Учебно-воспитательный (технологический) комплекс N326
Технологическая школа ОРТ

А.В.Шанин

ЦИФРОВАЯ ЭЛЕКТРОНИКА

Учебное пособие по курсу “технология”
для старших классов средней школы

РИК Русанова
МОСКВА, 1998

ББК 32.965
Ш22
УДК 681.32

Рекомендовано к изданию методическим советом АНО "Образовательные ресурсы и технологический тренинг" (ОРТ).
Рецензент: к.ф.-м.н. Горинский С.Г.

ISBN 5-87414-123-5

Оглавление

Часть I

Двоичная арифметика и логика

Глава 1

Двоичная система счисления и некоторые другие представления чисел

Занятие 1

Положительное целое число в двоичной системе счисления; перевод из одной системы счисления в другую; сравнение двоичных чисел

Вычислительная техника использует числа в форме, отличающейся от традиционной десятичной записи. В этой главе описаны различные способы представления чисел в ЭВМ и алгоритмы выполнения арифметических операций с ними.

Прежде всего, рассмотрим **строение десятичной системы счисления**. Как будет видно из дальнейшего, все операции с двоичными числами производятся так же, как и с десятичными. Рассмотрим какое-нибудь положительное целое число, например, 5078. Эта запись представляет собой сумму

$$5078 = 5 \times 10^3 + 0 \times 10^2 + 7 \times 10^1 + 8 \times 10^0$$

Очевидно, особую роль в этой записи играет число 10. Во-первых, существует всего 10 десятичных цифр от 0 до 9. Во-вторых, каждый разряд в числе является множителем при соответствующей степени десятки. Способ записи числа теми или иными значками называется системой счисления. Десятичная система является позиционной, т.е. "вес" каждого знака числа зависит от места, которое этот знак занимает. Существуют системы счисления (например, римская), формирующиеся по другим принципам. Число различных цифр называется *основанием* позиционной

системы счисления. Степени основания системы являются множителями при цифрах числа.

Все сказанное позволяет легко построить **двоичную систему счисления**, т.е. систему с основанием 2. Прежде всего, существуют всего 2 цифры: 0 и 1. Запись целого положительного двоичного числа представляет собой сумму степеней двойки, умноженных на соответствующие цифры, например:

$$1010111_2 = 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

Значок 2 справа и снизу от числа указывает на то, что число записано в двоичной системе. В отсутствие этого значка имеются в виду обычные десятичные числа. Помимо значка 2 в уголке, на двоичную запись может указывать латинская буква b (от слова binary) спереди или сзади от числа. Такую запись можно встретить в книгах по программированию (и в текстах программ).

Обратимся к некоторым частным случаям двоичных чисел. Будем действовать по аналогии с десятичной системой. Числа вида $1000\ldots0$ представляют собой степени десятки: 10^n. Точно так же, $1000\ldots0_2 = 2^n$.

Уместно рекомендовать запомнить первые 10 степеней двойки:

$$\begin{array}{ll} 2^0 = 1 & 2^5 = 32 \\ 2^1 = 2 & 2^6 = 64 \\ 2^2 = 4 & 2^7 = 128 \\ 2^3 = 8 & 2^8 = 256 \\ 2^4 = 16 & 2^9 = 512 \end{array}$$

$$2^{10} = 1024$$

С помощью этой таблицы легко **переводить двоичные числа в десятичные**. Для этого достаточно над каждой единицей в числе написать соответствующую степень двойки и просуммировать степени, стоящие над единицами:

$$\begin{array}{ccccccccccccccc} 64 & + & (32) & + & 16 & + & (8) & + & 4 & + & 2 & + & 1 & = & 47 \\ 1 & & 0 & & 1 & & 0 & & 1 & & 1 & & 1 & & \end{array}$$

(Числа в скобках не участвуют в сумме).

Продолжим строить аналогию между десятичной и двоичной системами. В десятичной системе

$$\underbrace{99\ldots9}_{n} = 10^n - 1,$$

а в двоичной

$$\underbrace{111\ldots1}_{n}\,\mathrm{b} = 2^n - 1$$

(Отметим, что $2^n - 1 = (10^n - 1)\,\mathrm{b}$, т.е. правые части равенств вообще не отличаются).

Примеры:

$$1\,\mathrm{b} = 2^1 - 1 = 1,$$

$$11\,\mathrm{b} = 2^2 - 1 = 3,$$

$$111\,\mathrm{b} = 2^3 - 1 = 7.$$

Выше был рассмотрен способ перевода чисел из десятичной системы в двоичную с помощью суммирования. К сожалению, этот способ не так удобен для перевода из **десятичной системы в двоичную**. Для этого рекомендуется использовать более сложный способ. К его описанию приступим издалека.

Прежде всего, заметим, что двоичное число, заканчивающееся нулем (имеющее младший, крайний правый разряд, равный нулю), четно. Если же младший разряд равен 1, то число нечетно. Также, если двоичное число заканчивается двумя нулями, то оно делится на $2^2 = 4$. Очевидна аналогия с десятичной системой, где число, кончающееся n нулями, делится на 10^n.

Пусть имеется некоторое десятичное число, например 23. Исходя из сказанного выше, последняя цифра записи этого числа в двоичной системе равна единице (т. к. число нечетно). Таким образом,

$$23 = 22 + 1 = 2 \times 11 + 1.$$

Двоичная запись числа 11 также имеет последнюю цифру 1:

$$23 = 2 \times (2 \times 5 + 1) + 1.$$

Двоичная запись числа 5 имеет последнюю цифру 1:

$$23 = 2 \times (2 \times (4 + 1) + 1) + 1 = 2 \times (2 \times (2 \times 2 + 1) + 1) + 1 =$$

$$= 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 10111\,\mathrm{b}.$$

Внимательный читатель мог заметить, что процедура, описанная выше, представляла собой ни что иное, как последовательное деление на 2 с остатком. Формальное описание процедуры таково.

Разделим исходное число на 2 с остатком. Остаток будет представлять собой младшую двоичную цифру. Частное снова разделим на 2 с

остатком и т.д. Остановиться надо в тот момент, когда частное станет равным нулю, т.е. деление дальше не имеет смысла (все остатки и частные будут равными нулю). Каждый следующий остаток будет давать следующий (старший) разряд числа. В качестве примера переведем число 221 в двоичную систему счисления:

$$\begin{aligned}
221 &= 2 \times 110 + \underline{1} \quad \text{— младший разряд,}\\
110 &= 2 \times 55 + \underline{0},\\
55 &= 2 \times 27 + \underline{1},\\
27 &= 2 \times 13 + \underline{1},\\
13 &= 2 \times 6 + \underline{1},\\
6 &= 2 \times 3 + \underline{0},\\
3 &= 2 \times 1 + \underline{1},\\
1 &= 2 \times 0 + \underline{1} \quad \text{— старший разряд.}
\end{aligned}$$

$$221 = 11011101\,\mathrm{b}.$$

Правильность вычислений можно проверить суммированием (обратным переводом). Еще раз хотелось бы отметить, что остатки представляют собой двоичные разряды числа, начиная с младшего.

Приведем **еще один способ перевода чисел из десятичной системы в двоичную**. Возьмем число, например 221. Выясним, какая наибольшая степень двойки помещается в этом числе. Это 128:

$$221 = 128 + 93.$$

Далее найдем наибольшую степень двойки, которая помещается в числе 93 и т.д.:

$$221 = 128 + 93 = 128 + 64 + 29 = 128 + 64 + 16 + 13 =$$

$$= 128 + 64 + 16 + 8 + 5 = 128 + 64 + 16 + 8 + 4 + 1.$$

В результате проделанной процедуры исходное число оказывается представленным в виде суммы степеней двойки.

Далее выпишем последовательные степени двойки справа налево (начиная с нулевой!) и под каждой степенью поставим 0 или 1 в зависимости от того, встречается ли данная степень в разложении исходного числа:

128,	64,	32,	16,	8,	4,	2,	1
1	1	0	1	1	1	0	1

Получается представление исходного числа в двоичной системе.

Обратимся к вопросу о **сравнении двоичных чисел**. Процедура сравнения двоичных чисел такая же, как и для десятичных чисел. Во-первых, больше то число, которое имеет больше разрядов (которое длиннее). Если числа имеют одинаковую длину (одинаковое число разрядов), будем сравнивать разряды чисел, начиная со старшего, до тех пор, пока в каком-либо разряде не встретятся различные цифры. Большим является то число, у которого цифра больше. Отметим, что в описанном алгоритме вообще никак на встречается указание на систему счисления, т.е. безразлично, десятичные это числа, двоичные или какие-либо еще.

Подчеркнем, что все сказанное относится к целым положительным числам.

ВОПРОСЫ И ЗАДАЧИ

1. Какие цифры могут встречаться в числе, записанном в троичной системе (в позиционной системе счисления с основанием 3)? Как перевести десятичное число в троичное и наоборот?

2. Сколько целых чисел k обладает свойством $0 \le k \le \underbrace{11\ldots1}_{n}$?

3. Как найти следующее и предыдущее число для данного двоичного числа? Найти следующее и предыдущее число для чисел
а) 1011000 b,
б) 1011111 b,
в) 1010101 b.

4. Сравнить числа
а) 1011011 b и 93;
б) 10111011101 b и 111010111 b.

5. а) Перевести в десятичную систему числа
110110110 b, 101011011 b,
б) Перевести в двоичную систему числа
270, 312.

6. Почему число различных цифр в произвольной позиционной системе счисления должно быть равно основанию степеней, являющихся весами при различных цифрах?

Пусть имеются цифры 0, 1, 2, 3 ... M ($M+1$ — число различных цифр, а число $\overline{abcd}$ есть сумма $d \times N^0 + c \times N^1 + b \times N^2 + a \times N^3$.

(Черта над буквами означает, что это не произведение, а запись числа. N - основание степеней). Рассмотреть случаи
а) $N > M + 1$,
б) $N < M + 1$.

Все ли целые числа могут быть представлены и все ли представляются единственным способом?

Занятие 2

Двоичные дроби; числа с плавающей запятой; шестнадцатиричное представление двоичных чисел; двоично-десятичное представление чисел

Обратимся к построению **дробных чисел** в двоичной арифметике. Прототипом, как и раньше, будет десятичная система. Рассмотрим десятичную дробь, например, 50.678. Эта дробь представляет собой сокращенную запись суммы :

$$50.678 = 5 \times 10^1 + 0 \times 10^0 + 6 \times 10^{-1} + 7 \times 10^{-2} + 8 \times 10^{-3}.$$

Точно так же, двоичную дробь $10.111\,\mathrm{b}$ будем расшифровывать как

$$10.111\,\mathrm{b} = 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} = 2.875,$$

т.е. каждый следующий разряд после запятой умножается на очередную отрицательную степень двойки. Уместно напомнить, что

$$\begin{aligned} 2^{-1} &= 1/2 = 0.5, \\ 2^{-2} &= 1/4 = 0.25, \\ 2^{-3} &= 1/8 = 0.125, \\ 2^{-4} &= 1/16 = 0.0625\ldots \end{aligned}$$

Тем самым решается вопрос о переводе двоичной дроби в десятичную. Так же, как и в десятичной системе, существуют бесконечные и периодические двоичные дроби.

Та часть числа, что стоит слева от десятичной точки, является его целой частью, а та, что справа, — дробной. Если одно и то же число записано в двоичной и десятичной системе, то целая часть одной записи соответствует целой части другой, а дробная — дробной.

Ранее была описана процедура перевода целой части числа из десятичной системы в двоичную. Выясним, как **перевести дробную десятичную часть в двоичную систему счисления**. Отметим, что

аналогично десятичным дробям, перемещение запятой в двоичной дроби на один разряд вправо соответствует умножению числа на 2, а на один разряд влево — делению числа на 2. Предположим, что дробная часть десятичного числа уже переведена в двоичную систему и представляет собой дробь $0.1101\ldots\,\mathrm{b}$. Умножим эту дробь на 2 (перенесем запятую вправо на 1 разряд). Получится число $1.101\ldots\,\mathrm{b}$. Заметим, что первый двоичный знак после запятой превратился в его целую часть. Отбросим его и снова умножим число на 2. Получится $1.01\ldots\,\mathrm{b}$. Теперь второй разряд после запятой исходной двоичной дроби превратился в целую часть. Снова отбросим целую часть и умножим число на 2 и т.д.

Двоичные разряды будут по очереди появляться слева от запятой.

Теперь заметим, что описанная процедура (умножать на 2 и отбрасывать целую часть) может быть проделана и с исходной десятичной дробью. Поясним это на примере. Пусть дана десятичная дробь 0.625. Ей соответствует некая двоичная дробь

$$0.xyz\,\mathrm{b},$$

которую надо найти (x, y, z могут быть равны 0 или 1). Итак,

$$0.625 = 0.xyz\,\mathrm{b}.$$

Умножим обе части равенства на 2:

$$1.25 = x.yz\,\mathrm{b}.$$

Приравнивая отдельно целую и дробную часть равенства, получаем

$$x = 1, \qquad 0.25 = 0.yz\,\mathrm{b}.$$

Умножаем обе части последнего равенства на 2. Получаем

$$0.5 = y.z\,\mathrm{b},$$

откуда

$$y = 0\,\mathrm{b}, \qquad 0.5 = 0.z\,\mathrm{b}.$$

Наконец,

$$z = 1\,\mathrm{b}.$$

Итак,

$$0.625 = 0.101\,\mathrm{b}.$$

Рассмотрим более сложный пример. Переведем в двоичную систему дробь 0.7:

$$\begin{aligned} 0.7 \times 2 &= \underline{1}.4, \\ 0.4 \times 2 &= \underline{0}.8, \\ 0.8 \times 2 &= \underline{1}.6, \\ 0.6 \times 2 &= \underline{1}.2, \\ 0.2 \times 2 &= \underline{0}.4, \\ 0.8 \times 2 &= \underline{1}.6, \\ &\ldots \end{aligned}$$

$$0.7 = 0.1(0110)\,\mathrm{b}.$$

То, что целая часть произведений подчеркивалась, означало, что эта целая часть записывалась в качестве очередного разряда двоичной дроби и в дальнейших умножениях не участвовала (отбрасывалась). Скобочки обозначают период дроби.

Итак, чтобы перевести некоторое десятичное число в двоичную систему, необходимо сначала перевести его целую часть при помощи деления с остатком, а затем дробную часть при помощи умножения с отбрасыванием.

Положительным десятичным числам, записанным в нормальной форме ("стандартном виде") соответствуют **двоичные числа с плавающей запятой**. Число представляется в виде $M \times 2^P$, где M — мантисса, обладающая свойством $1/2 < M < 1$, P — порядок. В этом формате компьютер хранит два числа (M и P), и арифметические действия проделывает с ними по отдельности по особым правилам. Далее мы не будем касаться этих вопросов.

Очевидным недостатком двоичной системы счисления является то, что сравнительно небольшие числа выглядят слишком длинными. Например, число $255 = 11111111\,\mathrm{b}$ содержит целых восемь разрядов. Для удобства записи двоичных чисел обычно используют еще одну — **шестнадцатиричную систему счисления**.

Казалось бы, использование еще одной системы счисления усложнит всю арифметику — понадобится переводить числа из одной системы в другую и т.д. Однако, это не так.

Прежде всего, в 16-ричной системе необходимо 16 различных цифр. Первые 10 известны (0, 1, 2 ... 9). Откуда же взять еще 6? В качестве цифры, обозначающей число 10 берут цифру A, 11—B, 12—C, 13—D, 14—E, 15—F. Число 16 в шестнадцатиричной системе счисления выглядит как

10. Для того, чтобы отличать десятичные числа от шестнадцатиричных, используют следующие обозначения: либо перед 16-ричным числом ставят хорошо знакомый символ \$, либо после числа — символ h.

Как и в любой позиционной системе счисления, каждое число представляет собой сумму:

$$\mathrm{CE01\,h} = 12 \times 16^3 + 14 \times 16^2 + 0 \times 16^1 + 1 \times 16^0.$$

Опишем теперь, как быстро и просто **перевести число из двоичной системы в 16-ричную**. Для этого разобьем биты двоичного числа на четверки, начиная от запятой и добавляя нули при необходимости перед числом и после него. Например, число 1011101.101 b запишется как

$$\underline{0101}\,\underline{1101}.\underline{1010}$$

После этого заменим каждую четверку 16-ричной цифрой по таблице. Эту таблицу настоятельно рекомендуется запомнить наизусть.

двоичное	16-ричное	10-чное		двоичное	16-ричное	10-чное
0000	0	0		1000	8	8
0001	1	1		1001	9	9
0010	2	2		1010	A	10
0011	3	3		1011	B	11
0100	4	4		1100	C	12
0101	5	5		1101	D	13
0110	6	6		1110	E	14
0111	7	7		1111	F	15

Например, $\mathrm{1011101.101\,b} = \mathrm{0101\,1101.1010\,b} = \mathrm{5D.A\,h}$.

Для того, чтобы **перевести число из 16-ричной системы в двоичную**, необходимо произвести обратную подстановку, т.е. вместо 16-ричной цифры подставить четверку двоичных знаков по таблице.

Обратим внимание на то, что вместо цифры 0 необходимо вставить 0000.

Выполнять арифметические операции в 16-ричной системе не слишком удобно, поэтому в дальнейшем мы будем пользоваться этой системой просто как краткой записью соответствующих двоичных чисел.

Последнее, о чем необходимо упомянуть в этом разделе курса, посвященном различным представлениям чисел, является **двоично-десятичное представление** (BCD). Этот вид чисел используется только для индикации (вывода чисел на дисплей) или передачи информации. Двоично-десятичное представление получается, если каждую десятичную цифру

числа заменить четверкой двоичных цифр по приведенной выше таблице. Например,

$$99 = 1001\,1001\,\mathrm{BCD}.$$

Обратите внимание, что 99 — обычное десятичное число.

Использовать такую форму записи чисел при выполнении арифметических действий почти невозможно.

ВОПРОСЫ И ЗАДАЧИ

1. Почему при переводе десятичной дроби в двоичную с помощью умножения целая часть всегда получается равной 0 или 1 ?

2. Получится ли одинаковый результат, если некоторое число перевести из 10-чной системы в 16-ричную двумя способами: а) при помощи последовательных делений на 16 с остатком, б) сначала перевести в двоичную, а затем заменить цифры по таблице?

3. Показать, что двоичная и двоично-десятичная формы записи не совпадают.

4. Перевести дробь в двоичную систему с точностью до 5 знака после лвоичной точки
21.356, 52.121.

5. Перевести в 16-ричную систему числа
а) 10101011.0110101 b,
б) 322.

6. Перевести в десятичную систему числа (один из способов — сначала перевести в двоичную, а затем в десятичную)
C0FE h, ABBA h.

7. Какие дроби (в обычном понимании) изображаются конечными двоичными дробями?

Глава 2

Арифметические операции в двоичной системе

Занятие 3

Разрядная сетка; сложение двоичных чисел; вычитание; переполнение

Разумеется, двоичные числа могут иметь сколь угодно много знаков как до запятой, так и после. Однако все не так просто, если мы имеем дело с компьютером.

Прибор, который выполняет арифметические операции, называется АЛУ (арифметико – логическое устройство) и входит в процессор. Числа приходят в процессор по проводам. На каждом из проводов могут быть напряжения (относительно некоторого "общего" провода) двух типов: обозначающие логическую единицу (обычно это 5 В) и логический ноль (0 В). Один из проводов переносит младший разряд числа (нулевой), другой — первый разряд и т.д. Совокупность этих проводов называется *шиной данных*.

Если число передается по 8 проводам и содержит 8 разрядов, то процессор называется 8-разрядным. Каждую операцию он выполняет с 8-разрядными числами. Если в этом случае возникает необходимость сложить два 16-разрядных числа, их необходимо разбить на два куска по 8 разрядов и выполнить с ними несколько операций по определенным правилам, что весьма хлопотно.

Бывают 8, 16 и 32-разрядные процессоры, а также некоторые другие (например, простые микрокалькуляторы имеют 4-разрядные АЛУ). Кроме того, 16-тиразрядный процессор может иметь 8-разрядную шину данных. В этом случае каждое число передается в два приема: сна-

чала одна половина, потом другая (так работает, например, процессор i8088). Процессор i80386SX 32-разрядный, однако имеет 16-разрядную шину данных.

Набор разрядов числа называется **разрядной сеткой**. Например, для 8-разрядного процессора разрядная сетка состоит из 8 позиций. Каждая из позиций — место для двоичного разряда. Младший разряд обычно имеет индекс 0 (A_0, B_0, D_0 или др.), старший — индекс 7.

D =

1	0	1	1	0	1	0	1
D_7							D_0

Говоря о разрядной сетке, необходимо определить, в каком месте находится двоичная точка (если операции выполняются с целыми числами, очевидно, она находится сразу после младшего разряда), а также выяснить, имеет ли число знак (т.е. все числа могут считаться положительными, а могут быть положительными и отрицательными. Об этом см. ниже.) Положение двоичной точки и наличие знака определяется не возможностями процессора. Программист должен решить для себя эти вопросы, а затем учитывать эти особенности при написании программ. Обычно процессор считает все числа целыми и допускает как беззнаковое, так и знаковое представление чисел. В дальнейшем все примеры будут приводиться для 8-разрядной сетки.

Формулы комбинаторики свидетельствуют о том, что существует 2^n различных комбинаций из n цифр, каждая из которых может принимать значения 0 и 1. (Сравните этот факт с результатом задачи 2 к занятию 1).

Обратимся к выполнению арифметических операций в двоичной системе. Прежде всего, проследим, как выполняется **сложение положительных чисел**.

Проще всего складывать маленькие числа. Составим следующую табличку (все числа записаны в двоичной системе):

$$\begin{aligned} 0+0 &= 0 \\ 0+1 &= 1 \\ 1+1 &= 10 \\ 1+1+1 &= 11 \end{aligned}$$

(Убедитесь в справедливости этих равенств :).

Более длинные числа складываются при помощи этой таблицы столбиком. Пусть необходимо сложить числа 01011101 и 01100111 (оба числа записаны в 8-разрядной сетке). Разместим эти числа одно под другим:

$$\begin{array}{rr} & 01011101 \\ + & \underline{01100111} \end{array}$$

Теперь сложим младшие разряды этих чисел. Результат, очевидно, $1_2 + 1_2 = 10_2$. Ноль запишем как младший разряд суммы, а единицу перенесем в следующий разряд:

$$\begin{array}{rr} & 0101110\dot{1}01 \\ + & \underline{01100111} \\ & 0 \end{array}$$

В следующем разряде приходится складывать $0_2 + 1_2 + 1_2 = 10_2$ (последнее слагаемое — результат переноса из младшего разряда). Опять, ноль — разряд суммы, а единица переносится в старший разряд:

$$\begin{array}{rr} & 0101110\dot{1}01 \\ + & \underline{01100111} \\ & 00 \end{array}$$

В третьем разряде необходимо сложить $1_2 + 1_2 + 1_2 = 11_2$

$$\begin{array}{rr} & 0101\dot{1}\dot{1}0\dot{1} \\ + & \underline{01100111} \\ & 100 \end{array}$$

и т.д. Результат операции:

$$\begin{array}{rrcr} & \dot{0}\dot{1}\dot{0}\dot{1}\dot{1}\dot{1}0\dot{1} & = & 93 \\ + & \underline{01100111} & = & 103 \\ & 11000100 & = & 196 \end{array}$$

Нетрудно заметить, что процедура сложения столбиком полностью аналогична той, что проходят в 1 классе.

