

С. Н. Лехин

СХЕМОТЕХНИКА ЭВМ

- Арифметические и логические основы цифровой техники
- Схемотехника цифровых логических элементов
- Комбинационные и последовательностные схемы
- Полупроводниковые запоминающие устройства
- Программируемая логика
- Цифроаналоговые и аналого-цифровые преобразователи



bhy®



УДК 681.3.06
ББК 32.973.26-018.2
Л152

Лехин С. Н.

Л152 Схемотехника ЭВМ. — СПб.: БХВ-Петербург, 2010. — 672 с.: ил. —
(Учебная литература для вузов)

ISBN 978-5-9775-0353-2

Рассматриваются схемотехнические решения, используемые при построении цифровых логических элементов, вопросы синтеза комбинационных и последовательностных цифровых устройств по заданному алгоритму работы, а также процедуры анализа их функционирования. Освещены методы анализа помех в линиях передачи цифровых сигналов и цепях питания. Приведены структуры и схемотехника полупроводниковых запоминающих устройств, схем программируемой логики и вспомогательных узлов цифровой техники. Рассмотрены вопросы схемотехники цифроаналоговых и аналого-цифровых преобразователей информации.

*Для студентов технических вузов, инженеров и специалистов,
работающих в области разработки цифровой аппаратуры*

УДК 681.3.06
ББК 32.973.26-018.2

Рецензенты:

Ю. М. Смирнов, д. т. н., профессор, член-корреспондент РАН, завкафедрой "Интегрированные системы управления" Санкт-Петербургского государственного политехнического университета;

Г. М. Емельянов, д. т. н., профессор, завкафедрой программного обеспечения вычислительной техники института электронных и информационных систем Новгородского государственного университета им. Я. Мудрого.

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Юрий Рожко</i>
Компьютерная верстка	<i>Натальи Караваевой</i>
Корректор	<i>Виктория Пиотровская</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Елены Беляевой</i>
Фото	<i>Кирилла Сергеева</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 01.10.09.

Формат 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 32,25.

Тираж 1500 экз. Заказ №

"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0353-2

© Лехин С. Н., 2009
© Оформление, издательство "БХВ-Петербург", 2009

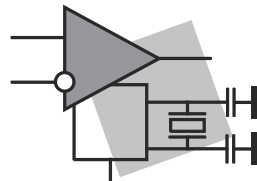
Оглавление

Введение.....	1
Глава 1. Арифметические и логические основы цифровой техники	3
1.1. Системы счисления, кодированное представление чисел для обработки в ЦВМ.....	3
1.2. Логические переменные и функции.....	10
Контрольные вопросы	30
Глава 2. Схемотехника цифровых логических элементов	31
2.1. Принципы построения логических элементов.....	31
2.2. Основные параметры и характеристики логических элементов.....	41
2.3. Транзисторный ключ.....	48
2.4. Схемотехника ТТЛ-вентилей	55
2.5. Разновидности ТТЛ и ТТЛШ логических элементов	84
2.6. ЭСЛ логические элементы.....	94
2.7. Логические элементы на КМОП-структурах	104
2.8. Способы согласования логических элементов	119
Контрольные вопросы	125
Глава 3. Помехи в цепях питания цифровых узлов и линиях передачи управляющих сигналов.....	127
3.1. Причины возникновения помех по цепям питания и методы борьбы с ними	127
3.2. Влияние параметров линий связи на процессы передачи цифровых сигналов.....	137
3.3. Методы согласования линий связи	165
3.4. Способы уменьшения помех при передаче цифровых сигналов	173
Контрольные вопросы	176

Глава 4. Цифровые узлы и устройства комбинационного типа	177
4.1. Классификация цифровых устройств	177
4.2. Состязания в комбинационных схемах	180
4.3. Преобразователи кода	194
4.4. Дешифраторы	197
4.5. Шифратор приоритетов	213
4.6. Мультиплексоры	220
4.7. Схемы контроля четности	230
4.8. Мажоритарные элементы	235
4.9. Цифровые компараторы	238
4.10. Сумматоры двоичных кодов чисел	243
4.11. Устройства вычитания двоичных кодов чисел	258
4.12. Сумматоры двоично-десятичных кодов	262
4.13. Арифметико-логические устройства	267
4.14. Умножители двоичных кодов чисел	270
Контрольные вопросы	274
Глава 5. Цифровые устройства последовательностного типа.....	275
5.1. Триггеры	275
5.2. Регистры	301
5.3. Накапливающий сумматор	318
5.4. Кольцевой счетчик.....	321
5.5. Счетчик Джонсона.....	332
5.6. Двоичные счетчики.....	340
5.7. Счетчики с произвольным и управляемым модулем счета	360
5.8. Делители и синтезаторы частоты.....	382
Контрольные вопросы	393
Глава 6. Полупроводниковые запоминающие устройства	395
6.1. Классификация и структурная организация полупроводниковых запоминающих устройств	395
6.2. Схемотехника ячеек накопителей статических запоминающих устройств	422
6.3. Динамические запоминающие устройства.....	430
6.4. Постоянные и перепрограммируемые запоминающие устройства	456
Контрольные вопросы	469
Глава 7. Программируемые логические интегральные схемы.....	471
7.1. Принципы обработки цифровых данных	471
7.2. Способы реализации логических функций	476

