Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Информационные технологии»

В. В. Кротенок

ЛОГИЧЕСКИЕ И АРИФМЕТИЧЕСКИЕ ОСНОВЫ И ПРИНЦИПЫ РАБОТЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КУРС ЛЕКЦИЙ
для студентов специальности 1-40 01 02
«Информационные системы
и технологии (по направлениям)»
дневной и заочной форм обучения

Электронный аналог печатного издания

Гомель 2012

УДК 004.3(075.8)
ББК 32.973.26-02я73
К83

Рекомендовано к изданию научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 4 от 28.11.2011 г.*)

Рецензент: зав. каф. «Электроснабжение» ГГТУ им. П. О. Сухого канд. техн. наук,
доц. *О. Г. Широков*

Кротенок, В. В.

К83 Логические и арифметические основы и принципы работы вычислительной техники : курс лекций для студентов специальности 1-40 01 02 «Информационные системы и технологии (по направлениям)» днев. и заоч. форм обучения / В. В. Кротенок. – Гомель : ГГТУ им. П. О. Сухого, 2012. – 68 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: http://lib.gstu.local. – Загл. с титул. экрана.

ISBN 978-985-535-116-1.

Изложен материал, связанный с арифметическими, логическими и схемотехническими вопросами построения ЭВМ. Включает большое количество иллюстраций, таблиц и примеров с решениями.

Для студентов специальности 1-40 01 02 «Информационные системы и технологии (по направлениям)» дневной и заочной форм обучения.

УДК 004.3(075.8)
ББК 32.973.26-02я73

ISBN 978-985-535-116-1

Введение

Начало развития машин для выполнения вычислений можно отнести к семнадцатому столетию, когда в 1642 г. выдающийся французский математик Блез Паскаль[1] изобрел и сконструировал первую суммирующую машину. Однако из-за несовершенства развития механики того времени созданная им машина не нашла практического применения. В конце этого же столетия, в 1694 г. немецкий математик Гофрид Лейбниц построил вычислительную машину, которая могла выполнять не только операции сложения, но и операции умножения. В 1874 г. российский инженер В. Т. Однер сконструировал арифмометр, которой был лучшим для того времени и долгое время оставался прототипом разрабатывавшихся впоследствии машин подобного назначения.

Все разрабатываемые машины для выполнения расчетов до 40-х гг. строились на основе механических узлов. Появление первых электронных вычислительных машин можно отнести к сороковым годам. Одной из первых ЭВМ является «ENIAC», созданная в США Джоном Мокли и Дж. Преспером Эккертом. При построении машины было использовано около 20 тыс. электронных ламп. Она имела колоссальные габариты (для ее размещения было построено специальное помещение площадью более 100 м) и имела фантастическое по тем временам быстродействие – 5 тыс. сложений в секунду.

С 40-х гг. развитие вычислительной техники с точки зрения используемой технологической базы можно подразделить на этапы создания ЭВМ следующих поколений:

– ЭВМ на электронных вакуумных приборах;
– ЭВМ на полупроводниковых приборах;
– ЭВМ на интегральных схемах;
– ЭВМ на больших интегральных схемах;
– ЭВМ на сверхбольших интегральных схемах.

При переходе от поколения к следующему поколению почти на порядок улучшались основные параметры ЭВМ, к числу которых относятся:

– быстродействие;
– емкость памяти;
– потребляемая мощность;
– габариты.

---

[1] Именем этого математика был назван один из самых распространенных современных алгоритмических языков программирования.

За это время существенно изменилась структурная организация ЭВМ, ее внешняя память, средства ввода-вывода информации.

Структура современной ЭВМ включает следующие основные компоненты:

– ОП – оперативная память, используемая для хранения исполняемых в данное время программ, исходных данных, промежуточных и окончательных результатов;

– процессор – устройство, осуществляющее основную обработку информации в соответствии с исполняемой программой;

– ПУ – периферийные устройства, включающие средства ввода-вывода информации, представленной в различной форме, осуществляющие двусторонний обмен данными с пользователем, а также устройства типа внешней памяти, имеющие огромную информационную емкость, позволяющую хранить все исходные данные, программы, промежуточные и конечные результаты обработки информации.

В данной книге рассматриваются основополагающие материалы, связанные с ЭВМ, а именно:

– арифметические основы ЭВМ;

– логические основы ЭВМ;

– схемотехнические основы ЭВМ.

Приведенные материала основываются на разделах «Физика», «Математика» и «Электротехника».

Полученные при их изучении знания могут быть использованы при изучении дисциплин по программированию («Конструирование программ и языки программирования», «Системное программное обеспечение» и др.) и всех дисциплин, связанных с аппаратной частью ЭВМ.

1. СИСТЕМЫ СЧИСЛЕНИЯ

Арифметическая обработка чисел во многом определяется системами счисления, представляющими собой совокупность используемых цифр и набором правил, позволяющих однозначно представлять числовую информацию.

В своей повседневной деятельности человек использует различные системы счисления, к числу которых относятся десятичная система счисления, римская система, система исчисления времени и т. д. Все системы счисления можно подразделить на позиционные и непозиционные.

В непозиционных системах счисления «доля» цифры или ее вес в количественном измерении записанного числа не зависит от местоположения данной цифры в записи этого числа. Типичным примером такой системы счисления является римская система счисления. В этой системе используются римские, десятичные цифры и эквиваленты римским цифрам:

I V X L C D M и т. д.

1 5 10 50 100 500 1000.

При количественной оценке числа его значение определяется как сумма значений цифр, составляющих запись числа, кроме пар, состоящих из цифры меньшего веса, предшествующей цифре большего веса, значение которой определяется как разность веса большей и меньшей цифр.

Например, значение числа MMMCMLIX определяется как сумма 1000 + 1000 + 1000 + (1000 – 100) + 50 + (10 – 1), что соответствует десятичному эквиваленту 3959.

Позиционная система счисления характеризуется тем, что «доля» некоторой цифры в количественной оценке записанного числа определяется не только видом цифры, но и местоположением (позицией) данной цифры в записи числа, т. е. каждая позиция (разряд) в записи числа имеет определенный вес.

Количественная оценка записанного числа в такой системе счисления определяется как сумма произведений значения цифр, составляющих запись числа, умноженных на вес позиции, в которой располагается цифра.

Примером такой системы счисления является широко используемая *десятичная система счисления*. Например, количественная

оценка десятичного числа 3959_{10} определяется как $3 \cdot 1000 + 9 \cdot 100 + 5 \cdot 10 + 9 \cdot 1$, где 1000, 100, 10, 1 – соответственно, веса четвертого, третьего, второго, первого разрядов записи оцениваемого числа.

Десятичная система счисления является также системой с равномерно распределенными весами, которые характеризуются тем, что соотношение весов двух любых соседних разрядов имеют для такой системы одинаковое значение. Это соотношение называется основанием системы счисления, которое в дальнейшем будем обозначать как «*q*».

Общая запись числа в системе с равномерно распределенными весами имеет вид:

$$N_q = A_n\ A_{n-1} \ldots A_2\ A_1\ A_0. \qquad (1)$$

Значение такого числа определяется как:

$$N_q = A_n q^n + A_{n-1} q^{n-1} + A_{n-2} q^{n-2} + A_2 q^2 + A_1 q^1 + A_0 q^0, \qquad (2)$$

где A_i – цифра записи числа, удовлетворяющая условию $0 <= A_i <= (q - 1)$; q – основание системы счисления.

При $q = 10$ A изменяется в диапазоне от 0 до 9, т. е. до (10 – 1).

Запись числа N в виде (1) называется кодированной, а запись в форме (2) называется расширенной записью.

Помимо $q = 10$ (*десятичная система счисления*) возможны другие значения для основания системы счисления:

– *двоичная система счисления*;

– *восьмеричная система счисления*;

– *шестнадцатеричная система счисления* и т. д.

Для обозначения цифр в различных системах счисления в качестве цифр используются обозначения соответствующих цифр десятичной системы счисления – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, а в случае, когда десятичных цифр «не хватает» (для систем счисления с основанием q, большим чем 10), для цифр, превышающих 9, вводятся дополнительные обозначения, например, для $q = 16$ это будут обозначения A, B, C, D, E, F, которые соответствуют шестнадцатеричным цифрам, десятичные эквиваленты которых равны, соответственно, 10, 11, 12, 13, 14, 15.

Примеры записи чисел в различных системах счисления:

$N_2 = 10011011 = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$;

$$N_8 = 471025 = 4 \cdot 8^5 + 7 \cdot 8^4 + 1 \cdot 8^3 + 0 \cdot 8^2 + 2 \cdot 8^1 + 5 \cdot 8^0;$$

$$N_{16} = 84FE4A = 8 \cdot 16^5 + 4 \cdot 16^4 + F \cdot 16^3 + E \cdot 16^2 + 4 \cdot 16^1 + A \cdot 16^0;$$

$$N_{10} = 35491 = 3 \cdot 10^4 + 5 \cdot 10^3 + 4 \cdot 10^2 + 9 \cdot 10^1 + 1 \cdot 10^0.$$

На основании вышеизложенного можно заключить, что запись одного и того же числа в различных системах счисления будет тем длиннее, чем меньше основание системы счисления. Например, число N, десятичное значение которого равно 2063, в различных системах счисления представляется как $N = 2063_{10} = 100000001111_2 = 4017_8 = 80F_{16}$.

Человек в своей практической деятельности наиболее часто использует *десятичную систему счисления*. *Двоичная система счисления* является удобной для обработки информации в ЭВМ. Промежуточное место между этими системами занимает *двоично-десятичная система счисления*. Эта система в принципе является десятичной, но отдельные десятичные цифры в ней записываются в виде набора двоичных разрядов. Существуют различные двоично-десятичные системы, которые отличаются способом представления набора двоичных разрядов десятичных цифр. Наиболее широкое распространение получила двоично-десятичная система 8, 4, 2, 1. Данная система характеризуется тем, что отдельные десятичные цифры в ней представляются их четырехбитовым двоичным эквивалентом, как это показано в табл. 1.

Таблица 1

Представление десятичных цифр эквивалентами

№ п/п	$q = 2$	$q = 8$	$q = 16$	$q = 10$	Десятичный эквивалент	Двоичный эквивалент
1	0	0	0	0	0	0000
2	1	1	1	1	1	0001
3		2	2	2	2	0010
4		3	3	3	3	0011
5		4	4	4	4	0100
6		5	5	5	5	0101
7		6	6	6	6	0110
8		7	7	7	7	0111
9			8	8	8	1000
10			9	9	9	1001
11			*A*		10	1010
12			*B*		11	1011

Окончание табл. 1

№ п/п	$q = 2$	$q = 8$	$q = 16$	$q = 10$	Десятичный эквивалент	Двоичный эквивалент
13			*C*		12	1100
14			*D*		13	1101
15			*E*		14	1110
16			*F*		15	1111

Например, десятичное число 804714 в двоично-десятичной системе 8, 4, 2, 1 представляется в виде 1000 0000 0100 0111 0001 0100.

В дальнейшем для сокращения будем использовать название «двоично-десятичная система», имея в виду двоично-десятичную систему 8, 4, 2, 1.

2. ПЕРЕВОД ЧИСЕЛ ИЗ ОДНОЙ СИСТЕМЫ СЧИСЛЕНИЯ В ДРУГУЮ

Наличие различных систем счисления предполагает использование разных способов перевода записи числа из одной системы счисления в другую. Для этой цели применяются следующие способы преобразований:

– метод преобразования с использованием весов разрядов в исходной и в искомой записи числа;

– метод деления (умножения) на новое основание;

– метод с использованием особого соотношения заданной и искомой систем счисления.

2.1. Метод преобразования с использованием весов разрядов

Метод преобразования с использованием весов разрядов записи числа в исходной и в искомой системах предполагает использование расширенной записи числа (2) в некоторой системе счисления.

Метод имеет две разновидности в зависимости от того, какая система счисления (исходная или искомая) является более привычной. Если более привычной является искомая система счисления, то на основании расширенной записи исходного числа подсчитываются значения ее отдельных разрядов в новой системе счисления. Далее полученные значения суммируются.

Например, при преобразовании целого двоичного числа N_2 = 110011010 в десятичную систему счисления исходное число представляется в расширенной записи $N = 2^8 + 2^7 + 2^4 + 2^3 + 2^1$ и рассчитывается вес отдельных (ненулевых) двоичных разрядов в десятичной системе счисления: 256, 128, 16, 8, 2.

Затем искомая запись числа определяется как сумма весов всех ненулевых разрядов записи числа в заданной системе счисления: 256 + 128 + 16 + 8 + 2 = 410.

При преобразовании правильных дробей в принципе используется тот же подход, но при расчете весов отдельных разрядов берутся отрицательные степени основания счисления.

Кроме того, учитывая, что при преобразовании правильных дробей в общем случае результат получается неточный, перед началом преобразования необходимо подсчитать количество разрядов представления числа в новой системе счисления. Разрядность результата выбирается таким образом, чтобы ошибка представления результата была бы не более половины единицы младшего разряда в исходной записи числа.

Например, при использовании двоичной и десятичной систем счисления берется соотношение, согласно которому один десятичный разряд соответствует точности представления четырехразрядного двоичного числа.

При преобразовании правильных дробей сначала ищется предварительное значение представления заданного числа в новой системе счисления с количеством разрядов, на единицу большим, чем расчетная разрядность представления числа в новой системе счисления. Дополнительный разряд в предварительном результате преобразования используется для округления, позволяющего с рассчитанным числом разрядов найти окончательный результат.

Пример. Представить правильную двоичную дробь 0,101 в десятичной системе счисления.

Перед началом преобразования определяется, что разрядность записи заданного числа в новой системе счисления должна быть равна 1, поэтому сначала ищется предварительная запись заданного числа в новой системе счисления с двумя десятичными разрядами $0{,}101_2 = 2^{-1} + 2^{-3} = 0{,}5 + 0{,}13 = 0{,}63$, и после округления $0{,}101_2 = 0{,}6_{10}$.

Если более привычной является исходная система счисления, то запись заданного числа в новой системе счисления определяется разряд

за разрядом, начиная со старшего. Первым значащим разрядом будет являться разряд с максимально возможным весом, но не превышающим значения преобразуемого числа. При этом, определив старшую цифру (старший разряд) с ненулевым значением, из исходного числа вычитаем вес этого разряда, таким образом формируя остаток, который должен быть представлен еще не найденным младшим разрядом искомой записи числа в новой системе счисления. Далее, используя полученный остаток, аналогичным приемом ищем второй старший разряд записи числа в новой системе счисления, определяем новый остаток и переходим к определению следующего разряда и т. п. Процесс этот напоминает процедуру взвешивания некоторого тела посредством уравновешивания его веса с помощью эталонных гирь.

Пример. Найти двоичный эквивалент десятичного числа: 43610 = ___ ? ___ 2.

Решение

Первый (старший) разряд, имеющий значение 1 в искомой двоичной записи числа, будет разряд весом $2^8 = 256$. С помощью остальных (младших) разрядов искомой записи числа необходимо представить значение 180 (180 – остаток, полученный как 436 – 256).

Второй разряд с весом $2^7 = 128$ будет иметь в искомой двоичной записи числа значение 1. С помощью остальных (более младших) разрядов искомой записи числа необходимо представить значение 52 (52 – остаток, полученный как 188 – 128).

Третий разряд с весом $2^6 = 64$ будет иметь в искомой двоичной записи числа значение 0.

Четвертый разряд с весом $2^5 = 32$ будет иметь в искомой двоичной записи числа значение 1, а остаток – 20.

Пятый разряд с весом $2^4 = 16$ будет иметь в искомой двоичной записи числа значение 1, а остаток – 4.

Шестой разряд с весом $2^3 = 8$ будет иметь в искомой двоичной записи числа значение 0.

Седьмой разряд с весом $2^2 = 4$ будет иметь в искомой двоичной записи числа значение 1, а остаток – 0.

Восьмой разряд с весом $2^1 = 2$ будет иметь в искомой двоичной записи числа значение 0.

Девятый разряд с весом $2^0 = 1$ будет иметь в искомой двоичной записи числа значение 0.

Таким образом, $436_{10} = 110110100_2$.

Пример. Найти двоичный эквивалент числа: $0{,}7_{10}$ = ___ ? ___ 2.

Предварительный результат ищется с точностью до пяти двоичных разрядов, причем пятый разряд используется только для округления при переходе к четырехразрядному окончательному результату.

Первый (старший) разряд с весом $2^{-1} = 0{,}5$ в искомой двоичной записи числа будет иметь значение 1. С помощью остальных (младших) разрядов искомой записи числа необходимо представить значение 0,2 (0,2 – остаток, полученный как 0,7 – 0,5 = 0,2).

Второй (старший) разряд с весом $2^{-2} = 0{,}25$ в искомой двоичной записи числа будет иметь значение 0.

Третий разряд с весом $2^{-3} = 0{,}13$ в искомой двоичной записи числа будет иметь значение 1. С помощью остальных (более младших) разрядов искомой записи числа необходимо представить значение 0,07 (0,07 – остаток, полученный как 0,2 – 0,13).