В некоторых случаях (как, например, при сложении чисел 10110000 b и 11000000 b), результат сложения может не уложиться в 8 разрядов. В таком случае говорят о *переполнении разрядной сетки*. Процессор удерживает лишь 8 младших разрядов и при этом сигнализирует о переполнении. Учитывая в программе этот сигнал, можно внести коррективы в вычисления.

Оформим процедуру сложения формально. Для этого построим следующую таблицу:

$$\begin{array}{rcrcrcrcr} 0 & + & 0 & & & = & 0 & & \\ 0 & + & 1 & & & = & 1 & & \\ 1 & + & 1 & & & = & 0 & + & 10 \\ 1 & + & 1 & + & 1 & = & 1 & + & 10 \end{array}$$

Попытайтесь понять, где в этой таблице скрыт разряд суммы, а где перенос в следующий разряд.

Построим теперь похожую таблицу для **вычитания столбиком**. Эта таблица будет описывать вычитание в каждом разряде. Соответственно, персонажами этой таблицы будут разряд вычитаемого, разряд уменьшаемого, разряд разности, занимание, приходящее из предыдущего разряда, занимание, уходящее в следующий разряд:

0	-	0	-	0	=	0		
0	-	0	-	1	=	1	-	10
0	-	1	-	0	=	1	-	10
1	-	0	-	0	=	1		
1	-	1	-	0	=	0		
1	-	0	-	1	=	0		
1	-	1	-	1	=	1	-	10

Пользуясь этой таблицей, решим несколько примеров:

$$\begin{array}{rll} & \dot{1}011 & (11) \\ - & \underline{0110} & (6) \\ & 0101 & (5) \end{array} \qquad \begin{array}{rll} & \dot{1}\dot{0}\dot{0}0 & (8) \\ - & \underline{0111} & (7) \\ & 0001 & (1) \end{array} \qquad \begin{array}{rll} & \dot{1}0\dot{1}\dot{1}0 & (22) \\ - & \underline{01001} & (9) \\ & 01101 & (13) \end{array}$$

Процесс вычитания также похож на вычитание в 10-чной системе.

Когда приходится складывать или вычитать столбиком дробные двоичные числа, необходимо проследить, чтобы точка оказалась под точкой.

Отметим, что рассмотренные примеры касались только вычитания меньшего числа из большего.

ВОПРОСЫ И ЗАДАЧИ

1. Выполните сложение в двоичной системе
$70\,\mathrm{h} + 22\,\mathrm{h}$, $51\,\mathrm{h} + 67\,\mathrm{h}$, $\mathrm{F8\,h} + 81\,\mathrm{h}$.
В каком случае имеет место переполнение 8-разрядной сетки?

2. Выполните вычитание в двоичной системе
$70\,\mathrm{h} - 22\,\mathrm{h}$, $67\,\mathrm{h} - 51\,\mathrm{h}$, $\mathrm{F8\,h} - 81\,\mathrm{h}$.

3. Выполните сложение столбиком

$$\begin{array}{rl} & 101110 \\ & 011101 \\ + & 110101 \\ & \underline{110111} \end{array}$$

Занятие 4

Отрицательные числа в дополнительном коде; различные способы представления отрицательных чисел

Процедура вычитания, рассмотренная выше, допускала вычитание меньшего числа из большего (так, чтобы результат получался положительным. Попробуем применить ту же процедуру при условии, что **уменьшаемое меньше вычитаемого**. При этом будем учитывать, что вычисления производятся в 8-разрядном коде:

$$\begin{array}{cll} & \dot{0}\dot{0}\dot{0}01000 & (8) \\ - & \underline{00010000} & (16) \\ & 11111000 & (248) \end{array}$$

(В старшем разряде приходится занимать из отсутствующего девятого разряда. Это своеобразное переполнение разрядной сетки при вычитании).

Очевидно, результат 248 не соответствует ожидаемому -8. Однако, можно заметить, что он равен в точности $2^8 - 8$. Это видно хотя бы из того, что если в 9-й разряд уменьшаемого поставить 1 (это означает добавить к уменьшаемому $256 = 2^8$), то все вычисления станут полностью корректными. Будем считать полученное число 11111000 своеобразной формой записи отрицательного числа -8. Проверим, можно ли выполнять с этим числом какие-либо арифметические действия. Попробуем сложить 10 (1010 b) и -8 в 8-разрядной сетке:

$$\begin{array}{cll} & 00001010 & (10) \\ + & \underline{11111000} & (-8) \\ & 00000010 & (2) \end{array}$$

Перенос в 9-й разряд игнорируем. Удивительно, но получился правильный результат.

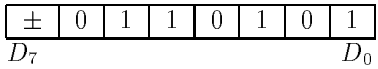
Проделаем более рискованную операцию: сложим $+6$ и -8:

$$\begin{array}{cll} & 00000110 & (6) \\ + & \underline{11111000} & (-8) \\ & 11111110 & (254 = 256 - 2) \end{array}$$

Получилось отрицательное число, записанное тем же способом: из 256 вычтен модуль числа.

Отметим, что в десятичной системе счисления отрицательные числа записывают вполне определенным способом: перед числом ставится знак

"–". В машинной записи это означает, что самый старший разряд становится знаковым. Обычно 0 в этом разряде означает, что число положительное, а 1 - отрицательное.



$D_7 = 0$ — число положительное,
$D_7 = 1$ — число отрицательное.

После знакового разряда можно записать модуль числа в двоичной системе (отметим, что на него останется на единицу меньше разрядов). Такая запись положительных и отрицательных чисел (знаковый разряд и за ним модуль числа) называется *прямым кодом*. Он неудобен для проведения арифметических вычислений, так как сложение двух положительных чисел будет производиться не так, как сложение положительного числа с отрицательным.

Именно поэтому в большинстве случаев применяют **дополнительный код**. Для 8-разрядной сетки этот код строится так. Положительные числа от 0 до $127 = 01111111\,\mathrm{b}$ записываются обычным образом в двоичной системе счисления. Отметим, что у всех этих чисел старший разряд равен 0. Отрицательные числа от -1 до -128 записываются следующим способом: вместо них берется дополнение их модуля до 256. У всех этих чисел старший разряд равен 1. Коротко алгоритм записывается так:

$$\text{Запись } A = \begin{cases} A, & \text{если} \quad 0 \le A \le 127 \\ 256 - |A|, & \text{если} \quad -128 \le A \le -1 \end{cases}$$

Заметим, что рассмотренные в начале этого занятия числа -8 и -2 записаны в дополнительном коде.

Так получается, что в дополнительном коде старший разряд играет роль знакового. Это дает возможность легко восстановить по записи числа в дополнительном коде само число. Восстановить число можно при помощи следующего алгоритма. Пусть B — запись числа A в дополнительном коде.

$$A = \begin{cases} B, & \text{если} \quad B_7 = 0 \\ -(256 - B), & \text{если} \quad B_7 = 1 \end{cases}$$

Так, число $00000101\,\mathrm{b}$ в дополнительном коде есть $+5$, а

$$10000101\,\mathrm{b} = -(256 - 133) = 123$$

Под "записью числа" мы понимаем то, что получится, если просто перевести 8-разрядное двоичное число в 10-чную систему.

Что же за число 11111000 b: 240 или -8? Это зависит от соглашения. Если в программе все числа предполагаются положительными, то это 248. Если же известно, что это число, записанное в дополнительном коде, то это -8. Процессору все равно, он производит сложение с числом 248 точно так же, как и с числом -8. При этом он информирует программиста о переполнении разрядной сетки и о некоторых других ситуациях. Реакция программы на эту информацию и определяет, считалось ли число положительным или отрицательным.

Еще одна полезная формула позволяет переводить отрицательные и положительные числа из дополнительного кода в десятичную систему:

$$\overline{A_7A_6A_5A_4A_3A_2A_1A_0} =$$

$$= -128A_7 + 64A_6 + 32A_5 + 16A_4 + 8A_3 + 4A_2 + 2A_1 + A_0$$

(обратите внимание на знак перед первым слагаемым).

ВОПРОСЫ И ЗАДАЧИ

1. Перевести в дополнительный двоичный код числа
5, 10, 13, -18, -100.

2. Перевести из дополнительного кода в 10-чную систему
F5 h, 61 h, 82 h.
Какие из этих чисел отрицательные?

3. Как выглядят в дополнительном коде числа
-1, -128?

4. Какое из чисел, записанных в дополнительном коде, больше?
а) F8 h или 84 h,
б) 65 h или 91 h.

5. Проделайте в дополнительном коде вычисления.
$61 - 75$, $61 + (-75)$, $(-10) - 13$, $-10 + (-13)$.
Результат перевести из дополнительного кода.

Занятие 5

Умножение и деление двоичных чисел

Умножение положительных двоичных чисел столбиком производится по обычным правилам, только немного проще. Пусть, к примеру, необходимо умножить число 101 b (5) на 110 b (6). Запишем одно число над другим:

$$\begin{array}{r} 110 \\ \underline{101} \end{array}$$

Умножим верхний множитель на младший разряд нижнего. Отметим, что умножать на двоичный разряд совсем не сложно. Если умножить число на 1, получится оно само, а если на 0, то 0.

Итак, в результате первого умножения будет

$$\begin{array}{r} 110 \\ \underline{101} \\ 110 \end{array}$$

Теперь умножим верхний множитель на второй разряд нижнего, причем результат сдвинем на 1 позицию:

$$\begin{array}{r} 110 \\ \underline{101} \\ 110 \\ 000 \end{array}$$

Ту же процедуру проделаем с 3 разрядом:

$$\begin{array}{r} 110 \\ \underline{101} \\ 110 \\ 000 \\ 110 \end{array}$$

Сложим столбиком полученные результаты. (Если чисел много, рекомендуется прибавлять по одному числу)

$$\begin{array}{rl} 110 & (6) \\ \underline{101} & (5) \\ 110 & \\ 000 & \\ \underline{110} & \\ 11110 & (30) \end{array}$$

Если множители имели дробную часть, необходимо отсчитать от младшего разряда столько двоичных знаков, сколько их после точки в обоих множителях, и поставить двоичную точку.

Деление также аналогично десятичному. Не будем комментировать, а лишь приведем пример вычислений:

```
11011   | 1011
1011    | 10.01110...
  10100
   1011
   10010
    1011
     1110
     1011
      ...
```

Особенно легко **умножать и делить на степени двойки**. Для того, чтобы умножить (поделить) двоичное число на 2, необходимо перенести запятую на n разрядов вправо (влево). В случае необходимости при этом добавить нужное количество нулей.

ВОПРОСЫ И ЗАДАЧИ

1. Сложить положительные числа и представить в шестнадцатиричном виде

$$\begin{array}{r} 1101101 \\ +\ 1011110 \\ \hline \end{array} \qquad \begin{array}{r} 1011011 \\ +\ 1100101 \\ \hline \end{array} \qquad \begin{array}{r} 011.1011 \\ +\ 111.1001 \\ \hline \end{array} \qquad \begin{array}{r} 101.0011 \\ +\ 010.101 \\ \hline \end{array}$$

2. а) Представить отрицательное число в 8-разрядном дополнительном коде:
$-65, \quad -48.$
б) Перевести число из дополнительного кода
FE h, 7D h, 5C h, EA h.

3. Проделать вычисления в двоичной системе
$77 + (-65), \quad 56 + (-48).$

4. Умножить в двоичной системе
$7 \times 11, \quad 13 \times 14.$

Глава 3

Логические операции

Занятие 6

Высказывания и их истинность; операции над высказываниями; основные логические элементы

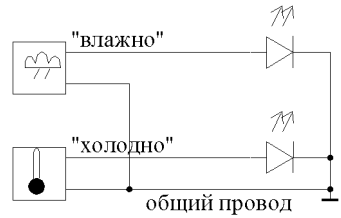
В этом разделе мы попытаемся дать представление о высказываниях, операциях над ними и о технической реализации — о логических сигналах и логических элементах. В сущности, весь последующий материал курса посвящен лишь все более и более сложным логическим функциям с большим числом переменных.

Представим себе "наблюдателя", сидящего в бункере глубоко под землей и желающего знать о погоде на поверхности. Наверху стоят датчики (любитель иностранных слов сказал бы "сенсоры", однако в отечественной технической литературе это понятие имеет несколько более узкий смысл) температуры, влажности и т.д. Пусть эти датчики очень просты, например, датчик температуры может передать не саму температуру, а лишь "тепло" или "холодно", датчик влажности — "дождь" или "сухо" и т.д.

Обычно датчики такого типа имеют 2 выходящих провода — "общий" и "сигнальный". (Иногда необходимо подвести еще и питание). Условимся, что если идет дождь, у датчика влажности напряжение между сигнальным и общим проводом 5 В, а если сухо, то 0 В.

Сигналы, принимающие только два значения, называют **логическими сигналами**.

Также, если холодно, между сигнальным и общим проводом датчика температуры 5 В, а если тепло — 0 В.



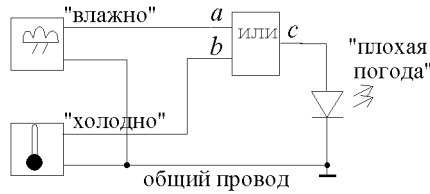
лог. 1	—	5 В	—	истинно
лог. 0	—	0 В	—	ложно

Чтобы не тянуть все провода, можно соединить общие провода непосредственно там, где стоят датчики и вести до бункера все сигнальные провода и один общий. Иногда общий провод называется шиной земли. В бункере к ним можно подсоединить светодиоды так, как показано на рисунке. Разумеется, светодиод горит, если к нему приложено напряжение. (В действительности, хорошо бы еще последовательно со светодиодом включить сопротивление $\sim 300\,\Omega$).

Если горит светодиод "дождь", значит, сработал датчик влажности и высказывание "идет дождь" истинно. Состояние провода, указывающее на истинность соответствующего высказывания, называют *логической единицей*, а на ложность — *логическим нулем*. Конкретные напряжения, указывающие на то и другое (в нашем случае 0 В и 5 В относительно общего) называют *логическими уровнями*.

Для простейшего анализа приходящей информации можно использовать *логические элементы*.

Предположим, что наблюдатель (с долей иронии можно назвать его "экологом") считает погоду плохой, если идет дождь **или** холодно (или и то и другое). В этом случае ему необходим такой прибор:



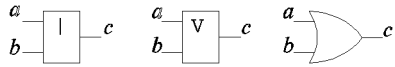
Прибор "ИЛИ" имеет два входа и один выход. На всех его входах и выходах могут быть только логические уровни напряжения (0 и 5 В относительно "земли"). Рассмотрим работу этого прибора. Очень легко

описать ее полностью. Для этого каждой комбинации логических уровней на входе (а их всего четыре) нужно сопоставить уровень на выходе. Составим таблицу, где входы слева, а выходы справа:

"ИЛИ"

a	b	c
0	0	0
0	1	1
1	0	1
1	1	1

Легко убедиться, что логическая единица на выходе появляется тогда, когда есть логическая единица на одном *или* на другом входе. Такую таблицу называют таблицей истинности. Элементы такого рода (все входы и выходы могут иметь лишь логические уровни) называют логическими элементами. Элемент "ИЛИ" могут обозначать так:



Соответственно, значки $\vee$, $|$ обозначают операцию "ИЛИ". Обратите внимание, что входы устройств всегда рисуют слева, а выходы справа.

К сожалению, общепринятого стандарта обозначений нет. Мы будем использовать последнее обозначение (оно встречается в большинстве иностраннных книжек). Однако вы можете встретить и другие обозначения из тех, что приведены на рисунке выше.

Операции с логическими сигналами соответствует операция с высказываниями. Пусть a — высказывание "идет дождь", b — "холодно", а c — "плохая погода". Используют следующие обозначения (на выбор)

$$\begin{aligned} c &= a \vee b \\ c &= a \,|\, b \\ c &= a + b \end{aligned}$$

Мы будем использовать последнее обозначение. Запомните, что знак $+$ есть просто обозначение слова "ИЛИ" (не путать с арифметическим сложением).

Операцию "ИЛИ" называют *логическим сложением* по причинам, которые станут ясны при изучении логических тождеств. Мы надеемся, что каждый раз будет ясно, какое сложение (арифметическое или логическое) имеется в виду. Обычно будет иметься в виду логическое. Во всяком случае, стоит запомнить, что это совершенно разные операции.

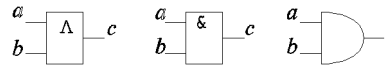
a, b и c можно рассматривать как переменные, которые могут принимать значения 0 (ложно) и 1 (истинно). Тогда логическое сложение — функция, имеющая 2 аргумента (вроде сложения или умножения). Узнать, как действует эта функция, можно из только что приведенной таблицы истинности.

Перейдем к описанию других логических функций. Пусть высказывание "очень плохая погода" истинно, когда идет дождь **и** холодно. Для того, чтобы заставить работать светодиод с надписью "очень плохая погода", нужен элемент "И" с такой таблицей истинности:

"И"

a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

Он обозначается так:

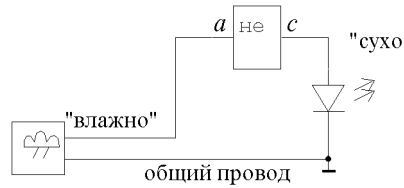


Соответственно, обозначения логической операции:

$$c = a \wedge b$$
$$c = a \,\&\, b$$
$$c = a \cdot b$$

Операцию "И" называют *логическим умножением*. Она также имеет мало общего с арифметическим умножением.

Чтобы заставить работать лампочку "сухо" (т.е. **не** влажно), нужен прибор "НЕ":

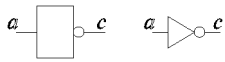


Его таблица истинности:

"НЕ"

a	c
0	1
1	0

Обозначают элемент "НЕ" следующим образом:



и пишут

$$c = \neg a$$
$$c = a'$$
$$c = \overline{a}$$

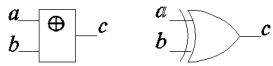
Такой прибор иногда называют инвертором.

Наконец, изощренный эколог может потребовать сигнал "погода плохая, но не очень", который равен логическое единице, если идет дождь или холодно, но не обе неприятности сразу. Для этого ему потребуется прибор, который называется **исключающее или**. Вот его таблица истинности:

"Исключающее ИЛИ"

a	b	c
0	0	0
0	1	1
1	0	1
1	1	0

Этот прибор обозначается так:



Логическая операция "исключающее ИЛИ" обозначается так:

$$c = a \oplus b$$

Важно понимать, что все описанные приборы не умозрительны, а очень широко встречаются на практике. Обычно они группируются (от 2–3 до десятков тысяч) и помещаются на микросхемы. Из них затем можно собирать разные хитрые устройства.

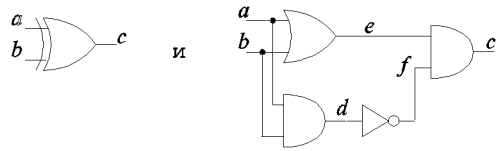
ВОПРОСЫ И ЗАДАЧИ

1. Нарисуйте из изученных приборов и сигналов схему, выдающую сигналы "неплохая погода" и "хорошая погода".

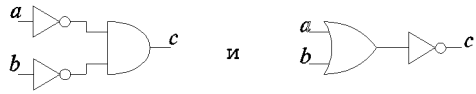
2. Пусть к сигналам a = "дождь", b ="холодно" добавили сигнал c = "пасмурно". Как будет выглядеть таблица истинности прибора со входами a, b, c и выходом d, если d = "плохая погода" и если d = "очень плохая погода".

3. Докажите (рассмотрев все возможные комбинации сигналов на входе), что два прибора работают совершенно одинаково:

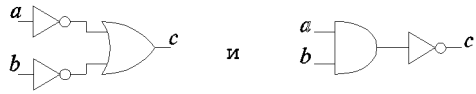
а)



б)



в)



Указание: для доказательства целесообразно составить таблицу

a	b	промежуточные точки	c
0	0		
0	1		
1	0		
1	1		

Здесь слева, как и раньше, стоят все возможные комбинации сигналов на входе, справа — выход, а в середине — последовательно вычисляемые сигналы в промежуточных точках (например, для пункта а) — в точках d, e, f).

Занятие 7

Логические формулы и высказывания; формулы и схемы; правила построения схем

Из материалов предыдущего занятия (а особенно из упражнений) уже стала ясна **связь между логическими формулами и логическими высказываниями**. Рассмотрим эту связь подробнее.

Пусть a — высказывание "холодно", b — высказывание "идет дождь", а c — "холодно или идет дождь, но не то и другое сразу". Тогда

$$c = (a + b) \cdot \overline{(a \cdot b)}.$$

Логика языка непосредственно указывает способ построения логической формулы. Обратите внимание на то, что "но" обычно означает то же, что и "И", а также на расположение скобок.

Большого числа скобок избегают так же, как и в арифметике, давая действиям определенный приоритет. Операция "НЕ" имеет самый старший приоритет (выполняется первой). Далее идет "И" ($\cdot$), а за ней "ИЛИ" ($+$). Операция $\oplus$ имеет самый низкий приоритет.

Операцию $\oplus$ иногда называют сложением по модулю 2. Смысл этого название вот в чем. Пусть 1 и 0 — не логические "истинно" и "ложно", а арифметические остатки некоторых чисел от деления на 2 ("нечет" и "чет"). Сравним таблицы:

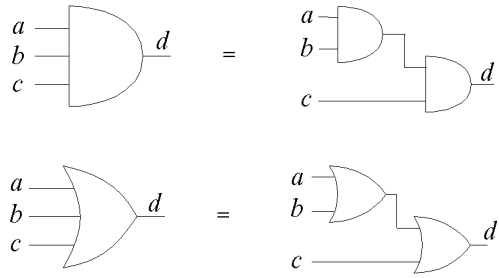
0	$\oplus$	0	=	0	чет	+	чет	=	чет
0	$\oplus$	1	=	1	чет	+	нечет	=	нечет
1	$\oplus$	0	=	1	нечет	+	чет	=	нечет
1	$\oplus$	1	=	0	нечет	+	нечет	=	чет

Очевидно, что сложение остатков соответствует логической операции $\oplus$.

Кроме того, важным свойством операции $\oplus$ является то, что когда она выполняется с одинаковыми символами ($1 \oplus 1$ или $0 \oplus 0$), результат равен 0, а когда с разными ($1 \oplus 0$ или $0 \oplus 1$), результат равен 1. В дальнейшем это поможет проверять равенство двоичных чисел.

Отметим, что результат **последовательного выполнения нескольких операций** "И" ($a \cdot b \cdot c$), а также "ИЛИ" ($a+b+c$), не зависит от последовательности их выполнения (см. упражнение 1). Результат выполнения операции $a \cdot b \cdot c$ равен единице, когда все переменные равны 1, в противном случае, результат равен 0. Для множественного "ИЛИ" ($a + b + c$) ситуация обратная — результат равен единице во всех случаях, кроме того,

когда все переменные равны 0. Этим операциям соответствуют сложные элементы:



Вернемся к построению логических формул. Важным случаем формулы является **логическое тождество**. Например, всегда верно, что $a + \overline{a} = 1$. Для доказательства этого тождества построим таблицу:

a	$\overline{a}$	$a + \overline{a}$
0	1	1
1	0	1

(Первый столбец строится с помощью таблицы истинности операции “НЕ”, второй — “ИЛИ”). Докажем более сложное тождество

$$a \cdot c + b \cdot c = (a + b) \cdot c$$

Построим таблицу:

a	b	c	$a \cdot c$	$b \cdot c$	$a \cdot c + b \cdot c$	$a + b$	$(a + b) \cdot c$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	1	0
0	1	1	0	1	1	1	1
1	0	0	0	0	0	1	0
1	0	1	1	0	1	1	1
1	1	0	0	0	0	1	0
1	1	1	1	1	1	1	1

Полное соответствие четвертого и шестого столбцов, изображающих правую и левую части тождества, указывает на то, что при любой комбинации a, b и c тождество верно.