7.3. Принципы построения и элементы программируемых логических интегральных схем	483
7.4. Периферийное сканирование цифровых устройств	496
Контрольные вопросы	500
Глава 8. Интерфейсные и вспомогательные цифровые узлы	501
8.1. Периферийные узлы цифровых устройств.....	501
8.2. Формирователи импульсов на логических элементах	506
8.3. Генераторы цифровых сигналов на логических элементах.....	514
8.4. Генераторы с кварцевой стабилизацией частоты	531
Контрольные вопросы	538
Глава 9. Элементы и системы отображения цифровой информации	539
9.1. Способы управления одиночными светодиодными индикаторами.....	539
9.2. Системы отображения многоразрядных цифровых данных	543
9.3. Жидкокристаллические индикаторы и способы управления ими	550
Контрольные вопросы	560
Глава 10. Цифроаналоговые и аналого-цифровые преобразователи.....	561
10.1. Основные параметры и характеристики ЦАП	568
10.2. Принципы построения ЦАП прямого преобразования.....	571
10.3. Умножающие ЦАП.....	594
10.4. ЦАП с косвенным преобразованием	598
10.5. Области применения ЦАП	602
10.6. Основные параметры и характеристики аналого-цифровых преобразователей	604
10.7. Аналого-цифровые преобразователи с непосредственным преобразованием	609
10.8. АЦП с косвенным преобразованием	629
10.9. Области применения АЦП	647
Контрольные вопросы	648
Литература	651
Предметный указатель	655

Глава 1



Арифметические и логические основы цифровой техники

1.1. Системы счисления, кодированное представление чисел для обработки в ЦВМ

Для записей результатов количественных (числовых) измерений используются наборы символов. Способ представления чисел, при котором они отображаются определенными комбинациями символов, называется *системой счисления*. Одно и то же число допускает различные варианты его представления.

Числовые данные отличаются от остальных тем, что с ними могут производиться арифметические операции (сложение, умножение и т. п.). От выбранного способа записи чисел, т. е. от системы счисления, зависит как наглядность их представления, так и сложность выполнения процедур обработки.

Обычно количество символов (цифр) для записи чисел ограничено, поэтому для больших чисел цифры в его записи будут повторяться. Если значение цифры не зависит от ее местоположения в записи числа, то такая система счисления называется *непозиционной*. Примером является римская, с помощью которой, например, размечаются циферблаты некоторых часов.

Здесь используются так называемые *римские цифры* I, V, X, L и другие. Первая соответствует единице, вторая пятерке, третья десятке, а четвертая — сотне. Запись XXVII означает $10 + 10 + 5 + 1 + 1 = 27$, запись XI соответствует числу $10 + 1$ (одиннадцать) и т. д. В римской и аналогичных непозиционных системах счисления можно записать любое число, однако для этого потребуется достаточно большое количество символов и, кроме того, имеются определенные проблемы с выполнением арифметических операций.

Поэтому в цифровой технике используются *позиционные системы счисления*. Их особенность в том, что значение одной и той же цифры зависит от ее местоположения в записи числа, т. е. от позиции, в которой она располагается. Например, запись 222 в десятичной системе счисления, где используются

десять цифр от 0 до 9, означает, что данное число содержит две сотни, два десятка и две единицы. Позиция, которую занимает цифра, называется *разрядом*. Таким образом, в рассматриваемой записи имеются разряды сотен, десятков и единиц. Числовое значение разряда соответствует его весу.

То же число 222 можно представить как $2 \cdot 100 + 2 \cdot 10 + 2 \cdot 1$, откуда следует, что веса разрядов отличаются в десять раз. Если отношение соседних весов одинаково и равно P , это число называется *основанием системы счисления*. Обычно в позиционных системах значения весов определяются степенями ее основания, и число 222 представляется как $2 \cdot 10^2 + 2 \cdot 10^1 + 2 \cdot 10^0$.

Номер разряда, отсчитываемый справа налево, начиная с нуля, соответствует показателю степени, в которую требуется возвести основание системы счисления, чтобы получить величину веса разряда. Количество символов (цифр) для отображения чисел в позиционных системах счисления не может превышать величины основания, в противном случае возникает неоднозначность записи числа.

Веса разрядов в позиционных системах счисления могут и не находиться в одинаковых отношениях, к примеру 40, 20, 10, 8, 4, 2, 1. В этом случае понятие основания не вводится, и запись произвольного n -разрядного числа $x_{n-1}x_{n-2} \dots x_1x_0$ может быть представлена следующим образом:

$$x_{n-1}P_{n-1} + x_{n-2}P_{n-2} + \dots + x_1P_1 + x_0P_0 = \sum_{i=0}^{n-1} x_iP_i. \quad (1.1)$$

Здесь x_i — цифра i -того разряда, а P_i — его вес.

