Министерство образования и науки Российской Федерации

Государственное образовательное учреждение
высшего профессионального образования
«Омский государственный технический университет»

И. В. Потапов

ЭЛЕМЕНТЫ ПРИКЛАДНОЙ ТЕОРИИ ЦИФРОВЫХ АВТОМАТОВ

Учебное пособие

Омск
Издательство ОмГТУ
2011

УДК 004.315(075)
ББК 32.973.2я73
П64

Рецензенты:

М. Ф. Шакиров, канд. техн. наук, доцент, зам. руководителя Управления Роскомнадзора по Омской области;

В. Т. Гиль, канд. техн. наук, доцент ОмА МВД России

Потапов, И. В.
П64 **Элементы прикладной теории цифровых автоматов**: учеб. пособие / И. В. Потапов. – Омск: Изд-во ОмГТУ, 2011. – 156 с.

ISBN 978-5-8149-1039-4

В пособии рассматриваются основы построения алгоритмов выполнения арифметических операций над числами, представленными прямыми и инверсными кодами в двоичной однородной позиционной системе счисления, а также в D-кодах. Даны подробные примеры выполнения арифметических операций. Рассмотрены различные способы представления числовой информации в ЭВМ. Описаны базовые принципы построения логических схем прямого распространения по их аналитическому описанию и методы минимизации таких схем. Рассмотрены базовые подходы к построению моделей функционирования и структурному синтезу цифровых автоматов с памятью.

Пособие предназначено для студентов технических вузов очной, заочной и дистанционной форм обучения, обучающихся по направлению «Информатика и вычислительная техника», изучающих базовые специальные дисциплины.

Печатается по решению редакционно-издательского совета Омского государственного технического университета

УДК 004.315(075)
ББК 32.973.2я73

ISBN 978-5-8149-1039-4

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Вычислительные машины в зависимости от принципа их работы подразделяются на две большие группы: вычислительные машины непрерывного действия и вычислительные машины дискретного действия (цифровые машины).

В вычислительных машинах непрерывного действия все входные, промежуточные и выходные величины представлены в виде некоторых физических величин, количественные характеристики которых соответствуют изображаемым ими числам. Математическим действиям с числами в таких машинах соответствуют определенные преобразования физических величин, при этом математическое описание этих преобразований совпадает с выполняемыми математическими действиями над числами.

В цифровых вычислительных машинах (ЦВМ) все исходные, промежуточные и выходные данные изображаются в виде совокупности цифр в позиционной системе счисления. Наибольшее распространение в цифровых машинах получила двоичная система счисления, в которой цифра имеет только два значения – «0» и «1». Это обусловлено близким к оптимальному соотношению между надежностью, помехоустойчивостью, быстродействием, функциональными возможностями, схемотехнической сложностью, технологичностью производства, удобством эксплуатации и т.д.

Таким образом, физическая величина, изображающая двоичную цифру, должна иметь только два четко различимых состояния. Дискретный характер информации позволяет реализовать запоминание больших объемов информации, что позволяет разворачивать вычислительный процесс во времени. На любом этапе вычислений исходные данные выдаются запоминающим устройством на входы арифметического устройства, а промежуточные результаты вычислений вновь запоминаются в устройстве хранения информации до тех пор, пока они не понадобятся для дальнейших вычислений.

Блоки цифровых вычислительных машин, представляющие собой совокупность оборудования, выполняющую определенные законченные действия, состоят, как правило, из двух частей: операционного автомата, выполняющего действия по обработке информации (например, арифметические операции над поступающими на его вход операндами), и управляющего автомата, осуществляющего сбор информации о функционировании операционного автомата и вырабатывающего в соответствии с заданным алгоритмом функционирования необходимые

управляющие сигналы. Процесс работы блока может быть представлен как совокупность конечного числа элементарных операций, таких, как суммирование, сдвиг, прием и выдача кода и др. Элементарные операции выполняются с помощью узлов, таких, как регистры, сумматоры, счетчики и др.

Решение задач на ЦВМ сводится к выполнению арифметических и логических операций над исходными данными и промежуточными результатами в соответствии с заданным алгоритмом. Перед решением задачи алгоритм ее решения записывается в виде последовательности простейших операций, выполнение которых предусмотрено при проектировании вычислительного устройства. Очевидно, программа строится так, чтобы ее выполнение не зависело от конкретных значений чисел. Все вычислительные операции в ЦВМ выполняются либо аппаратно (с помощью специально предусмотренного для этих целей оборудования), либо программным способом путем разбиения заданной операции на ряд простейших (например, арифметическая операция умножения может быть сведена к последовательному выполнению серии сложений и сдвигов).

Вся информация (числовая и управляющая) в ЦВМ кодируется совокупностями цифр. В свою очередь цифры представляют собой квантованные по нескольким уровням электрические сигналы. Множество значений, которые могут принимать сигналы в ЦВМ, называют *алфавитом*. Элементы алфавита называют *буквами*, а конечные упорядоченные последовательности букв – *словами* в данном алфавите. В настоящее время наибольшее распространение имеет алфавит из двух букв (двоичный алфавит), в котором буквы кодируются значениями 0 и 1.

Процесс обработки информации в ЦВМ может быть представлен как последовательность операций над словами. Эти операции, в свою очередь, могут рассматриваться как совокупности элементарных операций над буквами алфавита. Для обеспечения правильности работы цифрового вычислительного устройства сигналы, представляющие буквы алфавита, должны быть разделены либо во времени, либо в пространстве. Временному разделению сигналов соответствуют так называемые *последовательные коды*. В этом случае передача информации осуществляется по одной шине последовательно, буква за буквой. Пространственное разделение соответствует параллельным кодам, когда для каждой буквы слова имеется отдельная шина и передача всех букв осуществляется одновременно. Разделение сигналов во времени и пространстве необходимо при передаче информации и ее обработке.

Возможен также смешанный способ разделения сигналов – последовательно-параллельные коды.

Настоящее учебное пособие посвящено элементарным вопросам представления чисел в машинах дискретного действия и построения арифметических процедур, применяемых в цифровых вычислительных устройствах для выполнения числовых расчетов. Также в пособии представлены основы синтеза логических комбинационных схем, т.е. схем прямого распространения, предназначенных для дискретного преобразования информации в соответствии с описывающими их логическими функциями. Отдельная глава учебного пособия посвящена вопросам представления процессов функционирования и синтеза автоматов с памятью, для полного понимания которой необходимо тщательное изучение материала предшествующих глав, касающегося вопросов синтеза комбинационных схем.

Для понимания излагаемого материала читатель должен быть знаком с простейшими арифметическими действиями (сложение, сдвиг) над числами в двоичной и других однородных позиционных системах счисления, а также иметь хотя бы начальные представления об устройстве и функциональных возможностях базовых элементов цифровой вычислительной техники – триггеров, регистров, сумматоров и т.п.

Для более полного освоения специализированных дисциплин студентам рекомендуется самостоятельно освоить материал источников, приведенных в разделе «Библиографический список» [1–12].

1. ПРЕДСТАВЛЕНИЕ ЧИСЕЛ В ЭВМ

1.1. Позиционные системы счисления

Системой счисления называется совокупность приемов записи чисел. Запись числа в некоторой системе счисления называют *кодом числа*. Отдельную позицию в изображении числа называют *разрядом*, а номер позиции – *номером разряда*. Любая система счисления, предназначенная для практического использования, должна обладать следующими свойствами:

– возможностью представления любого числа в заданном диапазоне;

– однозначностью представления чисел;

– простотой оперирования числами.

Системы счисления можно разделить на *позиционные* и *непозиционные*. К последним относится, например, римская система счисления. Основными недостатками непозиционных систем счисления являются:

– отсутствие нуля;

– необходимость использования бесконечного количества символов;

– сложность арифметических операций над числами.

Позиционными называют системы счисления, алфавит которых содержит ограниченное число символов, а значение каждой цифры числа определяется ее местоположением в числе. Например, десятичное число 545 означает 5 сотен, 4 десятка и 5 единиц.

В общем виде число в позиционной системе счисления может быть представлено как

$$A_{(p)} = a_n p_{n-1} \ldots p_1 + a_{n-1} p_{n-2} \ldots p_1 + \ldots + a_2 p_1 + a_1,$$

где a_i – цифра i-го разряда числа, $a_i \in \{0,1,\ldots, p_i - 1\}$; p_i – основание системы счисления.

Позиционные системы счисления подразделяются на *неоднородные* и *однородные*.

В неоднородных системах счисления основания не зависят друг от друга и могут принимать любые значения, поэтому такие системы называют еще *системами со смешанным основанием*. Примером неоднородной позиционной системы счисления является система представления времени. Так, например, число

$$A = 5 \cdot 365 \cdot 24 \cdot 60 \cdot 60 \cdot 1 + 10 \cdot 24 \cdot 60 \cdot 60 \cdot 1 + 22 \cdot 60 \cdot 60 \cdot 1 + 11 \cdot 60 \cdot 1 + 50 \cdot 1$$

является записью времени в 5 лет, 10 суток, 22 часа, 11 минут, 50 секунд. В этом примере $p_4 = 365$ суток, $p_3 = 24$ часа, $p_2 = 60$ минут, $p_1 = 60$ секунд, $p_0 = 1$ секунда.

Однородные позиционные системы счисления характеризуются тем, что в них веса отдельных разрядов числа представляют собой ряд членов геометрической прогрессии со знаменателем p. Число в таких системах счисления представляется полиномом вида

$$A_{(p)} = a_n p^n + a_{n-1} p^{n-1} + ... + a_1 p^1 + a_0 p^0 + a_{-1} p^{-1} + ... + a_{-m} p^{-m},$$

причем целой части числа соответствует полином

$$A_{1(p)} = a_n p^n + a_{n-1} p^{n-1} + ... + a_1 p^1 + a_0 p^0,$$

а дробной части – полином

$$A_{2(p)} = a_{-1} p^{-1} + ... + a_{-m} p^{-m}.$$

Здесь, как и ранее, a_i – цифры числа, p – натуральное целое число, называемое *основанием системы счисления*. Следует помнить, что основание p записывается числом «10» в своей системе счисления.

Различают также кодированные позиционные системы счисления, в которых цифры системы счисления с основанием p кодируются при помощи комбинаций цифр другой системы счисления с основанием q. Такое число может быть представлено в виде полинома

$$A = (a_{n,k} q^n + a_{n-1,k} q^{n-1} + ... + a_{1,k} q^1 + a_{0,k} q^0) p^k + (a_{n,k-1} q^n + ... + a_{0,k-1} q^0) p^{k-1} + ...$$
$$... + (a_{n,0} q^n + ... + a_{0,0} q^0) p^0.$$

Примером такой системы может служить так называемая двоично-десятичная система представления чисел (D-код), в которой каждой цифре десятичного числа соответствует двоичная тетрада с весами 8421. Например, десятичное число 259 в двоично-десятичной системе с естественными весами 8421 запишется как

0010	0101	1001
2	5	9.

D-код 8421 редко применяется в вычислительной технике, поскольку он не является самодополняющимся, т.е. двоичные коды любых двух десятичных цифр, дополняющих друг друга до 9 (их сумма равна 9), не дополняют друг друга до 15. Примером самодополняющегося D-кода является D-код 8421+3 с потетрадным избытком 3, в котором вышеприведенное число 259 запишется как

0101 1000 1100
2+3 5+3 9+3.

Более подробно о D-кодах см. в разделе 1.5.

1.2. Обоснование применения в ЭВМ двоичной системы счисления

При выборе системы счисления для применения в ЭВМ следует учитывать совокупность различных параметров, среди которых можно выделить:

– наличие физических элементов для представления и хранения цифр системы;

– экономичность системы, т.е. количество элементов, требуемых для представления и хранения многоразрядных чисел;

– быстродействие вычислительных устройств;

– высокую помехоустойчивость кодирования цифр.

Очевидно, что по первому и последнему критериям двоичная система счисления наиболее выгодна в применении, поскольку для кодирования двух цифр этой системы могут использоваться элементы, принимающие только два устойчивых состояния.

Остановимся более подробно на оценке экономичности и быстродействия вычислительных устройств, использующих системы счисления с различными основаниями.

Оценим экономичность использования различных систем счисления. Для этого введем некоторую оценочную величину $D_i = p_i \cdot n_i$, пропорциональную количеству деталей оборудования [4], где индекс i соответствует i-й системе счисления, а n_i – число разрядов.

Количество чисел, которые можно представить в i-й системе счисления, определяется следующим образом:

$$N_i = p_i^{n_i},$$

откуда

$$n_i = \log_{p_i} N_i.$$

Таким образом, выражение для D_i записывается как

$$D_i = p_i \log_{p_i} N_i = \frac{p_i}{\log_{N_i} p_i}.$$

Предположим, что величина p_i изменяется не дискретно, а непрерывно и количество чисел $N_i = N$ одинаково для всех систем. Тогда после исследования на экстремум функционала

$$D(p) = \frac{p}{\log_N p}$$

получим следующий вывод: оптимальное значение основания системы счисления равно основанию натурального логарифма, т.е. $p_{\text{опт}} = \text{e} \approx 2{,}7182$. Для определения экономичности системы счисления с целочисленным основанием введем относительное значение

$$D_i^{\text{отн}} = \frac{D_i}{D_{\text{опт}}} = \frac{D_i}{\text{e} \cdot \ln N},$$

откуда, с учетом предыдущих формул, получим

$$D_i^{\text{отн}} = \frac{p_i \log_{p_i} N}{\text{e} \cdot \ln N} = \frac{p_i}{\text{e} \cdot \ln p_i}.$$

Сведем в таблицу решения вышеприведенного уравнения для различных целочисленных оснований систем счисления.

Таблица 1.1

Основание	2	3	4	5	10
$D_i^{\text{отн}}$	1,062	1,005	1,062	1,143	1,598

Из табл. 1.1 следует, что в цифровых вычислительных устройствах наиболее выгодно применять систему счисления с основанием $p = 3$. Однако для хранения произвольной троичной цифры требуется элемент с тремя устойчивыми состояниями, уступающий в плане

помехоустойчивости и простоты физической реализации элементу с двумя устойчивыми состояниями.

Рассмотрим еще один критерий оценки систем счисления с разными основаниями – по быстродействию выполнения арифметических операций. В качестве характеристики быстродействия выберем количество элементарных сложений, выполняемых при умножении *n*-разрядных *p*-ичных чисел.

В каждом цикле умножения максимальное количество сложений не превышает величину $(p_i - 1)$. Учитывая, что количество циклов умножения пропорционально разрядности множителя n_i, с применением вышеприведенных формул общее количество сложений запишется как

$$K_i = (p_i - 1)n_i = (p_i - 1)\log_{p_i} N.$$

Сведем в таблицу значения параметра оценки быстродействия вычислительного устройства при использовании систем счисления с различными целыми основаниями (табл. 1.2). Для этого нормируем последнее выражение относительно $p_i = 2$ и вычислим относительную характеристику быстродействия по формуле

$$K_i^{\text{отн}} = \frac{(p_i - 1)\log_{p_i} N}{\log_2 N} = (p_i - 1)\log_{p_i} 2 = \frac{(p_i - 1)}{\log_2 p_i}.$$

Таблица 1.2

Основание	2	3	4	5	10
$K_i^{\text{отн}}$	1,000	1,262	1,500	1,725	2,709

Таким образом, цифровое вычислительное устройство, работающее в двоичной системе счисления, характеризуется более высоким быстродействием относительно систем счисления с другими целыми основаниями.

1.3. Представление двоичных чисел с фиксированной и плавающей запятой

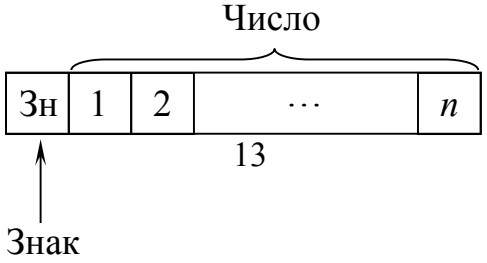
Известны три формы представления чисел: естественная (в некоторых источниках естественной формой представления чисел в ЦВМ называют форму представления с фиксированной запятой, которую здесь

мы рассмотрим отдельно), с фиксированной запятой и с плавающей запятой (нормальная или полулогарифмическая). Кроме перечисленных (получивших наибольшее распространение) форм представления чисел могут также использоваться формы, получаемые путем функционального преобразования чисел. Например, логарифмическая форма, позволяющая заменять операции умножения и деления чисел операциями сложения и вычитания их логарифмов.

При естественной форме представления целая и дробная части числа отделяются друг от друга запятой и для каждого произвольного числа необходимо указывать положение запятой в одном из разрядов кода. Такая форма представления чисел не получила широкого распространения в цифровых вычислительных устройствах, поскольку для ее использования требуется дополнительное оборудование и существуют трудности при оперировании очень большими или очень малыми по абсолютной величине числами.

При использовании формы представления чисел с фиксированной запятой определяется место фиксации запятой после старшего разряда в разрядной сетке ЭВМ или перед младшим разрядом. Поскольку для любого числа положение запятой строго фиксировано, специальный разряд для нее не отводится. Если запятая фиксирована после старшего разряда, то вычислительное устройство оперирует с дробями, а если перед младшим – то устройство оперирует с целыми числами. В дальнейшем будем полагать, что запятая фиксирована после старшего разряда, если обратное специально не оговорено. При этом числа должны быть представлены в виде правильных дробей, для чего используются специальные масштабные коэффициенты. Следует учитывать, что при такой форме представления возможны потери старших значащих цифр числа вследствие переполнения разрядной сетки, т.е. в том случае, когда результат выполнения арифметической операции является неправильной дробью (по модулю больше 1). Это обстоятельство накладывает ряд ограничений на используемые при вычислениях числа: во-первых, модуль суммы двух чисел не должен превышать единицу; во-вторых, модуль делимого должен быть меньше модуля делителя.

На рис. 1.1 изображено машинное представление *n*-разрядного числа с фиксированной после знакового разряда запятой.



Старший разряд числа с фиксированной запятой используется для кодирования знака. При этом обычно знак положительного числа кодируется наименьшей цифрой, а знак отрицательного числа – наибольшей.

В двоичной системе счисления знаку «плюс» соответствует цифра 0, а знаку «минус» – цифра 1. В некоторых случаях для кодирования знака может быть использовано более одного разряда.

Величины двоичных чисел (правильных дробей), представляемых в машинах с фиксированной запятой, лежат в пределах

$$2^{-n} \le |A| \le 1 - 2^{-n},$$

а диапазон представления чисел в машине с фиксированной запятой равен

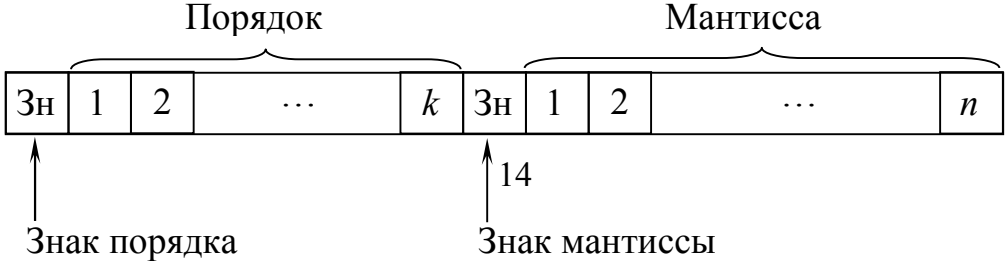
$$D_{\text{фикс}} = \frac{|A_{\max}|}{|A_{\min}|} = 2^n - 1.$$

Наибольшее распространение в ЦВМ получила нормальная форма представления чисел, которая также называется *формой представления с плавающей запятой*. При этом числа представляются в виде

$$A = a \cdot p^m,$$

где a – правильная дробь, удовлетворяющая условию $0{,}5 \le |a| < 1$, называемая мантиссой числа; p – основание системы счисления; m – целое положительное или отрицательное число, называемое порядком числа, указывающее местоположение запятой в числе.

На рис. 1.2 изображено машинное представление числа с плавающей запятой.



Максимальное по абсолютной величине число, которое может быть представлено в машине с плавающей запятой, определяется как

$$2^{2^k-1}(1-2^{-n}),$$

однако, учитывая, что при больших n величина 2^{-n} мала, ею можно пренебречь.

Таким образом, величины двоичных чисел, представляемых в машинах с плавающей запятой, лежат в пределах

$$2^{-2^k} \le |A| \le 2^{2^k-1},$$

а диапазон представления чисел в машине с плавающей запятой равен

$$D_{\text{плав}} = \frac{|A_{\max}|}{|A_{\min}|} = 2^{2^{k+1}-1}.$$

Формы представления чисел с фиксированной и плавающей запятой обладают как достоинствами, так и недостатками. Так, например, в качестве недостатков можно отметить, что при использовании формы представления чисел с фиксированной запятой возникает необходимость в масштабировании, что усложняет программирование вычислений, а в случае использования формы представления с плавающей запятой возрастает сложность оборудования и снижается быстродействие за счет необходимости выполнения операций с порядками и нормализации мантисс (приведения к форме $0{,}5 \le |a| < 1$). Однако, учитывая положительные качества обеих форм представления, в универсальных ЦВМ они могут использоваться совместно. Так, например, форма представления с фиксированной запятой может использоваться при решении задач целочисленной арифметики.

1.4. Прямой и инверсные коды чисел

Различают прямой код числа и инверсные коды, к которым относятся обратный и дополнительный коды.

Прямым кодом двоичного числа называется его изображение в естественной записи, причем в знаковом разряде отрицательного числа записывается единица, а положительного числа – ноль. Таким образом, прямой код двоичной правильной дроби определяется выражением

$$[A]_{пр} = \begin{cases} A, & \text{если } A \geq 0; \\ 1 + |A|, & \text{если } A \leq 0. \end{cases}$$

На рис. 1.3 представлена геометрическая интерпретация области чисел и области их изображений в прямом коде.

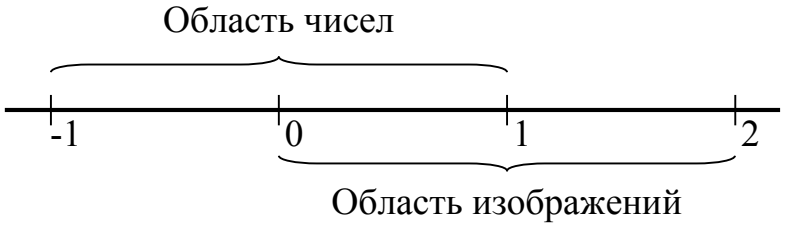


Рис. 1.3

Таким образом, область положительных чисел совпадает с областью их изображений, а область отрицательных чисел преобразуется в область изображений по формуле $1 + |A|$.

Ноль в прямом коде имеет два абсолютно эквивалентных значения:

$$0{,}000\ldots0;$$
$$1{,}000\ldots0.$$

При выполнении операции вычитания, заменяемой в вычислительных устройствах операцией сложения чисел с разными знаками, использование прямого кода неудобно, поскольку требуется специальная процедура формирования знака результата. Поэтому для кодирования отрицательных чисел используются так называемые инверсные коды.

Дополнительный код двоичной правильной дроби определяется выражением

$$[A]_{д} = \begin{cases} A, & \text{если } A \geq 0; \\ 2 - |A|, & \text{если } A < 0, \end{cases}$$

а дополнительный код целого двоичного n-разрядного числа – выражением

$$[A]_{д} = \begin{cases} A, & \text{если } A \geq 0; \\ 2^{n+1} - |A|, & \text{если } A < 0. \end{cases}$$

Из приведенных выражений следует, что дополнительный код положительного числа совпадает с его изображением в прямом коде. Дополнительный код отрицательного двоичного числа образуется путем инвертирования всех разрядов прямого кода числа и прибавления к младшему разряду единицы по правилам двоичной арифметики. В знаковый разряд отрицательного числа записывается единица.

Число ноль в дополнительном коде имеет только одно изображение:

$$0{,}000\ldots0.$$

Различают также модифицированный дополнительный код, отличающийся наличием удвоенного знакового разряда. Два знаковых разряда используются для обнаружения переполнения разрядной сетки при выполнении сложения чисел с одинаковыми знаками, модуль суммы которых превышает единицу. Модифицированный дополнительный код определяется выражением

$$[A]_{\text{д}}^{\text{м}} = \begin{cases} A, & \text{если } A \geq 0; \\ 4 - |A|, & \text{если } A < 0, \end{cases}$$

при этом знак положительного числа кодируется двумя нулями, знак отрицательного числа – двумя единицами. Ноль также имеет единственный код

$$00{,}000\ldots0.$$

На рис. 1.4 представлена геометрическая интерпретация области чисел и области их изображений в модифицированном дополнительном коде.

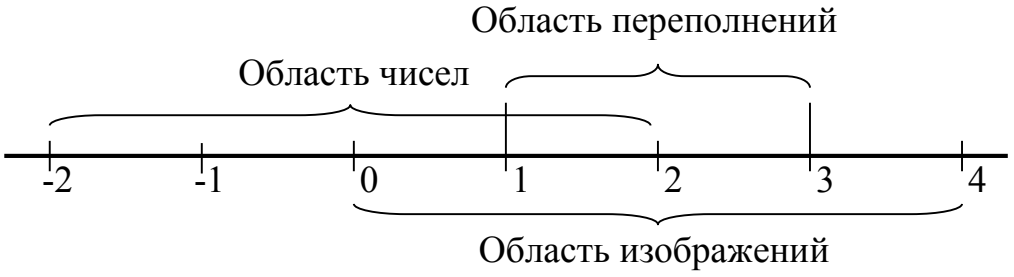


Рис. 1.4

Различают также еще один инверсный код, называемый *обратным кодом* и определяемый для *n*-разрядных двоичных правильных дробей выражением

$$[A]_{\text{об}} = \begin{cases} A, & \text{если } A \geq 0; \\ 2 - 2^{-n} - |A|, & \text{если } A \leq 0. \end{cases}$$

Из приведенного выражения следует, что для положительных чисел обратный код совпадает с прямым кодом. Обратный код отрицательных чисел определяется путем инвертирования разрядов прямого кода и установления единицы в знаковом разряде.

Ноль в обратном коде имеет два значения:

$$0{,}000\ldots0;$$
$$1{,}111\ldots1.$$

Как и в случае дополнительного кода, для обнаружения переполнений можно использовать модифицированный обратный код, отличающийся двойным знаковым разрядом.

Модифицированный обратный код *n*-разрядной двоичной правильной дроби определяется выражением

$$[A]_{\text{об}} = \begin{cases} A, & \text{если } A \geq 0; \\ 4 - 2^{-n} - |A|, & \text{если } A \leq 0. \end{cases}$$

Ноль в модифицированном обратном коде записывается двумя способами:

$$00{,}000\ldots0;$$
$$11{,}111\ldots1.$$

1.5. Двоично-десятичные коды чисел

Исходя из вышеизложенного, можно заключить, что создание универсальных цифровых вычислительных устройств, функционирующих в наиболее удобной для человека десятичной системе представления чисел (квантование сигналов осуществляется по десяти уровням), не является рациональным, поскольку характеристики десятичных схем уступают по некоторым параметрам характеристикам двоичных схем. Поэтому в ряде случаев для синтеза десятичных вычислительных устройств используют двоично-десятичное представление чисел, т.е. кодирование десятичных цифр комбинациями двоичных.

Очевидно, что кодирование десятичных цифр комбинациями двоичных цифр может быть осуществлено различными способами. При этом должно выполняться условие единственности, т.е. в выбранной системе кодов каждому десятичному числу ставится в соответствие единственная комбинация двоичных цифр и наоборот. Для этого число разрядов двоичного кода должно удовлетворять условию $n \geq [\log_2 10]$, где квадратные скобки означают округление до большего целого, т.е. $n \geq 4$. Если учесть, что при переборе различных систем кодирования любой десятичной цифре можно поставить в соответствие любое из 2^n n-разрядных двоичных чисел, то число способов кодирования определяется как

$$N = \frac{2^n!}{(2^n - 10)!},$$

т.е. как число размещений из 2^n по 10, поскольку эти соединения элементов отличаются друг от друга самими элементами или их порядком.

Из соображений минимизации наибольшее распространение получили двоично-десятичные коды, в которых десятичные цифры кодируются четырехразрядными двоичными комбинациями (двоичными тетрадами), т.е. $n = 4$. Отсюда число способов кодирования $N = 29\,059\,430\,400$. Известны также и другие коды с числом разрядов $n > 4$.

Среди такого большого числа кодов наибольшее распространение вследствие своей эффективности и изученности получили коды, удовлетворяющие условию единственности и обладающие свойствами аддитивности, упорядоченности, четности, самодополняемости и взвешенности.

Свойство аддитивности заключается в том, что код суммы десятичных цифр может быть получен как сумма кодов слагаемых.

Упорядоченность двоично-десятичных кодов определяется выполнением одного из условий:

$$K_0 < K_1 < ... < K_9 ,$$
$$K_0 > K_1 > ... > K_9 ,$$

где K_i – двоично-десятичный код *i*-й десятичной цифры.

Свойство четности заключается в том, что всем четным десятичным цифрам должны соответствовать только четные или только нечетные коды (аналогично для нечетных десятичных цифр).

Свойство самодополняемости кратко рассмотрено в разделе 1.1 и заключается в том, что сумма двоичного кода любой десятичной цифры и ее обратного двоично-десятичного кода должна быть равна двоично-десятичному коду цифры 9.

Свойством взвешенности обладают такие коды, в которых каждая десятичная цифра X может быть представлена полиномом вида

$$X = \sum_{i=1}^{n} a_i x_i ,$$

где $x_i \in \{0,1\}$ – двоичные цифры кода; a_i – некоторые веса, соответствующие разрядам двоичных кодовых комбинаций.

В табл. 1.3 представлены некоторые двоично-десятичные коды ($n = 4$) десятичных цифр.

Таблица 1.3

Десятичные цифры	Коды				
	8421	8421+3	8421+6	2421	3321
0	0000	0011	0110	0000	0000
1	0001	0100	0111	0001	0111
2	0010	0101	1000	1000 0010	0010
3	0011	0110	1001	1001 0011	1000 0100 0011
4	0100	0111	1010	1010 0100	1001 0101
5	0101	1000	1011	0101 1011	1010 0110
6	0110	1001	1100	0110 1100	1100 1011 0111
7	0111	1010	1101	0111 1101	1101
8	1000	1011	1110	1110	1110
9	1001	1100	1111	1111	1111

Код 8421 по сути является простейшим D-кодом с естественным порядком весов, т.е. в двоичной тетраде старший разряд имеет наибольший вес – 8 (2^3), а младший разряд – наименьший вес – 1 (2^0).

Этот код обладает свойствами аддитивности, упорядоченности, четности и взвешенности, однако, как указывалось выше, не обладает свойством самодополняемости. Для получения обратного кода необходимо увеличить значения, записанные в каждой тетраде числа на 6, т.е. преобразовать код 8421 в код 8421 с избытком 6 (8421+6), а затем инвертировать двоичные разряды.

Код 8421 с потетрадным избытком 3 (8421+3) обладает свойством самодополняемости, поэтому для получения обратного кода отрицательного числа достаточно просто инвертировать двоичные разряды в тетрадах.

Коды 2421 и 3321 являются кодами с искусственным порядком весов. В таких кодах некоторые десятичные цифры могут быть представлены несколькими кодовыми комбинациями. Для выполнения условия единственности некоторые разрешенные кодовые комбинации считают запрещенными.

Подробнее об арифметике двоично-десятичных кодов см. в разделе 2.7.

Вопросы для самоконтроля

1. Что такое позиционные системы счисления и в чем их отличие от непозиционных? Какие позиционные системы счисления вам известны?
2. Какие способы представления чисел в ЦВМ вам известны? В чем заключаются особенности представления чисел с фиксированной запятой?
3. В чем заключаются особенности представления чисел с плавающей запятой?
4. Какие инверсные коды чисел вам известны? Как они образуются и в чем их отличие от прямого кода?
5. Что такое двоично-десятичные коды чисел? Какие двоично-десятичные коды получили наибольшее распространение в цифровой вычислительной технике? Какими свойствами они характеризуются?

2. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ В ДВОИЧНЫХ КОДАХ

2.1. Сложение двоичных кодов

Сложение прямых кодов двоичных чисел редко применяется в вычислительной технике, поскольку при сложении чисел с разными знаками для формирования знака суммы требуется выполнение дополнительной нетривиальной процедуры. Гораздо проще складывать числа, представленные инверсными кодами, поскольку в этом случае знаковые разряды операндов, как и все остальные разряды, можно складывать по правилам двоичной арифметики, что приводит к автоматическому формированию знака суммы.

При сложении чисел с плавающей запятой на первом этапе необходимо сравнить порядки операндов. Если порядки слагаемых разные, это означает, что разряды мантисс операндов имеют разные веса и прямое сложение таких операндов невозможно. Поэтому перед выполнением сложения мантисс следует осуществить выравнивание порядков операндов.

Для выравнивания порядков операндов вычисляется разность порядков первого второго слагаемых $\Delta P = (P_1 - P_2)$ на сумматоре порядков. Очевидно, что операция вычитания $(P_1 - P_2)$ должна заменяться операцией сложения $P_1 + (-P_2)$, при этом оба порядка должны быть представлены в инверсном коде с учетом инвертированного знакового разряда вычитаемого.

Если $\Delta P = 0$, то порядки слагаемых равны и можно переходить к сложению мантисс. Если разность порядков положительная ($\Delta P > 0$), то первое слагаемое больше второго. В этом случае следует сдвинуть мантиссу второго слагаемого на ΔP разрядов вправо, а порядок второго слагаемого увеличить на ΔP. Когда разность порядков отрицательная ($\Delta P < 0$), первое слагаемое меньше второго, поэтому следует сдвинуть мантиссу первого слагаемого на $|\Delta P|$ разрядов вправо, а порядок первого слагаемого увеличить на $|\Delta P|$.

Возможен вариант, когда $|\Delta P|$ больше, чем число цифровых разрядов мантиссы, что приводит к обнулению одного из операндов. Тогда вместо уравнивания порядков следует сразу сформировать код суммы, равный коду большего слагаемого.

Поскольку порядки слагаемых могут быть с разными знаками, то при нахождении их разности может возникнуть переполнение сумматора порядков. Для фиксации переполнения часто используют

модифицированный код с удвоенным знаковым разрядом, поскольку в результате переполнения при сложении единица переноса выходит из старшего разряда мантиссы суммы и знак суммы будет изменен.

При использовании модифицированного кода двоичная комбинация «01» в знаковых разрядах суммы указывает на положительное переполнение в сумматоре порядков. В этом случае в качестве результата сложения следует выдать код первого слагаемого. Если в знаковых разрядах суммы окажется двоичная комбинация «10», это указывает на отрицательное переполнение. Тогда в качестве результата сложения следует выдать код второго слагаемого.

После выравнивания порядков производится сложение мантисс операндов по правилам двоичной арифметики и округление результата (подробнее об операции округления можно прочитать в разделе 2.3 настоящего пособия). При этом возможна денормализация результата как влево (переполнение в сумматоре мантисс), так и вправо. Денормализованное на k разрядов вправо число оказывается по модулю меньше 0,5. В табл. 2.1 представлены изображения денормализованных чисел в прямом и инверсном кодах.

Таблица 2.1

	Положительное	Отрицательное
Прямой код	$0,\underbrace{00\ldots01}_{k}\ldots$	$1,\underbrace{00\ldots01}_{k}\ldots$
Инверсный код	$0,\underbrace{00\ldots01}_{k}\ldots$	$1,\underbrace{11\ldots10}_{k}\ldots$

При фиксации в сумматоре мантисс переполнения необходимо ликвидировать его путем сдвига содержимого сумматора вправо на один разряд, при этом в освобождающийся старший знаковый разряд заносится цифра, оказывающаяся в результате сдвига в младшем знаковом разряде, а порядок суммы увеличивается на 1. Если при сложении мантисс возникла денормализация результата на k разрядов вправо, то для ее устранения необходимо сдвинуть результат на k разрядов влево, а порядок результата следует уменьшить на k. При этом необходимо учитывать возможность

появления запрещенной в дополнительном коде комбинации «1,00…0» и равенство суммы нулю.

В том случае, когда денормализованное вправо отрицательное число представлено в обратном коде, освобождающиеся при сдвиге влево разряды мантиссы следует заполнять единицами, в других случаях – нулями.

Поскольку при устранении денормализации должно изменяться значение порядка, то может иметь место переполнение в сумматоре порядков. При положительном переполнении результат приравнивается к бесконечности, знак которой определяется знаком мантиссы. При отрицательном переполнении в сумматоре порядков результат считается бесконечно малой ненулевой величиной (если мантисса отлична от нуля).

Рассмотрим несколько примеров сложения чисел в дополнительном коде. Обозначим мантиссы операндов через $[m_X]$, порядки – через $[P_X]$.

Пример 1.

$A_{(10)} = 20{,}25;\ B_{(10)} = -0{,}75;\ C = A + B;$

$A_{(2)} = 10100{,}01;\ B_{(2)} = -00000{,}11;$

$[m_A]_{\text{д}} = 0{,}1010001;\ [P_A]^{\text{м}}_{\text{д}} = 00{,}101;$

$[m_B]_{\text{д}} = 1{,}0100000;\ [P_B]^{\text{м}}_{\text{д}} = 00{,}000.$

1. Для выравнивания порядков вычислим $\Delta P = P_A - P_B$.

$$\begin{aligned} \text{СМ} = [P_A]^{м}_{д} &= 00{,}101 \\ + \quad [-P_B]^{м}_{д} &= \underline{00{,}000} \\ [\Delta P]^{м}_{д} &= \ 00{,}101 \end{aligned}$$

Поскольку в сумматоре порядков образовалось положительное число, операнд A больше операнда B. Для выравнивания порядков сдвинем мантиссу второго операнда на ΔP разрядов вправо:

$[m_B]_{\text{д}} = 1{,}1111101;\ [P_B]^{\text{м}}_{\text{д}} = 00{,}101.$

2. Выполним сложение мантисс, при этом в сумматоре введем дополнительный младший разряд для округления.

$$\begin{aligned} \text{СМ} = [m_A]^{\text{м}}_{\text{д}} &= 00{,}1010001\,|\,0 \\ + \quad [m_B]^{\text{м}}_{\text{д}} &= \underline{11{,}1111101\,|\,0} \\ [m_{C=A+B}]^{\text{м}}_{\text{д}} &= 00{,}1001110\,|\,0 \end{aligned}$$

После округления до семи разрядов после запятой получим

$[m_C]_{\text{д}} = 0{,}1001110;\ [P_C]^{\text{м}}_{\text{д}} = 00{,}101.$

Таким образом, $C_{(2)} = 10011{,}10;\ C_{(10)} = 19{,}5.$

Пример 2.

$A_{(2)} = 0{,}01101011;\ B_{(2)} = -0{,}01000111;\ C = A + B;$

$[m_A]_{\text{д}} = 0{,}11010110,\ [P_A]^{\text{м}}_{\text{д}} = 11{,}111;$

$[m_B]_{\text{д}} = 1{,}01110010,\ [P_B]^{\text{м}}_{\text{д}} = 11{,}111.$

1. Для выравнивания порядков вычислим $\Delta P = P_A - P_B$.

$$\begin{aligned} \text{СМ} = [P_A]^{\text{м}}_{\text{д}} &= 11{,}111 \\ + \\ [-P_B]^{\text{м}}_{\text{д}} &= \underline{00{,}001} \\ [\Delta P]^{\text{м}}_{\text{д}} &= 00{,}000 \end{aligned}$$

Разность порядков равна 0, следовательно, уравнивание не требуется.