Обратите внимание на способ, с помощью которого в левой колонке таблицы истинности перебираются все возможные комбинации a, b

и c. Тройки этих переменных представляют собой *последовательно все 3-разрядные двоичные числа*. Как доказывается в комбинаторике, количество различных комбинаций n переменных, принимающих значения 0 и 1, равно 2^n. Сравните этот результат с упражнением 2 занятия 1.

Если переменные, входящие в выражение, принимают определенные значения, появляется возможность найти значение выражения. (В учебнике математики младших классов есть упражнения типа "найти значение выражения при ...").

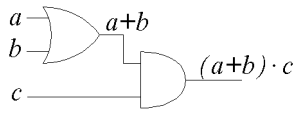
Легко видеть, что в левой колонке стоят комбинации переменных, а в остальных — значения выражений, соответствующие данным комбинациям.

Единственная хитрость в доказательстве тождеств при помощи таблицы истинности заключается в том, чтобы выбрать промежуточные колонки в таблице (в этом примере это колонки $a \cdot c$, $b \cdot c$, $a + b$). Выбирать промежуточные колонки надо с таким расчетом, чтобы каждый следующий результат получался из предыдущих, а в конце цепочки стояло выражение, равное правой или левой части тождества.

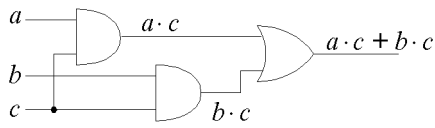
Рассмотрим теперь **связь между логическими формулами и логическими схемами.** Эта связь взаимно однозначна. Для каждого логического выражения рассмотрим схему, входами которой являются сигналы — переменные (a, b, c ...), а выходом — сигнал со значением, равным значению данного выражения.

Вместо того, чтобы объяснять, как строить такие схемы, рассмотрим несколько примеров, а затем сформулируем некоторые правила.

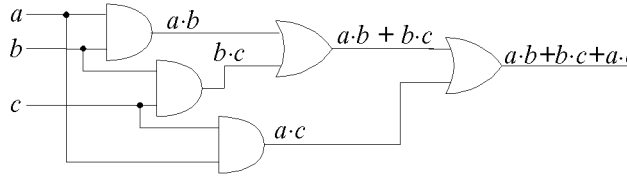
$(a + b) \cdot c$:



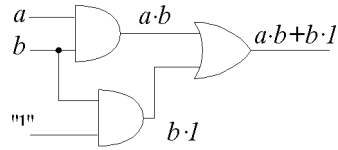
$a \cdot c + b \cdot c$:



$a \cdot b + b \cdot c + a \cdot c$:



Вместо одной из переменных может стоять ее значение — лог. 0 или 1. $a \cdot b + b \cdot 1$:



Для логической схемы, изображенной на последнем рисунке, это означает, что один из ее входов подсоединен к источнику напряжения с заданным логическим уровнем (0 В или +5 В).

Сформулируем **правила работы с логическими схемами.**

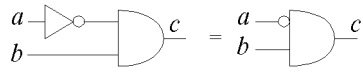
- Каждый вход логической схемы должен быть соединен либо с входным сигналом, либо с выходом какого-либо логического элемента схемы, либо с источником напряжения 0 или 1.

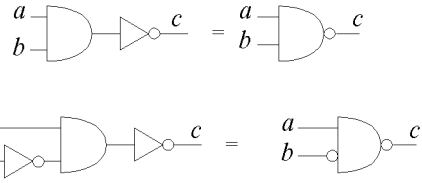
- Нельзя соединять вместе 2 (или более) выхода логических элементов (если на них появятся разные значения, неизвестно, что получится).

- Можно объединять сколько угодно входов элементов вместе (в реальных схемах число входов, подсоединенных к одному выходу обычно не должно превышать 10).

Приведем несколько полезных **правил рисования логических схем**.

- Обычно инверторы не рисуют в виде отдельных устройств. Если инвертор стоит до или после другого логического элемента, инвертор изображают кружочком:





- Пересечение проводов изображенное так:

не предполагает соединения проводов в точке пересечения.

А такое пересечение:

обозначает соединение проводов.

- Если несколько проводов выполняют близкие функции (например, несут разряды одного числа), обычно нет необходимости рисовать их все. В таких случаях рисуют *шину*:

При этом пучки проводов рисуют толстой линией, а провода помечают на входе и выходе.

- Нет необходимости рисовать общий провод (землю). Вместо этого, точки, которые к нему подключаются, помечают уже известным значком:

ВОПРОСЫ И ЗАДАЧИ

1. Доказать тождества
$(a \cdot b) \cdot c = a \cdot (b \cdot c)$
$(a + b) + c = a + (b + c)$
(ассоциативность)

2. Доказать тождества
$\overline{a} + \overline{b} = \overline{a \cdot b}$
$\overline{a} \cdot \overline{b} = \overline{a + b}$
(Сравните с результатами упражнения к предыдущему занятию).

3. Нарисовать таблицу истинности и схему для формул
$(\overline{a + b} \cdot c) \cdot (\overline{a} + (b + \overline{c}))$
$(\overline{c} \cdot (b + \overline{a})) + (a + \overline{b \cdot c})$

Занятие 8

Синтез схем по логике их функционирования и по заданным таблицам истинности; КНФ и ДНФ

В настоящем занятии мы рассмотрим, как решается одна из основных задач, стоящих перед инженером, а именно, как построить схему, выполняющую требуемые действия.

Рассмотрим метод, позволяющий **строить логические схемы, выполняющие требуемые действия.**

Пусть необходимо построить автомат, который продает *сникерсы* по 22 копейки. Пусть автомат имеет 3 дырочки, в которые можно засовывать деньги. В дырочку a можно засунуть 20 коп., в дырочку b — 15 коп., в дырочку c — 10 коп. В каждую дырочку можно засунуть *не более одной монеты* (это не совсем логично, но на данный момент это единственное решение). Аппарат анализирует, какие монеты в данный момент засунуты в дырочки, и если сумма больше или равна 22 коп., выдает сникерс. Сдачу или два сникерса автомат не дает.

Соответственно, имеем три входных логических сигнала a, b, c, соответствующих высказываниям "засунута такая-то монетка", и один выходной сигнал s — "дать сникерс". Проанализируем варианты, при которых автомат дает сникерс:

Высказывание	Формула
20 коп. *и* (10 коп. *или* 15 коп.)	$a \cdot (b + c)$
15 коп. *и* 10 коп.	$b \cdot c$

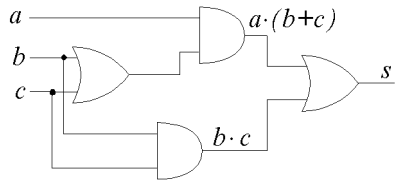
Заметим, что в последнем случае сигнал a в формулу не входит, поскольку все равно, есть ли 20 коп. Кроме того, сникерс дают, когда выполняется первый *или* второй случай.

Легко видеть, что в рассмотренных случаях сникерс дать надо, а в других — нет.

Построим логическую формулу:

$$s = a \cdot (b + c) + b \cdot c$$

и соответствующую ей схему:



Иногда сходу написать формулу не выходит (так это бывает в большинстве случаев). Тогда можно **построить схему по таблице истинности.** При этом таблица истинности строится, исходя из того, что должна делать требуемая схема.

Как же построить формулу по заданной таблице истинности? Рассмотрим, к примеру, таблицу истинности функции от 3 переменных a, b, c:

a	b	c	$f(a,b,c)$	
0	0	0	0	
0	0	1	0	
0	1	0	1	$\overline{a} \cdot b \cdot \overline{c}$
0	1	1	1	$\overline{a} \cdot b \cdot c$
1	0	0	1	$a \cdot \overline{b} \cdot \overline{c}$
1	0	1	0	
1	1	0	0	
1	1	1	0	

Она содержит единиц меньше, чем нулей. Рассмотрим первую единицу сверху. Можно сформировать функцию от 3 переменных такую, что она будет иметь в правой колонке таблицы истинности только одну единицу на нужном месте, а остальные — нули. (Воспользуемся элементом “многовходовое И”. Для того, чтобы на выходе этого элемента появилась 1, все аргументы операции “многовходовое И” должны быть равны 1. Для

комбинации $a = 0$, $b = 1$, $c = 0$ переменные a и c надо инвертировать. Получится $\overline{a} \cdot b \cdot \overline{c}$). Такие функции выписаны для каждой единицы в правой колонке таблицы.

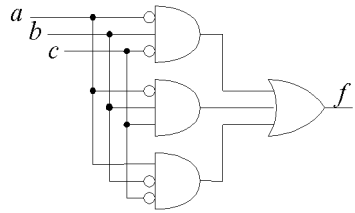
Построим таблицу

a	b	c	$f(a,b,c)$	$\overline{a} \cdot b \cdot \overline{c}$	$\overline{a} \cdot b \cdot c$	$a \cdot \overline{b} \cdot \overline{c}$	$\overline{a} \cdot b \cdot \overline{c} + \overline{a} \cdot b \cdot c + a \cdot \overline{b} \cdot \overline{c}$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	1	1	0	0	1
0	1	1	1	0	1	0	1
1	0	0	1	0	0	1	1
1	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0

В правой колонке стоит логическая формула, соответствующая исходной таблице.

$$f(a,b,c) = \overline{a} \cdot b \cdot \overline{c} + \overline{a} \cdot b \cdot c + a \cdot \overline{b} \cdot \overline{c}$$

Этой формуле соответствует схема

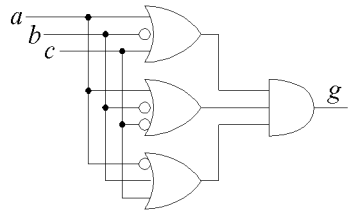


Очевидно, такую процедуру можно проделать с любой таблицей истинности от любого числа переменных. Полученное выражение называется *дизъюнктивной нормальной формой* (ДНФ). Это название происходит от слова "дизъюнкция" — операция "ИЛИ". Недостатком такой формулы является ее громоздкость. Строить ДНФ имеет смысл в том случае, если в таблице истинности мало единиц.

Если в таблице истинности мало нулей, строят *конъюнктивную нормальную форму* (КНФ). Конъюнкция — название операции "И".

a	b	c	$f(a,b,c)$	$a+\overline{b}+c$	$a+\overline{b}+\overline{c}$	$\overline{a}+b+c$	$(a+\overline{b}+c)\cdot$ $\cdot(a+\overline{b}+\overline{c})\cdot$ $\cdot(\overline{a}+b+c)$
0	0	0	1	1	1	1	1
0	0	1	1	1	1	1	1
0	1	0	0	0	1	1	0
0	1	1	0	1	0	1	0
1	0	0	0	1	1	0	0
1	0	1	1	1	1	1	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1

$$g(a,b,c) = (a+\overline{b}+c)\cdot(a+\overline{b}+\overline{c})\cdot(\overline{a}+b+c)\ :$$



Для каждой таблицы истинности можно построить множество формул, ее реализующих, и потому тождественно равных между собой.

ВОПРОСЫ И ЗАДАЧИ

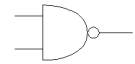
1. Пусть сникерс стоит 40 коп., а дырочки соответствуют монетам 50 коп., 20 коп., 20 коп., 15 коп., 10 коп., 5 коп. Необходимо
а) построить автомат с выходом “дать сникерс” по правилам, рассмотренным выше,
б) построить автомат с выходами “дать сникерс” и “дать второй сникерс” (последний срабатывает, когда сумма больше или равна 80 коп.)

2. Для таблицы истинности

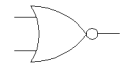
a	b	c	$f(a,b,c)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

а) Построить КНФ
б) Построить ДНФ
в) Придумать формулу, содержащую не более 4 операций
г) Нарисовать схемы для пунктов а), б), в)

3. Будем исходить из элемента "И-НЕ"



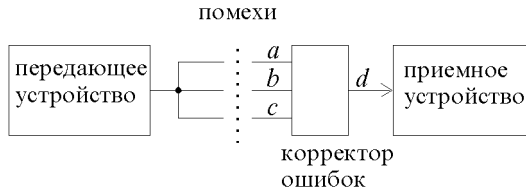
Последовательно построить только из этих элементов элементы "НЕ", "И", "ИЛИ", $\oplus$. То же проделать с элементом "ИЛИ-НЕ"



Элементы "И-НЕ" и "ИЛИ-НЕ", таким образом, могут быть *базовыми элементами логики*.

4. Рассмотрим устройство, относящееся к так называемой *мажоритарной логике*. Пусть имеется линия связи, по которой передаются логические сигналы (0 или 1). Предположим, что в результате помех 0 может превращаться в 1, а 1 в 0, причем вероятность такого изменения крайне мала. Для уменьшения влияния помех (повышения помехоустойчивости), поступим так: будем передавать сигнал

по 3 проводам сразу:



Вероятность того, что ошибка произошла на двух или сразу на трех проводах гораздо меньше, чем вероятность ошибки на одном проводе. Поэтому можно поставить на принимающем конце устройство—корректор, исправляющее однократные ошибки. Это устройство анализирует 3 входных логических сигнала и на выходе выдает такой логический уровень, каких больше на входе (если из трех два или все три равны 1, то надо выдать 1, в противном случае—0).

а) Составить таблицу истинности корректора:

a	b	c	d

б) Составить КНФ или ДНФ по таблице.

в) Нарисовать схему.

Занятие 9

Логические тождества; упрощение логических формул и схем

Два предыдущих занятия были посвящены изложению методов построения схем устройств с заданной логикой функционирования.

В некоторых случаях схему удается построить непосредственно исходя из логики функционирования. В противном случае надо выполнить следующую последовательность шагов:

- Необходимо выяснить, какие входы и выходы должно иметь требуемое устройство.
- Необходимо построить таблицу истинности устройства. Для этого надо перебрать все возможные комбинации входов (проще всего это сделать, перебрав все двоичные числа нужной разрядности) и поставить каждой комбинации входов свою комбинацию выходов.

- Для каждого выхода необходимо построить ДНФ или КНФ, в зависимости от того, чего больше в столбце, соответствующем выходу, единиц или нулей.

- В соответствии с построенной КНФ или ДНФ необходимо построить логическую схему устройства.

Как уже отмечалось, построенная таким образом схема может оказаться слишком громоздкой. Поэтому обычно следует попробовать **упростить логическую формулу**, прежде чем строить схему. Заметим, что для синтеза схем с помощью КНФ или ДНФ существуют (и описаны в предыдущем занятии) четкие *алгоритмы*, следовательно эта процедура должна дать результат в любом случае. Результат упрощения формул, напротив, зависит от сообразительности того, кто этим занимается. Здесь алгоритма нет, есть только описание возможных способов, каждый из которых может дать результат, а может и не дать. Здесь мы разберем несколько примеров успешного упрощения.

Для упрощения логических выражений можно использовать **логические тождества**. Часть из этих тождеств уже рассматривалась в занятии 7. Если какое-либо тождество вам незнакомо (или кажется подозрительным), попытайтесь доказать его с помощью таблиц истинности.

1. **Идемпотентность:**

$$a + a = a, \qquad a \cdot a = a.$$

2. **Коммутативность** (перестановочность):

$$a + b = b + a, \qquad a \cdot b = b \cdot a.$$

3. **Ассоциативность** (сочетательные законы):

$$a + (b + c) = (a + b) + c, \qquad a \cdot (b \cdot c) = (a \cdot b) \cdot c.$$

4. **Дистрибутивность** (распределительные законы):

$$(a \cdot b) + c = (a + c) \cdot (b + c), \qquad (a + b) \cdot c = (a \cdot c) + (b \cdot c).$$

5. **Законы поглощения:**

$$(a \cdot b) + a = a, \qquad (a + b) \cdot a = a.$$

6. "Отрицание отрицания":

$$\overline{\overline{a}} = a.$$

7. Законы де Моргана:

$$\overline{a \cdot b} = \overline{a} + \overline{b}, \qquad \overline{a + b} = \overline{a} \cdot \overline{b}.$$

8. Свойства 0 и 1:

$$0 \cdot a = 0, \qquad 1 + a = 1,$$

$$1 \cdot a = a, \qquad 0 + a = a.$$

9. Другие свойства 0 и 1:

$$\overline{0} = 1, \qquad \overline{1} = 0,$$

$$a \cdot \overline{a} = 0, \qquad a + \overline{a} = 1.$$

Иногда тождества применяются "в обратную сторону". Например, могут оказаться полезными следующие преобразования:

$$b = b \cdot 1 = b \cdot (a + \overline{a}) = a \cdot b + \overline{a} \cdot b.$$

(Укажите, какие тождества использовались при этих преобразованиях).

Разберем несколько примеров:

Пример 1: Упростим функцию

$$f = \overline{a} \cdot b + a \cdot \overline{b} + a \cdot b.$$

применяя логические тождества, получим

$$f = \overline{a} \cdot b + a \cdot \overline{b} + a \cdot b = b \cdot (a + \overline{a}) + a \cdot \overline{b} = b + a \cdot \overline{b} =$$

$$= b \cdot (a + 1) + a \cdot \overline{b} = b + a \cdot b + a \cdot \overline{b} = b + a(b + \overline{b}) = a + b.$$

Итак, с помощью указанных тождеств и некоторой ловкости получено тождество:

$$\overline{a} \cdot b + a \cdot \overline{b} + a \cdot b = a + b.$$

Пример 2: Назовем "оригинальным учеником" школьника, который удовлетворяет хотя бы одному из следующих признаков:

- имеет хорошую успеваемость и примерное поведение, а общественной работой не занимается;

- имеет хорошую успеваемость и плохое поведение;
- ведет себя примерно и занимается общественной работой.

Пусть a — "примерное поведение", b — "хорошая успеваемость", c — "общественная работа", f — "оригинальный ученик". В соответствии с требованиями,

$$f = a \cdot b \cdot \overline{c} + \overline{a} \cdot b + a \cdot c.$$

Применим логические тождества:

$$f = a \cdot b \cdot \overline{c} + \overline{a} \cdot b + a \cdot (b + \overline{b}) \cdot c =$$

$$= a \cdot b \cdot \overline{c} + \overline{a} \cdot b + a \cdot b \cdot c + a \cdot \overline{b} \cdot c =$$

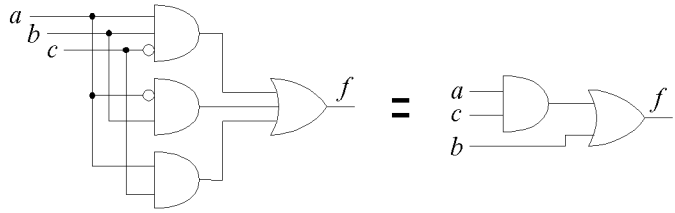
$$= a \cdot b \cdot (c + \overline{c}) + \overline{a} \cdot b + \overline{b} \cdot a \cdot c = b \cdot (a + \overline{a}) + \overline{b} \cdot a \cdot c =$$

$$= b \cdot (1 + a \cdot c) + \overline{b} \cdot a \cdot c = b + a \cdot c \cdot (b + \overline{b}) = b + a \cdot c.$$

Итак,

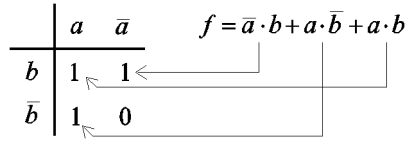
$$f = b + a \cdot c.$$

Логические схемы, соответствующие начальной и упрощенной формулам, показаны на рисунке:

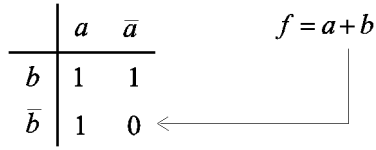


Другим популярным способом упрощения логических выражений являются **карты Карно**. Строятся они так. Пусть известна дизъюнктивная нормальная форма, реализующая данное логическое выражение (ее легко построить, зная таблицу истинности). Предыдущий пример представлял собой уже построенную ДНФ.

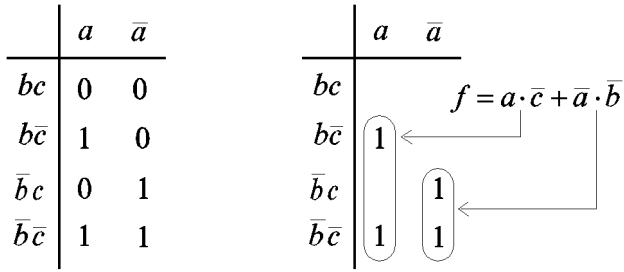
Пусть дана функция двух переменных $f = \overline{a} \cdot b + a \cdot \overline{b} + a \cdot b$. Построим таблицу в соотвествии с ДНФ.



Такая таблица упрощается очевидным образом:

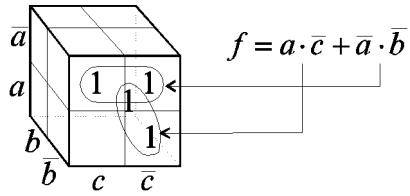


Для ДНФ функции 3 переменных таблица будет выглядеть так:



$$f = a \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot \bar{c} + a \cdot b \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c$$

Последний пример можно (при известном пространственном воображении) представить в виде куба:



В клеточки этой своеобразной пространственной таблицы ставятся нолики и единицы. Очевидно, сокращению с помощью распределительного закона поддаются группы единиц, стоящие в одном столбце, в одной строке или, еще лучше, полностью занимающие целую грань.

Построение ранее “плоская” таблица является ни чем иным, как двумя слоями куба (вдоль оси b), расположенными один под другим.

Наконец, карты Карно функций 4 переменных строятся так:

	$a\cdot b$	$a\cdot\overline{b}$	$\overline{a}\cdot b$	$\overline{a}\cdot\overline{b}$
$c\cdot d$	0	1	0	0
$c\cdot\overline{d}$	0	1	0	0
$\overline{c}\cdot d$	0	0	0	0
$\overline{c}\cdot\overline{d}$	1	1	1	1

$$f = a\cdot\overline{b}\cdot c + \overline{c}\cdot\overline{d}$$

$$f = a\cdot\overline{b}\cdot c\cdot d + a\cdot\overline{b}\cdot c\cdot\overline{d} + a\cdot b\cdot\overline{c}\cdot\overline{d} + \\ + a\cdot\overline{b}\cdot\overline{c}\cdot\overline{d} + \overline{a}\cdot\overline{b}\cdot\overline{c}\cdot\overline{d} + \overline{a}\cdot b\cdot\overline{c}\cdot\overline{d}$$

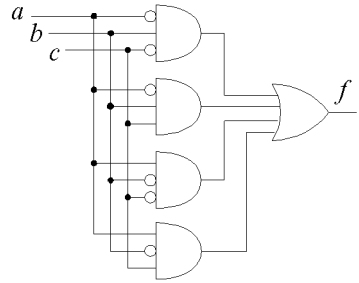
ВОПРОСЫ И ЗАДАЧИ

1. Упростить формулы

а) $f = \overline{a}\cdot b\cdot\overline{c} + \overline{a}\cdot\overline{b}\cdot\overline{c} + a\cdot b\cdot\overline{c} + \overline{a}\cdot\overline{b}\cdot c$

б) $f = a\cdot\overline{b}\cdot c + \overline{a}\cdot b + \overline{b\cdot c}$

2. Построить таблицу истинности и упростить схему, стараясь использовать минимальное число элементов.