Четвертый разряд с весом $2^{-4} = 0{,}06$ в искомой двоичной записи числа будет иметь значение 1, а остаток – 0,01.

Пятый разряд с весом $2^{-5} = 0{,}03$ искомой двоичной записи числа будет иметь значение 0.

Таким образом, десятичное число $0{,}7_{10} = 0{,}10110_2$.

После округления имеет место $0{,}7_{10} = 0{,}1011_2$.

2.2. Метод деления (умножения) на новое основание

Метод деления (умножения) имеет две разновидности, соответственно, для преобразования целых и дробных чисел.

Преобразование целых чисел

Задачу представления числа N, заданного в системе q_i, в системе счисления с основанием q_2 можно рассматривать как задачу поисков коэффициентов полинома, представляющего собой расширенную запись числа N в системе счисления q_2.

Отсюда вытекает правило формирования коэффициентов полинома или разрядов записи заданного числа N в системе счисления с основанием q_2:

– необходимо разделить исходное число N_{q_1} на новое основание q_2, при этом получив целое частное и остаток;

– полученный остаток снова необходимо разделить на q_2, процесс деления продолжается до тех пор, пока частное будет не меньше нового основания q_2. Если очередное сформированное частное будет

меньше, чем q_2, то процесс формирования записи заданного числа в новой системе с основанием q_2 считается законченным, а в качестве искомых разрядов новой записи числа используются результаты выполненных операций деления следующим образом. В качестве старшего разряда берется значение последнего частного, для остальных разрядов используются значения остатков в порядке, обратном порядку их получения.

Пример. Найти запись в двоичной форме десятичного числа $N_{10} = 436$.

Решение

Делим сначала исходное число N_{10}, а затем получаемые частные на значение нового основания 2 до получения частного со значением меньше, чем два:

436/2 – > int(436/2) = 218 и rest(436/2) = 0;

218/2 – > int(218/2) = 109 и rest(218/2) = 0;

109/2 – > int(109/2) = 54 и rest(109/2) = 1;

54/2 – > int(54/2) = 27и rest(54/2) = 0;

27/2 – > int(27/2) = 13 и rest(27/2) = 1;

13/2 – > int(13/2) = 6 и rest(13/2) = 1;

6/2 – > int(6/2) = 3 и rest(6/2) = 0;

3/2 – > int(3/2) = 1 и rest(3/2) = 1.

Таким образом, 436 = 110110100.

Преобразование дробных чисел

Задачу представления дробного числа M_{q_1}, заданного в системе q_1, в системе счисления с основанием q_2, можно рассматривать как задачу поиска коэффициентов полинома, представляющего собой расширенную запись числа M в системе счисления q_2.

Отсюда вытекает следующее правило формирования коэффициентов полинома, которые одновременно являются разрядами записи заданного числа M в системе счисления с основанием q_2:

– определяется количество разрядов «n» в записи числа M_{q_2} в новой системе счисления;

– исходное число M_{q_i} умножается на q_2, при этом будет получено смешанное число;

– дробная часть полученного произведения снова умножается на q_2 и т. д.; процесс умножения повторяется n раз. В качестве искомых разрядов новой записи числа используются результаты выполненных операций деления следующим образом: в качестве первого старшего разряда искомой записи числа в новом основании берется значение целой части первого произведения, в качестве второго старшего разряда искомой записи числа в новом основании берется значение целой части второго произведения и т. д.

Пример. Найти запись в двоичной форме десятичного числа $M_{10} = 0{,}7$.

Определяем количество разрядов числа M_2. Так как исходная запись числа содержит один десятичный разряд, то запись данного числа в двоичном основании должна содержать четыре разряда. Учитывая округление, ищем предварительный двоичный эквивалент с пятью разрядами.

Умножаем исходное число M_{10}, а затем дробные части последовательно получаемых произведений на новое основание 2. Выполняется пять таких операций умножения, в результате получаем:

$0{,}7 \cdot 2 = 1{,}4$(int$(0{,}7 \cdot 2) = 1$ и DF(rest$(0{,}7 \cdot 2) = 0{,}4$);

$0{,}4 \cdot 2 = 0{,}8$(int$(0{,}4 \cdot 2) = 0$ и DF(rest $(0{,}4 \cdot 2) = 0{,}8$);

$0{,}8 \cdot 2 = 1{,}6$(int$(0{,}8 \cdot 2) = 1$ и DF(rest $(0{,}8 \cdot 2) = 0{,}6$);

$0{,}6 \cdot 2 = 1{,}2$(int$(0{,}6 \cdot 2) = 1$ и DF(rest $(0{,}6 \cdot 2) = 0{,}2$);

$0{,}2 \cdot 2 = 0{,}4$(int$(0{,}2 \cdot 2) = 0$ и DF(rest $(0{,}2 \cdot 2) = 0{,}4$).

Таким образом, 0,7 = 0,10110, а окончательный результат перехода в двоичную систему будет $0{,}7_{10} = 0{,}1011_2$.

2.3. Метод с использованием особого соотношения оснований заданной и искомой систем счисления

Данный метод применим тогда, когда исходное q_1 и новое q_2 основания могут быть связаны через целую степень, т. е. когда выполняется условие $q_1^m = q_2$ (условие 1) или $q_2^m = q_1$ (условие 2).

Если имеет место *условие* 2, то для преобразования заданного числа запись его в новом основании q_2 ищется следующим образом:

$$N = a_n a_{n-1} a_{n-2} \ldots a_1 a_0. \qquad (3)$$

Каждому разряду a_i исходной записи числа ставится в соответствие его m-разрядный эквивалент в системе счисления с основанием q_2.

Исходная запись всего заданного числа формируется за счет объединения всех полученных m-разрядных групп.

Если имеет место *условие* 1, то запись заданного числа (3) в новом основании q_2 ищется следующим образом:

– исходная запись числа разбивается на группы по m разрядов, двигаясь от точки вправо и влево (недостающие разряды в крайних группах (слева и справа) дополняются нулями;

– каждой полученной группе ставится в соответствие цифра новой системы счисления;

– искомая запись заданного числа в новой системе счисления образуется из цифр, соответствующих группам, на которые была разбита исходная запись.

Пример 1. Найти двоичный эквивалент восьмеричного числа 67401,648.

Решение

Основания исходной и новой систем счисления можно выразить через целую степень $2^3 = 8$.

Поэтому применяем третий метод для случая перехода из системы с большим основанием в систему с меньшим основанием. Ставим в соответствие каждой цифре исходной записи числа трехразрядный двоичный эквивалент:

6	7	4	0	1	6	4
110	111	100	000	001	110	100.

Формируем окончательный результат посредством объединения полученных трехразрядных двоичных чисел в единый двоичный эквивалент:

$$67401{,}64_8 = 110111100000001{,}110100.$$

Пример 2. Найти шестнадцатеричный эквивалент двоичного числа

$$N = 11100101110110{,}111011001_2.$$

Решение

Основания исходной и новой систем счисления можно выразить через целую степень: $2^4 = 16$.

Поэтому применяем третий метод для случая перехода из системы с меньшим основанием в систему с большим основанием. Разбиваем исходную запись числа на группы по четыре разряда вправо и влево от

точки, в крайних левой и правой группах недостающие разряды заполняем нулями и каждой полученной группе из четырех разрядов ставим в соответствие цифру шестнадцатеричной системы счисления:

0011	1001	0111	0110	1110	1100	1000
3	9	7	6	*E*	*C*	8.

Формируем окончательный результат посредством объединения полученных цифр в единый шестнадцатеричный эквивалент $11100101110110,111011001_2 = 3\ 9\ 7\ 6,EC8_{16}$.

Пример 3. Найти шестнадцатеричный эквивалент числа $67401,64_8$, представленного в восьмеричной системе счисления.

Решение

Основания исходной q_1 и новой q_2 систем счисления не могут быть связаны через целую степень, поэтому напрямую третий метод перехода неприменим. Однако существует система с двоичным основанием, для которой допустим третий метод перехода в восьмеричную (исходную для данного примера) и в шестнадцатеричную (новую систему для данного примера) системы счисления, так как $2^3 = 8$ и $2^4 = 16$.

Поэтому в данном случае для решения поставленной задачи целесообразно использовать два быстрых перехода из восьмеричной системы счисления в двоичную (промежуточную), а затем из двоичной системы счисления в шестнадцатеричную третьим методом. Это будет гораздо быстрее, чем использовать для заданного преобразования второй или третий метод.

Таким образом, поставленная задача решается следующими действиями: $67401,64_8 = 110111100000001,110100_2$; $111100000001,110100_2 = 0110111100000001,11010000_2 = 6F01,D0_{16}$, т. е. $67401,64_8 = 6F01,D0_{16}$.

Пример 4. Найти двоичный эквивалент числа 6740_{10}.

Решение

Основания исходной q_1 и новой q_2 систем счисления не могут быть связаны через целую степень, поэтому третий метод перехода неприменим. В принципе здесь целесообразно использовать второй метод – метод деления на новое основание. Однако в этом случае потребуется большое количество операций деления на два. Для сокращения количества операций деления может оказаться целесообразным решить эту задачу за счет перехода с использованием второго метода в промежуточную шестнадцатеричную систему счисления,

а затем, используя третий метод, быстро перейти в заданную двоичную систему счисления:

– выполняем переход в промежуточную систему счисления:

6740/16 = 421 (остаток 4);

421/16 = 26 (остаток 5);

26/16 = 1 (остаток 10);

– для промежуточной системы счисления имеем:

$6740_{10} = 1A54_{16}$;

– выполняем переход из промежуточной системы счисления в заданную: $1A54_{16} = 0001101001010100_2$.

Таким образом, для заданного перехода потребовалось выполнить только 4 операции деления на 16 вместо 13-ти операций деления на 2.

3. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ НАД ПОЛОЖИТЕЛЬНЫМИ ЧИСЛАМИ

3.1. Операции сложения в двоичной системе счисления

При выполнении любой операции результат ищется согласно соответствующим правилам, которые удобно представлять в табличной форме, где для всех возможных комбинаций значений одноразрядных операндов приводятся значения результата.

Правила выполнения операции сложения в двоичной системе счисления задаются в виде табл. 2.

Таблица 2

Таблица истинности

+	0	1
0	0	1
1	1	0*

Все возможные значения первого слагаемого задаются во второй и третьей строках первой колонки. Все возможные значения второго слагаемого задаются во второй и третьей колонках первой строки. На пересечении отмеченных значениями операндов строк и колонок располагается результат их сложения. В таблице знаком *

отмечен случай, когда в текущем разряде результата получен ноль и имеет место перенос в ближайший старший разряд.

Пример. 1011100110+0101101101 = 10001010011.

В общем случае при формировании значения в текущем разряде результата приходится дважды применять приведенную таблицу сложения: первый раз – при сложении соответствующих разрядов операндов, формируя так называемую поразрядную сумму, и второй раз – при сложении разряда сформированной поразрядной суммы и переноса, пришедшего из ближайшего младшего разряда.

При машинной реализации операции сложения сначала формируется поразрядная сумма операндов без учета переноса, далее формируется код переноса и затем с помощью специальных логических цепей учитываются возникшие переносы. При этом перенос, возникший в некотором разряде, может изменить не только ближайший старший разряд, но и целую группу старших разрядов. В худшем случае перенос, возникший в самом младшем разряде, может изменить значение старших разрядов сформированной поразрядной суммы, вплоть до самого старшего.

При формировании поразрядной суммы и учете возникших переносов используется следующая классификация разрядов складываемых операндов:

– разряд, генерирующий перенос (оба операнда в этом разряде имеют 1);

– разряд, пропускающий перенос (операнды в этом разряде имеют разные значения);

– разряд, блокирующий распространение переноса (операнды в этом разряде имеют одинаковые значения).

3.2. Операция вычитания

Правила выполнения операции вычитания в двоичной системе счисления задаются в виде табл. 3.

Таблица 3

Таблица истинности

–	0	1
0	0	1
1	1*	0

Все возможные значения вычитаемого задаются во второй и третьей строках первой колонки. Все возможные значения уменьшаемого задаются во второй и третьей колонках первой строки. На пересечении отмеченных значениями операндов строк и колонок располагается результат вычитания второго операнда из первого операнда. В табл. 4 знаком * отмечен случай, когда в текущем разряде результата получена единица при заеме из ближайшего старшего разряда.

Пример. 1000111001 – 0101101101 = 0011001100.

Как видно из таблицы и приведенного примера, реализация операция вычитания не сложнее операции сложения.

В ЭВМ никогда в перечне выполняемых операций арифметического устройства не присутствует одновременно операция сложения и операция вычитания. При этом, как правило, присутствует только операция сложения. Что же касается операции вычитания, то она реализуется за счет прибавления к уменьшаемому значения вычитаемого, взятого с противоположным знаком.

3.3. Операция умножения

Правила умножения в двоичной системе счисления задаются в виде табл. 4.

Таблица 4

Таблица истинности

*	0	1
0	0	0
1	0	1

Все возможные значения множимого задаются во второй и третьей строках первой колонки. Все возможные значения множителя задаются во второй и третьей колонках первой строки. На пересечении отмеченных значениями операндов строк и колонок располагается результат умножения первого операнда на второй операнд.

При умножении многоразрядных операндов как правило (особенно в десятичной системе счисления) используется метод, при котором формирование произведения выполняется за счет суммирования частичных произведений, которые формируются посредством умножения множимого на отдельные разряды множителя с учетом веса соответствующего разряда множителя.

Таблица умножения одноразрядных операндов в двоичной системе существенно упрощает задачу формирования частичного произведения: частичное произведение для разряда множителя равняется нулю, если этот разряд равен нулю; частичное произведение для разряда множителя равняется множимому, взятому с соответствующим весом, если разряд множителя равен единице.

При последовательном способе формирования частичных произведений последние могут рассчитываться поочередно для отдельных разрядов множителя, начиная с младшего или старшего разряда. При десятичном основании, как правило, формирование частичных произведений осуществляется, начиная с младшего разряда множителя.

Пример 1. Найти произведение двоичных чисел 1011 и 1101, начиная формирование частичных произведений со старшего разряда множителя.

Решение

При формировании частичных произведений, начиная со старшего разряда множителя, процесс формирования произведения заданных операндов можно представить следующим образом:

1 0 1
1 10 1
+ 10 11
+ 10 11
+ 0000
+ 10 11
10001111.

При реализации умножения рассматриваемым способом требуется использование $2n$-разрядного сумматора для подсчета промежуточных и конечного произведения и $2n$-разрядного сдвигающего регистра для хранения множителя.

3.4. Деление двоичных чисел

Деление является неточной операцией, поэтому при ее выполнении прежде всего устанавливается количество разрядов частного, которые подлежат определению.

Деление в двоичной системе счисления может выполняться точно так же, как и в десятичной, однако формирование частного двоичных операндов реализуется гораздо проще, чем в десятичной системе, так как:

– упрощается процедура подбора очередной цифры вследствие того, что в двоичной системе очередной цифрой может быть одна из двух – либо 0, либо 1;

– упрощается процедура умножения найденной цифры частного на делитель.

4. АРИФМЕТИКА С ПОЛОЖИТЕЛЬНЫМИ ДВОИЧНО-ДЕСЯТИЧНЫМИ ЧИСЛАМИ

В ЭВМ часто предусматривается обработка чисел не только в двоичной системе счисления, но и в двоично-десятичной. При этом как правило стремятся реализовать двоично-десятичную арифметику по правилам двоичной с введением ограниченного количества коррекций.

Сложение двоично-десятичных чисел

Рассмотрим на конкретном примере реализацию этой операции.

Пример. Найти сумму двух десятичных чисел с использованием двоично-десятичной системы счисления: $A = D + C$, где $D = 3927$; $C = 856$.

Решение

Составляем двоично-десятичную запись для чисел D и C:

$D = 3927 = 0011100100100111$;

$C = 4856 = 0100100001010110$.

Найти значение A можно, реализовав следующую последовательность операций из двоичного сложения и операции коррекции:

0011100100100111 – D

+ 01001000 0101 0110 – C

1000000101111101 – двоичная сумма

+01100110 – коррекция

1000011110000011 – двоично-десятичная сумма.

Для получения двоично-десятичной суммы A на основании результата сложения операндов по правилам двоичной арифметики необходимо добавить шестерку (0110) в те тетрады, из которых был перенос. В данном примере это вторая тетрада (отмечена *). Необходимость такой коррекции обусловливается тем, что перенос, сформированный по правилам двоичного суммирования, унес из тетрады шестнадцать, а для десятичного сложения перенос должен был унести десять, т. е. перенос, сформированный по правилам двоичной арифметики, унес лишнюю шестерку. Кроме этого, шестерка добавляется в те тетрады, в которых получено значение, большее девяти.

5. АРИФМЕТИКА С АЛГЕБРАИЧЕСКИМИ ЧИСЛАМИ

5.1. Кодирование алгебраических чисел

Для представления чисел со знаком используются специальные коды:

– *прямой код*;

– *дополнительный код*;

– *обратный код*.

Во всех трех случаях используется следующий формат представления числа, содержащий два поля – поле знака и поле модуля (рис. 1).