2. Выполним сложение мантисс.

$$\begin{aligned} \text{СМ} = [m_A]^{\text{м}}_{\text{д}} &= 00{,}11010110 \\ + \\ [m_B]^{\text{м}}_{\text{д}} &= 11{,}01110010 \\ [m_{C=A+B}]^{\text{м}}_{\text{д}} &= 00{,}01001000 \end{aligned}$$

После округления до восьми разрядов после запятой получим

$[m_C]_{\text{д}} = 0{,}01001000,\ [P_C]_{\text{д}} = 1{,}111.$

Мантисса суммы оказалась денормализованной на один разряд вправо. Для нормализации результата сдвинем мантиссу суммы на один разряд влево, а значение порядка суммы уменьшим на 1.

Окончательно $[m_C]_{\text{д}} = 0{,}10010000,\ [P_C]_{\text{д}} = 1{,}110.$

В прямом коде результат сложения запишется следующим образом:

$[m_C]_{\text{пр}} = 0{,}10010000,\ [P_C]_{\text{пр}} = 1{,}010.$

Большое количество примеров сложения двоичных чисел с плавающей запятой можно найти в [5].

2.2. Вычитание двоичных кодов

Схемотехническая реализация устройства для выполнения операции вычитания сложнее, чем для выполнения операции сложения, поэтому в цифровых вычислительных устройствах заменяют вычитание сложением

по формуле $(A - B) = A + (-B)$. Очевидно, что в этом случае также следует использовать инверсные коды, поскольку в прямых кодах сложение выполняется нетривиально.

Для выполнения операции вычитания над числами с плавающей запятой на первом этапе необходимо уравнять порядки операндов. При этом возможны те же сложности с переполнением сумматора порядков, что и при сложении.

Сложение мантисс выполняется следующим образом. Вначале у мантиссы вычитаемого необходимо изменить знак на противоположный. Поскольку вычитание удобнее выполнять в инверсных кодах, то для изменения знака вычитаемого на противоположный следует инвертировать все разряды вычитаемого (включая знак). При этом разряды вычитаемого будут содержать обратный код $(-B)$. Для получения дополнительного кода к младшему разряду инвертированной мантиссы вычитаемого следует прибавить 1.

При суммировании мантисс возможна денормализация результата вправо и переполнение сумматора мантисс в случае вычитания чисел с разными знаками. Действия по устранению переполнения и денормализации такие же, как и в случае выполнения операции сложения.

2.3. Выполнение операции округления чисел

2.3.1. Округление прямых кодов

Известны различные способы округления чисел в ЭВМ. Все они предполагают, что для округления числа до n разрядов необходимо вычислять $k = n + t$ разрядов округляемого числа, где $t = 1,\ 2, ..., m$.

Различают три вида округления:

1. Округление к нулю, когда число, т.е. правильная дробь, округляется до ближайшего к нулю машинного числа.

2. Округление от нуля, когда число округляется до машинного числа, лежащего дальше от нуля, чем округляемое число.

3. Округление по дополнению, когда округление производится до ближайшего машинного числа.

Рассмотрим несколько методов округления и дадим для них оценку погрешности.

Округление к нулю выполняется методом усечения, который заключается в отбрасывании t избыточных разрядов числа. Такой метод округления наиболее прост в реализации и не требует дополнительного

времени выполнения. В этом случае максимальная величина модуля погрешности (для двоичной правильной дроби) определяется выражением

$$\Delta_{\max} = 2^{-n}(1 - 2^{-t}).$$

Если появление чисел с разными знаками равновероятно и всевозможные двоичные комбинации дополнительных разрядов числа также равновероятны, математическое ожидание погрешности округления для метода усечения равно нулю. Если же появление чисел с разными знаками не равновероятно, математическое ожидание погрешности округления методом усечения запишется как

$$\overline{\Delta} = \frac{2^{-n}(1 - 2^{-t})}{2},$$

т.е. при действиях с числами одного знака будет иметь место систематическое накопление погрешности округления.

Округление от нуля предполагает анализ избыточных разрядов на равенство нулю, при этом избыточная часть отбрасывается и, если отбрасываемая часть не равна нулю, в младший разряд оставшейся части добавляется единица.

Основные характеристики этого метода округления полностью совпадают с характеристиками метода усечения. При этом округление от нуля требует использования дополнительного времени и схемотехнических ресурсов для анализа избыточных разрядов, а также для формирования распространения переносов при добавлении единицы в младший разряд округляемого числа, поэтому использование метода округления от нуля невыгодно.

Округление по дополнению представляет собой комбинацию двух предыдущих методов и заключается в том, что к младшему разряду сохраняемой части числа (двоичной правильной дроби) прибавляется некоторая величина $\alpha \in \{0,1\}$, определяемая по старшей цифре избыточной части числа. Если $(n+1)$-й разряд числа содержит единицу, то после отбрасывания избыточной части к младшему разряду сохраняемой части числа добавляется единица, т.е. 2^{-n}. В противном случае к младшему разряду сохраняемой части добавляется нуль. Для простоты реализации можно не выполнять анализ дополнительного $(n+1)$-го разряда, а всегда прибавлять единицу округления в $(n+1)$-й разряд. Если в нем содержится единица, то будет сформирован перенос в младший разряд сохраняемой части, что эквивалентно прибавлению 2^{-n}.

При таком методе округления значение максимума погрешности составляет

$$\Delta_{\max} = \frac{2^{-n}}{2},$$

а математическое ожидание погрешности округления (в случае равновероятного появления как чисел с разными знаками, так и нулей и единиц в отбрасываемой части числа) равно нулю. Если появление чисел разного знака не равновероятно, как часто бывает при выполнении различных вычислений, то математическое ожидание погрешности округления не равно нулю и поэтому возможны систематические ошибки округления. Однако эти ошибки меньше, чем при усечении, поэтому в дальнейшем будем использовать метод округления по дополнению, если не оговорено иное.

Метод округления по дополнению имеет ряд модификаций для устранения имеющихся недостатков. Так, например, систематические ошибки округления сводятся к нулю, если решение о коррекции сохраняемой части принимается на основании анализа не одного дополнительного разряда, а всей отсекаемой части. Такой метод носит название *усовершенствованного округления по дополнению*. Однако он не лишен недостатков, связанных с временными и аппаратурными затратами на анализ отбрасываемой части числа и распространение переноса, поэтому его применение целесообразно в вычислительных устройствах, требующих повышенной точности вычислений.

Для устранения сложностей с распространением переноса используется так называемый *метод упрощенного округления по дополнению*. Его суть заключается в том, что младший разряд сохраняемой части устанавливается равным единице в том случае, когда старший дополнительный разряд равен единице. В противном случае коррекция не выполняется. При этом, если в неокругленном числе младший разряд сохраняемой части равен единице, его значение не изменяется.

Если в неокругленном числе младший разряд сохраняемой части равен нулю, то максимальная погрешность округления при установке его в единицу может составить

$$\Delta^0_{\max} = \frac{2^{-n}(1-2^{-t})}{2}.$$

В противном случае максимальная погрешность составит

$$\Delta^{1}_{\max} = 2^{-n}(1 - 2^{-t}).$$

При равновероятном появлении нуля и единицы в младшем разряде сохраняемой части для знакопеременных чисел значение математического ожидания погрешности равно нулю.

Метод упрощенного округления по дополнению в общем случае имеет математическое ожидание погрешности округления, отличное от нуля. Однако в том случае, когда число избыточных разрядов $t = 1$, округление происходит без накопления систематических ошибок.

2.3.2. Округление инверсных кодов

Поскольку для положительных чисел прямой, обратный и дополнительный коды совпадают, остановимся подробно на особенностях округления отрицательных чисел.

В случае округления по дополнению обратного кода отрицательного числа необходимо из дополнительного разряда вычесть единицу, что эквивалентно прибавлению кода 1,11…10, где нуль соответствует дополнительному разряду. При этом циклический перенос выходящей из знакового разряда единицы должен охватывать и дополнительный разряд.

При округлении отрицательных чисел, представленных в дополнительном коде, следует учитывать, что прибавление единицы к дополнительному разряду в некоторых случаях может исказить код результата. Поэтому если справа от дополнительного разряда находится хотя бы одна единица, то в дополнительный разряд прибавляется единица округления. Если в дополнительном разряде находится единица и эта единица младшая в коде числа, то все разряды, начиная с дополнительного, просто отбрасываются.

2.4. Умножение двоичных кодов

При выполнении операции умножения чисел с плавающей запятой порядок произведения определяется как сумма порядков сомножителей. При этом возможно переполнение сумматора порядков, которое должно обрабатываться как переполнение, возникающее при нормализации мантиссы суммы двух чисел (см. п. 2.1 пособия). При умножении в

инверсных кодах знак произведения формируется автоматически. При умножении в прямом коде знак произведения определяется до начала умножения путем сложения по модулю 2 знаков сомножителей, после чего знаки обоих сомножителей устанавливаются равными нулю. В конце процедуры умножения результату следует присвоить знак произведения, определенный на первом этапе. Затем производится перемножение мантисс операндов аналогично умножению чисел с фиксированной запятой и выполняется округление результата. Следует помнить, что при умножении нормализованных чисел произведение может быть денормализовано только на один разряд вправо. Поэтому после перемножения мантисс следует проверить результат на денормализацию и при необходимости нормализовать его путем сдвига влево на один разряд, при этом порядок произведения должен быть уменьшен на единицу.

Операция умножения в ЭВМ может выполняться, начиная с младших или со старших разрядов множителя. Пусть сомножители представлены n-разрядными кодами, где n – число разрядов после запятой. Кроме того, будем рассматривать дополнительный $(n+1)$-й разряд в сумматоре мантисс, предназначенный для округления произведения.

2.4.1. Умножение прямых кодов чисел

Рассмотрим сначала умножение младшими разрядами вперед. Для получения произведения необходимо выполнить n циклов умножения по числу разрядов сомножителей. При этом каждый цикл состоит из двух тактов. В первом такте анализируется младший разряд регистра, осуществляющего хранение множителя. Если в младшем разряде регистра множителя содержится единица, то к текущему содержимому сумматора следует прибавить содержимое регистра множимого, в противном случае к сумматору добавляется ноль, что эквивалентно пропуску такта сложения. Во втором такте каждого цикла умножения сумматор мантисс и регистр множителя следует сдвинуть вправо на один разряд. При этом сдвиг сумматора должен быть логическим, т.е. в освобождающийся знаковый разряд заносится 0. Это обстоятельство становится очевидным, если в сумматоре содержится модифицированный код, поскольку в случае переполнения сумматора в первом такте старший знаковый разряд остается неизменным и равным 0. В результате на сумматоре будет сформировано произведение исходных сомножителей.

После выполнения n циклов умножения необходимо выполнить округление результата путем прибавления к сумматору единицы в дополнительный $(n+1)$-й разряд. Далее результат нормализуется, если это необходимо, и ему присваивается знак.

Пример.

$[A]_{\text{пр}} = 0{,}110001$, $[P_A]_{\text{пр}} = 1{,}011$;

$[B]_{\text{пр}} = 1{,}100101$, $[P_B]_{\text{пр}} = 0{,}101$;

$C = A \times B$.

Занесем операнды на регистры.

$\text{P}_1 = A = 0{,}110001$; $\text{P}_2 = B = 1{,}100101$.

1. Определение знака произведения.

$\text{Зн}_C = \text{Зн}_A \oplus \text{Зн}_B = 0 \oplus 1 = 1$.

$\text{P}_2 = \mid B \mid = 0{,}100101$.

2. Определение порядка произведения.

$P_C = P_A + P_B$.

Выполним сложение порядков в модифицированном дополнительном коде.

$$[P_A]^{\text{м}}_{\text{д}} = 11{,}101$$
$$+$$
$$[P_B]^{\text{м}}_{\text{д}} = \underline{00{,}101}$$
$$[P_C]^{\text{м}}_{\text{д}} = 00{,}010$$

3. Умножение мантисс.

$\text{СМ} = 0{,}0000000$ $\quad \text{P}_2 = 0{,}100101$ $\quad \text{P}_1 = 0{,}110001$

1. $+\text{P}_1 = \underline{0{,}1100010}$

$\text{СМ} = 0{,}1100010$

$\overrightarrow{\text{СМ}} = 0{,}0110001$ $\quad \vec{\text{P}}_2 = 0{,}010010$

2. $+0 = \underline{0{,}0000000}$

$\text{СМ} = 0{,}0110001$

$\overrightarrow{\text{СМ}} = 0{,}0011000$ $\quad \vec{\text{P}}_2 = 0{,}001001$

3. $+P_1 = 0,1100010$

$CM = 0,1111010$

(окончание примера на следующей странице)

$\overrightarrow{CM} = 0,0111101$ | $\vec{P}_2 = 0,000100$

4. $+0 = 0,0000000$

$CM = 0,0111101$

$\overrightarrow{CM} = 0,0011110$ | $\vec{P}_2 = 0,000010$

5. $+0 = 0,0000000$

$CM = 0,0011110$

$\overrightarrow{CM} = 0,0001111$ | $\vec{P}_2 = 0,000001$

6. $+P_1 = 0,1100010$

$CM = 0,1110001$

$\overrightarrow{CM} = 0,0111000$ | $\vec{P}_2 = 0,000000$

Окр. + $0,0000001$

$CM = 0,0111001$

Отбросим дополнительный седьмой разряд и выполним проверку правильности умножения в десятичной системе счисления.

$C = 0,011100_{(2)} = 1/4 + 1/8 + 1/16 = 0,4375_{(10)}$;

$A \times |B| = 0,110001_{(2)} \times 0,100101_{(2)} = 0,765625_{(10)} \times 0,578125_{(10)} \approx 0,4426_{(10)}$.

Таким образом, погрешность составляет 0,0051, т.е. меньше половины веса шестого разряда.

4. Ограничение результата шестью разрядами, нормализация и присвоение знака.

Нормализация: $\overleftarrow{CM} = 0,111000$; $[P_C]^{м}_{д} = 00,001$;

Окончательно $[C]_{пр} = 1,111000$; $[P_C]_{пр} = 0,001$.

Как было сказано ранее, умножение можно выполнять не только младшими, но и старшими разрядами вперед. Для этого следует выполнить *n* циклов умножения по числу разрядов множителя. Каждый цикл распадается на два такта. В первом такте регистр множимого

сдвигается на один разряд вправо, регистр множителя сдвигается на один разряд влево, сумматор в обоих тактах неподвижен. Во втором такте выполняется арифметическая операция сложения: если в знаковом разряде множителя содержится единица, то во втором такте к содержимому сумматора прибавляется множимое, в противном случае к сумматору прибавляется ноль. Такт суммирования очередного частичного произведения с нулем можно пропускать.

Следует помнить, что сумматор и регистр множимого должны иметь дополнительные разряды, так как в результате сдвига вправо множимое теряет значащие разряды. Количество дополнительных разрядов k должно удовлетворять следующему неравенству:

$$k \geq \log_2(n-k).$$

Все остальные этапы умножения такие же, как и в случае умножения младшими разрядами вперед.

Пример.

$[A]_{\text{пр}} = 0{,}110001$, $[P_A]_{\text{пр}} = 1{,}011$;

$[B]_{\text{пр}} = 1{,}100101$, $[P_B]_{\text{пр}} = 0{,}001$;

$C = A \times B$.

Занесем операнды на регистры, при этом для удобства учтем дополнительные разряды, число которых определяется подбором. Учитывая, что число основных разрядов $n = 6$, достаточно будет ввести $k = 2$ дополнительных разряда, поскольку $\log_2(6-2) = 2$.

$\text{Р}_1 = A = 0{,}110001\,|\,00$; $\text{Р}_2 = B = 1{,}100101$.

1. Определение знака произведения.

$\text{Зн}_C = \text{Зн}_A \oplus \text{Зн}_B = 0 \oplus 1 = 1$.

$\text{Р}_2 = |B| = 0{,}100101$.

2. Определение порядка произведения.

$P_C = P_A + P_B$.

Выполним сложение порядков в модифицированном дополнительном коде.

$$[P_A]^{\text{м}}_{\text{д}} = 11{,}101$$
$$+$$
$$[P_B]^{\text{м}}_{\text{д}} = \underline{00{,}001}$$
$$[P_C]^{\text{м}}_{\text{д}} = 11{,}110$$

В прямом коде порядок произведения может быть записан как $[P_C]_{\text{пр}} = 1{,}010$.

3. Умножение мантисс.

1.	СМ = 0,00000000	$\vec{P}_1 = 0{,}011000\|10$	$\overleftarrow{P}_2 = 1{,}001010$
	$+P_1 = \underline{0{,}01100010}$		
2.	СМ = 0,01100010	$\vec{P}_1 = 0{,}001100\|01$	$\overleftarrow{P}_2 = 0{,}010100$
	$+0 = \underline{0{,}00000000}$		
3.	СМ = 0,01100010	$\vec{P}_1 = 0{,}000110\|00$	$\overleftarrow{P}_2 = 0{,}101000$
	$+0 = \underline{0{,}00000000}$		
4.	СМ = 0,01100010	$\vec{P}_1 = 0{,}000011\|00$	$\overleftarrow{P}_2 = 1{,}010000$
	$+P_1 = \underline{0{,}00001100}$		
5.	СМ = 0,01101110	$\vec{P}_1 = 0{,}000001\|10$	$\overleftarrow{P}_2 = 0{,}100000$
	$+0 = \underline{0{,}00000000}$		
6.	СМ = 0,01101110	$\vec{P}_1 = 0{,}000000\|11$	$\overleftarrow{P}_2 = 1{,}000000$
	$+P_1 = \underline{0{,}00000011}$		
	СМ = 0,01110001		
Окр.	$+ \quad \underline{0{,}00000010}$		
	СМ = 0,01110011		

4. Ограничение результата шестью разрядами, нормализация и присвоение знака.

Нормализация: $\overleftarrow{\text{СМ}} = 0{,}111000$; $[P_C]^{\text{м}}_{\text{д}} = 11{,}101$;

Окончательно $[C]_{\text{пр}} = 1{,}111000$; $[P_C]_{\text{пр}} = 1{,}011$.

2.4.2. Ускоренное выполнение операции умножения

Рассмотрим два логических метода ускорения выполнения операции умножения младшими разрядами вперед: метод умножения с анализом последовательностей «0» и «1» в множителе (метод «преобразования» множителя, ускоренное умножение № 1) и метод расшифровки двух разрядов множителя (ускоренное умножение № 2). Первый метод позволяет уменьшить число арифметических операций на сумматоре, необходимых для формирования произведения (при этом число циклов умножения не сокращается). Второй метод позволяет сократить число

циклов умножения вдвое, поскольку в каждом цикле множимое умножается сразу на два разряда множителя.

Метод «преобразования» множителя заключается в том, что в некоторых случаях число содержащихся в множителе единиц можно уменьшить, заменив их нулями, что приводит к пропуску такта суммирования, поскольку при умножении на 0 содержимое сумматора не изменяется. Следует понимать, что рассматриваемое преобразование множителя выполняется логически, т.е. реальное содержимое регистра множителя остается неизменным, но меняется процедура его обработки. Поэтому слово «преобразование» в названии метода взято в кавычки.

Пусть в множителе три или более идущих друг за другом разрядов $a_m, a_{m-1}, ..., a_k$ равны единице. В соответствии с элементарной двоичной арифметикой такая последовательность может быть представлена как разность двух чисел: $2^{-(m+1)} - 2^{-k}$. Покажем это на примере множителя $B = 0{,}011100$:

$$\begin{array}{r} 0{,}100000 \\ -\underline{0{,}000100} \\ 0{,}011100. \end{array}$$

Таким образом, арифметическую операцию сложения при умножении на множитель подобного вида можно заменить операцией вычитания множимого из суммы частичных произведений. После окончания последовательности подряд следующих единиц необходимо снова выполнять операцию сложения.

Отдельно стоящие нули или единицы могут увеличить число единиц в преобразованном множителе. Рассмотрим, например, фрагмент множителя, состоящий из нулей и единицы в *i*-м разряде, и его преобразованный вид:

$$\begin{array}{l} 0001000 \\ 001\bar{1}000. \end{array}$$

В этом случае умножение множимого A на преобразованный множитель должно выполняться в соответствии с выражением

$$A(2^{i+1} - 2^i) = A \cdot 2^i.$$

Если фрагмент множителя с нулем в i-м разряде и его преобразование имеют вид

$$01110111$$
$$100\bar{1}100\bar{1},$$

то умножение множимого A на преобразованный множитель должно выполняться в соответствии с выражением

$$A(-2^{i+1}+2^{i})=-A\cdot 2^{i}.$$

Для определения отдельно стоящих нулей и единиц будем анализировать одновременно два разряда множителя, а для выявления последовательностей идущих подряд нулей или единиц в множителе введем в устройство специальный запоминающий элемент, называемый *триггером*, имеющий два устойчивых состояния – «0» и «1». Тогда арифметические операции на сумматоре и установки триггера сдвига определятся в соответствии с табл. 2.2.

Таблица 2.2

Разряды множителя		Состояние триггера Тг	Действия
$n-1$	n		
0	0	0	–
0	1	0	СМ + A
1	0	0	–
1	1	0	СМ – A, Тг = 1
0	0	1	СМ + A, Тг = 0
0	1	1	–
1	0	1	СМ – A
1	1	1	–

Следует отметить, что перед началом выполнения умножения триггер должен быть установлен в состояние «0». По окончании умножения триггер также должен установиться в «0», поэтому в некоторых случаях требуется выполнение дополнительного цикла умножения (точнее – полуцикла, поскольку такт сдвига не выполняется) для обработки последней единицы в множителе, который может быть совмещен с округлением результата.

При умножении мантисс следует использовать модифицированный код, поскольку в некоторых случаях может возникать переполнение разрядной сетки.

Определение знака и порядка произведения, его нормализация выполняются так же, как при неускоренном умножении.

Пример.

Умножим числа с фиксированной запятой.

$[A]_{пр} = 0{,}11001001$;

$[B]_{пр} = 0{,}11101011$;

$C = A \times B$.

1. Определение знака произведения.

$Зн_C = Зн_A \oplus Зн_B = 0 \oplus 0$.

2. Умножение мантисс.

$P_1 = 00{,}11001001$;

	СМ = 00,00000000	0	$P_2 = 00{,}11101011$ Тг = 0
1.	$-P_1 =$ 11,00110111	0	
	СМ = 11,00110111	0	Тг = 1
	$\overrightarrow{СМ} = 11{,}10011011$	1	$\vec{P}_2 = 00{,}01110101$
2.	$\overrightarrow{СМ} = 11{,}11001101$	1	$\vec{P}_2 = 00{,}00111010$
3.	$-P_1 =$ 11,00110111	0	
	СМ = 11,00000100	1	
	$\overrightarrow{СМ} = 11{,}10000010$	0	$\vec{P}_2 = 00{,}00011101$
4.	$\overrightarrow{СМ} = 11{,}11000001$	0	$\vec{P}_2 = 00{,}00001110$
5.	$-P_1 = 11{,}00110111$	0	
	СМ = 10,11111000	0	
	$\overrightarrow{СМ} = 11{,}01111100$	0	$\vec{P}_2 = 00{,}00000111$
6.	$\overrightarrow{СМ} = 11{,}10111110$	0	$\vec{P}_2 = 00{,}00000011$
7.	$\overrightarrow{СМ} = 11{,}11011111$	0	$\vec{P}_2 = 00{,}00000001$
8.	$\overrightarrow{СМ} = 11{,}11101111$	1	$\vec{P}_2 = 00{,}00000000$
9.	$+P_1 =$ 00,11001001	0	Тг = 0
	СМ = 00,10111000	1	При совмещении операции округления с
Окр. +	00,00000000	1	дополнительным 9-м циклом регистр мно-
	СМ = 00,10111001	0	жимого имеет вид $P_1 = 00{,}11001001\ 1$.

3. Ограничение результата восемью разрядами и присвоение знака.

Окончательно $[C]_{пр} = 0{,}10111001$.

Рассмотрим второй ускоренный метод умножения, заключающийся в умножении на два разряда множителя одновременно. При этом возможны четыре двоичные комбинации младших разрядов:

1. Комбинация «00» означает умножение на 0, т.е. пропуск такта суммирования.

2. Комбинация «01» означает умножение на 1, т.е. прибавление множимого к предыдущему частичному произведению.

3. Комбинация «10» означает умножение на 2, т.е. прибавление к предыдущему частичному произведению удвоенного множимого (сдвинутого на один разряд влево).

4. Комбинация «11» означает умножение на 3. В этом случае вместо утроения множимого комбинацию «11» заменяют комбинацией «$100-01$», т.е. «четыре минус один». Это означает, что из текущего частичного произведения следует вычесть множимое и сформировать единицу переноса в следующую пару разрядов множителя.

Таким образом, умножение выполняется за $n/2$ циклов, при этом каждый цикл распадается на два такта. В первом такте выполняется арифметическая операция, определяемая комбинацией двух младших разрядов множителя и переносом из предыдущей пары разрядов (табл. 2.3). Во втором такте содержимое сумматора и регистр множителя сдвигаются на два разряда вправо. Для запоминания переноса в следующую пару разрядов множителя можно использовать дополнительный триггер (Тг). При формировании переноса из двух старших разрядов множителя необходимо выполнить дополнительный такт сложения содержимого сумматора и множимого (полуцикл умножения).

Таблица 2.3

Разряды множителя		Перенос из предыдущей пары	Перенос в следующую пару	Действие
$n-1$	n			
0	0	0	0	–
0	1	0	0	$+A$
1	0	0	0	$+2A$
1	1	0	1	$-A$
0	0	1	0	$+A$

0	1	1	0	$+2A$
1	0	1	1	$-A$
1	1	1	1	–

Поскольку в некоторых случаях возможно переполнение разрядной сетки с искажением второго знакового разряда, следует использовать код с тремя знаковыми разрядами.

Определение знака и порядка произведения, его нормализация выполняются так же, как при неускоренном умножении.

Пример.

Умножим числа с фиксированной запятой.

$[A]_{пр} = 0{,}11101011;$

$[B]_{пр} = 0{,}11001001;$

$C = A \times B.$

1. Определение знака произведения.

$Зн_C = Зн_A \oplus Зн_B = 0 \oplus 0.$

2. Умножение мантисс.

```
P1 = 000,11101011;
СМ = 000,00000000 | 0      P2 = 000,11001001 Тг = 0
1. +P1 = 000,11101011 | 0
СМ = 000,11101011 | 0
2→СМ = 000,00111010 | 1     2→P2 = 000,00110010
2. +←P1 = 001,11010110 | 0
СМ = 010,00010000 | 1
2→СМ = 000,10000100 | 0     2→P2 = 000,00001100
3. 2→СМ = 000,00100001 | 0  2→P2 = 000,00000011
4. −P1 = 111,00010101 | 0
СМ = 111,00110110 | 0       Тг = 1
2→СМ = 111,11001101 | 1     2→P2 = 000,00000000
5. +P1 = 000,11101011 | 0
СМ = 000,10111000 | 1       Тг = 0;
Окр. +   000,00000000 | 1   Округление результата может быть
СМ = 000,10111001 0         совмещено с 5-м циклом в один такт.
```

3. Ограничение результата восемью разрядами и присвоение знака.

Окончательно $[C]_{\text{пр}} = 0{,}10111001$.

Ясно, что умножение рассмотренным методом может выполняться только начиная с младших разрядов множителя.

2.4.3. Умножение инверсных кодов чисел

Выполнение умножения инверсных кодов чисел аналогично умножению прямых кодов, но с некоторыми отличиями. Во-первых, при использовании модифицированного сдвига (сохраняющего знак сумматора или регистра множимого) знак произведения формируется автоматически. Во-вторых, при отрицательном множителе необходима коррекция произведения.

Коррекция произведения выполняется следующим образом. Пусть отрицательный множитель B задан в инверсном коде. Тогда в его цифровых разрядах (разрядах после запятой) содержатся величины $(1-B)$ или $(1-B-2^{-n})$ для дополнительного и обратного кода соответственно. В результате выполнения n циклов умножения на сумматоре в случае дополнительного кода будет сформировано произведение $C = A(1-B) = A - AB$, а в случае обратного кода $C = A(1-B-2^{-n}) = A - A\cdot 2^{-n} - AB$. В обоих случаях истинное произведение представляет собой $(-AB)$.

Для выполнения коррекции при использовании обратного кода к содержимому сумматора после умножения необходимо прибавить сначала $(-A)$, а в следующем такте $(A\cdot 2^{-n})$. При умножении младшими разрядами вперед поправка $(A\cdot 2^{-n})$ может быть заменена прибавлением величины $(+A)$ перед выполнением умножения, а поправка $(-A)$ делается после умножения. При умножении обратных кодов чисел старшими разрядами вперед поправка $(A\cdot 2^{-n})$ после выполнения n циклов умножения автоматически образуется в регистре множимого и ее необходимо прибавить к сумматору, а поправка $(-A)$ делается перед началом выполнения умножения, поэтому в рассматриваемом случае для коррекции произведения перед выполнением умножения из сумматора вычитается содержимое регистра множимого (прибавляется $(-A)$), а после выполнения n циклов умножения содержимое регистра множимого снова прибавляется к содержимому сумматора, но без изменения знака.

Коррекция произведения дополнительных кодов чисел при умножении младшими разрядами вперед (неподвижное множимое) осуществляется путем вычитания содержимого регистра множимого из сумматора после выполнения n циклов умножения. При умножении старшими разрядами вперед необходимо вычитать из содержимого сумматора множимое (прибавлять $(-A)$) до начала выполнения умножения.

Сказанное можно компактно представить в виде табл. 2.4.

Таблица 2.4

В обратном коде		В дополнительном коде	
мл. разрядами	ст. разрядами	мл. разрядами	ст. разрядами
1. $+A$	1. $-A$	1. Умножение	1. $-A$
2. Умножение	2. Умножение	2. $-A$	2. Умножение
3. $-A$	3. $+A \cdot 2^{-n}$ (сдвинутое на n разрядов содержимое регистра множимого)		

Пример.

Умножение младшими разрядами вперед.

$[A]_д = 0{,}110001$;

$[B]_д = 1{,}011011$;

$C = A \times B$.

```
СМ = 00,000000 | 0        P2 = 11,011011 P1 = 0,110001
1. +P1 = 00,110001 | 0
СМ = 00,110001 | 0
СМ→ = 00,011000 | 1       P2→ = 11,101101
2. +P1 = 00,110001 | 0
СМ = 01,001001 | 1
СМ→ = 00,100100 | 1       P2→ = 11,110110
3. СМ→ = 00,010010 | 0     P2→ = 11,111011
4. +P1 = 00,110001 | 0
СМ = 01,000011 | 0
СМ→ = 00,100001 | 1       P2→ = 11,111101
```

5. $+P_1 = 00,110001\ 0$

$CM = 01,010010\ 1$

$\overrightarrow{CM} = 00,101001\ 0$ $\qquad \vec{P}_2 = 11,111110$

6. $\overrightarrow{CM} = 00,010100\ 1$ $\qquad \vec{P}_2 = 11,111111$

7. $-P_1 = 11,001111\ 0$

$CM = 11,100011\ 1$

Окр. $+ \quad 00,000000\ 1$

$CM = 11,100100\ 0$

Таким образом, ограничивая результат шестью разрядами, получим:

$$[C]_д = 1,100100.$$

Это число денормализовано, поэтому в случае умножения с плавающей запятой необходимо выполнить нормализацию.

В прямом коде результат умножения запишется следующим образом: $[C]_{пр} = 1,011100.$

2.5. Деление двоичных кодов

Рассмотрим выполнение операции деления над числами, представленными в прямом коде.

Арифметическая операция деления выполняется в два этапа. На первом этапе, как при умножении, формируется знак частного путем сложения по модулю 2 знаков операндов. Одновременно с этим определяется порядок частного (при делении чисел в машинах с плавающей запятой) как разность порядка делимого и порядка делителя. При этом возможно переполнение сумматора порядков, которое следует интерпретировать так же, как и при выполнении нормализации результата сложения двух чисел. На втором этапе выполняется деление модуля мантиссы делимого на модуль мантиссы делителя, которое в общем виде представляет собой циклическое вычитание делителя из удвоенного очередного частичного остатка. Для определения *n* цифр частного необходимо выполнить *n* циклов деления, при этом в каждом цикле определяется очередная цифра частного в зависимости от знака очередного частичного остатка.

Деление в машинах с фиксированной запятой возможно только в том случае, когда модуль делимого меньше модуля делителя. В противном случае произойдет переполнение разрядной сетки. Для предотвращения

переполнения перед началом выполнения операции следует произвести проверку на возможность деления, заключающуюся в вычитании делителя из делимого. Если результат меньше нуля, то делимое меньше делителя и выполнение операции деления возможно. Если результат больше нуля, то делимое больше делителя и выполнение операции деления следует остановить.

При делении чисел с плавающей запятой в случае неудовлетворительного результата выполнения проверки на деление процесс можно не останавливать, а сформировать первую цифру частного, равную единице, и продолжать деление в соответствии с выбранным алгоритмом. В этом случае после выполнения n циклов деления в регистре частного будет сформировано $(n+1)$ цифр частного, старшая из которых окажется в знаковом разряде. Таким образом, произойдет единственно возможная при делении денормализация результата на один разряд влево. Для устранения денормализации следует сдвинуть содержимое регистра частного на один разряд вправо, а порядок частного увеличить на единицу.

2.5.1. Деление прямых кодов чисел

Рассмотрим два базовых метода выполнения операции деления: деление с восстановлением остатка и деление без восстановления остатка.

При делении с восстановлением остатка каждый цикл распадается на три такта. В первом такте очередной частичный остаток и регистр частного сдвигаются на один разряд влево. Во втором такте из сдвинутого частичного остатка вычитается делитель. В третьем такте в младший разряд регистра частного заносится очередная цифра, определяемая инверсией знакового разряда сумматора, и в случае отрицательного сумматора выполняется восстановление остатка путем прибавления к содержимому сумматора положительного делителя. Выполнение проверки на возможность деления можно рассматривать как нулевой цикл с пропущенным первым тактом, т.е. в случае отрицательного сумматора в младший разряд регистра частного заносится нуль, а делимое подлежит восстановлению.

Пример.
$[A]_{\text{пр}} = 0{,}110001$, $[P_A]_{\text{пр}} = 1{,}011$;
$[B]_{\text{пр}} = 1{,}100101$, $[P_B]_{\text{пр}} = 0{,}001$;
$C = A/B$.

1. Определение знака частного.

$Зн_C = Зн_A \oplus Зн_B = 0 \oplus 1 = 1.$

$P_2 = |B| = 0{,}100101.$

2. Определение порядка частного.

$P_C = P_A - P_B.$

$$[P_A]^{м}_{д} = 11{,}101$$
$$+$$
$$[P_B]^{м}_{д} = \underline{11{,}111}$$
$$[P_C]^{м}_{д} = 11{,}100$$

В прямом коде порядок частного может быть записан как $[P_C]_{пр} = 1{,}100$.

3. Деление мантисс.

	Частное	Делитель
	$СМ = 0{,}110001$	$P_1 = 0{,}000000$	$P_2 = 0{,}100101$
0.	$-P_2 = \underline{1{,}011011}$		
	$СМ = 0{,}001100 > 0$, переполнение	$P_1 = 0{,}000001$
1.	$\overleftarrow{СМ} = 0{,}011000$	$\overleftarrow{P_1} = 0{,}000010$
	$-P_2 = \underline{1{,}011011}$	
	$СМ = 1{,}110011$	
	$+P_2 = \underline{0{,}100101}$	$P_1 = 0{,}000010$
	$СМ = 0{,}011000$	
2.	$\overleftarrow{СМ} = 0{,}110000$	$\overleftarrow{P_1} = 0{,}000100$
	$-P_2 = \underline{1{,}011011}$	
	$СМ = 0{,}001011$	
		$P_1 = 0{,}000101$
3.	$\overleftarrow{СМ} = 0{,}010110$	$\overleftarrow{P_1} = 0{,}001010$
	$-P_2 = \underline{1{,}011011}$	
	$СМ = 1{,}110001$	
	$+P_2 = \underline{0{,}100101}$	$P_1 = 0{,}001010$
	$СМ = 0{,}010110$	
4.	$\overleftarrow{СМ} = 0{,}101100$	$\overleftarrow{P_1} = 0{,}010100$
	$-P_2 = \underline{1{,}011011}$	
	$СМ = 0{,}000111$	
		$P_1 = 0{,}010101$
5.	$\overleftarrow{СМ} = 0{,}001110$	$\overleftarrow{P_1} = 0{,}101010$
	$-P_2 = \underline{1{,}011011}$	
	$СМ = 1{,}101001$	
	$+P_2 = \underline{0{,}100101}$	$P_1 = 0{,}101010$

$$
\begin{array}{lll}
 & \text{СМ}=0,001110 & \\
6. & \overleftarrow{\text{СМ}}=0,011100 & \overleftarrow{P_1}=1,010100 \\
 & -P_2=\underline{1,011011} & \\
 & \text{СМ}=1,110111 & \\
 & +P_2=\underline{0,100101} & P_1=1,010100 \\
 & \text{СМ}=0,011100 &
\end{array}
$$

Таким образом, в результате выполнения шести основных циклов деления получены семь цифр частного. Единица в знаковом разряде регистра частного указывает на денормализацию влево. После нормализации и последующего присвоения знака результат может быть записан как

$$[C]_{\text{пр}} = 1,101010,\ [P_C]_{\text{пр}} = 1,011.$$

В ЭВМ деление чисел с восстановлением остатка практически не применяется, поскольку восстановление остатка требует дополнительных временных затрат, а если оно не требуется, то цикл деления получается неравномерным.

Рассмотрим еще один метод деления чисел, при котором в случае появления отрицательного частичного остатка восстановление не производится, а происходит переход к следующему циклу деления, в котором отрицательный частичный остаток удваивается и к нему прибавляется положительный делитель. При этом цифры частного формируются так же, как и в предыдущем методе, т.е. путем инвертирования знакового разряда. При выполнении проверки на деление в нулевом цикле отрицательный сумматор можно не восстанавливать.

Таким образом, каждый из n циклов деления состоит из двух тактов. В первом такте очередной частичный остаток и регистр частного сдвигаются на один разряд влево. Во втором такте выполняется арифметическая операция, определяемая знаком сумматора или цифрой частного, определенной в предыдущем цикле. Если сумматор больше нуля (предыдущая цифра частного – единица), то из удвоенного (сдвинутого влево) очередного частичного остатка вычитается делитель. В противном случае к удвоенному очередному частичному остатку следует прибавить положительный делитель.

При использовании немодифицированного кода анализ знака сумматора необходимо делать до начала очередного цикла деления, так как в некоторых случаях при удвоении остатка его знак может быть потерян.

В этом случае алгоритм распадается на две части (две ветви) со сдвигом в каждой, т.е. анализ знака сумматора выполняется до сдвига.

При использовании модифицированного кода необходимость в разделении алгоритма на две ветви или в анализе предыдущей цифры частного отпадает, так как тип арифметической операции может быть определен по старшему знаковому разряду сумматора.

Пример.

$[A]_{\text{пр}} = 0{,}110001$, $[P_A]_{\text{пр}} = 0{,}100$;
$[B]_{\text{пр}} = 1{,}100101$, $[P_B]_{\text{пр}} = 0{,}011$;
$C = A/B$.

1. Определение знака частного.

$\text{Зн}_C = \text{Зн}_A \oplus \text{Зн}_B = 0 \oplus 1 = 1$.

$P_2 = |B| = 0{,}100101$.

2. Определение порядка частного.

$$[P_A]^{\text{м}}_{\text{д}} = 00{,}100$$
$$+$$
$$[P_B]^{\text{м}}_{\text{д}} = \underline{11{,}101}$$
$$[P_C]^{\text{м}}_{\text{д}} = 00{,}001$$

3. Деление мантисс.