3. Построить наиболее простую схему по таблице истинности.

а)

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

б)

a	b	c	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Занятие 10

Двоичное число как совокупность логических сигналов; арифметические и логические операции над числами

В материалах предыдущих занятий логические переменные принимали значения 0 и 1, т.е., как говорят, были битовыми. Во многих случаях в вычислительной технике логические сигналы объединяются группами. Например, шину данных, рассмотренную в третьем занятии, можно считать совокупностью восьми логических сигналов (8 сигналов называют *байтом*).

Пусть есть два 8-разрядных двоичных числа A и B, причем число A состоит из разрядов (логических сигналов) $A_7 \ldots A_0$ (A_0—младший), а число B — из сигналов $B_7 \ldots B_0$. Записывают $A = \overline{A_7 A_6 \ldots A_0}$.

Рассмотрим операцию **логического сложения над числами**. Что означает запись $A + B$ ($A \cdot B$, $\overline{A}$, $A \oplus B$ или какая-либо другая)? Эта запись означает, что указанная логическая операция производится между разрядами чисел A и B, имеющих одинаковый индекс (номер). Так, число $C = A + B$ состоит из 8 разрядов $C_7 \ldots C_0$, причем $C_0 = A_0 + B_0$, $C_1 = A_1 + B_1$, $\ldots$, $C_7 = A_7 + B_7$.

Например, пусть

$$\begin{array}{rl} A & 1001\ 1110 \\ \overline{A} & 0110\ 0001 \end{array}$$

$$\begin{array}{rl} A & 1001\ 1110 \\ B & 0101\ 1000 \\ \hline A \cdot B & 0001\ 1000 \\ A + B & 1101\ 1110 \\ A \oplus B & 1100\ 0110 \end{array}$$

Далее в материале этого занятия и упражнениях к нему встречается только обычное (арифметическое) сложение.

Во многих случаях процессор не может выполнять логические операции с отдельным разрядом числа, а выполняет операции сразу со всеми разрядами.

Более интересным является другой вопрос: каким логическим операциям соответствуют арифметические операции? Ответ на этот вопрос важен поскольку он дает возможность построить из логических элементов устройства, выполняющие сложение, вычитание и т.д.

Прежде всего, отметим, что арифметические операции не соответствуют ни одной из только что рассмотренных логических операций сразу со всеми разрядами. Разряды будут входить в них строго индивидуально.

Для ответа на поставленный вопрос воспользуемся алгоритмами и таблицами, рассмотренными в занятии 3. Рассмотрим **сложение**.

Помимо разрядов слагаемых A_i и B_i будем рассматривать разряды суммы S_i ($S = A + B$) и переносы P (P_i—перенос, возникающий в i-том разряде и прибавляемый к $i+1$-му разряду).

Начнем с нулевых разрядов. Перенос к ним не прибавляется. Очевидно (проверьте), что

$$S_0 = A_0 \oplus B_0, \qquad P_0 = A_0 \cdot B_0$$

Второе равенство следует из того, что перенос из нулевого разряда в первый происходит только тогда, когда младшие разряды обоих слагаемых равны 1.

При формировании S_1 и P_1 необходимо учитывать сигналы A_1, B_1 и P_0. Воспользуемся таблицей сложения из занятия 3. Нетрудно заметить, что эта таблица может быть трансформирована в следующую таблицу истинности:

A_1	B_1	P_0	S_1	P_1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Эта таблица истинности определяет логические функции $S_1(A_1, B_1, P_0)$ и $P_1(A_1, B_1, P_0)$ в явном виде. Можно построить схемы соответствующих устройств с помощью КНФ или ДНФ, однако в следующих разделах будет приведен более практичный вид этих функций.

Если заменить аргументы (A_1, B_1, P_0) на тройку (A_2, B_2, P_1), не меняя функций S_1 и P_1, то получатся формулы для второго разряда суммы и второго переноса. Рассуждения дальше, можно утверждать, что функции, определяемые приведенной выше таблицей истинности, есть функции i-того разряда суммы и i-того переноса $S_i(A_i, B_i, P_{i-1})$ и $P_i(A_i, B_i, P_{i-1})$.

Точно так же, используя результаты занятия 3, можно построить

функции S_i и P_i для вычитания. Здесь A будет уменьшаемым, B—вычитаемым, S—разностью, а P_i — занимание из $i+1$-того разряда в i-тый.

A_i	B_i	P_{i-1}	S_i	P_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Для A_0 и B_0 надо выбрать строки в этой таблице, где p_{i-1} равны 0:

A_0	B_0	S_0	P_0
0	0	0	0
0	0	1	1
0	1	1	0
0	1	0	0

Построенные в этом занятии таблицы могут быть использованы для создания сумматора, который будет складывать и вычитать числа.

Наконец, отметим, что ранее логические переменные a, b, c ...использовались в связи с истинностью тех или иных высказываний (дождь, холодно и т.д.). В этом занятии логические переменные A_i, B_i, C_i имеют несколько иную природу — они являются разрядами чисел A, B, C. Поставим вопрос так: истинность какого высказывания выражает например, переменная—разряд A_0? Легко видеть, что это высказывание таково: "младший разряд числа A равен 1". Действительно, если это высказывание истинно, то $A_0 = 1$, а если ложно, то $A_0 = 0$.

ВОПРОСЫ И ЗАДАЧИ

1. Сумматор выполняет сложение двух 8-разрядных чисел, не обращая внимание на то, имеют ли эти числа знак (записаны в дополнительном коде) или это беззнаковые целые. Рассмотрим такой пример

$$5\mathrm{F\,h} + 4\mathrm{C\,h} = \mathrm{AB\,h}.$$

Если числа не имеют знака, то результат абсолютно правилен. Если же эти числа трактуются как дополнительный код, то налицо

ошибка: слагаемые положительны, а сумма отрицательна. Другой пример такого рода:

$$85\,\mathrm{h} + \mathrm{B3\,h} = 38\,\mathrm{h}.$$

Для того, чтобы информировать пользователя о возникшей ошибке, процессор формирует логический сигнал OV (от слова OVERFLOW — переполнение).
а) Сформируйте сигнал OV из старших разрядов слагаемых A_7, B_7 и суммы S_7, т.е.
$\mathrm{OV} = \mathrm{OV}\,(\mathrm{A}_7, \mathrm{B}_7, \mathrm{S}_7)$.
б) Сформируйте сигнал OV из старших разрядов переносов P_6 и P_7:
$\mathrm{OV} = \mathrm{OV}\,(\mathrm{P}_6, \mathrm{P}_7)$.

2. Сформулируем признак делимости положительного целого числа на 3: Для того, чтобы найти остаток от деления на 3, надо сложить все двоичные цифры, стоящие на нечетных местах, считая от младшего $(a_0 + a_2 + a_3 + \ldots)$. и вычесть из полученной суммы сумму цифр, стоящих на четных местах $a_1 + a_3 + a_5 + \ldots$.

Остаток от деления на 3 исходного числа равен остатку от деления на 3 полученной разности $(a_0 + a_2 + a_3 + \ldots) - (a_1 + a_3 + a_5 + \ldots)$. В частности, число делится на 3 (имеет остаток 0) тогда и только тогда, когда полученная разность имеет остаток 0. (Обратите внимание на следующее: есть определенная путаница: цифры, стоящие на нечетных местах имеют четный номер; признак делимости на 3 в двоичной системе аналогичен признаку делимости на 11 в десятичной).
а) Проверьте признак на нескольких примерах.
б) Докажите (изучив доказательство признака делимости на 11).

3. а) Составить таблицу сложения по модулю 3 (сложение остатков от деления на 3):

+	0	1	2
0			
1			
2			

Для того, чтобы найти сумму двух остатков, надо взять любые два числа, имеющие заданные остатки, сложить их и найти остаток суммы.

б) Запишем остатки от деления на 3 двоичными числами поразрядно

$$A = \overline{A_1 A_0},$$

$$B = \overline{B_1 B_0},$$

$$S = A + B = \overline{C_1 C_0},$$

(сложение производится по правилам, определенным в пункте а). Построить таблицы истинности логических функций $S_0(A_0, B_0, A_1, B_1)$, $S_1(A_0, B_0, A_1, B_1)$.
в) Построить тем или иным способом логические функции S_0 и S_1 и схемы для них.

Остатки от деления на 3 используются в вычислительной технике для контроля правильности выполнения арифметических операций. Дело в том, что остаток от деления суммы (разности, произведения) двух чисел от деления на 3 равен сумме (разности, произведению) остатков аргументов, причем операции с остатками производятся по своим правилам, аналогичным п. а).

Поэтому можно вычислить остатки слагаемых (например, как в задаче 2), сложить сами числа, вычислить остаток суммы и сравнить его с суммой остатков. Если остаток суммы не равен сумме остатков, значит в вычислениях произошла ошибка.

4. Построим умножение в 8-разрядной арифметике.

Пусть имеются числа A и B в дополнительном коде. Процессор (например, i8051) проделывает операцию умножения следующим образом. Он считает, что A и B — запись целых положительных чисел, и умножает их. При умножении двух 8-разрядных чисел может получиться не более, чем 16-разрядное число. Младшие 8 бит процессор помещает в одну ячейку (назовем ее C), а старшие 8 бит в другую (в D). Разумеется, результат получается верным, если A и B — положительные числа. Задача состоит в том, чтобы преобразовать C и D таким образом, чтобы результат получался правильными для A и B, представленных в форме с фиксированной запятой и знаковым разрядом. (Необходимо сделать так, чтобы C и D были младшим и старшим байтами 16-разрядного двоичного числа—произведения, записанного в дополнительном коде).

Разрешается: перемещать биты с места на место (сдвигать), вычитать и складывать, очищать (делать равными 0) и выставлять

(делать равными 1) определенные биты. Необходимо придумать алгоритмы преобразования произведения, если
а) A и B — положительные числа;
б) одно из чисел положительно, другое отрицательно;
в) оба числа отрицательны.

Часть II

Цифровая схемотехника

Глава 4

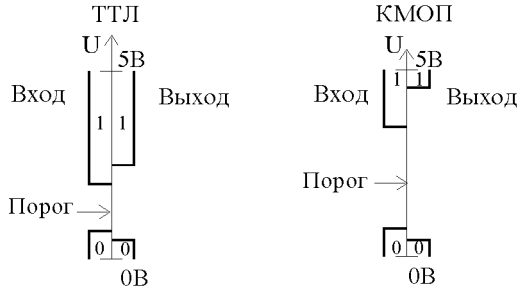
Логические сигналы, зависящие от времени

Занятие 11

Графики логических сигналов; основные параметры логических элементов; временные диаграммы

Прежде всего, определим подробнее логические уровни, рассмотренные в предыдущей главе. Проще всего было бы определить, что логический 0 всегда 0 В, а логическая 1—5 В. Однако, в этом случае логические схемы на смогли бы работать, поскольку всегда есть какие-нибудь шумы, напряжение питания нестабильно, напряжения на выходах логических схем падают при подключении сопротивления нагрузки и т.д. Все это приводит к необходимости использовать в качестве 0 и 1 не какие-то определенные уровни напряжения относительно общего провода, а небольшие диапазоны напряжений. Главное, чтобы эти диапазоны не перекрывались, тогда схема не будет путать 0 и 1. Другим фактором, влияющим на положение логических уровней, является технология производства логических схем. В настоящее время существует насколько семейств логических схем, различающихся структурами полупроводников: `ТТЛ`, `ТТЛШ`, `КМОП`, `ЭСЛ` и др. Эти семейства имеют различные быстродействия, энергопотредление, устойчивость к шумам и др. В частности, они имеют и различные логические уровни. Поэтому микросхемы различных семейств не всегда совместимы между собой.

Наиболее популярными являются `ТТЛ` и `КМОП` серии. На рисунке показаны диапазоны напряжений, соответствующих логическому 0 и 1 в этих сериях:



Отметим, что диапазоны выходных напряжения уже диапазонов входных. В противном случае схемы будут работать нестабильно, т.к. на определенные выходные напряжения устройство—приемник не будет реагировать как на 1.

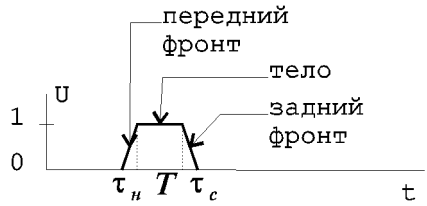
Реакция логической схемы на входные напряжения, не принадлежащие указанным диапазонам, нельзя предсказать достоверно, однако, в большинстве случаев существует так называемый логический порог, т.е. определенное напряжение, выше которого схема распознает вход как 1, а ниже— 0. Вблизи этого уровня логическая схема работает как усилитель.

Если же не вдаваться в подробности, на грамотно нагружаемом выходе `ТТЛ` микросхемы лог. 0 — 0 В, лог. 1 — 3.0–3.5 В, на выходе `КМОП` микросхемы 1 — напряжение питания, 0 — 0 В.

До сих пор рассматривались только логические сигналы, постоянные во времени. Однако ничто не мешает в определенный момент изменить состояния входов некоторой логической схемы и проследить за изменениями на выходе и в промежуточных точках.

Рассмотрим некоторые основные формы логических сигналов:

- **Положительный импульс** (сигнал на небольшое время переходит из 0 в 1):

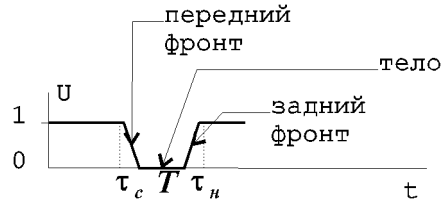


$\tau_н$ — время нарастания
$\tau_с$ — время спада
T — длительность импульса

На рисунке показана форма положительного импульса и основные его части, Следует отметить, что длительность импульса для устойчивой работы схем должна значительно превышать времена нарастания и спада:

$$\tau_н \ll T, \qquad \tau_с \ll T$$

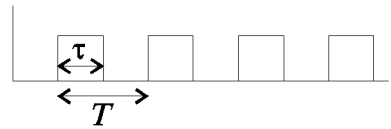
Отрицательный импульс



Смысл величин $\tau_н$, $\tau_с$ и T остается прежним. Иногда про сигналы, для которых 0 — состояние покоя, а положительный импульс вызывает какие-либо изменения, говорят, что это сигналы с активной единицей, а про сигналы, для которых 1 — состояние покоя, а отрицательный импульс вызывает изменения, говорят, что это сигналы с активным нулем. Например, все сигналы, которые управляют работой памяти в IBM-совместимых компьютерах, имеют активными нулевые уровни.

Наконец, получить из положительного импульса отрицательный очень легко. Для этого достаточно пропустить его через элемент "НЕ".

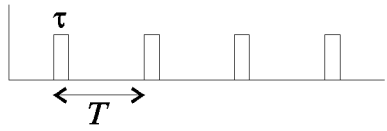
- **Меандр** представляет собой периодическую последовательность логических импульсов. Обычно, хотя и не всегда, длительность участков 0 совпадает с длительностью 1.



τ — длительность импульса
T — период

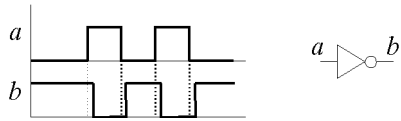
С понятием периода тесно связано понятие частоты. Если сигнал полностью повторяется за T сек, то за одну секунду пройдет $f = 1/T$ импульсов. (Частота — количество импульсов в секунду). Частота измеряется в герцах. Так 10 Гц — 10 импульсов в секунду. Цифровая техника работает на частотах ~100 МГц (тактовая частота вычислительных устройств.

- **Гребенка** Это то же, что и меандр, только $\tau \ll T$.



При этом нельзя забывать, что времена нарастания и спада каждого импульса в гребенке много меньше τ.

Для того, чтобы наглядно фиксировать поведение логических сигналов в различных точках логической схемы на бумаге, применяют временные диаграммы. Эти диаграммы представляют собой несколько совмещенных графиков зависимостей различных логических сигналов от времени, причем ось времени общая для всех сигналов, а ось напряжения для каждого сигнала своя. Например, временная диаграмма работы элемента "НЕ" для определенного входного сигнала выглядит так:



Обратите внимание на то, что элемент реагирует на изменения входного сигнала с некоторой задержкой.

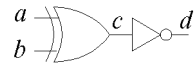
Обычно задержки времени нарастания и спада логических импульсов, вызванные временем срабатывания микросхемы, несущественны. В дальнейшем, кроме тех случаев, когда оговорено противное, временные диаграммы будут строиться упрощенно (идеализированно). При этом будет предполагаться что:

- Логический уровень 1 равен точно 5 В, а 0 — 0 В.
- $\tau_н = \tau_с = 0$, т.е. фронты представляются строго вертикальными линиями.

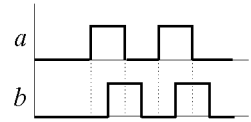
- Схема реагирует на входное воздействие сразу.

Обратимся к **построению временных диаграмм**. Из последнего пункта следует, что ожидать изменения выходного сигнала можно только в моменты изменения входных. Т.е., в простейшем случае временная диаграмма строится так. Вся временная ось разбивается на интервалы, в пределах которых входные сигналы постоянны. На каждом из этих интервалов определяются значения всех промежуточных и выходных сигналов, после чего рисуются фронты.

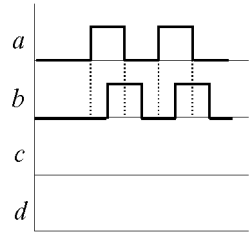
Рассмотрим пример. Пусть необходимо нарисовать диаграмму работы схемы



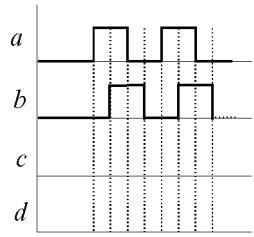
при входных сигналах



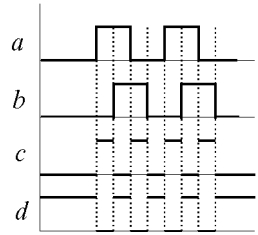
Добавим недостающие точки



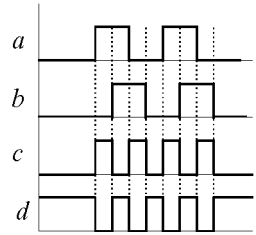
Отметим моменты времени, в которые происходят изменения входных сигналов



Вычислим значение сигналов в точках c и d на интервалах между точками изменения:



Нарисуем фронты:



Идеализированной процедурой построения временной диаграммы нельзя воспользоваться, например, когда техника работает на предельно высоких частотах, т.е. когда период следования импульсов сравним с суммарным временем задержки на схеме. Это время определяется как сумма времен задержки на всех элементах, принадлежащих самой длинной цепочке в схеме, по которой может проходить сигнал.

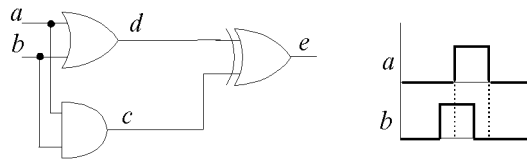
Типичное время задержки на одном элементе составляет для ТТЛШ элементов 20 нс ($20 \cdot 10^{-9}$ c), а для КМОП — 100 нс.

Другие примеры, когда нельзя считать задержку на логическом элементе равной нулю, приведены в упражнениях 2 и 3, а также в практической работе 11.

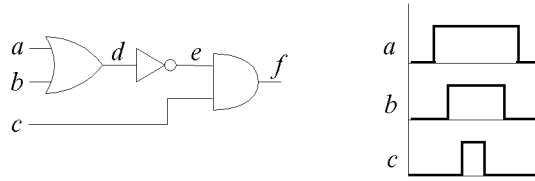
ВОПРОСЫ И ЗАДАЧИ

1. Построить временную диаграмму для схемы и входных сигналов.

а)

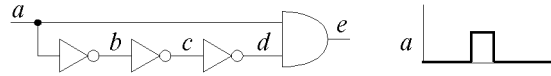


б)

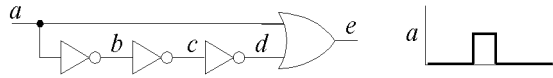


2. Не пренебрегая задержкой на элементах, рассмотрите работу схем (реакцию на входной импульс).

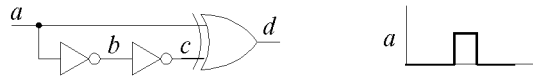
а)



б)



в)

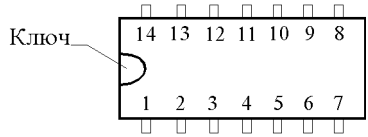


Занятие 12

Интегральные микросхемы: питание, маркировка, выходы

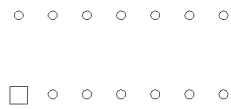
Логические элементы, а также устройства, составленные из этих элементов, выполняются в виде интегральных микросхем. Обычно микросхемы выглядят как пластмассовые прямоугольнички с большим количеством ножек:

Вид сверху:



С одной стороны микросхема имеет выемку или метку краской, называемую *ключом*. Эта метка служит для правильной (однозначной) нумерации ножек микросхемы. Нумерация показана на рисунке.

Первая ножка обычно отмечается на печатной плате, куда впаивается микросхема. На толково (по стандарту) выполненной микросхеме контактная площадка под первую ложку квадратная, а остальные круглые:



Иногда на печатной плате помечают ключ.

Обычно два вывода микросхемы слежат для **подключения питания**, остальные — логические входы и выходы. Во многих случаях питание подключают к уголкам: для микросхемы с 14 выводами 7 вывод — земля (0 В), а 14 вывод — питание +5 В, для микросхемы с 16 выводами 8 вывод — земля, 16 — питание и т.д. Из-за этого можно по ошибке включить микросхему наоборот, т.е. к 7 выводу подать +5 В, а 14 подсоединить к земле. На микросхему будет подано питание обратной полярности и она выйдет из строя. **Причин выхода из строя микросхем** в основном три:

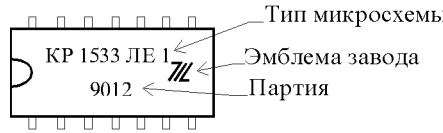
- Механические повреждения: обламывание ножек, перегрев паяльником и т.д.
- Неправильное включение питания.
- Пробой статическим электричеством. (Попробуйте снять с себя синтетический свитер и коснуться пальцами выводов КМОП или n-МОП микросхемы — эффект не заставит себя ждать).

Часто вышедшая из строя микросхема сильно греется.

Также не рекомендуется соединять с шиной земли накоротко выходы логических элементов.

Для того, чтобы определить, с какой микросхемой имеете дело, достаточно взглянуть на маркировку.

Предположим, микросхема выглядит так:



Тип микросхемы расшифровывается следующим образом:

`К`	Для гражданского применения
`Р`	Тип корпуса (пластмассовый)
`1533`	Серия (улучшенная ТТЛ-Шоттки)
`ЛЕ`	Функциональное назначение (ИЛИ-НЕ)
`1`	Номер (разводка)

Наиболее важными параметрами являются серия, функциональное назначение и номер. В справочнике надо искать схему ЛЕ 1, остальные буквы и цифры определяют чисто технические папраметры.