Поле знака	Поле модуля

Рис. 1. Формат представления числа

Поле знака представлено одним разрядом, в котором устанавливается 0, если число положительное, и 1, если число отрицательное.

Поле модуля отражает количественную оценку числа и для каждого кода формируется по-разному. Количество разрядов поля модуля определятся диапазоном изменения отображаемых чисел или точностью их представления.

В прямой код запись целого числа A формируется по следующему правилу:

$$A_{\text{пр}} = \begin{cases} 0, A, & \text{если } A >= 0; \\ 1, |A|, & \text{если } A < 0. \end{cases} \qquad (4)$$

В дополнительном коде запись целого числа A формируется по следующему правилу:

$$A_{\text{дк}} = \begin{cases} 0, A, & \text{если } A >= 0; \\ 1, q^n + A, & \text{если } A < 0, \end{cases} \qquad (5)$$

где n – разрядность модульного поля; q – основание системы счисления; q^n – максимальная невключенная граница диапазона изменения представляемых чисел, так как диапазон изменения чисел A определяется как $q^n > |A| >= 0$.

Для случая правильной дроби запись числа A в дополнительном коде имеет вид:

$$A_{\text{дк}} = \begin{cases} 0, A, \text{ если } A >= 0; \\ 1, (1 + A), \text{ если } A < 0, \end{cases} \quad (6)$$

где 1 – максимальная невключенная граница диапазона изменения представляемых чисел, т. е. диапазон изменения чисел A определяется как $1 > |A| >= 0$.

В обратном коде запись целого числа A формируется по следующему правилу:

$$A_{\text{ок}} = \begin{cases} 0, A, \text{ если } A >= 0; \\ 1, ((q^n - 1) + A), \text{ если } A < 0, \end{cases} \quad (7)$$

где n – разрядность модульного поля; q – основание системы счисления; $(2^n - 1)$ – максимальная включенная граница диапазона изменения представляемых чисел, т. е. диапазон изменения чисел A определяется как $(q^n - 1) >= |A| >= 0$.

Для случая правильной дроби запись числа B в обратном коде имеет вид:

$$A_{\text{ок}} = \begin{cases} 0, A, \text{ если } A >= 0; \\ 1, (1 - q^{-n} + A), \text{ если } A < 0, \end{cases} \quad (8)$$

где $(1 - q^{-n})$ – максимальная включенная граница диапазона изменения представляемых чисел, т. е. диапазон изменения чисел A определяется как $(1 - q^n) >= |A| >= 0$.

Легко показать, что перевод отрицательного числа из обратного или дополнительного кода в прямой выполняется по тому же правилу, что и перевод числа из прямого кода в обратный или *дополнительный код*: для того чтобы перевести отрицательное число из обратного в *прямой код*, необходимо дополнить его модуль до включенной границы; для того чтобы перевести отрицательное число из дополнительного в *прямой код*, необходимо дополнить его модуль до невключенной границы.

5.2. Логические операции с двоичными кодами

Над двоичными кодами могут выполняться различные логические операции, среди которых особое место занимают:

– *логическое суммирование* (обозначения – ИЛИ, OR);

– *логическое умножение* (обозначения – И, AND);

– *логическое отрицание* (обозначения – НЕТ, NOT, т. е. штрих над отрицаемым кодом *x*);

– *суммирование по модулю* 2 (обозначается *mod* 2);

– операции *сдвига*.

Операция *логического суммирования* выполняется над двумя кодами и генерирует код той же разрядности, что и операнд, у которого в некотором *i*-м разряде находится единица, если хотя бы в одном операнде в *i*-м разряде имеет место единица.

Пример. 10001101 + 11110000 = 11111101.

Операция *логического умножения* выполняется над двумя кодами и генерирует код той же разрядности, что и операнд, у которого в некотором *i*-м разряде находится единица, если оба операнда в этом *i*-м разряде имеют единицу, и ноль – во всех других случаях.

Пример. 10001101 + 11110000 = 10000000.

Операция *суммирования по модулю* 2 выполняется над двумя кодами и генерирует код той же разрядности, что и операнд, у которого в некотором *i*-м разряде находится единица, если два заданных операнда в *i*-м разряде имеют противоположные значения. Иногда эта операция называется «исключающее ИЛИ».

Пример. 10001101 + 11110000 = 01111101.

Операция *логического отрицания* выполняется над одним кодом и генерирует результирующий код той же разрядности, что и операнд, у которого в некотором *i*-м разряде находится значение, противоположное значению в *i*-м разряде отрицаемого кода.

Операции *сдвига* в свою очередь подразделяются на логические сдвиги, в своих разновидностях – сдвиг вправо, сдвиг влево, циклический сдвиг вправо, циклический сдвиг влево; арифметический сдвиг вправо и влево, выполнение которого зависит от знака и кода сдвигаемого числа.

Логические сдвиги

Сдвиг *влево* выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем младшем разряде (освобождающийся самый правый, т. е. самый младший, разряд заполняется 0, а «выталкиваемый» разряд пропадает). Например, код 11001110 после сдвига влево будет иметь вид 10011100.

Сдвиг *вправо* выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем старшем раз-

ряде (в освобождающийся самый левый, т. е. самый старший, разряд заполняется 0, «выталкиваемый» разряд пропадает). Например, код 11001110 после сдвига влево будет иметь вид 01100111.

Циклический сдвиг влево – выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем младшем разряде (в освобождающийся самый правый, т. е. самый младший, разряд заносится значение старшего, т. е. самого левого разряда исходного кода). Например, код 11001110 после сдвига влево будет иметь вид 10011101.

Циклический сдвиг вправо – выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем старшем разряде (в освобождающийся самый левый, т. е. самый старший, разряд заполняется значение в самом младшем разряде исходного кода). Например, код 11001110 после сдвига влево будет иметь вид 01100111.

Арифметический сдвиг

Арифметические сдвиги обеспечивают выполнение умножения (сдвиги влево) или операции деления (сдвиги вправо) двоичных кодов на два точно так же, как сдвиги вправо и влево десятичного числа обеспечивают выполнение деления и умножение на 10.

Арифметические сдвиги влево двоичного прямого кода выполняются в зависимости от того, какое сдвигается число – положительное или отрицательное.

Если *сдвигается положительное* число, то сдвиг (вправо или влево) выполняется как соответствующий логический сдвиг (влево или вправо), с той лишь разницей, что предусматриваются средства определения факта переполнения при сдвиге влево, что реализуется и при всех других арифметических операциях. При любом сдвиге вправо предусматриваются средства для округления после завершения нужного количества сдвигов и средства обнаружения обнуления сдвигаемой величины после очередного сдвига.

6. ПРЕДСТАВЛЕНИЕ ЧИСЕЛ С ФИКСИРОВАННОЙ ТОЧКОЙ

Числовая информация представляется в машине в форме с фиксированной или с плавающей точкой. При представлении с фиксированной точкой положение последней в записи числа фиксировано.

Как правило, при использовании фиксированной точки числа представляются в виде целого числа или правильной дроби, форматы которых приведены на рис. 2.

Номер знака	1 разряд	2 разряд	3 разряд	4 разряд	…	(n–1) разряд	Послед-ний раз-	«.»

Номер знака	«.»	1 разряд	2 разряд	3 разряд	4 разряд	…	(n–1) разряд	Послед-ний разряд

Рис. 2. Формат целого и дробного числа

К заданному виду (целым числам или правильной дроби) исходные числа приводятся за счет введения масштабных коэффициентов.

Точка в записи числа не отображается, а так как она находится всегда в одном месте, то указание на ее положение в записи числа отсутствует. При n-разрядном представлении модульной части форма с фиксированной точкой обеспечивает диапазон изменения абсолютного значения числа A, для которого выполняется неравенство $2^n > |A| >= 0$.

Одним из важнейших параметров представления чисел является ошибка представления. Ошибка представления может быть абсолютной (Δ) или относительной (δ). Для фиксированной точки максимальные значения этих ошибок определяются следующим образом.

В случае целых чисел: $\Delta_{max} = 0{,}5$; $\delta max = \Delta_{max}/A_{min} = 0{,}5$, где A_{min} – минимальное, отличное от нуля, значение числа.

В случае дробных чисел:

$$A = 0{,}5 \cdot 2^{-n} = 2^{-(n+1)};\ \delta = \Delta max / A_{min} = 2^{-(n+1)} / 2^{-n} = 0{,}5, \qquad (9)$$

т. е. в худшем случае относительная ошибка при фиксированной точке может достигать сравнительно большого значения – 50 %.

6.1. Арифметические операции над числами, представленными с фиксированной точкой

К числу основных арифметических операций, непосредственно реализуемых в ЭВМ, относятся операции сложения, умножения, деления. Остальные операции (например, возведение в степень, извлечение квадратного корня) реализуются программным способом.

Выполнение операций с числами, представленными с фиксированной точкой, рассмотрено в рамках материала по выполнению операций с алгебраическими числами (разд. 4).

Выполнение таких длинных операций, как умножение и деление, реализуется в два этапа:

– на первом этапе формируется знак искомого результата,

– на втором этапе, используя абсолютные значения операндов, ищем результат (произведение или частное), которому затем присваивается предварительно определенный знак.

Операнды, как правило, представлены в прямом коде, и знак результата, независимо от того, частное это или произведение, ищется за счет сложения по модулю 2 знаковых разрядов операндов. В результате этого знак результата положителен, если операнды имеют одинаковые знаки, или отрицательный, если операнды имеют разные знаки.

6.2. Деление с фиксированной точкой

Реализация второго этапа деления, т. е. формирование частного двоичных положительных чисел в ЭВМ, представленных правильной дробью, может быть выполнена двумя методами:

– деление с восстановлением остатка;

– деление без восстановления остатка.

7. ПРЕДСТАВЛЕНИЕ ЧИСЕЛ С ПЛАВАЮЩЕЙ ТОЧКОЙ

При представлении числа с *плавающей точкой* число в общем случае представляет собой смешанную дробь и имеет формат, приведенный на рис. 3.

1 разряд	2 разряд	3 разряд	…	*к*-й разряд	«.»	(*к*+1) разряд	(*к*+2) разряд	…	(*n*+1) разряд	Последний разряд

Рис. 3. Представление числа с плавающей точкой

Местоположение точки в записи числа может быть различным, а так как сама точка в записи числа не присутствует, то для однозначного задания числа необходима не только его запись, но и информа-

ция о том, где в записи числа располагается точка, отделяющая целую и дробную части.

Поэтому в случае с плавающей точкой число X представляется в виде двух частей:

– *мантисса* ($x_м$), отображающая запись числа, представляется в виде правильной дроби с форматом фиксированной точки;

– *порядок* ($x_п$), отображающий местоположения в этой записи положение точки, представляется в виде целого числа с форматом фиксированной точки. Количественная оценка числа X определяется как

$$X = q^{x_n} X_{\text{М}}, \qquad (10)$$

где q – основание счисления.

В ЭВМ числа с плавающей точкой представляются в так называемой нормализованной форме, при которой в прямом коде мантисса нормализованного числа в старшем разряде модуля имеет ненулевое значение, а для двоичной системы счисления – нормализованная мантисса должна иметь в старшем разряде модуля прямого кода значение 1, т. е. для двоичной системы мантисса должна удовлетворять неравенству $1 > |x_m| >= 0{,}5$.

7.1. Арифметика с плавающей точкой

Операция сложения

Операция сложения чисел предполагает наличие одинаковых масштабов складываемых величин. Для случая представления чисел с плавающей точкой это предполагает наличие одинаковых порядков у операндов, подлежащих суммированию. Поэтому при выполнении операции сложения чисел с плавающей точкой в общем случае должно быть реализовано три этапа:

– выравнивание порядков;

– сложение мантисс операндов, имеющих одинаковые порядки;

– определение нарушения нормализации и при необходимости ее устранение.

8. ПРЕДСТАВЛЕНИЕ ДАННЫХ В ЭВМ

Как правило, в качестве элементарной единицы информации для представления данных в машине используется байт, который обычно представляет восемь двоичных бит.

Можно выделить два основных вида данных:

– символьные данные;

– числовые данные.

Элементы данных, как правило, представляются в виде последовательности байт переменной длины, где на представление каждого символа отводится один *байт*. Исключение составляют десятичные числа, для которых может использоваться *упакованная форма*, при которой в одном байте располагается по две цифры, т. е. одна десятичная цифра в двоично-десятичной системе занимает четыре бита, т. е. тетраду (рис. 4).

5	4	8	6
байт	байт	байт	байт

5	4	8	6
тетрада	тетрада	тетрада	тетрада
байт		байт	

Рис. 4. Запись десятичного числа. Посимвольная запись десятичного числа 5486. Упакованная запись десятичного числа 5486

Для представления двоичного числа обычно используется ограниченный набор форматов, например, один, два, четыре байта.

Пример представления чисел с плавающей точкой в 2-байтном формате приведен на рис. 5.

Рис. 5. Представление чисел с плавающей точкой

При использовании 4-байтного формата вводимые дополнительные два байта как правило используются для расширения мантиссы представляемого числа.

9. ЛОГИЧЕСКИЕ ОСНОВЫ ЦИФРОВОЙ ТЕХНИКИ

9.1. Основные законы и следствия Булевой алгебры

Для описания логических операций используется математический аппарат, получивший название алгебры логики, или Булевой алгебры.

В алгебре логики рассматриваются переменные, которые могут принимать только два значения: 1 и 0.

Основные логические функции:

1. Логическое отрицание НЕ (инверсия). Обозначается в виде черточки над аргументом: $y = \bar{x}$. В качестве примера цепи, реализующей функцию НЕ, можно привести размыкающий контакт реле. При срабатывании реле цепь, в которую входит такой контакт, будет размыкаться.

Логическое умножение И (конъюнкция). Символически обозначается: $y = x1 \cdot x2 \cdot x3 \ldots$ или $y = x1 \wedge x2 \wedge x3 \wedge \ldots$.

В качестве примера электрической цепи, реализующей функцию И, можно привести соединение последовательно замыкающих контактов нескольких реле. Цепь в этом случае будет замкнута тогда и только тогда, когда сработают все реле.

Логическое сложение ИЛИ (дизъюнкция). Операция обозначается выражениями: $y = x1 + x2 + x3 \ldots$ либо $y = x1 \vee x2 \vee x3 \vee \ldots$.

В качестве примера электрической цепи, реализующей функцию ИЛИ, можно привести параллельное соединение замыкающих контактов нескольких реле. Цепь, в которую входят эти контакты, будет замкнута, если сработает хотя бы один контакт.

Основные законы алгебры логики:

– *переместительный*:

$ab = ba,\ a + b = b + a;$

– *сочетательный*:

$(a + b) + c = a + (b + c),\ a(bc) = (ab)c;$

– *распределительный*:

$a(b + c) = ab + ac,\ a + bc = (a + b)(a + c);$

– *закон поглощения*:

$a + ab = a,\ \bar{a} + \bar{a}\,\bar{b} = \bar{a};$

– *закон склеивания*:

$ab + a\bar{b} = a,\ \bar{a}\,\bar{b} + \bar{a}\,b = \bar{a};$

– *закон отрицания*, *или правило де Моргана*:

$\overline{a+b} = \bar{a}\,\bar{b},\ \overline{ab} = \bar{a} + \bar{b}.$

Правило де Моргана справедливо для любого числа переменных:

$\overline{a+b+c+\ldots z} = \bar{a}\,\bar{b}\,\bar{c}\ldots\bar{z};$

$\overline{a\,b\,c\ldots z} = a+b+c+\ldots+z.$

Для алгебра логики справедливы следующие соотношения:

1) $x \vee 0 = x;$
2) $x \vee x = x;$
3) $x \vee 1 = 1;$
4) $x \wedge 0 = 0;$
5) $x \wedge x = x;$
6) $x \wedge 1 = x.$

9.2. Минимизация логических функций с помощью алгебраических преобразований

Минимизация логических функций применяется при синтезе комбинационных логических цепей (КЛЦ). КЛЦ – это такие цепи, выходные сигналы которых не зависят от предыстории и однозначно определяются сигналами, поступающими на их входы в рассматриваемый момент времени.

Синтез КЛЦ проводят в следующей последовательности:

1. Составляется таблица истинности. Эта таблица показывает, чему равен выходной сигнал цепи при различных комбинациях входных сигналов.

2. Исходя из таблицы истинности, записывается логическая функция.

3. Логическая функция минимизируется и преобразуется к удобному виду для реализации на логических ячейках заданного типа.

Рассмотрим работу мажоритарной ячейки на 3 входа. Строим таблицу истинности (табл. 5).