		Частное	Делитель
	$\text{СМ} = 00{,}110001$	$P_1 = 00{,}000000$	$P_2 = 00{,}100101$
0.	$-P_2 = \underline{11{,}011011}$		
	$\text{СМ} = 00{,}001100 > 0$, переполнение		
		$P_1 = 00{,}000001$	
1.	$\overleftarrow{\text{СМ}} = 00{,}011000$	$\overleftarrow{P_1} = 00{,}000010$	
	$-P_2 = \underline{11{,}011011}$		
	$\text{СМ} = 11{,}110011$	$P_1 = 00{,}000010$	
2.	$\overleftarrow{\text{СМ}} = 11{,}100110$	$\overleftarrow{P_1} = 00{,}000100$	
	$+P_2 = \underline{00{,}100101}$		
	$\text{СМ} = 00{,}001011$	$P_1 = 00{,}000101$	
3.	$\overleftarrow{\text{СМ}} = 00{,}010110$	$\overleftarrow{P_1} = 00{,}001010$	
	$-P_2 = \underline{11{,}011011}$		
	$\text{СМ} = 11{,}110001$	$P_1 = 00{,}001010$	

4. $\overleftarrow{СМ} = 11,100010$ $\quad \overleftarrow{Р_1} = 00,010100$

$+Р_2 = \underline{00,100101}$

$СМ = 00,000111$ $\quad Р_1 = 00,010101$

5. $\overleftarrow{СМ} = 00,001110$ $\quad \overleftarrow{Р_1} = 00,101010$

$-Р_2 = \underline{11,011011}$

$СМ = 11,101001$ $\quad Р_1 = 00,101010$

6. $\overleftarrow{СМ} = 11,010010$ $\quad \overleftarrow{Р_1} = 01,010100$

$+Р_2 = \underline{00,100101}$

$СМ = 11,110111$ $\quad Р_1 = 01,010100$

После устранения денормализации и присвоения знака результат может быть записан в следующем виде:

$$[C]_{пр} = 1,101010,\ [P_C]_{пр} = 0,010.$$

2.5.2. Ускоренное выполнение операции деления

В быстродействующих цифровых вычислительных устройствах применение вышеописанных алгоритмов деления нецелесообразно, поскольку для формирования *n*-разрядного частного необходимо выполнить не менее *n* арифметических операций (сложения или вычитания). Рассмотрим два логических метода ускорения выполнения операции деления, позволяющих сократить число выполняемых арифметических операций. Сразу оговоримся, что число циклов деления при использовании этих методов не уменьшается. Обязательным условием правильного выполнения приведенных алгоритмов является нормализованный делитель.

Первый ускоренный метод деления с анализом одного разряда основан на следующих соображениях. Будем полагать, что очередной частичный остаток $a_i > 0$, а нормализованный делитель $B > 2a_i$, т.е. делитель больше остатка, сдвинутого влево на один разряд. Очевидно, что в следующем цикле деления из удвоенного частичного остатка следует вычесть делитель. В результате на сумматоре сформируется число $(2a_i - B) < 0$, что приводит к формированию очередной цифры частного, равной нулю.

В следующем цикле деления очередной отрицательный частичный остаток удвоится, к нему прибавится делитель *B*, и на сумматоре сформируется

число $2(2a_i - B) + B = 4a_i - B$. Аналогичный остаток может быть сформирован на сумматоре в случае пропуска операции вычитания делителя в $(i+1)$-м цикле. Действительно, в результате двух сдвигов влево на один разряд остаток a_i будет умножен на 4, а поскольку он положительный, то в $(i+2)$-м цикле из него следует вычесть делитель. Для случая $a_i < 0$ рассуждения аналогичны.

Таким образом, если удвоенный частичный остаток содержит в знаковом и старшем после запятой разрядах двоичную комбинацию «0,0», то во втором такте арифметическая операция не выполняется и очередная цифра частного формируется равной знаковому разряду сумматора, т.е. нулю. Если удвоенный частичный остаток содержит в знаковом и старшем после запятой разрядах двоичную комбинацию «1,1», то во втором такте арифметическая операция также не выполняется и очередная цифра частного равна единице. В случае несовпадения разрядов справа и слева от запятой требуется выполнение арифметической операции.

Пример.

$[A]_{\text{пр}} = 0{,}101001$;
$[B]_{\text{пр}} = 0{,}110011$;
$C = A/B$.

1. Определение знака частного.

$\text{Зн}_C = \text{Зн}_A \oplus \text{Зн}_B = 0 \oplus 0 = 0$.

2. Деление мантисс.

		Частное	Делитель
	$\text{СМ} = 0{,}101001$	$\text{P}_1 = 0{,}000000$	$\text{P}_2 = 0{,}110011$
0.	$-\text{P}_2 = \underline{1{,}001101}$		
	$\text{СМ} = 1{,}110110 < 0$, деление без переполнения		
1.	$\overleftarrow{\text{СМ}} = 1{,}101100$	$\overleftarrow{\text{P}_1} = 0{,}000000$	
		$\text{P}_1 = 0{,}000001$	
2.	$\overleftarrow{\text{СМ}} = 1{,}011000$	$\overleftarrow{\text{P}_1} = 0{,}000010$	
	$+\text{P}_2 = \underline{0{,}110011}$		
	$\text{СМ} = 0{,}001011$	$\text{P}_1 = 0{,}000011$	
3.	$\overleftarrow{\text{СМ}} = 0{,}010110$	$\overleftarrow{\text{P}_1} = 0{,}000110$	
		$\text{P}_1 = 0{,}000110$	
4.	$\overleftarrow{\text{СМ}} = 0{,}101100$	$\overleftarrow{\text{P}_1} = 0{,}001100$	

$-P_2 = 1,001101$

$СМ = 1,111001$ $\quad P_1 = 0,001100$

5. $\overleftarrow{СМ} = 1,110010$ $\quad \overleftarrow{P_1} = 0,011000$

$P_1 = 0,011001$

6. $\overleftarrow{СМ} = 1,100100$ $\quad \overleftarrow{P_1} = 0,110010$

$P_1 = 0,110011$

Таким образом, частное $C = 0,110011$.

Описанный метод позволяет ускорить выполнение операции деления только в тех циклах, в которых остаток денормализован, а в остальных циклах деление происходит по неускоренному алгоритму. Поэтому в худшем случае, при постоянном формировании нормализованного остатка, этот метод не дает прироста производительности.

Рассмотрим еще один метод ускорения выполнения операции деления, основанный на анализе двух разрядов очередного частичного остатка, при котором арифметическая операция может быть пропущена, если абсолютное значение удвоенного частичного остатка меньше делителя. В противном случае требуется выполнение арифметической операции, определяемой знаком сумматора. Поскольку в результате удвоения очередного частичного остатка может произойти переполнение сумматора, необходимо использовать модифицированный код. При переполнении сумматора содержащееся в нем число по модулю больше единицы, т.е. заведомо больше делителя, поэтому также следует выполнять арифметическую операцию, определяемую старшим знаковым разрядом сумматора.

В табл. 2.5 приведены различные сочетания пар старших разрядов регистра делителя и сумматора.

Таблица 2.5

Старшие разряды сумматора	Старшие разряды модуля делителя	
	0,10	0,11
0,00 1,11	Арифм. операция не выполняется	Арифм. операция не выполняется
0,01 1,10	Арифм. операция не выполняется	Арифм. операция не выполняется
0,10 1,01	Требуется арифм. операция	Арифм. операция не выполняется

0,11 1,00	Требуется арифм. операция	Требуется арифм. операция

Пример.

Разделим мантиссы чисел из предыдущего примера. Частное, формируемое в первом регистре, в обоих случаях должно совпадать.

		Частное	Делитель
	$СМ = 00,101001$	$P_1 = 00,000000$	$P_2 = 00,110011$
0.	$-P_2 = \underline{11,001101}$		
	$СМ = 11,110110 < 0$, деление без переполнения		
1.	$\overleftarrow{СМ} = 11,101100$	$\overleftarrow{P_1} = 00,000000$	
		$P_1 = 00,000001$	
2.	$\overleftarrow{СМ} = 11,011000$	$\overleftarrow{P_1} = 00,000010$	
		$P_1 = 00,000011$	
3.	$\overleftarrow{СМ} = 10,110000$	$\overleftarrow{P_1} = 00,000110$	
	$+P_2 = \underline{00,110011}$		
	$СМ = 11,100011$	$P_1 = 00,000110$	

(окончание примера на следующей странице)

4.	$\overleftarrow{СМ} = 11,000110$	$\overleftarrow{P_1} = 00,001100$	
	$+P_2 = \underline{00,110011}$		
	$СМ = 11,111001$	$P_1 = 00,001100$	
5.	$\overleftarrow{СМ} = 11,110010$	$\overleftarrow{P_1} = 00,011000$	
		$P_1 = 00,011001$	
6.	$\overleftarrow{СМ} = 11,100100$	$\overleftarrow{P_1} = 00,110010$	
		$P_1 = 00,110011$	

2.5.3. Деление дополнительных кодов чисел

Деление чисел, представленных в дополнительном коде, выполняется в три этапа.

На первом этапе вычисляется знак частного как результат сложения по модулю 2 знаков операндов.

На втором этапе производится операция проверки на возможность деления (на денормализацию частного влево) путем алгебраического сложения делимого с делителем, которому приписывается знак, противоположный знаку делимого. При этом если делимое

положительное, то цифра частного, оказывающаяся в знаковом разряде, определяется инверсией знакового разряда сумматора. Если делимое отрицательное, то текущая цифра частного равна прямому значению знакового разряда сумматора.

На третьем этапе выполняются n циклов деления по обычному алгоритму с некоторыми отличиями, заключающимися в том, что при отрицательном делителе цифры частного формируются равными прямому значению знакового разряда очередного частичного остатка. При этом частное получается в обратном коде. Кроме n основных циклов, обязательно вычисление $(n+1)$-й цифры частного в дополнительном цикле, поскольку в дальнейшем нормализованное при необходимости частное поступает на сумматор, где к $(n+1)$-му разряду добавляется единица для округления и перевода частного из обратного кода в дополнительный.

Читателю предлагается самостоятельно разобрать примеры, взяв в качестве операндов мантиссы из предыдущих примеров. При этом следует помнить, что отрицательные операнды должны быть представлены в дополнительном коде.

Пример.

$[A]_д = 0{,}101;$

$[B]_д = 1{,}010;$

$C = A/B.$

$Зн_C = Зн_A \oplus Зн_B = 0 \oplus 1 = 1.$

```
СМ=00,101 P1=00,0000 P2=11,010
0. -P2=11,010
СМ=11,111
```

Поскольку делимое положительное, переполнения разрядной сетки не произойдет (в младший разряд регистра частного заносится инверсия знака сумматора).

Так как делитель отрицательный, цифры частного будут определяться прямыми значениями знакового разряда сумматора.

```
      СМ=11,111
1.    ←СМ=11,110        ←P1=00,0000
     +P2=00,110
      СМ=00,100         P1=00,0000
2.    ←СМ=01,000        ←P1=00,0000
     −P2=11,010
      СМ=00,010         P1=00,0000
3.    ←СМ=00,100        ←P1=00,0000
     −P2=11,010
      СМ=11,110         P1=00,0001
4.    ←СМ=11,100        ←P1=00,0010
     +P2=00,110
      СМ=00,010         P1=00,0010
```

Округление и преобразование в дополнительный код.

```
5.    СМ=11,0010        ←P1=00,0010;
     +    00,0001
      СМ=11,0011
```

Таким образом, $[C]_д = 1{,}001.$

В прямом коде результат запишется как $[C]_{пр} = 1{,}111.$

2.6. Извлечение квадратного корня

Операция извлечения квадратного корня включается в совокупность арифметических операций ЭВМ в том случае, когда она составляет не менее 2 % от общего числа операций или является составной частью специализированных алгоритмов, выполняемых с высоким быстродействием. В противном случае квадратный корень можно извлекать итеративно, например, по формуле Герона.

Процедура извлечения квадратного корня сводится к процедуре деления, при этом делитель в каждом цикле формируется из полученных на предыдущем этапе цифр частного путем приписывания к частному некоторой двоичной комбинации.

В общем случае извлечение квадратного корня из числа, представленного с плавающей запятой, осуществляется в два этапа. На первом этапе формируется порядок результата, а на втором – мантисса.

Для определения порядка результата порядок исходного операнда следует разделить на 2. Если порядок подкоренного числа четный, то деление на 2 выполняется путем сдвига его кода вправо на один двоичный разряд (отрицательный порядок должен быть представлен инверсным кодом). Если порядок нечетный, то перед делением на 2 его значение следует увеличить на единицу, что повлечет за собой необходимость денормализовать мантиссу. Таким образом, при нечетном порядке к его значению в первом такте прибавляют 1, а во втором такте одновременно сдвигают мантиссу и увеличенное значение порядка на один разряд вправо. При четном или нечетном порядке операнда результат выполнения операции все равно будет получаться нормализованным.

Как и в случае деления, квадратный корень можно извлекать с восстановлением и без восстановления остатка. При этом результат всегда будет получаться с недостачей, поэтому необходимо вычислять дополнительную цифру и производить округление.

Рассмотрим сначала метод извлечения квадратного корня с восстановлением остатка.

Для получения $(n+1)$ цифр корня следует выполнить $(n+1)$ циклов, каждый из которых распадается на три такта. В первом такте из содержимого сумматора, т.е. из очередного частичного остатка, вычитается делитель, сформированный из уже полученных цифр частного и приписки «01». Очевидно, что в первом цикле делитель представляет собой число «0,01». Во втором такте в случае отрицательного сумматора

требуется процедура восстановления остатка путем прибавления к сумматору положительного делителя. В третьем такте формируется очередная (начиная со старшей) цифра корня, равная нулю, если в первом такте сумматор оказался отрицательным, или единице, если в первом такте сумматор оказался положительным. В этом же такте выполняется сдвиг очередного частичного остатка на один разряд влево. По окончании выполнения операции производится округление результата.

Пример.

$[A]_{\text{пр}} = 0{,}11010;\ [P_A]_{\text{пр}} = 0{,}011;$

$B = \sqrt{A}.$

1. Определение порядка результата.

$P_B = P_A/2$:

$\mathrm{P}_1 = 0{,}11010;\ \ \mathrm{P}_{\text{п}} = 0{,}011;$

$\mathrm{P}_{\text{п}} + 1 = 0{,}100;$

$\vec{\mathrm{P}}_1 = 0{,}01101;\ \ \vec{\mathrm{P}}_{\text{п}} = 0{,}010.$

Таким образом, порядок результата $P_B = 0{,}010$.

2. Определение мантиссы.

```
      СМ = 0,0110100      P1 = 0,0100000
1.   -P1 = 1,1100000
      СМ = 0,0010100      P1 = 0,1010000
     ←СМ = 0,0101000
2.   -P1 = 1,0110000
      СМ = 1,1011000
     +P1 = 0,1010000
      СМ = 0,0101000      P1 = 0,1001000
     ←СМ = 0,1010000
3.   -P1 = 1,0111000
      СМ = 0,0001000      P1 = 0,1010100
     ←СМ = 0,0010000
4.   -P1 = 1,0101100
      СМ = 1,0111100
     +P1 = 0,1010100
      СМ = 0,0010000      P1 = 0,1010010
```

(окончание примера на следующей странице)

$\overleftarrow{CM} = 0,0100000$

5. $-P_1 = \underline{1,0101110}$

$CM = 1,1001110$

$+P_1 = \underline{0,1010010}$

$CM = 0,0100000 \qquad P_1 = 0,1010001$

$\overleftarrow{CM} = 0,1000000$

6. $-P_1 = \underline{1,0101111}$

$CM = 1,1101111$

$+P_1 = \underline{0,1010001}$

$CM = 0,1000000 \qquad P_1 = 0,1010000$

$\overleftarrow{CM} = 1,0000000$

Окр. $CM = 0,1010000$

$\underline{0,0000010}$

$CM = 0,1010010$

Окончательно $[B]_{\text{пр}} = 0{,}10100;\ [P_B]_{\text{пр}} = 0{,}010.$

Извлечение квадратного корня без восстановления остатка является модификацией предыдущего алгоритма. В каждом цикле выполняются два такта. В первом такте в зависимости от знака сумматора выполняется арифметическая операция. Если сумматор положительный, то из него вычитается делитель. Если сумматор отрицательный, то, наоборот, к нему прибавляется положительный делитель. Во втором такте анализируется знак очередного остатка. Если остаток положительный, то очередная цифра частного – единица и приписка к частному для формирования делителя «01». В противном случае очередная цифра частного – нуль и приписка к частному для формирования делителя «11». Одновременно с формированием частного и переменного делителя выполняется удвоение очередного частичного остатка. При этом знаковый разряд может быть искажен, поэтому для сохранения знака следует использовать модифицированный код.

Пример.

Рассмотрим число из предыдущего примера.

$CM = 00,0110100 \qquad P_1 = 00,0100000$

1. $-P_1 = \underline{11,1100000}$

$CM = 00,0010100 \qquad P_1 = 00,1010000$

(окончание примера на следующей странице)

$\overleftarrow{\text{СМ}}=00,0101000$

2. $-P_1=\underline{11,0110000}$

$\text{СМ}=11,1011000 \qquad P_1=00,1011000$

$\overleftarrow{\text{СМ}}=11,0110000$

3. $+P_1=\underline{00,1011000}$

$\text{СМ}=00,0001000 \qquad P_1=00,1010100$

$\overleftarrow{\text{СМ}}=00,0010000$

4. $-P_1=\underline{11,0101100}$

$\text{СМ}=11,0111100 \qquad P_1=00,1010110$

$\overleftarrow{\text{СМ}}=10,1111000$

5. $+P_1=\underline{00,1010110}$

$\text{СМ}=11,1001110 \qquad P_1=00,1010011$

$\overleftarrow{\text{СМ}}=11,0011100$

6. $+P_1=\underline{00,1010011}$

$\text{СМ}=11,1101111 \qquad P_1=00,1010001$

$\overleftarrow{\text{СМ}}=11,1011110$

Окр. $\text{СМ}=00,1010001$

$\underline{00,0000010}$

$\text{СМ}=00,1010011$

Окончательно $[B]_{\text{пр}}=0{,}10100;\ [P_B]_{\text{пр}}=0{,}010.$

2.7. Выполнение арифметических операций в D-кодах

О представлении чисел двоично-десятичными кодами см. в разделах 1.1 и 1.5.

2.7.1. Сложение в D-кодах

Вначале рассмотрим сложение чисел в D-коде 8421.

Выполнение операции сложения в D-коде 8421 имеет ряд особенностей: во-первых, возникающий при сложении межтетрадный перенос вместо необходимых десяти единиц (как это принято в десятичной системе счисления при формировании переноса в следующий разряд) уносит шестнадцать (16 – удвоенный вес старшего разряда тетрады), т.е. значение, содержащееся в тетраде, из которой возник перенос, на шесть единиц меньше необходимого; во-вторых, при сложении тетрад могут получаться запрещенные комбинации, соответствующие

числам больше девяти, но меньше пятнадцати; в-третьих, отрицательные числа необходимо представлять в инверсном коде, но, как указывалось выше, D-код 8421 не обладает свойством самодополняемости, что необходимо учитывать.

Все указанные трудности решаются путем прибавления тетрады 0110, соответствующей коду цифры 6. Действительно, если межтетрадный перенос уменьшает значение в тетраде на шесть лишних единиц, то коррекция может быть осуществлена путем прибавления недостающих шести единиц. Далее, для коррекции тетрад, содержащих запрещенные комбинации, из них, очевидно, необходимо вычесть 10. Вычитание 10-ти можно заменить прибавлением дополнения 10-ти до 16-ти, т.е. для коррекции к значению запрещенной комбинации следует прибавить 6. Для получения обратного кода все тетрады числа (кроме знака) необходимо сначала увеличить на 6, а затем поразрядно инвертировать. Дополнительный код может быть получен из обратного путем прибавления единицы к младшему разряду младшей тетрады числа.

Рассмотрим пример.

$A = -0{,}567;\ B = 0{,}742;$

$C = A + B.$

$[A]_{\text{пр}} = 1001{,}0101\ 0110\ 0111;$

$[B]_{\text{пр}} = 0000{,}0111\ 0100\ 0010.$

Поскольку первое слагаемое отрицательное, его необходимо представить в инверсном коде, например, дополнительном.

$$
\begin{array}{rl}
[A]_{\text{пр}} = & 1001,0101\ 0110\ 0111 \\
+ & \underline{0110\ 0110\ 0110} \\
& 1001,1011\ 1100\ 1101 \\
[A]_{\text{об}} = & 1001,0100\ 0011\ 0010 \\
[A]_{\text{д}} = & 1001,0100\ 0011\ 0011
\end{array}
$$

Так как дополнительный код положительного слагаемого совпадает с его прямым кодом, переходим непосредственно к сложению.

$$
\begin{array}{rll}
[A]_{\text{д}} = & 1001,0100\ 0011\ 0011 & \\
+[B]_{\text{д}} = & \underline{0000,0111\ 0100\ 0010} & \\
& 1001,1011\ 0111\ 0101 & \\
+ & \underline{0000,0110\ 0000\ 0000} & \text{Коррекция} \\
& 1010,0001\ 0111\ 0101 & \\
+ & \underline{0110} & \text{Коррекция (нули опустим)}
\end{array}
$$

$[C]_{\text{д}} = 0000,0001\ 0111\ 0101$

Таким образом, $C = 0{,}175$.

При выполнении сложения в D-коде 8421+3 обратный код отрицательного операнда может быть получен путем инвертирования двоичных разрядов кода (дополнительный код получается из обратного прибавлением единицы к младшему разряду).

После сложения операндов необходимо всегда выполнять коррекцию результата. Легко видеть, что если при сложении из тетрады не было переноса, то ее значение следует уменьшить на три единицы (вычесть 0011), а если перенос был, то увеличить на три единицы (прибавить 0011). Очевидно, вычитание 0011 можно заменить прибавлением дополнения 3 до 16, т.е. для коррекции к тетрадам, из которых при сложении не возник перенос, необходимо прибавить 1101. Следует помнить, что при выполнении коррекции в D-коде 8421+3 переносы между тетрадами не распространяются.

Рассмотрим пример. Значения слагаемых возьмем из предыдущего примера.

$A = -0{,}567$; $B = 0{,}742$;

$C = A + B$.

$[A]^{3}_{\text{пр}} = 1100,1000\ 1001\ 1010$;

$[B]^{3}_{\text{пр}} = 0011,1010\ 0111\ 0101$.

Получим дополнительный код первого операнда вышеописанным способом и выполним сложение.

$[A]^{3}_{\text{д}} = 1100,0111\ 0110\ 0110$

$+[B]^{3}_{\text{д}} = \underline{0011,1010\ 0111\ 0101}$

$\overleftarrow{\ }\ 0000,\overleftarrow{0001}\ 1101\ 1011$ (переносы показаны стрелками)

$\underline{\ 0011,0011\ 1101\ 1101}$ Коррекция (без переносов)

$[C]^{3}_{\text{д}} = 0011,0100\ 1010\ 1000$

Таким образом, $C = 0{,}175$.

В дальнейшем будем рассматривать выполнение арифметических операций в D-коде 8421+3.

2.7.2. Умножение в D-кодах

Для выполнения операции умножения в D-кодах необходимо выполнить n циклов по числу десятичных разрядов множителя. После каждого цикла выполняется сдвиг содержимого сумматора и регистра множителя на один десятичный разряд влево. После выполнения умножения производится округление результата путем добавления к младшей (дополнительной) тетраде кода цифры 5. Знак произведения определяется в результате анализа знаков сомножителей. Очевидно, в простейшем случае в каждом цикле множимое прибавляется столько раз, сколько единиц содержится в очередной цифре множителя. Однако такой метод умножения характеризуется большими вычислительными, а следовательно, временными затратами.

Для ускорения выполнения операции умножения в D-кодах можно, например, представить каждую десятичную цифру множителя как совокупность сложений с удвоенным или учетверенным множимым. Тогда, например, вместо выполнения семи сложений при умножении на цифру 7 достаточно выполнить три сложения: $+(4A)+(2A)+(A)=+7A$, где A – множимое. Однако такой метод потребует дополнительных аппаратных ресурсов для хранения удвоенного и учетверенного множимых, так как процедура их получения предполагает помимо сдвига прибавление поправки к отдельным тетрадам, что исключает использование простых регистров.

Если при умножении на некоторые цифры множителя заменить операцию прибавления к содержимому сумматора на вычитание, то в некоторых случаях можно обойтись без учетверенного множимого либо уменьшить число операций на сумматоре. При этом, однако, может потребоваться корректировка отдельных цифр множителя.

Рассмотрим три варианта ускорения умножения (табл. 2.6). При использовании первых двух из них необходимо корректировать цифры множителя. Так, если очередная цифра множителя больше пяти, то последующая должна быть увеличена на единицу. Для хранения и модификации очередной цифры множителя в структуру устройства введем счетчик.

В качестве примера рассмотрим умножение двух чисел, представленных в D-коде 8421+3. В этом случае удвоение множимого выполняется следующим образом: множимое сдвигается на один двоичный разряд вправо, затем выполняется коррекция, которая

заключается в прибавлении поправки 1101 к тем тетрадам, которые до сдвига содержали коды цифр 0–4, и поправки 0011 к тетрадам, содержавшим до сдвига коды цифр 5–9 (в случае D-кода 8421 коррекция делается путем прибавления поправки 0110 к тетрадам, содержавшим до сдвига коды цифр 5–9).

Таблица 2.6

Цифра множителя	Варианты выполнения операций на сумматоре		
0	$+0$	$+0$	$+0$
1	$+A$	$+A$	$+A$
2	$+2A$	$+2A$	$+2A$
3	$+2A+A$	$+2A+A$	$+2A+A$
4	$+2A+2A$	$+4A$	$+4A$
5	$+2A+2A+A$	$+4A+A$	$+4A+A$
6	$-2A-2A$	$-4A$	$+4A+2A$
7	$-2A-A$	$-2A-A$	$+4A+2A+A$
8	$-2A$	$-2A$	$+4A+4A$
9	$-A$	$-A$	$+4A+4A+A$

При выполнении умножения воспользуемся первым столбцом табл. 2.6. Для сохранения знака в сумматоре используем модифицированный код.

Пример.

$A=0{,}423;\ B=0{,}361;$

$C=A\times B.$

$P_1 = 0011,0111\ 0101\ 0110;$

$2P_1 = 0011,1011\ 0111\ 1001;$

$P_2 = 0011,0110\ 1001\ 0100.$

При выполнении примера из соображений удобства и наглядности содержимое счетчика будем изображать десятичной цифрой.

$$
\begin{array}{rll}
CM = & 0011\ 0011,0011\ 0011\ 0011\ 0011 & Сч=1 \\
+P_1 = & \underline{0011\ 0011,0111\ 0101\ 0110\ 0011} & \\
CM = & 0110\ 0110,1010\ 1000\ 1001\ 0110 & \\
 & \underline{1101\ 1101,1101\ 1101\ 1101\ 1101} & \text{Коррекция} \\
CM = & \overline{0011\ 0011,0111\ 0101\ 0110\ 0011} & \\
\overrightarrow{CM} = & 0011\ 0011,0011\ 0111\ 0101\ 0110 & \vec{P}_2 = 0011,0011\ 0110\ 1001 \\
-2P_1 = & \underline{1100\ 1100,0100\ 1000\ 0110\ 1101} & Сч=6 \\
CM = & \overline{1111\ 1111,0111\ 1111\ 1100\ 0011} & \\
 & \underline{1101\ 1101,1101\ 1101\ 1101\ 0011} & \text{Коррекция} \\
CM = & \overline{1100\ 1100,0100\ 1100\ 1001\ 0110} &
\end{array}
$$

(окончание примера на следующей странице)

```
−2P1 =1100 1100,0100 1000 0110 1101
      ←    ←    ←    ←
 СМ =1001 1000,1001 0101 0000 0011
     0011 0011,1101 0011 0011 0011   Коррекция
 СМ =1100 1011,0110 1000 0011 0110
 СМ→=1100 1100,1011 0110 1000 0011 P2→=0011,0011 0011 0110
+2P1 =0011 0011,1011 0111 1001 0011   Сч=3+1=4
      ←    ←    ←    ←
 СМ =0000 0000,0110 1110 0001 0110
     0011 0011,0011 1101 0011 1101   Коррекция
 СМ =0011 0011,1001 1011 0100 0011
+2P1 =0011 0011,1011 0111 1001 0011
      ←    ←
 СМ =0110 0111,0101 0010 1101 0110
     1101 1101,0011 0011 1101 1101   Коррекция
 СМ =0011 0100,1000 0101 1010 0011
 СМ→=0011 0011,0100 1000 0101 1010 P2→=0011,0011 0011 0011
                                1000   Округление
                                ←
 СМ =0011 0011,0100 1000 0110 0010
```

Таким образом, округляя до трех разрядов после запятой, получим $C = 0{,}153$.

Действительно, $0{,}423 \times 0{,}361 = 0{,}152703$, что после округления до исходных трех разрядов после запятой дает полученный выше результат.

2.7.3. Деление в D-кодах

По причине больших временных затрат алгоритм деления в D-кодах с восстановлением остатка обычно не реализуется в цифровых вычислительных устройствах, поэтому здесь будем рассматривать только алгоритм деления без восстановления остатка. В качестве упражнения читателю может быть предложено самостоятельно выполнить рассматриваемую арифметическую операцию с восстановлением и без восстановления остатка и оценить разницу в количестве выполняемых действий.

Рассмотрим следующую методику выполнения операции деления. Знак частного определяется в результате анализа знаков операндов по обычным правилам арифметики, после чего оба операнда делаются положительными. Чтобы исключить ситуацию с переполнением, будем рассматривать случай, когда модуль делимого меньше модуля делителя. Поскольку в рассматриваемом алгоритме делитель сдвигается вправо, для его хранения необходимо использовать регистр с k дополнительными двоичными разрядами; k определяется из неравенства $k \geq \log_2(m-k)$, где m – число двоичных разрядов в представлении операндов. Очевидно, в

сумматоре, содержащем делимое и частичные остатки, также должны быть предусмотрены k дополнительных разрядов. Учитывая особенности представления чисел в D-кодах, число дополнительных разрядов, определяемое вышеприведенным неравенством, при необходимости должно быть увеличено до образования соответствующего числа двоично-десятичных тетрад. Помимо сумматора и двух регистров (частного и делителя) устройство также должно содержать счетчик, используемый для формирования цифр частного.

Для получения n цифр частного необходимо выполнить n циклов деления. В первом такте каждого цикла содержимое регистра делителя сдвигается вправо на один десятичный разряд, а содержимое регистра частного сдвигается на один десятичный разряд влево. При этом в младший разряд регистра частного заносится текущее содержимое счетчика. Циклы делятся на четные и нечетные. Перед началом выполнения нечетных циклов в счетчике устанавливается цифра 0. Затем выполняется вычитание делителя до тех пор, пока содержимое сумматора не изменит знак на противоположный. В каждом такте вычитания (кроме последнего, соответствующего изменению знака) содержимое счетчика увеличивается на единицу. Перед началом выполнения четных циклов в счетчике устанавливается цифра 9. Затем выполняется прибавление делителя до тех пор, пока содержимое сумматора не изменит знак. В каждом такте сложения (кроме последнего) содержимое счетчика уменьшается на единицу. После выполнения n циклов необходимо выполнить действия, соответствующие первому такту $(n+1)$-го цикла, для передачи последней определенной цифры частного из счетчика в соответствующий регистр.

Пример.

Операнды представим в D-коде 8421+3.

$A = 0{,}15;\ B = 0{,}53;$

$C = A/B.$

$\text{СМ} = 0011,0100\ 1000\ 0011 \qquad \text{Сч} = 0$

$P_2 = 0011,1000\ 0110\ 0011 \qquad P_1 = 0011,0011\ 0011$

$\vec{P}_2 = 0011,0011\ 1000\ 0110 \qquad \vec{P}_1 = 0011,0011\ 0011$

$\text{СМ} = 0011,0100\ 1000\ 0011$

$-P_2 = \underline{1100,1100\ 0111\ 1010}$

$\text{СМ} = \overleftarrow{0000,0000\ 1111\ 1101}$

$0011,0011\ 1101\ 1101$ Коррекция

$CM = \overline{0011,0011\ 1100\ 1010} \quad > 0 \Rightarrow Сч = Сч + 1 = 1$

(окончание примера на следующей странице)

$-P_2 = 1100,1100\ 0111\ 1010$

$CM = \overleftarrow{0000},\overleftarrow{0000}\ \overleftarrow{0100}\ \overleftarrow{0100}$

$0011,0011\ 0011\ 0011$ Коррекция

$CM = 0011,0011\ 0111\ 0111 \quad > 0 \Rightarrow Сч = Сч + 1 = 2$

$-P_2 = 1100,1100\ 0111\ 1010$

$CM = 1111,1111\ 1111\overleftarrow{0001}$

$1101,1101\ 1101\ 0011$ Коррекция

$CM = 1100,1100\ 1100\ 0100 \quad < 0$

$\vec{P}_2 = 0011,0011\ 0011\ 1000 \qquad \overleftarrow{P_1} = 0011,0011\ 0101 \qquad Сч = 9$

$CM = 1100,1100\ 1100\ 0100$

$+P_2 = 0011,0011\ 0011\ 1000$

$CM = 1111,1111\ 1111\ 1100$

$1101,1101\ 1101\ 1101$ Коррекция

$CM = 1100,1100\ 1100\ 1001 \quad < 0 \Rightarrow Сч = Сч - 1 = 8$

$+P_2 = 0011,0011\ 0011\ 1000$

$CM = \overleftarrow{0000},\overleftarrow{0000}\ \overleftarrow{0000}\ \overleftarrow{0001}$

$0011,0011\ 0011\ 0011$ Коррекция

$CM = 0011,0011\ 0011\ 0100 \quad > 0$

$\vec{P}_2 = 0011,0011\ 0011\ 0011 \qquad \overleftarrow{P_1} = 0011,0101\ 1011 \qquad Сч = 0$

Таким образом, $C = 0{,}28$.

Вопросы для самоконтроля

1. В чем заключаются основные этапы выполнения операции сложения? Чем отличается выполнение операции вычитания?

2. В чем заключаются основные этапы выполнения операции умножения прямых кодов чисел? В чем различие методов умножения старшими и младшими разрядами вперед?

3. В чем заключаются основные этапы рассмотренных методов ускоренного выполнения операции умножения?

4. Чем отличается умножение инверсных кодов чисел от умножения прямых кодов?

5. В чем заключаются основные этапы выполнения операции деления прямых кодов чисел? В чем различие методов деления с восстановлением и без восстановления остатка?

6. В чем заключаются основные этапы рассмотренных методов ускоренного выполнения операции деления?

7. Чем отличается деление инверсных кодов чисел от деления прямых кодов?

8. В чем заключаются основные этапы выполнения операции извлечения квадратного корня? В чем различие методов с восстановлением и без восстановления остатка?

9. Особенности выполнения арифметических операций в D-кодах.

3. ПЕРЕКЛЮЧАТЕЛЬНЫЕ ФУНКЦИИ

В цифровых устройствах информация кодируется наборами цифр, которые, в свою очередь, представляют собой электрические сигналы различного уровня, соответствующего той или иной цифре. Пусть некоторое устройство имеет *n* входов, сигналы на которых обозначаются переменными x_i, $(i = 1, 2, ..., n)$, и *m* выходов, на которых появляются сигналы, обозначаемые y_j, $(j = 1, 2, ..., m)$. Будем называть *алфавитом переменной* множество значений, которые она может принимать, а элементы этого множества будем называть *буквами алфавита* или просто *буквами*. В цифровых вычислительных машинах традиционно используется двоичный алфавит $x_i, y_i \in \{0,1\}$. Конечные упорядоченные последовательности букв будем называть *словами* в данном алфавите.

Следует различать алфавит переменной и входной или выходной алфавиты схемы, представляющие сбой наборы вида $(a_1, a_2, ..., a_n)$, $(a_1, a_2, ..., a_m)$, в которых элементами являются буквы алфавитов соответствующих входных переменных. Таким образом, если число букв в алфавитах входных переменных равно r_i, $(i = 1, 2, ..., n)$, а в алфавитах выходных переменных q_j, $(j = 1, 2, ..., m)$, то число букв *R* во входном и *Q* в выходном алфавитах схемы будет равно

$$R = \prod_{i=1}^{n} r_i \text{ и } Q = \prod_{j=1}^{m} q_j .$$

Ясно, что в случае двоичных алфавитов переменных $R = 2^n$, $Q = 2^m$.

Поскольку число входов и выходов цифровых схем, а также число букв в алфавитах переменных конечно, то алфавиты таких схем также будут конечными.

Если значения выходных переменных схемы зависят только от значений входных переменных и наборы выходных переменных однозначно определяются соответствующими наборами входных переменных, такие схемы называются *комбинационными схемами* (КС) или схемами прямого распространения (автоматами без памяти).

Зависимость выходных реакций КС от входного воздействия описывается функциями специального вида, называемыми *логическими* или *переключательными функциями* (ПФ).

Исходными данными для выполнения процедуры логического синтеза комбинационной схемы являются: совокупность логических функций, реализуемых синтезируемой КС, и набор логических элементов,

используемых для построения устройства. Результатом синтеза является схема соединения логических элементов, реализующая заданную совокупность логических функций.

В дальнейшем по умолчанию будем рассматривать логические функции и их аргументы, могущие принимать дискретные значения 0 и 1. Такие логические функции называются *двоичными функциями* или *булевыми функциями*. Таким образом, для любой рассматриваемой далее ПФ и ее аргументов справедливо: $f(x_1, x_2, ..., x_n) \in \{0,1\}$, $x_i \in \{0,1\}$, если иное не оговорено отдельно.

3.1. Основные определения и способы задания ПФ

Вектор входных переменных переключательной функции $\mathbf{x} = (x_1, x_2, ..., x_n)$ называется *набором*. Любой набор значений аргументов ПФ будем рассматривать как целое двоичное число, определяющее его номер при перечислении: 000...00 – «ноль» (нулевой набор), 000...01 – «один» (первый набор), 000...10 – «два» (второй набор), и т.д.

Поскольку общее число наборов равно 2^n и на каждом из них ПФ может принимать одно из двух значений: 0 или 1, максимальное число различных переключательных функций, зависящих от n переменных, равно 2^{2^n}.

Любая ПФ, зависящая от n аргументов, может быть определена на $p \le 2^n$ наборах. Если для некоторой функции $p = 2^n$, то такая функция называется *полностью определенной*. В противном случае функция называется *не полностью определенной*.

Пусть s – число наборов, на которых значение функции не определено. Не полностью определенную функцию можно доопределить значениями 0 или 1 на всех s наборах, при этом выбор совокупности значений функции может определяться произвольно или исходя из каких-либо рациональных соображений. Таким образом, при произвольном доопределении не полностью определенной функции можно получить 2^s различных полностью определенных функций.

Переключательную функцию можно задать одним из трех способов: аналитическим, геометрическим или матричным [4]. Простейшей разновидностью матричного способа задания является составление таблицы истинности ПФ. Таблица истинности булевой функции представляет собой совокупность наборов входных аргументов и

соответствующие этим наборам выходные значения задаваемой функции (табл. 3.1).

Таблица 3.1

№ набора	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$f_1(x_1, x_2, x_3, x_4)$	1	0	1	1	1	0	0	0	0	0	1	1	0	1	1	0
$f_2(x_1, x_2, x_3, x_4)$	1	1	0	0	0	1	1	0	1	0	0	0	0	0	0	1
$f_3(x_1, x_2, x_3, x_4)$	1	1	0	0	1	1	1	1	0	0	1	1	0	0	0	0

Задание ПФ при помощи таблицы истинности не всегда удобно, так как размеры таблицы растут пропорционально 2^n, где n – количество входных переменных.

При аналитическом способе задания ПФ определяемая функция описывается выражениями с использованием последовательностей входных переменных и операций алгебры логики (более подробно этот способ задания ПФ рассматривается в разделе 3.5).