Серия определяет технологию изготовления схемы. От технологии зависит быстродействие, энергопотребление, а также выходные уровни.

К `ТТЛ`-сериям относятся: `131`, `133`, `134`, `155` (сняты с производства), `531`, `555`, `1531`, `1533`. Наиболее популярны `555` и `1533`. Современные `КМОП`-серии: `561`, `564`, `1561`, `1564`. Популярна серия `561`. Сложные микросхемы (процессоры, память и т.д.), специальные (телевизионные, телефонные, аналоговые и т.д.), микросхемы, а также микросхемы, выполненные по другим технологиям (`ЭСЛ`, `AsGa` и т.д.) имеют другие серии.

Часто приходится иметь дело с микросхемами, выпущенными на западе. Существует соответствие между советскими и западными обозначениями. Например, микросхема `К1533ЛЕ1` это то же, что и `74ALS02`. При этом

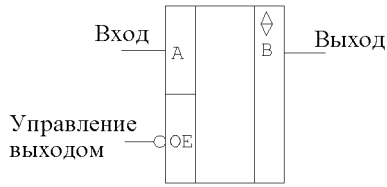
$$\texttt{К1533} = \texttt{74ALS}, \qquad \texttt{ЛЕ1} = 02.$$

Микросхемы всех серий, кроме `134` (`54L`), имеющие один тип и функциональное назначение, обычно имеют одинаковую цоколевку и могут заменять друг друга.

Последнее, о чем необходимо сказать, это о **специфическом характере входов и выходов** некоторых микросхем. Помимо обычных (логических) входов и выходов, а также входов питания бывают:

- **Трехстабильные выходы**. Имеют 3 состояния : 1, 0 и Z. Последнее — т.н. высокоимпедансное состояние. Соответствующий выход просто *электрически отключен* от устройства. Для управления таким выходом применяют специальный вход $\overline{OE}$ (Output Enable — разрешение выхода). Если на входе 1, выходы отключены, если 0 — подсоединены.

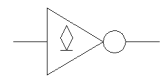
  Трехстабильные выходы позволяют соединять по выходу различные элементы без применения коммутаторов. Обозначаются микросхемы с тремя состояниями например так:



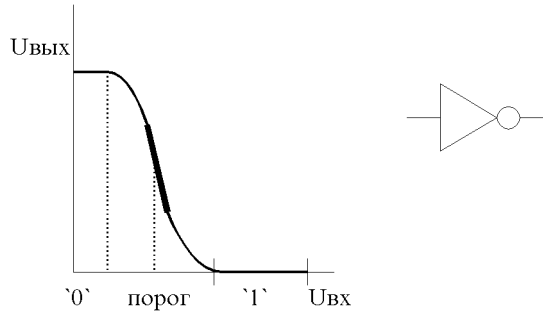
- **Вход** $\overline{CS}$ (Chip Select) — выбор микросхемы. Обычно применяется в сложных микросхемах. Когда $\overline{CS} = 1$, микросхема не выбрана, отключена и потребляет очень мало энергии. Если же $\overline{CS} = 0$, микросхема включается в работу.

  Обратите внимание на то, что активные уровни сигналов $\overline{OE}$ и $\overline{CS}$ низкие. Это связано с тем, что интервал напряжений, соответствующих ТТЛ-нулю значительно меньше, чем ТТЛ-единицы. Таким образом, использование активного уровня 0 и пассивного 1 значительно снижает вероятность ошибочного срабатывания микросхемы в результате случайных выбросов напряжения на шине земли или питания.

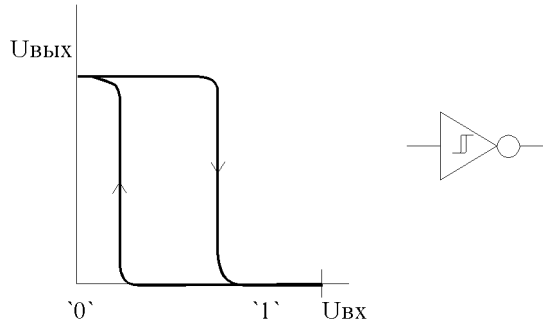
- **Выходы с открытым коллектором** (ОК). На таких выходах логический уровень формируется только в случае подключения к шине питания (возможно, отличающегося от 5 В) через сопротивления нагрузки. Используются для подключения лампочек, светодиодов и других мощных нагрузок. Обозначается ОК так:



- **Выходы с триггером Шмитта**. В некоторых случаях (например, при перегрузке) входное напряжение логического элемента может выйти за пределы областей, соответствующих 0 и 1, и приблизиться к пороговому напряжению. Вблизи порога *обычная* логическая микросхема ведет себя как усилитель:



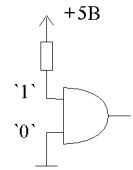
Микросхема с триггером Шмитта ведет себя совершенно иначе. Она обладает *гистерезисом*, т.е. при увеличении входного напряжения ведет себя не так, как при уменьшении. Обозначается Т.Ш. так:



Такое поведение увеличивает помехоустойчивость.Также триггер Шмитта может использоваться совместно с кнопками для формирования резких фронтов сигналов.

- **Аналоговые входы—выходы**. Некоторые КМОП-микросхемы могут использоваться для коммутации аналоговых сигналов.

Последнее, о чем необходимо упомянуть: для подачи на вход микросхемы лог. 0 достаточно подсоединить этот вход в шине земли, для подачи лог. 1 — к шине питания через резистор 5 кОм:



Кроме того, рекомендуется на каждые 2–3 микросхемы ТТЛ и каждую КМОП-микросхему ставить фильтрующий керамический конденсатор емкостью от 0.1 мкф между шиной земли и шиной питания 5 В для подавления помех.

Глава 5

Комбинационная логика

Занятие 13

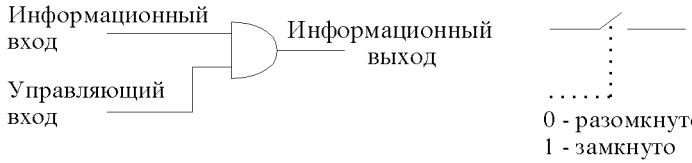
Вентиль; коммутаторы информации; дешифратор и шифратор; демультиплексор и мультиплексор

Все логические устройства делятся на два больших класса: *комбинационные* и *последовательные*. Комбинационные устройства обладают тем свойством, что текущее состояние их выходов однозначно зависит только от текущего состояния их входов. Устройства же последовательной логики, которые будут рассмотрены в следующей главе, имеют возможность определенным образом "запомнить" состояния в какие-то предыдущие моменты времени и использовать запомненную информацию при формировании состояния выходов. В настоящем параграфе рассматриваются комбинационные устройства.

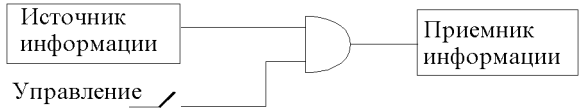
Далее до конца главы будет описано много устройств, их схемы, назначение и временные диаграммы. Важно понимать, что эти устройства являются лишь элементами-"кирпичиками" более сложных устройств, в том числе и тех, которые придется придумывать в качестве решения задач.

На каждое из устройств есть две различные точки зрения. С одной стороны, необходимо представлять себе *схему* устройства. С другой стороны, еще более важно точно знать *что делает* устройство.

Первым и наиболее простым устройством является **вентиль**. Для его реализации возьмем элемент "И" и назовем один из входов "информационным входом", а другой — "входом управления".



Смысл этих обозначений можно понять, исходя из схемы, в которой это устройство может использоваться.



Источником информации может быть любое устройство, общающееся с потребителем по одному проводу. Самой информацией является последовательность нулей и единиц, несущая какой-то смысл. Важно понимать, что ни ноль, ни единица сами по себе не являются информацией.

Рассмотрим работу вентиля. Пусть на управляющем входе 0. Тогда на выходе 0, и выход никак не реагирует на состояние информационных входов. Таким образом, если на управляющем входе лог. 0, то вентиль заперт, а на его выходе 0.

Если на управляющем входе лог. 1, то, исходя из тождества

$$a \cdot 1 = a$$

следует, что выход повторяет состояние входа (что на входе, то и на выходе). Таким образом, вентиль открыт.

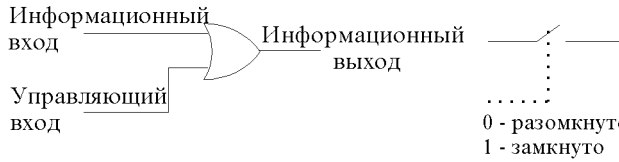
Итак, вентиль открыт, если состояние выхода изменяется при изменении (из 1 в 0 или из 0 в 1) состояния информационного входа.

Вентиль используется для того, чтобы перекрывать поток информации. Вы можете считать, что в состоянии "открыто" информационный вход соединен с выходом, а в состоянии "закрыто" выход подключен к общему проводу.

Другим примером вентиля является элемент "ИЛИ". Если на его входе управления 0, то выход повторяет состояние входа:

$$a + 0 = a,$$

и вентиль открыт. Если на входе управления 1, то и на выходе 1, т.е. он не зависит от состояния информационного входа и вентиль заперт.



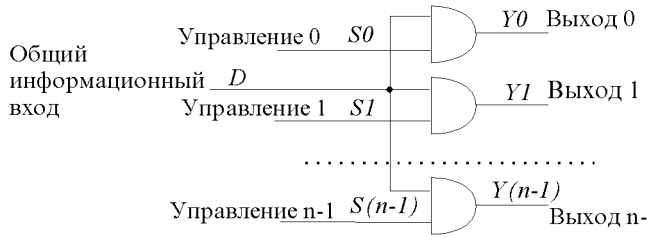
Изложенную информацию можно отразить в таблице

вентиль	управление	выход	состояние
И	0	0	закрыт
	1	вход	открыт
ИЛИ	0	вход	открыт
	1	1	закрыт

Именно за эти свойства элементы “И” и “ИЛИ” по-английски называют “GATES” (ворота).

Важным применением вентиля являются **коммутаторы информации**, предназначенные для подключения разных передающих (приемных) устройств к одному приемному (передающему).

Схема коммутатора “1 передатчик $\to$ n приемников” на основе вентиля “И”:

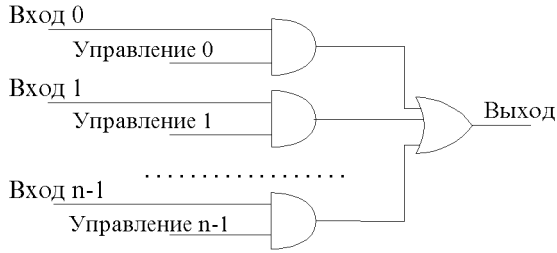


На схеме отмечены информационные и управляющие входы вентилей. (Отметим, что технически разницы между этими входами нет).

Входы S_0, S_1, $\ldots$, S_{n-1} являются управляющими. Если на любом из этих выходов 1 то информация от передатчика поступает к соответствующему приемнику. Если же на управляющем входе 0, то соответствующий приемник “видит” только ноль.

Отметим, что единица может присутствовать сразу на нескольких управляющих входах. При этом информация передается на несколько приемников без искажений.

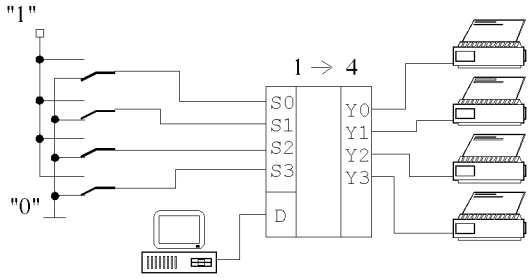
Похожим образом строится коммутатор “n передатчиков $\to$ 1 приемник”.



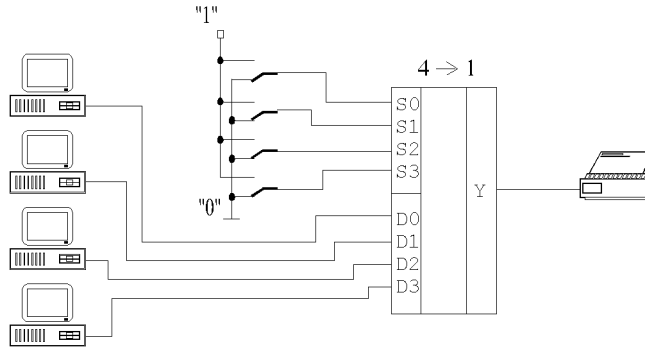
Это устройство направляет один из n информационных потоков к приемнику. Для того, чтобы один из передатчиков соединить с приемником, необходимо подать на соответствующий управляющий вход единицу, а на остальные управляющие входы — нули. Не следует подавать единицу сразу на несколько управляющих входов, поскольку при этом информация будет "смешиваться".

Для того, чтобы понять разницу между информационным и управляющим входами, рассмотрим следующий пример. Пусть имеется компьютер и четыре разных принтера. Пусть все принтеры являются *последовательными*, т.е. информация к ним поступает по одному проводу.

Для того, чтобы иметь возможность печатать на одном из принтеров, необходимо подключить взять коммутатор $1 \to 4$, подключить к информационному входу компьютер, а к выходам — принтеры. Кроме того, необходимо взять четыре тумблера (выключателя), которые могут давать на выходе 0 или 1, и подсоединить их к управляющим входам. Каждый тумблер будет соответствовать одному из принтеров, т.е. для того, чтобы печатать на принтере с номером 1, необходимо включить тумблер с номером 1.

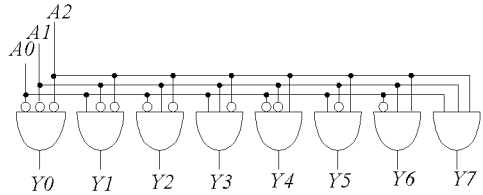


Пусть имеется четыре компьютера и один принтер. Для того, чтобы каждый мог печатать на принтере, а другие ему в это время не мешали, между компьютерами и принтером можно поставить коммутатор $4 \to 1$:



Теперь предположим, что принтеров (в первом случае) или компьютеров (во втором) не 4, а, скажем, 16. Не слишком удобно иметь 16 управляющих сигналов. Присвоим каждому принтеру (или компьютеру) двоичный номер. Очевидно, для номеров с 0 по 15 достаточно всего *четырех* двоичных разрядов. Хорошо было бы иметь устройство, которое преобразовывало бы двоичный номер в сигнал управления, соответствующий устройству с данным номером. Такое устройство существует и называется **дешифратором**.

Изобразим на рисунке схему чуть более простого устройства с 3 входами и 8 выходами (дешифратор $3 \to 8$):

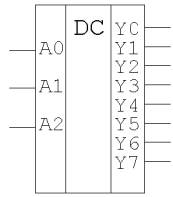


Эта схема работает следующим образом. Трехвходовые “И” срабатывают только тогда, когда на всех входах сигналы (возможно, прошедшие через инверторы) равны 1. Обратите внимание на то, что все комбинации инверторов разные, причем присутствует ровно 8 комбинаций (т.е., все возможные). Таким образом, если сформировать какую-нибудь комбинацию нулей и единиц на входах, “отзовется” ровно одна схема “И”. У этой схемы “И” инверторы стоят именно там, где пришли нули во входной комбинации.

Представим себе, что входные сигналы — разряды двоичного трехзначного числа. В таком случае каждому числу от 0 до 7 соответствует свой выход дешифратора. На этом выходе формируется единица, когда на вход дешифратора подается данное число.

Например, пусть на эту схему пришли сигналы $A_0 = 0$, $A_1 = 1$, $A_2 = 1$. Сформируем из них число, причем A_0 будет младшим разрядом, а A_2— старшим. Тогда $\overline{A_2 A_1 A_0} = 110\,\text{b} = 6$. В результате на 6 выходе (Y_6) будет единица, а на остальных 7 выходах нули.

Дешифратор обозначается следующим образом:



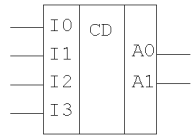
Разумеется, бывают дешифраторы не только $3 \to 8$, но и $2 \to 4$, $4 \to 16$ и т.д. Кроме того, бывают двоично-десятичные дешифраторы (т.е. такие, которые каждому двоичному числу от 0 до 9 ставят в соответствие провод). Они обозначаются $4 \to 10$.

В реальных дешифраторах обычно:

а) Есть один или несколько разрешающих входов. Они обозначаются E (от слова Enable). Для того, чтобы был выбран один из выходов, необходимо помимо задания номера подать на входы E определенную комбинацию. Если вход обозначен как E, то для срабатывания схемы необходимо подать на него 1, а если $\overline{E}$, то 0. (Для реализации этого ставят дополнительные вентили).

б) Невыбранное состояние выходов 1, а выбранное 0. Для того, чтобы достичь этого, необходимо просто инвертитовать все выходы. Это связано с использованием дешифраторов в вычислительной технике для выбора микросхем и будет объяснено позднее. Например, популярный дешифратор `К555ИД7` (`74LS138`), который можно встретить, например, в некоторых старых компьютерных платах расширения, имеет один вход E и два $\overline{E}$ и инвертированные выходы.

Устройством, обратным дешифратору, является **шифратор**. Шифратор $8 \to 3$ имеет 8 входов и 3 выхода:

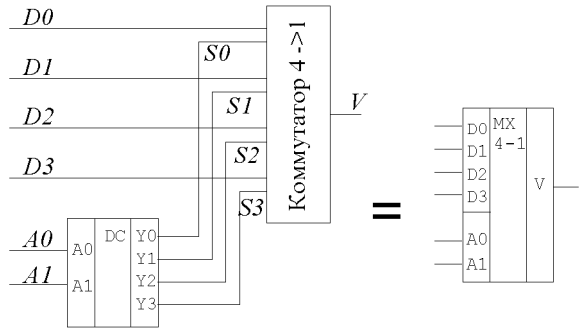


Если на всех проводах нули, а на одном единица, то шифратор выдает на выходах номер этого провода (что происходит, если на всех проводах

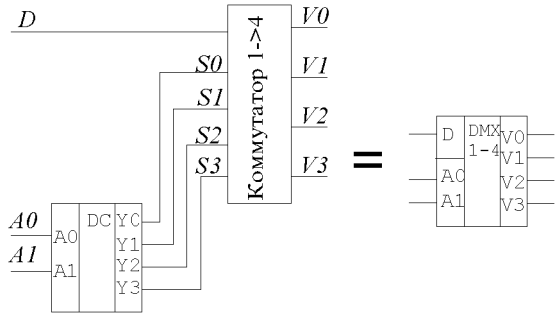
нули, мы не рассматриваем).

Обычно шифраторы бывают приоритетными. Это значит, что при одновременной подаче активного уровня (единицы) на 2 или более проводов устройство выдает больший (или меньший — смотря какой шифратор) номер из выбранных. Устройство приоритетного шифратора с большим числом входов сложно и потому выносится в упражнения.

Если скомбинировать дешифратор и коммутатор, то получится **мультиплексор**:



или **демультиплексор**:



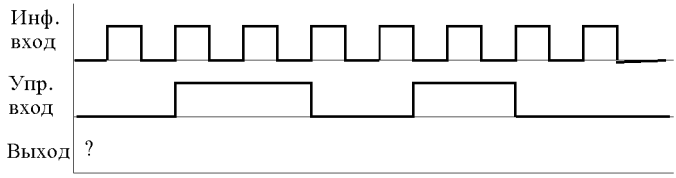
Работу мультиплексора и демультиплексора можно сравнить с работой телефонистки: вы ей сообщаете номер, а она соединяет ваш провод с другим проводом, обладающим заданным номером.

Иногда номер $\overline{A_1 A_0}$ называют *адресом*.

ВОПРОСЫ И ЗАДАЧИ

1. Можно ли использовать как вентиль элемент "Исключающее ИЛИ"?

2. Нарисуйте временную диаграмму

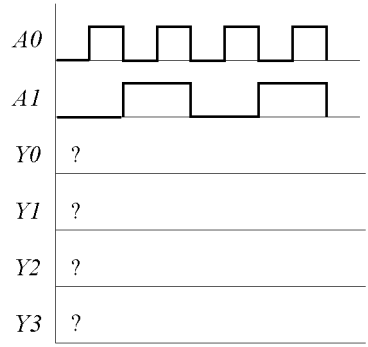


для вентилей а) "И", б) "ИЛИ".

3. Построить коммутатор $4 \to 1$ с помощью вентилей "ИЛИ".

4. Построить дешифратор $1 \to 2$ самым простым способом.

5. Нарисовать временную диаграмму работы дешифратора $2 \to 4$.

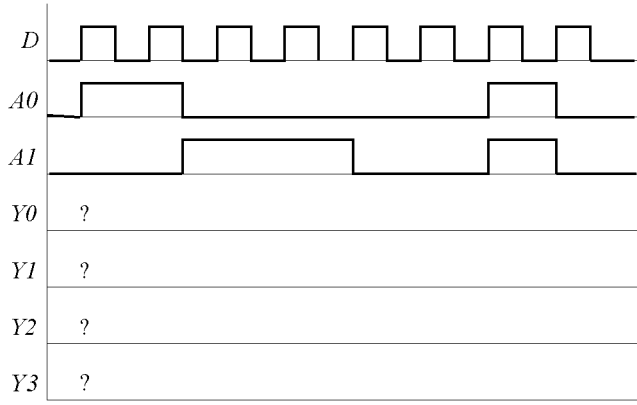


(Рекомендуется в каждом столбце узнать, что за число $\overline{A_1 A_0}$).

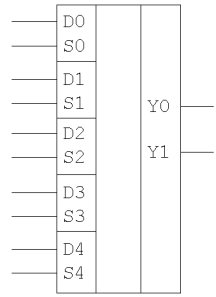
6. Объясните, каким образом можно использовать микросхему `555ИД7` в качестве демультиплексора, ничего к ней не добавляя.

7. Построить приоритетный шифратор $3 \to 2$ при помощи таблицы истинности. ($3 \to 2$ значит, что если не выбран ни один вход, то на выходе 00, если старший выбранный провод имеет номер 1, то на выходе 01, если старший 2, то 10, если старший 3, то 11. Старший выбранный провод — провод с наибольшим номером из тех, на которых присутствует активный уровень (логическая единица).

8. Нарисовать временную диаграмму демультиплексора $1 \to 4$



9. (Сложная задача) Нарисовать схему коммутатора $5 \to 2$



При выборе двух входов из пяти (подаче на 2 управляющих провода лог. единиц) на выходы должны подаваться соответствующие входы, причем на Y_1 — выбранный вход с большим номером.

10. (Сложная задача) Построить схему приоритетного шифратора $7 \to 3$.

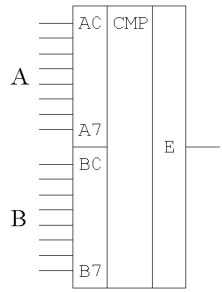
Занятие 14

Устройства арифметики: компаратор, сумматор

Рассмотрим устройства, позволяющие выполнять различные арифметические действия. Прежде всего, построим прибор, определяющий равны ли два числа. Такой прибор называют **компаратором**. Некоторые

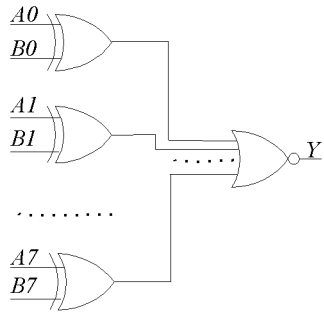
виды компараторов могут определить, какое из чисел больше, однако они сложнее. Их устройство рассматривается в упражнении 5.