Если веса находятся в отношениях, пропорциональных основанию P , то предыдущее соотношение можно преобразовать к виду:

$$x_{n-1}P^{n-1} + x_{n-2}P^{n-2} + \dots + x_1P^1 + x_0P^0 = \sum_{i=0}^{n-1} x_iP^i. \quad (1.2)$$

Аналогичным образом представляются и числа меньшие единицы, для чего используются отрицательные степени основания. К примеру, число 12,34 записывается как $1 \cdot 10^1 + 2 \cdot 10^0 + 3 \cdot 10^{-1} + 4 \cdot 10^{-2}$.

Основание системы счисления может быть произвольным, но обычно оно выбирается из требований удобства выполнения действий над числовыми данными в конкретной ситуации. В частности, когда требуется обрабатывать данные в цифровых вычислительных машинах, целесообразным является использование двоичной системы счисления.

Это связано с тем, что в цифровой технике применяются элементы, обладающие двумя устойчивыми состояниями, одному из которых можно припи-

сать нулевое значение, а другому — единичное. В *двоичной системе счисления* основание равно двум и для отображения чисел имеется лишь две цифры 0 и 1.

Любое целое число в такой системе может быть представлено в виде

$$N = a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \dots + a_12^1 + a_02^0. \quad (1.3)$$

Здесь веса разрядов равны соответствующим степеням двойки, т. е. 1, 2, 4, 8, 16 и т. д.

При необходимости перевода числа из десятичной системы счисления в двоичную его требуется представить в виде суммы степеней основания, т. е. двойки. Например: $21 = 16 + 4 + 1 = 2^4 + 2^2 + 2^0$. Однако, с учетом того, что в записи числа должны присутствовать все разряды, полученное соотношение требуется дополнить недостающими. Чтобы результат в целом остался неизменным, эти разряды необходимо умножить на ноль, а остальные на единицу. В итоге запись числа 21 будет иметь вид $1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$, а само число в двоичной системе счисления запишется следующим образом, 10101.

Существуют и иные подходы для перевода десятичных чисел в двоичную систему. Один из них основан на вычислении остатков от последовательного деления исходного числа на два. Если 21 поделить на два, то получится целое число 10 и остаток, равный 1. Он будет младшей значащей цифрой двоичного кода этого числа. Далее, полученное целое число опять делится на два, что дает 5 и 0 в остатке. Это вторая значащая цифра. Деление продолжают, пока получившееся целое число не станет равным единице, она же будет и цифрой старшего двоичного разряда. Данную процедуру удобно записывать в виде следующей цепочки действий:

$$21/2=10 \text{ ост. } 1$$

$$10/2=5 \text{ ост. } 0$$

$$5/2=2 \text{ ост. } 1$$

$$2/2=1 \text{ ост. } 0$$

$$1/2 \text{ ост. } 1$$

Осуществить преобразование двоичного кода в десятичный можно, воспользовавшись формулой (1.3). Например,

$$10011_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 16 + 2 + 1 = 19_{10}.$$

Кроме двоичной и десятичной систем счисления достаточно часто используют *восьмеричную* и *шестнадцатеричную*. В первой для представления разрядов чисел применяют десятичные цифры от 0 до 7, а во второй для ото-

бражения цифр требуется 16 символов. В качестве первых десяти используют цифры 0, 1, 2, ... 9, а остальные шесть обозначаются буквами A, B, C, D, E, F, которые соответствуют числам 10, 11, 12, 13, 14 и 15.

Правила перевода из одной системы счисления в другую аналогичны рассмотренным ранее. Например:

$$123_8 = 1 \cdot 8^2 + 2 \cdot 8^1 + 3 \cdot 8^0 = 64 + 16 + 3 = 83_{10},$$

$$AF5_{16} = 10 \cdot 16^2 + 15 \cdot 16^1 + 5 \cdot 16^0 = 10 \cdot 256 + 15 \cdot 16 + 5 = 2805_{10}.$$

Так как основания восьмеричной и шестнадцатеричной систем счисления являются степенями двойки, то это существенно облегчает перевод чисел из этих систем в двоичную и обратно. Для перевода из восьмеричной системы в двоичную достаточно каждую цифру числа представить трехразрядным двоичным кодом (*триадой*) и полученные комбинации нулей и единиц выписать в ряд.

Например: $123_8 = (001_2) (010_2) (011_2) = 1\ 010\ 011_2$. При необходимости перевода из шестнадцатеричной системы в двоичную требуется проделать аналогичную процедуру, но преобразуя каждую цифру шестнадцатеричного кода в четырехразрядную комбинацию (*тетраду*) двоичного:

$$AF5_{16} = (1010_2) (1111_2) (0101_2) = 1010\ 1111\ 0101_2.$$

Так же просто осуществляется и обратное преобразование, т. е. трансформация из двоичного кода в восьмеричный и шестнадцатеричный. В первом случае требуется двоичное число, начиная справа, разбить на триады, а во втором — на тетрады, и каждую из них заменить цифрой или буквой в соответствующем коде. Например:

$$10111100101_2 = 010\ 111\ 100\ 101 = 2745_8$$

$$10111100010_2 = 0101\ 1110\ 0010 = 5E2_{16}.$$

Такой подход нельзя использовать для преобразования двоичного кода в десятичный и наоборот, т. к. отношение между основаниями этих систем счисления не кратно степени двойки.