Рассмотрим геометрический способ задания ПФ. Каждый двоичный набор n переменных $\mathbf{x} = (x_1, x_2, ..., x_n)$ в геометрическом смысле можно интерпретировать как вектор единичной длины, определяющий точку n-мерного пространства. Таким образом, множество 2^n наборов определяет множество вершин n-мерного гиперкуба. Для $n = 3$ на рис. 3.1 представлен трехмерный куб, где каждой вершине соответствует один из возможных наборов входных переменных. Для задания ПФ графическим способом необходимо определить ее значения в вершинах n-мерного гиперкуба.

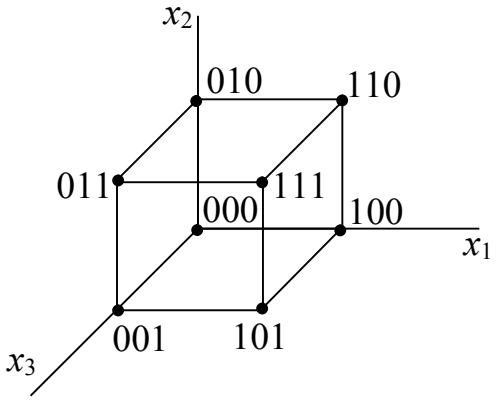


Рис. 3.1

Переключательная функция называется *существенно зависящей от переменной* x_i, если хотя бы для одного набора переменных $(x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$ выполняется следующее соотношение:

$$f(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \neq f(x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n).$$

Если такое соотношение не выполняется, то функция $f(x_1, x_2, ..., x_n)$ не зависит от переменной x_i. В этом случае переменная x_i называется *фиктивной*. Фиктивные переменные можно исключить, так как от этого значения функции не изменяются.

Один из способов определения фиктивных переменных заключается в следующем. Разобьем множество наборов входных переменных на два подмножества. В первое подмножество входят наборы, на которых функция принимает значение 1, а во второе подмножество входят наборы, на которых функция принимает значение 0. Для проверки фиктивности переменной x_i необходимо вычеркнуть соответствующие ей столбцы из обоих подмножеств. Отсутствие в подмножествах одинаковых наборов указывает на фиктивность аргумента x_i.

В качестве примера рассмотрим функцию f_3, заданную табл. 3.1. Разобьем множество ее аргументов на два подмножества (табл. 3.2, 3.3). Вычеркнем в этих таблицах столбцы, соответствующие переменной x_4. Поскольку в таблицах отсутствуют одинаковые наборы, можно сделать вывод, что переменная x_4 является фиктивной.

Таблица 3.2

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

Таблица 3.3

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

$f_3(x_1, x_2, x_3, x_4) = 1$ $f_3(x_1, x_2, x_3, x_4) = 0$

3.2. Элементарные логические функции

Существуют четыре различные ПФ, зависящие от одного аргумента. При этом для функций $f_0(x)$ и $f_3(x)$ аргумент x является фиктивным (табл. 3.4).

Таблица 3.4

$f(x) \setminus x$	0	1	Условное обозначение	Название функции
$f_0(x)$	0	0	0	Константа 0
$f_1(x)$	0	1	x	Переменная x
$f_2(x)$	1	0	$\bar{x}$	Инверсия x
$f_3(x)$	1	1	1	Константа 1

Существуют 16 различных ПФ, зависящих от двух аргументов (табл. 3.5). При этом для функций $f_0(x_1,x_2)$ и $f_{15}(x_1,x_2)$ оба аргумента x_1, x_2 являются фиктивными, а для функций $f_3(x_1,x_2)$, $f_5(x_1,x_2)$, $f_{10}(x_1,x_2)$ и $f_{12}(x_1,x_2)$ один из аргументов является фиктивным.

Таблица 3.5

x_1 / $f(x_1,x_2)$ / x_2	0 0	0 1	1 0	1 1	Условное обозначение	Название функции
$f_0(x_1,x_2)$	0	0	0	0	0	Константа 0
$f_1(x_1,x_2)$	0	0	0	1	$x_1 \cdot x_2$	Конъюнкция
$f_2(x_1,x_2)$	0	0	1	0	$x_1 \Delta x_2$	Запрет по x_2
$f_3(x_1,x_2)$	0	0	1	1	x_1	Переменная x_1
$f_4(x_1,x_2)$	0	1	0	0	$x_2 \Delta x_1$	Запрет по x_1
$f_5(x_1,x_2)$	0	1	0	1	x_2	Переменная x_2
$f_6(x_1,x_2)$	0	1	1	0	$x_1 \oplus x_2$	Сложение по модулю 2
$f_7(x_1,x_2)$	0	1	1	1	$x_1 \vee x_2$	Дизъюнкция
$f_8(x_1,x_2)$	1	0	0	0	$x_1 \downarrow x_2$	Стрелка Пирса
$f_9(x_1,x_2)$	1	0	0	1	$x_1 \sim x_2$	Эквивалентность
$f_{10}(x_1,x_2)$	1	0	1	0	$\bar{x}_2$	Инверсия x_2
$f_{11}(x_1,x_2)$	1	0	1	1	$x_2 \rightarrow x_1$	Импликация x_2 в x_1
$f_{12}(x_1,x_2)$	1	1	0	0	$\bar{x}_1$	Инверсия x_1
$f_{13}(x_1,x_2)$	1	1	0	1	$x_1 \rightarrow x_2$	Импликация x_1 в x_2
$f_{14}(x_1,x_2)$	1	1	1	0	$x_1 \mid x_2$	Штрих Шеффера

$f_{15}(x_1, x_2)$	1	1	1	1	1	Константа 1

Все ПФ одного аргумента, а также функции двух аргументов с номерами 1, 6, 7, 8, 9, 11, 14 называют *элементарными* и используют для построения более сложных функций путем изменения номеров аргументов и с помощью суперпозиции, т.е. подстановки вместо аргументов других переключательных функций.

3.3. Основные законы алгебры логики

Для логических операций справедливы следующие закономерности:

1) Ассоциативность
$$x_1 \vee x_2 \vee x_3 = x_1 \vee (x_2 \vee x_3);$$
$$x_1 \cdot x_2 \cdot x_3 = (x_1 \cdot x_2) \cdot x_3.$$

2) Коммутативность
$$x_1 \vee x_2 \vee x_3 = x_3 \vee x_2 \vee x_1;$$
$$x_1 \cdot x_2 \cdot x_3 = x_3 \cdot x_2 \cdot x_1.$$

3) Дистрибутивность
$$(x_1 \vee x_2) \cdot x_3 = (x_1 \cdot x_3) \vee (x_2 \cdot x_3);$$
$$x_1 \vee x_2 x_3 = (x_1 \vee x_2)(x_1 \vee x_3).$$

Кроме коммутативного, ассоциативного и дистрибутивного законов, выделяют законы де Моргана, которые записываются следующим образом:

$$\overline{x_1 \vee x_2} = \bar{x}_1 \cdot \bar{x}_2; \quad \overline{x_1 \cdot x_2} = \bar{x}_1 \vee \bar{x}_2.$$

Отметим некоторые свойства элементарных функций, применяемые при различных преобразованиях логических функций.

$$\bar{\bar{x}} = x;$$
$$x \cdot x = x;$$
$$x \vee x = x;$$
$$x \vee \bar{x} = 1;$$
$$x \cdot \bar{x} = 0;$$
$$x \vee 1 = 1;$$
$$x \vee 0 = x;$$
$$x \cdot 1 = x;$$
$$x \cdot 0 = 0;$$

$x_1(\bar{x}_1 \vee x_2) = x_1 x_2$;

$x_1 \vee \bar{x}_1 x_2 = x_1 \vee x_2$;

$x_1 x_2 \vee x_1 \bar{x}_2 = x_1$ – правило склеивания;

$x_1 x_2 \vee x_1 = x_1$ – правило поглощения;

$x_1 \oplus x_2 = \bar{x}_1 x_2 \vee x_1 \bar{x}_2$;

$\overline{x_1 \oplus x_2} = \bar{x}_1 \bar{x}_2 \vee x_1 x_2$;

$x_1 \downarrow x_2 = \overline{x_1 \vee x_2} = \overline{\bar{x}_1 \mid \bar{x}_2}$;

$x_1 \mid x_2 = \overline{x_1 \cdot x_2} = \overline{\bar{x}_1 \downarrow \bar{x}_2}$;

$x_1 \to x_2 = \bar{x}_1 \vee x_2$.

3.4. Полные системы переключательных функций

Одной из наиболее важных задач, решаемой в процессе синтеза комбинационной схемы, является рациональный выбор логических элементов, которые будут использоваться при технической реализации логических схем. Поэтому основное требование, предъявляемое к набору логических элементов, заключается в том, что на основе этого набора элементов, реализующих элементарные логические функции, можно построить комбинационную схему, реализующую произвольную ПФ или (в общем случае) систему ПФ.

Следовательно, система логических функций, описывающих синтезируемую КС, должна быть представлена в виде переключательных функций, реализуемых выбранным набором логических элементов.

Система ПФ $(f_1, f_2, ..., f_m)$ называется *функционально полной*, если любую булеву функцию произвольной сложности можно записать в виде формулы с использованием функций этой системы $f_1, f_2, ..., f_m$.

Функционально полная система ПФ называется *базисом*. Наибольшее распространение получили базисы {И, ИЛИ, НЕ}, {И-НЕ} (штрих Шеффера), {ИЛИ-НЕ} (стрелка Пирса).

Минимальным базисом называется такой базис, для которого исключение хотя бы одной из функций, образующих этот базис, превращает систему переключательных функций в неполную.

Базис может быть образован элементарными функциями, удовлетворяющими условиям теоремы Поста – Яблонского.

Теорема. Для того, чтобы система элементарных ПФ была функционально полной, необходимо и достаточно, чтобы она включала

хотя бы одну функцию, не сохраняющую ноль, не сохраняющую единицу, не самодвойственную, не монотонную, не линейную.

Доказательство этой теоремы можно найти в [6].

Сформулируем ряд определений для классов переключательных функций, упоминаемых в теореме Поста – Яблонского.

Первый класс составляют функции, сохраняющие константу 0. Для таких функций должно выполняться условие $f(0,0,...,0) = 0$.

Второй класс составляют функции, сохраняющие константу 1. Для таких функций должно выполняться условие $f(1,1,...,1) = 1$.

Третий класс составляют самодвойственные функции, которые на паре противоположных аргументов принимают противоположные значения. Для самодвойственных функций выполняется следующее равенство: $f(x_1, x_2, ..., x_n) = \overline{f(\bar{x}_1, \bar{x}_2, ..., \bar{x}_n)}$. Примером такой функции может служить отрицание.

Четвертый класс составляют линейные ПФ. *Линейной ПФ* называется функция, которая может быть представлена полиномом по модулю 2 первой степени. В общем виде линейная ПФ может быть записана следующим образом: $f(x_1, x_2, ..., x_n) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus ... \oplus a_n x_n$, где $a_i \in \{0,1\}$, $i = 0,1,...,n$.

Пятый класс составляют монотонные ПФ. Введем критерий сравнения двух наборов аргументов. Два набора аргументов называются *сравнимыми*, если значение каждого аргумента одного набора больше или равно значению соответствующих аргументов второго набора, т.е. $x_i \le x_i'$, $i = 1,2,...,n.$

Монотонной называется такая ПФ, для которой при любом возрастании набора значения этой функции не убывают, т.е. $f(x_1, x_2, ..., x_n) \le f(x_1', x_2', ..., x_n')$.

Принадлежность некоторых элементарных функций к тому или иному классу представлена в табл. 3.6. На пересечении строки и столбца поставлена отметка, если данная элементарная ПФ принадлежит к указанному классу.

Таблица 3.6

Класс / ПФ	Сохр. 0	Сохр. 1	Самодвойст-венная	Монотонная	Линейная
Константа 0	∨			∨	∨
Конъюнкция	∨	∨		∨	
Дизъюнкция	∨	∨		∨	

Сложение по модулю 2	∨				∨

Окончание табл. 3.6

Эквивалентность		∨			∨
Стрелка Пирса					
Штрих Шеффера					
Импликация		∨			
Отрицание			∨		∨
Константа 1		∨		∨	∨

Из приведенной таблицы следует, что канонический базис булевой алгебры И-ИЛИ-НЕ, включающий конъюнкцию, дизъюнкцию и отрицание, является функционально полным, соответствующим требованиям теоремы Поста – Яблонского. Такой базис не является минимальным, так как из него без ущерба для полноты системы ПФ можно исключить конъюнкцию либо дизъюнкцию. При этом получаются системы ПФ, эквивалентные базисам И-НЕ, ИЛИ-НЕ.

Из табл. 3.6 можно получить, например, следующие минимальные базисы.

Базисы, состоящие из одной элементарной ПФ:

1) стрелка Пирса;
2) штрих Шеффера.

Базисы, состоящие из двух элементарных ПФ:

1) отрицание, конъюнкция;
2) константа 0, импликация;
3) отрицание, дизъюнкция.

3.5. Канонические формы аналитического представления ПФ

Конституентой единицы называется функция, принимающая значение 1 только на одном наборе переменных. Конституента единицы записывается как логическое произведение *n* различных переменных, некоторые из которых или все могут быть с отрицаниями.

Конституентой нуля называется функция, принимающая значение 0 на единственном наборе переменных. Конституента нуля записывается как логическая сумма *n* различных переменных, часть которых или все могут быть с отрицаниями.

Любую конституенту 1 можно записать в виде конъюнкции всех ее аргументов, при этом переменная x_i записывается с инверсией, если в наборе, на котором функция принимает значение 1, переменная x_i принимает значение 0. Например, логическое произведение $\bar{x}_1 x_2 \bar{x}_3$ соответствует конституенте 1, принимающей единичное значение на наборе $x_1 = 0$, $x_2 = 1$, $x_3 = 0$. На всех остальных наборах, как следует из определения, конституента 1 принимает значение 0. Вышеприведенное утверждение может быть доказано, например, методом перебора.

Каждому *i*-му набору аргументов $(x_1, x_2, ..., x_n)_i$ поставим в соответствие конституенту единицы K_i, принимающую единичное значение только на этом наборе.

Любую переключательную функцию, зависящую от *n* аргументов, можно представить в виде разложения Шеннона:

$$f(x_1, x_2, ..., x_n) = x_1 f(1, x_2, ..., x_n) \vee \bar{x}_1 f(0, x_2, ..., x_n).$$

Произведем разложение Шеннона по всем переменным:

$$f(x_1, x_2, ..., x_n) = x_1 x_2 ... x_n f(1,1,...,1) \vee x_1 x_2 ... x_{n-1} \bar{x}_n f(1,1,...,1,0) \vee ...$$
$$... \vee \bar{x}_1 \bar{x}_2 ... \bar{x}_{n-1} x_n f(0,0,...,0,1) \vee \bar{x}_1 \bar{x}_2 ... \bar{x}_n f(0,0,...,0).$$

В полученном выражении заменим наборы аргументов, представленных конъюнкциями переменных, на соответствующие конституенты 1. Тогда переключательная функция в общем виде может быть представлена следующим образом:

$$f(x_1, x_2, ..., x_n) = K_0 f_0 \vee K_1 f_1 \vee ... \vee K_n f_n = \bigvee_{i=0}^{n} K_i f_i,$$

где f_i – значение функции на *i*-м наборе.

Воспользовавшись тем, что $0 \cdot x = 0$ и $0 \vee x = x$, в последнем выражении можно исключить слагаемые, для которых $f_i = 0$. Учитывая, что $1 \cdot x = x$, можно не писать коэффициенты $f_i = 1$.

Таким образом, искомую функцию $f(x_1, x_2, ..., x_n)$ можно представить в виде дизъюнкции конституент 1, соответствующих наборам аргументов, на которых она принимает значение 1. Такая форма записи называется *совершенной дизъюнктивной нормальной формой* (СДНФ). Такое название объясняется тем, что каждый дизъюнктивный член включает в себя произведение всех *n* аргументов (совершенная форма) и каждое произведение является элементарным, т.е. под знаком инверсии могут быть только отдельные переменные (нормальная форма).

В качестве примера рассмотрим процедуру записи СДНФ логической функции $f(x_1, x_2, x_3)$, заданной таблицей истинности (табл. 3.7). Для представления этой функции в виде СДНФ необходимо выполнить следующие действия, основанные на вышеприведенных рассуждениях.

Таблица 3.7

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
$f(x_1, x_2, x_3)$	0	1	1	0	1	0	1	1

На первом этапе запишем функцию в виде дизъюнкции конституент единицы, соответствующих тем наборам, на которых функция принимает значение 1:

$$f(x_1, x_2, x_3) = K_1 \vee K_2 \vee K_4 \vee K_6 \vee K_7.$$

Затем, выразив конституенты единицы через произведения аргументов, получим искомую СДНФ:

$$f(x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 x_3 \vee \bar{x}_1 x_2 \bar{x}_3 \vee x_1 \bar{x}_2 \bar{x}_3 \vee x_1 x_2 \bar{x}_3 \vee x_1 x_2 x_3.$$

Из СДНФ функции и СДНФ ее отрицания, используя законы де Моргана, можно получить еще семь канонических форм.

Другой канонической формой записи является *совершенная конъюнктивная нормальная форма* (СКНФ). Ее отличие от СДНФ заключается в том, что СКНФ записывается как конъюнкция конституент нуля, соответствующих тем наборам, на которых функция принимает значение 0. При этом конституенты 0 записываются в виде элементарных дизъюнкций аргументов x_i, инверсии которых соответствуют единичному значению аргумента.

Для примера запишем СКНФ функции, заданной табл. 3.7, при этом выполним последовательность действий, аналогичную той, которая была приведена в предыдущем примере.

На первом этапе запишем функцию в виде конъюнкции конституент нуля, соответствующих тем наборам, на которых функция принимает значение 0:

$$f(x_1, x_2, x_3) = K_0 \cdot K_3 \cdot K_5.$$

Далее, выразив конституенты нуля через дизъюнкции аргументов, получим искомую СКНФ:

$$f(x_1, x_2, x_3) = (x_1 \vee x_2 \vee x_3) \cdot (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \cdot (\bar{x}_1 \vee x_2 \vee \bar{x}_3).$$

С помощью законов де Моргана СКНФ функции можно получить из СДНФ ее отрицания, записываемой по нулям. В качестве примера рассмотрим преобразование отрицания СДНФ функции, заданной табл. 3.7:

$$\overline{f}(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1 x_2 x_3 \vee x_1\bar{x}_2 x_3 = \overline{\overline{\bar{x}_1\bar{x}_2\bar{x}_3} \cdot \overline{\bar{x}_1 x_2 x_3} \cdot \overline{x_1\bar{x}_2 x_3}} =$$
$$= \overline{(x_1 \vee x_2 \vee x_3) \cdot (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \cdot (\bar{x}_1 \vee x_2 \vee \bar{x}_3)},$$

откуда, избавляясь от операции отрицания в начальной и конечной частях выражения, получим

$$f(x_1, x_2, x_3) = (x_1 \vee x_2 \vee x_3) \cdot (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \cdot (\bar{x}_1 \vee x_2 \vee \bar{x}_3).$$

В СДНФ можно заменить операцию дизъюнкции операцией суммы по модулю 2. В том, что равенство при этом сохранится, можно убедиться путем сопоставления таблиц истинности получившихся выражений. Для рассмотренного выше примера:

$$f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2 x_3 \oplus \bar{x}_1 x_2\bar{x}_3 \oplus x_1\bar{x}_2\bar{x}_3 \oplus x_1 x_2\bar{x}_3 \oplus x_1 x_2 x_3.$$

Такая форма называется *совершенной полиномиальной нормальной формой* (СПНФ).

Воспользуемся соотношением $1 \oplus x = \bar{x}$ и заменим в СПНФ все переменные с отрицаниями на суммы вида $(1 \oplus x)$. После упрощения полученного выражения образуется канонический полином Жегалкина, который в общем виде можно записать следующим образом [4]:

$$f(x_1, x_2, ..., x_n) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus ... \oplus a_n x_n \oplus a_{12} x_1 x_2 \oplus a_{13} x_1 x_3 \oplus ...$$
$$... \oplus a_{12...n} x_1 x_{21} ... x_n,\ a_i \in \{0,1\}.$$

Для рассматриваемого примера:

$$f(x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 x_3 \oplus \bar{x}_1 x_2 \bar{x}_3 \oplus x_1 \bar{x}_2 \bar{x}_3 \oplus x_1 x_2 \bar{x}_3 \oplus x_1 x_2 x_3 = (x_1 \oplus 1)(x_2 \oplus 1) x_3 \oplus$$
$$\oplus (x_1 \oplus 1) x_2 (x_3 \oplus 1) \oplus x_1 (x_2 \oplus 1)(x_3 \oplus 1) \oplus x_1 x_2 (x_3 \oplus 1) \oplus x_1 x_2 x_3 = x_1 x_2 x_3 \oplus x_3 \oplus$$
$$\oplus x_1 x_3 \oplus x_2 x_3 \oplus x_1 x_2 x_3 \oplus x_1 x_2 \oplus x_2 x_3 \oplus x_2 \oplus x_1 x_2 x_3 \oplus x_1 x_2 \oplus x_1 x_3 \oplus x_1 \oplus$$
$$\oplus x_1 x_2 x_3 \oplus x_1 x_2 \oplus x_1 x_2 x_3 = x_1 \oplus x_2 \oplus x_3 \oplus x_1 x_2 \oplus x_1 x_2 x_3.$$

3.6. Кубическое представление ПФ

Будем полагать, что каждый набор *n* аргументов ПФ задает вершину *n*-мерного куба и называется 0-кубом. Определив множество 0-кубов, на которых значения функции равны единице, можно получить представление ПФ в виде кубического комплекса K^0. Рассмотрим функцию, заданную табл. 3.7. Она может быть представлена в виде следующего кубического комплекса:

$$K^0(f) = \begin{Bmatrix} 001 \\ 010 \\ 100 \\ 110 \\ 111 \end{Bmatrix},$$

где столбцам соответствуют переменные x_1, x_2, x_3.

Из способа построения кубического комплекса K^0 следует, что каждая ПФ может иметь единственное представление такого вида.

Для ПФ, в общем случае зависящей от *n* аргументов, могут быть построены кубические комплексы размерности: $K^0, K^1, K^2, \ldots, K^n$. При этом каждый комплекс K^i строится по комплексу K^{i-1} $(i = 1,2,...,n)$ путем образования *i*-кубов из $(i-1)$-кубов, отличающихся только по одной переменной. Переменная (координата), по которой отличаются сравниваемые кубы, называется *независимой* и заменяется символом «X».

В качестве примера рассмотрим функцию

$$f_1(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2 x_3 \vee \bar{x}_1 x_2\bar{x}_3 \vee x_1\bar{x}_2\bar{x}_3 \vee x_1 x_2\bar{x}_3 \vee x_1 x_2 x_3.$$

Для нее кубические комплексы K^0, K^1, K^2 и K^3 могут быть построены следующим образом:

$$K^0(f_1) = \begin{Bmatrix} 000 \\ 001 \\ 010 \\ 100 \\ 110 \\ 111 \end{Bmatrix}; \qquad K^1(f_1) = \begin{Bmatrix} 00\text{X} \\ 0\text{X}0 \\ \text{X}00 \\ \text{X}10 \\ 1\text{X}0 \\ 11\text{X} \end{Bmatrix}; \qquad K^2(f_1) = \begin{Bmatrix} 00\text{X} \\ \text{XX}0 \\ 11\text{X} \end{Bmatrix}.$$

Поскольку кубический комплекс $K^2(f_1)$ не содержит 2-кубов, отличающихся только по одной переменной, то кубический комплекс $K^3(f_1)$ будет представлен пустым множеством.

Объединение кубов комплексов $K^0, K^1, K^2, \ldots, K^n$ образует кубический комплекс K. Таким образом,

$$K(f) = \bigvee_{i=1}^{n} K^i(f).$$

Для рассмотренной выше функции $K(f_1)$ можно записать в следующем виде:

$$K(f_1) = \left\{ \begin{matrix} 000 \\ 001 \\ 010 \\ 100 \\ 110 \\ 111 \\ 00X \\ 0X0 \\ X00 \\ X10 \\ 1X0 \\ 11X \\ 00X \\ XX0 \\ 11X \end{matrix} \right\}.$$

3.7. Синтез комбинационных схем

3.7.1. Синтез КС на логических элементах

Для синтеза КС на логических элементах, реализующих элементарные логические функции (дизъюнкцию, конъюнкцию, отрицание, штрих Шеффера и т.п.), удобно использовать аналитическое представление ПФ. В этом случае синтез схем сводится к соединению входов и выходов логических элементов в соответствии с заданными алгебраическими выражениями, т.е. логические элементы выполняют соответствующие логические операции над входными сигналами. При этом многоуровневые выражения могут потребовать многоуровневой схемной реализации.

Рассмотрим пример. Пусть подлежащая реализации функция задана таблицей истинности (табл. 3.8).

Таблица 3.8

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

x_3	0	1	0	1	0	1	0	1
$f(x_1, x_2, x_3)$	0	0	0	1	0	0	1	1

Рассматриваемая функция принимает значение 1 в тех случаях, когда выполняется условие ($x_1 = 1$ и $x_2 = 1$) или условие ($x_2 = 1$ и $x_3 = 1$).

Таким образом, логические выражение, описывающее выход синтезируемой схемы, может быть записано как

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

Для реализации рассматриваемой функции с использованием элементов функционально полного логического базиса «И, ИЛИ, НЕ» согласно приведенному выражению потребуются два логических элемента, реализующих конъюнкции $x_1 x_2$ и $x_2 x_3$, и один логический элемент, реализующий дизъюнкцию их выходов.

Пример КС, реализующей рассматриваемую функцию, приведен на рис. 3.2 (здесь и далее на рисунках изображения элементов схем могут не соответствовать ГОСТ).

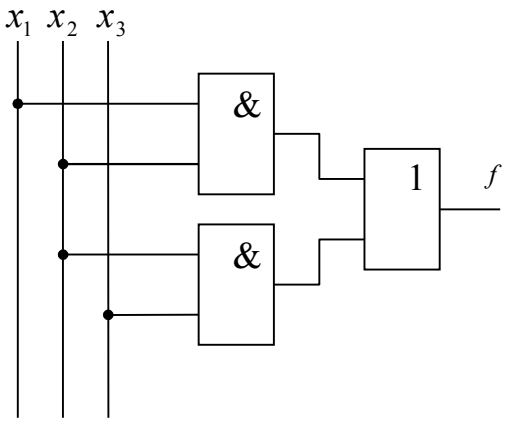


Рис. 3.2

Для реализации рассмотренной ПФ может быть синтезирована более простая схема. Действительно, воспользовавшись для преобразования исходной записи функции дистрибутивным законом (см. раздел 3.3), имеем:

$$f(x_1, x_2, x_3) = x_1 x_2 \vee x_2 x_3 = x_2 \cdot (x_1 \vee x_3) .$$

Отсюда следует, что для реализации заданной функции необходимо использовать двухвходовый логический элемент, реализующий дизъюнкцию $(x_1 \vee x_3)$, выходной сигнал с которого совместно с сигналом x_2 поступает на вход двухвходового логического элемента, реализующего конъюнкцию входов. Данная КС изображена на рис. 3.3.

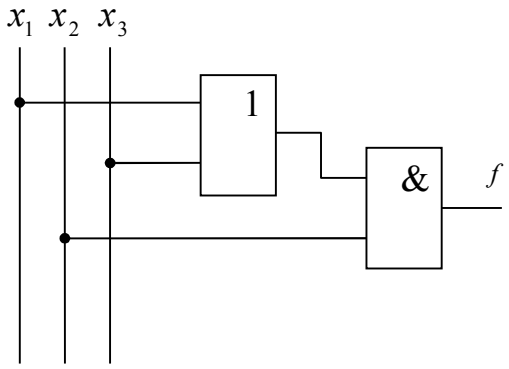


Рис. 3.3

Помимо элементов, реализующих элементарные логические функции, в некоторых случаях более удобным с точки зрения упрощения процедуры синтеза КС является использование элементов, реализуемых в виде интегральных схем средней степени интеграции, выходы которых описываются более сложными выражениями. Упрощение процедуры синтеза достигается за счет универсальности и больших функциональных возможностей по сравнению с простыми логическими элементами. В качестве примера рассмотрим основные этапы синтеза КС с использованием дешифраторов и мультиплексоров.

3.7.2. Синтез КС на дешифраторах

Дешифратором называется комбинационная схема, имеющая n основных входов и $m \le 2^n$ выходов. При $m = 2^n$ дешифратор называется *полным*. Здесь будем рассматривать только *полные дешифраторы*. Помимо основных у дешифратора могут быть дополнительные входы, сигналы на которых разрешают или запрещают реализацию выходных функций. Входы и выходы дешифратора могут быть прямыми или инверсными. В дальнейшем входы и выходы схем будем считать прямыми, если не оговорено иное. Выходы дешифратора со входами связывает следующая система ПФ:

$$\begin{cases} y_0 = f_0(\overline{x_1, x_n}) = \bar{x}_1\bar{x}_2\bar{x}_3 \ldots \bar{x}_n; \\ y_1 = f_1(\overline{x_1, x_n}) = x_1\bar{x}_2\bar{x}_3 \ldots \bar{x}_n; \\ y_2 = f_1(\overline{x_1, x_n}) = \bar{x}_1 x_2\bar{x}_3 \ldots \bar{x}_n; \\ y_3 = f_1(\overline{x_1, x_n}) = x_1 x_2\bar{x}_3 \ldots \bar{x}_n; \\ \vdots \\ y_{2^{n-1}} = f_{2^{n-1}}(\overline{x_1, x_n}) = x_1 x_2 x_3 \ldots x_n. \end{cases}$$

Таким образом, при соответствующем подключении входных сигналов логическая единица формируется на *i*-м выходе дешифратора, если на его входы подана двоичная комбинация, представляющая собой число *i*.

Синтез схем с использованием дешифраторов будем рассматривать на примерах. Пусть требуется реализовать логическую функцию $f = x_1\bar{x}_3 \vee \bar{x}_2\bar{x}_3$ на трехвходовом дешифраторе с одним дополнительным (разрешающим) входом.

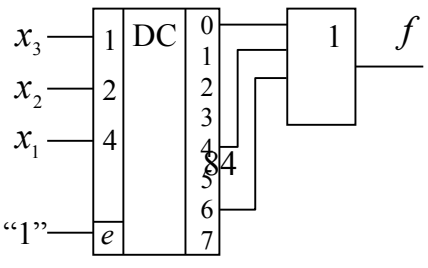
Таблица истинности реализуемой функции имеет вид (табл. 3.9):

Таблица 3.9

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
$f(x_1, x_2, x_3)$	1	0	0	0	1	0	1	0

Для тривиального определения выходов дешифратора, формирующих логические единицы реализуемой функции, необходимо подать сигналы на входы дешифратора таким образом, чтобы весам входов соответствовали веса двоичных разрядов, образующих код набора значений аргументов функции. Пусть в рассматриваемом примере (табл. 3.9) аргумент x_3 имеет наименьший вес (2^0), а аргумент x_1 – наибольший (2^2). Подключая шины входных сигналов к входам дешифратора в порядке возрастания двоичных весов, определим, что единичные сигналы будут формироваться на выходах 0 (двоичный код набора – 000), 4 (100), 6 (110).

Соответствующая схема изображена на рис. 3.4.



Теперь рассмотрим пример, в котором число аргументов реализуемой функции больше числа основных входов дешифратора.

Пусть требуется реализовать функцию $f(\overline{x_1, x_4})$, заданную таблицей истинности (табл. 3.10) с использованием дешифраторов, имеющих три основных и один дополнительный вход.

Очевидно, одного дешифратора с тремя основными входами недостаточно для реализации заданной ПФ. Покажем, как заданная функция может быть реализована на двух дешифраторах.

Таблица истинности заданной функции может быть разбита на две части таким образом, чтобы в первую из них вошли восемь строк, для которых значение одного из аргументов функции $x_i = 0$, а во вторую – другие восемь строк, для которых значение того же аргумента $x_i = 1$. Аргумент можно выбрать любой, поэтому из соображений наглядности выберем x_1. Тогда табл. 3.10 разбивается на две части следующим образом: первые восемь строк подряд (номера наборов 0–7) и вторые восемь строк (номера наборов 8–15). Исключая из рассмотрения столбец x_1, нетрудно заметить, что две сформированные на предыдущем этапе части табл. 3.10 являются по сути таблицами истинности двух ПФ трех аргументов (x_2, x_3, x_4), каждая из которых может быть реализована на дешифраторе с тремя основными входами. Таким образом, исходная функция может быть реализована по частям на двух дешифраторах, если использовать значения аргумента x_1 для разрешения (запрещения) работы схем.

Таблица 3.10

№	x_1	x_2	x_3	x_4	$f(\overline{x_1, x_4})$
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1

5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

Схема, реализующая заданную функцию, приведена на рис. 3.5. Нетрудно заметить, что первая часть функции (табл. 3.10, наборы 0–7, $x_1 = 0$) формируется с помощью дешифратора DC_1, а вторая (наборы 8–15, $x_1 = 1$) – с помощью дешифратора DC_2.

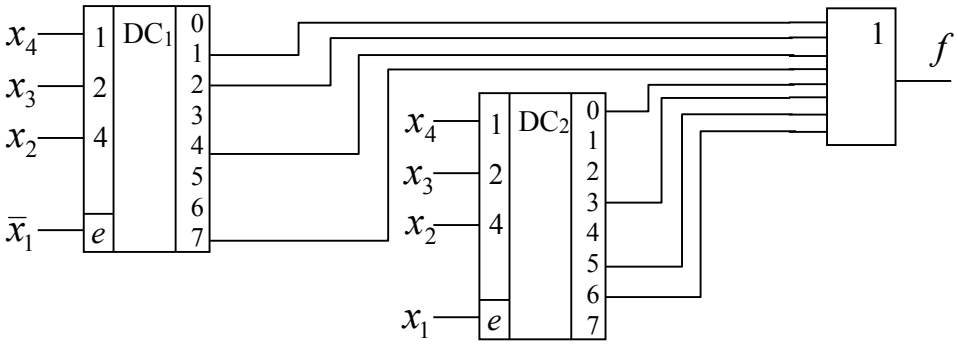


Рис. 3.5

Очевидно, при таком подключении одновременная работа двух дешифраторов невозможна, так как при $x_1 = 1$ (разрешение работы схемы) сигнал $\bar{x}_1 = 0$ и наоборот.

3.7.3. Синтез КС на мультиплексорах

Мультиплексором называется КС, имеющая *m* управляющих входов, 2^m информационных входов и один выход. Входы мультиплексора могут быть прямыми или инверсными. Выходной сигнал мультиплексора

совпадает с сигналом на *i*-м информационном входе, если на его управляющие входы подан двоичный код числа *i*.

Обозначим управляющие входы мультиплексора – $A_1, A_2, \ldots, A_m$, а информационные – $B_0, B_1, \ldots, B_{2^m-1}$. Тогда выходная функция мультиплексора может быть записана следующим образом:

$$F = B_0 \overline{A}_1 \overline{A}_2 \ldots \overline{A}_m \vee B_1 A_1 \overline{A}_2 \ldots \overline{A}_m \vee \ldots \vee B_{2^m-1} A_1 A_2 \ldots A_m .$$

Пусть требуется синтезировать КС, реализующую некоторую функцию $f(x_1, x_2, ..., x_{m+1})$. Подадим сигналы $x_1, x_2, \ldots, x_m$ на управляющие входы $A_1, A_2, \ldots, A_m$. Определим значения сигналов, которые необходимо подавать на информационные входы. Для этого в таблице истинности функции $f(x_1, x_2, ..., x_{m+1})$ выделим пары строк, отличающиеся только значением аргумента x_{m+1}. Сравним для каждой пары значение функции со значением x_{m+1}. Результат сравнения определяет, что должно быть подано на информационный вход мультиплексора, номер которого определяется значениями сигналов на управляющих входах. Для того, чтобы номера информационных входов соответствовали двоичным числам, определяемым совокупностями значений сигналов на управляющих входах, эти сигналы должны быть поданы в соответствии с двоичными весами входов: $2^0, 2^1, \ldots, 2^m$.

Рассмотрим пример. Пусть переключательная функция задана табл. 3.11. На рис. 3.6 изображен мультиплексор, реализующий эту функцию.

Таблица 3.11

$x_1 x_2 x_3$	$f(\overline{x_1, x_3})$	Информационный вход
0 0 0	1	B_0
0 0 1	0	B_0
0 1 0	1	B_1
0 1 1	1	B_1
1 0 0	0	B_2
1 0 1	0	B_2
1 1 0	0	B_3
1 1 1	1	B_3

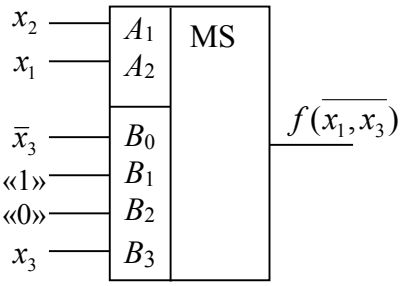


Рис. 3.6

Комбинации значений сигналов x_1, x_2 определяют номер информационного входа, сигнал с которого проходит на выход мультиплексора. Рассмотрим первую пару наборов и определим, что следует подать на информационный вход B_0, чтобы на выходе мультиплексора формировались требуемые значения функции. Нетрудно видеть, что в рассматриваемой паре строк таблицы истинности значения реализуемой ПФ противоположны значениям аргумента x_3. Следовательно, на информационный вход B_0 необходимо подавать инвертированное значение данного сигнала, т.е. $\bar{x}_3$ (рис. 3.6). Во второй и третьей парах строк табл. 3.11 реализуемая ПФ вне зависимости от x_3 принимает значения 1 (вторая пара строк) и значения 0 (третья пара строк). Следовательно, на информационные входы B_1 и B_2 необходимо подавать логическую единицу и логический нуль соответственно. В последних двух строках таблицы значения реализуемой функции совпадают со значениями x_3, следовательно, на информационный вход B_3 необходимо подавать прямое значение x_3.

При наличии у мультиплексора дополнительных разрешающих входов, для реализации ПФ *n* аргументов могут быть использованы мультиплексоры с числом управляющих входов меньше $(n-1)$. В этом случае, очевидно, как в рассмотренном выше примере с дешифраторами, для реализации заданной функции необходимо использовать несколько мульти-плексоров, разрешая (запрещая) их работу соответствующими комбинациями сигналов, т.е. разбивать таблицу истинности на части,

каждая из которых реализуется на отдельном мультиплексоре с последующим объединением их выходов.

3.7.4. Синтез многовыходных схем

Выше были рассмотрены примеры синтеза одновыходных схем, выходные реакции которых описываются отдельными логическими выражениями – ПФ. Для описания выходных реакций многовыходных схем можно поступить следующим образом: для каждого выхода схемы составить ПФ, определяющую зависимость значений формируемых выходных сигналов от значений сигналов, поступающих на входы, и синтезировать КС по этим ПФ, пользуясь вышеописанными подходами. В этом случае, очевидно, КС будет состоять из нескольких обособленных подсхем, имеющих общие входы (общие шины входных сигналов).

Рассмотренный выше прием не является рациональным, поскольку не учитывает структурные особенности синтезируемой КС. Другим способом описания и синтеза многовыходной КС является объединение отдельных ПФ в систему и дальнейшее преобразование этой системы ПФ таким образом, чтобы с учетом структурных особенностей цена синтезируемой схемы уменьшалась. Примеры синтеза многовыходных КС можно найти в разделе 4.7.

Для уменьшения стоимости разрабатываемого цифрового устройства синтез КС следует осуществлять, пользуясь минимальными или близкими к минимальным формами представления ПФ и систем ПФ.