Пусть по 16 проводам на вход устройства поступают два 8-разрядных числа: $A = \overline{A_7 \ldots A_0}$ и $B = \overline{B_7 \ldots B_0}$. Они равны тогда и только тогда, когда попарно совпадают разряды: $A_0 = B_0$, $A_1 = B_1$, $\ldots A_7 = B_7$. Выходы устройства является логический сигнал E (от слова Equal), равный 0, если числа не равны и 1, если равны.



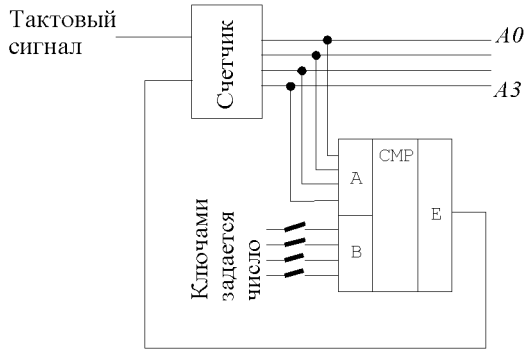
Для построения схемы этого устройства воспользуемся элементом "Исключающее ИЛИ". Заметим, что на выходе у него 1, если на двух входах разные сигналы и 0, если одинаковые. Это значит, что с помощью такого прибора можно попарно сравнивать разряды.

Необходимо проанализировать выходы элементов $\oplus$ и выдать на E единицу только в том случае, если на всех выходах $\oplus$ нули (все разряды чисел попарно одинаковы). С этим справится элемент многовходовое "ИЛИ-НЕ":



Такой компаратор можно с успехом использовать во многих сложных устройствах. Например, с его помощью можно управлять счетчиком (это устройство, подряд перебирающее двоичные числа). Чтобы заставить

счетчик считать до какого-то определенного числа, надо подать выход счетчика на входы A компаратора, а на входы B подать число, на единицу большее максимально допустимого. (Если хочется, чтобы счетчик считал до 6, надо выставить 7. Сделать это можно с помощью переключателей или перемычек). С выхода компаратора снимается сигнал "сброс", возвращающий счетчик в 0, либо "запрет", прекращающий счет:



Сигнал сброса счетчика

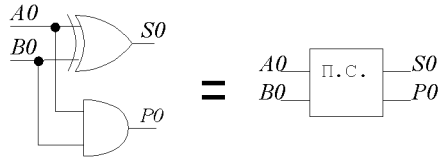
Другим важным применением компаратора служит использование его в вычислительной технике для распознавания адреса. Предположим, что некоторое устройство в компьютере имеет адрес 0056 h. Это значит, что когда на шине адреса появляется это число, устройство должно "понять", что к нему обращаются, и выполнить некоторые действия. Для этого в состав устройства включается компаратор. Входы A компаратора подключаются к шине адреса, а на входы B выставляется адрес устройства. Когда числа совпадут, сигнал E с компаратора даст знать устройству, что к нему обращаются. Обычно сигнал E инвертируется (имеет активный низкий уровень) и называется $\overline{CS}$ (Chip Select — выбор микросхемы).

Другим важным устройствам, реализующим арифметическую функцию, является **сумматор**. Уже исходя из названия, можно догадаться, что устройство предназначено для сложения чисел.

Для понимания устройства сумматора потребуется освежить в памяти материал занятии 10. Там рассматривались логические функции, связанные со сложением. Эти логические функции ложатся в основу схемы.

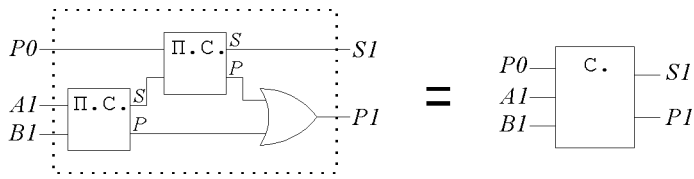
Прежде всего, построим устройство, которое складывает младшие разряды чисел. Его входами будут A_0 и B_0,а выходами — младший разряд суммы S_0 и перенос в следующий разряд P_0. В соответствии с при-

веденной в занятии 10 таблицей истинности, устройство может быть собрано так:



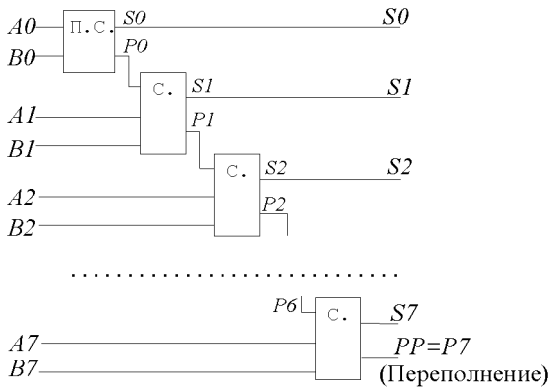
Полученное устройство называют **полусумматором**.

Для вычисления следующего разряда суммы необходимо анализировать состояние трех входов: A_1, B_1 и P_0. Выходами будут S_1 и P_1. Таблица истинности соответствующего устройства известна, поэтому можно воспользоваться КНФ, ДНФ и как-либо упростить полученные формулы. Здесь, однако, мы приведем стандартное решение:



Данное устройство называется **ячейкой сумматора.** Заметим, что входы сумматора, (как и входы полусумматора) совершенно равноправны, а выходы нет.

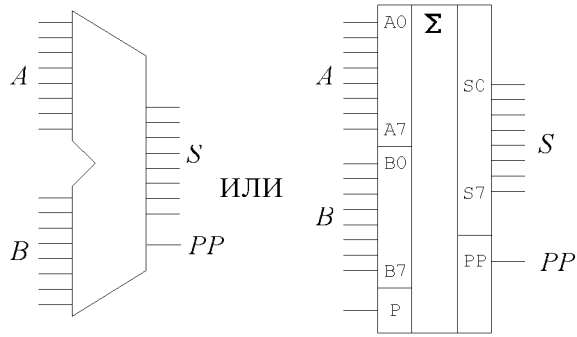
Дальнейшие разряды обрабатываются так же, как только что рассмотренный. В соответствии с этим, сумматор строится из ячеек



Перенос из старшего разряда сигнализирует о переполнении разрядной сетки.

Иногда вместо полусумматора, обслуживающего младшие разряды, ставят ячейку сумматора, и на один из входов этой ячейки подают лог. 0. Такая конструкция позволяет использовать сумматор для построения сумматоров с большей разрядностью. Такой прием называется **каскадированием**.

Как правило, сумматор обозначают так:



Формулу, описывающую действие сумматора можно записать так:

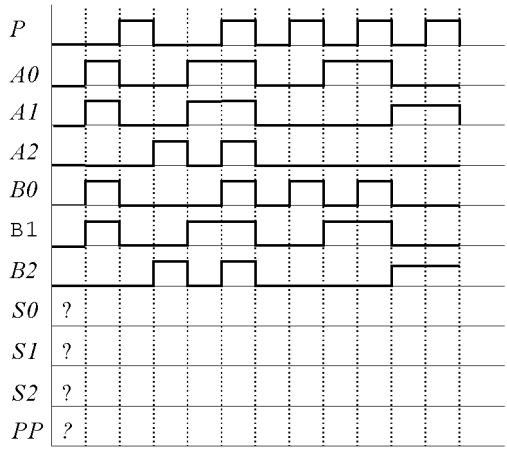
$$P + A + B = S + PP * 2^n,$$

где n — разрядность сумматора.

ВОПРОСЫ И ЗАДАЧИ

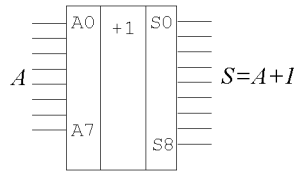
1. Убедиться, что схема ячейки сумматора реализует таблицу истинности, приведенную в занятии 10.

2. Построить временную диаграмму работы 3-разрядного сумматора:

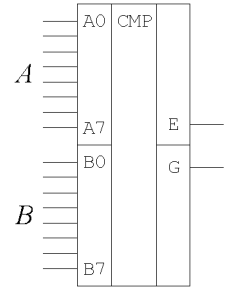


3. По таблице истинности, приведенной в занятии 10, построить ячейку вычитателя и сам вычитатель (устройство, формирующее по числам A и B разность $A-B$ и сигнал занимания из старшего разряда PP).

4. Построить схему устройства, прибавляющего к восьмиразрядному числу A единицу.



5. (Трудная задача) Построить схему компаратора, который умеет проверять, какое число больше:



Если $A = B$, то $E = 1$ и $G = 0$, если $A > B$, то $E = 0$ и $G = 0$, если $A < B$, то $E = 0$ и $G = 0$.

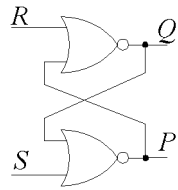
Глава 6

Последовательная логика

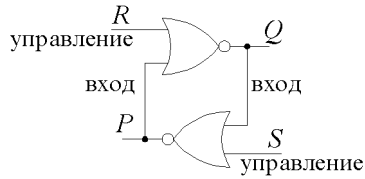
Занятие 15

RS-триггер; синхронизация; D-триггер; JK-триггер; T-триггер

Рассмотрим следующее устройство (оно называется RS-**триггером**):



Можно представить эту схему как кольцевое соединение двух вентилей:

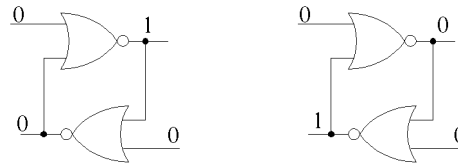


Как известно, вентиль “ИЛИ” открыт, если на его управляющем входе 0 и закрыт в противном случае. Предположим сначала, что хотя бы один из вентилей закрыт, т.е. на R или S присутствует 1.

В таком случае легко определить состояние сначала закрытого вентиля (состояние вентиля — это то, что у него на выходе), а затем и оставшегося открытым:

R	S	Q	P
0	1	1	0
1	0	0	1
1	1	0	0

Если же оба вентиля открыты ($R = S = 0$), то возможны два устойчивых состояния петли из вентилей:

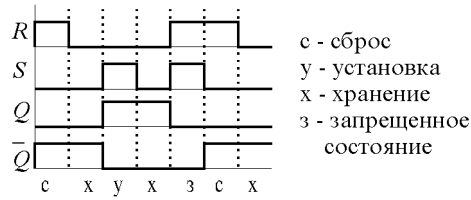


Для того, чтобы сказать, какое из этих состояний реализуется, необходимо помимо текущего состояния входов S и R, знать "историю", т.е. предыдущие состояния системы. Если перед переходом в состояние $S = R = 0$ на входах присутствовало состояние $S = 1$, $R = 0$, а на выходах, соответственно, $Q = 1$, $P = 0$, то это состояние выходов и сохранится после открывания обоих вентилей. Этим устройства последовательной логики (устройства, содержащие разновидности триггеров) отличаются от комбинационных, выходы которых однозначно определяются состоянием выходов в текущий момент времени.

Вернемся к триггеру. Состояние $R = S = 0$ называется **режимом хранения**, поскольку пока на входах нули, сохраняется предыдущее состояние выходов триггера. Триггер называют "установленным", (или "установленным в единицу"), если $Q = 1$, $P = 0$, и "сброшенным" или "установленным в ноль", если $Q = 0$, $P = 1$. Поскольку в обоих устойчивых состояниях $P = \overline{Q}$, этот выход триггера всегда обозначают $\overline{Q}$.

Состояние входов $S = 1$, $R = 0$ называют **установкой** триггера, поскольку после этого Q становится равным 1, а состояние $S = 0$, $R = 1$ называют **сбросом**. Отсюда становятся понятными обозначения: S—Set, R—Reset.

Состояние $S = 1$, $R = 1$ называют *запрещенным*, поскольку если сразу после него выставить $S = 0$, $R = 0$, триггер случайным образом примет одно из двух своих устойчивых состояний, а никакой случайности в работе схемы допускать нельзя. При этом никто не запрещает устанавливать $S = 1$, $R = 1$. Ничего не испортится, надо только следить, чтобы выход из этого состояния осуществлялся через установку или сброс триггера. Приведем пример временной диаграммы работы RS-триггера:

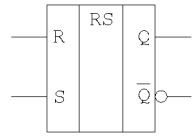


и таблицу истинности

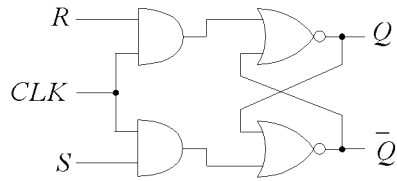
R	S	Q	$\overline{Q}$	
0	0	?	?	хранение
0	1	1	0	установка
1	0	0	1	сброс
1	1	0	0	запрещенное состояние

Признаками триггера являются: а) наличие двух (или более) устойчивых состояний, которые могут наблюдаться при одинаковых состояниях входов и б) возможность переводить триггер в каждое из этих состояний при помощи манипулирования входами. Подробнее с этим можно познакомиться в упражнении 1.

Обозначается триггер так:



Предположим, мы хотим, чтобы триггер реагировал на изменения входов только в определенные моменты времени. Для этого пропустим входные сигналы через два вентиля:

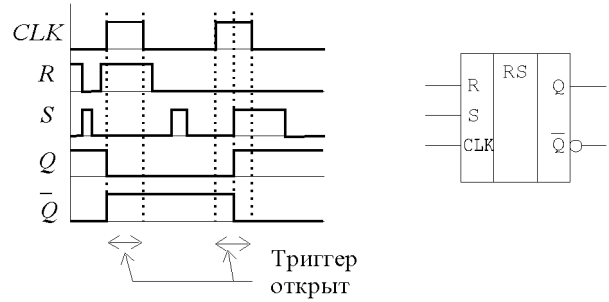


Смысл этой операции очень простой. Теперь триггер будет находиться в режиме хранения, пока $CLK = 0$ (триггер закрыт), реагировать на R и S, когда $CLK = 1$.

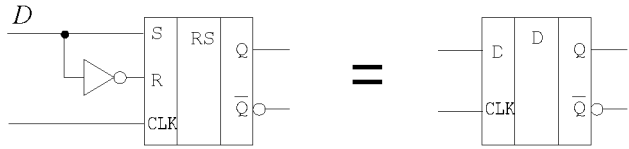
Говорят, что это триггер, **тактируемый** высоким уровнем или синхронный. Введение тактового сигнала называют *синхронизацией* или *тактированием*.

Тактирование имеет смысл, когда вместе работает несколько устройств, содержащих триггеры. Использование для всех устройств одного и того же сигнала CLK позволяет добиться синхронности работы всех устройств и избежать характерных ошибок, вызванных задержкой при прохождении через логические элементы (см. упражнение 2 и практическую работу 11).

Пример временной диаграммы и обозначение синхронного RS-триггера:



Установкой и сбросом тактируемого триггера можно управлять при помощи одного сигнала D:



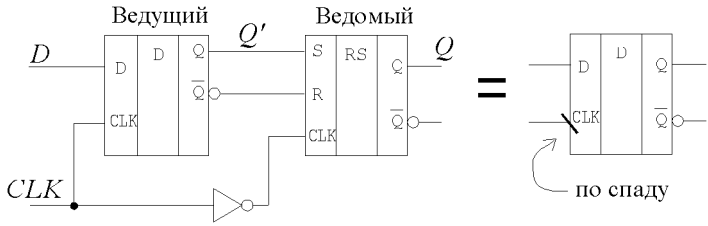
Это устройство называется D-**триггером**. Теперь при $CLK = 1$ триггер будет устанавливаться ($D = 1$) или сбрасываться ($D = 0$) и хранить информацию при $CLK = 0$ (хранить информацию = сохранять состояние).

D-триггер "помнит" на выходе Q состояние входа D при $CLK = 0$ (т.е. он "считывает" состояние входа D при $CLK = 1$ и запоминает его). Использование D-триггеров для хранения чисел и некоторых других действий будет рассмотрено в следующем занятии.

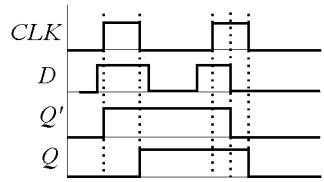
Рассмотренный триггер имеет один важный недостаток. Его входы открыты в течение всего времени, пока $CLK = 1$. В течение этого времени состояние входа D (и, соответственно, выхода Q) может измениться

несколько раз. Это недопустимо в некоторых важных случаях, в частности, когда ко входу триггера подсоединяется его же выход, как в JK- или T-триггере. Поэтому очень часто применяются триггеры, **тактируемые фронтом**.

Рассмотрим один из таких триггеров, его еще называют "триггер типа ведущий-ведомый":



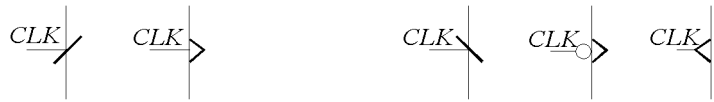
Эта схема состоит из 2 синхронных триггеров. Рассмотрим временную диаграмму. Пусть в начальный момент триггер был сброшен:



Очевидно, состояние выхода изменяется только в момент спада сигнала CLK (перехода из 1 в 0).

Такой триггер называют непрозрачным, поскольку во время срабатывания состояния его выходов не могут измениться более одного раза.

Заметим, что работа такого триггера полностью аналогична работе обычного D-триггера, только состояние входа он читает в строго определенный момент — по спаду CLK. Тот факт, что устройство срабатывает по положительному (отрицательному) фронту обозначают так:



по фронту по спаду

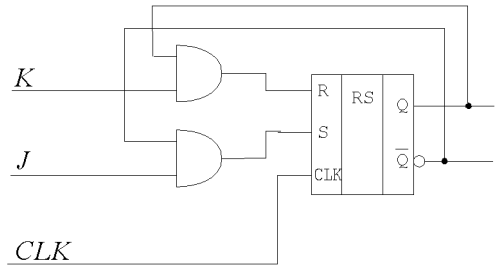
Сам фронт (спад) обозначают так:

фронт спад

Обозначение типа

не имеет смысла.

Заменив ведущий D-триггер на RS-триггер, построим RS-триггер, срабатывающий по фронту. Он используется для построения ***JK*-триггера**.



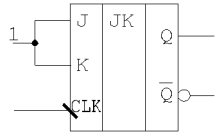
Если на входы J и K приходят пары значений (0,1) или (1,0), то триггер работает как RS-триггер (сбрасывается или устанавливается по фронту CLK). Состояние $J = 0$, $K = 0$ — режим хранения. Единственное отличие наблюдается при $J = 1$, $K = 1$ — триггер меняет свое состояние по фронту CLK. Т.е., если до прохождения фронта CLK (при $J = K = 1$) было $Q = 0$, $\overline{Q} = 1$, то после будет $Q = 1$, $\overline{Q} = 0$ и наоборот. Это может быть отражено в таблице

J	K	Q	$\overline{Q}$	
0	0	?	?	хранение
0	1	0	1	установка
1	0	1	0	сброс
1	1	?	?	переключение

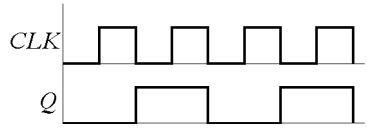
Напомним, что все описанные в таблице изменения происходят только при прохождении фронта сигнала CLK.

Обычно в промышленно-изготавливаемых триггерах есть различные входы (J-K, R-S, D), причем часть из них тактируемые, а часть нет (могут работать без CLK). Нетактируемые входы называются *асинхронными*. Схема одного из таких триггеров приведена в упражнении 4.

Из JK-триггера нетрудно построить **счетный триггер**:

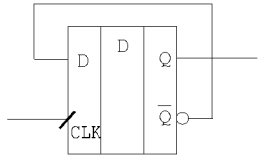


По спаду каждого из импульсов CLK состояние Q будет меняться на противоположное:



Аналогичный прибор можно сделать из триггера, срабатывающего по фронту (тогда он будет переключаться по фронту CLK).

Кроме того, такое же устройство может быть построено из D-триггера, срабатывающего по фронту или спаду:

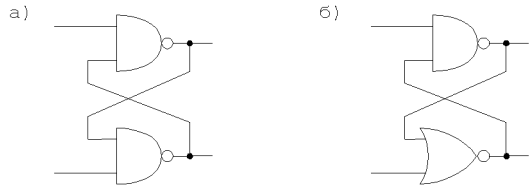


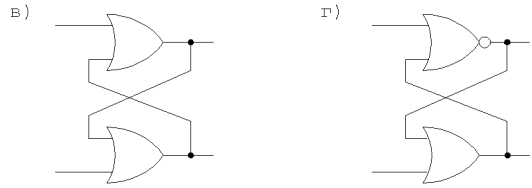
Отметим, что из прозрачного триггера (управляемого уровнем) такое устройство сделать нельзя (объясните почему).

Название “счетный триггер” объясняется использованием этих устройств в счетчиках.

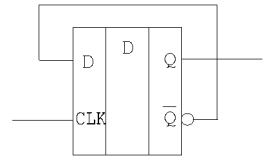
ВОПРОСЫ И ЗАДАЧИ

1. Выясните, являются ли триггерами устройства :

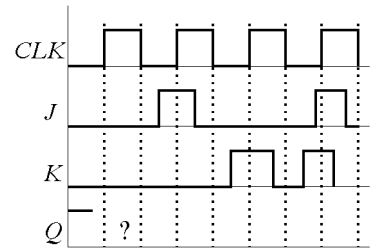




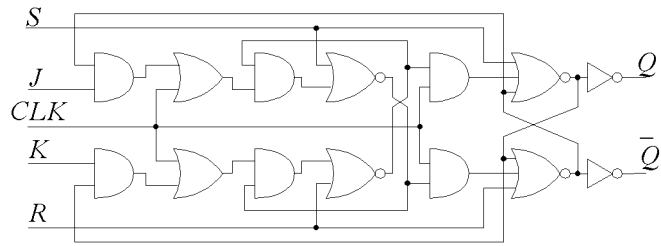
2. Что произойдет, если в устройстве использовать триггер, управляемый уровнем



3. Построить временную диаграмму работы JK-триггера, тактируемого по фронту:

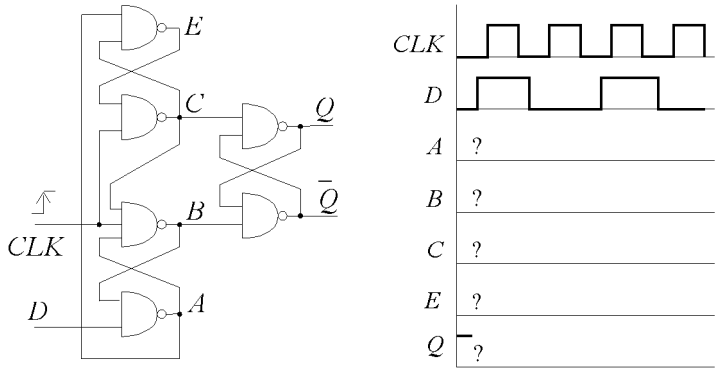


4. Рассмотреть схему триггера из практикума:



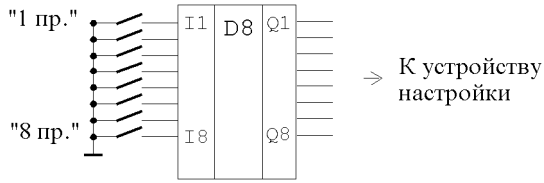
а) определить, какие из входов синхронные, а какие — асинхронные;
б) выделить основные узлы устройства (пары вентилей, триггеры "ведущий" и "ведомый" и т.д.;
в) определить, по фронту или по спаду CLK срабатывает триггер.