Формализовать процедуру трансформации десятичного кода в код, аналогичный двоичному, можно, преобразовывая каждую цифру десятичного числа в тетраду двоичного кода. Тогда трансформация, к примеру, числа 175_{10} будет выглядеть следующим образом: $1 \rightarrow 0001$; $7 \rightarrow 0111$; $5 \rightarrow 0101$, а его запись примет вид $175_{10} = 0001\ 0111\ 0101$.

Однако это выражение будет существенно отличаться от представления этого же числа в двоичном коде, $175_{10} = 10101111_2$. Отличие возникает из-за того, что в предыдущем примере разряды в пределах каждой тетрады имеют двоичные веса, меняющиеся от единицы до восьми, и, кроме того, каждая

из тетрад имеет и свой десятичный вес — 1, 10, 100. При таком подходе образуется новая система счисления, которая в данном случае называется *двоично-десятичной* и относится к позиционным системам счисления с двойным взвешиванием.

В ряде случаев использование такой системы оказывается достаточно удобным из-за упрощения процедуры преобразования. Однако выполнение арифметических операций в двоично-десятичной и аналогичных системах счисления является более сложной процедурой, чем в обычных позиционных. В табл. 1.1 приведены представления чисел от нуля до двадцати в различных системах счисления, в том числе и позиционных.

Как уже отмечалось, в позиционных системах счисления веса разрядов могут быть произвольными. При этом одна и та же последовательность чисел в системах с одинаковым основанием будет представляться по-разному в зависимости от соотношения весов разрядов. В табл. 1.2 приведены различные, используемые на практике, способы кодирования чисел от нуля до девяти.

В первом столбце представлены комбинации, соответствующие коду 8-4-2-1. Название кода состоит из значений весовых коэффициентов соответствующих разрядов. Если веса имеют значения 2-4-2-1, то для ряда десятичных цифр кодовые комбинации будут выглядеть по-иному. Код 7-4-2-1 интересен тем, что любая кодовая комбинация содержит не более двух единиц. Такое свойство кода полезно для выявления ошибок при передаче сигналов, т. к. если в принятой тетраде окажется три единицы, то это будет сигналом о возникновении ошибки вследствие того, что таких комбинаций в данном коде не существует. В позиционных системах счисления с одинаковым основанием, но с различными соотношениями весов разрядов, одна и та же последовательность чисел будет представляться по-разному.

Таблица 1.1. Представление чисел в различных системах счисления

Дес. число	Непозиционная (римская)	Восьмеричная	Шестнадцатеричная	Двоичная	Двоично-десятичная
0		0	0	0000	0000
1	I	1	1	0001	0001
2	II	2	2	0010	0010
3	III	3	3	0011	0011
4	IV	4	4	0100	0100
5	V	5	5	0101	0101

Таблица 1.1 (окончание)

Дес. число	Непозиционная (римская)	Восьмеричная	Шестнадцатеричная	Двоичная	Двоично-десятичная
6	VI	6	6	0110	0110
7	VII	7	7	0111	0111
8	VIII	10	8	1000	1000
9	IX	11	9	1001	0001
10	X	12	A	1010	1 0000
11	XI	13	B	1011	1 0001
12	XII	14	C	1100	1 0010
13	XIII	15	D	1101	1 0011
14	XIV	16	E	1110	1 0100
15	XV	17	F	1111	1 0101
16	XVI	20	11	1 0000	1 0110
17	XVII	21	12	1 0001	1 0111
18	XVIII	22	13	1 0010	1 1000
19	XIX	23	14	1 0011	1 1001

Особенность кодов, представленных в трех последних столбцах таблицы, в том, что они относятся к классу непозиционных или невзвешенных. В коде с избытком 3 сумма двоичных чисел в первой и последней строках, второй и предпоследней и т. д. всегда дает кодовую комбинацию 1111. Сформировать этот код можно путем прибавления к соответствующей десятичной цифре тройки и преобразования полученного результата в двоичный код.