3.8. Риски сбоя в комбинационных схемах

Рассмотрим работу КС, изображенной на рис. 3.7, реализующей функцию $f(\overline{x_1, x_3}) = x_1 x_2 \vee \bar{x}_1 x_3$.

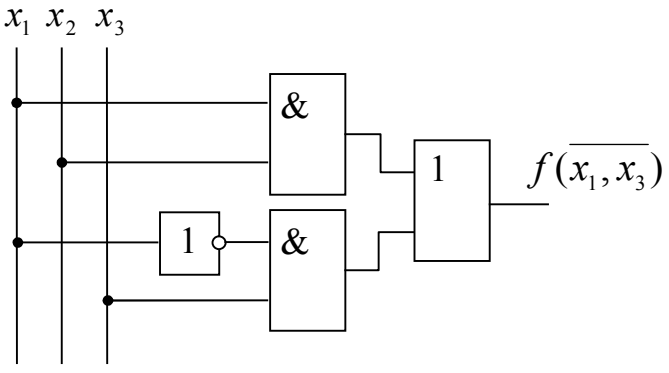


Рис. 3.7

Пусть на входах схемы последовательно формируются следующие сигналы: $x_2 = x_3 = 1$, $x_1 = 1 \to 0$, т.е. сигналы на входах x_2 и x_3 остаются неизменными, а сигнал на входе x_1 изменяется из 1 в 0. Логические элементы схемы вносят различные задержки в процесс распространения сигналов. На рис. 3.8 приведены временные диаграммы для возможного изменения сигналов в рассматриваемой схеме.

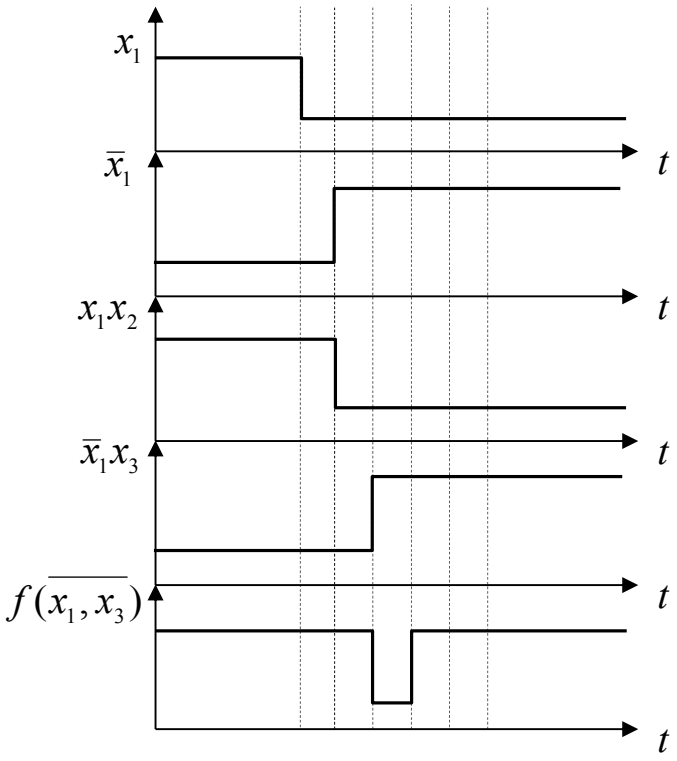


Рис. 3.8

Несмотря на то, что для рассматриваемой функции и реализующей ее схемы $f(1,1,1) = f(1,1,0) = 1$, в течение некоторого времени на выходе схемы может быть сформирован уровень логического нуля, являющийся в данном случае ложным сигналом. Появление ложных сигналов на выходах схемы называется *риском сбоя*. Риски сбоя являются следствием конечной скорости распространения сигналов и ненулевых задержек логических элементов.

Различают статические и динамические риски сбоя. При статических рисках сбоя во время переходного процесса значение на выходе схемы изменяется четное число раз. При динамических рисках сбоя, являющихся следствием статических, – нечетное число раз.

Риски сбоя в схемах могут вызывать ложные срабатывания управляемых устройств, для которых сигналы КС являются входными.

С целью исключения рисков сбоя в схемах, для которых в любой момент времени может изменяться значение только одного входного сигнала, необходимо выполнять их синтез таким образом, чтобы любые пары единиц функции, которым соответствуют конституенты 1, отличающиеся по одной переменной, были представлены произведением в ДНФ. Для этих целей удобно использовать таблично-графическое представление ПФ, называемое картой Карно (диаграммой Вейча). Карты Карно подробно рассмотрены в разделе 4.1 учебного пособия.

Приведем карту Карно для рассматриваемой функции (табл. 3.12).

Таблица 3.12

x_1 \ x_2				
	1	1	0	0
	0	1	1	0

x_3

Таким образом, действуя по приведенному выше правилу, для исключения рисков сбоя в схеме, реализующей данную функцию, необходимо ввести в ДНФ функции конъюнкцию x_2x_3, а в КС – соответствующий этому произведению дополнительный логический элемент.

Рассмотрим еще один способ устранения рисков сбоя для ситуации, когда изменяются более одного входного сигнала. В этом случае ячейки карты Карно, соответствующие реализуемым единичным сигналам функции, не будут являться соседними.

Пусть реализуемая функция имеет вид $f(\overline{x_1, x_3}) = x_2x_3 \vee \bar{x}_1x_3$. Соответствующая карта Карно представлена на рис. 3.9. Предположим, что в процессе работы схемы на ее входах последовательно появляются наборы (1,1,1) и (0,0,1).

Поскольку абсолютно одновременное изменение сигналов на входе КС невозможно, переход от единицы функции, реализуемой на первом из рассматриваемых наборов входных сигналов, к единице, соответствующей второму набору, может быть выполнен одним из двух способов.

x_1 \ x_2				
	0	1	0	0

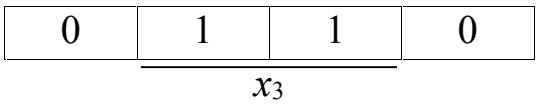


Рис. 3.9

В первом случае, если входной сигнал x_1 изменится быстрее x_2, произойдет переход $x_1x_2x_3 \to \bar{x}_1x_2x_3 \to \bar{x}_1\bar{x}_2x_3$ и на выходе схемы единичный сигнал не изменится. В другом случае сигнал x_2 изменится быстрее x_1 и произойдет переход $x_1x_2x_3 \to x_1\bar{x}_2x_3 \to \bar{x}_1\bar{x}_2x_3$. Поскольку $f(1,0,1)=0$, в данной ситуации может возникнуть изменение выходного сигнала вида $1 \to 0 \to 1$, указывающее наличие в схеме риска сбоя. Рассмотренное явление носит название *функционального риска сбоя*.

В некоторых случаях подобные риски сбоя могут быть устранены. Если, например, в рассматриваемом примере работа схемы организована таким образом, что после входного набора (1, 1, 1) следует набор (0, 0, 1) и отсутствует обратная последовательность, то введение достаточной задержки в процесс изменения сигнала x_2 относительно изменения x_1 может обеспечить требуемый безопасный вариант переключения схемы.

Вопросы для самоконтроля

1. Каково максимальное число ПФ, зависящих от *n* переменных?
2. Какие способы задания ПФ вам известны?
3. Какие элементарные логические функции вам известны и для чего они используются?
4. Каким условиям должен удовлетворять набор логических функций, образующий функционально полную систему (базис)?
5. Какие канонические формы аналитического представления ПФ вам известны и как они формируются?
6. Что представляют собой дешифраторы и мультиплексоры и как с их помощью могут быть реализованы переключательные функции?
7. Что понимают под рисками сбоя и каковы причины их появления?

4. МИНИМИЗАЦИЯ ПЕРЕКЛЮЧАТЕЛЬНЫХ ФУНКЦИЙ

Рассмотрим СДНФ некоторой функции $f_1(x_1, x_2, x_3)$ и выполним ряд элементарных преобразований, воспользовавшись соотношениями, приведенными в разделе 3.3.

$$f_1(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 \vee \overline{x}_1\overline{x}_2\overline{x}_3 \vee \overline{x}_1\overline{x}_2\overline{x}_3 \vee x_1\overline{x}_2\overline{x}_3 \vee x_1x_2\overline{x}_3 \vee x_1x_2x_3 =$$
$$= \overline{x}_1\overline{x}_2(\overline{x}_3 \vee x_3) \vee \overline{x}_1\overline{x}_3(\overline{x}_2 \vee x_2) \vee \overline{x}_2\overline{x}_3(\overline{x}_1 \vee x_1) \vee x_1x_2(\overline{x}_3 \vee x_3) =$$
$$= \overline{x}_1\overline{x}_2 \vee \overline{x}_1\overline{x}_3 \vee \overline{x}_2\overline{x}_3 \vee x_1x_2 = \overline{x}_1\overline{x}_2 \vee \overline{x}_3(\overline{x}_1 \vee \overline{x}_2) \vee x_1x_2 =$$
$$= \overline{x}_1\overline{x}_2 \vee \overline{x}_3(\overline{x_1x_2}) \vee x_1x_2 = \overline{x}_1\overline{x}_2 \vee (x_1x_2 \vee \overline{x}_3)(x_1x_2 \vee \overline{x_1x_2}) = \overline{x}_1\overline{x}_2 \vee x_1x_2 \vee \overline{x}_3.$$

Справедливость полученного равенства можно проверить, сравнив, например, таблицы истинности для исходной и конечной записи ПФ.

Для технической реализации функции $f_1(x_1, x_2, x_3)$ по исходной СДНФ в каноническом базисе И-ИЛИ-НЕ необходимо использовать шесть трехвходовых элементов «И», реализующих конъюнкции переменных, три элемента «НЕ», реализующих инверсии переменных, и один шестивходовый элемент «ИЛИ», реализующий дизъюнкцию всех заданных конъюнкций входных переменных. Для реализации той же ПФ по аналитической записи, полученной после выполнения преобразований, потребуются два двухвходовых элемента «И», три инвертора и один трехвходовый элемент «ИЛИ».

Таким образом, если принять за единицу цены схемы один логический элемент, используемый при технической реализации, и один вход логического элемента, то можно дать оценку сложности технической реализации КС путем суммирования количества необходимых для синтеза схемы логических элементов и количества их входов. Возможны и другие методы оценки стоимости технической реализации КС.

Для рассмотренной ПФ цена схемы, синтезируемой по СДНФ, составляет (без учета инверторов): $C_1 = 6+(6\cdot3)+1+6 = 31$. Цена схемы, синтезируемой по упрощенной форме представления ПФ, составляет (также без учета инверторов): $C_2 = 2+(2\cdot2)+1+3 = 10$.

Задача минимизации ПФ заключается в отыскании минимальных форм представления функций, поскольку им соответствуют наиболее простые технические реализации (с наименьшей ценой схемы). Иными словами, при решении задачи минимизации ПФ требуется найти аналитическое выражение заданной ПФ или системы ПФ, содержащее

наименьшее число переменных и логических функций. Решение этой задачи в общем случае затруднено. Наиболее подробно исследовано решение задачи минимизации для канонического базиса И-ИЛИ-НЕ. Также известны подходы к решению этой задачи для ПФ, заданных в базисах И-НЕ, ИЛИ-НЕ.

Введем ряд определений.

Элементарной конъюнкцией называется конъюнкция конечного числа различных переменных, входящих в нее в прямом или инверсном значении.

Дизъюнктивной нормальной формой (ДНФ) называется дизъюнкция элементарных конъюнкций.

ДНФ переключательной функции называется *минимальной*, если она содержит наименьшее число букв по отношению к любой другой ДНФ той же функции. Следует различать буквы и переменные. Например, ДНФ $x_1x_2 \vee x_1\bar{x}_2x_3 \vee x_2x_3$ содержит семь букв.

Если некоторая булева функция g принимает значение 0 на тех же наборах, на которых принимает значение 0 другая булева функция f, то функция g называется *импликантой функции f*. Это определение можно также сформулировать в другом виде. Функция g называется импликантой функции f, если для любого набора переменных, на котором $g = 1$, функция f также принимает значение 1.

Простой импликантой переключательной функции f называется импликанта g, если никакие отдельные части этой импликанты не являются сами по себе импликантами f.

Проиллюстрируем последние два определения примером. В табл. 4.1 заданы функция f и ее импликанты $g_1, g_2, \ldots, g_7$.

Таблица 4.1

x_1	x_2	x_3	$\boldsymbol{f}$	g_1	g_2	g_3	g_4	g_5	g_6	g_7
0	0	0	**0**	0	0	0	0	0	0	0
0	0	1	**1**	0	0	0	1	1	1	1
0	1	0	**1**	0	1	1	0	0	1	1
0	1	1	**0**	0	0	0	0	0	0	0
1	0	0	**0**	0	0	0	0	0	0	0
1	0	1	**1**	1	0	1	0	1	0	1
1	1	0	**0**	0	0	0	0	0	0	0
1	1	1	**0**	0	0	0	0	0	0	0

$$f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2x_3;$$

$$g_1 = x_1\bar{x}_2x_3;$$

$$g_2 = \bar{x}_1x_2\bar{x}_3;$$

$$g_3 = \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2x_3;$$

$$g_4 = \bar{x}_1\bar{x}_2x_3;$$

$$g_5 = \bar{x}_1\bar{x}_2x_3 \vee x_1\bar{x}_2x_3 = \bar{x}_2x_3(\bar{x}_1 \vee x_1) = \bar{x}_2x_3;$$

$$g_6 = \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3;$$

$$g_7 = \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2x_3.$$

Из приведенных определений следует, что импликанты g_2 и g_5 являются простыми, а, например, импликанта g_4 не является простой, так как ее часть $\bar{x}_2x_3$ также является импликантой функции *f*.

Из определения импликанты следует, что дизъюнкция любого числа импликант функции *f* также является импликантой этой функции.

Сокращенной ДНФ называется форма представления переключательной функции *f*, записываемая в виде дизъюнкции всех простых импли- кант *f*.

Для ПФ из рассмотренного выше примера сокращенная ДНФ запишется следующим образом:

$$f(x_1, x_2, x_3) = g_2 \vee g_5 = \bar{x}_1x_2\bar{x}_3 \vee \bar{x}_2x_3.$$

В некоторых случаях из сокращенной ДНФ можно исключить одну или несколько простых импликант таким образом, чтобы полученная функция была эквивалентна исходной. Такие простые импликанты называются *лишними*.

Сокращенная ДНФ булевой функции называется *тупиковой* (ТДНФ), если из нее не удается исключить лишние простые импликанты. Произвольная ПФ может иметь несколько ТДНФ.

Тупиковые ДНФ логической функции, содержащие минимальное число букв, являются минимальными. Произвольная ПФ может иметь несколько минимальных ДНФ.

Процесс отыскания минимальных ДНФ состоит из двух этапов: нахождения простых импликант и исключения лишних импликант.

4.1. Минимизация ПФ с помощью карт Карно

Карта Карно (диаграмма Вейча) для функции $f(x_1, x_2, ..., x_n)$ представляет собой прямоугольную таблицу, состоящую из 2^n клеток, где каждой строке (столбцу) или группе строк (столбцов) соответствует прямое или инверсное значение одной из переменных. Каждой клетке такой таблицы ставится в соответствие набор аргументов функции. В клетке карты Карно записывается единица, если функция $f(x_1, x_2, ..., x_n)$ на соответствующем этой клетке наборе принимает значение 1, в противном случае в соответствующую клетку записывается 0. Таким образом, карта Карно является разновидностью таблицы истинности булевой функции.

В табл. 4.2–4.5 приведены карты Карно в общем виде для функций двух, трех, четырех и пяти аргументов. В ячейках этих карт Карно записаны соответствующие им наборы аргументов.

Таблица 4.2

$x_1 \backslash x_2$	
$x_1 x_2$	$x_1 \bar{x}_2$
$\bar{x}_1 x_2$	$\bar{x}_1 \bar{x}_2$

Таблица 4.3

$x_1 \backslash x_2$			
$x_1 x_2 \bar{x}_3$	$x_1 x_2 x_3$	$x_1 \bar{x}_2 x_3$	$x_1 \bar{x}_2 \bar{x}_3$
$\bar{x}_1 x_2 \bar{x}_3$	$\bar{x}_1 x_2 x_3$	$\bar{x}_1 \bar{x}_2 x_3$	$\bar{x}_1 \bar{x}_2 \bar{x}_3$
	x_3		

Таблица 4.4

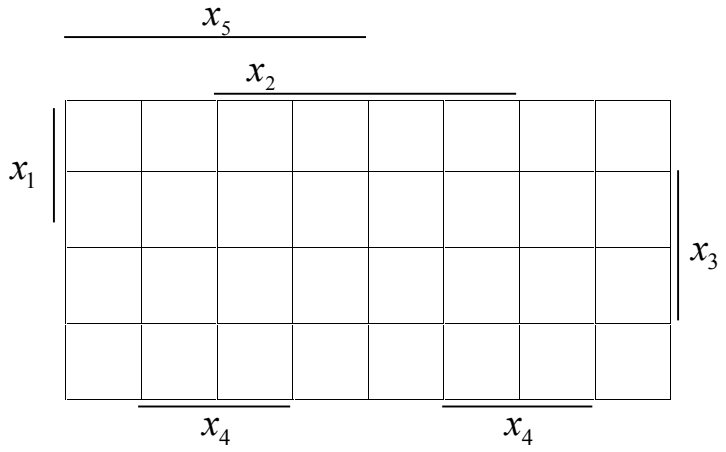
x_1 \ x_2			
$x_1 x_2 \bar{x}_3 \bar{x}_4$	$x_1 x_2 \bar{x}_3 x_4$	$x_1 \bar{x}_2 \bar{x}_3 x_4$	$x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4$
$x_1 x_2 x_3 \bar{x}_4$	$x_1 x_2 x_3 x_4$	$x_1 \bar{x}_2 x_3 x_4$	$x_1 \bar{x}_2 x_3 \bar{x}_4$
$\bar{x}_1 x_2 x_3 \bar{x}_4$	$\bar{x}_1 x_2 x_3 x_4$	$\bar{x}_1 \bar{x}_2 x_3 x_4$	$\bar{x}_1 \bar{x}_2 x_3 \bar{x}_4$

x_3

$\bar{x}_1 x_2 \bar{x}_3 \bar{x}_4$	$\bar{x}_1 x_2 \bar{x}_3 x_4$	$\bar{x}_1 \bar{x}_2 \bar{x}_3 x_4$	$\bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4$

x_4

Таблица 4.5



Карты Карно для функций большего числа аргументов строятся аналогичным образом. При этом в любой карте Карно соседние наборы, отличающиеся значением только одной переменной, находятся в соседних клетках. Крайние столбцы и строки карты Карно также содержат соседние наборы.

Метод минимизации ПФ с помощью карт Карно основан на так называемой операции полного склеивания: $x_1 \bar{x}_2 \vee x_1 x_2 = x_1$. При этом говорят, что два дизъюнктивных члена склеиваются по переменной x_2. Таким образом, соседние наборы склеиваются по одной из переменных.

Общее правило склеивания на картах Карно можно сформулировать следующим образом: склеиванию подлежат совокупности соседних ячеек, содержащих одинаковые значения заданной ПФ, общее число которых кратно 2^k, $k = 1, 2, ..., n$. Результатом такого склеивания является элементарное произведение переменных, значения которых не изменялись на всех склеиваемых наборах.

Для получения минимальной ДНФ по карте Карно необходимо выполнить следующие действия.

1. Определить совокупности 2^k соседних ячеек с единичным значением функции. Необходимо стремиться объединить в одну совокупность как можно большее количество соседних ячеек карты, а общее число совокупностей должно быть минимальным. При этом

некоторые единицы уже выделенных совокупностей могут участвовать в формировании других совокупностей соседних ячеек.

2. Определить элементарные произведения, получающиеся путем склеивания соседних ячеек, определенных на предыдущем шаге.

3. Записать минимальную ДНФ функции как дизъюнкцию полученных элементарных конъюнкций.

Рассмотрим несколько примеров минимизации по картам Карно.

Пример 1. Минимизировать с помощью карты Карно ПФ, заданную следующим выражением:

$$f(x_1, x_2, x_3) = \bar{x}_1 x_2 x_3 \vee \bar{x}_1 x_2 \bar{x}_3 \vee x_1 \bar{x}_2 \bar{x}_3 \vee x_1 x_2 \bar{x}_3 \vee x_1 x_2 x_3.$$

Карта Карно для этой функции имеет вид, представленный в табл. 4.6.

Таблица 4.6

x_1 \ x_2				
	1	1	0	1
	1	1	0	0

x_3

Прямоугольниками обозначены совокупности соседних 2^k ячеек, выделяемых для операции склеивания. Единица в ячейке $x_1 \bar{x}_2 \bar{x}_3$ (крайний правый столбец) может объединяться с единицей $x_1 x_2 \bar{x}_3$, поскольку эти две ячейки являются соседними.

Для определения результатов склеивания достаточно записать элементарные произведения переменных, которые накрывают выделенные совокупности ячеек. Из табл. 4.5 следует, что совокупность из четырех единиц (наборы $x_1 x_2 \bar{x}_3$, $x_1 x_2 x_3$, $\bar{x}_1 x_2 \bar{x}_3$, $\bar{x}_1 x_2 x_3$) может быть накрыта только переменной x_2, а остальные две переменные накрывают только часть этой совокупности, поэтому первым элементарным произведением будет x_2. Вторая выделенная совокупность соседних ячеек (наборы $x_1 \bar{x}_2 \bar{x}_3$, $x_1 x_2 \bar{x}_3$) может быть накрыта элементарным произведением $x_1 \bar{x}_3$, поскольку эти две ячейки расположены в строке, отмеченной переменной x_1, и столбцы, соответствующие этой совокупности ячеек, отмечены инверсным значением переменной x_3.

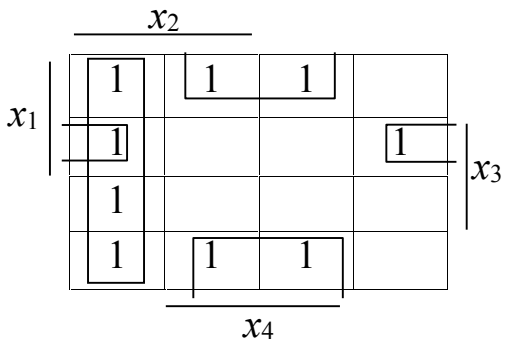
Таким образом, минимальная ДНФ для заданной функции имеет вид

$$f(x_1, x_2, x_3) = x_2 \vee x_1 \bar{x}_3.$$

Рассмотрим еще несколько примеров получения минимальной ДНФ по карте Карно. При этом будем задавать исходные ПФ в виде карт Карно, так как процедура перехода от СДНФ к карте и наоборот выполняется элементарно. Также для простоты будем записывать в ячейки только единичные значения функции, а ячейки с нулевыми значениями будем оставлять незаполненными.

Пусть минимизируемая функция задана картой Карно (табл. 4.7).

Таблица 4.7

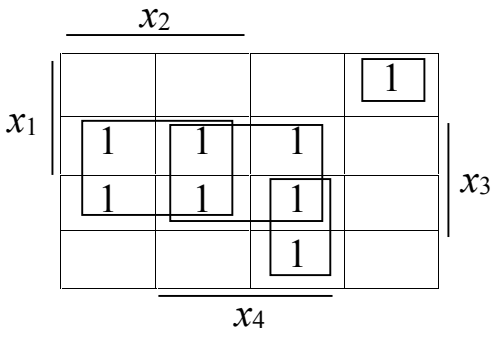


Поскольку минимальное число совокупностей соседних ячеек, объединяющих все единицы исходной функции, равно трем, минимальная ДНФ для функции, заданной табл. 4.6, включает в себя три слагаемых:

$$f(x_1, x_2, x_3, x_4) = x_2 \bar{x}_4 \vee \bar{x}_3 x_4 \vee x_1 x_3 \bar{x}_4.$$

Пусть минимизируемая функция задана картой Карно (табл. 4.8).

Таблица 4.8



Поскольку единица в ячейке $x_1\bar{x}_2\bar{x}_3\bar{x}_4$ не имеет подходящих для склеивания соседей, она входит в минимальную ДНФ в виде элементарного произведения всех четырех аргументов:

$$f(x_1, x_2, x_3, x_4) = x_3x_4 \vee x_2x_3 \vee \bar{x}_1\bar{x}_2x_4 \vee x_1\bar{x}_2\bar{x}_3\bar{x}_4 .$$

4.2. Минимизация ПФ методом Квайна

Идея минимизации ПФ методом Квайна заключается в применении двух соотношений:

1) $x_1\bar{x}_2 \vee x_1x_2 = x_1 \vee x_1\bar{x}_2 \vee x_1x_2$ – соотношение (неполного) склеивания;
2) $x_1x_2 \vee x_1 = x_1, \quad x_2 \in \{x_2, \bar{x}_2\}$ – соотношение поглощения.

Справедливость этих соотношений можно проверить при помощи элементарных преобразований или по таблицам истинности.

Теорема Квайна. Если в совершенной дизъюнктивной нормальной форме ПФ провести все операции неполного склеивания и затем все операции поглощения, то в результате получится сокращенная ДНФ исходной ПФ, т.е. дизъюнкция всех ее простых импликант.

Доказательство этой теоремы приводится, например, в [7], при этом используется операция развертывания, обратная операции склеивания:

$$x_1(\bar{x}_2 \vee x_2) = x_1\bar{x}_2 \vee x_1x_2 .$$

Из формулировки теоремы Квайна следует, что в качестве исходной формы представления ПФ должна использоваться СДНФ. Если функция задана в произвольной форме, то вначале ее следует преобразовать к СДНФ. При этом переход от произвольной ДНФ к СДНФ осуществляется путем применения операции развертывания.

Практическое применение метода Квайна можно условно разделить на два этапа. На первом этапе минимизации производятся все возможные склеивания и поглощения по заданной СДНФ минимизируемой функции. В результате получается сокращенная ДНФ.

На втором этапе для получения минимальной ДНФ необходимо исключить из сокращенной ДНФ все лишние простые импликанты. Это делается с помощью так называемой импликантной матрицы Квайна, которая представляет собой таблицу, столбцы которой отмечаются

простыми импликантами, т.е. дизъюнктивными членами полученной на предыдущем этапе сокращенной ДНФ, а строки – членами СДНФ минимизируемой ПФ. При этом в ячейке импликантной матрицы на пересечении *i*-й строки и *j*-го столбца ставится отметка, если соответствующая этому столбцу импликанта является частью члена СДНФ, которому соответствует *j*-я строка.

Минимальная ДНФ строится по импликантной матрице следующим образом. Анализируются все возможные совокупности простых импликант (столбцы матрицы), накрывающие все члены СДНФ (строки матрицы). Простая импликанта накрывает член СДНФ, если на пересечении соответствующей строки и столбца матрицы имеется отметка. Одна импликанта может накрывать несколько строк. Из этих совокупностей выбирается одна (или несколько) с наименьшим числом членов (простых импликант), которая определяет минимальную ДНФ.

Для сокращения перебора можно пользоваться следующим подходом. Вначале по возможности определяются строки импликантной матрицы, в которых имеется только одна отметка. Простые импликанты, которым соответствуют эти отметки, называются *базисными* и обязательно входят в минимальную ДНФ функции, так как только эти простые импликанты в минимальной ДНФ представляют члены исходной СДНФ, записанные в найденных строках. Остальные строки импликантной матрицы должны быть накрыты минимальным числом простых импликант. В некоторых случаях базисные импликанты могут накрывать все строки.

Рассмотрим пример.

Пусть ПФ задана картой Карно (табл. 4.6). Ее СДНФ имеет вид

$$f(x_1, x_2, x_3, x_4) = x_1 x_2 \bar{x}_3 \bar{x}_4 \vee x_1 x_2 x_3 \bar{x}_4 \vee \bar{x}_1 x_2 x_3 \bar{x}_4 \vee \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 \vee x_1 x_2 \bar{x}_3 x_4 \vee$$
$$\vee \bar{x}_1 x_2 \bar{x}_3 x_4 \vee x_1 \bar{x}_2 \bar{x}_3 x_4 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 x_4 \vee x_1 \bar{x}_2 x_3 \bar{x}_4.$$

На первом этапе выполним все склеивания конституент 1 по одной переменной. При этом будет получена совокупность элементарных произведений, являющихся импликантами и представляющих собой общие части склеиваемых конституент. Для удобства присвоим каждому получаемому элементарному произведению порядковый номер. Не участвовавшие в очередном склеивании конституенты 1 или импликанты

должны входить в последующие совокупности импликант в неизмененном виде.

1) $x_1 x_2 \bar{x}_4$; 6) $\bar{x}_1 x_2 \bar{x}_4$; 11) $\bar{x}_2 \bar{x}_3 x_4$.
2) $x_2 \bar{x}_3 \bar{x}_4$; 7) $\bar{x}_1 x_2 \bar{x}_3$;
3) $x_1 x_2 \bar{x}_3$; 8) $x_2 \bar{x}_3 x_4$;
4) $x_2 x_3 \bar{x}_4$; 9) $x_1 \bar{x}_3 x_4$;
5) $x_1 x_3 \bar{x}_4$; 10) $\bar{x}_1 \bar{x}_3 x_4$;

Повторим процедуру склеивания по одной переменной над каждой парой элементов совокупности импликант, полученных на предыдущем шаге. Будем повторять эту процедуру до тех пор, пока остаются элементарные произведения, между которыми возможно склеивание. Каждому элементу новой совокупности поставим в соответствие индекс, составленный из индексов импликант, участвующих в склеивании.

1,6) $x_2 \bar{x}_4$;
2,4) $x_2 \bar{x}_4$;
2,8) $x_2 \bar{x}_3$;
3,7) $x_2 \bar{x}_3$;
8,11) $\bar{x}_3 x_4$;
9,10) $\bar{x}_3 x_4$;
5) $x_1 x_3 \bar{x}_4$.

Полученную совокупность необходимо дополнить определенной на предыдущем этапе импликантой с порядковым номером 5, так как эта импликанта не участвовала в последующем склеивании (является простой).

Над полученной совокупностью элементарных произведений необходимо выполнить процедуру поглощения. В рассматриваемом примере совокупность элементарных произведений не содержит элементов, способных поглощать другие элементы, за исключением повторяющихся.

Окончательно, после исключения повторяющихся элементов, совокупность импликант запишется в следующем виде:

$x_2\bar{x}_4$;

$x_2\bar{x}_3$;

$\bar{x}_3x_4$;

$x_1x_3\bar{x}_4$.

Дальнейшее выполнение склеиваний и поглощений невозможно.

Запишем сокращенную ДНФ минимизируемой функции:

$$f(x_1, x_2, x_3, x_4) = x_2\bar{x}_4 \vee x_2\bar{x}_3 \vee \bar{x}_3x_4 \vee x_1x_3\bar{x}_4.$$

Построим импликантную матрицу Квайна (табл. 4.9).

Таблица 4.9

№	Конституенты 1	Простые импликанты			
		$x_2\bar{x}_4$	$x_2\bar{x}_3$	$\bar{x}_3x_4$	$x_1x_3\bar{x}_4$
1	$x_1x_2\bar{x}_3\bar{x}_4$	+	+		
2	$x_1x_2x_3\bar{x}_4$	+			+
3	$\bar{x}_1x_2x_3\bar{x}_4$	+			
4	$\bar{x}_1x_2\bar{x}_3\bar{x}_4$	+	+		
5	$x_1x_2\bar{x}_3x_4$		+	+	
6	$\bar{x}_1x_2\bar{x}_3x_4$		+	+	
7	$x_1\bar{x}_2\bar{x}_3x_4$			+	
8	$\bar{x}_1\bar{x}_2\bar{x}_3x_4$			+	
9	$x_1\bar{x}_2x_3\bar{x}_4$				+

Поставим отметки в ячейках импликантной матрицы, находящихся на пересечении таких строк и столбцов, в которых простая импликанта является частью соответствующей конституенты 1.

Отыщем минимальное покрытие столбцами всех строк таблицы. Столбец импликантной матрицы, которому соответствует простая импликанта $\bar{x}_3x_4$, обязательно должен войти в минимальное покрытие, поскольку только эта импликанта накрывает строки 7 и 8 матрицы.

Кроме того, простая импликанта $\bar{x}_3x_4$ может представлять в минимальной совокупности еще две конституенты 1, находящиеся в строках 5 и 6.

Аналогичные рассуждения можно провести в отношении простой импликанты $x_1x_3\bar{x}_4$, поскольку только она может представлять в минимальной совокупности простых импликант строку 9 импликантной

матрицы. Отметим, что эта импликанта также накрывает конституенту 1, записанную в строке 2.

Таким образом, задача упростилась и заключается в нахождении минимального покрытия оставшихся трех конституент 1 (строки 1, 3, 4) оставшимися простыми импликантами.

Очевидно, что простая импликанта $x_2\bar{x}_4$ накрывает оставшиеся три строки матрицы. Следовательно, простая импликанта $x_2\bar{x}_3$ оказалась лишней и в минимальную ДНФ не входит.

Окончательно минимальная ДНФ запишется в виде

$$f(x_1, x_2, x_3, x_4) = x_2\bar{x}_4 \vee \bar{x}_3 x_4 \vee x_1 x_3 \bar{x}_4 .$$

Метод Квайна является базовым методом минимизации ПФ.

4.3. Минимизация методом Квайна – Мак-Класки

Метод Квайна – Мак-Класки отличается от метода Квайна большей формализацией. Это достигается путем использования кубического представления ПФ (см. п. 3.6 учебного пособия) и сокращения перебора при выполнении операции склеивания. С учетом большей формализации метод Квайна – Мак-Класки удобно использовать при машинной реализации алгоритмов минимизации ПФ. Рассмотрим основные этапы этого метода.

На первом этапе необходимо представить ПФ в виде кубического комплекса K^0, представляющего собой совокупность 0-кубов, на которых минимизируемая ПФ принимает значение 1. Для этого в СДНФ функции каждый дизъюнктивный член заменяется n-разрядной двоичной комбинацией (n – число аргументов функции), представляющей собой номер конституенты 1, соответствующей этому дизъюнктивному члену. Иными словами, записываются n-разрядные двоичные наборы, на которых значение функции равно 1.

Далее все 0-кубы полученного кубического комплекса разбиваются на группы по числу единиц, входящих в их запись. Таким образом, максимальное число групп не превышает $(n+1)$.

Производится склеивание 0-кубов, которое возможно только между соседними группами. Результаты склеивания составляют новый кубический комплекс K^1. Если часть 0-кубов не участвовала в склеивании, то такие 0-кубы являются простыми импликантами.

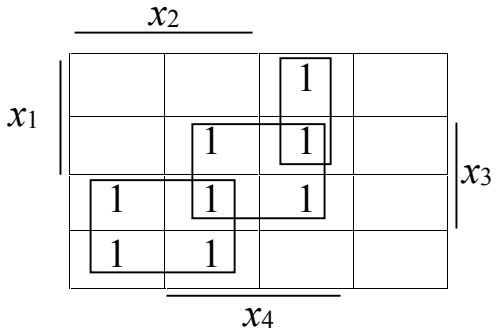
Формирование кубических комплексов продолжается до тех пор, пока не будет получен комплекс K^m, не содержащий m-кубов, отличающихся только по одной координате. При этом сохраняется

разбиение на группы по количеству единиц. Если на каком-либо этапе часть *i*-кубов не участвовала в склеивании, то такие *i*-кубы являются простыми импликантами и входят в минимальную ДНФ.

Рассмотрим пример.

Пусть подлежащая минимизации ПФ задана картой Карно (табл. 4.10).

Таблица 4.10



Заранее отметим на ней возможные простые импликанты для проверки правильности решения примера.

Для рассматриваемой функции СДНФ запишется в следующем виде:

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1 x_2 x_3 \bar{x}_4 \vee \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 \vee x_1 x_2 x_3 x_4 \vee \bar{x}_1 x_2 x_3 x_4 \vee \bar{x}_1 x_2 \bar{x}_3 x_4 \vee$$
$$\vee x_1 \bar{x}_2 \bar{x}_3 x_4 \vee x_1 \bar{x}_2 x_3 x_4 \vee \bar{x}_1 \bar{x}_2 x_3 x_4.$$

По СДНФ функции сформируем кубический комплекс K^0, каждый элемент которого представляет собой четырехразрядный двоичный набор. Если переменная x_i входит в запись конституенты 1 без инверсии, то *i*-й разряд двоичного набора, соответствующего этой конституенте 1, принимает значение 1. В противном случае *i*-й разряд двоичного набора, соответствующего этой конституенте 1, принимает значение 0.

$$K^0(f) = \begin{Bmatrix} 0110 \\ 0100 \\ 1111 \\ 0111 \\ 0101 \\ 1001 \\ 1011 \\ 0011 \end{Bmatrix}.$$

Сведем в таблицу (табл. 4.11) полученные 0-кубы, упорядоченные по числу единиц.

Таблица 4.11

Кол-во единиц	0-кубы
0	–
1	0100
2	0110, 0101, 1001, 0011
3	0111, 1011
4	1111

Произведем склеивание по одной переменной между элементами соседних групп. В результате будут получены элементы кубического комплекса K^1, которые также сведем в таблицу (табл. 4.12).

Таблица 4.12

Кол-во единиц	1-кубы
1	01X0, 010X,
2	011X, 01X1, 10X1, 0X11, X011
3	X111, 1X11

Необходимо включать во все последующие таблицы 0-кубы и импликанты, не участвовавшие в склеивании, а затем исключать лишние по правилу поглощения.

Будем продолжать процедуру склеивания до тех пор, пока элементы очередного кубического комплекса могут склеиваться по одной переменной. При этом склеивание возможно только между *i*-кубами, имеющими одинаковые несущественные переменные (имеющими символ «X» в одинаковых позициях).

Для рассматриваемого примера кубический комплекс K^2, сформированный путем склеивания элементов комплекса K^1, представлен в табл. 4.13. Повторяющиеся 2-кубы исключены.

Таблица 4.13

Кол-во единиц	2-кубы
1	01XX, ~~01XX~~
2	10X1, XX11, ~~XX11~~

Дальнейшее склеивание невозможно, поэтому кубический комплекс K^2 представляет собой совокупность простых импликант.

Составим импликантную матрицу Квайна (табл. 4.14) для нахождения минимальной совокупности простых импликант, представляющих в минимальной ДНФ все конституенты единицы исходной СДНФ. В рассматриваемом примере требуется найти минимальную совокупность 2-кубов, накрывающих все 0-кубы минимизируемой ПФ.

Таблица 4.14

№	0-кубы	2-кубы		
		01XX	10X1	XX11
1	0100	+		
2	0110	+		
3	0101	+		
4	1001		+	
5	0011			+
6	0111	+		+
7	1011		+	+
8	1111			+

Из полученной импликантной матрицы видно, что все найденные на предыдущем этапе минимизации 2-кубы входят в минимальную совокупность простых импликант, т.е. все элементы кубического комплекса K^2 образуют минимальную ДНФ заданной ПФ.

Окончательно,

$$K^2(f) = \begin{Bmatrix} 01XX \\ 10X1 \\ XX11 \end{Bmatrix},$$

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1 x_2 \vee x_1 \bar{x}_2 x_4 \vee x_3 x_4 .$$

Полученное решение нетрудно проверить с помощью карты Карно (см. табл. 4.10).