Таблица 5

Таблица истинности

№	*X*1	*X*2	*X*3	*F*
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

$$F = \overline{X1} \cdot X2 \cdot X3 \vee X1 \cdot \overline{X2} \cdot X3 \vee X1 \cdot X2 \cdot \overline{X3} \vee X1 \cdot X2 \cdot X3 =$$

$$= (X1 \cdot X2 \cdot X3 \vee X1 \cdot X2 \cdot X3) \vee (X1 \cdot \overline{X2} \cdot X3 \vee X1 \cdot X2 \cdot X3) \vee$$

$$\vee (X1 \cdot X2 \cdot \overline{X3} \vee X1 \cdot X2 \cdot X3) =$$

$$= X2 \cdot X3(X1 \vee \overline{X1}) \vee X1 \cdot X3 \vee X1 \cdot X2 =$$

$$= X2 \cdot X3 \vee X1 \cdot X3 \vee X1 \cdot X2 = \overline{\overline{X2 \cdot X3} \cdot \overline{X1 \cdot X3} \cdot \overline{X1 \cdot X3}}.$$

Построим схему по полученному выражению:

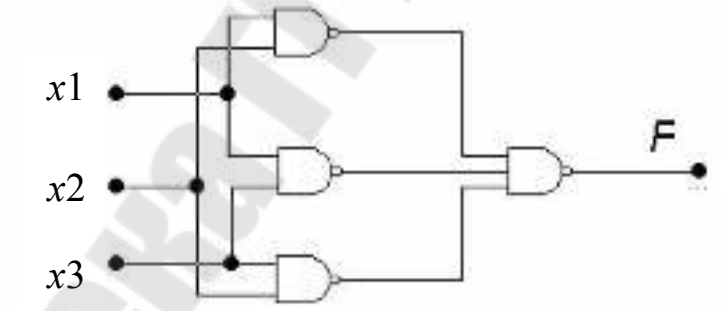

Рис. 6. Реализация мажоритарной ячейки на 3 входа
по минимизированному выражению

10. ОСНОВНЫЕ ПАРАМЕТРЫ ЦИФРОВЫХ СХЕМ

Основными параметрами цифровых интегральных схем являются быстродействие, потребляемая мощность, коэффициент объединения по входу, коэффициент разветвления по выходу, устойчивость против внешних воздействий, помехоустойчивость, степень интеграции, надежность, пороговое напряжение.

11. ЭЛЕМЕНТ И-НЕ В ДТЛ

Цифровые схемы могут быть построены по-разному, но в их основе, как правило, лежат схемы, выполняющие функции И-НЕ либо ИЛИ-НЕ. Поэтому интегральные схемы содержат обычно схемы И либо ИЛИ, выполненные на резисторах, диодах или транзисторах, и транзисторные инверторы. Транзисторный инвертор может быть простейшим – на одном транзисторе, включенном по схеме с общим эмиттером, или сложным – многотранзисторным с каскадным включением транзисторов в выходном каскаде.

Разберем работу схемы И-НЕ с ДТЛ, работающую от положительных сигналов (рис. 7). Схема состоит из двух частей. В первой входные переменные подаются на диодный элемент И. Вторая часть выполнена на транзисторе и представляет собой инвертор. Таким образом в схеме последовательно выполняется логическая операция И-НЕ. Диоды *VD*3, *VD*4 называются смещающими диодами и предназначены для надежного закрывания транзистора.

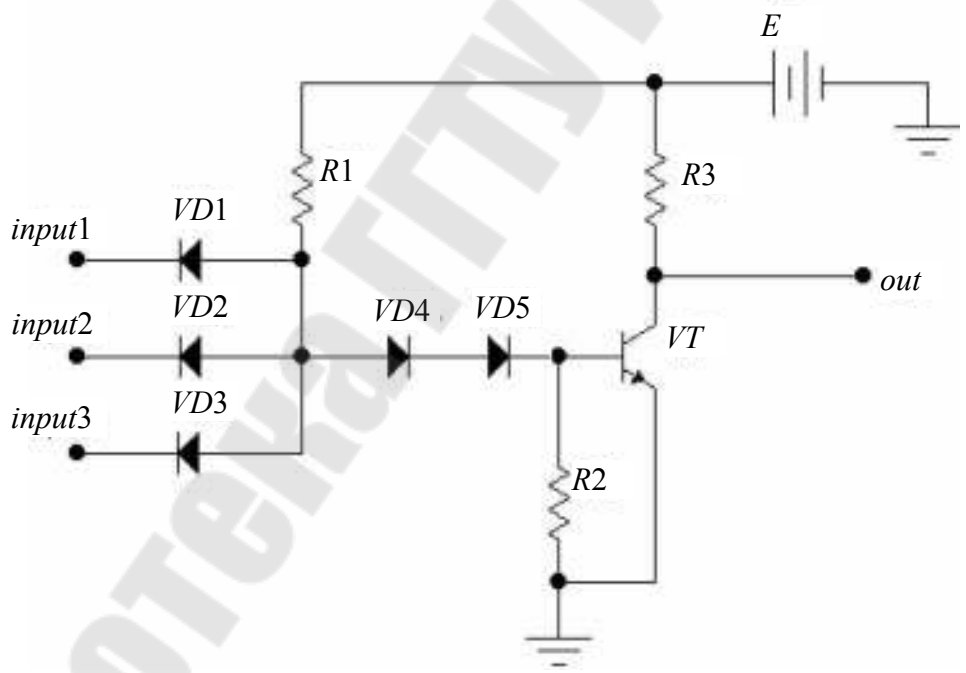

Рис. 7. Элемент И-НЕ в ДТЛ

ДТЛ-элементы обладают большим быстродействием, большим коэффициентом объединения по входу, высокой помехозащищенностью и широко используются в системах цифровой техники. Отсутствие конденсаторов и высокоомных резисторов делает их удобными для микроэлектронного исполнения. Однако ТТЛ-элементы получили

в настоящее время большее распространение. В первую очередь это связано с тем, что серьезным недостатком ДТЛ является большое количество диодов, каждый из которых необходимо тщательно изолировать, что увеличивает площадь микросхемы.

12. ЭЛЕМЕНТ И-НЕ В ТТЛ

По принципу действия, а также по основным характеристикам ТТЛ-элементы близки к ДТЛ.

Рассмотрим работу элемента ТТЛ, выполняющего логическую операцию И-НЕ. Роль изолированных диодов играет эмиттерный переход многоэмиттерного транзистора. Многоэмиттерный транзистор – чисто интегральный прибор, у которого область базы и область коллектора объединяют до 8 эмиттерных переходов. Принципиальная схема элемента И-НЕ на ТТЛ показана на рис. 8.

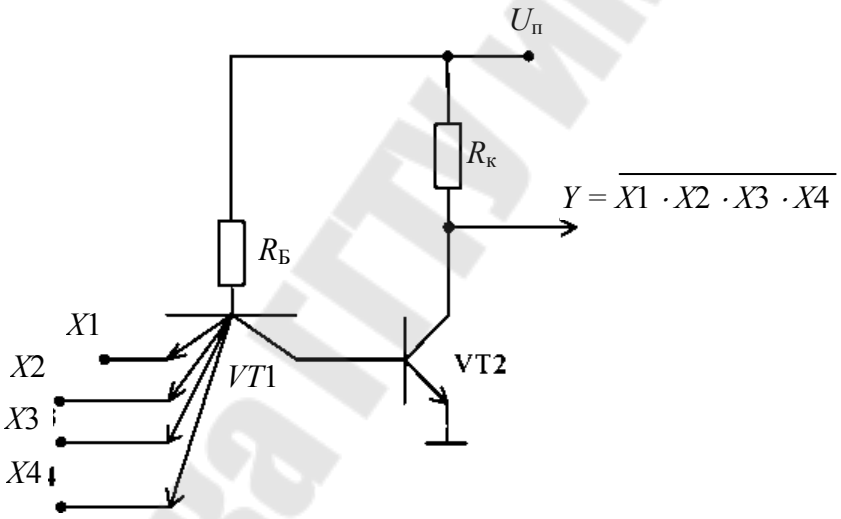

Рис. 8. Элемент И-НЕ в ТТЛ

Данная схема является прототипом трехвходового логического элемента ТТЛ серии К155. Типичные значения параметров транзисторно-транзисторной логики следующие: Напряжение питания 5 В, уровень логической единицы более 2,8 В, уровень логического нуля менее 0,5 В, средние времена задержки в диапазоне 10...20 нс, средняя потребляемая мощность 10, 15 Вт, коэффициент разветвления по выходу 10.

Недостатком элементов ТТЛ является кратковременное увеличение мощности, потребляемой в переходном режиме, что вызывает появление в цепях питания импульсных помех.

13. ТТЛ СО СЛОЖНЫМ ИНВЕРТОРОМ

В простой схеме И-НЕ, подключая выходы схемы на входы многоэмиттерного транзистора, мы искажаем распределение тока в выходной цепи и можем исказить сигнал. Для предотвращения этого используют сложный инвертор.

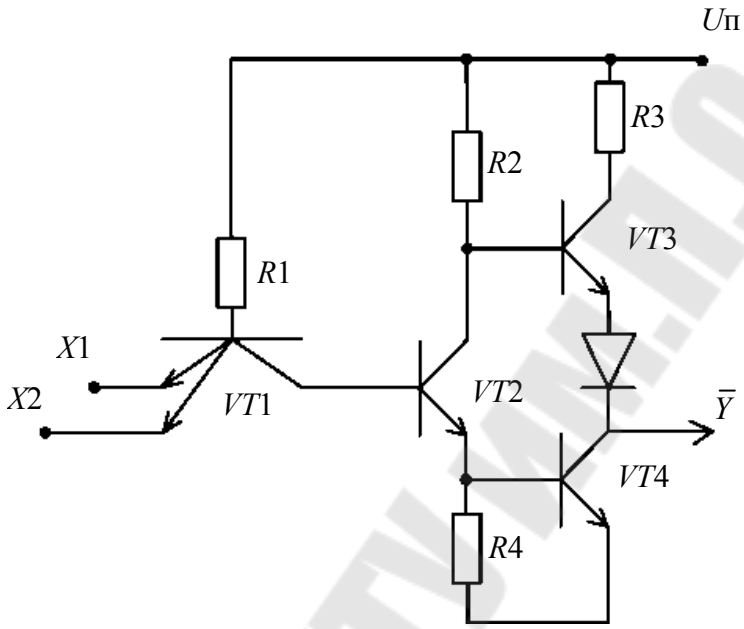

Рис. 9. Базовая схема серии К155 со сложным инвертором

При наличии на входах *X*1 и *X*2 напряжения логического 0 транзистор *VT*1 находится в режиме насыщения, а транзистор *VT*2 закрыт. Следовательно, закрыт и транзистор *VT*4, так как ток через *R*4 не протекает и напряжение на базе *VT*4 равно нулю. Транзистор *VT*3 открыт, так как его база подключена к источнику питания через *R*2. Сопротивление *R*3 невелико и *VT*3 работает как эмиттерный повторитель. Через *VT*3 и открытый диод протекает ток нагрузки логического элемента. Напряжение на выходе соответствует уровню логической 1.

При увеличении напряжения на всех входах потенциал базы *VT*2 возрастает и при $U_{\text{вх}} > U^0_{\text{пор}}$ транзистор *VT*2 открывается, начинает протекать коллекторный ток через *R*2 и *R*4. В результате базовый ток *VT*3 уменьшается, падение напряжения на нем увеличивается и выходное напряжение в схеме снижается. Дальнейшее увеличение входных напряжений приводит к насыщению *VT*2 и *VT*4 и запиранию *VT*1. *VT*3 и диод также закрыты, на выходе имеем уровень логического 0.

14. ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ С ТРЕМЯ УСТОЙЧИВЫМИ СОСТОЯНИЯМИ

В элементах с тремя устойчивыми состояниями выходные транзисторы заперты сигналом управляющего вывода. Выходное сопротивление запертых транзисторов велико, и микросхема практически полностью отключена от нагрузки. Такое состояние называется высокоимпедансным.

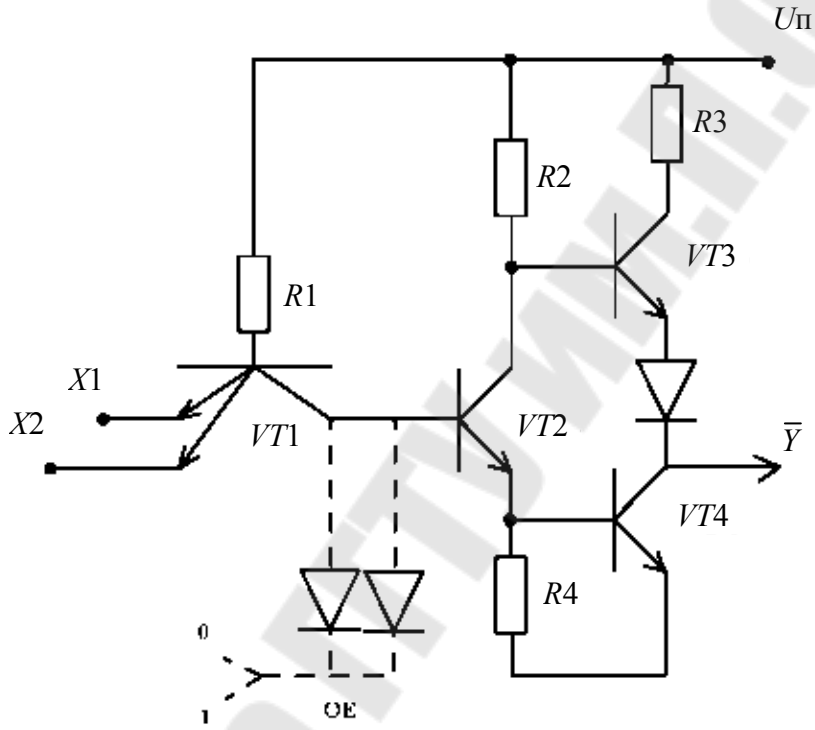

Рис. 10. Схема элемента И-НЕ с тремя устойчивыми состояниями

При использовании нескольких ЛЭ с тремя состояниями их выходы подключают к одной нагрузке. Управление микросхем осуществляется так, что в любой момент времени все микросхемы, кроме одной, находятся в высокоимпедансном состоянии. Таким образом удается по одной шине передавать в разных направлениях информацию от нескольких источников сигнала и сократить количество информационных магистралей. Вход включения третьего состояния имеет метку *EZ*, а выход, имеющий состояние высокого импеданса, обозначается через *Z*, либо *V*.

15. ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ С ОТКРЫТЫМ КОЛЛЕКТОРОМ

Логические элементы с открытым коллектором приведены на рис. 11.

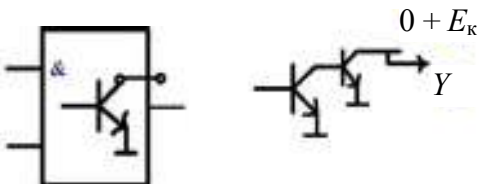

Рис. 11. Логические элементы с открытым коллектором

ТТЛ элементы имеют небольшие значения выходных сопротивлений. Поэтому нельзя объединить между собой выходы нескольких ТТЛ ЛЭ, так как в случае разных выходных сигналов через выходные транзисторы ЛЭ будут протекать большие токи. Для расширения функциональных возможностей у отдельных типономиналов на выходе ставят транзистор, коллекторная цепь которого оставлена свободной. Это ЛЭ с открытым коллектором (ОК). При использовании подобной ИС коллектор соединяют с источником напряжения через нагрузочный резистор. Роль его может выполнять резистор, обмотка реле, светодиод. Причем напряжение, к которому подключают нагрузочное сопротивление, может быть значительно больше напряжения питания ТТЛ ЛЭ. Так, например, в МС К155ЛА11 возможно подключение нагрузки к напряжению до 30 В.

Кроме того, ЛЭ с ОК позволяют осуществить непосредственное соединение между собой выходов нескольких ИС.

Наконец, подавая разные значения напряжения питания в схеме с ОК, можно получить разные уровни выходного сигнала. Это позволяет осуществить согласование микросхем серии ТТЛ с другими сериями, имеющими другие значения U^0 и U^1 без использования дополнительных преобразователей уровня.

16. РЕЖИМ НЕИСПОЛЬЗУЕМЫХ ВХОДОВ ЦИФРОВЫХ МИКРОСХЕМ

Типовым является случай наличия у элементов лишних входов, неиспользуемых входов, наличия в корпусах интегральных схем лишних элементов, нехватка у имеющихся элементов необходимого числа входов или нагрузочных способностей. Вопрос об использовании неиспользуемых входов решается по-разному для конкретных типов логик.

Пример. Необходимо получить $F = \overline{X1 \cdot X2 \cdot X3 \cdot X4 \cdot X5}$. При этом элемента 5 И-НЕ нет в наличии, зато имеется элемент 8 И-НЕ. Возможные варианты:

1. Не обращать внимания на «лишние» входы (т. е. оставить их разомкнутыми) (рис. 12).

2. Подсоединить их к задействованным входам (рис. 13).

3. Подать на них некие константы (рис. 14).