Таблица 1.2. Кодированное представление чисел в системах счисления с основанием два

Дес. цифра	Код 8-4-2-1	Код 2-4-2-1	Код 7-4-2-1	Код с избытком 3	Код 2 из 5	Код Грея
0	0000	0000	0000	0011	11000	0000
1	0001	0001	0001	0100	01100	0001
2	0010	0010	0010	0101	00110	0011
3	0011	0011	0011	0110	00011	0010

Таблица 1.2 (окончание)

Дес. цифра	Код 8-4-2-1	Код 2-4-2-1	Код 7-4-2-1	Код с избытком 3	Код 2 из 5	Код Грея
4	0100	0100	0100	0111	10001	0110
5	0101	1011	0101	1000	10100	0111
6	0110	0110	0110	1001	01010	0101
7	0111	0111	1000	1010	00101	0100
8	1000	1110	1001	1011	10010	1100
9	1001	1111	1010	1100	01001	1101

В коде 2 из 5 используются пятиразрядные комбинации нулей и единиц. Его особенность в том, что в любой кодовой комбинации содержится по две единицы. Еще одна разновидность довольно часто применяемого кода — код Грея, который обладает тем свойством, что представления соседних чисел отличаются состоянием только одного из разрядов.

Все ранее рассмотренные способы кодирования основывались на предположении, что исходное число положительно. Однако для удобства выполнения ряда арифметических операций в любой системе счисления можно ввести понятие отрицательных чисел, значения которых будут меньше нуля. В десятичной системе счисления для их записи используется знак "-", а такое же по величине (по модулю) положительное число дополняется знаком "+".

Так как в цифровых вычислительных машинах используются двухуровневые сигналы, одному из которых приписывается значение логического нуля, а другому единицы, то ввести аналогичные знаки не представляется возможным, т. к. любое число и символ должны быть выражены через комбинации нулей и единиц. То есть и знак числа требуется отображать этими же символами.

При одном из подходов положительному знаку ставится в соответствие 0, а отрицательному 1 и знаковый разряд записывается левее самого старшего значащего. Таким образом, число +9 будет выглядеть как 0 1001, а -9 следующим образом 1 1001. Однако в этом случае потребуются обязательное указание на то, что старший разряд является знаковым, иначе первое число будет воспринято как 9, а второе как 25. В так называемом модифицированном коде под знак отводят два дополнительных разряда, положительный кодируется 00, а отрицательный 11.

Некоторые виды арифметических операций в цифровых вычислительных машинах удобнее производить, используя обратные и дополнительные

двоичные коды чисел. Формально *обратный код* получается из двоичного (*прямого*) путем замены во всех разрядах нулей на единицы, а единиц на нули. Таким образом, если прямой двоичный код числа 9 выглядит как 1001, то обратный будет представлен комбинацией 0110.

Дополнительный код образуется путем арифметического прибавления единицы в младший разряд обратного кода числа. Отсюда следует, что дополнительный код 9 отобразится кодовой комбинацией 0111. Понятие дополнительного кода можно ввести для любой позиционной системы счисления. Он представляет собой число, дополняющее исходное до значения веса следующего по старшинству разряда системы счисления.

В рассмотренном примере для представления числа 9 используется четырехразрядная двоичная кодовая комбинация. Вес следующего, более старшего разряда в двоичной системе будет $2^4 = 16$. То есть дополнительным кодом девятки будет число $16 - 9 = 7$, двоичный код которого имеет вид 0111. Если речь вести о десятичной системе счисления, то дополнительный код 9 должен определяться как $10^2 - 9 = 91$.

1.2. Логические переменные и функции

При математическом описании различных процессов вводится понятие *переменной*. Это некоторая независимая величина, которая принимает ряд значений в определенном диапазоне. Множество значений переменной может быть как непрерывным, так и дискретным. В первом случае переменная принимает любое значение из области, в которой она определена, а во втором лишь ряд конкретных. Примером переменной первого вида является температура. Она меняется непрерывно и принимает любое значение из соответствующего диапазона, причем соседние могут отличаться на бесконечно малую величину. Примером дискретной переменной может служить цена товара. Ее минимальные изменения кратны одной копейке, т. к. меньших денежных единиц нет.

Над переменными можно проводить определенные математические действия. Совокупность этих действий и правил их выполнения называется *алгеброй* соответствующих *переменных*. Значениям одной переменной могут быть поставлены в соответствие значения другой. Закон, определяющий это соответствие, называется *функцией*.

В особую группу выделяются переменные, принимающие лишь два фиксированных значения. Например, если переменная описывает состояния переключателя, который может находиться либо во включенном, либо в выключенном состояниях. Значению переменной для одного из них можно присвоить

название "Вкл", а для другого "Выкл", либо обозначить их по иному "А" и "В", или 0 и 1, учитывая в последнем случае, что это не цифры, а просто символы для описания состояния переменной.

Переменные, имеющие лишь два значения, часто называются *логическими* или *Булевыми*. Первое связано с тем, что они могут выступать как результат анализа логического рассуждения, который бывает истиной или ложью. Совокупность законов преобразования этих переменных и правил действий над ними называется *Булевой алгеброй* или *алгеброй логики*.

Таблица 1.3. Операция логического умножения двух переменных

x_1	x_2	$x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1

В обычной алгебре для двух переменных A и B существует три возможных отношения между их значениями: A может быть равно, больше или меньше B . В алгебре логики определено лишь отношение эквивалентности, т. е. переменные здесь могут быть либо равны, либо не равны. Вопрос, какая из них больше, а какая меньше, не имеет смысла. Кроме того, для таких переменных определены три основных операции или действия: конъюнкция, дизъюнкция и инверсия.

