Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра радиотехнических систем

В. Н. Левкович, А. В. Мартинович

ЦИФРОВЫЕ И МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА

Методическое пособие к практическим занятиям
для студентов радиотехнических специальностей
всех форм обучения

Минск БГУИР 2009

УДК 004.3(076)
ББК 32.973.26-04я7
Л37

Р е ц е н з е н т :
доцент кафедры сетей и устройств телекоммуникаций БГУИР,
канд. техн. наук И. И. Астровский

Левкович, В. Н.
Л37 Цифровые и микропроцессорные устройства : метод. пособие к практ. занятиям для студ. радиотех. спец. всех форм обуч. / В. Н. Левкович, А. В. Мартинович. – Минск : БГУИР, 2009. – 36 с. : ил.
ISBN 978-985-488-411-0

В пособии представлены 8 тем, каждая из которых содержит краткие теоретические сведения по арифметическим и логическим основам цифровой техники, примеры решения типовых задач, а также задания для самостоятельного выполнения.

Рекомендуется для подготовки к практическим занятиям.

УДК 004.3(076)
ББК 32.973.26-04я7

ISBN 978-985-488-411-0

Тема 1. Представление и преобразование чисел в системах счисления, применяемых в цифровой технике

1.1. Общие сведения

Система счисления – совокупность символов и правил для обозначения чисел. Все системы счисления разделяются на позиционные и непозиционные. Наиболее древними системами счисления являются непозиционные, например, римская система счисления. В настоящее время они почти полностью заменены позиционными.

В позиционных системах числа представляются в виде последовательности цифр $X_k = X_1 X_2 \ldots X_i \ldots X_n$, в которой значение каждой цифры X_i зависит от места ее расположения в последовательности.

Любое число в позиционной системе счисления с постоянным основанием можно представить в виде следующего полинома:

$$C = \pm \sum_{k=1}^{n} C_k q^{p-k}, \tag{1.1}$$

где q – основание системы счисления; X_k – цифра k-го разряда числа; n – количество разрядов в числе; p – порядок числа (целое число, показывающее место запятой в числе).

Например, для числа 128, у которого $q = 10$ и $p = 3$, получаем

$$128 = 1 \cdot 10^2 + 2 \cdot 10^1 + 8 \cdot 10^0 = 100 + 20 + 8.$$

1.2. Двоичная система счисления

В этой системе счисления для представления любого разряда двоичного числа достаточно иметь один физический элемент только с двумя явно различимыми состояниями, одно из которых изображается цифрой 1, а другое – цифрой 0.

Любое двоичное число может быть представлено в следующей форме:

$$(A)_2 = \pm(a_{m-1} 2^{m-1} + a_{m-2} 2^{m-2} + \ldots + a_0 + a_{-1} 2^{-1} + \ldots + a_{-k} 2^{-k}). \tag{1.2}$$

Здесь $p = 2$, коэффициенты a могут принимать значение 0 или 1.

Запишем в двоичной системе счисления 4-разрядное число согласно выражению (1.2):

$$(B)_2 = b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0,$$

где $p = 2$ – основание двоичной системы счисления, b – коэффициенты, принимающие значения 0, 1. Для данного случая $B_2 = 8 + 4 + 2 + 1$ при $b_3 = b_2 = b_1 = b_0 = 1$, тогда $B_2 = 1111_2 = 15_{10}$.

Достоинства двоичной системы счисления:

– простота выполнения арифметических и логических операций и, как следствие, простота устройств, реализующих эти операции;

– возможность использования аппарата алгебры логики для анализа и синтеза операционных устройств.

Недостаток двоичной системы счисления – громоздкость по сравнению с десятичной для использования человеком.

1.3. Восьмеричная система счисления

Основанием восьмеричной системы счисления является число 8. Для записи чисел от 0 до 7 используются соответствующие цифры *0, 1, 2, 3, 4, 5, 6, 7*. Для изображения чисел, больших 7, применяется позиционный способ записи, при котором значение цифры зависит от её положения в числе.

Восьмеричные числа можно представить в виде суммы значений (от 0 до 7) отдельных коэффициентов, умноженных на соответствующие степени основания 8.

$$(C)_8 = \ldots + k_3 \cdot 8^3 + k_2 \cdot 8^2 + k_1 \cdot 8^1 + k_0 \cdot 8^0,$$

где k – коэффициенты системы счисления.

1.4. Шестнадцатеричная система счисления

Основанием системы является число 16. Для записи чисел от 0 до 15 используются символы *0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F*. Для представления чисел, больших 15, применяется позиционный способ записи.

Шестнадцатеричные числа можно записать в виде суммы значений (от 0 до 16) отдельных коэффициентов, умноженных на соответствующие степени числа 16.

$$(D)_{16} = \ldots + k_3 \cdot 16^3 + k_2 \cdot 16^2 + k_1 \cdot 16^1 + k_0 \cdot 16^0,$$

где k – коэффициенты системы счисления.

Системы счисления с основаниями 8 и 16 применяются для краткой записи многоразрядных двоичных чисел в цифровых системах.

1.5. Двоично-кодированная десятичная система счисления

Представляя каждую десятичную цифру совокупностью из четырех разрядов (*тетрад*), можно получить комбинированную систему счисления, которая обладает достоинствами двоичной системы и удобством десятичной. В вычислительной технике наибольшее применение нашли системы кодирования с весами разрядов в пределах тетрады *8421, 2421 и 8421 + 3*.

Код *8421* называется кодом с *естественными весами*, здесь цифры *8, 4, 2, 1* – веса двоичных разрядов тетрад. Любая десятичная цифра в этом коде изображается ее эквивалентом в двоичной системе счисления. Этот код нашел наибольшее применение при кодировании десятичных чисел в устройствах ввода–вывода.

Особенность кодов *2421* и *8421 + 3* состоит в том, что кодирование любой десятичной цифры и дополнительной к ней цифры до 9 осуществляется взаимно дополняющимися тетрадами. Эта особенность дает простой способ получения дополнения до 9 путем инвертирования двоичных цифр тетрады.

В табл. 1.1 представлены десятичные числа 0…20 и их двоичные, восьмеричные, шестнадцатеричные и двоично-десятичные эквиваленты.

Таблица 1.1

Десятичная	Двоичная	Восьмеричная	Шестнадцатеричная	Двоично-десятичная с естественными весами
0	00	0	0	0000
1	01	1	1	0001
2	10	2	2	0010
3	11	3	3	0011
4	100	4	4	0100
5	101	5	5	0101
6	110	6	6	0110
7	111	7	7	0111
8	1000	10	8	1000
9	1001	11	9	1001
10	1010	12	A	01 0000
11	1011	13	B	01 0001
12	1100	14	C	01 0010
13	1101	15	D	01 0011
14	1110	16	E	01 0100
15	1111	17	F	01 0101
16	10000	20	10	01 0110
17	10001	21	11	01 0111
18	10010	22	12	01 1000
19	10011	23	13	01 1001
20	10100	24	14	10 0000

1.6. Перевод чисел из одной системы счисления в другую

Правило перевода целых чисел

Для перевода целого числа N_p из системы счисления с основанием p в систему счисления с основанием q необходимо данное число разделить на основание q (по правилам в системе счисления с основанием p) до получения целого остатка, меньшего q. Полученное частное снова необходимо разделить на основание q до получения целого остатка, меньшего q, и так до тех пор, пока последнее частное не станет меньше q. Число N_q в системе счисления с основанием q представляется в виде упорядоченной последовательности остатков от деления в порядке, обратном их получению, а старший разряд числа N_q дает последнее частное.

Пример. Перевести число 132 из десятичной системы счисления в двоичную.

Решение. Процесс перевода числа 132 из десятичной системы в двоичную показан на рис. 1.1. Десятичному числу $N_{10} = 132$ соответствует двоичное $N_2 = 10000100$.

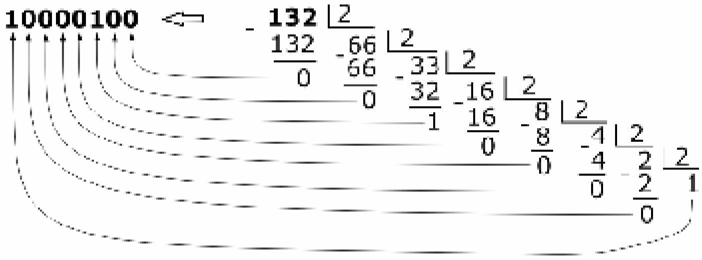


Рис. 1.1. Процесс перевода числа 132 из десятичной системы в двоичную

<u>*Правило перевода правильной дроби*</u>

Перевод правильной дроби N_p из системы счисления с основанием p в систему счисления с основанием q заключается в последовательном умножении этой дроби на основание q (по правилам системы счисления с основанием p), причем перемножению подвергаются только дробные части. Дробь N_q в системе счисления с основанием q представляется в виде упорядоченной последовательности целых частей произведений в порядке их получения, где старший разряд искомого числа представлен первой цифрой произведения. Если требуемая точность перевода составляет q^{-k}, то число указанных последовательных произведений равно k.