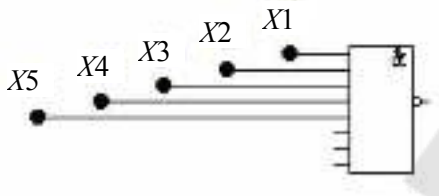

Рис. 12. Режим неиспользуемых входов цифровых микросхем. «Лишние» входы

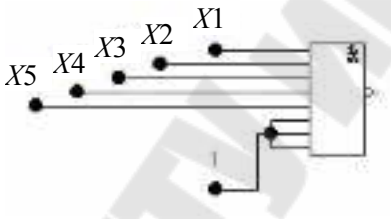

Рис. 13. Режим неиспользуемых входов цифровых микросхем. Подсоединение к константам

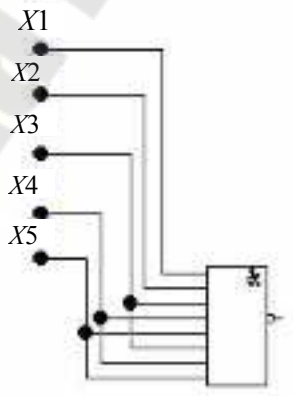

Рис. 14. Режим неиспользуемых входов цифровых микросхем. Подсоединение к задействованным входам

С точки зрения логических операций все три возможности правомерны. Но для разных схемотехнологий выбор соответствующего варианта является определяющим.

Для ЭСЛ: можно по 1-му варианту.

Это объясняется тем, что в схемах самих элементов уже предусмотрены резисторы, связанные с ИП, которые обеспечивают необходимые условия «лишним» входам.

Для КМОП: по 1-му – нельзя! У КМОП ИС большие входные сопротивления, поэтому на разомкнутых входах легко наводятся паразитные потенциалы, которые могут изменить работу схемы.

Для ТТЛШ: строгого запрета на оставление разомкнутых входов нет, но делать это незачем, так как пострадает быстродействие.

Для КМОП и ТТЛШ 2-й вариант принципиально возможен, но нежелателен в связи с тем, что это приводит к увеличению нагрузки на источник сигнала, что сопровождается снижением быстродействия источника.

Таким образом, наиболее рациональный 3-й вариант (рис. 15).

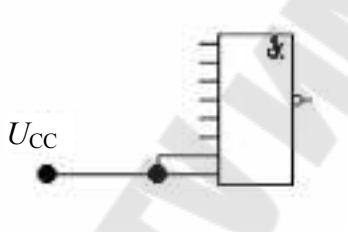

Рис. 15. Вариант решения проблемы лишних входов в КМОП логике. Для ТТЛШ схем

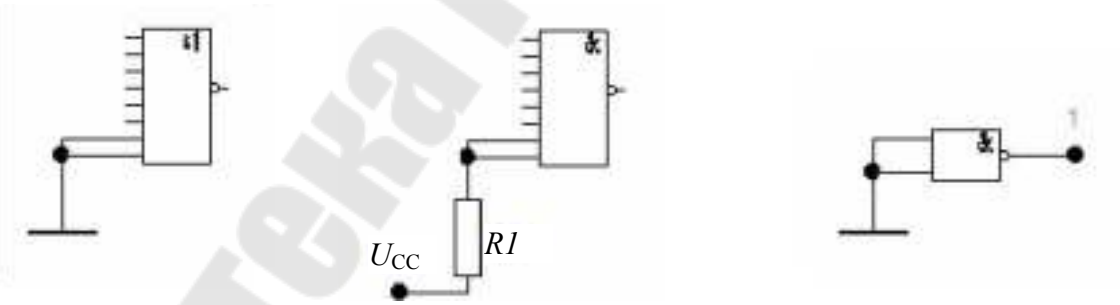

Рис. 16. Вариант решения проблемы лишних входов в ТТЛШ логике

17. ФУНКЦИОНАЛЬНЫЕ УЗЛЫ КОМБИНАЦИОННОГО ТИПА

Цифровым функциональным узлом называется устройство, предназначенное для выполнения определенных действий с двоичными переменными: хранение, сложение, счет, преобразование и т. д.

Различают функциональные устройства комбинационного и последовательностного типов.

Комбинационными называются функциональные узлы, выходные сигналы которых определяются комбинацией логических сигналов на входах, действующих в данный момент времени.

Комбинационные цифровые устройства (КЦУ) имеют в своем составе только набор логических вентилей и не обладают памятью. К ним относятся: шифраторы, дешифраторы, мультиплексоры, демультиплексоры, сумматоры, преобразователи кодов и т. д.

17.1. Шифратор. Построение на логических элементах

Шифратор – функциональный узел, предназначенный для преобразования поступающих на его входы управляющих сигналов (команд) в *n*-разрядный двоичный код. В частности, десятичные числа с помощью шифратора преобразуются в двоичный код. При подаче сигнала на один из входов на выходе появляется двоичный код, численно равный номеру возбужденного входа. Шифратор осуществляет «сжатие» информации, так как зашифрованная информация передается по меньшему числу выходных линий связи.

Шифратор, у которого при $m = 2n$ входах имеется n выходов, называется полным. Если $m < 2n$, шифратор называется неполным.

Функционирование неполного шифратора описывается табл. 6.

Таблица 6

Таблица истинности

	***A*1**	***A*2**	***A*1**	***A*0**
*X*0	0	0	0	0
*X*1	0	0	0	1
*X*2	0	0	1	0
*X*3	0	0	1	1
*X*4	0	1	0	0
*X*5	0	1	0	1
*X*6	0	1	1	0
*X*7	0	1	1	1
*X*8	1	0	0	0
*X*9	1	0	0	1

На основании этой таблицы выражения для каждого выхода имеют вид:

$$A0 = X1 \vee X3 \vee X7 \vee X5 \vee X9;$$

$$A1 = X2 \vee X3 \vee X6 \vee X7;$$

$$A2 = X4 \vee X5 \vee X6 \vee X7;$$

$$A3 = X8 \vee X9.$$

Для построения шифратора на логических элементах необходимы следующие логические вентили: 5ИЛИ, 4ИЛИ, 4ИЛИ, 2ИЛИ. Построим неполный двоичный шифратор (рис. 17).

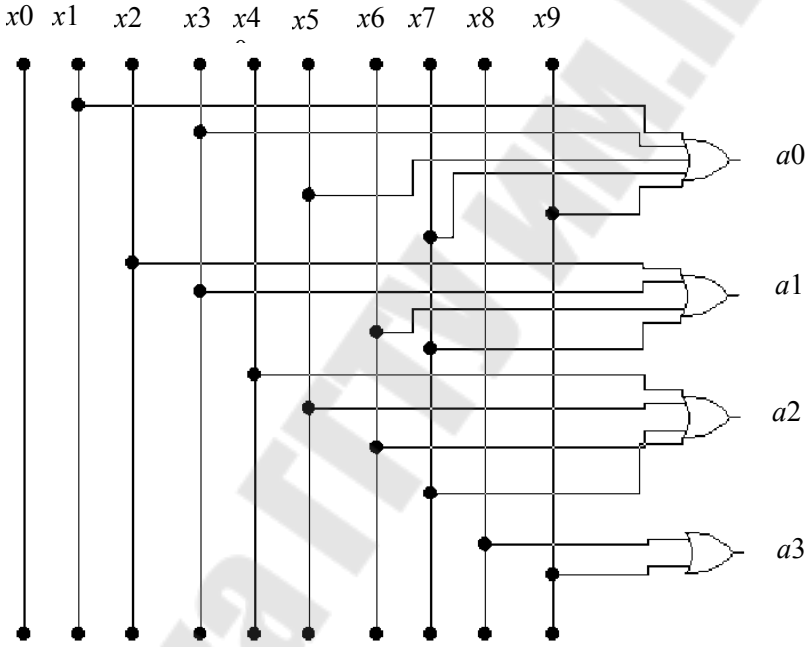

Рис. 17. Схема неполного двоичного шифратора на логических элементах

17.2. Дешифратор. Построение на логических элементах

Дешифратор – функциональный узел, предназначенный для преобразования *n*-разрядного двоичного кода в комбинацию управляющих выходных сигналов. Если число выходов $m < 2n$ – то это неполный дешифратор, если же $m = 2n$ – то это полный дешифратор.

Условное обозначение: *DC*(*decoder*)

Правила его работы определяются табл. 7.

Таблица 7

Таблица истинности

*A*3	*A*2	*A*1	*A*0	*X*0	*X*I	*X*2	*X*3	*X*4	*X*5	*X*6	*X*7	*X*8	*X*9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…
1	0	0	1	0	0	0	0	0	0	0	0	0	1

В соответствии с таблицей истинности функция, реализуемая по каждому выходу, имеет вид:

$$X0 = \overline{A0} \cdot \overline{A1} \cdot \overline{A2} \cdot \overline{A3};$$

$$X1 = A0 \cdot \overline{A1} \cdot \overline{A2} \cdot \overline{A3};$$

$$X2 = \overline{A0} \cdot A1 \cdot \overline{A3};$$

$$X3 = A0 \cdot A1 \cdot \overline{A2};$$

$$X4 = \overline{A0} \cdot \overline{A1} \cdot A2;$$

$$X5 = A0 \cdot \overline{A1} \cdot A2;$$

$$X6 = \overline{A0} \cdot A1 \cdot A2;$$

$$X7 = A0 \cdot A1 \cdot A2;$$

$$X8 = \overline{A0} \cdot A3;$$

$$X9 = A0 \cdot A3.$$

Таким образом, в состав данного дешифратора входят 10 схем И и 4 схемы НЕ. Отметим, что при любой комбинации сигналов на входах сигнал логической 1 наблюдается только на одном из выходов.

Схема представлена на рис. 18:

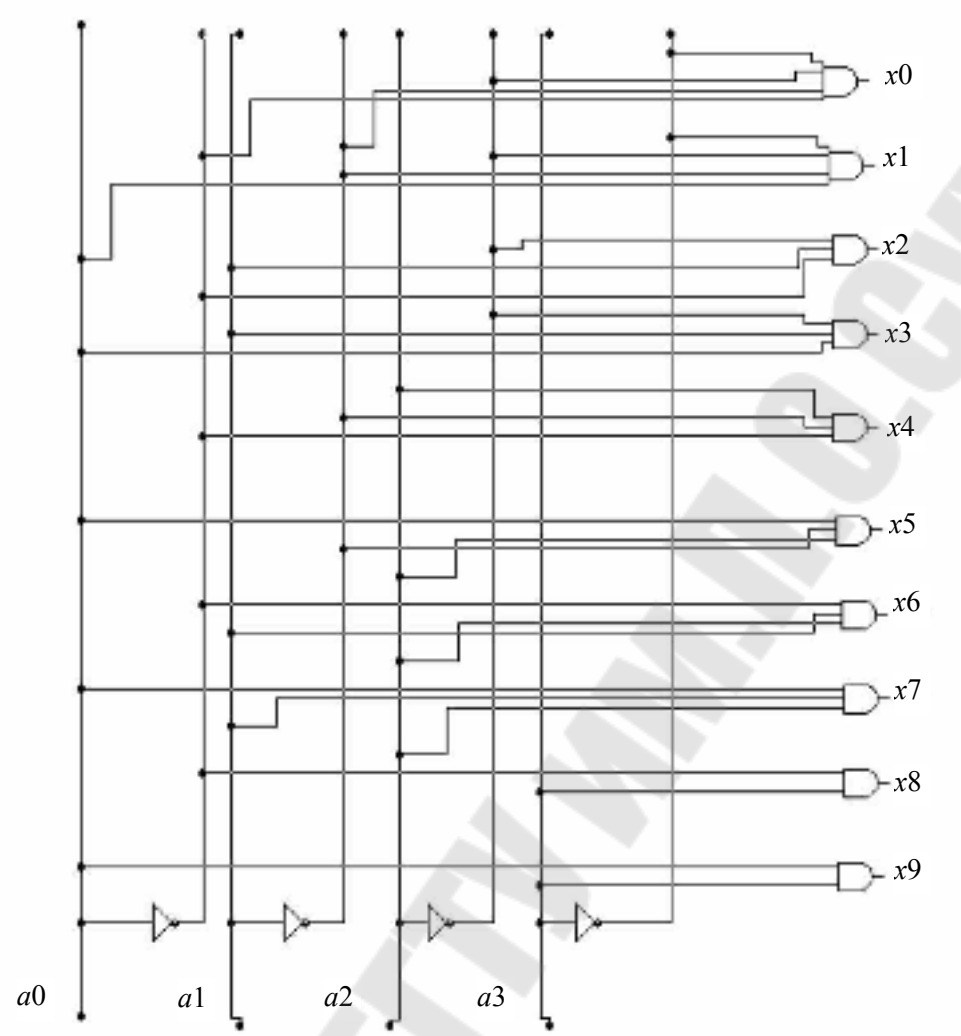

Рис. 18. Схема двоичного дешифратора на логических элементах

17.3. Наращивание размерности дешифратора

Из малоразрядных дешифраторов можно построить схему, эквивалентную схеме дешифратора большей разрядности. Для этого входные слова делятся на поля. Разрядность поля младших разрядов соответствует числу входов имеющегося дешифратора. Оставшиеся поля старших разрядов служат для получения сигналов разрешения работы одного из дешифраторов, декодирующих поле младших разрядов. Работу схемы рассмотрим на примере числа 11001. Это число 25 в десятичной системе счисления. На входе дешифратора первого яруса имеется код 11, его выход № 3, что разрешает работу четвертого дешифратора 2 яруса. На входе дешифратора № 4 действует код 001, поэтому единица появится на его первом выходе, т. е. на 25-м выходе всей схемы.

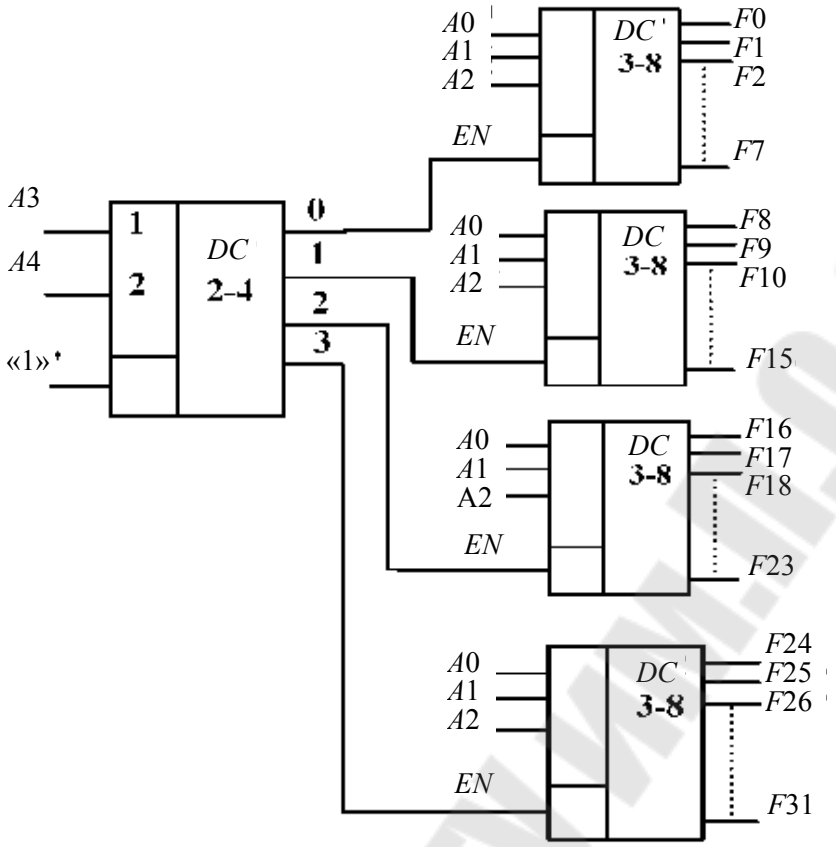

Рис. 19. Наращивание размерности дешифратора

Разрешение работы дешифратора осуществляется подачей «1» на разрешающий вход дешифратора первого яруса.

17.4. Воспроизведение произвольных логических функций с помощью дешифратора

Дешифраторы со схемами ИЛИ можно использовать для воспроизведения произвольных логических функций. На выходах дешифратора вырабатываются все конъюнктивные термы, которые только можно составить из данного числа аргументов. Логическая функция в СДНФ есть дизъюнкция некоторого числа таких термов. Собирая нужные термы по схеме ИЛИ, можно получить любую функцию данного числа аргументов.

Разберем пример соединения (рис. 20).

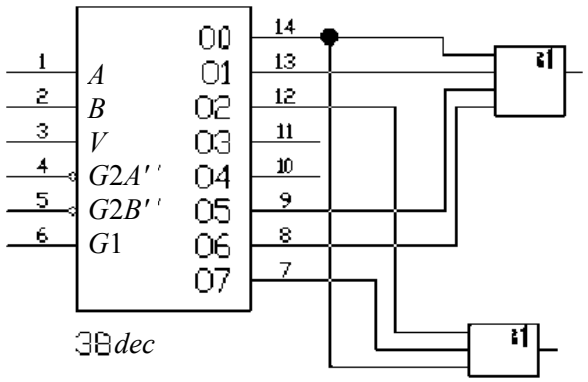

Рис. 20. Пример соединения дешифратора и логических вентилей ИЛИ для получения произвольных логических функций

$$00: \overline{X3} \cdot \overline{X2} \cdot \overline{X1};$$

$$01: \overline{X3} \cdot \overline{X2} \cdot X1;$$

$$02: \overline{X3} \cdot X2 \cdot \overline{X1};$$

$$03: \overline{X3} \cdot X2 \cdot X1;$$

$$04: X3 \cdot \overline{X2} \cdot \overline{X1};$$

$$05: X3 \cdot \overline{X2} \cdot \overline{X1};$$

$$06: X3 \cdot X2 \cdot \overline{X1};$$

$$07: X3 \cdot X2 \cdot X1;$$

$$F1: \overline{X3} \cdot \overline{X2} \vee X3 \cdot X1;$$

$$F2: \overline{X3} \cdot X2 \cdot X1 \vee X2 \cdot \overline{X1}.$$

Подобное решение может быть целесообразно при необходимости выработки нескольких функций одних и тех же переменных.

