», 1967, 11, N 1, стр. 25—33.

U d a g a w a K . , G o t o M . , Generalized minimal representation of binary numbers and its application to nonrestoring binary division, «Electronics and Communications in Japan», 1965, 48, N 5, стр. 79—89.

И т о К и и т и р о , Способ сложения и вычитания, яп. пат., кл. 114 А 513, № 21268.

V o l t i n I . V ., Floating point arithmetic circuit, пат. США, кл. 235-164, № 3193669.

W a g n e r E . G ., High speed binary counter, пат. США, кл. 340-174, № 3177474.

W a k u l i c z A . , Podstawy arytmetyczne kodów liczbowych, добавление к книге K a r c e w M ., Arytmometry elektronicznych maszyn cyfrowych, Warszawa, 1961.

W a l k e r J . A ., Simplify NAND-circuits cynthesis, «Electron Design», 1966, 14, N 6, стр. 203—209.

W a l k e r M . , The neglected EXCIUSIVE OR «EDN Electron. Engr's Design. Mag.», 1967, 12, N 6, стр. 78—81.

W a l l a c e C . S ., A suggestion for a fast multiplier, «IEEE Trans. Electron. Comput.», 1964, 13, N 1, стр. 14—17.

W a r l i t z G . , Uber eine Schaltungsalgebra für NOR and NAND, «Electron. Datenverarb.», 1967, 9, N 2, стр. 65—73.

W e b b J . A . , High speed binary adder and/or substractor circuit, пат. США, кл. 235-175, № 3170063.

W e b b J . A ., A high-speed digital computation technique, «Electron. Design», 1964, 12, N 14, стр. 30—33.

W e i n m a n n A . , Langzeitgenaue fur Regelungszwecke bestim-mte Rechenwerke nach dem Rücksprungverfahre, «Elin-Z.», 1966, 18, N 1, стр. 32—87, II, III.

W e i n s t e i n H . A . , A high-speed «compare» circuit, «IEEE Trans. Electron. Comput.», 1963, 12, N 4, стр. 410—411.

W i l h e l m H . , Kombinierter Serienvolladdierer/substrahierer, пат. ФРГ, кл. 42m, 14, № 1128687.

W i l h e l m H ., et al., Schaltungsanordnung zum Divideren von Binärzahlen, пат. ФРГ, кл. 42m, 14 (G06f), № 1167070.

W o o d J . R . , Arithmetic element, пат. США, кл. 235-175, № 3089045.

Y o u n g J . F ., Counters and shift registers using static switching units, «Electron, Eng», 1965, 37, N 445, стр. 161—163.

Z s c h e k e l H . , Multipikationsanordnung, пат. ФРГ, кл. 42m, 14 (G06f), № 1141810.

Михаил Александрович Карцев

Арифметика цифровых машин

М., 1969 г., 576 стр. с илл.

Редактор *И. М. Овчинникова*

Техн. редактор *К.Ф.Брудно*

Корректоры *В. П. Горячева* и *Г. С. Иванова*

Сдано в набор 8/X 1968 г.
Подписано к печати 21/1II 1969 г.
Бумага 84×108×1/32. Физ. печ. л. 18. Условн. печ. л. 30,24.
Уч.-изд. л. 28,24. Тираж 10700 экз. Т-04830.
Цена книги I р. 98 к. Заказ 1174

Издательство «Наука»
Главная редакция
физико-математической литературы
Москва В-71, Ленинский проспект, 15

2-я типография издательства «Наука», Шубинский пер., 10