4.4. Минимизация ПФ методом Блейка – Порецкого

Применение метода Квайна и метода Квайна – Мак-Класки для минимизации булевых функций требует преобразования произвольной ДНФ минимизируемой функции к СДНФ. В общем случае для осуществления такого преобразования можно пользоваться следующим соотношением: $x_1(x_2 \vee \bar{x}_2) = x_1 x_2 \vee x_1 \bar{x}_2$. Таким образом, для преобразования дизъюнктивного члена некоторой произвольной ДНФ необходимо выполнить умножение этого дизъюнктивного члена на дизъюнкцию прямого и инверсного значений отсутствующей переменной. В качестве примера можно привести следующее преобразование:

$$f(x_1, x_2, x_3, x_4) = x_1 x_2 \vee \bar{x}_2 \bar{x}_3 x_4 = x_1 x_2 (x_3 \vee \bar{x}_3)(x_4 \vee \bar{x}_4) \vee (x_1 \vee \bar{x}_1)\bar{x}_2 \bar{x}_3 x_4 =$$
$$= x_1 x_2 x_3 x_4 \vee x_1 x_2 x_3 \bar{x}_4 \vee x_1 x_2 \bar{x}_3 x_4 \vee x_1 x_2 \bar{x}_3 \bar{x}_4 \vee x_1 \bar{x}_2 \bar{x}_3 x_4 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 x_4 .$$

Как видно из этого условного примера, преобразование произвольной ДНФ к СДНФ требует довольно громоздких вычислений даже для функции четырех аргументов. Для функций большего числа аргументов процесс преобразования еще больше усложняется. Поэтому для минимизации функций, заданных произвольными ДНФ, удобно применять метод Блейка – Порецкого, не требующий задания ПФ в виде СДНФ.

Докажем справедливость следующего равенства, которое называется *формулой обобщенного склеивания*:

$$x_1 x \vee x_2 \bar{x} = x_1 x \vee x_2 \bar{x} \vee x_1 x_2 .$$

Равенство легко доказывается путем выполнения преобразований над его правой или левой частью. Преобразуем правую часть равенства.

$$x_1x \vee x_2\bar{x} \vee x_1x_2 = x_1x \vee x_2\bar{x} \vee x_1x_2(x \vee \bar{x}) = x_1x \vee x_2\bar{x} \vee x_1x_2x \vee x_1x_2\bar{x} =$$
$$= (x_1x \vee x_1x_2x) \vee (x_2\bar{x} \vee x_1x_2\bar{x}) = x_1x \vee x_2\bar{x}.$$

Поскольку в результате преобразований из правой части формулы обобщенного склеивания получена левая часть, то равенство доказано.

Метод Блейка – Порецкого основывается на утверждении: если в произвольной ДНФ минимизируемой ПФ произвести все возможные обобщенные склеивания и все возможные поглощения, то в результате будет получена сокращенная ДНФ исходной функции.

Для получения минимальной ДНФ необходимо составить импликантную матрицу Квайна и определить по ней минимальное покрытие единиц минимизируемой функции простыми импликантами.

Рассмотрим пример. Пусть подлежащая минимизации функция задана в следующем виде:

$$f(x_1, x_2, x_3, x_4) = x_1x_2x_4 \vee x_2x_3x_4 \vee x_1\bar{x}_2x_4 \vee \bar{x}_2x_3x_4.$$

На первом этапе минимизации проведем всевозможные обобщенные склеивания.

$$x_1x_2x_4 \vee x_1\bar{x}_2x_4 = x_1x_2x_4 \vee x_1\bar{x}_2x_4 \vee x_1x_4 = x_1x_4;$$
$$x_1x_2x_4 \vee \bar{x}_2x_3x_4 = x_1x_2x_4 \vee \bar{x}_2x_3x_4 \vee x_1x_3x_4;$$
$$x_2x_3x_4 \vee x_1\bar{x}_2x_4 = x_2x_3x_4 \vee x_1\bar{x}_2x_4 \vee x_1x_3x_4;$$
$$x_2x_3x_4 \vee \bar{x}_2x_3x_4 = x_2x_3x_4 \vee \bar{x}_2x_3x_4 \vee x_3x_4 = x_3x_4.$$

Исключив повторения элементарных произведений, получим ДНФ:

$$f(x_1, x_2, x_3, x_4) = x_1x_4 \vee x_1x_2x_4 \vee \bar{x}_2x_3x_4 \vee x_1x_3x_4 \vee x_2x_3x_4 \vee x_1\bar{x}_2x_4 \vee x_3x_4.$$

Выполнив все возможные поглощения, получим сокращенную ДНФ:

$$f(x_1, x_2, x_3, x_4) = x_1x_4 \vee x_3x_4.$$

Построив импликантную матрицу (табл. 4.15), легко убедиться, что полученная сокращенная ДНФ является минимальной.

Таблица 4.15

№	Исходные слагаемые	Простые импликанты	
		x_1x_4	x_3x_4
1	$x_1x_2x_4$	+	

2	$x_2x_3x_4$		+
3	$x_1\bar{x}_2x_4$	+	
4	$\bar{x}_2x_3x_4$		+

4.5. Минимизация ПФ, заданных в конъюнктивной форме

Для минимизации ПФ, заданных в виде КНФ, можно использовать все вышеизложенные методы с незначительными отличиями. Отличия заключаются в записи соотношений склеивания и поглощения в конъюнктивной форме, а также в использовании некоторых понятий, характерных для КНФ.

Конституентой 0 называется функция, принимающая значение 0 на одном наборе значений аргументов. В конъюнктивной форме конституента 0 записывается как элементарная дизъюнкция всех переменных, причем переменная входит в дизъюнкцию с отрицанием, если ей соответствует цифра 1 в наборе. Например, конституента 0 $(\bar{x}_1 \vee x_2 \vee x_3 \vee x_4)$ принимает значение 0 на наборе (1000).

Понятие импликанты и простой импликанты заменяется для КНФ на понятие имплиценты и простой имплиценты.

Имплицентой функции f называется функция, принимающая значение 0 на тех же наборах (или на части тех же наборов), на которых функция *f* принимает нулевое значение.

Простой имплицентой функции f называется имплицента, никакая часть которой не является сама по себе имплицентой *f*.

Для минимизации ПФ, заданной КНФ, как и в методах минимизации ДНФ, необходимо выполнить два этапа. На первом этапе осуществляется нахождение сокращенной КНФ, т.е. конъюнкции всех простых имплицент. При этом можно пользоваться соотношениями склеивания и поглощения, адаптированными для КНФ. Формула склеивания записывается как

$$(x_1 \vee \bar{x}_2)(x_1 \vee x_2) = x_1(x_1 \vee \bar{x}_2)(x_1 \vee x_2),$$

а формула обобщенного склеивания как

$$(x_1 \vee x)(x_2 \vee \bar{x}) = (x_1 \vee x)(x_2 \vee \bar{x})(x_1 \vee x_2).$$

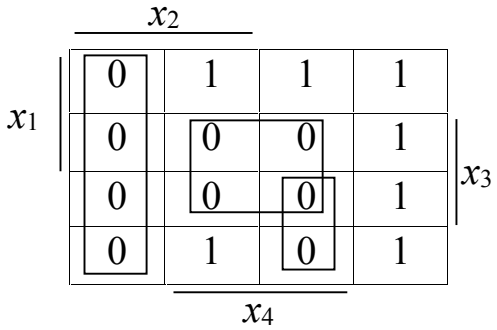
После нахождения всех возможных склеиваний следует выполнить все возможные поглощения по формуле

$$(x_1 \vee x_2)x_1 = x_1 \vee x_2 x_1 = x_1.$$

Для получения минимальной КНФ по сокращенной КНФ следует сформировать имплицентную матрицу Квайна, по которой определяется минимальное покрытие простыми имплицентами всех конституент 0 минимизируемой функции.

При минимизации ПФ с помощью карт Карно главное отличие заключается в поиске совокупностей, состоящих из 2^k нулей. Все остальное можно оставить без изменений, но при этом в записи минимальной КНФ значения переменных следует инвертировать. Например, пусть подлежащая минимизации ПФ задана в виде карты Карно (табл. 4.16).

Таблица 4.16



Тогда ее минимальная КНФ запишется в следующем виде:

$$f(x_1, x_2, x_3, x_4) = (\bar{x}_2 \vee x_4) \cdot (\bar{x}_3 \vee \bar{x}_4) \cdot (x_1 \vee x_2 \vee \bar{x}_4).$$

В общем случае до решения задачи минимизации ПФ нельзя сказать, какая из минимальных форм (ДНФ или КНФ) окажется проще. В некоторых случаях удобно использовать для получения минимальной ДНФ минимизацию по нулям. При этом будет получена минимальная ДНФ для функции $\bar{f}$.

4.6. Минимизация не полностью определенных ПФ

ПФ называется *не полностью определенной* или *частично определенной*, если число наборов *n* аргументов, на которых ее значение задано, меньше 2^n. С точки зрения практической реализации такой ПФ это может означать, что наборы аргументов, на которых значение функции не

определено, никогда не появляются на входах КС либо формируемые на таких наборах выходные сигналы являются несущественными.

Из этого следует, что на таких наборах аргументов функция может быть произвольно доопределена значениями 0 или 1. Обычно доопределение производят, исходя из соображений минимизации. Рассмотрим ПФ, заданную картой Карно (табл. 4.17).

Таблица 4.17

x_1 \ x_2				
	1	-	0	1
	1	-	0	0
		x_3		

Значения этой функции определены на шести наборах. Если функцию не доопределять, то минимальная ДНФ запишется в следующем виде:

$$f(x_1, x_2, x_3) = x_2\bar{x}_3 \vee x_1\bar{x}_3.$$

Доопределим заданную функцию единицами на оставшихся двух наборах (табл. 4.18).

Таблица 4.18

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

Тогда минимальную ДНФ можно записать в более простом виде:

$$f(x_1, x_2, x_3) = x_2 \vee x_1\bar{x}_3.$$

В некоторых случаях для получения минимальной ДНФ выгодно на некоторых наборах доопределить функцию единицами, а на остальных наборах – нулями или оставить неопределенной. Пусть функция задана картой Карно (табл. 4.19). Доопределим ее единицами сначала на всех

неопределенных наборах (табл. 4.20), а затем только на одном из них (табл. 4.21), и запишем для каждой карты Карно получающиеся минимальные ДНФ.

Таблица 4.19

x_1 \ x_2	1	1	0	-
	0	-	1	0

x_3

$$f(x_1, x_2, x_3) = x_1 x_2 \vee \bar{x}_1 \bar{x}_2 x_3.$$

Таблица 4.20

x_1 \ x_2	1	1	0	1
	0	1	1	0

x_3

$$f(x_1, x_2, x_3) = x_1 x_2 \vee x_1 \bar{x}_3 \vee \bar{x}_1 x_3.$$

Таблица 4.21

x_1 \ x_2	1	1	0	-
	0	1	1	0

x_3

$$f(x_1, x_2, x_3) = x_1 x_2 \vee \bar{x}_1 x_3.$$

Частичное доопределение не полностью определенной ПФ требует перебора большого числа возможных вариантов и может дать ощутимую выгоду при минимизации на картах Карно для функций небольшого числа аргументов.

Для применения других методов минимизации (например, метод Квайна – Мак-Класки) удобно доопределить функцию на всех неопределенных наборах (единицами или нулями) и выполнить следующие шаги:

1) любым способом найти сокращенную ДНФ (КНФ);

2) по импликантной матрице Квайна определить минимальную совокупность простых импликант, накрывающую только те конституенты 1 (конституенты 0), которые соответствуют наборам, на которых функция была изначально определена.

4.7. Минимизация систем ПФ

Комбинационные схемы многих цифровых устройств являются многовыходными, поэтому аналитическое описание таких схем представляет собой совокупность переключательных функций, каждая из которых описывает один из выходов КС. Таким образом, *m*-выходная КС может быть задана системой *m* ПФ, где $y_1, y_2, ..., y_m$ – выходные сигналы:

$$\begin{cases} y_1 = f_1(x_1, x_2, ..., x_n); \\ y_2 = f_2(x_1, x_2, ..., x_n); \\ ... \\ y_m = f_m(x_1, x_2, ..., x_n). \end{cases}$$

Систему ПФ можно минимизировать двумя способами. Если минимизировать каждую ПФ системы независимо от других ПФ, то в результате будут получены описания *m* изолированных подсхем, каждая из которых задает один из выходов КС. Если же при минимизации системы ПФ учитывать, что некоторые части подсхем, реализующие одинаковые логические произведения или суммы, могут быть использованы при синтезе нескольких различных функций системы, то можно получить такую минимальную систему ПФ, цена которой будет ниже, чем для системы, полученной первым способом.

Рассмотрим систему ПФ, каждая функция которой задана в СДНФ:

$$\begin{cases} f_1(x_1, x_2, x_3, x_4) = x_1 x_2 \bar{x}_3 x_4 \vee \bar{x}_1 x_2 x_3 x_4; \\ f_2(x_1, x_2, x_3, x_4) = x_1 x_2 \bar{x}_3 x_4 \vee \bar{x}_1 x_2 x_3 x_4 \vee x_1 x_2 \bar{x}_3 \bar{x}_4 \vee x_1 x_2 x_3 \bar{x}_4 \vee \bar{x}_1 x_2 x_3 \bar{x}_4 \vee \\ \qquad\qquad \vee \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4. \end{cases}$$

Минимизация каждой функции по отдельности дает следующую систему ПФ:

$$\begin{cases} f_1(x_1, x_2, x_3, x_4) = x_1 x_2 \bar{x}_3 x_4 \vee \bar{x}_1 x_2 x_3 x_4; \\ f_2(x_1, x_2, x_3, x_4) = x_2 \bar{x}_4 \vee x_1 x_2 \bar{x}_3 \vee \bar{x}_1 x_2 x_3. \end{cases}$$

Для технической реализации такой системы ПФ в базисе И-ИЛИ-НЕ потребуются (без учета инверторов) два четырехвходовых элемента «И», два трехвходовых элемента «И», один двухвходовый элемент «И» и два элемента «ИЛИ», один из которых двухвходовый, а второй – трехвходовый.

Для наглядности изобразим исходную систему ПФ в виде карт Карно (табл. 4.22, 4.23) и выполним склеивания в соответствии с выделенными в этих картах совокупностями единиц.

Тогда минимальная система ПФ запишется следующим образом:

$$\begin{cases} f_1(x_1, x_2, x_3, x_4) = x_1 x_2 \bar{x}_3 x_4 \vee \bar{x}_1 x_2 x_3 x_4; \\ f_2(x_1, x_2, x_3, x_4) = x_2 \bar{x}_4 \vee f_1. \end{cases}$$

Таблица 4.22

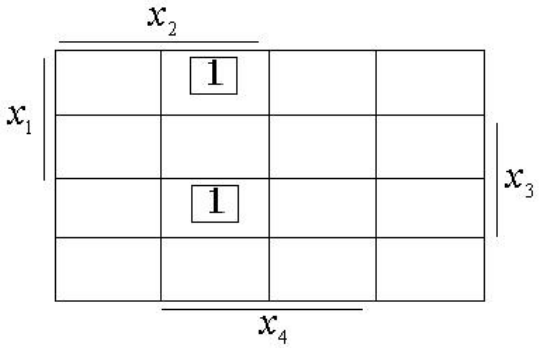
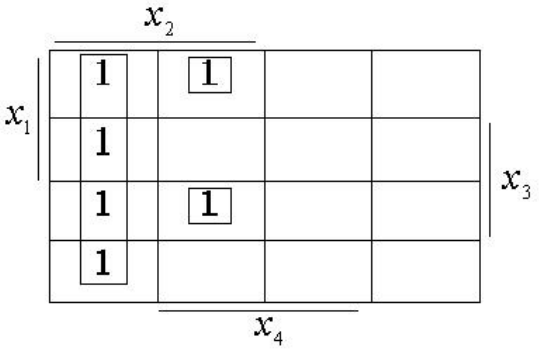


Таблица 4.23



Очевидно, что за счет использования одинаковых частей схемы для реализации обеих функций удалось снизить общую стоимость КС. Пример построения такой схемы изображен на рис. 4.1.

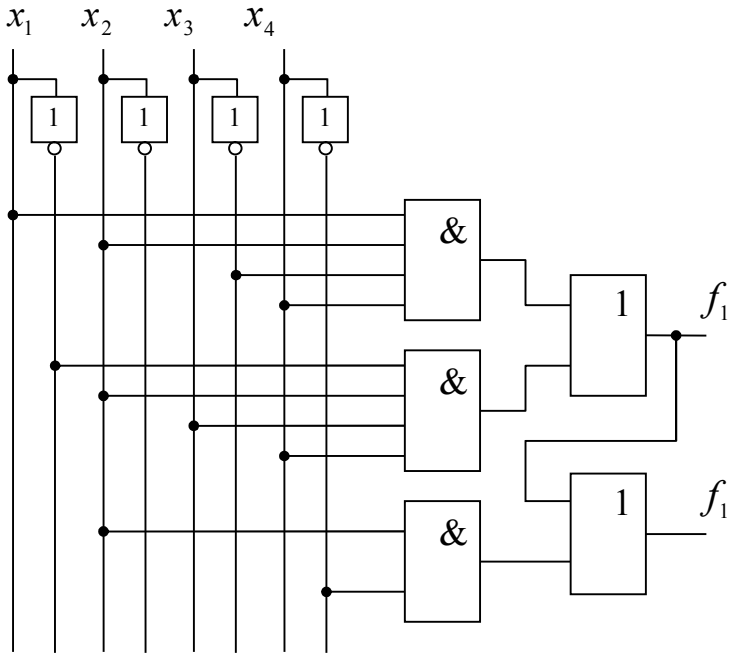


Рис. 4.1

Решение задачи минимизации систем ПФ рассмотрим на основе модифицированного метода Квайна [4]. Будем полагать, что функции системы заданы в виде ДНФ.

Введем множество A, элементами которого являются все различные элементарные произведения минимизируемой системы ПФ. Такое множество будем называть *полным множеством элементарных произведений*. Систему ДНФ переключательных функций будем называть *минимальной*, если полное множество элементарных произведений этой совокупности содержит минимальное количество букв и каждая ПФ включает минимальное число дизъюнктивных членов. Очевидно, что ДНФ функции в минимальной системе в общем случае может не совпадать с минимальной ДНФ этой функции.

На первом шаге минимизации необходимо сформировать полное множество элементарных произведений минимизируемой системы. При этом необходимо использовать представление всех ПФ в СДНФ. Каждому элементу множества A следует присвоить индекс, содержащий номера функций системы, в которые входит это элементарное произведение. Пусть элементы сформированного множества A образуют СДНФ булевой функции g.

Далее следует минимизировать полученную на предыдущем шаге функцию g с целью получения совокупности простых импликант системы ПФ. При выполнении этой процедуры каждому элементарному произведению, получающемуся в результате склеивания конституент 1, присваивается индекс, состоящий из номеров функций, общих для обеих склеиваемых конституент. Склеивание конституент 1 производится только в том случае, если их индексы содержат общие номера. Поглощение также может быть выполнено только для элементарных произведений, у которых одинаковые номера в индексах.

Затем строится импликантная матрица Квайна, столбцы которой содержат все простые импликанты, полученные на предыдущем шаге. Строки импликантной матрицы содержат конституенты 1, причем каждая из них может присутствовать в нескольких столбцах по числу функций системы ПФ, в которые она входит. Ячейки импликантной матрицы отмечаются так же, как в методе Квайна для одной ПФ, но только в том случае, когда индекс импликанты совпадает с индексом конституенты 1.

Рассмотрим пример.

Пусть подлежащая минимизации система ПФ задана имеет вид

$$\begin{cases} f_1(x_1, x_2, x_3) = x_1x_2x_3 \vee \bar{x}_1x_2\bar{x}_3 \vee \bar{x}_1x_2x_3 \vee \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3; \\ f_2(x_1, x_2, x_3) = x_1x_2x_3 \vee \bar{x}_1x_2\bar{x}_3 \vee \bar{x}_1x_2x_3 \vee x_1x_2\bar{x}_3 \vee x_1\bar{x}_2x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3; \\ f_3(x_1, x_2, x_3) = x_1x_2x_3 \vee \bar{x}_1x_2x_3 \vee x_1\bar{x}_2x_3 \vee x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3. \end{cases}$$

Сформируем полное множество элементарных конъюнкций, каждому элементу множества присвоим индекс, соответствующий номеру функции, содержащей это произведение.

$$A = \{x_1x_2x_3(1,2,3);\ \bar{x}_1x_2\bar{x}_3(1,2);\ \bar{x}_1x_2x_3(1,2,3);\ \bar{x}_1\bar{x}_2x_3(1);\ \bar{x}_1\bar{x}_2\bar{x}_3(1,2,3);$$
$$x_1x_2\bar{x}_3(2);\ x_1\bar{x}_2x_3(2,3);\ x_1\bar{x}_2\bar{x}_3(3);\}.$$

Из элементов множества A составим СДНФ функции g:

$$g = x_1x_2x_3(1,2,3) \vee \bar{x}_1x_2\bar{x}_3(1,2) \vee \bar{x}_1x_2x_3(1,2,3) \vee \bar{x}_1\bar{x}_2x_3(1) \vee \bar{x}_1\bar{x}_2\bar{x}_3(1,2,3) \vee$$
$$\vee x_1x_2\bar{x}_3(2) \vee x_1\bar{x}_2x_3(2,3) \vee x_1\bar{x}_2\bar{x}_3(3).$$

Выполним склеивания по формуле

$$x_1\bar{x}_2 \vee x_1x_2 = x_1 \vee x_1\bar{x}_2 \vee x_1x_2.$$

Результаты склеивания будем записывать в отдельных строках.

$x_2x_3(1,2,3) \vee x_1x_2x_3(1,2,3) \vee \bar{x}_1x_2x_3(1,2,3);$

$x_1x_2(2) \vee x_1x_2x_3(1,2,3) \vee x_1x_2\bar{x}_3(2);$

$x_1x_3(2,3) \vee x_1x_2x_3(1,2,3) \vee x_1\bar{x}_2x_3(2,3);$

$\bar{x}_1x_2(1,2) \vee \bar{x}_1x_2\bar{x}_3(1,2) \vee \bar{x}_1x_2x_3(1,2,3);$

$\bar{x}_1\bar{x}_3(1,2) \vee \bar{x}_1x_2\bar{x}_3(1,2) \vee \bar{x}_1\bar{x}_2\bar{x}_3(1,2,3);$

$x_2\bar{x}_3(2) \vee \bar{x}_1x_2\bar{x}_3(1,2) \vee x_1x_2\bar{x}_3(2);$

$\bar{x}_1x_3(1) \vee \bar{x}_1x_2x_3(1,2,3) \vee \bar{x}_1\bar{x}_2x_3(1);$

$\bar{x}_1\bar{x}_2(1) \vee \bar{x}_1\bar{x}_2x_3(1) \vee \bar{x}_1\bar{x}_2\bar{x}_3(1,2,3);$

$\bar{x}_2\bar{x}_3(3) \vee \bar{x}_1\bar{x}_2\bar{x}_3(1,2,3) \vee x_1\bar{x}_2\bar{x}_3(3);$

$x_1\bar{x}_2(3) \vee x_1\bar{x}_2x_3(2,3) \vee x_1\bar{x}_2\bar{x}_3(3).$

Осуществив поглощения, получим

$$g(x_1,x_2,x_3) = x_2x_3(1,2,3) \vee x_1x_2(2) \vee x_1x_3(2,3) \vee \bar{x}_1x_2(1,2) \vee \bar{x}_1\bar{x}_3(1,2) \vee$$
$$\vee x_2\bar{x}_3(2) \vee \bar{x}_1x_3(1) \vee \bar{x}_1\bar{x}_2(1) \vee \bar{x}_2\bar{x}_3(3) \vee x_1\bar{x}_2(3).$$

Два последних шага будем повторять до тех пор, пока возможны склеивания и поглощения в получающихся ДНФ. Окончательно получим

$$g(x_1,x_2,x_3) = x_2(2) \vee \bar{x}_1(1) \vee x_2x_3(1,3) \vee \bar{x}_1\bar{x}_3(2) \vee x_1x_3(2,3) \vee \bar{x}_2\bar{x}_3(3) \vee x_1\bar{x}_2(3)$$

Импликантная матрица Квайна представлена в табл. 4.24.

Таблица 4.24

Конституенты 1 функции g	Простые импликанты						
	$x_2(2)$	$\bar{x}_1(1)$	$x_2x_3(1,3)$	$\bar{x}_1\bar{x}_3(2)$	$x_1x_3(2,3)$	$\bar{x}_2\bar{x}_3(3)$	$x_1\bar{x}_2(3)$
$x_1x_2x_3(1)$			+				
$x_1x_2x_3(2)$	+				+		
$x_1x_2x_3(3)$			+		+		
$\bar{x}_1x_2\bar{x}_3(1)$		+					
$\bar{x}_1x_2\bar{x}_3(2)$	+			+			
$\bar{x}_1x_2x_3(1)$		+	+				
$\bar{x}_1x_2x_3(2)$	+						
$\bar{x}_1x_2x_3(3)$			+				
$\bar{x}_1\bar{x}_2x_3(1)$		+					
$\bar{x}_1\bar{x}_2\bar{x}_3(1)$		+					
$\bar{x}_1\bar{x}_2\bar{x}_3(2)$				+			
$\bar{x}_1\bar{x}_2\bar{x}_3(3)$						+	
$x_1x_2\bar{x}_3(2)$	+						
$x_1\bar{x}_2x_3(2)$					+		
$x_1\bar{x}_2x_3(3)$					+		+
$x_1\bar{x}_2\bar{x}_3(3)$						+	+

После обработки импликантной матрицы можно записать:

$$g = x_2(2) \vee \bar{x}_1(1) \vee x_2x_3(1,3) \vee \bar{x}_1\bar{x}_3(2) \vee x_1x_3(2,3) \vee \bar{x}_2\bar{x}_3(3).$$

Запишем минимальную систему ПФ, включая в ДНФ *i*-й функции простые импликанты, в индексе которых содержится *i*:

$$\begin{cases} f_1(x_1, x_2, x_3) = x_2x_3 \vee \bar{x}_1; \\ f_2(x_1, x_2, x_3) = x_2 \vee x_1x_3 \vee \bar{x}_1\bar{x}_3; \\ f_3(x_1, x_2, x_3) = x_2x_3 \vee x_1x_3 \vee \bar{x}_2\bar{x}_3. \end{cases}$$

На рис. 4.2 изображен пример практической реализации КС, описываемой полученной минимальной системой ПФ.

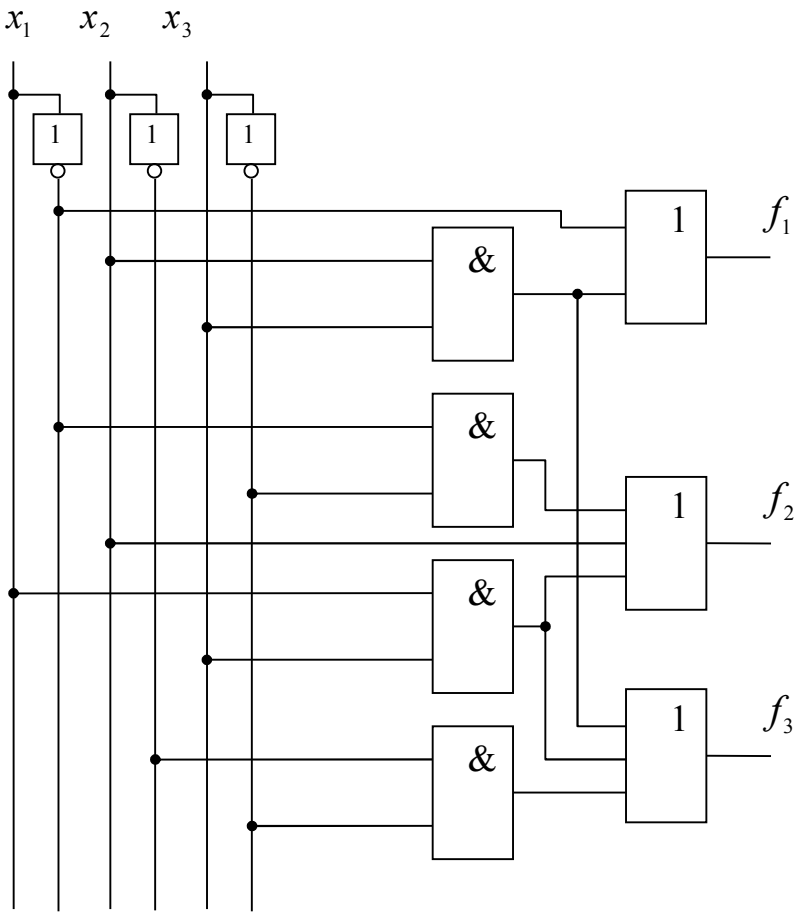


Рис. 4.2

4.8. Минимизация ПФ в универсальных базисах И-НЕ, ИЛИ-НЕ

Одной из причин широкого распространения интегральных логических схем, включающих логические элементы, реализующие функции
«И-НЕ» и «ИЛИ-НЕ» (штрих Шеффера и стрелка Пирса), является функциональная полнота базисов Шеффера и Пирса. При синтезе КС в каноническом базисе И-ИЛИ-НЕ для реализации всех возможных конъюнкций, дизъюнкций и инверсий в общем случае требуются три типа логических элементов. Поскольку базисы Пирса и Шеффера обладают свойством функциональной полноты, любая ПФ или система ПФ может быть представлена в этих базисах. Следовательно, любая КС может быть синтезирована в базисе Пирса или Шеффера с использованием однотипных логических элементов.

Базисы Пирса и Шеффера связаны с каноническим базисом формулами де Моргана, которые в общем виде записываются следующим образом:

$$x_1 \downarrow x_2 \downarrow \ldots \downarrow x_n = \overline{x_1 \vee x_2 \vee \ldots \vee x_n} = \bar{x}_1 \cdot \bar{x}_2 \cdot \ldots \cdot \bar{x}_n;$$
$$x_1 \mid x_2 \mid \ldots \mid x_n = \overline{x_1 \cdot x_2 \cdot \ldots \cdot x_n} = \bar{x}_1 \vee \bar{x}_2 \vee \ldots \vee \bar{x}_n.$$

Минимизация логических функций в базисах И-НЕ, ИЛИ-НЕ требует разработки специальных правил преобразования для выполнения склеиваний и поглощений. Гораздо более простым подходом является применение хорошо разработанных методов минимизации ПФ для канонического базиса И-ИЛИ-НЕ (например, метод Квайна, метод Блейка – Порецкого) с последующим переходом к базису Пирса или Шеффера. Строго говоря, при таком способе минимизации в общем случае получаются не минимальные, а близкие к минимальным формы представления ПФ.

Пусть подлежащая минимизации ПФ задана в СДНФ или СКНФ. Рассмотрим универсальный метод перехода к базисам И-НЕ, ИЛИ-НЕ.

Введем следующие обозначения. Каждую конституенту 1, входящую в СДНФ логической функции, представим в виде элементарной конъюнкции K_i, а каждую конституенту 0, входящую в СКНФ, – в виде элементарной дизъюнкции D_j:

$$K_i = \tilde{x}_1 \cdot \tilde{x}_2 \cdot \ldots \cdot \tilde{x}_n;$$
$$D_j = \tilde{x}_1 \vee \tilde{x}_2 \vee \ldots \vee \tilde{x}_n;$$
$$\tilde{x}_k \in \{x, \bar{x}\}, \quad k = 1,2,\ldots,n.$$

Тогда СДНФ и СКНФ в общем виде можно записать как

$$\text{СДНФ}: \quad f(x_1, x_2, \ldots, x_n) = \bigvee_{i=1}^{r} K_i;$$
$$\text{СКНФ}: \quad f(x_1, x_2, \ldots, x_n) = \bigwedge_{j=1}^{s} D_j.$$

Воспользовавшись законом де Моргана, получим

$$\text{СДНФ} \;:\; f(x_1, x_2, \ldots, x_n) = \overline{\overline{\bigvee_{i=1}^{r} K_i}} = \overline{\bigwedge_{i=1}^{r} \overline{K_i}};$$
$$\text{СКНФ} \;:\; f(x_1, x_2, \ldots, x_n) = \overline{\overline{\bigwedge_{j=1}^{s} D_j}} = \overline{\bigvee_{j=1}^{s} \overline{D_j}}.$$

В приведенных выражениях инверсии K_i и D_j могут быть выражены через операции «стрелка Пирса» и «штрих Шеффера»:

$$\overline{K_i} = \overline{\tilde{x}_1 \cdot \tilde{x}_2 \cdot \ldots \cdot \tilde{x}_n} = \tilde{x}_1 \mid \tilde{x}_2 \mid \ldots \mid \tilde{x}_n;$$
$$\overline{D_i} = \overline{\tilde{x}_1 \vee \tilde{x}_2 \vee \ldots \vee \tilde{x}_n} = \tilde{x}_1 \downarrow \tilde{x}_2 \downarrow \ldots \downarrow \tilde{x}_n,$$

откуда

$$f(x_1, x_2, \ldots, x_n) = \overline{\overline{\bigvee_{i=1}^{r} K_i}} = (\tilde{x}_1 \mid \tilde{x}_2 \mid \ldots \mid \tilde{x}_n)_1 \mid (\tilde{x}_1 \mid \tilde{x}_2 \mid \ldots \mid \tilde{x}_n)_2 \mid \ldots \mid (\tilde{x}_1 \mid \tilde{x}_2 \mid \ldots \mid \tilde{x}_n)_r;$$
$$f(x_1, x_2, \ldots, x_n) = \overline{\overline{\bigwedge_{j=1}^{s} D_j}} = (\tilde{x}_1 \downarrow \tilde{x}_2 \downarrow \ldots \downarrow \tilde{x}_n)_1 \downarrow (\tilde{x}_1 \downarrow \tilde{x}_2 \downarrow \ldots \downarrow \tilde{x}_n)_2 \downarrow \ldots \downarrow$$
$$\downarrow (\tilde{x}_1 \downarrow \tilde{x}_2 \downarrow \ldots \downarrow \tilde{x}_n)_s.$$

Таким образом, для перехода от произвольной ДНФ к базису И-НЕ (штрих Шеффера) необходимо в исходном выражении: поставить знак инверсии над всем выражением, заменить все знаки дизъюнкции на знаки конъюнкции, над каждым дизъюнктивным членом, входившим в ДНФ, поставить знак инверсии (заменить все знаки дизъюнкции и конъюнкции на знак операции «штрих Шеффера»).

Для перехода от произвольной КНФ к базису ИЛИ-НЕ (стрелка Пирса) необходимо в исходном выражении: поставить знак инверсии над всем выражением, заменить все знаки конъюнкции на знаки дизъюнкции, над каждым конъюнктивным членом, входившим в КНФ, поставить знак инверсии (заменить все знаки дизъюнкции и конъюнкции на знак операции «стрелка Пирса»).

Рассмотрим пример.

Осуществим переход от КНФ к базису ИЛИ-НЕ (стрелка Пирса).

$$f(x_1,x_2,x_3,x_4) = (\bar{x}_2 \vee x_4)\cdot(\bar{x}_3 \vee \bar{x}_4)\cdot(x_1 \vee x_2 \vee \bar{x}_4) =$$
$$= \overline{\overline{(\bar{x}_2 \vee x_4)} \vee \overline{(\bar{x}_3 \vee \bar{x}_4)} \vee \overline{(x_1 \vee x_2 \vee \bar{x}_4)}} = (\bar{x}_2 \downarrow x_4) \downarrow (\bar{x}_3 \downarrow \bar{x}_4) \downarrow (x_1 \downarrow x_2 \downarrow \bar{x}_4).$$

Осуществим переход от ДНФ к базису И-НЕ (штрих Шеффера).

$$f(x_1,x_2,x_3,x_4) = \bar{x}_1x_2 \vee x_1\bar{x}_2x_4 \vee x_3x_4 = \overline{\overline{(\bar{x}_1x_2)}\cdot\overline{(x_1\bar{x}_2x_4)}\cdot\overline{(x_3x_4)}} =$$
$$= (\bar{x}_1 \mid x_2) \mid (x_1 \mid \bar{x}_2 \mid x_4) \mid (x_3 \mid x_4).$$

При таком переходе могут возникать ситуации, когда в записи ПФ, кроме операций И-НЕ (ИЛИ-НЕ), присутствуют операции инверсии НЕ. Инверторы могут быть представлены в универсальных базисах в соответствии с формулами

$$\bar{x} = \overline{x \cdot x} = x \mid x,$$
$$\bar{x} = \overline{x \vee x} = x \downarrow x,$$

т.е. в качестве инвертора выступает элемент И-НЕ (ИЛИ-НЕ), все входы которого объединены в один.

При синтезе КС в универсальных базисах может возникнуть необходимость в переходе от представления функции в ДНФ к представлению в базисе ИЛИ-НЕ или в переходе от представления функции в КНФ к представлению в базисе И-НЕ. В такой ситуации можно пользоваться следующими формулами:

$$\bigvee_{i=1}^{r} K_i = \overline{(\bar{\bar{x}}_1 \downarrow \bar{\bar{x}}_2 \downarrow \ldots \downarrow \bar{\bar{x}}_n)_1 \downarrow (\bar{\bar{x}}_1 \downarrow \bar{\bar{x}}_2 \downarrow \ldots \downarrow \bar{\bar{x}}_n)_2 \downarrow \ldots \downarrow (\bar{\bar{x}}_1 \downarrow \bar{\bar{x}}_2 \downarrow \ldots \downarrow \bar{\bar{x}}_n)_r};$$

$$\bigwedge_{j=1}^{s} D_j = \overline{(\bar{\bar{x}}_1 \mid \bar{\bar{x}}_2 \mid \ldots \mid \bar{\bar{x}}_n)_1 \mid (\bar{\bar{x}}_1 \mid \bar{\bar{x}}_2 \mid \ldots \mid \bar{\bar{x}}_n)_2 \mid \ldots \mid (\bar{\bar{x}}_1 \mid \bar{\bar{x}}_2 \mid \ldots \mid \bar{\bar{x}}_n)_s}.$$

На практике в некоторых случаях применяются логические элементы с ограниченным числом входов, например, элементы 2И-НЕ, т.е. в КС могут входить только двухвходовые элементы. Выражения для функций, описывающих такую КС, могут быть получены путем выполнения преобразований.

Для перехода к базису 2И-НЕ:

$$x_1 \mid x_2 \mid x_3 = \overline{x_1 \cdot x_2 \cdot x_3} = \overline{\overline{(x_1 x_2)} \cdot x_3} = \overline{(x_1 \mid x_2)} \mid x_3;$$

$$x_1 \mid x_2 \mid x_3 \mid x_4 = \overline{x_1 \cdot x_2 \cdot x_3 \cdot x_4} = \overline{\overline{(x_1 x_2)} \cdot \overline{(x_3 x_4)}} = \overline{(x_1 \mid x_2)} \mid \overline{(x_3 \mid x_4)}.$$

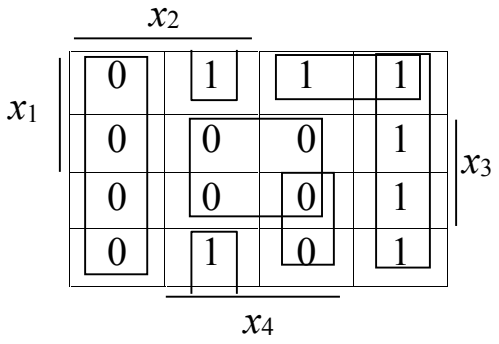
Для перехода к базису 2ИЛИ-НЕ:

$$x_1 \downarrow x_2 \downarrow x_3 = \overline{x_1 \vee x_2 \vee x_3} = \overline{\overline{(x_1 \vee x_2)} \vee x_3} = \overline{(x_1 \downarrow x_2)} \downarrow x_3;$$

$$x_1 \downarrow x_2 \downarrow x_3 \downarrow x_4 = \overline{x_1 \vee x_2 \vee x_3 \vee x_4} = \overline{\overline{(x_1 \vee x_2)} \vee \overline{(x_3 \vee x_4)}} = \overline{(x_1 \downarrow x_2)} \downarrow \overline{(x_3 \downarrow x_4)}.$$

Рассмотрим пример минимизации ПФ, заданной картой Карно (табл. 4.25) с последующим переходом к универсальному базису с произвольным числом входов логических элементов, а затем к универсальному базису двухвходовых логических элементов.