17.5. Мультиплексор

Мультиплексор – это функциональный узел, который осуществляет управляемую коммутацию информации, поступающей по N параллельным каналам, на одну выходную линию. Коммутация есть последовательный опрос и передача информации. Коммутация определенной входной линии происходит в соответствии с двоичным адресным кодом $A_n - 1A_n...A_1A_0$. Входы $A_0 \ldots A_{n-1}$ являются адресными.

Их значение определяет одну из переменных X, передаваемых на выход F. Если адресный код имеет n-разрядов, то можно осуществить $N = 2^n$ комбинаций адресных сигналов, каждый из которых обеспечивает коммутацию информации, поступающей по одному из N входов на выход. В простейшем случае двухразрядового адресного кода $n = 2$ и максимальное число входных линий $N = 4$. Таблица истинности такого мультиплексора 4-1 (*Мих* 4-1) представлена в табл. 8.

Таблица 8

Таблица истинности

***A*0**	***A*1**	***F***
0	0	X_0
0	1	X_1
1	0	X_2
1	1	X_3

Исходя из таблицы получаем характеристическое уравнение такого мультиплексора:

$$F = \overline{A0} \cdot \overline{A1} \cdot X0 \vee \overline{A0} \cdot A1 \cdot X1 \vee A0 \cdot \overline{A1} \cdot X2 \vee A0 \cdot A1 \cdot X3.$$

По этому выражению построим мультиплексор на логических элементах. Из полученного выражения следует, что в состав структурной схемы такого мультиплексора входят два инвертора, четыре схемы И и одна ИЛИ.

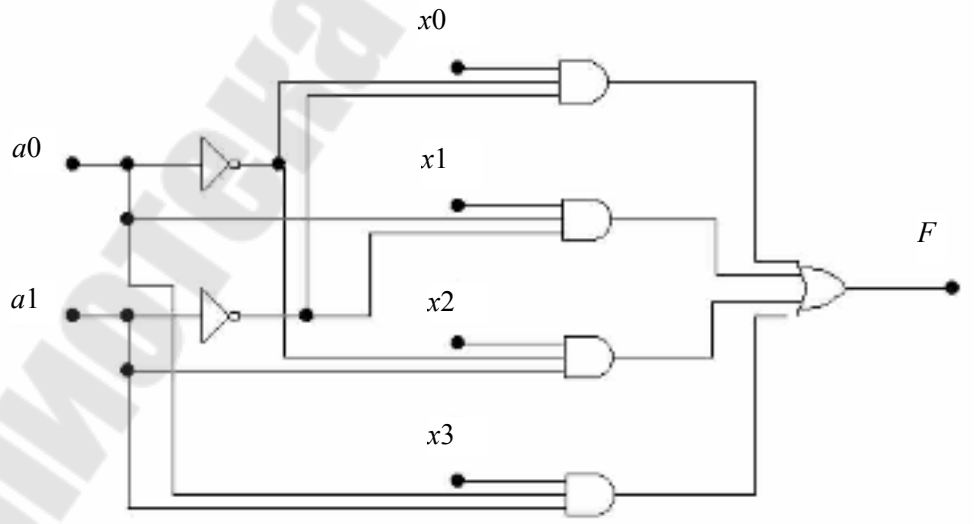

Рис. 21. Схема мультиплексора 4-1 на логических элементах (условные обозначения приведены на рис. 22)

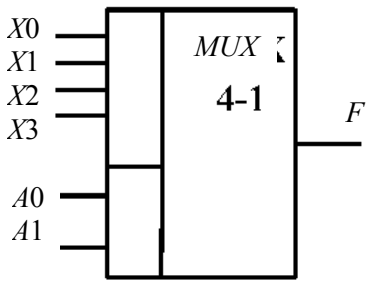

Рис. 22. Условное обозначение мультиплексора 4-1

17.6. Демультиплексор

Демультиплексор – функциональный узел, осуществляющий управляемую коммутацию информации, поступающей по одному каналу на N выходов. В общем случае число выходов линий N определяется количеством адресных входов n, т. е. $N = 2n$.

Для $n = 2$ функционирование демультиплексора осуществляется в соответствии с табл. 9.

Таблица 9

Таблица истинности

***A*0**	***A*1**	***X*0**	***X*1**	***X*2**	***X*3**
0	0	*F*	0	0	0
0	1	0	*F*	0	0
1	0	0	0	*F*	0
1	1	0	0	0	*F*

Из таблицы следует, что информация F, в зависимости от адресных входов, направляется в разные выходные линии X_i. При этом на остальные линии информация не поступает. Характеристические уравнения демультиплексора в соответствии с таблицей истинности будут:

$$X0 = \overline{A0} \cdot \overline{A1} \cdot F;\ X1 = \overline{A0} \cdot A1 \cdot F;\ X2 = A0 \cdot \overline{A1} \cdot F;\ X3 = A0 \cdot A1 \cdot F.$$

Соответствующая этим функциям структурная схема представлена на рис. 23.

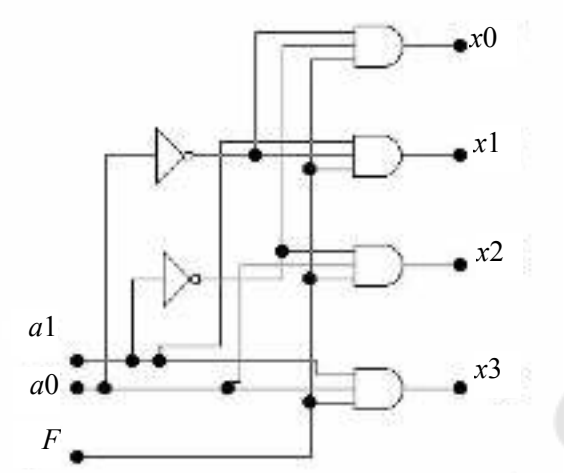

Рис. 23. Структурная схема демультиплексора 1-4

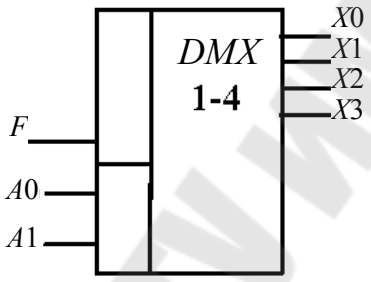

Рис. 24. Условное обозначение демультиплексора 1-4

17.7. Наращивание размерности мультиплексора

В стандартных сериях размерности мультиплексоров не превышают 16-1 (серия КП1). Наращивание размерности мультиплексора возможно с помощью пирамидальной структуры из нескольких мультиплексоров. Первый ярус схемы представляет столбец, содержащий столько мультиплексоров, сколько необходимо для получения нужного числа информационных входов. Все мультиплексоры столбца адресуются одним и тем же кодом, составленным из соответствующего числа младших разрядов общего адресного кода. Старшие разряды адресного кода используются во втором ярусе, мультиплексор которого обеспечивает поочередную работу мультиплексоров первого яруса на общий выходной канал.

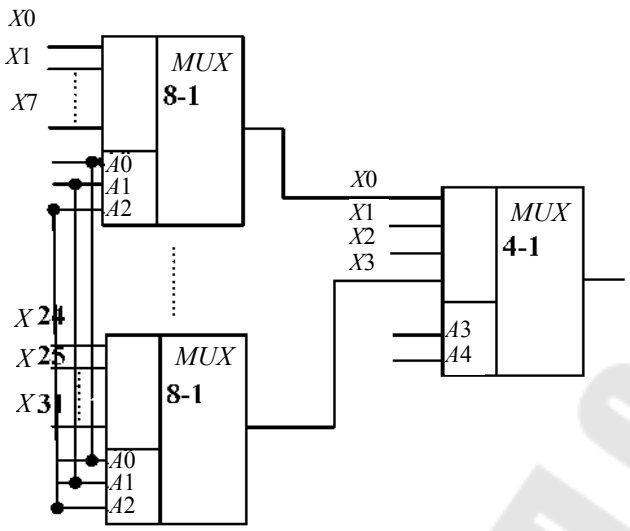

Рис. 25. Пример построения мультиплексора 32-1
из мультиплексоров 8-1

Таблица 10

Таблица истинности

***X*4**	***X*3**	$F_{ост}$
0	0	*X*1 *X*2
0	1	1
1	0	*X*1 *X*2
1	1	*X*1 *X*2

17.8. Полусумматор. Синтез полного сумматора из полусумматоров

Сумматор – функциональный узел, осуществляющий сложение двоичных чисел. Различают сумматор неполный (полусумматор – *Half Adder*) и полный. Простейшим является неполный одноразрядный сумматор.

Состояния такого сумматора показаны в табл. 11.

Таблица 11

Таблица истинности

A_0	B_0	S_0	P_i
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Здесь A_0 и B_0 – одноразрядные суммируемые числа, а S_0 и P_1 – значения суммы в данном разряде (частичная сумма) и перенос в следующий разряд, соответственно. Выражения для P_1 и S_0 получаются на основании таблицы: $P1 = A0 \cdot B0$; $S0 = A0 \oplus B0$. Из чего следует, что формирование переноса можно осуществить на элементе И, а частичной суммы – на элементе исключающее ИЛИ (И2ЛИ).

В полусумматоре не учитывается перенос из младшего разряда.

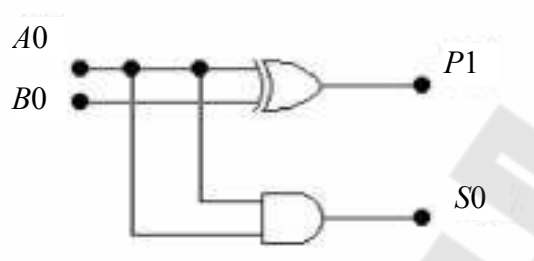

Рис. 26. Структурная схема полусумматора на логических элементах

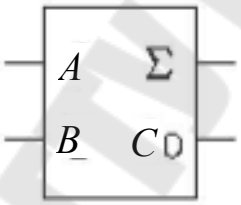

Рис. 27. Условное обозначение полусумматора

Если в сумматоре учитывается перенос из младшего разряда в старший, то это полный сумматор.

Операции сложения и вычитания начинаются с младших разрядов. При сложении двух единиц в одном разряде записывается 0 и образуется 1 переноса в следующий разряд, которая складывается с числами этого разряда:

$$10110+1001=11111.$$

Сумматор выполняет арифметическое сложение, в противовес логическому (операции дизъюнкции).

Полный сумматор может быть синтезирован на основе двух полусумматоров и логического вентиля ИЛИ (рис. 28).

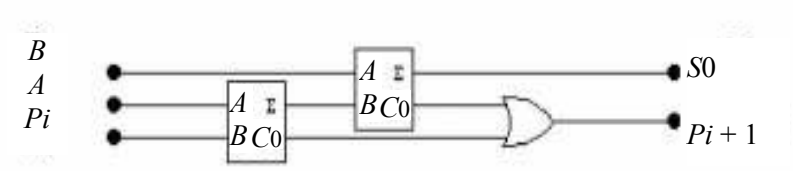

Рис. 28. Пример построения полного сумматора
из двух полусумматоров

Синтез полного сумматора на логических элементах

В полусумматоре не учитывается перенос из младшего разряда. Он может применяться для сложения чисел только в нулевом разряде. Если в сумматоре учитывается перенос из младшего разряда в старший, то это полный сумматор. Он осуществляет арифметическое сложение одноразрядных двоичных чисел *a* и *b* и числа *c*; перенос из младшего разряда, образуя на выходах значения суммы *S*, и число переноса в старший разряд *c* + 1. Такая операция осуществляется при сложении многоразрядных чисел. Обозначим:

Ai, *Bi* – складываемые числа;

Si – результат арифметического сложения в данном разряде;

Ci – перенос из младшего разряда;

Ci + 1 – перенос в следующий разряд.

Для построения полного сумматора необходимы 1 элемент 4 И-НЕ, 5 элементов 3 И-НЕ, 3 элемента 2И-НЕ и 3 инвертора, т. е. всего 12 вентилей.

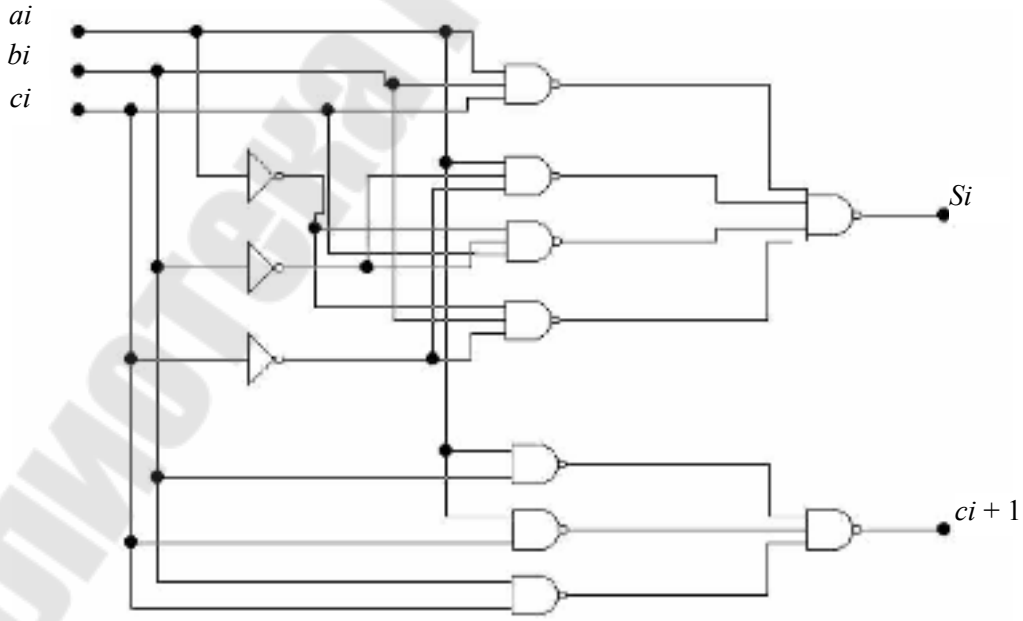

Рис. 29. Построение схемы полного сумматора
на логических вентилях

Сумматор параллельного действия с последовательным переносом

Из одноразрядного сумматора можно построить сумматор параллельного действия с последовательным переносом на любое количество разрядов.

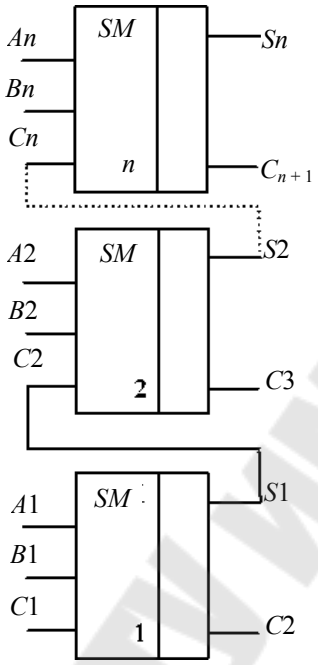

Рис. 30. Построение сумматора параллельного действия с последовательным переносом

При подаче слагаемых цифры их разрядов поступают на соответствующий одноразрядный сумматор. Каждый из одноразрядных сумматоров формирует на своих выходах цифру соответствующего разряда суммы и перенос, передаваемый на вход одноразрядного сумматора, следующего, более старшего разряда.

Параллельный сумматор с параллельным переносом

Данный сумматор разработан для получения максимального быстродействия, не имеет последовательного переноса. Во всех разрядах результаты вырабатываются одновременно, параллельно во времени. Сигнал переноса для данного разряда формируется специальными схемами, на входы которых поступают все переменные, необходимые для выработки переноса. К этим величинам относятся внешний входной перенос (если он есть) и значения всех разрядов слагаемых, младших относительно данного.

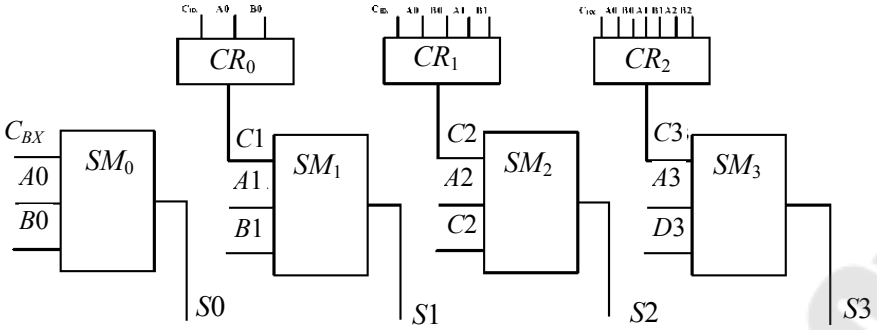

Рис. 31. Построение параллельного сумматора
с параллельным переносом

Одноразрядные сумматоры для каждого разряда здесь упрощены, так как от них выход переноса не требуется, достаточно одного выхода сумматора.