Пример. Перевести десятичную дробь 0,7689 в двоичную с точностью 2^{-6}. Десятичной дроби N_{10} = 0,7689 соответствует двоичная дробь N_2 = 0,110001 с точностью 2^{-6}, полученная шестикратным умножением дробных частей на основание, равное 2. Процесс перевода правильной дроби 0,7689 из десятичной системы в двоичную показан на рис. 1.2.

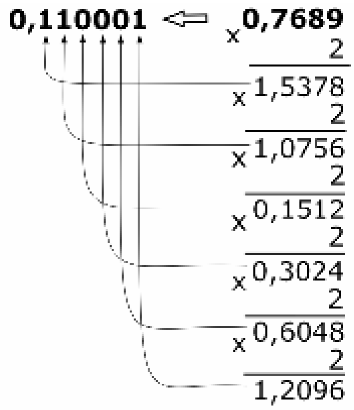


Рис. 1.2. Процесс перевода правильной дроби 0,7689 из десятичной системы в двоичную

Перевод неправильной дроби

Для чисел, имеющих как целую, так и дробную часть, перевод из одной системы счисления в другую осуществляется отдельно для целой и дробной части по правилам, указанным выше. Пример перевода неправильной десятичной дроби в двоичную систему счисления приведен на рис. 1.3.

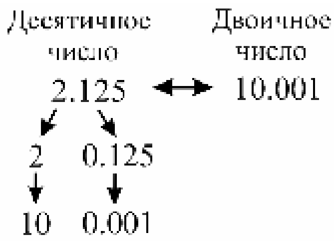


Рис. 1.3. Пример перевода неправильной десятичной дроби в двоичную систему счисления

1.7. Перевод чисел из двоичной системы в восьмеричную и шестнадцатеричную и обратно

Для перевода двоичного числа в восьмеричную систему счисления необходимо исходное число, начиная с правого, младшего разряда, разбить на триады. Недостающие разряды слева следует заменить нулями, а затем каждую триаду записать в виде эквивалентного восьмеричного числа. При переводе обратно каждое восьмеричное число переводится обратно в триады. Для шестнадцатеричной системы производятся аналогичные действия, только вместо триад деление ведется по тетрадам.

Для перехода из восьмеричной или шестнадцатеричной систем в двоичную производят обратные действия, т.е. каждую восьмеричную или шестнадцатеричную цифру заменяют эквивалентной двоичной триадой или тетрадой.

Пример перевода числа из двоичной системы в восьмеричную показан на рис. 1.4.

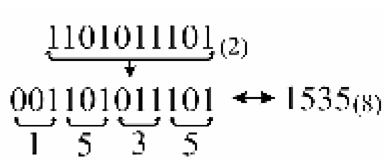


Рис. 1.4. Пример перевода числа из двоичной системы в восьмеричную

1.8. Перевод из двоичной системы счисления в десятичную

На практике для этих целей используют формулу полиномиального представления числа в позиционной системе счисления.

Для двоичного числа данная формула имеет вид

$$(A)_2 = \pm(a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \ldots + a_0 + a_{-1}2^{-1} + \ldots + a_{-k}2^{-k}).$$

Здесь основание системы счисления равно 2, а коэффициент a может принимать значение *0* или *1*.

Пример. Запишем в двоичной системе счисления 4-разрядное число B в соответствии с записанным выше выражением:

$$(B)_2 = b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0,$$

где 2 – основание двоичной системы счисления, b – коэффициенты, принимающие значения 0 или 1.

При $b_3 = b_2 = b_1 = b_0 = 1$, имеем $B_2 = 1111_2 = 15_{10}$.

1.9. Задания для самостоятельного выполнения

1. Выполнить преобразования из одной системы счисления в другую:

а) $1101011_2 = X_{16}$; б) $174003_8 = X_2$;

в) $1011111_2 = X_{16}$; г) $67{,}24_8 = X_2$;

д) $10100{,}1101_2 = X_{16}$; е) $F3F5_{16} = X_2$;

ж) $11011001_2 = X_8$; з) $AB3D_{16} = X_2$;

и) $101111{,}0111_2 = X_8$; к) $15C{,}38_{16} = X_2$.

2. Преобразовать следующие восьмеричные числа в двоичные и шестнадцатеричные:

а) $1023_8 = X_2 = X_{16}$; б) $761302_8 = X_2 = X_{16}$;

в) $163417_8 = X_2 = X_{16}$; г) $552273_8 = X_2 = X_{16}$;

д) $5436{,}15_8 = X_2 = X_{16}$; е) $13705{,}207_8 = X_2 = X_{16}$.

3. Преобразовать следующие шестнадцатеричные числа в двоичные и восьмеричные:

а) $1023_{16} = X_2 = X_8$; б) $7E6A_{16} = X_2 = X_8$;

в) $ABCD_{16} = X_2 = X_8$; г) $C350_{16} = X_2 = X_8$;

д) $9E36{,}7A_{16} = X_2 = X_8$; е) $DEAD{,}BEEF_{16} = X_2 = X_8$.

4. Преобразовать следующие числа в десятичные:

а) $1101011_2 = X_{10}$; б) $174003_8 = X_{10}$;

в) $10110111_2 = X_{10}$; г) $67{,}24_8 = X_{10}$;

д) $10100{,}1101_2 = X_{10}$; е) $F3A5_{16} = X_{10}$;

ж) $12010_3 = X_{10}$; з) $AB3D_{16} = X_{10}$;

и) $7156_8 = X_{10}$; к) $15C{,}38_{16} = X_{10}$.

5. Выполнить преобразования из одной системы счисления в другую:

а) $125_{10} = X_2$; б) $3489_{10} = X_8$;

в) $209_{10} = X_2$; г) $9714_{10} = X_2$;

д) $132_{10} = X_2$; е) $23851_{10} = X_{16}$;

ж) $727_{10} = X_8$; з) $57190_{10} = X_{16}$;

и) $1435_{10} = X_8$; к) $65113_{10} = X_{16}$.

Тема 2. Представление и кодирование чисел в цифровых устройствах

В цифровых устройствах оперируют целыми и действительными числами. В свою очередь числа могут быть представлены в формах с фиксированной и плавающей запятой.

Знаки действительных чисел кодируются цифрами: «плюс» − нулем (0), «минус» − единицей (1). При этом код знака записывается перед старшим разрядом числа.

2.1. Представление чисел в форме с фиксированной запятой

В случаях фиксированной запятой (ФЗ) для всех чисел, над которыми выполняются операции в машине, положение запятой строго фиксировано, т.е. порядок числа p в машине постоянен. Выбор величины порядка p при использовании формы с ФЗ может быть в принципе произволен, но при этом необходимо учитывать следующее.

1. Положение запятой закрепляется в определенном месте относительно разрядов числа и сохраняется неизменным для всех чисел, изображаемых в принятой разрядной сетке машины.

2. Хотя запятую и фиксируют, но это никак не выделяется в коде числа, а только подразумевается.

Например, число –10,01 может быть размещено в 8-разрядной ячейке памяти разными способами. Два из возможных вариантов показаны на рис. 2.1.

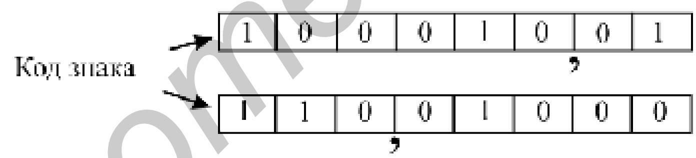


Рис. 2.1. Варианты размещения числа с фиксированной запятой

На практике применяются два способа расположения запятой:

1) перед старшим разрядом (порядок числа $p = 0$);

2) после младшего разряда (порядок числа $p = n$, где n – число разрядов, отведенных для представления числа).

Разрядная сетка для размещения чисел по первому варианту расположения запятой при $p = 0$ показана на рис. 2.2, а на рис. 2.3 – по второму варианту, когда $p = n$.

В микропроцессорных системах длина чисел в форме с ФЗ принимается равной длине машинного слова, или нескольких машинных слов, или части машинного слова.

Важнейшим достоинством представления чисел в форме с ФЗ является возможность построения несложных арифметико-логических устройств (АЛУ) с высоким быстродействием.

Недостатки представления чисел в форме с ФЗ состоят в ограниченном диапазоне представимых чисел и ограниченной точности.

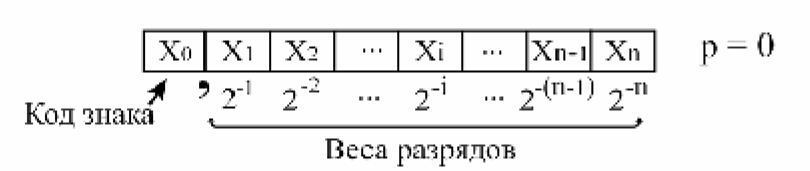


Рис. 2.2. Разрядная сетка для размещения чисел в форме ФЗ при $p = 0$

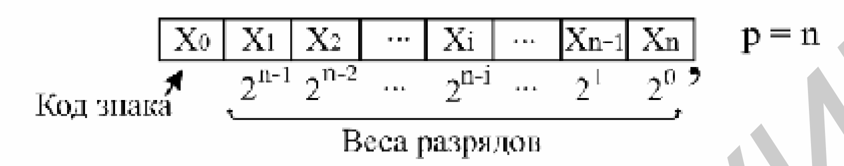


Рис. 2.3. Разрядная сетка для размещения чисел в форме ФЗ при $p = n$

2.2. Представление чисел в форме с плавающей запятой

Представление чисел в форме с плавающей запятой (ПЗ) позволяет:

1) избежать масштабирования исходных чисел;
2) снять необходимость слежения за местоположением запятой;
3) расширить диапазон и точность представляемых чисел.