Таблица 4.25



Минимальные ДНФ и КНФ для этой функции можно записать в виде

$$f(x_1, x_2, x_3, x_4) = \bar{x}_2 \bar{x}_4 \vee x_1 \bar{x}_2 \bar{x}_3 \vee x_2 \bar{x}_3 x_4;$$

$$f(x_1, x_2, x_3, x_4) = (\bar{x}_2 \vee x_4) \cdot (\bar{x}_3 \vee \bar{x}_4) \cdot (x_1 \vee x_2 \vee \bar{x}_4).$$

Поскольку минимальная КНФ содержит меньшее число букв, используем ее для получения формы представления, близкой к минимальной, в базисе ИЛИ-НЕ.

$$f(x_1, x_2, x_3, x_4) = \overline{\overline{(\bar{x}_2 \vee x_4)} \vee \overline{(\bar{x}_3 \vee \bar{x}_4)} \vee \overline{(x_1 \vee x_2 \vee \bar{x}_4)}}.$$

Преобразуем полученную форму переключательной функции к представлению в базисе 2ИЛИ-НЕ.

$$f(x_1, x_2, x_3, x_4) = \overline{\overline{(\overline{x}_2 \vee x_4)} \vee \overline{(\overline{x}_3 \vee \overline{x}_4)} \vee \overline{(x_1 \vee x_2 \vee \overline{x}_4)}}.$$

На рис. 4.3 изображена КС, соответствующая полученной в результате минимизации ПФ, представленной в базисе 2ИЛИ-НЕ. В качестве инверторов также используются логические элементы 2ИЛИ-НЕ.

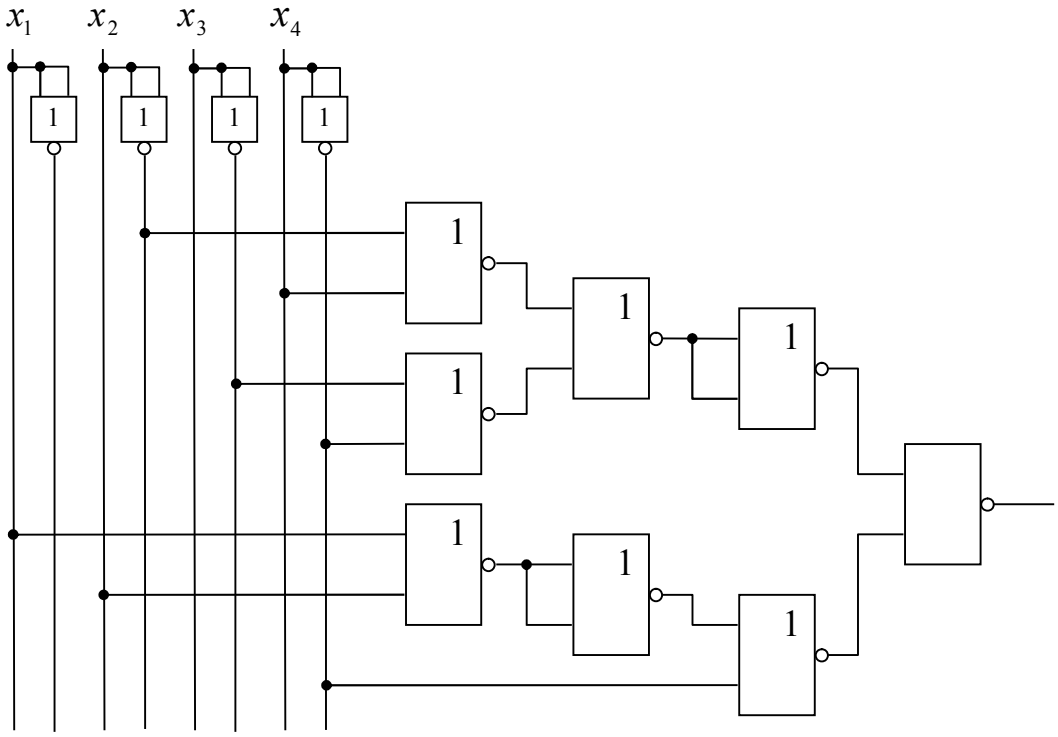


Рис. 4.3

Вопросы для самоконтроля

1. Для чего необходимо выполнять процедуру минимизации ПФ и систем ПФ?
2. Дайте определение импликанты ПФ и простой импликанты ПФ.
3. Сформулируйте основную идею метода минимизации ПФ с помощью карт Карно. В чем заключаются достоинства и недостатки такого метода?
4. Сформулируйте основные этапы минимизации ПФ по методу Квайна. Для чего предназначена импликантная матрица Квайна и как она используется в процессе минимизации ПФ?
5. В чем заключаются отличия метода Квайна – Мак-Класки от базового метода Квайна?

6. Сформулируйте основные этапы минимизации ПФ по методу Блейка – Порецкого.

7. В чем заключаются особенности минимизации ПФ, заданных в конъюнктивной форме, не полностью определенных ПФ, систем ПФ и ПФ, представляемых в универсальных базисах Пирса и Шеффера?

5. МОДЕЛИРОВАНИЕ РАБОТЫ И СИНТЕЗ АВТОМАТОВ С ПАМЯТЬЮ

5.1. Основные модели, понятия и определения

5.1.1. Общее понятие цифрового автомата с памятью

Рассмотрим устройство, имеющее два двоичных входа x_1, x_2 и один двоичный выход y. Пусть зависимость сигналов на выходе от наборов входных сигналов представлена временной диаграммой, изображенной на рис. 5.1, и может быть описана логическим выражением $y = \bar{x}_1 x_2 \vee x_1 \bar{x}_2$. Таким образом, рассматриваемое устройство является комбинационной схемой.

Далее рассмотрим устройство с одним входом x и одним выходом y, функционирующее в соответствии с временной диаграммой, изображенной на рис. 5.2.

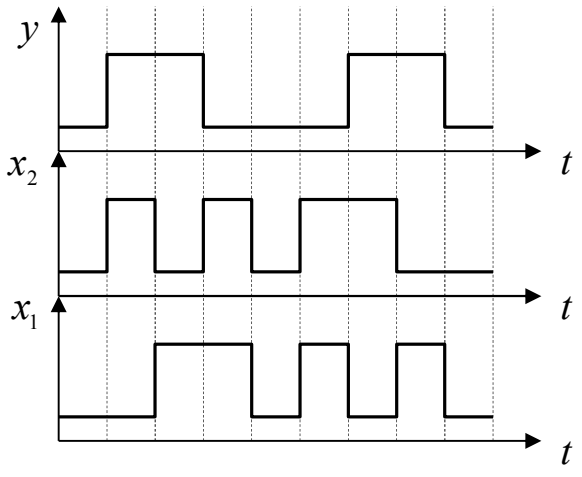


Рис. 5.1

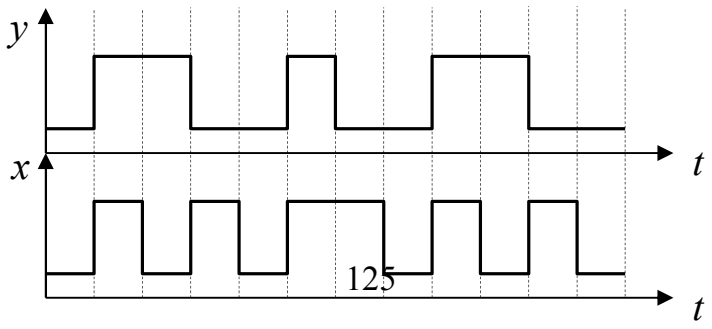


Рис. 5.2

Легко видеть, что выходной сигнал рассматриваемого устройства зависит не только от входного воздействия, но и от состояния в предыдущий момент времени. Следовательно, такое устройство должно обладать свойством запоминания, и в его структуре должны быть обратные связи, т.е. рассматриваемое устройство не является комбинационной схемой.

Работа данного устройства может быть описана графом, изображенным на рис. 5.3.

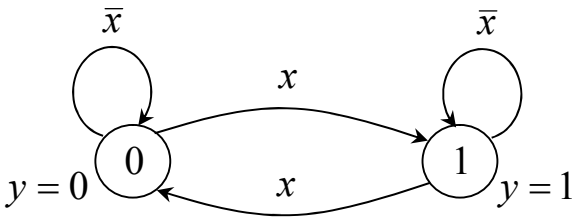


Рис. 5.3

В рамках прикладной теории цифровых автоматов принято выделять автоматы без памяти, называемые комбинационными схемами, рассмотренные выше в главах 3 и 4, и автоматы с памятью, которым посвящена данная глава учебного пособия. В дальнейшем терминами «автомат» или «цифровой автомат» будем обозначать автомат с памятью.

Введем ряд понятий и определений, которые понадобятся для дальнейшего изложения материала.

Цифровым автоматом (ЦА) будем называть дискретный преобразователь информации, способный под воздействием входных сигналов переходить из состояния в состояние и формировать выходные сигналы. Предполагается, что моменты времени, в которые автомат может переходить из состояния в состояние, дискретны.

Если множество состояний автомата, а также множества входных и выходных сигналов конечны, то автомат называется *конечным автоматом*. В дальнейшем будем рассматривать только конечные автоматы.

Совокупности входных символов ЦА образуют *входной алфавит* автомата. Отдельные символы алфавита называются *буквами*. Законченные последовательности букв называются *словами*.

Цифровой автомат называется *правильным*, если его выходные сигналы однозначно определяют состояние, в котором он находится.

Два автомата, имеющие одинаковые входные и выходные алфавиты, называются *эквивалентными*, если на любые одинаковые входные последовательности они отвечают одинаковыми выходными последовательностями. При этом число их внутренних состояний может быть различным.

Автомат называется *синхронным*, если моменты фиксации состояний задаются тактовым генератором. В *асинхронных* автоматах моменты переходов из состояния в состояние заранее не определены.

В данном учебном пособии наибольшее внимание уделено синхронным автоматам.

В теории цифровых автоматов условно можно выделить два основных направления исследований: абстрактное и структурное. При абстрактном анализе и синтезе автомата главную роль играет не способ построения автомата, а те переходы из состояния в состояние, которые совершает автомат под воздействием входных сигналов, и те выходные сигналы, которые автомат при этом формирует. Структурный анализ и синтез подразумевают рассмотрение структуры самого автомата и его входных и выходных сигналов, а также исследование способов построения автоматов из набора логических элементов и элементарных автоматов, способов кодирования внутренних состояний и т.д.

5.1.2. Основные модели цифровых автоматов

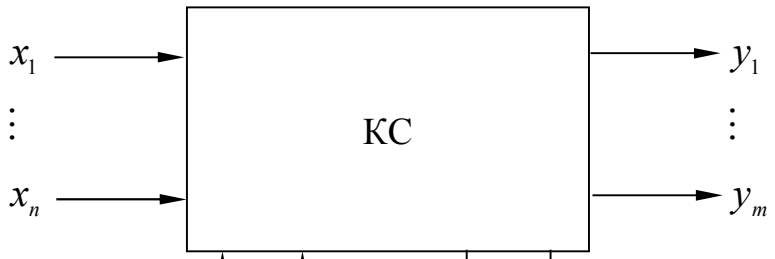
Структура ЦА в общем виде представлена на рис. 5.4, где введены следующие обозначения:

$x_1, x_2, ..., x_n$ – входные сигналы автомата;

$y_1, y_2, ..., y_m$ – выходные сигналы автомата;

$q_1, q_2, ..., q_k$ – выходы КС, управляющие состояниями элементов памяти;

$Q_1, Q_2, ..., Q_k$ – выходные сигналы элементов памяти, поступающие на входы КС.



Каждому ЦА можно поставить в соответствие следующую совокупность объектов.

1. Входной алфавит автомата $X = \{X_1, X_2, ..., X_r\}$.

2. Выходной алфавит автомата $Y = \{Y_1, Y_2, ..., Y_s\}$.

3. Множество внутренних состояний $A = \{a_1, a_2, ..., a_z\}$.

4. Функцию переходов, определяющую состояние автомата в момент времени $(t+1)$ в зависимости от его состояния в момент времени *t* и от входного воздействия в момент времени *t*.

5. Функцию выходов, определяющую зависимость выходных реакций автомата от его состояния и входного воздействия.

Следует отличать входной и выходной алфавиты абстрактного автомата от алфавитов структурной схемы.

Предположим, что задержка в КС равна нулю, а задержка в блоке памяти равна τ (обычно под τ понимают единицу автоматного времени). Тогда система уравнений, описывающих работу автомата, может быть записана в общем виде следующим образом:

$$y_1(t) = f_1\big(x(t), Q(t)\big);$$
$$\dots$$
$$y_m(t) = f_m\big(x(t), Q(t)\big);$$

$$q_1(t) = \varphi_1\big(x(t), Q(t)\big);$$
$$\dots$$
$$q_k(t) = \varphi_k\big(x(t), Q(t)\big);$$

$$Q_1(t+\tau) = \psi_1\big(q_1(t), Q_1(t)\big);$$
$$\dots$$
$$Q_k(t+\tau) = \psi_k\big(q_k(t), Q_k(t)\big).$$

Поскольку функции $q(t)$ зависят от совокупностей внешних входных сигналов $x(t)$ и состояний автомата $Q(t)$, то можно переписать последние k уравнений системы в виде

$$Q_1(t+\tau) = F_1\big(x(t), Q(t)\big);$$
$$\dots$$
$$Q_k(t+\tau) = F_k\big(x(t), Q(t)\big).$$

Приведенные уравнения описывают функционирование автомата I рода (автомат Мили), выходные сигналы которого формируются в момент перехода автомата из одного состояния в другое. Если выходные сигналы автомата $y(t)$ зависят только от состояний элементов памяти $Q(t)$, то такой автомат называется автоматом II рода (автомат Мура). Выходные сигналы последнего формируются в момент фиксации очередного состояния.

5.1.3. Описание функционирования цифровых автоматов

Процесс функционирования ЦА может иметь подробное словесное описание, характеризующее взаимосвязь выходных реакций автомата, входных воздействий и его внутренних состояний. Однако такой способ представления функционирования обладает рядом недостатков, среди которых можно выделить громоздкость и отсутствие наглядности.

Хорошим формализованным способом представления работы автомата является так называемая совмещенная таблица переходов-

выходов ЦА (*таблица переходов*). Одновременно с таблицами переходов будем рассматривать способ представления, называемый *графом переходов* автомата, поскольку обе эти формы представления однозначно соответствуют друг другу.

Рассмотрим табл. 5.1. В верхней строчке таблицы записаны номера внутренних состояний автомата a_j. В клетке таблицы на пересечении i-й строки и j-го столбца записывается номер состояния, в которое переходит автомат из j-го состояния под воздействием i-го входного сигнала. Для автомата Мили в этой же клетке указывается выходной сигнал, формируемый автоматом при таком переходе. Выходные сигналы автомата Мура, формирующиеся в моменты фиксации состояний, записываются в строке таблицы, расположенной над строкой внутренних состояний (табл. 5.2).

Альтернативной формой представления данных о переходах автомата является граф переходов. Состояния автомата представляются вершинами графа, переходы между ними – ориентированными ребрами, которые указывают начальное и последующее состояния перехода автомата. Ребро обозначается входным сигналом, под воздействием которого осуществляется соответствующий ему переход. Если строится граф переходов для автомата Мили, то переход также отмечается выходным сигналом. Выходные сигналы автомата Мура указываются рядом с теми состояниями, которым они соответствуют.

Справа от табл. 5.1 и 5.2 изображены соответствующие им графы переходов (рис. 5.5, 5.6).

Таблица 5.1

$x \setminus a$	1	2	3	4
x_1	1 / y_1	3 / y_3	3 / y_1	1 / y_2
x_2	2 / y_2	4 / y_2	4 / y_4	2 / y_3

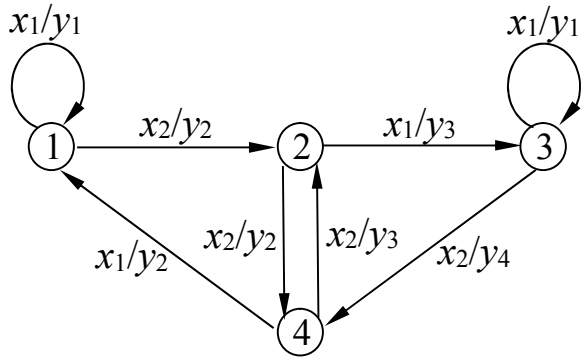
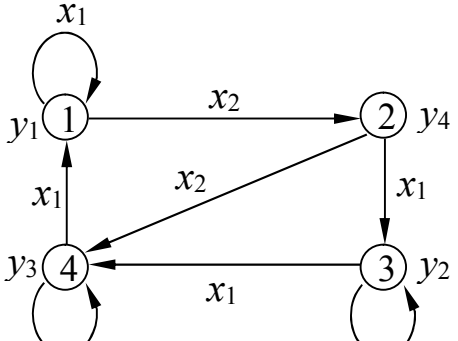


Рис. 5.5

Таблица 5.2



y	y_1	y_4	y_2	y_3
$x \setminus a$	1	2	3	4
x_1	1	3	4	1
x_2	2	4	3	4

Рис. 5.6

5.1.4. Задание цифровых автоматов

Цифровой автомат считается заданным, если известны следующие параметры:

1) Множество состояний автомата $A = \{a_1, a_2, ..., a_z\}$.

2) Входной алфавит $X = \{X_1, X_2, ..., X_r\}$.

3) Выходной алфавит $Y = \{Y_1, Y_2, ..., Y_s\}$.

4) Функции переходов между состояниями $a(t+1) = f\big(x(t), a(t)\big)$ и функции выходов.

5) Начальное состояние.

Задание ЦА возможно путем словесного описания его работы с последующим применением специальных формализованных способов описания, например, регулярных выражений. Далее выполняется переход к табличному или графовому представлению автомата.

Функционирование управляющего микропрограммного автомата может быть определено таблицей или графом переходов, полученным по размеченной схеме алгоритма его работы. Таким образом, для задания автомата по схеме алгоритма необходимо выполнить ее разметку в соответствии с типом задаваемого автомата.

Для автомата Мили разметка должна осуществляться следующим образом.

Все операторные вершины должны быть отмечены символами y_1, y_2, …, y_m. Одинаковые операторные вершины отмечаются одинаковыми символами y_i.

Все условные вершины отмечаются символами x_1, x_2, …, x_n. Одинаковые операторные вершины отмечаются одинаковыми символами X_i.

Выходы операторных вершин отмечаются символами внутренних состояний автомата a_1, a_2, …, a_z.

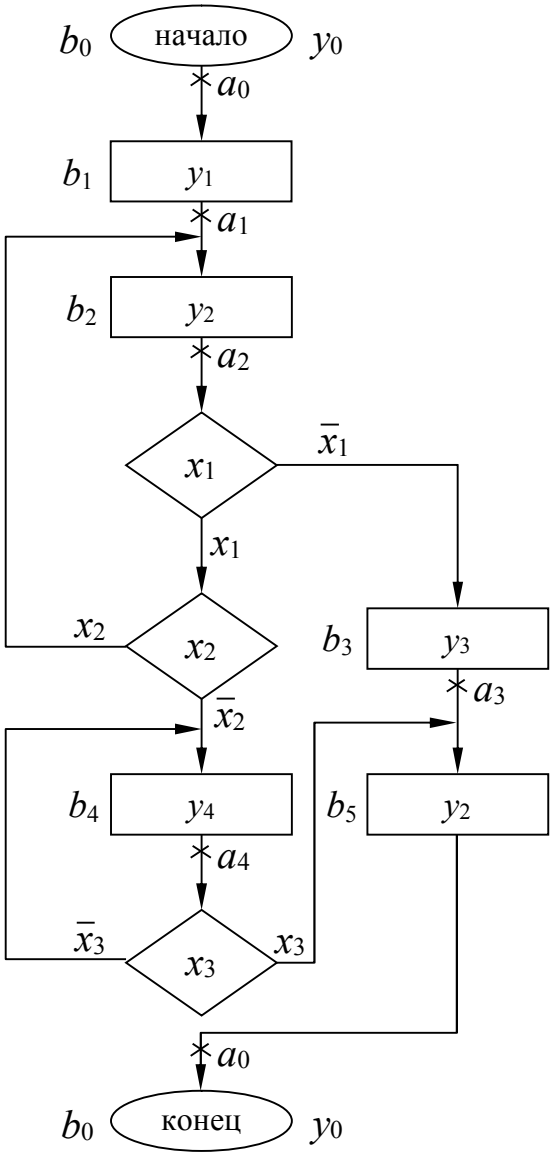

b_0
начало
y_0
a_0
b_1
y_1
a_1
b_2
y_2
a_2
x_1
$\bar{x}_1$
x_1
x_2
x_2
b_3
y_3
a_3
$\bar{x}_2$
b_4
y_4
b_5
y_2
a_4
$\bar{x}_3$
x_3
x_3
a_0
b_0
конец
y_0

Рис. 5.7

Для автомата Мура разметка выполняется аналогично за исключением символов состояний b_1, b_2, …, b_l, которыми отмечаются операторные вершины схемы.

По окончании выполнения алгоритма автоматы обоих типов должны возвращаться в исходное состояние.

Переходы автомата из состояния в состояние осуществляются под воздействием входных сигналов, соответствующих выполнению или невыполнению условия x_i. Невыполнению условия соответствует инверсное значение переменной x_i, выполнению условия – прямое значение переменной. При выполнении или невыполнении нескольких условий одновременно переход осуществляется под воздействием сигнала, представляемого конъюнкцией соответствующих прямых или инверсных значений переменных.

Рассмотрим пример. На рис. 5.7 изображена схема алгоритма, размеченная для задания автомата Мили и автомата Мура. Составим соответствующие им графы переходов (рис. 5.8, 5.9).

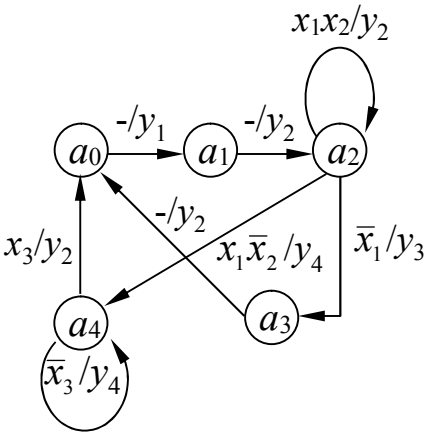


Рис. 5.8

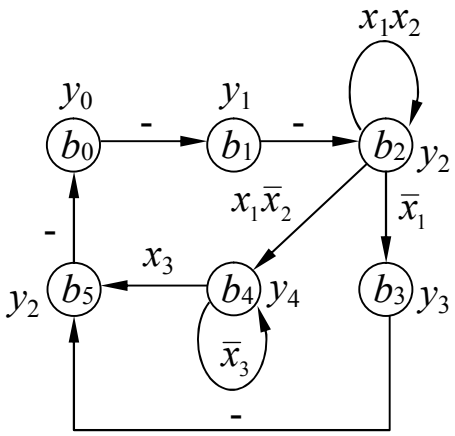


Рис. 5.9

5.1.5. Правила перехода между моделями Мили и Мура

Для каждого автомата Мура может быть найден эквивалентный ему автомат Мили, и наоборот. Рассмотрим правила перехода между этими двумя моделями.

I. **Переход от модели Мура к модели Мили**.

1. Входной и выходной алфавиты, множество состояний и таблица переходов автомата Мили те же, что и для автомата Мура.

2. Выходной сигнал автомата Мили, формируемый при переходе в состояние a_i, совпадает с выходным сигналом, которым отмечено состояние a_i в автомате Мура (табл. 5.3, 5.4).

II. **Переход от модели Мили к модели Мура**.

1. Входной и выходной алфавиты совпадают.

2. Формируется промежуточная таблица, в которой каждой паре автомата Мили $x_i a_j$ ставится в соответствие состояние автомата Мура b_{ij}. Кроме того, начальному состоянию автомата Мили ставится в соответствие начальное состояние автомата Мура b_0.

3. Строится таблица переходов автомата Мура, которая заполняется следующим образом. Каждое состояние b_{ij} отмечается выходным сигналом, записанным в ячейке автомата Мили на пересечении i-й строки и j-го столбца. Последующими для состояний автомата Мура будут состояния из промежуточной таблицы, записанные в столбце с номером a_k, где a_k – состояние автомата Мили, записанное в таблице переходов автомата Мили в ячейке на пересечении i-й строки и j-го столбца.

4. Последующими состояниями для b_0 будут последующие состояния для начального состояния автомата Мили в промежуточной таблице. Состояние b_0 отмечается таким выходным сигналом, чтобы столбец b_0 совпал с каким-либо другим столбцом таблицы переходов автомата Мура (табл. 5.5 – 5.7).

Рассмотрим примеры.

1. Автомат Мура → автомат Мили.

Таблица 5.3

y	1	4	2	3
$x\backslash b$	1	2	3	4
x_1	1	3	2	1
x_2	2	4	4	4

Таблица 5.4

$x\backslash a$	1	2	3	4
x_1	1/1	3/2	2/4	1/1
x_2	2/4	4/3	4/3	4/3

2. Автомат Мили → автомат Мура.

Таблица 5.5

$x\backslash a$	1	2	3	4
x_1	3/1	1/3	1/2	2/1
x_2	2/2	4/1	3/3	3/2

Таблица 5.6

$x\backslash a$	$b_0\backslash 1$	2	3	4
x_1	b_{11}	b_{12}	b_{13}	b_{14}
x_2	b_{21}	b_{22}	b_{23}	b_{24}

Таблица 5.7

y	3	1	3	2	1	2	1	3	2
$x\backslash b$	b_0	b_{11}	b_{12}	b_{13}	b_{14}	b_{21}	b_{22}	b_{23}	b_{24}
x_1	b_{11}	b_{13}	b_{11}	b_{11}	b_{12}	b_{12}	b_{14}	b_{13}	b_{13}
x_2	b_{21}	b_{23}	b_{21}	b_{21}	b_{22}	b_{22}	b_{24}	b_{23}	b_{23}

Пусть выходной сигнал для состояния b_0 такой же, как и для состояния b_{12}. Тогда столбец b_0 полностью совпадает со столбцом b_{12} и, следовательно, эти два состояния могут быть представлены одним, которое обозначим b_0. При этом столбец b_{12} может быть удален из таблицы, а все переходы в состояние b_{12} должны быть изменены на переходы в b_0.

5.2. Минимизация числа состояний цифровых автоматов

В общем случае таблица переходов ЦА, полученная тем или иным способом, может содержать избыточное число состояний. Поскольку от количества состояний абстрактного автомата зависит число элементов памяти, используемых при структурном синтезе ЦА, ставится задача сокращения избыточных состояний ЦА. При этом процесс минимизации числа состояний заключается в исключении некоторых состояний абстрактного автомата, а также в замене ряда состояний одним таким образом, чтобы получившийся в результате автомат был эквивалентен исходному, т.е. на одинаковую последовательность входных сигналов оба автомата (исходный и минимизированный, находящиеся в начальном состоянии) должны отвечать одинаковыми последовательностями выходных сигналов.

Рассмотрим приемы, позволяющие проводить предварительное сокращение числа состояний.

Во-первых, из таблицы переходов можно удалить все недостижимые состояния, т.е. такие, в которые автомат не сможет попасть из начального состояния под воздействием любой последовательности входных сигналов.

Во-вторых, из таблицы переходов автомата Мили можно исключить все состояния, которым соответствуют столбцы таблицы переходов, содержащие в своих ячейках только прочерки, т.е. неопределенные состояния перехода. При этом все вхождения исключенных по данному правилу состояний в упрощенной таблице переходов заменяются прочерками.

В-третьих, из таблицы переходов автомата Мура могут быть исключены все столбцы, соответствующие состояниям, отмеченным неопределенными выходными сигналами. Вхождения исключенных состояний в упрощенную таблицу переходов заменяются прочерками.

Кроме того, каждая группа состояний, которой соответствуют одинаковые столбцы таблицы переходов, может быть заменена одним из состояний группы, при этом все вхождения исключенных состояний в таблице переходов заменяются на состояние, номер которого представляет группу одинаковых столбцов.

Рассмотрим пример. Сформируем упрощенную таблицу переходов автомата, заданного табл. 5.8.

Таблица 5.8

$x\backslash a$	1	2	3	4	5	6	7
x_1	1,0	5,1	1,0	–,1	4,1	7,0	5,1
x_2	2,1	3,0	2,1	6,1	5,0	–,1	3,0

В этой таблице одинаковые столбцы соответствуют состояниям (1, 3) и (2, 7). Введем обозначения: группа состояний (1, 3) – 1, группа (2, 7) – 2.

Тогда столбцы табл. 5.8, соответствующие состояниям 3 и 7, можно исключить, а все вхождения этих состояний заменить состояниями 1 и 2 соответственно. В результате получим упрощенную таблицу переходов (табл. 5.9).

Таблица 5.9

$x\backslash a$	1	2	4	5	6
x_1	1,0	5,1	–,1	4,1	2,0
x_2	2,1	1,0	6,1	5,0	–,1

5.2.1. Минимизация числа состояний синхронного автомата методом Полла-Ангера

Будем полагать, что подлежащий минимизации автомат является синхронным автоматом Мили и дальнейшее сокращение числа его состояний с использованием вышеописанных приемов невозможно.

Совокупность состояний таблицы переходов, представляемых состоянием какой-либо другой таблицы переходов, будем называть *совместимым множеством* или *множеством совместимых состояний*. Задача минимизации заключается в нахождении множеств совместимых состояний с последующей заменой каждого из множеств одним состоянием.

Рассмотрим условия, которым должны удовлетворять совместимые состояния.

Первое условие совместимости – формирование непротиворечивых выходных сигналов при одинаковой входной переменной. Это означает, что в тех строках, где выходные сигналы определены, они должны совпадать. В табл. 5.10 состояния 1 и 2 являются совместимыми по выходам. Также совместимыми по выходам являются пары состояний 2, 3; 2, 4; 3, 4.

Таблица 5.10

$x \setminus a$	1	2	3	4
x_1	2,1	2,1	4,1	1,1
x_2	–,–	3,1	3,1	3,–
x_3	4,0	4,–	–,1	4,1
x_4	1,1	1,1	3,1	–,1

Второе условие совместимости состояний заключается в том, что переходы из этих состояний под воздействием одинакового входного сигнала должны осуществляться в одинаковые состояния, если оба они определены. Пара состояний 1, 2 удовлетворяет этому условию, так как под воздействием всех входных сигналов осуществляются переходы в одинаковые состояния. Пара состояний 2, 4 не удовлетворяет второму условию совместимости, так как под воздействием входного сигнала x_1 автомат переходит из состояний 2, 4 в состояния 2, 1.

Пары состояний, подобные 2, 4, называются *условно совместимыми*, так как их совместимость зависит от совместимости пары 2, 1 (что имеет место в данном примере). Пара состояний 2, 3 также является условно совместимой, так как их совместимость зависит от совместимости пар 2, 4 и 1, 3. Несовместимыми являются пары 1, 4 и 1, 3, поскольку для них не выполняется условие совместимости по выходам.

Для определения множеств совместимых состояний минимизируемого автомата можно пользоваться специальной треугольной таблицей, строки и столбцы которой соответствуют внутренним состояниям автомата. Столбцы обозначаются слева направо номерами состояний $1, 2, ..., z–1$. Строки нумеруются сверху вниз номерами состояний $2, 3, \ldots, z$. Каждой паре состояний соответствует одна клетка. В клетке ставится символ «X», если данная пара состояний несовместима по выходам. Совместимым парам состояний соответствуют клетки с

символом «V». Если совместимость рассматриваемой пары состояний зависит от совместимости других пар, то последние записываются в эту клетку.

После занесения в треугольную таблицу информации о каждой паре состояний необходимо все пары условно совместимых состояний перевести либо в разряд несовместимых, либо в разряд совместимых. Если условная совместимость пары зависит от пары, являющейся несовместимой, или в свою очередь являющейся условно совместимой, зависящей от несовместимой пары, то эта условно совместимая пара переводится в разряд несовместимых. Если на совместимость условно совместимой пары не влияют несовместимые пары состояний, то это пара является совместимой. Окончательный вид треугольной таблицы, где все пары являются совместимыми или несовместимыми, будем называть *картой финальных пар* (КФП).

Дальнейшая минимизация числа состояний заключается в попытке объединении пар совместимых состояний в более крупные группы, все состояния которых совместимы между собой, и замене каждой группы состояний одним, представляющим в минимальном автомате все состояния группы. Подобные группы состояний будем называть *максимально совместимыми множествами* (МС-множества).

Рассмотрим подход к формированию МС-множеств на примере. Пусть подлежащий минимизации автомат задан табл. 5.11. Определим для него совместимые пары состояний, которые запишем в треугольную таблицу, соответствующую рассматриваемому автомату (табл. 5.12).

Таблица 5.11

$x \setminus a$	1	2	3	4	5	6	7
x_1	3,0	–,–	4,1	6,1	5,–	4,1	2,0
x_2	–,–	6,0	5,–	–,–	1,–	7,1	3,0

Таблица 5.12

	1			
2	V	2		
3	X	5,6	3	
4	X	V	4,6	4

5	3,5	1,6	4,5 1,5	5,6	5	
6	X	X	5,7	V	4,5 1,7	6
7	2,3	3,6	X	X	2,5 1,3	X

В таблице 5.12 пронумерованы строки и столбцы. В дальнейшем будем использовать совмещенную нумерацию.

Переведем теперь условно совместимые пары состояний в разряд совместимых или в разряд несовместимых пар. Будем полагать, что по умолчанию все условно совместимые пары состояний могут быть совместимыми, если нет информации о том, что для каких-то из них это не так.

Рассмотрим пару состояний 2, 5. Согласно табл. 5.12, совместимость этой пары зависит от совместимости пары 1, 6, которая, как указано в таблице, является несовместимой. Следовательно, рассматриваемая пара состояний 2, 5 также будет являться несовместимой. Вследствие несовместимости 2, 5 также оказывается несовместимой пара 5, 7. Далее отмечаем несовместимость пар состояний 3, 6 и, как следствие, 2, 7. Остальные условно совместимые пары состояний полагаем совместимыми.

Составим отдельной таблицей карту финальных пар (табл. 5.13).

Таблица 5.13

1						
V	2					
X	V	3				
X	V	V	4			
V	X	V	V	5		
X	X	X	V	V	6	
V	X	X	X	X	X	7

Для нахождения МС-множеств можно воспользоваться следующим подходом.

Сначала для каждого состояния автомата из КФП определяются все пары совместимых состояний, в которые это состояние входит. Каждая пара входит в запись только один раз в столбце таблицы, соответствующем меньшему по номеру состоянию данной пары (табл. 5.14).

Таблица 5.14

1	2	3	4	5	6	7
1, 2	2, 3	3, 4	4, 5	5, 6	–	–
1, 5	2, 4	3, 5	4, 6			
1, 7						
(1, 2), (1, 5), (1, 7)	(2, 3, 4)	(3, 4, 5)	(4, 5, 6)	(5, 6)		

Далее проверяется возможность объединения пар в более крупные группы. Поскольку в данном примере столбцы 5, 6, 7 содержат не более одной пары, для них эта возможность не проверяется.

Последовательно рассмотрим интересующие нас столбцы табл. 5.14 слева направо. Объединять пары состояний можно только тогда, когда всевозможные пары состояний получаемой группы являются совместимыми. Для этого необходимо проанализировать содержимое столбцов таблицы, расположенных справа от рассматриваемого: если в этих столбцах имеются все необходимые пары, то объединение пар совместимых состояний в укрупненную группу возможно.

Для рассматриваемого примера искомые МС-множества записаны в табл. 5.14 отдельной строкой снизу.

МС-множества, являющиеся подмножествами других МС-множеств должны быть исключены. Если совокупность МС-множеств не содержит какие-либо состояния исходного автомата, они должны быть добавлены.

Совокупность МС-множеств содержит 6 элементов: C={1, 2; 1, 5; 1, 7; 2, 3, 4; 3, 4, 5; 4, 5, 6}, что соответствует 6 состояниям новой таблицы переходов эквивалентного автомата. Однако нетрудно заметить, что полученная совокупность МС-множеств содержит повторяющиеся состояния, которые необходимо попытаться исключить. Это может

привести к уменьшению числа МС-множеств и, как следствие, дальнейшему сокращению числа состояний автомата.

При исключении повторяющихся состояний следует понимать, что, во-первых, исключаемое из какого-либо МС-множества состояние должно остаться в каком-нибудь другом МС-множестве, а во-вторых, в первую очередь необходимо исключать такие повторяющиеся состояния, исключение которых приведет к нарушению совместимости как можно меньшего числа пар состояний.

Нетрудно видеть, что исключение МС-множества (1, 2) из C вполне допустимо, поскольку от совместимости этой пары не зависит совместимость других пар. Если исключить МС-множество (1, 5), согласно табл. 5.12 нарушится совместимость пары 3, 5, вследствие чего МС-множество (3, 4, 5) будет преобразовано к виду (3, 4). Таким образом, C ={1, 7; 2, 3, 4; 3, 4; 4, 5, 6} и МС-множество (3, 4) может быть исключено, поскольку оно является подмножеством (2, 3, 4). В оставшихся МС-множествах повторяется состояние 4, которое может быть исключено из (2, 3, 4), так как от совместимости пар состояний 2, 4 и 3, 4 не зависит совместимость других пар. Однако это не приведет к уменьшению числа МС-множеств, т.е. с точки зрения уменьшения числа состояний эквивалентного автомата исключение состояния 4 не принципиально.

Окончательно, C ={1, 7; 2, 3; 4, 5, 6}.

Для составления таблицы переходов минимального автомата необходимо обозначить полученные МС-множества номерами его состояний. Пусть номерам состояний 1, 2, 3 минимального автомата соответствуют множества состояний исходного автомата (1, 7), (2, 3) и (4, 5, 6). В принципе определение соответствия может быть выполнено произвольно, однако в некоторых случаях может быть удобным, если начальное состояние минимального автомата соответствует МС-множеству, содержащему начальное состояние исходного автомата.

Таблица переходов минимального автомата для рассматриваемого примера имеет следующий вид (табл. 5.15):

Таблица 5.15

$x \setminus a$	1	2	3
x_1	2,0	3,1	3,1
x_2	2,0	3,0	1,1

В заключение отметим, что рассмотренный подход без изменений может быть применен для минимизации числа состояний автоматов Мура.

5.2.2. Минимизация числа состояний автомата Мура методом *l*-эквивалентных разбиений

Вначале дадим несколько определений.

Два состояния автомата Мура называются *0-эквивалентными*, если они отмечены одинаковым выходным сигналом.

Совокупности 0-эквивалентных состояний образуют 0-класс.

Если два 0-эквивалентных состояния, принадлежащие к одному классу, любым входным сигналом переводятся также в 0-эквивалентные состояния, то они являются *1-эквивалентными*.

Совокупности 1-эквивалентных состояний образуют 1-класс.

Если разбиение на i-классы совпадает с разбиением на (i+1)-классы, то (i+1)-класс называется *∞-классом*.

Рассмотрим процесс минимизации на примере. Пусть исходный автомат задан табл. 5.16.

Таблица 5.16

y	0	1	1	1	0	1	0	0
$x \setminus a$	1	2	3	4	5	6	7	8
x_1	1	7	4	1	7	3	7	4
x_2	4	2	8	4	6	1	2	8
x_3	8	8	3	8	8	2	8	3

Произведем два последовательных разбиения: на 0-классы (табл. 5.17), а затем на 1-классы (табл. 5.18) и 2-классы (табл. 5.19).