17.9. Цифровые компараторы

Компаратор – это устройство, предназначенное для сравнения двух чисел A и B, каждое из которых представлено в двоичной форме исчисления входными сигналами.

Простейший компаратор производит проверку равенства двух чисел. Два числа равны, если равны все соответствующие разряды этих двух чисел. Сравнение начинается со старшего разряда. Если два числа равны, то выход компаратора $y = 1$; если нет, то $y = 0$.

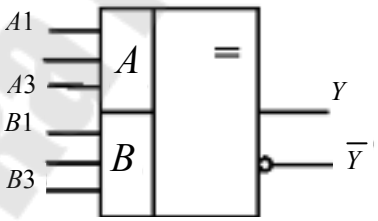

Рис. 32. Условное обозначение цифрового компаратора

Работа компаратора описывается уравнением $Z = Ai \cdot Bi \vee \overline{Ai} \cdot \overline{Bi} = \overline{A \oplus B}$ – функция равнозначности. Построение компаратора на логических элементах показано на рис. 33.

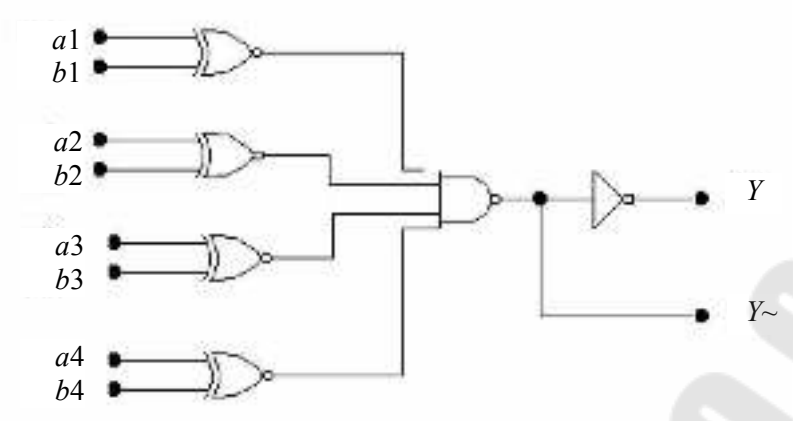

Рис. 33. Схема цифрового компаратора на логических элементах

Более универсальны компараторы, которые помимо констатации равенства двух чисел могут установить, какое из них больше. Простейшая задача состоит в сравнении двух одноразрядных чисел A и B. Такое сравнение реализуется переключательными функциями:

$$Y_{a=b} = AB \vee \overline{A}\,\overline{B},$$

$$Y_{a>b} = A\overline{B},$$

$$Y_{a<b} = \overline{A}B.$$

Схема подобного компаратора на один разряд выглядит следующим образом:

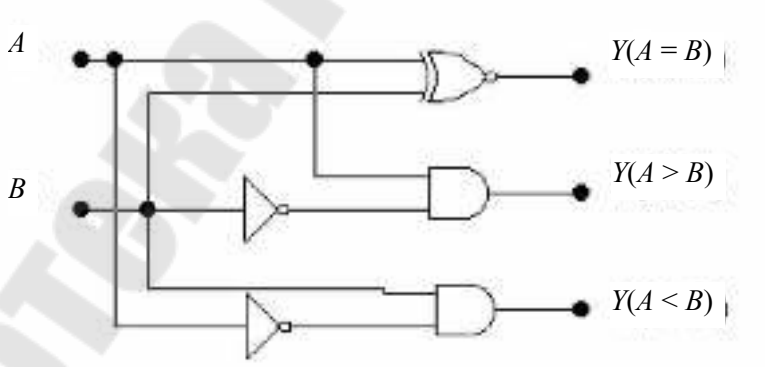

Рис. 34. Схема цифрового компаратора, определяющего, какое из чисел больше

При сравнении многоразрядных чисел алгоритм такой: сначала сравнивают значения старших разрядов; если они различны, то эти разряды и определяют результат сравнения. Если же они равны, то необходимо сравнивать следующие за ним более младшие разряды и т. д.

17.10. Преобразователь кода для семисегментной индикации

Один из способов цифровой индикации состоит в следующем. Имеется семь сегментов (рис. 35).

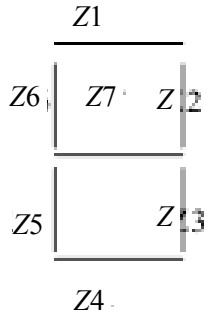

Рис. 35. Преобразователь кода семисегментный

Так выглядит индикатор. Изображение символов в индикаторе формируется высвечиванием необходимой комбинации сегментов. Смена изображений достигается путем соответствующей коммутации сегментов.

Десятичные цифры, отображение которых необходимо вызвать, задаются обычно в двоичном коде. Через $Y1 \ldots Y7$ обозначим управляющие импульсы. Если элемент светится, то он находится в состоянии «1», если нет – «0».

Однако управление цифровым индикатором осуществляется, как правило, по несколько иному алгоритму. Управление осуществляется следующим образом: уровень логической единицы на входе индикатора вызывает его погашение.

Например, чтобы увидеть цифру «6», необходимо: $y2 = 1$, $Z2 = 0$; т. е. $y = \overline{z}$.

$$y1 = \overline{x1} \cdot \overline{x2} \cdot x3 \vee x1 \cdot \overline{x2} \cdot \overline{x3} \cdot \overline{x4} = \overline{\overline{x1} \cdot \overline{x2} \cdot x3} \cdot \overline{x1 \cdot \overline{x2} \cdot \overline{x3} \cdot \overline{x4}};$$

$$y2 = \overline{x1} \cdot x2 \cdot x3 \vee x1 \cdot \overline{x2} \cdot x3 = \overline{\overline{x1} \cdot x2 \cdot x3} \cdot \overline{x1 \cdot \overline{x2} \cdot x3};$$

$$y3 = \overline{x1} \cdot x2 \cdot x3;$$

$$y4 = \overline{x1} \cdot \overline{x2} \cdot x3 \vee x1 \cdot \overline{x2} \cdot \overline{x3} \cdot \overline{x4} \vee x1 \cdot x2 \cdot x3 =$$

$$= \overline{\overline{x1} \cdot \overline{x2} \cdot x3} \cdot \overline{x1 \cdot \overline{x2} \cdot \overline{x3} \cdot \overline{x4}} \cdot \overline{x1 \cdot x2 \cdot x3};$$

$$y5 = \overline{x1} \cdot \overline{x2} \cdot x3 \vee x1 = \overline{\overline{x1} \cdot \overline{x2} \cdot x3 \cdot \overline{x1}};$$

$$y6 = x1 \cdot \overline{x2} \cdot \overline{x3} \cdot \overline{x4} \vee \overline{x1} \cdot x2 \cdot \overline{x3} \vee x1 \cdot x2 = \overline{\overline{x1 \cdot \overline{x2} \cdot \overline{x3} \cdot \overline{x4}} \cdot \overline{\overline{x1} \cdot x2 \cdot \overline{x3}} \cdot \overline{x1 \cdot x2}};$$

$$y7 = x1 \cdot x2 \cdot x3 \vee \overline{x2} \cdot \overline{x3} \cdot \overline{x4} = \overline{\overline{x1 \cdot x2 \cdot x3} \cdot \overline{\overline{x2} \cdot \overline{x3} \cdot \overline{x4}}}.$$

Используя эти выражения, можно построить схему преобразователя кода (рис. 36).

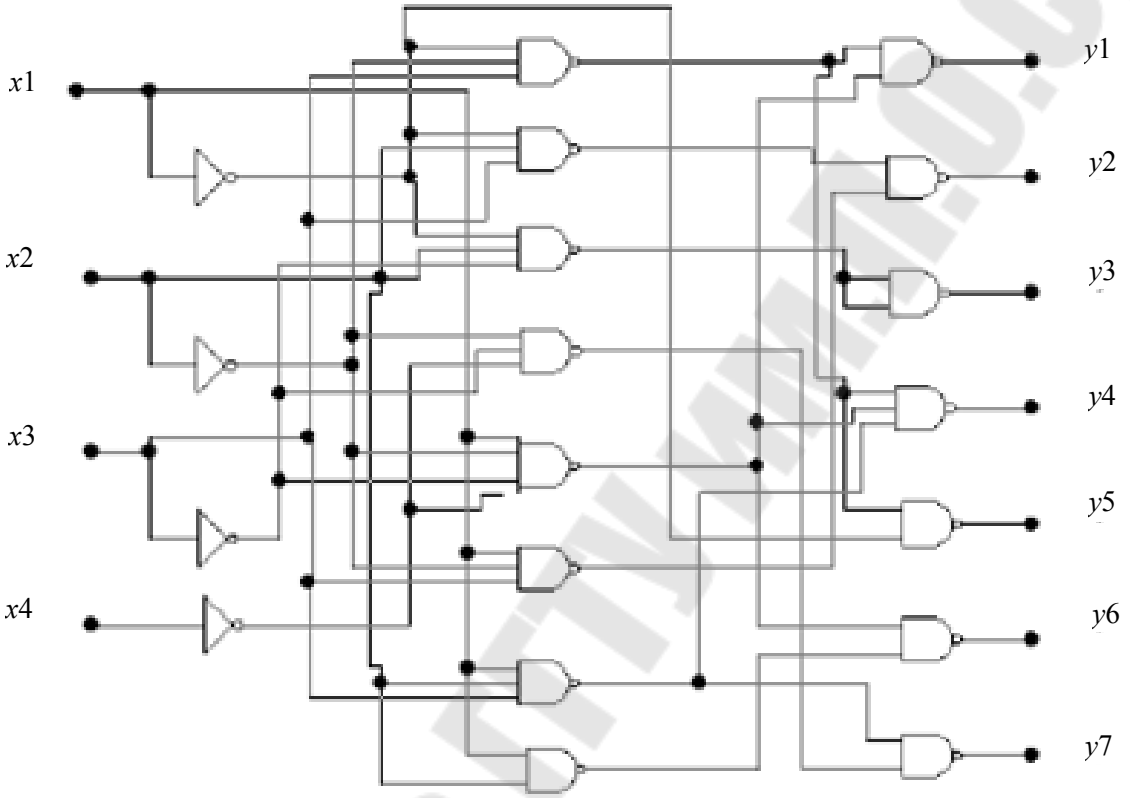

Рис. 36. Схема преобразователя кода для семисегментного индикатора

17.11. Асинхронный *RS*-триггер с прямыми входами

Схема триггера, выполненного на двух логических элементах 2ИЛИ-НЕ. Он содержит два информационных входа *R*(*Reset*) и *S*(*Set*), а правила его функционирования определяются табл. 12.

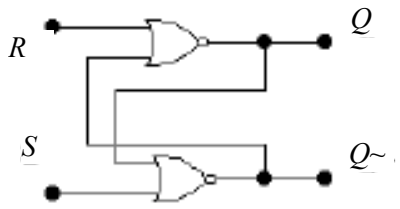

Рис. 37. Структурная схема асинхронного *RS*-триггера на логических элементах ИЛИ-НЕ

Таблица 12

Таблица истинности

Такт t_n		Такт t_{n+1}	
S^n	R^n	Q^{n+1}	Q^{n+1}
0	0	Q_t	Q_t
0	1	0	1
1	0	1	0
1	1	–	–

Из таблицы следует, что комбинация входных сигналов $S^n = R^n = 0$ не изменяет состояния триггера. Комбинация входных сигналов $S^n = 1, R^n = 0$ переводит *RS*-триггер в единичное состояние. Поэтому вход S называют единичным входом: появление логической 1 на его входе гарантирует наличие логической 1 на прямом выходе триггера независимо от его первоначального состояния. Комбинация входных сигналов $S^n = 0, R^n = 1$ обеспечивает нулевое состояние триггера. По этой причине вход *R* называют нулевым входом.

$S^n = R^n = 1$ – запрещенная комбинация, так как при ней нарушается логика триггера.

Переключение триггера под действием входных сигналов описывается таблицей переключений либо характеристическим уравнением.

17.12. Асинхронный *RS*-триггер с инверсными входами

Асинхронный *RS*-триггер можно выполнить и на двухвходовых ЛЭ И-НЕ. В отличие от *RS*-триггера на ЛЭ ИЛИ-НЕ переключение данного триггера осуществляется сигналами логического 0. Такой триггер называют триггером с инверсным управлением.

Таблица функционирования *RS*-триггера представлена ниже.

Таблица 13

Таблица истинности

Такт t_n		Такт t_{n+1}	
S^n	R^n	Q^{n+1}	$\overline{Q}^{n+1}$
0	0	–	–
0	1	1	0
1	0	0	1
1	1	Q_t	Q_t

17.13. Асинхронный *JK*-триггер

JK-триггер отличается от *RS*-триггера отсутствием запрещенных комбинаций информационных сигналов *J* и *K*.

Построение *JK*-триггера на элементах И-НЕ показано на рис. 38.

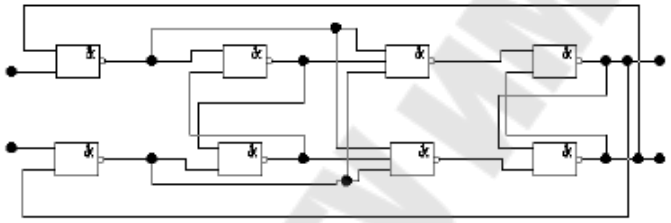

Рис. 38. Структурная схема *JK*-триггера на логических элементах И-НЕ

Временная диаграмма такого триггера представлена на рис. 39.

Выход данного *JK*-триггера переключается после окончания импульса.

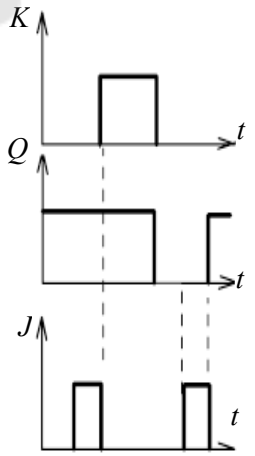

Рис. 39. Временные диаграммы *JK*-триггера

Функционирование *JK*-триггера описывается в таблице переключений (табл. 14).

Таблица 14

Таблица истинности

J^n	K^n	Q^n	Q^{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

17.14. Счетный *T*-триггер

Свойство *JK*-триггера переходить в инверсное состояние при одновременной подаче входных сигналов $J = K = 1$ позволяет создать на его основе счетный *T*-триггер. Для этого достаточно объединить входы *J* и *K*. Счетный *T*-триггер также называется триггером со счетным запуском. Он откликается на каждую «1». В счетном триггере входы *J* и *K* объединяются между собой. Объединенный вход носит название *T*-вход. Поэтому схема *T*-триггера будет такой же, как и *JK*-триггера, но входы *J* и *K* объединяют.

Функциональное назначение входов триггеров

В табл. 15 представлено назначение функциональных входов триггеров.

Таблица 15

Функциональное назначение входов триггеров

Обозначение	**Назначение**
	Информационные входы
S	Вход для раздельной установки *RS*-триггера в состояние «1»
R	Вход для раздельной установки *RS*-триггера в состояние «0»
J	Вход для установки *JK*-триггера в состояние «1»
K	Вход для установки *JK*-триггера в состояние «0»
T	Счетный вход триггера
D	Вход для установки триггера в состояние «0» или «1»

Окончание табл. 15

Обозначение	Назначение
	Информационные входы
V	Управляющие входы
	Подготовительный вход для разрешения приема информации
C	Подготовительный вход для осуществления приема информации или вход синхронизации

17.15. Синхронные триггеры на логических элементах

Триггеры, реализуемые на логических элементах, имеют конечное время переключения t_{sw}, определяемое суммарным средним временем задержки распространения сигнала $\sum_n t_{del\,md}$, где m – число логических элементов, составляющих триггер. В результате выходной сигнал триггера в течение времени t_{sw} после подачи входного сигнала сохраняет значение, не соответствующее этому сигналу, т. е. является ложным. Это обстоятельство может привести к ошибкам в работе устройства обработки информации, состоящего из большого числа логических элементов. Поэтому считывание информации осуществляется в те моменты времени, когда появление ложного сигнала заведомо исключено.

17.16. Синхронный *RS*-триггер

Структурная схема и условное обозначение синхронного *RS*-триггера представлена на рис. 40.