Число в форме с ПЗ представляется в следующем виде:

$$X = \pm M_x \cdot q^p .$$

Здесь X – само число, $\pm$ – знак числа, M_x – мантисса числа, q – основание системы счисления, p – порядок числа.

В микропроцессорах применяется *нормализованная* форма представления мантиссы, при этом на нее накладываются два ограничения (условия):

1) мантисса по модулю должна быть меньше 1;
2) первая цифра после запятой должна быть отлична от нуля.

В общем случае мантисса в нормализованной форме имеет вид $M_x = X_0 \cdot 1 \cdot X_2 \cdot X_3 \cdot X_4 \cdot \ldots \cdot X_n$. Здесь X_0 – код знака числа.

Разрядная сетка для хранения чисел в форме с плавающей запятой показана на рис. 2.4.

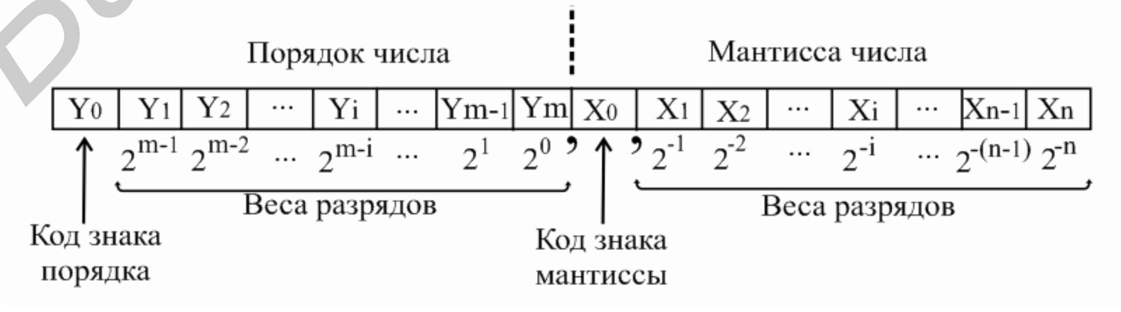


Рис. 2.4. Разрядная сетка для хранения чисел в форме с плавающей запятой

Пример. Разместить в 14-разрядной ячейке памяти число -101,01 в форме с плавающей запятой.

Решение. Преобразуем запись числа из формы с фиксированной запятой в форму с плавающей запятой:

$$-101{,}01 \longrightarrow -0{,}10101 \cdot 2^{+3} \longrightarrow -0{,}10101 \cdot 2^{+011}$$

Выделяем для размещения порядка 6 разрядов, а для размещения мантиссы – 8 разрядов. Записываем в выделенные разряды двоичные коды порядка и мантиссы (рис. 2.5).

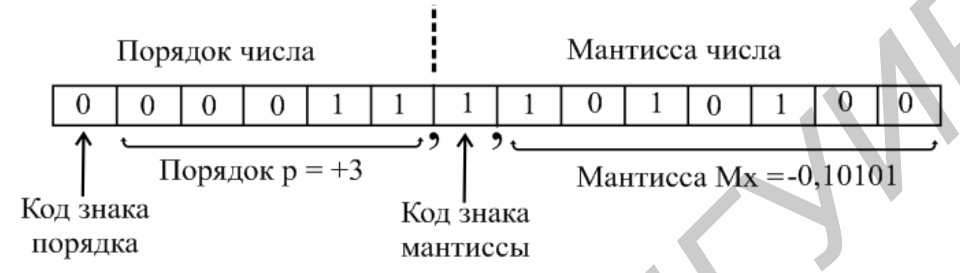


Рис. 2.5. Размещение числа 101,01 в форме с плавающей запятой в 14-разрядной ячейке

Теоретически плавающая запятая имеет преимущества перед фиксированной.

Достоинство формы представления с ПЗ в том, что относительная ошибка не зависит от *порядка* числа. При этом *точность* представления больших и малых чисел изменяется незначительно. Такая форма представления снимает с программиста обязанность отслеживать положение запятой в вычислениях и значительно упрощает сам процесс программирования.

Недостатком формы представления чисел с ПЗ является то, что для ее реализации требуется сложное оборудование с большим числом микроопераций и соответственно меньшим быстродействием.

2.3. Кодирование отрицательных чисел

С целью упрощения аппаратной реализации арифметических устройств в вычислительной технике для хранения и обработки чисел используют специальные машинные коды: прямой, обратный и дополнительный. При таком кодировании знак числа «+» представляется как «0», а «–» представляется как «1». Код знака всегда располагается перед старшим значащим разрядом кода модуля числа. Для всех трех кодов изображения **положительных чисел** совпадают с их изображением в прямом коде, отличаются лишь изображения отрицательных чисел.

Прямой код (ПК) двоичного числа получается в виде его абсолютного значения и кода знака

Здесь в прямых кодах разделительные точки поставлены условно для выделения кода знака числа, в реальных цифровых устройствах их местоположения лишь подразумеваются.

Прямой код используется для хранения чисел в запоминающих устройствах и устройствах ввода–вывода.

Обратный код (ОК) отрицательного двоичного числа X представляет собой дополнение его модуля до константы *K*, т.е.

$$[X]_{\text{ОК}} = K - |X|,$$

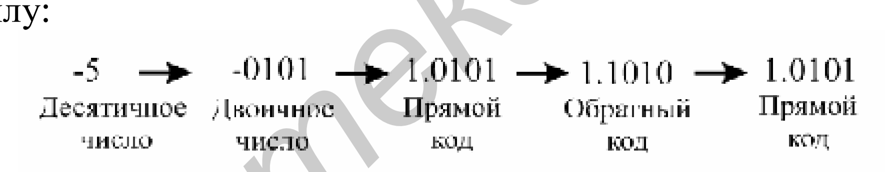
где $K = 2^{n-1} - 1$, n – количество разрядов, выделенных для хранения кода числа, включая знаковый.

На практике для получения ОК пользуются простым алгоритмом.

Простой алгоритм получения ОК

В знаковый разряд кода отрицательного числа записать «1», в цифровых разрядах «1» заменить на «0», а «0» заменить на «1».

Обратное преобразование от обратного к прямому коду осуществляется по тому же правилу:



Дополнительный код (ДК) отрицательного двоичного числа X представляет собой дополнение его модуля до константы *K*, т.е.

$$[X]_{ДК} = K - |X|,$$

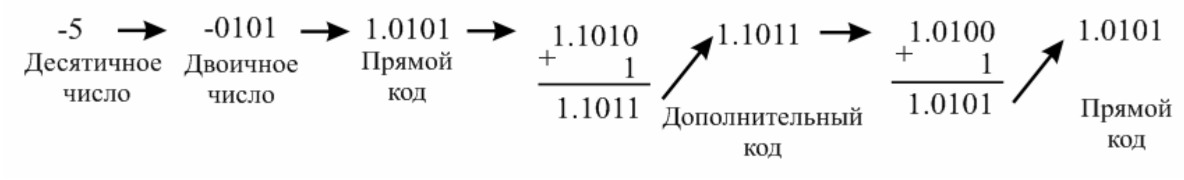
где $K = 2^{n-1}$, n – количество разрядов, выделенных для хранения кода числа, включая знаковый.

На практике для получения ДК пользуются простым алгоритмом.

Простой алгоритм получения ДК

В знаковый разряд кода отрицательного числа записать «1», в цифровых разрядах «1» заменить на «0», а «0» заменить на «1». После этого к младшему разряду прибавить «1».

Обратное преобразование от дополнительного к прямому коду осуществляется по тому же правилу:



Дополнительный и обратный коды в вычислительной технике используются для выполнения арифметических операций над числами со знаками.

Важным достоинством дополнительного и обратного кодов является то, что при выполнении арифметических операций сложения и вычитания цифру знакового разряда и цифровую часть числа можно рассматривать как единое целое и обращаться со знаковым разрядом так же, как и с разрядами цифровой части числа.

2.4. Задания для самостоятельного выполнения

1. Записать в прямом коде следующие числа:
а) 0,101; б) –0,011; в) 0,0001; г) –0,0101; д) 1101; е) –1011101; ж) 1011101.

2. Записать в обратном и дополнительном кодах следующие числа:
а) 0,1011; б) –0,1011; в) 0,1101; г) –0,01001; д) –0,1000; е) –0,0001; ж) –0,111111; з) 1101; и) –101101; к) 1011101.

3. Каким числам соответствуют следующие дополнительные коды (запятая в кодах правильных дробей условно отделяет знаковый разряд от дробной части, в кодах целых чисел запятая отсутствует):
а) 0,1100; б) 1,0001; в) 1,0111; г) 1,10000; д) 1,0101; е) 0,1010; ж) 1,01001; з) 01011; и) 101111; к) 1010110?