Конъюнкция, иначе называется операцией *логического умножения*, или операцией "И". Она обозначается значком " \wedge " либо точкой "-", которой в обычной алгебре соответствует умножение. Иногда эту точку не ставят.

Таблица 1.4. Операция логического сложения двух переменных

x_1	x_2	$x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

Правило выполнения операции конъюнкции для двух логических переменных x_1 и x_2 может быть описано словесно, а также представлено в виде таб-

лицы (табл. 1.3). Результат логического умножения равен единице, только если обе переменные одновременно принимают единичные значения. Данная таблица полностью совпадает с таблицей умножения чисел 0 и 1.

Операция *дизъюнкции*, называемая иначе операцией *логического сложения* либо операцией *"ИЛИ"*, обозначается как значком " \vee ", так и "+". Данная операция над двумя переменными определяется табл. 1.4.

Инверсия, называемая также операцией *логического отрицания* либо операцией *"НЕ"*, реализуется над одной переменной, преобразуя ее значение в противоположное. Эта операция обозначается в виде черточки сверху над соответствующей переменной, т. е. $\bar{1}=0$, $\bar{0}=1$.

В отличие от алгебры обычных переменных в алгебре логики не существует операций умножения, деления, возведения в степень. Это связано с тем, что логические переменные не принимают числовых значений, не могут быть отрицательными, дробными и т. п.

Таблица 1.5. Основные соотношения алгебры логики для одной переменной

$x + 0 = x$	$x \cdot 0 = 0$
$x + 1 = 1$	$x \cdot 1 = x$
$x + x = x$	$x \cdot x = x$
$x + \bar{x} = 1$	$x \cdot \bar{x} = 0$
$\bar{\bar{0}} = 1, \bar{\bar{1}} = 0$	$\bar{\bar{x}} = x$

В Булевой алгебре справедливы соотношения, приведенные в табл. 1.5, которые можно проверить простым перебором значений переменной x , воспользовавшись правилами выполнения соответствующих операций.

Для алгебры логики, как и для обычной алгебры, определен ряд законов выполнения действий над переменными, в частности, коммутативный, ассоциативный и дистрибутивный.

Первый, иначе называемый *переместительным законом*, записывается следующим образом: $x_1 + x_2 = x_2 + x_1$ и $x_1 \cdot x_2 = x_2 \cdot x_1$, из которого вытекает, что при сложении и умножении логических переменных их можно менять местами.

Второй, *ассоциативный закон*, иначе называется *сочетательным*. Для трех переменных его можно представить как:

$$x_1 + x_2 + x_3 = x_1 + x_2 + x_3 = x_1 + x_2 + x_3 ,$$

$$x_1 \cdot x_2 \cdot x_3 = x_1 \cdot x_2 \cdot x_3 = x_1 \cdot x_2 \cdot x_3 ,$$

т. е. при выполнении логических операций, переменные можно объединять в группы и выполнять соответствующие действия по очереди.

Дистрибутивный, или *распределительный* закон устанавливает правила выполнения скобочных действий $x_1 \cdot x_2 \cdot x_3 = x_1 \cdot x_2 + x_1 \cdot x_3$, или $x_1 \cdot x_2 + x_1 \cdot x_3 = x_1 \cdot x_2 + x_3$. Данные выражения представляют собой тождества, т. е. они справедливы при любых значениях переменных.

К основным законам алгебры логики относятся и законы или правила *де Моргана*, которые связывают операции логического сложения и умножения. Если в обычной алгебре умножение можно представить как многократное сложение, то логическое сложение может быть выражено через логическое умножение следующим образом:

$$\overline{x_1 + x_2 + \dots + x_n} = \overline{x_1} \cdot \overline{x_2} \cdot \dots \cdot \overline{x_n} , \quad (1.4)$$

а умножение через сложение, как:

$$\overline{x_1 \cdot x_2 \cdot \dots \cdot x_n} = \overline{x_1} + \overline{x_2} + \dots + \overline{x_n} , \quad (1.5)$$

т. е. инверсия суммы логических переменных равна логическому произведению их инверсий, а инверсия произведения — сумме инверсий.

Если к обеим частям равенства применить одну и ту же процедуру, то оно не изменится. Отсюда следует, что, проинвертировав обе части приведенных соотношений, правила де Моргана можно представить в такой форме:

$$x_1 + x_2 + \dots + x_n = \overline{\overline{x_1} \cdot \overline{x_2} \cdot \dots \cdot \overline{x_n}} \quad (1.6)$$

$$x_1 \cdot x_2 \cdot \dots \cdot x_n = \overline{\overline{x_1} + \overline{x_2} + \dots + \overline{x_n}} .$$

Как и в алгебре непрерывных переменных, в алгебре логики под *функцией* понимается некий закон, или правило, по которому переменным из одного набора (множества) ставятся в соответствие переменные из другого набора (множества). В обычной алгебре аргумент и функция могут принимать целые и дробные, положительные, отрицательные значения, и количество функций от одного аргумента не ограничено. Например, $y = x$, $y = x^2$, $y = x^3$, $y = \sin x$, $y = \log x$ и т. п.