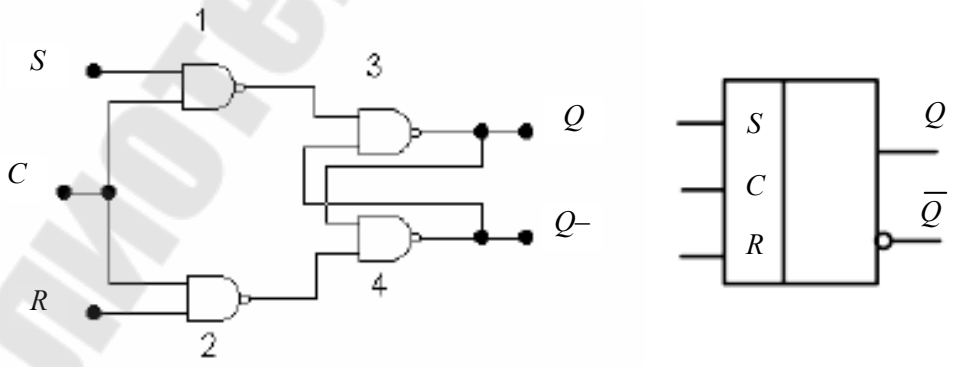

Рис. 40. Структурная схема синхронного *RS*-триггера.
Условное обозначение *RS*-триггера синхронного

Структурная схема синхронного *RS*-триггера помимо элементов *3* и *4*, составляющих триггер с инверсными входами, включает два входных элемента И-НЕ (1 и 2), обеспечивающих синхронный режим работы. Буквой *C* обозначен вход тактовых (синхронизирующих) импульсов. При наличии входного сигнала ($S = 1$ или $R = 1$) переключение триггера происходит только в момент поступления тактового импульса $C = 1$, так как при этом условии на одном из входов триггера возникает сигнал лог.0. Временные диаграммы работы синхронного и асинхронного *RS*-триггеров приведены на следующих рисунках. Синхронный *RS*-триггер также имеет запрещенную комбинацию входных сигналов: $R_n S_n C_n = 1$. Действительно, при такой комбинации входных сигналов выходные сигналы элементов И-НЕ будут соответствовать лог.1, что для *RS*-триггера недопустимо.

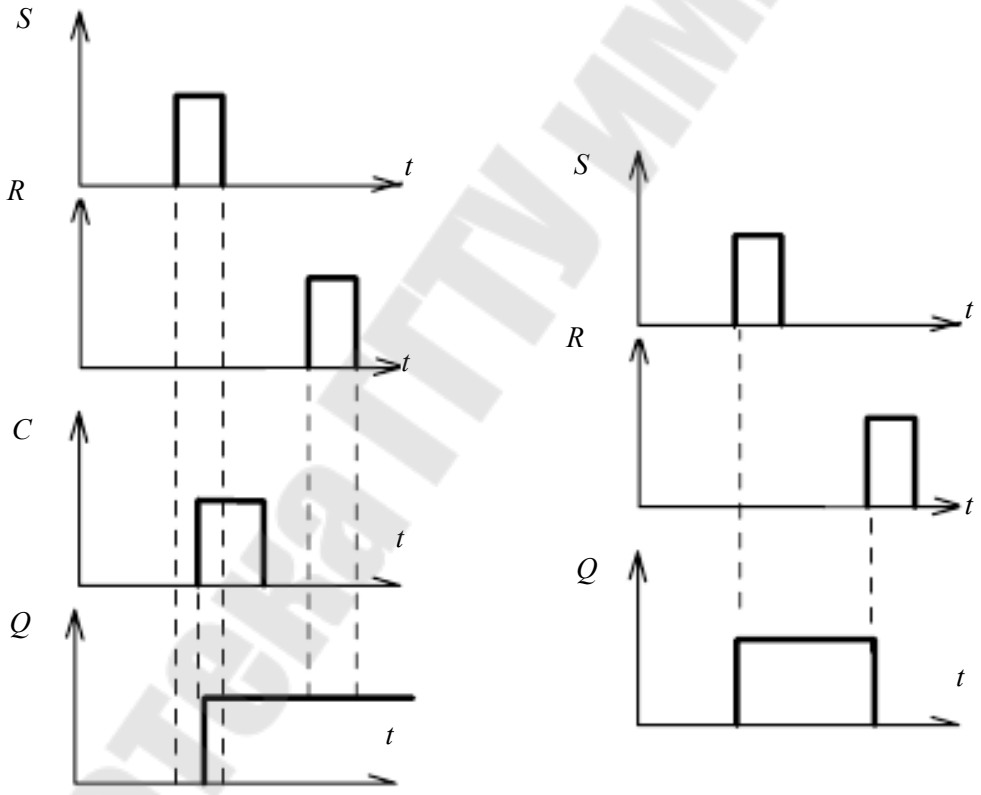

Рис. 41. Временные диаграммы синхронного и асинхронного *RS*-триггеров

Составим таблицу состояний синхронного *RS*-триггера.

Таблица 16

Таблица истинности

C^n	S^n	R^n	Q^n	Q^{n+1}
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
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	–

17.17. Синхронные *JK*- и *T*-триггеры

Имеют такую же структуру, что и асинхронный. Отличие состоит в том, что входные элементы И-НЕ имеют три входа. Входы *C* объединены и используются для подачи тактовых сигналов. Ниже приведена таблица функционирования *JK*-триггера.

Таблица 17

Таблица истинности

C^n	J^n	K^n	Q^{n+1}
0	0	0	Q^n
1	0	0	Q^n
1	0	1	0
1	1	0	1
1	1	1	$\overline{Q}^n$

Характеристическое уравнение синхронного *JK*-триггера имеет вид:

$$Q^{n+1} = \overline{K}^n Q^n + \overline{C}^n Q^n + C^n J^n \overline{K}^n.$$

Синхронный счетный *T*-триггер получают, объединяя информационные входы *J* и *K*. Управляющие сигналы *T* подаются на объединенный вход, а тактовые импульсы – на вход *C*. Характеристическое уравнение счетного триггера получается из соответствующего уравнения для *JK*-триггера путем замены *J* и *K* на *T*:

$$Q^{n+1} = (\overline{T}^n + \overline{C}^n)Q^n + C^n\ T^n\overline{Q}^n.$$

17.18. Синхронный триггер задержки (*D*-триггер)

Этот триггер можно получить, объединяя вход *J* со входом *K* через инвертор. При таком включении независимо от значения сигнала D^n на одном из управляющих входов имеется уровень лог.1: при $D^n = 1\ J^n = 1,\ K^n = 0;$ при $D^n = 0\ \ J^n = 0,\ K^n = 1.$ Таким образом, исключаются комбинации входных сигналов $J^nK^n = 1$ и $J^nK^n = 0.$

Характеристическое уравнение *D*-триггера может быть получено из уравнения для синхронного *JK*-триггера путем подстановки в него значений $J^n = D^n$ и $K^n = D^n$:

$$Q^{n+1} = Q^n\ (\overline{D}^n + \overline{C}^n) + C^nD^n\overline{Q}^n.$$

Из этого соотношения следует, что при наличии синхронизирующего сигнала (C_n = 1) на выходе триггера возникает сигнал, соответствующий входному сигналу, имевшему место в предшествующем такте $Q^{n+1} = D^n$. Таким образом, *D*-триггер осуществляет задержку сигнала на 1 такт.

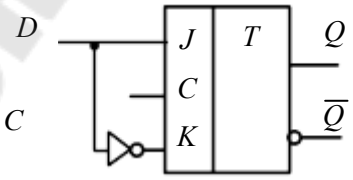

Рис. 42. Получение *D*-триггера из *JK*-триггера

17.19. Синхронные триггеры со статическим и с динамическим управлением

Синхронные триггеры делятся на два типа: со статическим и с динамическим управлением. В первых триггеры реагируют на сиг-

налы на информационных входах во время действия сигнала на синхронизирующем входе. Поэтому изменения информационных сигналов допускаются только при отсутствии сигнала на синхровходе. В синхронных триггерах с динамическим управлением прием сигналов с информационных входов происходит в течение малой длительности фронта (положительного или отрицательного) сигнала на синхровходе или на информационном входе при постоянном сигнале на синхровходе.

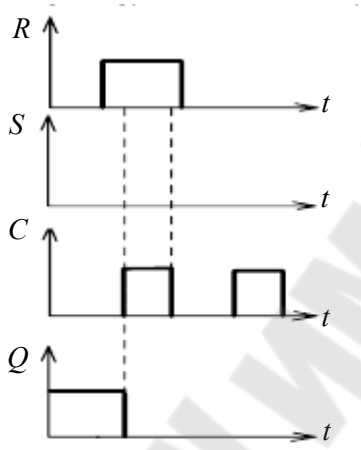

Рис. 43. Временные диаграммы статического *RS*-триггера

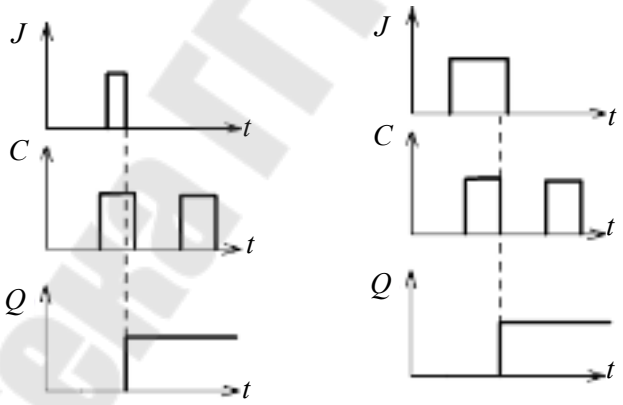

Рис. 44. Временные диаграммы *JK*-триггера
с динамическим управлением

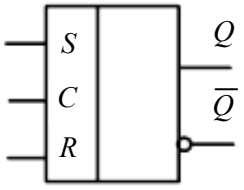

Рис. 45. Условное обозначение *RS*-триггера
со статическим управлением

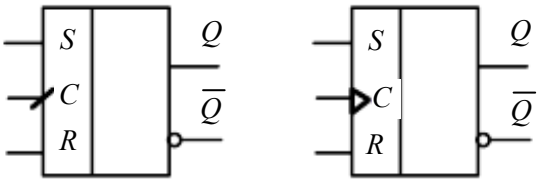

Рис. 46. Условное обозначение триггеров, переключающихся по положительному фронту (прямой динамический вход)

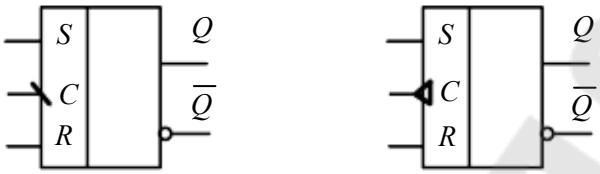

Рис. 47. Условное обозначение триггеров, переключающихся по отрицательному фронту (инверсный динамический вход)

17.20. Синхронные *MS*-триггеры

MS-триггеры (*Master* – ведущий, *Slave* – ведомый) – триггеры с двухступенчатым запоминанием информации. Оба триггера – *M* и *S* – функционируют как синхронные триггеры со статическим управлением. Если на синхровход подать «1», ведущий триггер *M* устанавливается в новое состояние в соответствии с сигналом, поступающим на его вход. Ведомый триггер имеет инверсный синхровход и невосприимчив к информации, поступающей на его вход с выхода триггера *M*. При изменении значения сигнала на синхровходе с «1» на «0» ведущий *M*-триггер отключается от информационных входов, а ведомый триггер *S* устанавливается в состояние, определенное выходом *M*-триггера.

Таким образом, управление в *MS*-триггере осуществляется двумя фронтами сигнала на синхровходе: на положительном фронте происходит установление ведущего триггера *M*, на отрицательном – ведомого *S*-триггера.

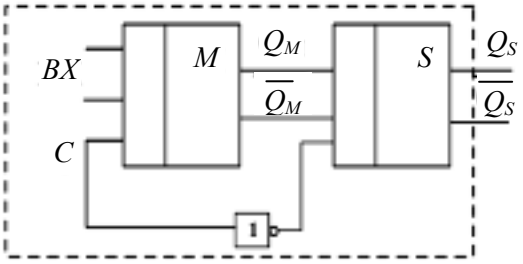

Рис. 48. Структурная схема двухступенчатого *MS*-триггера

В качестве простейшего примера рассмотрим построение счетного триггера со статическим управлением по принципу двухступенчатого запоминания информации.

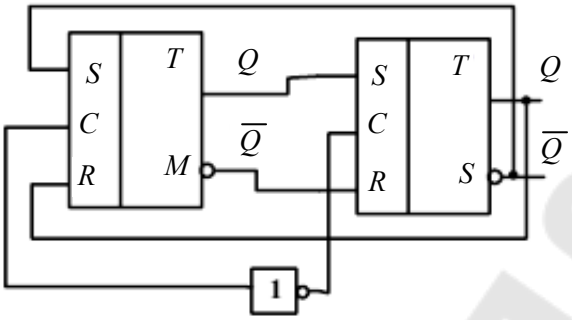

Рис. 49. Построение *MS*-триггера из двух *RS*-триггеров

При положительном фронте импульса, поступающего на вход *C*, триггер *M* устанавливается в состояние, противоположное состоянию *S*-триггера. При отрицательном фронте входного импульса *C* происходит передача сигнала, соответствующего состоянию триггера *M*, в триггер *S*.

17.21. Итоговая классификация триггеров

Итоговая классификация триггеров представлена на рис. 50.

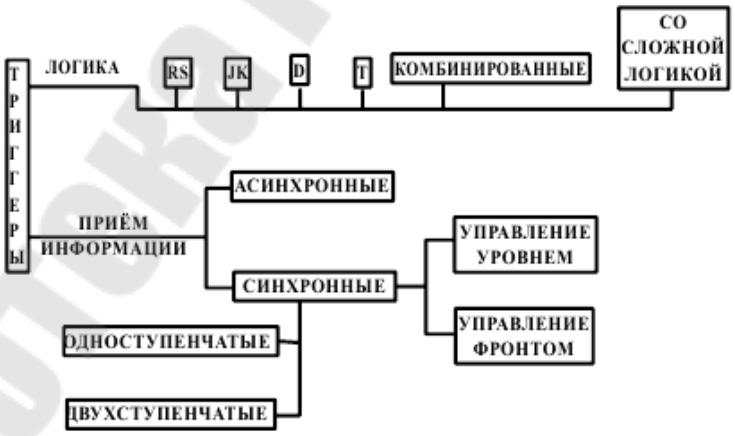

Рис. 50. Классификация триггеров

Литература

1. Галкин, В. И. Промышленная электроника / В. И. Галкин. – Минск : Выш. шк., 1989.

2. Гусев, В. Г. Электроника / В. Г. Гусев, Ю. М. Гусев. – М. : Высш. шк., 1991.

3. Гутников, В. С. Интегральная электроника в измерительных устройствах / В. С. Гутников. – Л. : Энергоатомиздат, 1990.

4. Шило, В. Л. Популярные цифровые микросхемы / В. Л. Шило. – М. : Металлургия, 1988.

5. Фролкин, В. Т. Импульсные цифровые устройства / В. Т. Фролкин, Л. Н. Попов. – М. : Радио и связь, 1992.

6. Быстров, Ю. А. Электронные цепи и устройства / Ю. А. Быстров, И. Г. Мироненко. – М. : Высш. шк., 1999.

7. Бирюков, С. А. Применение цифровых микросхем серий ТТЛ и КМОП / С. А Бирюков. – М. : МК, 2000.

8. Мнеян, М. Г. Физика машинной памяти / М. Г. Мнеян. – М. : Высш. шк., 1990.

9. Калабеков, Б. А. Цифровые устройства и микропроцессорные системы / Б. А. Калабеков. – М. : Горячая линия – телеком, 2000. – 336 с.

10. Угрюмов, Е. П. Цифровая схемотехника. БХВ / Е. П. Угрюмов. – Спб., 2000.

11. Токхейм, Р. Основы цифровой электроники / Р. Токхейм. – М. : Мир, 1998.

12. Кучумов, А. И. Электроника и схемотехника / А. И. Кучумов. – М. : Гелиос АРВ, 2002.

13. Новиков, Ю. В. Основы цифровой схемотехники / Ю. В. Новиков. – М. : Мир, 2001.

Содержание

Учебное электронное издание комбинированного распространения

Учебное издание

Кротенок Владимир Владимирович

ЛОГИЧЕСКИЕ И АРИФМЕТИЧЕСКИЕ ОСНОВЫ И ПРИНЦИПЫ РАБОТЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Курс лекций для студентов специальности 1-40 01 02 «Информационные системы и технологии (по направлениям)» дневной и заочной форм обучения

Электронный аналог печатного издания

Редактор *А. В. Власов*
Компьютерная верстка *Н. Б. Козловская*

Подписано в печать 14.05.12.
Формат 60x84/16. Бумага офсетная. Гарнитура «Таймс».
Ризография. Усл. печ. л. 3,95. Уч.-изд. л. 4,18.
Изд. № 8.
E-mail: ic@gstu.by
http://www.gstu.by

Издатель и полиграфическое исполнение:
Издательский центр Учреждения образования
«Гомельский государственный технический университет
имени П. О. Сухого».
ЛИ № 02330/0549424 от 08.04.2009 г.
246746, г. Гомель, пр. Октября, 48