4. Пользуясь прямым кодом, записать в 8-разрядной ячейке в форме с фиксированной запятой следующие двоичные числа:
а) 0,1011010; б) –0,0101011; в) –0,111100101.

5. Пользуясь прямым кодом, записать в 16-разрядной ячейке в форме с плавающей запятой следующие двоичные числа (для мантиссы и порядка выделить по 8 двоичных разрядов):
а) 0,1010010; б) –11011; в) 1010111; г) 0,0001011011101.

Тема 3. Сложение и вычитание двоичных чисел

3.1. Сложение и вычитание целых чисел

Пусть два числа представлены в *n*-разрядном коде, где *n* – число разрядов, включая знаковый. При выполнении над ними операций сложения и вычитания принято руководствоваться следующими правилами.

1. Вычитание производится через сложение путем замены знака вычитаемого на противоположный, т.е. ***a – b = a + (–b)***.

2. Операнды представляются либо в обратном либо в дополнительном кодах.

3. Коды чисел складываются по правилам сложения двоичных чисел без знака разряд за разрядом, включая знаковые.

4. Возможный перенос из знакового разряда полученной суммы для дополнительного кода отбрасывается, а для обратного кода прибавляется к младшему разряду полученного результата. Такой перенос называют *круговым*, а сложение с ним – *циклическим*.

5. Ответ получается соответственно в обратном или дополнительном коде и будет правильным, если лежит в пределах от $-(2^{n-1}-1)$ до $+(2^{n-1}-1)$.

При сложении чисел с одинаковыми знаками результат может не поместиться в отведенную разрядную сетку. Такое явление называют *переполнением*. Чтобы избежать этого, необходимо увеличивать число разрядов для представления чисел. Наиболее простой способ обнаружения переполнения при сложении заключается в анализе переносов в знаковый и из знакового разрядов. Переполнение есть, если названные переносы различны. При сложении в обратном коде анализ переполнения необходимо производить до выполнения кругового переноса.

Примеры выполнения сложения целых двоичных чисел в обратном и дополнительном кодах показаны на рис. 3.1 и 3.2 соответственно.

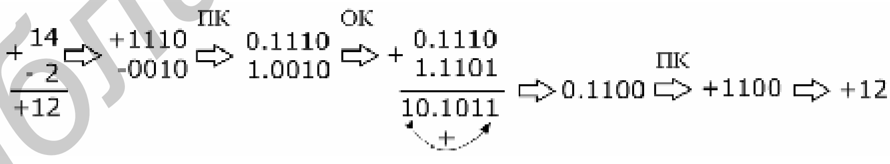


Рис. 3.1. Пример сложения в обратном коде

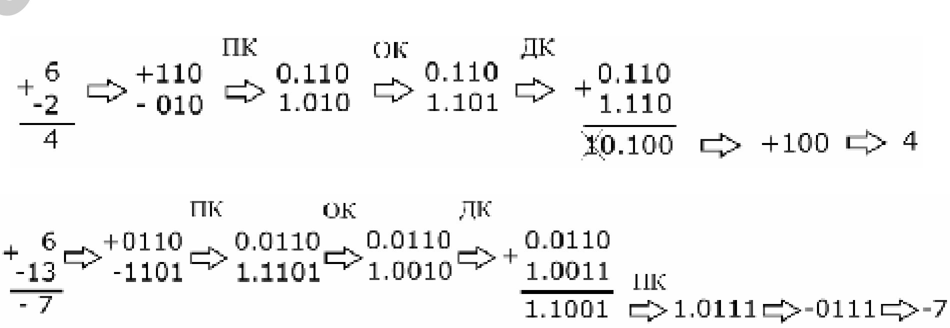


Рис. 3.2. Примеры сложения в дополнительном коде

На рис. 3.3 показано сложение в дополнительном коде при наличии переполнения разрядной сетки. Ответ неверный из-за переполнения разрядной сетки. В разряд знака был перенос, равный 0, а из разряда знака – равный 1. Это признак переполнения разрядной сетки.

Увеличив на единицу количество разрядов, выделяемых для представления обрабатываемых чисел, получаем правильный результат.

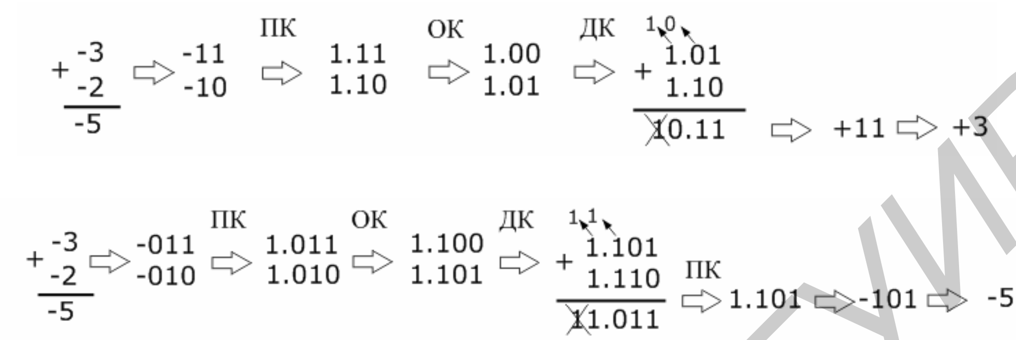


Рис. 3.3. Пример сложения в дополнительном коде при наличии переполнения разрядной сетки

3.2. Сложение и вычитание действительных чисел

Для чисел с фиксированной запятой при порядке, равном нулю, операции сложения и вычитания осуществляются над числами в пределах от –1 до +1. Для таких чисел методика выполнения операций сложения и вычитания в дополнительном и обратном кодах, включая признаки переполнения, аналогична методике сложения и вычитания целых чисел.

3.3. Задания для самостоятельного выполнения

1. Пользуясь обратными и дополнительными кодами, вычислить $S = A + B$, если:

а) A = –1010; B = 1000;
б) A = 0,11010; B = –0,10001;
в) A = –0,11010; B = 0,10001;
г) A = 0,1010; B = –0,1101;
д) A = 1011010; B = 1001001.

Проверить результаты вычислением в десятичной системе счисления.

2. Пользуясь обратными и дополнительными кодами, вычислить $R = A – B$, если:

а) A = 0,11101; B = 0,01011;
б) A = –0,0101; B = 0,1010;
в) A = –0,1001; B = –0,0111;
г) A = 1101010; B = 1010101;
д) A = 1010101; B = 1101010.

Проверить результаты вычислением в десятичной системе счисления.

3. Представить заданные числа в форме с плавающей запятой и вычислить сумму $S = A + B$, пользуясь дополнительными кодами:

а) A = 0,11101; B = 0,0101;

б) A = –0,0101; B = 0,101;

в) A = –0,1001; B = –0,0111;

г) A = 1101010; B = 1010101.

Проверить результаты вычислением в десятичной системе счисления.

Тема 4. Сложение и вычитание двоично-десятичных чисел

4.1. Сложение чисел без знака

Двоично-десятичные коды слагаемых складываются по правилам сложения двоичных чисел: разряд за разрядом, начиная с младшего (рис. 4.1). При этом разрешаются переносы между разрядами и между тетрадами, как показано на рис. 4.3.

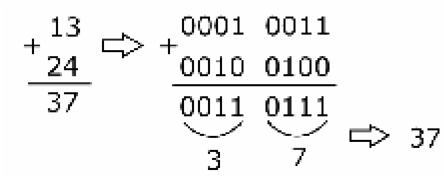


Рис. 4.1. Пример сложения двоично-десятичных чисел

Из-за того, что в тетрадах используются не все возможные комбинации, результат сложения требует корректировки. Корректировка производится в двух случаях:

1) если в тетрадах возникли недопустимые комбинации (10 … 15, т.е. 1010 … 1111);

2) если в процессе сложения были переносы между тетрадами.

Корректировка недопустимой комбинации осуществляется прибавлением корректирующего слагаемого 0110 к каждой недопустимой комбинации, при этом разрешается перенос единицы из одной тетрады в другую (рис. 4.2).

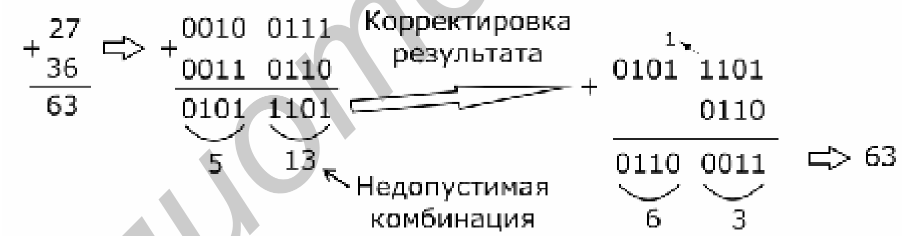


Рис. 4.2. Корректировка недопустимой комбинации

Корректировка при межтетрадных переносах производится также прибавлением корректирующего слагаемого 0110 к той тетраде, из которой был совершен перенос.