В алгебре логики из-за того, что как у переменной, так и у функции может быть только два значения, число последних конечно.

От одной логической переменной существует лишь четыре различных функции, приведенные в табл. 1.6. Они задаются следующим образом. Каждому из значений переменной может быть произвольным образом поставлено в соответствие значение функции равное 0, либо 1. Если любому значению аргумента функция f_0 ставит в соответствие 0, то она называется *тождественный ноль*.

Таблица 1.6. Функции одной логической переменной

x	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

Функция f_1 называется *функцией тождества*, т. к. ее значения совпадают со значениями аргумента. Значения функции f_2 противоположны, или инверсны, по отношению к значениям аргумента. Последняя функция f_3 обоим значениям аргумента x ставит в соответствие единицы и называется *тождественная единица*. Других видов функций от одной переменной нет.

Таблица 1.7. Функции двух логических переменных

x_0	x_1	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Как и в обычной алгебре, в алгебре логики существуют функции от нескольких аргументов или переменных, причем количество функций N связано с числом переменных n соотношением $N = 2^{2^n}$. Если переменных две — x_0 и x_1 , то их наборов будет четыре, а количество вариантов задания значений функций на этих наборах и, соответственно, число самих несовпадающих функций — 16. Они представлены в табл. 1.7.

Некоторые из функций, приведенных в таблице, имеют собственные названия. Функция f_1 называется функцией логического умножения, конъюнкцией, функцией И, а f_7 — функцией логического сложения, дизъюнкцией, либо функцией ИЛИ. Это объясняется тем, что значения данных функций эквивалентны результатам выполнения соответствующих логических операций.

С учетом количества обрабатываемых переменных f_1 часто называют функцией 2И, а f_7 — 2ИЛИ. Алгебраическая (символьная) запись этих функций выглядит следующим образом: $f_1 = x_1 \cdot x_2$ и $f_7 = x_1 + x_2$.

Таблица 1.8. Связь между логическими функциями

x_0	x_1	f_1	f_7	f_8	f_{14}	f_6	f_9
0	0	0	0	1	1	0	1
0	1	0	1	0	1	1	0
1	0	0	1	0	1	1	0
1	1	1	1	0	0	0	1

Как показано в табл. 1.8, функция f_8 отличается от f_7 тем, что нули заменены единицами и наоборот. То есть каждое значение f_7 проинвертировано. Поэтому функция f_8 называется функцией ИЛИ-НЕ (2ИЛИ-НЕ) и ее связь с f_7 можно отобразить таким образом $f_8 = \overline{f_7}$.

Аналогичная связь наблюдается и между функциями f_{14} и f_1 , вследствие чего f_{14} носит название функции И-НЕ (2И-НЕ) и соответственно $f_{14} = \overline{f_1}$. Отсюда следует, что количество логических функций не только ограничено, но они еще и определенным образом взаимосвязаны между собой.

Функция f_6 называется функцией логической неравнозначности, а f_9 — функцией логической равнозначности. Первая из них принимает единичное значение в случаях, когда аргументы x_0 и x_1 не равны, а вторая в противоположной ситуации.

В алгебре логики функции можно определить или задать как с помощью таблицы, отражающей связь значений аргументов и функции, так и в виде совокупности типовых логических операций, записанных в виде формулы. Первый способ называется *табличным*, а второй *аналитическим*.

Вследствие наличия определенной связи между операциями конъюнкции и дизъюнкции, которая описывается законами де Моргана, одна и та же функция аналитически может быть представлена по-разному. Например, $f(x_0, x_1) = x_0 \cdot x_1$ можно представить и как $\overline{\overline{x_0 + x_1}}$. Функция $f(x_0, x_1, x_2) = x_0 + x_1 \cdot x_2$ после тождественных преобразований может выглядеть следующим образом: $\overline{\overline{x_0 + (x_1 + x_2)}}$, или $\overline{\overline{x_0 \cdot (x_2 + x_3)}}$, либо как $\overline{\overline{x_0 \cdot x_2 + x_0 \cdot x_3}}$. Возможен также вариант $\overline{\overline{x_0 + x_1 + x_2 + x_3}}$.

Используя правила де Моргана, любую логическую функцию можно представить в двух разных, но эквивалентных формах: как сумму произведений логических переменных или как произведение сумм. Если логическое выражение функции представляет собой сумму компонент, каждая из которых является простой конъюнкцией аргументов, то такая форма называется *дизъюнктивной нормальной формой* — ДНФ. Когда в выражение, описывающее функцию, входят лишь произведения сумм прямых или инверсных значений аргументов, это соответствует второй, так называемой *конъюнктивной нормальной форме* или КНФ.