5. Рассмотреть работу триггера, срабатывающего по фронту. Построить временную диаграмму.



6. Придумать схему триггера, принимающего информацию по фронту импульса CLK, а выдающего по спаду.

7. (Трудная задача) Иногда бывает необходимо иметь триггер, обладающий более, чем 2 устойчивыми состояниями. Например, рассмотрим устройство выбора программы телевизора типа "Рубин". Оно имеет 8 кнопок (не фиксируемых механически) и содержит многостабильный триггер:



Работает многостабильный триггер так. В состоянии покоя (хранения) на всех входах I_i присутствует лог. 1. При этом на всех, кроме одного из выходов Q_i лог. 0, а на одном—1. Когда возникает необходимость настроиться на какую-то программу (например,

вторую), на короткое время замыкается соответствующая кнопка. При этом на входе I_2 появляется лог.0. После этого триггер должен выставить на Q_2 единицу, а на остальные Q_i — нули и остаться в этом состоянии до следующего нажатия кнопки. Спроектируйте такое устройство для

а) 3 входов и 3 выходов

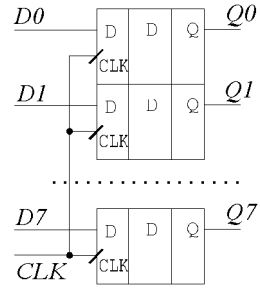
б) 8 входов и 8 выходов.

Занятие 16

Применение триггеров: регистр-защелка, сдвиговый регистр, счетчик

Триггеры, рассмотренные на предыдущем занятии, применяются в большом количестве устройств последовательной логики. Во всех приложениях они используются как элемент памяти.

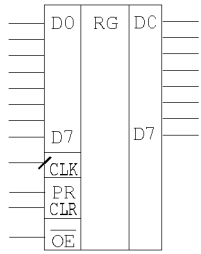
Прежде всего, рассмотрим самое простое применение триггеров в качестве памяти для одного 8-битного числа:



Сгруппировав 8 D-триггеров и объединив их входы синхронизации CLK, получим **регистр-защелку** (или параллельный регистр). Работа этого устройства очень проста. Сформируем на входах $D_0 \ldots D_7$ логические уровни, соответствующие разрядам некоторого двоичного числа. После этого подадим импульс CLK. После прохождения фронта импульса на выходах $Q_0 \ldots Q_7$ появятся разряды запомненного числа. Они будут сохранены на выходах Q_i до тех пор, пока на придет новый импульс CLK и т.д.

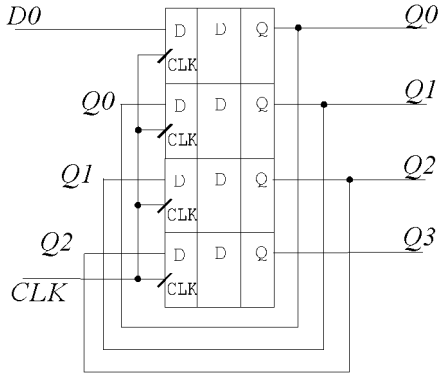
Помимо входов $D_0 \ldots D_7$ и CLK, регистр-защелка может иметь несколько входов, например, вход "общий сброс" (CLR), при подаче активного уровня на который происходит обнуление всех выходов Q_i (запись в триггеры нулей), вход "общая установка" (PR), осуществляющий запись в триггеры единиц, а также вход "разрешение выхода" ($\overline{OE}$). Входы CLR

и PR могут быть как синхронными, так и асинхронными. В принципиальных схемах регистр обозначается так:



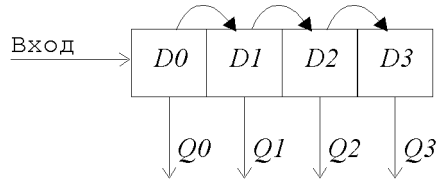
Разумеется, регистры могут иметь не только 8, но и любое другое количество разрядов.

Несколько более сложно описать работу **сдвигового регистра**. Простейший сдвиговый регистр можно получить из регистра-защелки, состоящего из непрозрачных D-триггеров:



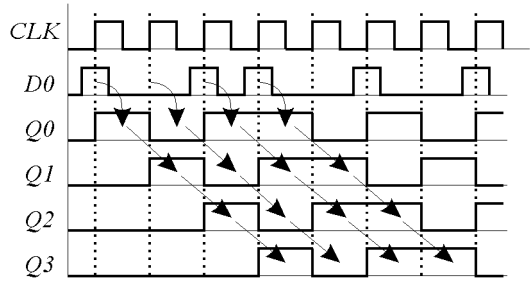
Очевидно, что при прохождении импульса CLK происходит следующее:
а) на триггер D_0 поступает информация со входа;
б) на триггер D_1 (D_2, D_3) поступает информация, хранившаяся до импульса CLK на триггере D_0 (D_1 ,D_2).

Можно представить себе сдвиговый регистр как ряд ячеек, в которых могут храниться нолики и единички:



При подаче импульса CLK, обитатель каждой ячейки переходит в соседнюю справа. Можно представлять сдвиговый регистр в виде очереди.

Временная диаграмма работы сдвигового регистра такова:



Рассмотренный нами регистр был типа "последовательный вход — параллельный выход", т.е. информация в него вводилась через вход D_0 бит за битом, а сниматься могла сразу со всех выходов Q_i. Встречаются регистры самых разных типов, например, "параллельный вход — параллельный выход", т.е. есть возможность в какой-то момент присвоить всем триггерам определенные (независимые друг от друга) значения. Кроме того, возможно наличие входов "сброс" и "предустановка", о которых говорилось ранее. Наконец, некоторые регистры предоставляют возможность выбирать направление сдвига (они имеют специальный вход "направление". Если на этом входе 0, то сдвиг происходит в одном направлении, а если 1, то в обратном).

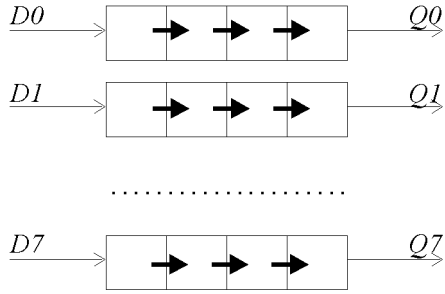
Сдвиговый регистр обозначается так же, как и защелка, только над символом RG ставится стрелочка: $\overrightarrow{RG}$, если регистр допускает сдвиг только в одну сторону и $\overleftrightarrow{RG}$, если можно выбирать направление.

Укажем несколько возможных применений сдвигового регистра:

а) Как уже отмечалось, сдвиг двоичного числа на 1 разряд влево представляет собой его умножение на 2, а вправо — деление на 2. Таким образом, сдвиговый регистр может использоваться в схемах умножителей. Очевидно, для этого подойдет регистр "параллельный вход — параллельный выход".

б) Важное применение находит сдвиговый регистр в устройствах обмена для перевода информации из последовательного вида в параллельный и наоборот. (Последовательный — это когда число передается по одному проводу бит за битом, параллельный — когда все разряды одновременно по разным проводам.)

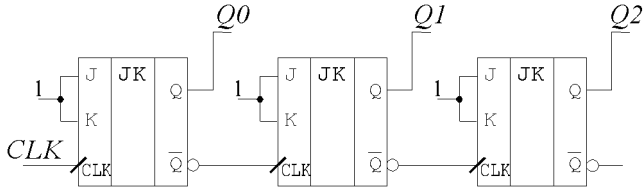
в) Сдвиговый регистр может использоваться в линиях задержки, т.е. устройствах, осуществляющих задержку информации на несколько тактовых сигналов. Такие устройства часто бывают необходимы при работе со звуковыми и локационными сигналами. Регистр, состоящий из 4 триггеров, осуществляет задержку на 4 тактовых импульса. Если необходимо задержать число, состоящее из нескольких разрядов, ставят несколько сдвиговых регистров (по одному на каждый разряд):



Сдвиговые регистры

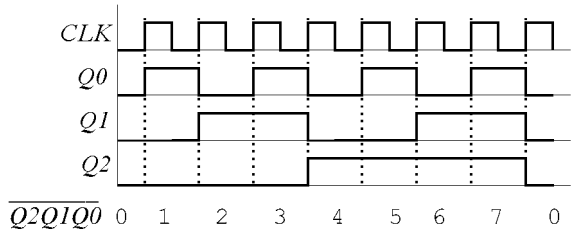
Такое устройство называют последовательно-параллельной памятью. Наконец, очень полезным и поучительным применением сдвигового регистра является генератор псевдослучайной последовательности.

Применением счетных JK-триггеров служат **счетчики**. Они бывают синхронные и асинхронные. Схема асинхронного счетчика такова:



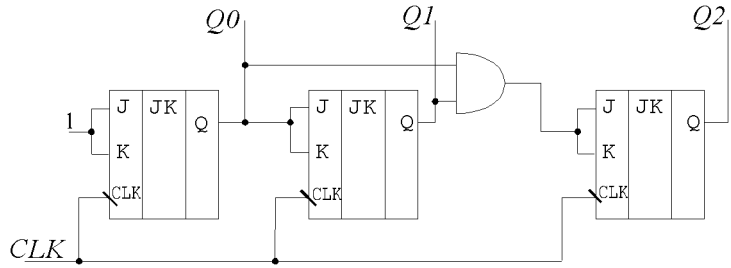
Входом этого устройства является вход CLK первого триггера, а выходами — Q_0, Q_1, Q_2 ...

Рассмотрим временную диаграмму работы счетчика. Пусть в начальный момент все триггеры установлены в ноль:



Заметим, что фронтом сигнала *CLK* каждого последующего триггера является спад сигнала Q предыдущего триггера. Под временной диаграммой выписано значение числа, состоящего из разрядов Q_2, Q_1, Q_0 в каждый момент времени. Очевидно, что счетчик считает фронты *CLK* (от 0 до 7).

Синхронный счетчик, считающий от 0 до 7, устроен следующим образом:

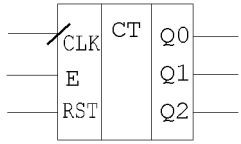


Рекомендуем рассмотреть работу этой схемы самостоятельно. Временная диаграмма его работы почти не отличается от диаграммы работы асинхронного счетчика. Разница заключается в том, что переключение всех триггеров в синхронном счетчике происходит одновременно. а в асинхронном, например, между фазами 7 и 0 бежит "волна" переключений.

В тот короткий промежуток времени, когда первый триггер уже переключился, а последний еще нет, на счетчике возможны состояния 6 и 4. Если к выходам триггера подключено какое-либо комбинационное устройство, например, компаратор, срабатывающий при появлении 4, то такое поведение может привести к ложным срабатываниям.

Счетчики могут считать как в прямом (инкрементные), так и в обратном (декрементные) направлении, или предоставлять выбор направления счета. Кроме того, некоторые счетчики имеют входы *RST* (Reset) — сброс в 0 и предоставляют возможность предварительной записи числа,

с которого начинается счет. Наконец, может присутствовать вход E (Enable), разрешающий счет. Обозначаются на схеме счетчики так:



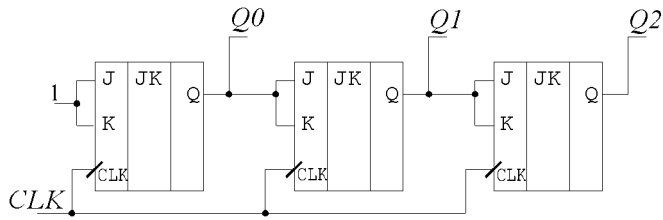
Обратим внимание на то, что частота импульсов на выходе Q_1 вдвое меньше частоты CLK, на выходе Q_2 — вчетверо меньше и т.д. Поэтому счетчики иногда называют делителями частоты на 2^n. Наконец, несложное дополнение заставит счетчик считать не до 8, 16, 32 и т.д., а до произвольного числа. Для этого необходимо снабдить счетчик компаратором, сравнивающим выход счетчика с наперед заданным числом (на единицу большим максимального в счете), а выход компаратора соединить со входом RST счетчика. Соответствующая схема приведена в предыдущем занятии. Надо только следить за тем, чтобы не было ложных срабатываний, например, поставив вентиль перед входом RST и открывая его сигналом CLK, задержанным на нескольких медленных КМОП-инверторах.

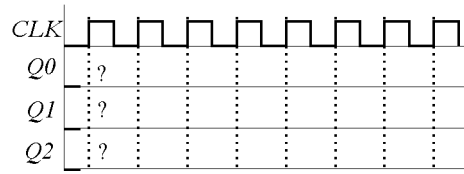
Применения счетчиков широки и разнообразны. Например, частотомер представляет собой счетчик, считающий количество импульсов за определенный интервал времени и выдающий их число на экран; электронные часы есть ни что иное, как счетчик, считабщий тактовые импульсы, идущие со строго заданной частотой и т.д.

ВОПРОСЫ И ЗАДАЧИ

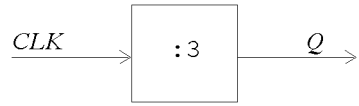
1. Придумайте схему декрементного 3-разрядного счетчика, т.е. считающего так: 0, 7, 6, 5, 4, 3, 2, 1, 0 ...

2. Нарисуйте временную диаграмму работы такого прибора:

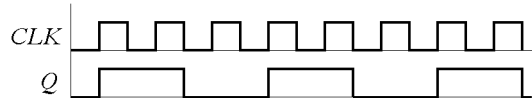




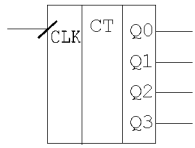
3. (Сложная задача) Придумайте схему делителя частоты на 3:

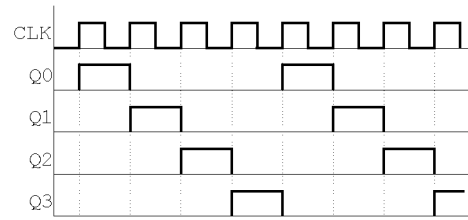


с такой временной диаграммой



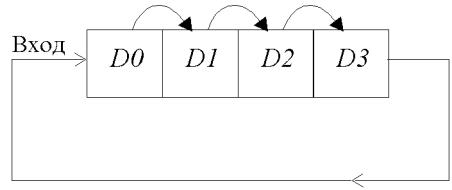
4. Придумать прибор, последовательно перебирающий 4 провода:





для его построения использовать
а) счетчик;
б) сдвиговый регистр.

5. Выход Q_3 сдвигового регистра соединен со входом:



Если в начальный момент $\overline{D_3 D_2 D_1 D_0} = 0001\,\mathrm{b}$, в последующие это число превращается последовательно в: 10 b, 100 b, 1000 b, 1 b, 10 b...

Перечислить все возможные цепочки 16-ричных чисел, получающихся друг из друга в сдвиговом регистре.

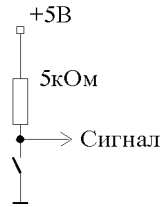
Глава 7

Устройства ввода-вывода

Занятие 17

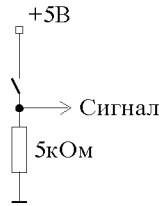
Кнопки; клавиатура; светодиод; 7-сегментный индикатор; линейка индикаторов

Ранее рассматривалось и использовалось такое устройство ввода, как **кнопка**. На самом деле это устройство имеет такую схему:



Если контакт разомкнут, то на выходе присутствует лог. 1, а если замкнут, то лог. 0. Если необходима кнопка, которая дает 1 при нажатии, следует поставить на выходе инвертор.

Отметим, что похожая, но несколько иная схема

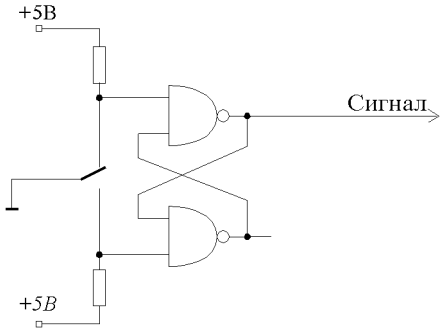


НЕПРАВИЛЬНО

работать в схемах ТТЛ-логики не будет. Это связано с тем, что для этого типа логических элементов характерен значительный входной ток (0.25 мА) при лог. 0 на входе. Такой ток (если его пропустить через резистор) может создать на входе напряжение, превышающее порговое, и схема воспримет состояние входа как лог. 1.

Другим неприятным свойством кнопки является **дребезг контактов**. В течение 1 мс после замыкания ключа его контакты входят в соприкосновение друг с другом от 10 до 100 раз. Это может привести к появлению на входе логической схемы последовательности импульсов:

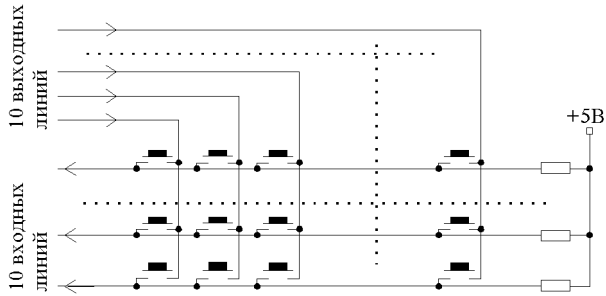
Для подавления дребезга используется RS-триггер:



(При переключении триггер устанавливается в 0 или 1 в момент первого касания контактов).

Альтернативой является программное подавление дребезга в микропроцессорных устройствах, т.е. состояние кнопки опрашивается, например, 100 раз за 1 мс ее состояние считается известным, только если результат всех 100 опросов оказался одинаковым.

Для опроса большого количества кнопок **клавиатуры** используют следующий прием. Пусть необходимо узнать состояние 100 кнопок (на клавиатуре компьютера 101 кнопка или около того). Совершенно не обязательно заводить на вход обрабатывающей микросхемы все 100 проводов. Построим схему так:



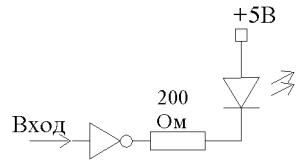
Работает эта схема следующим образом. Очень быстро (например, 100 раз в секунду. Для машины это медленно, а человек с такой скоростью нажимать на кнопки не может) устройство перебирает все выходные линии. При этом на выбранную линию выставляется 0, а на остальные 1. Это соответствует последовательному опросу различных столбцов кнопок. Если при опросе, например, третьего столбца на второй входной линии появляется 0, то это означает, что нажата кнопка, стоящая на пересечении третьего столцба и второй строки.

Таким образом, вместо 100 используется только 20 проводов. Процедура опроса достаточно сложна, поэтому разумно применять такой способ в микропроцессорных устройствах.

Подобный способ ввода называют **мультиплексированием** данных.

Обратимся к устройствам вывода информации. При этом не будем касаться таких сложных вопросов, как вывод на видеомонитор.

Самым простым и распространным устойством индикации является **светодиод**. Он характеризуется (в отличие от лампочки) малым потреблением тока (10 мА) и малыми габаритами. Обычно светодиод включается следующим образом:

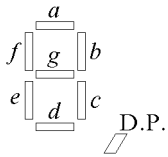


Если на входе 1, на выходе инвертора 0, и создается разность потенциалов на диоде. Резистор ограничивает ток через диод.

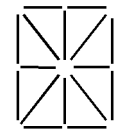
Для отображения символов светодиоды объединяют в группы, образующие **диодные матрицы**:

или как-то иначе.

Самыми поулярными являются 7- и 16-сегментные индикаторы:

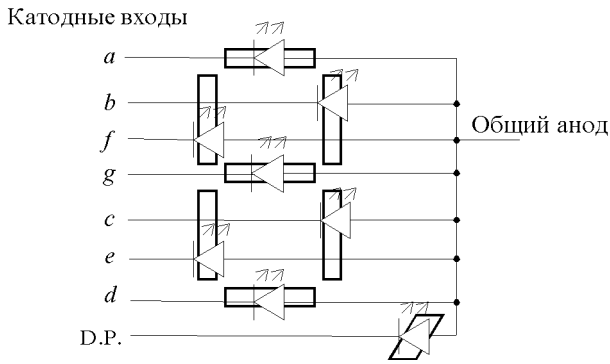


7-сегментный



16-сегментный

7-сегментный индикатор представляет собой 7 светящихся полосок-светодиодов, которые обозначаются $a \ldots g$ и служат для отображения символов, атакже светодиод *d.p.* для отображения десятичной точки (Decimal Point). Обычно устройство этого идикатора таково:

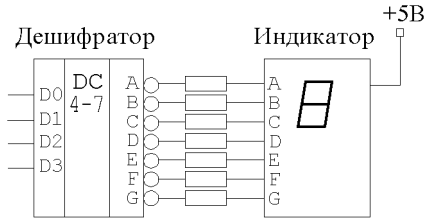


Бывают также индикаторы с общим катодом.

С помощью 7-сегментного индикатора можно отображать цифры $0 \ldots 9$, а также буквы $A \ldots F$ (что полезно для отображения 16-ричных чисел).

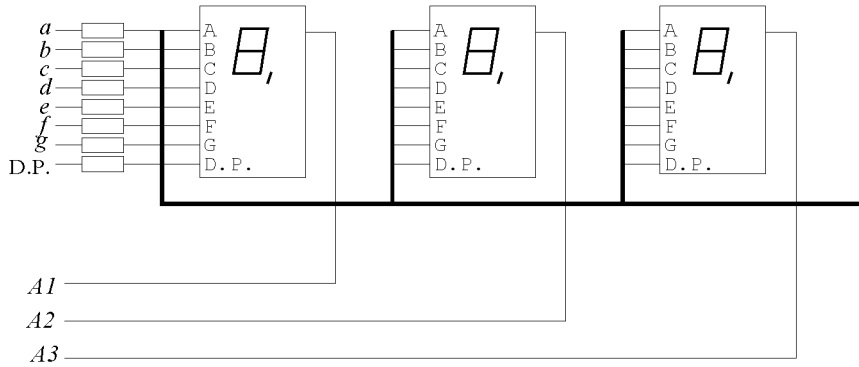
Для того, чтобы высветить на индикаторе какую-либо цифру, необходимо подключить общий анод к источнику питания +5 В и подать через резисторы лог. 0 на те катодные входы, которые должны в данный момент светиться.

Для того, чтобы преобразовать двоичный или двоично-десятичный код во входные сигналы индикатора, применяют дешифратор-формирователь:



На входы $D_0 \ldots D_3$ подается двоичное число $\overline{D_3 D_2 D_1 D_0}$, соответствующее 16-ричной цифре.

Если необходимо вывести длинное десятичное или 16-ричное число, делают **линейку индикаторов**. При этом применяют мультиплексирование выходной информации:



На аноды A_1, A_2, A_3 ...по очереди подают лог. 1, сопровождая это той комбинацией на катодах, которая должна быть высвечена на данном индикаторе.

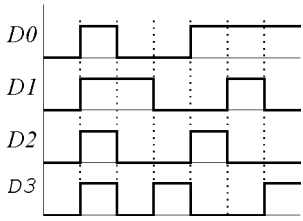
Отметим, что помимо светодиодных индикаторов, распространены жидкокристаллические (ЖКИ). Они требуют довольно сложной последовательности входных сигналов.