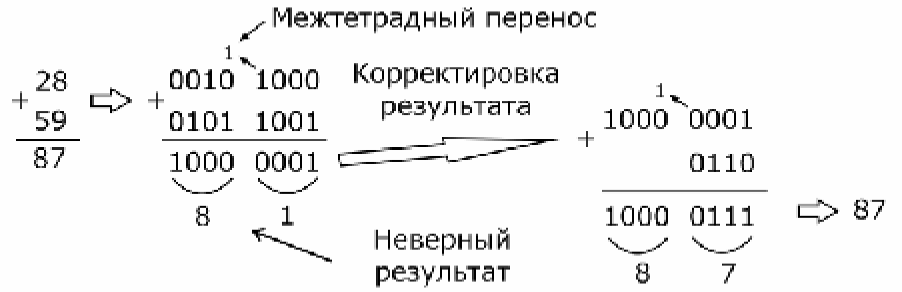


Рис. 4.3. Корректировка результата при межтетрадных переносах

4.2. Сложение чисел со знаками и вычитание

Сложение двоично-кодированных десятичных чисел со знаками производится с применением обратного и дополнительного кодов.

Вычитание реализуется через сложение путем замены знака вычитаемого на противоположный и представления операндов в обратном или дополнительном коде.

Обратный код двоично-десятичного числа $Y = y_0 y_1 y_2 \cdots y_i \cdots y_n$ получается из прямого кода по формуле $[Y]_{OK} = y'_0 y'_1 y'_2 \cdots y'_i \cdots y'_n$, где $y'_i = 9 - y_i$.

Дополнительный код двоично-десятичного числа Y получают из обратного по формуле $[Y]_{ДК} = [Y]_{OK} + 1$.

Пример выполнения сложения двоично-десятичных чисел со знаками с использованием обратного и дополнительного кодов показан на рис. 4.4.

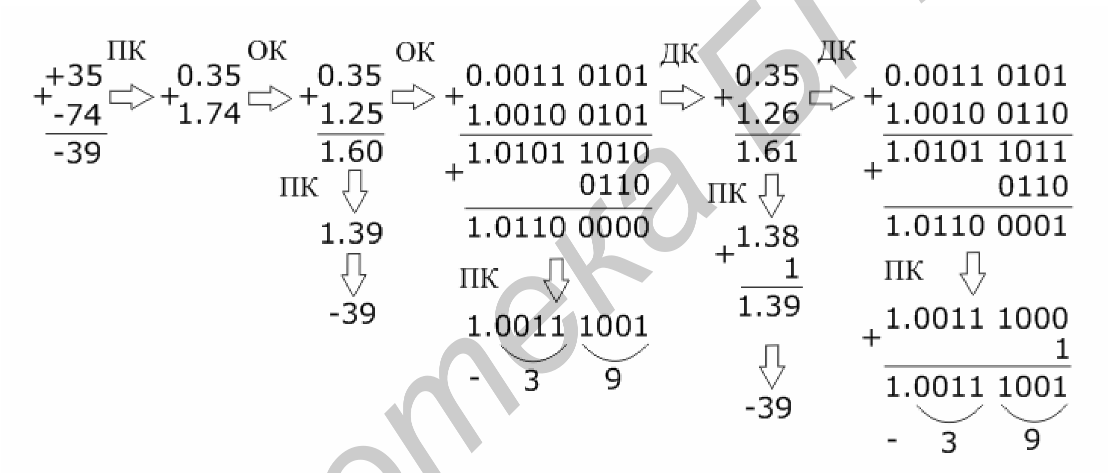


Рис. 4.4. Пример выполнения сложения двоично-десятичных чисел со знаками

4.3. Задания для самостоятельного выполнения

1. Перевести нижеуказанные числа А и В в двоично-десятичный код 8421, сложить в обратном и дополнительном кодах, проверить результаты вычислениями в десятичной системе счисления:

а) A = 356; B = 241;
б) A = 842; B = −456;
в) A = 123; B = −753;
г) A = 0,597; B = 0,346;
д) A = −0,2153; B = −0,1879;
е) A = –67; B = –94.

2. Перевести нижеуказанные числа A и B в двоично-десятичный код 8421, вычислить разность $R = A - B$, воспользовавшись дополнительными кодами, проверить результаты вычислениями в десятичной системе счисления:

а) A = 75; B = 24;
б) A = –82; B = –56;
в) A = 123; B = –753;
г) A = 0,597; B = 0,346;
д) A = –0,2153; B = –0,7879;
е) A = 67; B = 94.

Тема 5. Умножение и деление двоичных чисел

5.1. Умножение двоичных чисел в форме с фиксированной запятой

Умножение двоичных чисел реализуется через операции сложения и сдвига и выполняется в прямом коде. Знак произведения определяют по знаковым разрядам множимого и множителя: если знаки операндов одинаковы, результат положительный, если различны – отрицательный. Знак произведения не влияет на алгоритм выполнения операции умножения.

Наиболее простой алгоритм умножения заключается в многократном сложении множимого самого с собой. При этом количество операций сложения равно множителю. Данный алгоритм применим для перемножения целых чисел, но его недостатком является низкое быстродействие.

На практике чаще используют другой алгоритм – аналогичный привычному алгоритму умножения вручную в десятичной системе счисления. Результат получают суммированием частичных произведений, являющихся результатом умножения множимого на значения очередных разрядов множителя, начиная с младшего. При этом каждое следующее частичное произведение вдвое превышает предыдущее, что соответствует его сдвигу на один разряд влево. На рис. 5.1 приведена иллюстрация алгоритма умножения на примере перемножения чисел 13 и 11 в десятичной и двоичной системах счисления.

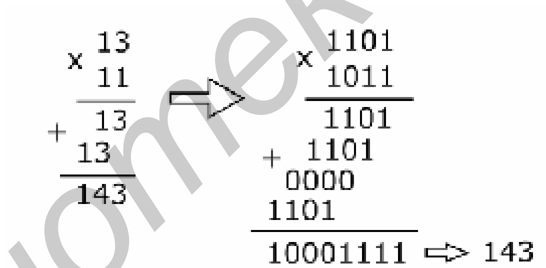


Рис. 5.1. Иллюстрация алгоритма умножения на примере перемножения чисел 13 и 11 в десятичной и двоичной системах счисления

Алгоритм перемножения правильных двоичных дробей ничем не отличается от алгоритма перемножения целых двоичных чисел. При перемножении правильных дробей младшие разряды, выходящие за пределы разрядной сетки, могут быть отброшены с округлением или без округления.

Наибольшую скорость умножения обеспечивают статические множительные устройства, хранящие таблицы умножения многоразрядных чисел. Однако эти устройства сложны и реализуемы для двоичных чисел сравнительно небольшой разрядности.

5.2. Деление двоичных чисел в форме с фиксированной запятой

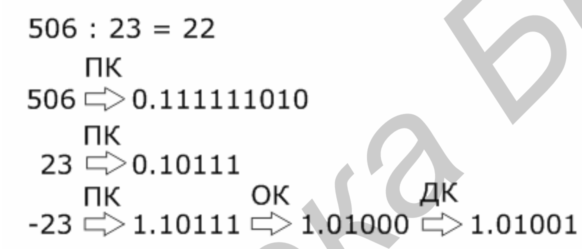
Операцию деления двоичных чисел выполняют способом, аналогичным делению вручную. Знак частного определяется так же, как и при умножении.

Наибольшее распространение получили два алгоритма деления – с *восстановлением* и *без восстановления остатка*.

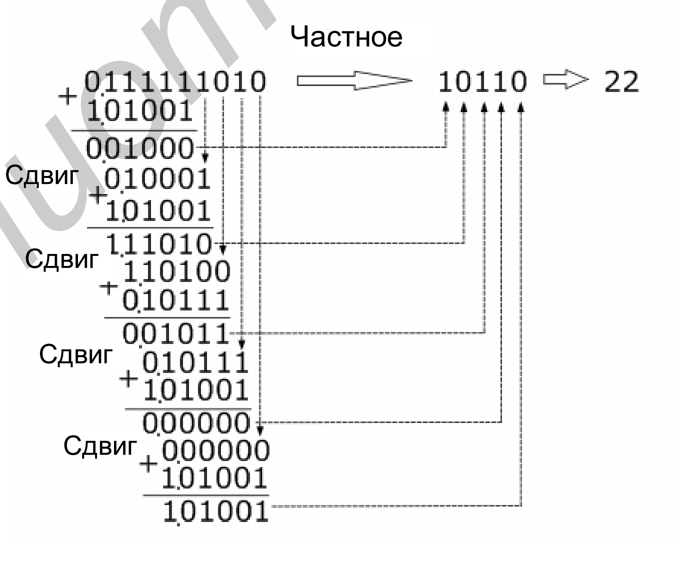
Алгоритм деления без восстановления остатка

1. Из делимого вычитается делитель.
2. Если остаток положителен, первая цифра частного равна единице, если отрицателен – нулю.
3. Остаток сдвигается влево на один разряд и к нему прибавляется делитель со знаком, обратным знаку остатка. Знак следующего остатка определяет следующую цифру частного.
4. Действие по п. 2 проводится до тех пор, пока не получится требуемое число разрядов частного или нулевой остаток.