Примером ДНФ является запись $f_1(x_0, x_1, x_2) = x_0 + x_1 \cdot \overline{x_2} + x_0 \cdot \overline{x_1} \cdot x_2 + \overline{x_0} \cdot x_2$, а КНФ может выглядеть следующим образом:

$$f_2(x_0, x_1, x_2) = (x_0 + x_1) \cdot x_2 \cdot (\overline{x_0 + x_1 + x_2}) \cdot (\overline{x_0 + x_2}).$$

Некоторые выражения не подпадают под эти определения, например $f_3(x_0, x_1, x_2) = x_0 + x_1 \cdot \overline{x_2} + x_0 \cdot \overline{x_1} \cdot x_2 + x_1 \cdot x_2$, т. к. здесь последнее слагаемое не является простой конъюнкцией, т. е. произведением соответствующих логических переменных. Однако после небольших преобразований его можно перевести в ДНФ такого вида $f_3(x_0, x_1, x_2) = x_0 + x_1 \cdot \overline{x_2} + x_0 \cdot \overline{x_1} \cdot x_2 + x_1 + x_2$.

ДНФ и КНФ — это две эквивалентные формы представления логических функций, которые, используя правила и законы алгебры логики, можно трансформировать одна в другую. Для функции f_1 процедура преобразования будет следующей:

$$\begin{aligned} f_1(x_0, x_1, x_2) &= x_0 + x_1 \cdot \overline{x_2} + x_0 \cdot \overline{x_1} \cdot x_2 + \overline{x_0} \cdot x_1 = \\ &= \overline{\overline{x_0 \cdot x_1 \cdot x_2}} \cdot \overline{\overline{x_0 \cdot x_1 \cdot x_2}} \cdot \overline{\overline{x_0 \cdot x_2}} = \\ &= \overline{\overline{x_0 \cdot x_1 + x_2}} \cdot \overline{\overline{x_0 + x_1 + x_2}} \cdot \overline{\overline{x_0 + x_2}}. \end{aligned} \quad (1.7)$$

Полученное выражение по определению не является КНФ. Однако если проинвертировать обе части равенства, то КНФ получится для функции $\overline{f_1}$.

После замены в КНФ логического умножения на сложение, запись f_2 примет вид $f_2(x_0, x_1, x_2) = \overline{\overline{(x_0 + x_1)} + x_2} + \overline{\overline{(x_0 + x_1 + x_2)} + (x_0 + x_2)}$, при котором функция оказывается представленной с использованием лишь двух операций — логического сложения (*ИЛИ*) и инверсии (*НЕ*). При замене сложения на умножение, получится соотношение, в которое войдут лишь операции логического умножения (*И*) и инверсии (*НЕ*).

Отсюда следует, что любая, сколь угодно сложная логическая функция представима с помощью двух простейших — *ИЛИ* и *НЕ* либо *И* и *НЕ*. Наборы функций, через которые можно выразить все остальные, называются *базисом*.

Следуя правилам алгебры логики, функцию как *НЕ*, так и *ИЛИ* можно представить, используя лишь одну операцию *ИЛИ-НЕ*. Действительно, $\overline{\overline{x + x}} = x + x$, $\overline{\overline{x_1 + x_2}} = x_1 + x_2$.

Таким образом, набор из двух функций *ИЛИ* и *НЕ* является избыточным, т. к. после соответствующих преобразований любую функцию можно реализовать, используя лишь функцию *ИЛИ-НЕ*. Поэтому она является представительницей минимального базиса.

Аналогичные рассуждения можно провести и по поводу функции *И-НЕ*. Действительно, $\overline{\overline{x \cdot x}} = x \cdot x$, $\overline{\overline{x_1 \cdot x_2}} = x_1 \cdot x_2$, а следовательно, и эта функция также может служить в качестве минимального базиса. Отсюда следует, что любую сколь угодно сложную функцию от произвольного количества логических переменных можно представить, используя только одну, причем любую из рассмотренных функций. Это обстоятельство в ряде случаев существенно облегчает построение устройств для обработки цифровых сигналов.

Кроме представления функций в форме ДНФ и КНФ существуют так называемые совершенная дизъюнктивная нормальная форма (СДНФ) и совершенная конъюнктивная нормальная форма (СКНФ). ДНФ функции называется совершенной, если в каждом ее слагаемом присутствуют все аргументы или их инверсии.

Функция $f_1(x_0, x_1) = x_0 + \overline{x_0} \cdot x_1$ не представлена в СДНФ, т. к. в первое слагаемое не входит переменная x_1 . А функция $f_2(x_0, x_1) = x_0 \cdot x_1 + \overline{x_0} \cdot \overline{x_1}$ записана в совершенной дизъюнктивной нормальной форме. Аналогичная ситуация справедлива и для конъюнктивных нормальных форм.

Любая функция, представленная в несовершенной форме, всегда может быть приведена к совершенной, причем единственным образом. В частности для функции $f_1(x_0, x_1)$ это делается умножением первого слагаемого на выраже-