Таблица 5.17

	A				*B*			
	1	5	7	8	2	3	4	6
x_1	*A*	*A*	*A*	*B*	*A*	*B*	*A*	*B*
x_2	*B*	*B*	*B*	*A*	*B*	*A*	*B*	*A*
x_3	*A*	*A*	*A*	*B*	*A*	*B*	*A*	*B*

Таблица 5.18

	A			*B*	*C*		*D*	
	1	5	7	8	2	4	3	6
x_1	*A*	*A*	*A*		*A*	*A*	*C*	*D*

x_2	*C*	*D*	*C*		*C*	*C*	*B*	*A*
x_3	*B*	*B*	*B*		*B*	*B*	*D*	*C*

Таблица 5.19

	A		*B*	*C*	*D*		*E*	*F*
	1	7	5	8	2	4	3	6
x_1	*A*	*A*			*A*	*A*		
x_2	*D*	*D*			*D*	*D*		
x_3	*C*	*C*			*C*	*C*		

Очевидно, что дальнейшие разбиения будут совпадать с разбиением на 2-классы. Таким образом, в эквивалентном автомате будут шесть состояний, соответствующих разбиению на 2-классы (A – 1, B – 2, C – 3, D – 4, E – 5, F – 6). В табл. 5.20 представлены переходы и выходные сигналы автомата.

Таблица 5.20

y	0	0	0	1	1	1
$x \setminus a$	1	2	3	4	5	6
x_1	1	1	4	1	4	5
x_2	4	6	3	4	3	1
x_3	3	3	5	3	5	4

5.2.3. Минимизация числа состояний автомата Мили методом *l*-эквивалентных разбиений

Два состояния автомата Мили будут 0-эквивалентными, если находящийся в этих состояниях автомат под воздействием эквивалентных входных сигналов выдает одинаковые выходные сигналы.

Два состояния a_i и a_j автомата Мили будут l-эквивалентными, если выполняются следующие условия:

1) a_i и a_j (l–1)-эквивалентны;

2) множества классов (l–1)-эквивалентных состояний, в которые осуществляются переходы из a_i и a_j, равны между собой;

3) множества выходных сигналов, формируемых на переходах из a_i и a_j в состояния из некоторого (l–1)-го класса эквивалентности, равны между собой;

4) входные сигналы, приводящие к формированию одинакового выходного сигнала из состояний a_i и a_j, эквивалентны.

Рассмотрим пример.

Представим таблицу переходов исходного автомата Мили в следующем виде (табл. 5.21).

Таблица 5.21

Исходное состояние	Состояние перехода	Входной сигнал	Выходной сигнал	Классы
1	2 3 4 5	$\overline{x}_1\overline{x}_2$ x_1x_4 $\overline{x}_1x_2$ $x_1\overline{x}_4$	1 3 2 6	B1
2	6 8 6 9	x_4x_3 $\overline{x}_4\overline{x}_3$ $\overline{x}_4x_3$ $x_4\overline{x}_3$	5 4 5 4	B2
3	6 7	x_3 $\overline{x}_3$	5 4	B2
4	6 8	x_3 $\overline{x}_3$	5 4	B2
5	2 3 4 5	$\overline{x}_1\overline{x}_2$ x_1x_4 $\overline{x}_1x_2$ $x_1\overline{x}_4$	1 3 2 6	B1
6	2 9	x_4 $\overline{x}_4$	3 5	B3
7	4 8	x_4 $\overline{x}_4$	3 5	B3
8	5	1	1	B4
9	1	1	1	B4

Из таблицы видно, что произведено разбиение на 4 класса: B1, B2, B3, B4. Для проведения следующего разбиения перепишем эту таблицу в другом виде (табл. 5.22).

Таблица 5.22

Исходное состояние	Состояние перехода	Входной сигнал	Выходной сигнал	Классы
1	B2 B2 B2 B1	$\bar{x}_1\bar{x}_2$ x_1x_4 $\bar{x}_1x_2$ $x_1\bar{x}_4$	1 3 2 6	C1
5	B2 B2 B2 B1	$\bar{x}_1\bar{x}_2$ x_1x_4 $\bar{x}_1x_2$ $x_1\bar{x}_4$	1 3 2 6	C1
2	B3 B4 B3 B4	x_4x_3 $\bar{x}_4\bar{x}_3$ $\bar{x}_4x_3$ $x_4\bar{x}_3$	5 4 5 4	C2
3	B3 B3	x_3 $\bar{x}_3$	5 4	C3
4	B3 B4	x_3 $\bar{x}_3$	5 4	C2
6	B2 B4	x_4 $\bar{x}_4$	3 5	C4
7	B2 B4	x_4 $\bar{x}_4$	3 5	C4
8	B1	1	1	C5
9	B1	1	1	C5

В результате очередного разбиения получены 5 классов: C1, C2, C3, C4, C5. Дальнейшее разбиение невозможно, поэтому сформируем таблицу переходов минимального автомата, имеющего пять внутренних состояний (табл. 5.23).

Таблица 5.23

Исходное состояние	Состояние перехода	Входной сигнал	Выходной сигнал
1	2 3 2 1	$\bar{x}_1\bar{x}_2$ x_1x_4 $\bar{x}_1x_2$ $x_1\bar{x}_4$	1 3 2 6
2	4 5	x_3 $\bar{x}_3$	5 4
3	4 4	x_3 $\bar{x}_3$	5 4
4	2 5	x_4 $\bar{x}_4$	3 5
5	1	1	1

5.3. Структурный синтез цифровых автоматов

Для структурной реализации автомата в виде цифровой схемы, состоящей, как указывалось ранее в подразделе 5.1.2 (см. рис. 5.4), из двух взаимосвязанных частей – КС и элементов памяти, необходимо задать набор элементов, обладающий свойством структурной полноты.

Набор элементарных автоматов (ЭА), выполняющих функции элементов памяти, и логических элементов, необходимых для построения комбинационной части автомата, будем называть *структурно полным*, если из элементов этого набора возможно построение схемы любого ЦА.

Будем говорить, что автомат обладает *полной системой переходов*, если для любой пары его состояний a_i, a_j найдется хотя бы один входной сигнал, переводящий его из состояния a_i в состояние a_j.

Автомат Мура будем называть *автоматом с полной системой выходов*, если каждому его состоянию a_i соответствует выходной сигнал, отличающийся от сигналов, соответствующих другим его состояниям.

Для того чтобы набор элементов для структурного синтеза автомата был функционально полным, необходимо и достаточно, чтобы он содержал хотя бы один элементарный автомат с двумя внутренними состояниями, характеризующийся полнотой систем переходов и выходов, а также набор логических элементов, образующих функционально полную систему ПФ (см. раздел 3.4).

5.3.1. Типы элементарных автоматов, обладающие полной системой переходов-выходов

1. Элементарный автомат с одним входом (рис. 5.10).

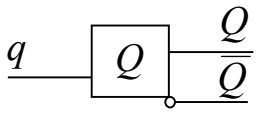


Рис. 5.10

Таблица 5.24

Функция возбуждения ЭА	Состояние ЭА	D	T
$q(t)$	$Q(t)$	$Q(t+1)$	$Q(t+1)$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

В табл. 5.24 представлены два варианта функционирования одновходового элементарного автомата, изображенного на рис. 5.10. Ясно, что вход и выходы схемы (прямой Q и инверсный $\overline{Q}$) двоичные: $q, Q \in \{0,1\}$.

Столбец «*D*» табл. 5.24 описывает работу D-триггера, называемого также *триггером задержки*. Функцию переходов такого элементарного автомата можно записать в виде

$$Q(t+1) = q(t) \cdot \overline{Q}(t) \vee q(t) \cdot Q(t) = q(t).$$

Таким образом, состояние D-триггера в момент времени (t+1) совпадает со значением сигнала на его входе в момент времени t.

Столбец «*T*» табл. 5.24 описывает работу T-триггера, называемого также *триггером со счетным входом*. Его функция переходов в аналитической форме имеет вид

$$Q(t+1) = q(t) \cdot \overline{Q}(t) \vee \overline{q}(t) \cdot Q(t) = q(t) \oplus Q(t).$$

В соответствии с функцией переходов, T-триггер меняет свое состояние на противоположное, когда на его входе образуется сигнал $q = 1$.

2. Элементарный автомат с двумя входами.

На рис. 5.11 изображен элементарный автомат с двумя входами, $q_0, q_1, Q \in \{0,1\}$. Функционирование такого ЭА в общем виде может быть представлено табл. 5.25.

Единичный сигнал на входе q_0 устанавливает состояние ЭА в момент времени (t+1) равным 0. Единичный сигнал на входе q_1 устанавливает состояние ЭА в момент времени (t+1) равным 1. Значения переменных a, b, c, d выбираются из $\{0,1,-\}$ в соответствии с конкретным типом ЭА.

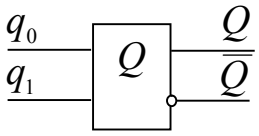


Рис. 5.11

Таблица 5.25

$q_0(t)$	$q_1(t)$	$Q(t)$	$Q(t+1)$
0	0	0	a
0	1	0	1
1	0	0	0
1	1	0	b
0	0	1	c
0	1	1	1
1	0	1	0
1	1	1	d

Введем обозначения $q_0(t) = R(t)$, $q_1(t) = S(t)$ и положим $a = 0$, $b = -$, $c = 1$, $d = -$. Тогда функционирование двухвходового ЭА может быть представлено табл. 5.26.

Таблица 5.26

$R(t)$	$S(t)$	$Q(t)$	$Q(t+1)$
0	0	$Q(t)$	$Q(t)$
1	0	$Q(t)$	0
0	1	$Q(t)$	1
1	1	$Q(t)$	–

Такой ЭА называется *RS-триггером*. Комбинация входных сигналов $R = 1$, $S = 1$ называется запрещенной. Она не должна появляться на входе RS-триггера, так как в случае появления этой входной комбинации состояние перехода триггера не определено. В зависимости от элементной базы, используемой для реализации RS-триггера, запрещенной комбинацией может быть $R = 0$, $S = 0$.

Если записать ограничение на появление на входах RS-триггера запрещенной комбинации $R(t)=1,\ S(t)=1$ в виде $R(t)\cdot S(t)=0$, то его функция переходов может быть представлена следующим образом:

$$Q(t+1)=S(t)\vee \overline{R}(t)\cdot Q(t),$$
$$R(t)\cdot S(t)=0.$$

Рассмотрим еще один ЭА с двумя входами, у которого отсутствует запрещенная комбинация входов (табл. 5.27).

Таблица 5.27

***K*(*t*)**	***J*(*t*)**	***Q*(*t*)**	***Q*(*t*+1)**
0	0	$Q(t)$	$Q(t)$
1	0	$Q(t)$	0
0	1	$Q(t)$	1
1	1	$Q(t)$	$\overline{Q}(t)$

Такой элементарный автомат называется *JK-триггером*. Вход J служит для установки 1, вход K служит для установки 0. При $J=K=1$ состояние JK-триггера меняется на противоположное.

При соответствующем подключении входов JK-триггера можно получить другие разновидности триггеров. Так, например, при объединении входов J и K в один вход JK-триггер функционирует в режиме Т-триггера, а при подключении входа K через инвертор к входу J схема работает в режиме D-триггера.

3. Элементарный автомат с тремя входами.

В качестве ЭА с тремя входами рассмотрим RST-триггер (табл. 5.28), который в зависимости от входных сигналов может работать в режиме RS- или Т-триггера. Разрешенные входные комбинации такого ЭА содержат не более одной единицы.

Таблица 5.28

***R*(*t*)**	***S*(*t*)**	***T*(*t*)**	***Q*(*t*)**	***Q*(*t*+1)**
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1

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

Функция переходов для такого ЭА может быть представлена как

$$Q(t+1) = S(t) \vee T(t) \cdot \overline{Q}(t) \vee \overline{R}(t) \cdot T(t) \cdot Q(t),$$

$$R(t) \cdot S(t) = R(t) \cdot T(t) = S(t) \cdot T(t) = 0,$$

где последнее выражение накладывает ограничения на разрешенные комбинации входных сигналов RST-триггера.

5.3.2. Основные этапы структурного синтеза

Процедуру структурного синтеза ЦА можно условно разделить на 5 этапов.

1. Определение числа физических входов, физических выходов и количества элементов памяти, необходимых для синтеза.

Число необходимых для синтеза элементов памяти определяется как

$$N =]\log_2 n[,$$

что дает наименьшее из решений также используемого для этих целей неравенства

$$N \geq \log_2 n,$$

где n – число состояний абстрактного автомата, $]x[$ – округление до ближайшего целого в большую сторону. Аналогично может быть определено число двоичных разрядов для кодирования входных и выходных сигналов.

В качестве элементов памяти используются элементарные автоматы, являющиеся автоматами Мура и имеющие два устойчивых состояния. Каждому состоянию ЭА соответствует свой выходной сигнал. Число входов обычно от 1 до 3.

2. Кодирование состояний автомата.

Под кодированием состояний ЦА понимается сопоставление каждому состоянию некоторого уникального двоичного кода, разрядность которого определяется числом используемых элементов памяти.

Основные этапы структурного синтеза рассмотрим на примере.

Пусть синтезируемый автомат Мили задан совмещенной таблицей переходов-выходов (табл. 5.29).

Таблица 5.29

$x \setminus a$	1	2	3	4
x_1	3/2	3/1	2/3	1/2
x_2	2/3	2/2	4/1	1/3

Для кодирования четырех состояний автомата понадобятся два двоичных разряда. Закодируем состояния автомата произвольными двухразрядными двоичными комбинациями Q_1Q_2 (табл. 5.30).

Кроме того, для удобства закодируем входные и выходные сигналы (табл. 5.31, 5.32). Для кодирования трех выходных сигналов понадобятся три двухразрядные двоичные комбинации Z_1Z_2. Для кодирования двух входных сигналов достаточно одного двоичного разряда S.

Таблица 5.30

$Q \setminus a$	1	2	3	4
Q_1	0	0	1	1
Q_2	0	1	0	1

Таблица 5.31

$Z \setminus y$	1	2	3
Z_1	0	1	1
Z_2	0	0	1

Таблица 5.32

$S \setminus x$	x_1	x_2
S	0	1

При кодировании входных и выходных сигналов автомата следует учитывать, что для синтеза схемы такого автомата может потребоваться реализация подсхем кодирования входных и декодирования выходных сигналов. Таким образом, общая цена схемы может увеличиться по отношению к цене схемы, синтезированной с использованием некодированных входных и выходных сигналов автомата.

В качестве ЭА, используемых для реализации памяти автомата, рассмотрим Т-триггеры (см. табл. 5.24).

3. Составление кодированной таблицы переходов и выходов синтезируемого автомата (табл. 5.33).

В каждую строку кодированной таблицы переходов и выходов записывают переход автомата из состояния a_i в состояние a_j под воздействием некоторого входного сигнала и формируемый при этом выходной сигнал. Состояния автомата записываются в таблицу в закодированном виде. Для автомата Мура выходные сигналы можно не указывать, так как они определяются только состояниями автомата и не зависят от входных сигналов.

В нижней строке табл. 5.33 указаны моменты формирования автоматом соответствующих сигналов.

Таблица 5.33

S	Q_1	Q_2	Q_1	Q_2	q_1	q_2	Z_1	Z_2
0	0	0	1	0	1	0	1	0
0	0	1	1	0	1	1	0	0
0	1	0	0	1	1	1	1	1
0	1	1	0	0	1	1	1	0
1	0	0	0	1	0	1	1	1
1	0	1	0	1	0	0	1	0
1	1	0	1	1	0	1	0	0
1	1	1	0	0	1	1	1	1
t	t		t+1		t		t	

В первом столбце табл. 5.33 записаны кодированные значения входных сигналов, формирующие переходы автомата.

В столбцах 2 и 3 записаны двоичные комбинации Q_1Q_2, кодирующие исходное состояние автомата в момент времени t, а в столбцах 4 и 5 – двоичные комбинации Q_1Q_2, кодирующие состояние автомата, в которое он переходит в момент времени (t+1) под воздействием соответствующего входного сигнала.

Столбцы 6, 7 (q_1 и q_2) составляются в соответствии с таблицей переходов выбранного типа ЭА, в данном примере Т-триггера (табл. 5.24), и описывают функции возбуждения элементов памяти, которые необходимо сформировать для осуществления переходов синтезируемого автомата. Например, если при выполнении перехода из некоторого состояния a_i в состояние a_j первый ЭА должен изменить свое значение из $Q_1(t)=1$ в $Q_1(t+1)=0$, то в соответствии с табл. 5.24 на его входе в момент времени t должен быть сформирован сигнал $q_1=1$.

Последние два столбца табл. 5.33 описывают выходные сигналы, формируемые синтезируемым автоматом Мили в момент перехода из одного состояния в другое. При синтезе автомата Мура последние два столбца не обязательны, поскольку его выходные сигналы однозначно определяются кодами состояний, в которых может находиться автомат.

В кодированной таблице переходов-выходов порядок заполнения строк, соответствующих переходам синтезируемого автомата, может быть произвольным. В некоторых случаях удобно упорядочить строки по какому-нибудь принципу. Например, в табл. 5.33 строки упорядочены по возрастанию двоичных кодов, соответствующих наборам сигналов (SQ_1Q_2), поступающих на входы КС автомата в момент времени t.

4. Задание и минимизация КС автомата.

По кодированной таблице переходов-выходов формируются функции выходов и функции возбуждения элементарных автоматов. В рассматриваемом автомате Мили функции возбуждения $q_1(t)$, $q_2(t)$ и функции выходов $Z_1(t)$, $Z_2(t)$ зависят от текущего состояния элементарных автоматов $Q_1(t)$, $Q_2(t)$ и от входного сигнала $S(t)$.

Составляя по кодированной таблице переходов-выходов соответствующие логические выражения для функций возбуждения элементов памяти и функций выходов, получим следующую систему ПФ, описывающую комбинационную часть синтезируемого автомата:

$$q_1 = \overline{S}\,\overline{Q_1}\,\overline{Q_2} \vee \overline{S}\,\overline{Q_1}Q_2 \vee \overline{S}\,\overline{Q_1}Q_2 \vee \overline{S}Q_1Q_2 \vee SQ_1Q_2;$$
$$q_2 = \overline{S}\,\overline{Q_1}Q_2 \vee \overline{S}\,\overline{Q_1}Q_2 \vee \overline{S}Q_1Q_2 \vee S\overline{Q_1}\,\overline{Q_2} \vee SQ_1\overline{Q_2} \vee SQ_1Q_2;$$
$$Z_1 = \overline{S}\,\overline{Q_1}\,\overline{Q_2} \vee \overline{S}Q_1\overline{Q_2} \vee \overline{S}Q_1Q_2 \vee S\overline{Q_1}\,\overline{Q_2} \vee S\overline{Q_1}Q_2 \vee SQ_1Q_2$$
$$Z_2 = \overline{S}Q_1\overline{Q_2} \vee S\overline{Q_1}\,\overline{Q_2} \vee SQ_1Q_2.$$

Ясно, что вышеприведенные функции следует минимизировать не по отдельности, а как систему ПФ (см. раздел 4.7). Однако это не всегда возможно вследствие ее громоздкости.

Для данного примера минимизированная система ПФ имеет вид

$$q_1 = \overline{S}\,\overline{Q_2} \vee \overline{S}Q_2 \vee Q_1Q_2;$$
$$q_2 = \overline{S}Q_1 \vee S\overline{Q_1}\,\overline{Q_2} \vee Q_1;$$
$$Z_1 = \overline{S}\,\overline{Q_2} \vee S\overline{Q_1}\,\overline{Q_2} \vee SQ_2 \vee Q_1Q_2$$
$$Z_2 = \overline{S}Q_1\overline{Q_2} \vee S\overline{Q_1}\,\overline{Q_2} \vee SQ_1Q_2.$$

5. Построение функциональной (логической) схемы автомата.

Пусть в рассматриваемом примере синтез комбинационной части автомата производится в базисе И-ИЛИ-НЕ. Логическая схема синтезируемого автомата изображена на рис. 5.12. Сигнал «*C*» снимается с выхода генератора синхронизирующих импульсов.

Структурный синтез цифрового автомата можно осуществлять с использованием некодированных входных и выходных сигналов.

Рассмотрим пример.

Осуществим структурный синтез автомата, заданного графом переходов, изображенным на рис. 5.13.

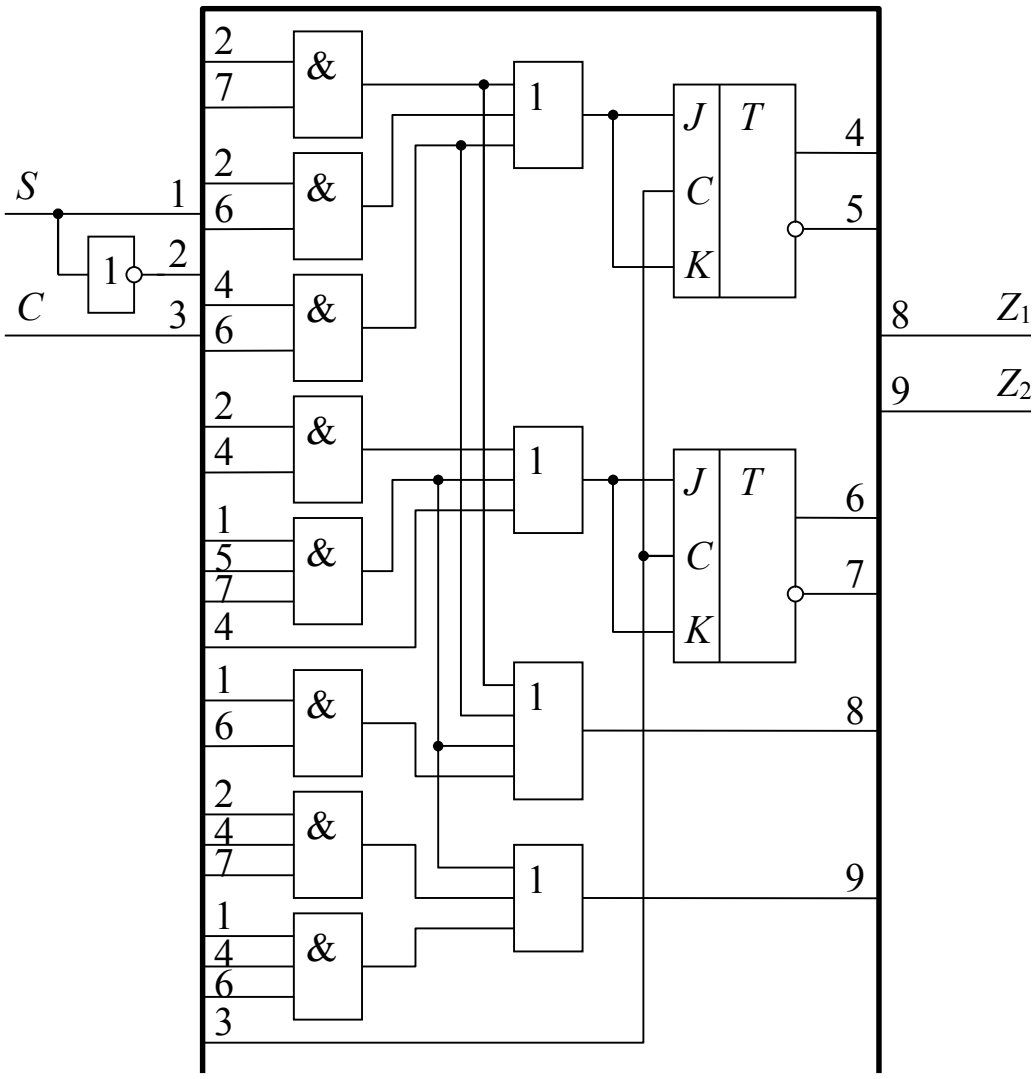
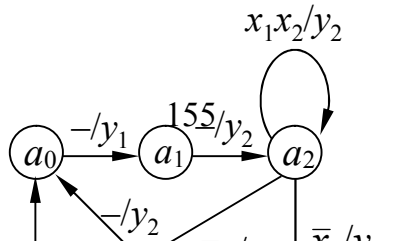


Рис. 5.12



Пусть в качестве элементарных автоматов используются Т-триггеры, а для реализации комбинационной схемы используется логический базис И-ИЛИ-НЕ.

Для кодирования пяти состояний потребуются три двоичных разряда Q_1, Q_2, Q_3. Закодируем состояния синтезируемого автомата в соответствии с табл. 5.34, а входные и выходные сигналы оставим в незакодированном виде.

Таблица 5.34

$a \setminus Q$	Q_1	Q_2	Q_3
0	0	1	1
1	1	0	0
2	0	0	0
3	0	0	1
4	0	1	0

Тогда кодированная таблица переходов-выходов может быть представлена в виде табл. 5.35.

Таблица 5.35

Вх. сигнал	a_i	$Q_1Q_2Q_3$	a_j	$Q_1Q_2Q_3$	$q_1q_2q_3$	Вых. сигнал
–	0	0 1 1	1	1 0 0	1 1 1	y_1
–	1	1 0 0	2	0 0 0	1 0 0	y_2

x_1x_2	2	0 0 0	2	0 0 0	0 0 0	y_2
$\bar{x}_1$	2	0 0 0	3	0 0 1	0 0 1	y_3
$x_1\bar{x}_2$	2	0 0 0	4	0 1 0	0 1 0	y_4
–	3	0 0 1	0	0 1 1	0 1 0	y_2
$\bar{x}_3$	4	0 1 0	4	0 1 0	0 0 0	y_4
x_3	4	0 1 0	0	0 1 1	0 0 1	y_2
t			t+1		t	

Запишем функции выходов и функции возбуждения:

$$y_1 = \overline{Q}_1 Q_2 Q_3;\ y_2 = Q_1\overline{Q}_2\overline{Q}_3 \vee x_1 x_2 \overline{Q}_1\overline{Q}_2\overline{Q}_3 \vee \overline{Q}_1\overline{Q}_2 Q_3 \vee x_3\overline{Q}_1 Q_2\overline{Q}_3;$$
$$y_3 = \bar{x}_1\overline{Q}_1\overline{Q}_2\overline{Q}_3;\ y_4 = x_1\bar{x}_2\overline{Q}_1\overline{Q}_2\overline{Q}_3 \vee \bar{x}_3\overline{Q}_1 Q_2\overline{Q}_3;$$
$$q_1 = \overline{Q}_1 Q_2 Q_3 \vee Q_1\overline{Q}_2\overline{Q}_3;$$
$$q_2 = \overline{Q}_1 Q_2 Q_3 \vee x_1\bar{x}_2\overline{Q}_1\overline{Q}_2\overline{Q}_3 \vee \overline{Q}_1\overline{Q}_2 Q_3;$$
$$q_3 = \overline{Q}_1 Q_2 Q_3 \vee \bar{x}_1\overline{Q}_1\overline{Q}_2\overline{Q}_3 \vee x_3\overline{Q}_1 Q_2\overline{Q}_3.$$

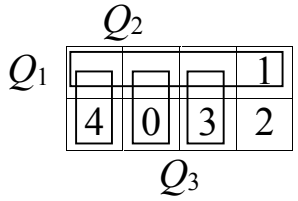
Введем следующие обозначения:

$$b_0 = \overline{Q}_1 Q_2 Q_3;\ b_1 = Q_1\overline{Q}_2\overline{Q}_3;\ b_2 = \overline{Q}_1\overline{Q}_2\overline{Q}_3;\ b_3 = \overline{Q}_1\overline{Q}_2 Q_3;\ b_4 = \overline{Q}_1 Q_2\overline{Q}_3;$$
$$b_5 = x_1 x_2 b_2;\ b_6 = x_1\bar{x}_2 b_2;\ b_7 = x_3 b_4.$$

Тогда выражения для функций возбуждения и функций выходов могут быть записаны следующим образом:

$$y_1 = b_0;\ y_2 = b_1 \vee b_5 \vee b_3 \vee b_7;$$
$$y_3 = \bar{x}_1 b_2;\ y_4 = b_6 \vee \bar{x}_3 b_4;$$
$$q_1 = b_0 \vee b_1;$$
$$q_2 = b_0 \vee b_6 \vee b_3;$$
$$q_3 = b_0 \vee y_3 \vee b_7.$$

Функции b_0, b_1, b_2, b_3, b_4 описывают дешифратор состояний. Для кодирования пяти состояний автомата использованы пять трехразрядных двоичных наборов. Остальные три набора являются несущественными и могут быть использованы для минимизации функций дешифрирования состояний. Воспользуемся картой Карно, в клетках которой записаны номера состояний, соответствующие выбранному варианту кодирования. Свободные ячейки можно использовать для минимизации кодов состояний:



Таким образом, функции $b_0 - b_4$ примут вид

$b_0 = Q_2 Q_3;\ b_1 = Q_1;\ b_2 = \overline{Q}_1 \overline{Q}_2 \overline{Q}_3;\ b_3 = \overline{Q}_2 Q_3;\ b_4 = Q_2 \overline{Q}_3.$

5.4. Рациональный выбор варианта кодирования состояний синхронных автоматов

При выполнении структурного синтеза автомата можно показать, что сложность функций возбуждения и функций выходов существенно зависит не только от используемого типа элементарных автоматов, но и от выбранного варианта кодирования состояний.

Кодирование состояний абстрактного автомата, при котором получаются минимальные (в смысле необходимого для технической реализации оборудования) функции возбуждения и выходов, – процесс довольно сложный, связанный с перебором большого числа вариантов. Поэтому в дальнейшем остановимся на рассмотрении правил кодирования, приводящих, в большинстве случаев, к получению близких к минимальным (а иногда и к минимальным) схемам на элементах И, ИЛИ, НЕ, реализующим функции возбуждения элементов памяти синтезируемого автомата. Все дальнейшие рассуждения будем вести в предположении, что для кодирования состояний всегда будет использоваться минимально возможное число элементов памяти.

Рассмотрим абстрактный автомат, заданный таблицей переходов-выходов (табл. 5.36).

Таблица 5.36

$x \setminus a$	1	2	3
x	3,0	1,1	2,0
$\bar{x}$	3,1	3,1	2,0

Пусть состояния автомата закодированы в общем виде в соответствии с табл. 5.37, где $b_{ij} \in \{0, 1\}$.

Таблица 5.37

$a \setminus Q$	Q_1	Q_2
1	b_{11}	b_{21}
2	b_{12}	b_{22}
3	b_{13}	b_{23}

Составим кодированную таблицу переходов (табл. 5.38).

Таблица 5.38

x	Q_1	Q_2	Q_1	Q_2
0	b_{11}	b_{21}	b_{13}	b_{23}
0	b_{12}	b_{22}	b_{13}	b_{23}
0	b_{13}	b_{23}	b_{12}	b_{22}
1	b_{11}	b_{21}	b_{13}	b_{23}
1	b_{12}	b_{22}	b_{11}	b_{21}
1	b_{13}	b_{23}	b_{12}	b_{22}
	t	t		t+1

Пусть $Q_i^{b_{ij}} = \begin{cases} \overline{Q}_i, \text{ если } b_{ij} = 0, \\ Q_i, \text{ если } b_{ij} = 1. \end{cases}$

Условимся, что в качестве элементарных автоматов используются D-триггеры, значения функций возбуждения которых, в соответствии с таблицей переходов, совпадают со значениями $Q(t+1)$. Тогда функции возбуждения элементарных автоматов будут иметь вид

$$q_1 = b_{11}(xQ_1^{b_{12}}Q_2^{b_{22}}) \vee b_{12}(Q_1^{b_{13}}Q_2^{b_{23}}) \vee b_{13}(Q_1^{b_{11}}Q_2^{b_{21}} \vee \bar{x}Q_1^{b_{12}}Q_2^{b_{22}});$$
$$q_2 = b_{21}(xQ_1^{b_{12}}Q_2^{b_{22}}) \vee b_{22}(Q_1^{b_{13}}Q_2^{b_{23}}) \vee b_{23}(Q_1^{b_{11}}Q_2^{b_{21}} \vee \bar{x}Q_1^{b_{12}}Q_2^{b_{22}}).$$

Приравнивание переменных b_{ij} к нулю или единице дает возможность оценить влияние на сложность схемы любого частного варианта кодирования.

Рациональный вариант кодирования выбирается таким образом, чтобы максимально упростить уравнения для q_1 и q_2. Для этого можно пользоваться двумя эмпирическими правилами.

1. Положить равными нулю переменные b_{ij}, которые в уравнениях для q имеют наиболее сложные сомножители (коэффициенты).

2. Переменным b_{ij} необходимо приписывать такие значения, чтобы ненулевые члены уравнений могли быть взаимно упрощены.

При использовании этих правил важно помнить, что каждому состоянию автомата должна соответствовать только одна кодовая комбинация.

В рассматриваемом примере в соответствии с первым правилом положим $b_{13} = b_{23} = 0$. Это приведет к тому, что в уравнениях для q_1 и q_2 последние слагаемые будут равны нулю и могут быть отброшены, а

состояние автомата a_3 будет кодироваться двоичным набором 00. Если теперь положить $b_{11} = 0$, то $b_{21} = 1$, так как кодовая комбинация 00 уже используется для кодирования состояния a_3. Таким образом, состояние a_1 кодируется двоичным набором 01. Для кодирования состояния a_2 используем набор 10, что означает приравнивание к нулю переменной b_{22} и приравнивание к единице переменной b_{12}.

Окончательно функции возбуждения примут вид

$$q_1 = \overline{Q}_1\overline{Q}_2;$$
$$q_2 = xQ_1\overline{Q}_2.$$

На основании приведенных выше правил и рассмотренного примера можно сделать вывод, что те состояния, в которые осуществляется большое число переходов, должны кодироваться наборами с минимальным числом единиц, а те состояния, между которыми имеются переходы, должны кодироваться наборами, отличающимися, по возможности, наименьшим количеством разрядов.

Вопросы для самоконтроля

1. Что представляет собой обобщенная схема ЦА? Какие компоненты включает и какие между ними установлены связи?
2. Какие модели и способы задания ЦА вам известны?
3. В чем состоит идея минимизации числа состояний абстрактного ЦА? Какие методы минимизации вам известны?
4. Каким условиям должен удовлетворять набор элементов для структурного синтеза ЦА?
5. Какие типы элементарных автоматов (триггеров) вам известны?
6. Каковы основные этапы структурного синтеза цифровых автоматов?
7. На что в автомате может повлиять выбор того или иного варианта кодирования его внутренних состояний?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Савельев, А. Я. Основы информатики : учеб. пособие для вузов / А. Я. Савельев. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2000. – 314 с.

2. Потапов, В. И. Компьютерная арифметика и алгоритмическое моделирование арифметических операций : учеб. пособие / В.И. Потапов, О.П. Шафеева. – Омск: Изд-во ОмГТУ, 2005. – 96 с.

3. Пальянов, И. А. Проектирование цифровых автоматов : учеб. пособие / И. А. Пальянов. – Изд. 2-е, испр. и доп. – Омск: Изд-во ОмГТУ, 2001. – 100 с.

4. Прикладная теория цифровых автоматов / К. Г. Самофалов [и др.]. – Киев : Вища шк.: Головное изд-во, 1987. – 470 с.

5. Пальянов, И. А. Операции двоичной и десятичной арифметики в ЭВМ: метод. указания к практическим занятиям по курсу «Прикладная теория цифровых автоматов» / И. А. Пальянов. – Омск : Изд-во ОмПИ, 1990. – 36 с.

6. Глушков, В. М. Синтез цифровых автоматов / В. М. Глушков. – М. : Физматгиз, 1962. – 476 с.

7. Вавилов, Е. Н. Синтез схем электронных цифровых машин / Е. Н. Вавилов, Г. П. Портной. – М. : Сов. радио, 1963. – 440 с.

8. Майоров, С. А. Принципы организации цифровых машин / С. А. Майоров, Г. И. Новиков. – Л. : Машиностроение, 1974. – 384 с.

9. Савельев, А. Я. Прикладная теория цифровых автоматов / А. Я. Савельев. – М. : Высш. шк., 1987. – 272 с.

10. Папернов, А. А. Логические основы цифровой вычислительной техники / А.А. Папернов. – Изд. 3-е, перераб. и доп. – М. : Сов. радио, 1972. – 392 с.

11. Самофалов, К. Г. Электронные цифровые вычислительные машины: учебник / К. Г. Самофалов, В. Н. Корнейчук, В. П. Тарасенко. – Киев : Вища шк., 1976. – 480 с.

12. Каган, Б. М. Электронные вычислительные машины и системы: учеб. пособие для вузов / Б. М. Каган. – М. : Энергия, 1979. – 428 с.

Приложение

Задания для выполнения самостоятельных работ

В табл. 1 даны варианты заданий. Из столбца «Номера заданий» определяются типы арифметических операций, представленные в табл. 2.

Таблица 1

Вариант	Номера заданий	Вариант	Номера заданий
1	(1,20)	**16**	(18,7)
2	(2,19)	**17**	(19,8)
3	(3,18)	**18**	(20,9)
4	(4,17)	**19**	(2,20)
5	(5,16)	**20**	(4,19)
6	(6,15)	**21**	(5,18)
7	(7,14)	**22**	(12,17)
8	(8,13)	**23**	(7,16)
9	(9,12)	**24**	(8,15)
10	(10,12)	**25**	(9,14)
11	(11,13)	**26**	(10,20)
12	(14,2)	**27**	(11,19)
13	(15,4)	**28**	(14,4)
14	(16,5)	**29**	(15,5)
15	(17,6)	**30**	(16,6)

Таблица 2

№	Арифметические операции
1	Неускоренное умножение младшими разрядами вперед; прямой код.
2	Неускоренное умножение старшими разрядами вперед; прямой код.
3	Неускоренное умножение в доп. коде младшими разрядами вперед.
4	Неускоренное умножение в обр. коде младшими разрядами вперед.
5	Неускоренное умножение в обр. коде старшими разрядами вперед.
6	Умножение с анализом последовательностей «0» и «1», прямой код, мл. р-ми вперед.
7	Умножение с анализом последовательностей «0» и «1», доп. код, мл. р-ми вперед.
8	Умножение с анализом последовательностей «0» и «1», обр. код, мл. р-ми вперед.
9	Умножение с анализом двух разрядов в прямом коде, мл. разрядами вперед.
10	Умножение с анализом двух разрядов в доп. коде, мл. разрядами вперед.
11	Умножение с анализом двух разрядов в обр. коде, мл. разрядами вперед.
12	Сложение (инверсные коды).
13	Вычитание (инверсные коды).
14	Неускоренное деление с восстановлением остатка в прямом коде.
15	Неускоренное деление без восстановления остатка в прямом коде.
16	Ускоренное деление с анализом старшего разряда, прямой код.
17	Ускоренное деление с анализом двух разрядов, прямой код.
18	Неускоренное деление без восстановления остатка в доп. коде.
19	Извлечение квадратного корня с восстановлением остатка, доп. код.
20	Извлечение квадратного корня без восстановления остатка, доп. код.

Учебное издание

Илья Викторович ПОТАПОВ, канд. техн. наук, доцент

ЭЛЕМЕНТЫ ПРИКЛАДНОЙ ТЕОРИИ ЦИФРОВЫХ АВТОМАТОВ

Учебное пособие

Редактор *Ю. Ю. Аптрашева*
Компьютерная верстка – *Е. В. Беспалова*

ИД № 06039 от 12.10.2001 г.
Сводный темплан 2011 г.
Подписано в печать 26.01.11. Формат 60×84 $^{1}/_{16}$. Бумага офсетная.
Отпечатано на дупликаторе. Усл. печ. л. 9,75. Уч.-изд. л.9,75.
Тираж 150 экз. Заказ 110.

Издательство ОмГТУ. 644050, г. Омск, пр. Мира, 11; т. 23-02-12
Типография ОмГТУ