На рис. 5.2 приведен пример деления целых чисел 506 на 22 алгоритмом без восстановления остатка. Предварительно операнды переведены в двоичную систему счисления, делитель 23 представлен в двух вариантах: как положительное и как отрицательное число в дополнительном коде для выполнения сложения и вычитания:



а



б

Рис. 5.2. Пример деления целых чисел 506 на 22 алгоритмом без восстановления остатка:
а – подготовка операндов; б – процесс деления

5.3. Умножение и деление двоичных чисел с плавающей запятой

Умножение чисел в форме с плавающей запятой, т.е. в показательной форме, производится в четыре этапа.

1. Определяется знак произведения.
2. Перемножаются мантиссы сомножителей по правилам для чисел с фиксированной запятой.
3. Устанавливается порядок произведения алгебраическим сложением порядков сомножителей по правилам суммирования целых чисел со знаком.
4. Производится нормализация полученного результата в случае необходимости.

Деление чисел в устройствах с плавающей запятой производится так же, как и умножение. Поскольку все числа в этом случае не должны превышать единицы, мантисса делимого всегда должна быть выбрана меньше мантиссы делителя, что достигается масштабированием. Порядок частного определяется алгебраическим вычитанием порядка делителя из порядка делимого. В случае необходимости также производится нормализация результата.

5.4. Задания для самостоятельного выполнения

1. Рассчитать произведения $P = A{\cdot}B$, если

а) A = 110110; B = 100101;
б) A = –110011; B = –10101;
в) A = 0,101001; B = –0,11001;
г) A = $–0,100111 \cdot 2^{+110}$; B = $0,1101 \cdot 2^{–11}$;
д) A = $0,100111 \cdot 2^{–111}$; B = $–0,1101 \cdot 2^{+101}$.

2. Рассчитать $D = A{:}B$ с точностью до 6 знаков после запятой, если

а) A = 110110; B = 100101;
б) A = –110011; B = –10101;
в) A = 0,101001; B = –0,11001;
г) A = $–0,100111 \cdot 2^{+110}$; B = $0,1101 \cdot 2^{–11}$;
д) A = $0,1110111 \cdot 2^{–111}$; B = $–0,1101 \cdot 2^{+101}$.

Тема 6. Тождественные преобразования логических выражений

6.1. Логические операции

В алгебре логики оперируют логическими переменными, которые могут принимать два значения, называемые «истина» и «ложь», или «логическая единица» и «логический ноль». Для простоты записи их обозначают соответственно 1 и 0. Между логическими переменными, которые принято обозначать символами латинского алфавита с индексами или без них, могут устанавливаться функциональные связи, которые могут задаваться в виде таблиц истинности или логических (булевых) выражений (уравнений). При аналитических записях функциональных связей между переменными используют три основные логические операции: *логическое отрицание (инверсия), логическое умножение (конъюнкция), логическое сложение (дизъюнкция)*.

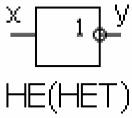
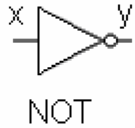
Таблица истинности содержит значения функции для всех возможных комбинаций значений аргументов.

Булево выражение представляет собой формулу, состоящую из логических констант и логических переменных, соединенных знаками операций. Иерархия логических операций такова: отрицание, логическое умножение, логическое сложение. Порядок выполнения операций может быть изменен скобками.

Для аппаратной реализации основных логических операций применяют соответствующие логические элементы (ЛЭ): НЕ (НЕТ), И, ИЛИ либо их англоязычные аналоги: NOT, AND, OR.

Логическим отрицанием (инверсией) называется такая логическая связь между входной логической переменной x и выходной логической переменной y, при которой y истинно (y = 1) только тогда, когда x ложно (x = 0) и наоборот. Инверсия здесь обозначена черточкой над переменной $\overline{x}$, при двойной инверсии $\overline{\overline{x}} = x$, $\overline{\overline{1}} = \overline{0} = 1$. Формы представления операции инверсии показаны в табл. 6.1.

Таблица 6.1

Таблица истинности	Логическое уравнение	Условное графическое изображение ЛЭ	Англоязычный аналог
x y 0 1 1 0	$y = \overline{x}$	 НЕ(НЕТ)	 NOT

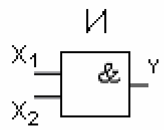
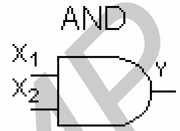
Операция инверсии на схемах обозначается кружком на выходе инвертора.

Логическим умножением (конъюнкцией) нескольких переменных x_n называется такая логическая функция, которая принимает истинное значение (y = 1), когда истинны одновременно все умножаемые переменные

$(x_1 \cdot x_2 \cdot \ldots \cdot x_n = 1)$, если хотя бы одна из входных переменных x = 0, то функция y = 0.

Формы представления операции логического умножения показаны в табл. 6.2.

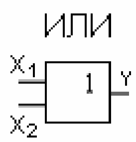
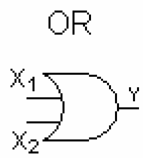
Таблица 6.2

Таблица истинности	Логическое уравнение	Условное графическое изображение ЛЭ	Англоязычный аналог
<table><tr><th>x_1</th><th>x_2</th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	$y = x_1 \cdot x_2 =$ $= x_1 x_2;$ $y = x_1 \wedge x_2$		

В правой части таблицы истинности приведены результаты логического умножения аргументов x_1 и x_2. Логическое умножение обозначают знаком конъюнкции ($\wedge$). Однако в инженерной практике употребляется либо точка ($\bullet$), либо знак вообще опускают.

Логическим сложением (дизъюнкцией) нескольких переменных x_n называется такая логическая функция, которая принимает ложное значение (y = 0), когда одновременно равны нулю все слагаемые переменные $(x_1 + x_2 + \ldots + x_n = 0)$. Если хотя бы одна из входных переменных $x_n = 1$, то функция y = 1. Формы представления операции логического сложения приведены в табл. 6.3.

Таблица 6.3

Таблица истинности	Логическое уравнение	Условное графическое изображение ЛЭ	Англоязычный аналог
<table><tr><th>x_1</th><th>x_2</th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	$y = x_1 + x_2;$ $y = x_1 \vee x_2$		

В правой части таблицы истинности приведены результаты логического суммирования аргументов x_1 и x_2. Логическое сложение обозначают знаком дизъюнкции ($\vee$), а арифметическое сложение – знаком (+).

Рассмотренные логические операции инверсии, логического умножения (конъюнкции) и логического сложения (дизъюнкции) называют для краткости операциями НЕ, И, ИЛИ, аналогично названиям логических элементов, реализующих эти операции.

Используя операции НЕ, И, ИЛИ, можно описать поведение любого логического устройства.

6.2. Основные законы и правила алгебры логики

Для преобразования сложных логических выражений в алгебре логики используют ряд законов. Ниже приведены формулировки каждого из законов в двух вариантах: сначала для логического сложения, затем для логического умножения.

1. Переместительный:

а) $x_1 \vee x_2 = x_2 \vee x_1$,

б) $x_1 \cdot x_2 = x_2 \cdot x_1$.

2. Сочетательный:

а) $(x_1 \vee x_2) \vee x_3 = x_1 \vee (x_2 \vee x_3)$,

б) $(x_1 \cdot x_2) \cdot x_3 = x_1 \cdot (x_2 \cdot x_3)$.

3. Распределительный:

а) $x_1 \cdot (x_2 \vee x_3) = x_1 \cdot x_2 \vee x_1 \cdot x_3$,

б) $$(x_1 \vee x_2) \cdot (x_1 \vee x_3) = x_1 \cdot x_1 \vee x_1 \cdot x_3 \vee x_1 \times$$
$$\times x_2 \vee x_2 \cdot x_3 = x_1 \cdot (1 \vee x_3 \vee x_2) \vee x_2 \cdot x_3 =$$
$$= x_1 \vee x_2 \cdot x_3.$$

4. Инверсии (правило де Моргана):

а) $\overline{x_1 \vee x_2} = \overline{x}_1 \cdot \overline{x}_2$; $\overline{\overline{a} \vee b \vee c \cdot z} = a \cdot \overline{b} \cdot \overline{c \cdot z}$,

б) $\overline{x_1 \cdot x_2} = \overline{x}_1 \vee \overline{x}_2$; $\overline{a \cdot b \cdot \overline{c \cdot z}} = \overline{a} \vee \overline{b} \vee c \cdot z$.

5. Склеивания:

а) $x_1 \cdot x_2 \vee x_1 \cdot \overline{x}_2 = x_1 \cdot (x_2 \vee \overline{x}_2) = x_1$,

б) $$(x_1 \vee x_2) \cdot (x_1 \vee \overline{x}_2) = x_1 \cdot x_1 \vee x_1 \cdot \overline{x}_2 \vee x_1 \cdot x_2 \vee x_2 \cdot \overline{x}_2 =$$
$$= x_1 \vee x_1 \cdot \overline{x}_2 \vee x_1 \cdot x_2 \vee 0 = x_1 \vee x_1(\overline{x}_2 \vee x_2) =$$
$$= x_1 \vee x_1 \cdot 1 = x_1.$$

6. Поглощения:

а) $x_1 \vee (x_1 \cdot x_2) = x_1 \cdot (1 \vee x_2) = x_1$,

б) $$x_1 \cdot (x_1 \vee x_2) = x_1 \cdot x_1 \vee x_1 \cdot x_2 =$$
$$= x_1 \vee x_1 \cdot x_2 = x_1 \cdot (1 \vee x_2) = x_1.$$

Операции логического умножения (конъюнкции) и логического сложения (дизъюнкции) для случая одной переменной x обладают рядом свойств, которые в литературе называют правилами (аксиомами) алгебры логики (табл. 6.4):

Таблица 6.4

Правило	Для логического сложения	Для логического умножения
1. Неизменности	$x \vee 0 = x$	$x \cdot 1 = x$
2. Повторения	$x \vee x = x$	$x \cdot x = x$
3. Универсального и нулевого множеств	$x \vee 1 = 1$	$x \cdot 0 = 0$
4. Дополнительности	$x \vee \overline{x} = 1$	$x \cdot \overline{x} = 0$
5. Двойного отрицания	$\overline{\overline{x}} = x$	

6.3. Дизъюнктивные нормальные формы логических выражений

Для записи одной и той же функции алгебры логики можно использовать много разных форм. Формы, которые представляют собой суммы элементарных произведений, называют дизъюнктивными нормальными формами (ДНФ). Под элементарным понимается такое произведение, в котором сомножителями являются только отдельные переменные или их отрицания.

Пример. ДНФ логического выражения:

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

Очевидно, что одна и та же функция может быть представлена множеством различных ДНФ. Однако существуют такие виды ДНФ, в которых каждая функция может быть записана единственным образом. Такие формы называют совершенными дизъюнктивными нормальными формами (СДНФ).

СДНФ представляет собой логическое выражение, состоящее из суммы элементарных произведений, в каждом из которых каждая из переменных встречается ровно один раз либо с отрицанием, либо без него.

Пример. СДНФ логического выражения:

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

СДНФ можно получить из ДНФ путем тождественных преобразований.

Наиболее простой способ получения СДНФ по таблице истинности функции. Для этого для всех комбинаций значений входных переменных, обращающих функцию в единицу, надо записать элементарные произведения, инвертируя переменные, принимающие на данной комбинации нулевые значения, а все полученные элементарные произведения соединить знаками логического суммирования.

Пример. Пусть функция алгебры логики двух переменных задана следующей таблицей истинности:

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

Тогда ее СДНФ будет иметь вид $y = f(x_1, x_2) = \overline{x}_1 x_2 \vee x_1 \overline{x}_2$.

6.4. Преобразования логических выражений

Рассмотрим примеры применения законов и правил алгебры логики для тождественных преобразования логических выражений.

Пример 1. Используя закон инверсии, преобразуем запись выражения так, чтобы выражения содержали только операции *И* и *НЕ*, а также *ИЛИ* и *НЕ*:

$$x_1 \vee \overline{x}_1 \cdot x_2 = \overline{\overline{x_1 \vee \overline{x}_1 \cdot x_2}} = \overline{\overline{x}_1 \cdot \overline{\overline{x}_1 \cdot x_2}} = \overline{\overline{x}_1 \cdot (x_1 \vee \overline{x}_2)} =$$
$$= \overline{\overline{x}}_1 \vee \overline{(x_1 \vee \overline{x}_2)} = x_1 \vee \overline{x_1 \vee \overline{x}_2}.$$

Пример 2. Упростить выражение, используя правила алгебры логики:

$$(\overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \vee \overline{x}_1 \cdot x_2) \cdot (x_1 \vee x_3) =$$
$$= \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \cdot x_1 \vee \overline{x}_1 \cdot x_2 \cdot x_1 \vee \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \cdot x_3 \vee \overline{x}_1 \cdot x_2 \cdot x_3 =$$
$$= 0 \vee 0 \vee \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \vee \overline{x}_1 \cdot x_2 \cdot x_3 = \overline{x}_1 \cdot x_3 \cdot (\overline{x}_2 \vee x_2) = \overline{x}_1 \cdot x_3.$$

Пример 3. Получить СДНФ функции, заданной ДНФ:

$$y = f(x_1, x_2) = x_1 \vee x_2 = x_1 \cdot (x_2 \vee \overline{x}_2) \vee x_2 \cdot (x_1 \vee \overline{x}_1) =$$
$$= x_1 \cdot x_2 \vee x_1 \cdot \overline{x}_2 \vee x_1 \cdot x_2 \vee \overline{x}_1 \cdot x_2 = x_1 \cdot x_2 \vee x_1 \cdot \overline{x}_2 \vee \overline{x}_1 \cdot x_2 .$$

Законы и правила алгебры логики широко используются при решении задач минимизации (упрощения) логических выражений.

6.5. Задания для самостоятельного выполнения

1. Получить СДНФ функций, заданных ДНФ, и построить для них таблицы истинности:

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

2. Упростить выражения, используя законы и правила алгебры логики:

$$y = f(x_1, x_2, x_3) = x_1 x_2 x_3 \vee \overline{x}_1 x_2 x_3 \vee x_1 \overline{x}_2 x_3 \vee x_1 x_2 \overline{x}_3 \vee x_1 \overline{x}_2 \overline{x}_3 .$$
$$y = f(x_1, x_2, x_3) = \overline{[(x_1 \vee x_2 \vee \overline{x}_3)\overline{x_1 \overline{x}_2}]} \vee \overline{x}_1 \overline{x}_3 .$$

3. Преобразовать логические выражения так, чтобы они содержали только операции *И* и *НЕ*, а также *ИЛИ* и *НЕ*:

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

Тема 7. Минимизация логических функций

7.1. Аналитические и табличные методы минимизации

Любая логическая функция реализуется с помощью определенного набора логических элементов и чем проще запись функции, тем проще и дешевле ее реализация.

Различают аналитические и табличные методы минимизации логических функций.

Аналитические методы основаны на рассмотренных выше тождественных преобразованиях логических выражений по законам и правилам алгебры логики. Аналитические методы не поддаются четкой алгоритмизации и успех здесь зависит только от опыта и квалификации исполнителя.

Табличные методы поддаются алгоритмизации. Наибольшее распространение получил метод карт Карно. Его применение эффективно при количестве переменных до шести.

Карта Карно – это графическое представление таблицы истинности функции алгебры логики. В карте Карно столько же клеток, сколько строк в таблице истинности, т.е. 2^n, где n – число логических переменных. Например, карта Карно функции четырех переменных имеет $2^4=16$ клеток.

В карте Карно значения функции записываются в клетки, которые размещаются в пространстве координат, образованных значениями аргументов этой функции. При этом клетки размещаются так, чтобы они были соседними в геометрическом смысле, т.е. их координаты отличались значением только одного из аргументов.

Структуры карт Карно для функций двух, трех и четырех переменных показаны на рис. 7.1, 7.2 и 7.3 соответственно.

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

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

Рис. 7.1. Структуры таблицы истинности и соответствующей ей карты Карно для функции двух переменных

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

Рис. 7.2. Структура карты Карно для функции трех переменных

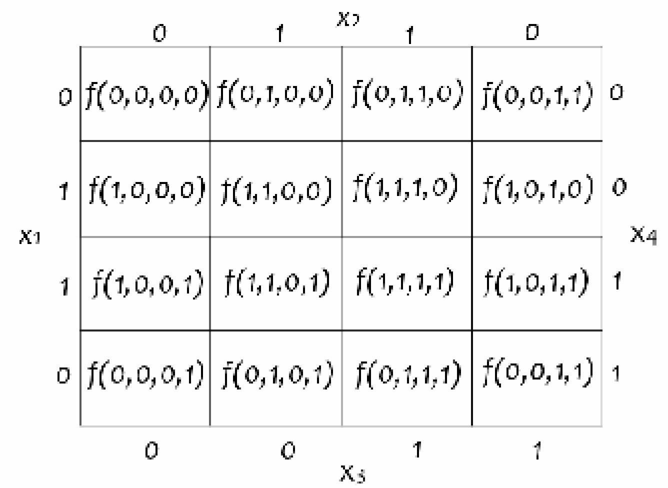


Рис. 7.3. Структура карты Карно для функции четырех переменных

Проиллюстрируем процесс минимизации функции с помощью карты Карно.

Пусть функция трех переменных задана таблицей истинности, показанной на рис. 7.4. Соответствующая ей карта Карно приведена на рис. 7.5. Там же изображены построения, выполняемые в процессе минимизации.

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

Рис. 7.4. Таблица истинности функции

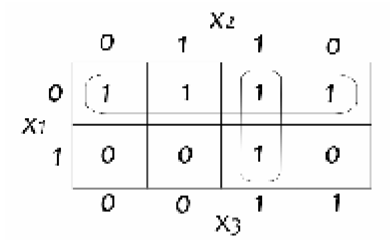


Рис. 7.5. Карта Карно функции

Процесс минимизации по карте Карно заключается в следующем.

1. Клетки, содержащие единицы, объединяются замкнутыми прямоугольными контурами в группы по правилам:

а) число единиц в группе должно быть 2^k, где k = 0, 1, 2, 3…, (т.е. допустимое число ячеек в контуре 1, 2, 4, 8,…);

б) внутри контура должны быть только ячейки, заполненные единицами, одна и та же ячейка может входить в несколько контуров;

в) крайние столбцы и строки карты Карно считаются соседними и их ячейки могут входить в общий контур.

2. Набор контуров, покрывающих все единичные клетки, называется *покрытием*. Вариантов покрытия может быть много. Среди них надо выбрать наилучшее. Качество покрытия оценивается коэффициентом покрытия $K_n = a/s$, где a – общее количество групп, s – их суммарная площадь.

Покрытие считается тем лучше, чем меньше коэффициент K_n.

Применительно к карте Карно, изображенной на рис. 7.5, коэффициент предложенного покрытия $K_n = 2/6 = 0{,}333$.

3. По найденному наилучшему покрытию записывается функция в *минимальной дизъюнктивной нормальной форме* (*МДНФ*), представляющей собой сумму элементарных произведений. При этом число произведений в минимальной функции равно числу прямоугольных контуров в покрытии. При записи произведений в них включаются только те переменные, от которых функция зависит в пределах контура.

Для рассматриваемого примера искомая функция примет следующий вид:

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

На рис. 7.6 показаны варианты наилучших покрытий карт Карно для ряда функций.

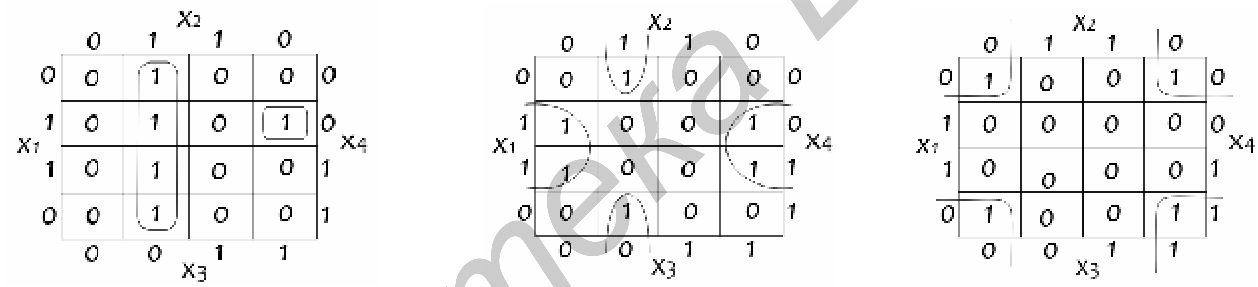


Рис. 7.6. Варианты наилучших покрытий карт Карно

7.2. Задания для самостоятельного выполнения

1. Минимизировать функции, заданные логическими выражениями, методом карт Карно и аналитическим методом, сравнить результаты:

$$y = f(x_1, x_2, x_3) = x_1x_2x_3 \vee \overline{x}_1x_2x_3 \vee x_1\overline{x}_2x_3 \vee x_1x_2\overline{x}_3;$$

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

2. Минимизировать с помощью карт Карно заданные таблицами истинности, приведенными на рис. 7.7 и 7.8, функции f_1 и f_2 четырех переменных:

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

Рис. 7.7. Таблица истинности функции f_1

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

Рис. 7.8. Таблица истинности функции f_2

Тема 8. Синтез комбинационных устройств на логических элементах

8.1. Порядок синтеза комбинационных устройств

Устройства, предназначенные для реализации логических функций, называются *логическими устройствами* (или *схемами*).

По способу функционирования логические устройства делятся на два типа: комбинационные и последовательностные.

В *комбинационных* логических схемах (сумматоры, дешифраторы, мультиплексоры) выходные сигналы зависят только от текущего значения входной комбинации логических сигналов без их запоминания. В комбинационных логических схемах отсутствуют обратные связи.

В *последовательностных* логических схемах (триггеры, счетчики, регистры) выходные сигналы зависят не только от набора входных сигналов, действующих в данный момент времени, но и от состояния внутреннего запоминающего устройства, сохраняющего сведения о последовательности значений входных сигналов в прошлом.

Порядок синтеза комбинационных устройств:

1. Словесная формулировка задачи, в которой оговаривается число входных и выходных переменных и функциональные связи между ними.
2. Составление таблицы истинности синтезируемой функции (или функций).
3. Запись функции (или функций) в виде логического выражения (или системы логических уравнений).
4. Минимизация функции (или функций) аналитическим или табличным методом.
5. Построение логической схемы из конкретных логических элементов с использованием логического уравнения (или системы уравнений) в минимальной форме как структурной формулы.

Проиллюстрируем данную методику на примере синтеза мажоритарного элемента $M \geq 2$ для трех переменных.

1. Формулируем связь между переменными и функцией.

Для мажоритарного элемента (рис. 8.1) значение функции $F(x)$ считается истинным, если хотя бы два аргумента из трех приняли единичные значения, в остальных случаях функция $F(x) = 0$.

2. Составляем таблицу истинности (рис. 8.2).

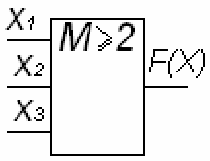


Рис. 8.1. Мажоритарный элемент

X_1	X_2	X_3	F(X)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Рис. 8.2. Таблица истинности мажоритарного элемента

3. По таблице истинности записываем функцию алгебры логики (ФАЛ) для мажоритарного элемента в СДНФ:

$$F(X) = \overline{X_1} \cdot X_2 \cdot X_3 + X_1 \cdot \overline{X_2} \cdot X_3 + X_1 \cdot X_2 \cdot \overline{X_3} + X_1 \cdot X_2 \cdot X_3 .$$

4. Используя законы и правила алгебры логики, произведем преобразования с целью получения минимальной формы логического выражения:

$$F(X) = \left(\overline{X_1} + X_1\right) \cdot X_2 \cdot X_3 + \left(\overline{X_2} + X_2\right) \cdot X_1 \cdot X_3 + \left(\overline{X_3} + X_3\right) \cdot X_1 \cdot X_2 =$$

$$= X_1 \cdot X_3 + X_1 \cdot X_2 + X_2 \cdot X_3.$$

5. Используя логические элементы, в соответствии с полученным выражением строим логическую схему (рис. 8.3).

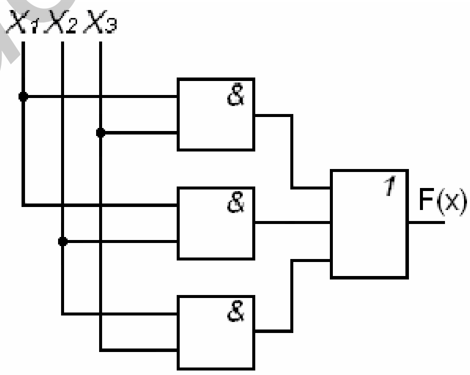


Рис. 8.3. Логическая схема мажоритарного элемента

8.2. Задания для самостоятельного выполнения

1. Синтезировать устройство арифметического сложения двух одноразрядных чисел. Реализовать его на элементах: 1) И, ИЛИ, НЕ; 2) И-НЕ; 3) ИЛИ-НЕ.

2. Синтезировать устройство, имеющее три входа и один выход, которое формирует на выходе логическую единицу, если на входах одновременно присутствует не более одной единицы. Реализовать его на элементах: 1) И, ИЛИ, НЕ; 2) И-НЕ; 3) ИЛИ-НЕ.

3. Синтезировать устройство сравнения двух двухразрядных чисел A и B (цифровой компаратор), формирующий три выходных сигнала: A = B, A > B и A < B.

4. Синтезировать устройство, имеющее четыре входа и один выход, которое формирует на выходе логическую единицу, если на входах одновременно присутствует не более двух единиц. Реализовать его на элементах: 1) И, ИЛИ, НЕ; 2) И-НЕ; 3) ИЛИ-НЕ.

Литература

1. Безуглов, Д. А. Цифровые устройства и микропроцессоры: учеб. пособие для вузов / Д. А. Безуглов, И. В. Калиенко. – Ростов н/Д : Феникс, 2008. – 468 с.

2. Браммер, Ю. А. Цифровые устройства: учеб. пособие для вузов / Ю. А. Браммер, И. Н. Пащук. – М. : Высш. шк., 2004. – 229 с.

3. Новожилов, О. П. Основы цифровой техники. – М. : ИП РадиоСофт, 2004. – 528 с.

4. Угрюмов, Е. П. Цифровая схемотехника: учеб. пособие для вузов / Е. П. Угрюмов. – СПб. : БХВ-Петербург, 2004. – 528 с.

СОДЕРЖАНИЕ

Св. план 2009, поз. 83

Учебное издание

Левкович Василий Николаевич

Мартинович Алексей Васильевич

ЦИФРОВЫЕ И МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА

Методическое пособие к практическим занятиям
для студентов радиотехнических специальностей
всех форм обучения

Редактор Г. С. Корбут
Корректор Е. Н. Батурчик
Компьютерная верстка Е. Г. Бабичева

Подписано в печать 08.05.2009.	Формат 60х84 1/16.	Бумага офсетная.
Гарнитура «Таймс».	Печать ризографическая.	Усл. печ. л. 2,21.
Уч.-изд. л. 2,0.	Тираж 150 экз.	Заказ 90.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0494175 от 03.04.2009.
220013, Минск, П. Бровки, 6