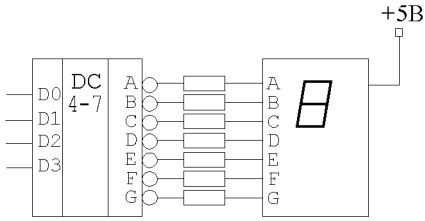
ВОПРОСЫ И ЗАДАЧИ

1. Построить таблицу истинности двоично–7-сегметного дешифратора:

D_3	D_2	D_1	D_0	a	b	c	d	e	f	g

2. Что высветит индикатор?





3. Придумать схему кнопки с электронным подхватом. Такая кнопка “зажигается” при первом нажатии и “гаснет” при втором.

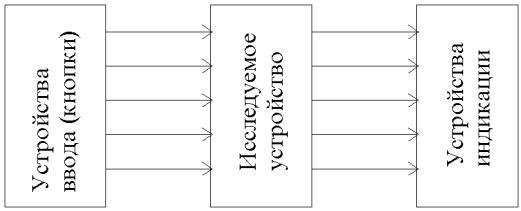
Часть III

Практические работы

Практическая работа 1

Построение таблиц истинности

Для построения таблиц истинности используется установка, состоящая из 3 элементов:



В качестве формирователя входных сигналов можно использовать блоки кнопок и тумблеров. Если кнопка нажата (тумблер включен), то на соответствующем выходе присутствует логическая единица, в противном случае—“0”. (Набор модулей избавляет вас от необходимости задумываться об устройстве тех или иных элементов. Создатели набора позаботились о том, чтобы все микросхемы были правильно включены, питание подавалося на нужные контакты и т.д.).

В качестве индикатора выходных сигналов будем использовать светодиоды. Светодиод на модуле светится, если к соответствующему входу приложена логическая единица.

В каждом из упражнений необходимо сделать следующее:

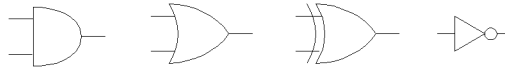
- Изобразить в тетради схему установки (включающую исследуемое устройство, ключи или кнопки, светодиоды). Изобразить в тетради таблицу истинности. Заполнить левую часть таблицы (входы). Правую пока не заполнять.

- Собрать схему из элементов, имеющихся в наборе.

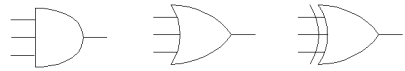
- По очереди выставлять на входах комбинации, соответствующие строкам таблицы истинности. Записывать состояния выходов в таблицу.

УПРАЖНЕНИЯ

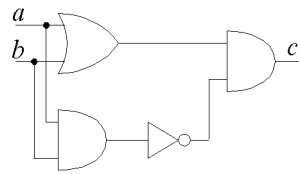
1. Проверить таблицы истинности элементов



2. Построить таблицу истинности многовходовых элементов



3. Построить таблицу истинности схемы



Практическая работа 2

Синтез схем с заданными характеристиками

Данная лабораторная работа относится к материалу занятий 7–10. Прежде чем начинать выполнение работы, рекомендуется ознакомиться с материалом этих занятий и выполнить упражнения к ним.

Для каждого упражнения необходимо:

- Построить таблицу истинности устройства.
- Построить логическую функцию, реализующую данное устройство.
- Попытаться упростить логическую функцию тем или иным способом. Выписать упрощенную формулу.
- Построить схему.
- Проверить, правильно ли работает схема. Для этого подать на входы все возможные комбинации нулей и единиц и сравнить состояние выхода с таблицей истинности.

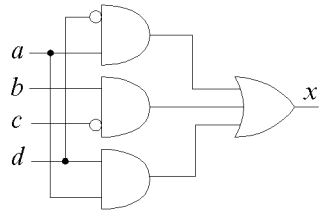
УПРАЖНЕНИЯ

1. Есть известная задачка про Волка, Козу и Капусту. Их надо перевести в лодке с берега на берег, но в лодку нельзя одновременно сажать Волка и Козу, Козу и Капусту. Кроме того, нельзя, чтобы лодка была пустой. Согласно этим правилам, построить прибор с тремя входами: a — Волк, b — Коза, c — Капуста и выходом f — “Можно сажать вместе”.

2. Построить схему по таблице истинности:

a	b	c	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

3. Упростить схему



Практическая работа 3

Шифратор и дешифратор

Необходимо изучить материал занятия 13 и выполнить упражнения к нему. В последнем упражнении используется материал занятия 17.

УПРАЖНЕНИЯ

1. **Приоритетный шифратор**
   Необходимо исследовать экспериментально таблицу истинности приоритетного шифратора. Для этого

   - Найти шифратор в наборе. (Он называется "8 to 3 priority encoder").
   - Соединить входы $A0$, $A1$, $A2$, $A3$ с тумблерами, выходы $X0$, $X1$ со светодиодами. Все остальные входы и выходы ни к чему не подсоединять. (Модуль устроен так, что на всех неподсоединенных входах — нули). При этом модуль работает как приоритетный шифратор $4 \to 2$.
   - Перебирая все возможные состояния входов и наблюдая состояния выходов, заполнить таблицу истинности

$A0$	$A1$	$A2$	$A3$	$X1$	$X0$	$\overline{X_1 X_0}$

   (В самой правой колонке писать число, которое выдает шифратор).

2. **Дешифратор**
   Исследовать работу дешифратора $3 \to 8$. Для этого

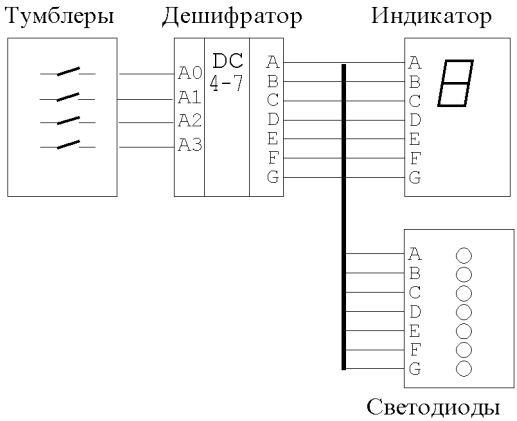
   - Найти в наборе демультиплексор (дешифратора в наборе нет, его придется сделать из демультиплесора). Демультиплексор называется "1 to 8 demultiplexer".
   - Подать на информационный вход D лог. 1. При этом демультиплексор становится дешифратором (объясните, почему).
   - Соединить входы $A0$, $A1$, $A2$ с тумблерами, выходы $X0$, $X1$, $\ldots$,$X7$ со светодиодами. Входы DIS и $\overline{OE}$ не подсоединять.
   - Перебирая все возможные состояния входов и наблюдая состояния выходов, заполнить таблицу истинности

$\overline{A_2A_1A_0}$	$A2$	$A1$	$A0$	$X0$	$X1$	...	$X7$

(В самой левой колонке писать число, которое подается на дешифратор).

3. **Дешифратор, обслуживающий семисегментный индикатор**
В данном упражнении строится самая простая система вывода цифровой информации.

- Найти в наборе 7-сегментный индикатор и дешифратор $4 \to 7$ (он называется "binary to 7 seg. decoder").
- Соединить входы $A0$, $A1$, $A2$, $A3$ с тумблерами, остальные входы ни к чему не подсоединять. Выходы $a \ldots g$ подсоединить к одноименным входам индикатора. Кроме того, входы $a \ldots g$ подсоединить к светодиодам:



- Перебирая все возможные состояния входов и наблюдая состояния выходов, заполнить таблицу истинности

$\overline{A_3A_2A_1A_0}$	A_3	A_2	A_1	A_0	a	b	c	d	e	f	g	

В самой правой колонке зарисовать символ, который отображается на индикаторе.

Практическая работа 4

Мультиплексор и демультиплексор; мультиплексирование данных при передаче по линии связи

Необходимо изучить материал занятия 13 и выполнить упражнения к нему.

УПРАЖНЕНИЯ

1. Мультиплексор

Исследуем работу мультиплексора.

- Найти мультиплексор в наборе (он называется "multiplexer 8 to 1"). В качестве входных сигналов использовать выходы генератора импульсов ("square wave generator"). Следует запомнить, что число, стоящее у выхода генератора, есть частота сигнала на этом выходе (в Герцах).
- Соединить выходы "1" ... "128" со входами $D0 \ldots D7$ мультиплексора.
- Соединить входы $A0$, $A1$, $A2$ с тумблерами.
- Соединить выход мультиплексора со светодиодом.
- Подавать на входы $A0$, $A1$, $A2$ последовательно двоичные числа $\overline{A_2A_1A_0} = 0 \ldots 7$. Наблюдать поведение выходного сигнала. При каждом значении $\overline{A_2A_1A_0}$ выяснить, с какого информационного входа сигнал подается на выход. (Может оказаться непросто различить сигналы 64 Гц и 128 Гц. В этом случае подсоединить соответствующий вход к свободному тумблеру и пощелкать).
- Составить таблицу

$\overline{A_2A_1A_0}$	$A2$	$A1$	$A0$	$D0$	$D1$	...	$D7$

В правой части таблицы ставить крестик под обозначением того входа, с которого подается сигнал на выход.

2. Демультиплексор

Исследуем работу демультиплексора.

- Найти демультиплексор в наборе (он называется “demultiplexer 1 to 8”). В качестве входного сигнала использовать выход 2 Гц генератора импульсов.

- Соединить выход “2” генератора со входом D демультиплексора.

- Соединить входы $A0$, $A1$, $A2$ с тумблерами.

- Соединить выходы мультиплексора со светодиодами.

- Подавать на входы $A0$, $A1$, $A2$ последовательно двоичные числа $\overline{A_2A_1A_0} = 0 \ldots 7$. Наблюдать поведение выходного сигнала. При каждом значении $\overline{A_2A_1A_0}$ выяснить, на какой информационный вход подается сигнал со входа.

- Составить таблицу

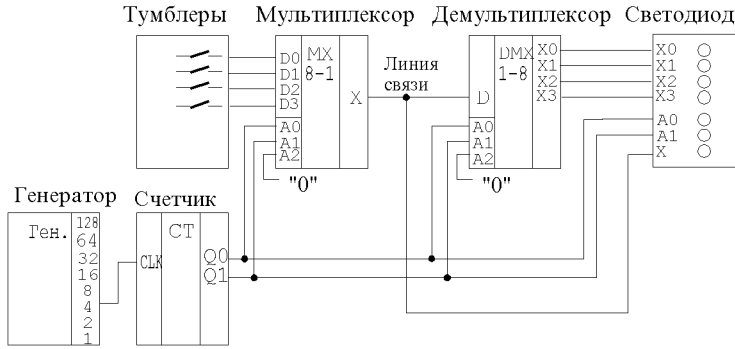
$\overline{A_2A_1A_0}$	$A2$	$A1$	$A0$	$X0$	$X1$	$\ldots$	$X7$

  В правой части таблицы ставить крестик под обозначением того выхода, на который подается сигнал со входа.

3. Линия передачи данных с мультиплексированием

Пусть необходимо передать данные с большого количества кнопок (скажем, 256) на такое же количество светодиодов. При этом состояние кнопок меняется медленно. В этом случае необязательно тянуть 256 проводов. Достаточно протянуть один провод, по которому (по очереди) будет передаваться состояние всех кнопок, и 8 адресных линий, на которых будет очень быстро перебираться 256 номеров входов-выходов. Когда на адресных линиях число 0, передается состояние кнопки с номером 0, когда на адресных линиях — 1, передается состояние кнопки с номером 1 и т.д. Для вывода состояния всех кнопок на одну линию используется мультиплексор, для обратного преобразования — демультиплексор.

В этой работе строится линия связи, соединяющая 4 тумблера с 4 светодиодами. При этом будет одна линия данных и две адресных линии:



На светодиоды выведено состояние выходов, а также **диагностические сигналы** (промежуточные сигналы, состояние которых поможет понять работу схемы). Это сигналы с линии данных D и адресных линий $A0$ и $A1$. Генератор и счетчик используются для быстрого перебора двоичных чисел от 0 до 3.

- Зарисовать схему в тетрадь.
- Найти в наборе мультиплесор, демультиплексор, генератор (square wave generator) и счетчик (binary up/down counter).
- Собрать схему, как показано на рисунке. Внимательно соединять входы и выходы в соответствии с обозначениями. На входы $A2$ мультиплексора и демультиплексора подать лог. 0 (соответствующий сигнал имеется на каждом модуле). Это соответствует тому, что из 8 входов (или выходов) выбираются первые 4.
- Подать на вход CLK счетчика сигнал частотой 1 Гц (с выхода 1 генератора). Убедиться в том, что на линиях A_1, $A0$ перебираются все 2-разрядные двоичные числа.
- Подать на вход CLK счетчика сигнал частотой 4 Гц. Поочередно пощелкать тумблерами. Пронаблюдать, что происходит на выходах $X0 \ldots X3$ и на линии X.
- Подать на вход CLK счетчика сигнал частотой 128 Гц. Поочередно пощелкать тумблерами. Пронаблюдать, что происходит на выходах $X0 \ldots X3$ и на линии X.

Практическая работа 5

Реализация логических функций с помощью дешифратора и мультиплексора

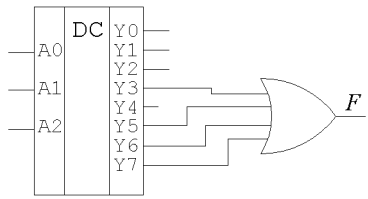
Реализация логических функций с помощью дешифратора Дешифратор может использоваться для простой реализации логических функций по таблице истинности. Рассмотрим в качестве примера устройство мажоритарной логики (оно рассматривалось в занятии 8). Его таблица истинности:

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Переобозначим входы как $A_2 = a$, $A_1 = b$, $A_0 = c$ и построим таблицу истинности

$\overline{A_2A_2A_0}$	A_2	A_1	A_0	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

Построим схему **(схема 1)**:

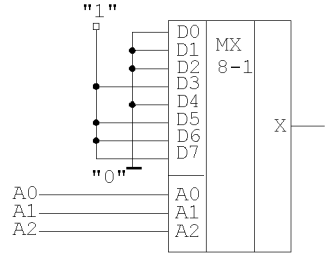


Выходы, соответствующие единицам в правом столбце таблицы истинности, подключены к схеме “ИЛИ”.

Разберите работу данной схемы самостоятельно.

Реализация логических функций с помощью мультиплексора

Рассмотрим как такая же таблица истинности с помощью мультиплексора **(схема 2)**:



Если при некоторых значениях A_0, A_1, A_2 на выходе (согласно таблице истинности) должен быть 0, то вход с номером $\overline{A_2 A_1 A_0}$ подключается к 0, а если должна быть 1, то вход подключается к 1.

УПРАЖНЕНИЯ

1. Проверка работы схемы 1

В этом задании необходимо собрать схему 1 и построить ее таблицу истинности.

- Найти в наборе демультиплексор (дешифратора в наборе нет, его придется сделать из демультиплесора). Демультиплексор называется “1 to 8 demultiplexer”.
- Подать на информационный вход D лог. 1. При этом демультиплексор становится дешифратором.
- Соединить входы $A0$, $A1$, $A2$ с тумблерами, Входы DIS и $\overline{OE}$ не подсоединять. Выходы $X3$, $X5$, $X6$ и $X7$ подсоединить к многовходовому “ИЛИ” (четырехвходовое “ИЛИ” можно собрать, соединив трехвходовое и двухвходовое). Выход схемы “ИЛИ” соединить со светодиодом.
- Перебирая все возможные состояния входов и наблюдая состояния выходов, заполнить таблицу истинности схемы (в тетради). Проверить, соответствует ли таблица истинности требуемой.

2. Проверка работы схемы 2

В этом задании необходимо собрать схему 2 и проверить ее таблицу истинности.

- Найти мультиплексор в наборе (он называется "multiplexer 8 to 1").
- Подать на входы $D0$, $D1$, $D2$, $D4$ лог. 0, на входы $D3$, $D5$, $D6$, $D7$ лог. 1. Соединить входы $A0$, $A1$, $A2$ с тумблерами. Соединить выход мультиплексора со светодиодом.
- Перебирая все возможные состояния входов и наблюдая состояния выходов, заполнить таблицу истинности схемы (в тетради). Проверить, соответствует ли таблица истинности требуемой.

3. Реализовать таблицу истинности

A_2	A_1	A_0	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

с помощью дешифратора и мультиплексора. Для этого

- Построить схему, реализующую данную таблицу истинности с помощью дешифратора.
- Повторить все пункты упражнения 1 данной работы для построенной схемы.
- Построить схему, реализующую данную таблицу истинности с помощью мультиплексора.
- Повторить все пункты упражнения 2 данной работы для построенной схемы.

Практическая работа 6

Устройства арифметики: компаратор, полусумматор, сумматор

Данная работа соответствует материалу занятия 14.

УПРАЖНЕНИЯ

1. **Компаратор**
   В этом упражнении исследуется работа компаратора (устройства для сравнения чисел). Для этого:

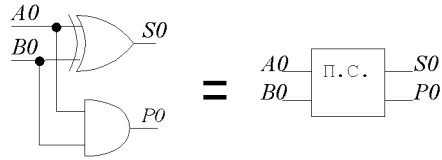
   - Найти в наборе компаратор (он называется 4-bit comparator).
   - Соединить входы $A0$, $A1$, $A2$, $A3$ с кнопками, входы $B0$, $B1$, $B2$, $B3$ с тумблерами.
   - Подать на вход $A < B_i$ лог. 0, на вход $A = B_i$ лог. 1. Эти входы предназначены для каскадирования (для включения нескольких компараторов таким образом, чтобы можно было сравнивать числа большой разрядности). Указанная комбинация соответствует первому (или младшему) компаратору.
   - Соединить все выходы со светодиодами.
   - Выставить на тумблерах число 8. (Для этого установить $B3$ в 1, а $B0$, $B1$, $B2$ — в 0.
   - Перебирая все возможные состояния входов $A0 \ldots A3$ с помощью кнопок и наблюдая состояния выходов, заполнить в тетради таблицу истинности устройства:

$A3$	$A2$	$A1$	$A0$	$A<B$	$A=B$	$A>B$	$A\leq B$	$A\neq B$	$A\geq B$

   - Объяснить работу всех выходов схемы.
   - Посвторить последние 3 пункта, выставляя на входах $B0 \ldots B3$ числа 0 и F.

2. Полусумматор

- Собрать полусумматор по схеме

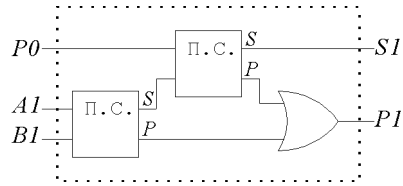


  Схему зарисовать.
- Подключить входы к тумблерам, выходы к светодиодам.
- Перебирая состояния входов и наблюдая состояния выходов, заполнить таблицу истинности

a	b	p	s

3. Ячейка сумматора

- Найти в наборе модуль с ячейками полусумматора (half adder/half subtractor, HA/HS).
- Собрать ячейку сумматора по схеме



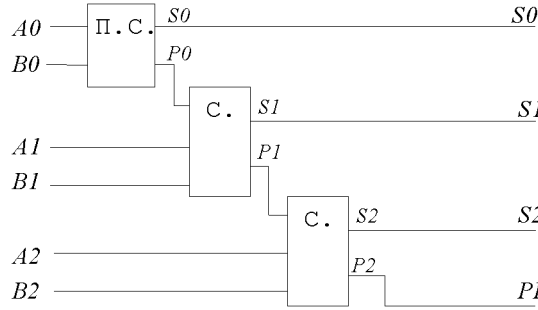
  Выход X модуля полусумматора соответствует разряду суммы S, а выход C — переносу P. Входы $\overline{A}/S$ не подключать. При этом на них будет лог. 0, что соответствует команде “суммировать”. (Если подать на эти входы 1, полусумматор превратится в полувычитатель).
- Подключить входы к тумблерам, выходы к светодиодам.
- Перебирая состояния входов и наблюдая состояния выходов, заполнить таблицу истинности

a_i	b_i	p_{i-1}	p_i	s_i

- Сравнить полученную таблицу истинности с требуемой.

4. Построение сумматора из ячеек

- Найти в наборе модуль с ячейками сумматора (triple binary full adder / subtractor, FA/FS).
- Собрать сумматор из ячеек по схеме



Выходы X ячеек соответствуют разрядам суммы S_i, а выходы C — переносам P_i.

- На вход C_i0 младшего разряда подать 0, входы $A0$, $A1$, $A2$ подключить к кнопкам, входы $B0$, $B1$, $B2$ подключить к тумблерам. Выходы схемы подключить к светодиодам. Входы $\overline{A}/S$ не подключать.
- Выполнить действия с помощью построенного сумматора:
  $0 + 5$,
  $1 + 2$,
  $4 + 4$,
  $5 + 6$,
  $6 + 7$.

5. Сумматор/вычитатель

- Найти в наборе модуль сумматора. Он называется “4-bit binary full adder / subractor” (4 bit add/sub).
- Соединить входы $A0$, $A1$, $A2$, $A3$ с кнопками, входы $B0$, $B1$, $B2$, $B3$ с тумблерами.
- Подать на вход $\overline{A}/S$ лог. 0. При этом модуль работает как сумматор. Вход C_i (перенос из предыдущего сумматора) не подключать.
- Подключить выходы к светодиодам.

- Выполнить на сумматоре действия:
  0 h + 3 h,
  5 h + 6 h,
  8 h + 8 h,
  9 h + A h,
  B h + C h.
  Записать ответы в тетради.
  В каком случае активируется сигнал C_{out}?
- Подать на вход $\overline{A}/S$ лог. 1. При этом модуль работает как вычитатель.
- Выполнить на сумматоре действия:
  0 h − 3 h,
  8 h − 6 h,
  8 h − 8 h,
  9 h − A h,
  C h − 8 h.
  Записать ответы в тетради.
  В каком случае активируется сигнал C_{out}?
- С помощью вычитателя перевести в дополнительный код числа
  5 h,
  9 h,
  C h.
  (Для этого числа надо вычитать из нуля).

Практическая работа 7

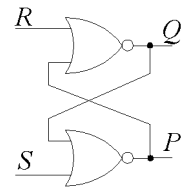
RS-триггеры; тактируемый RS-триггер; D-триггер; JK-триггер

Данная работа соответствует материалу занятия 15.

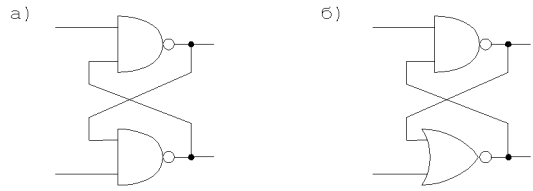
УПРАЖНЕНИЯ

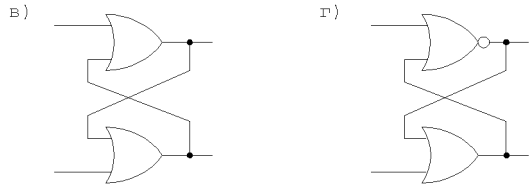
1. *RS*-триггер

- Собрать триггер по схеме



- Соединить входы R и S с кнопками, выходы со светодиодами.
- Подавая на входы комбинации $(1,0)$, $(0,1)$, $(1,1)$ пронаблюдать режимы установки, режим сброса и запрещенное состояние.
- Перевести триггер в режим установки, а затем в режим хранения (для этого надо нажать кнопку S отпустить). Пронаблюдать, что произойдет с триггером. Перевести триггер в режим сброса, а затем в режим хранения (Нажать кнопку R и отпустить). Пронаблюдать поведение триггера.
- Поочередно собрать схемы

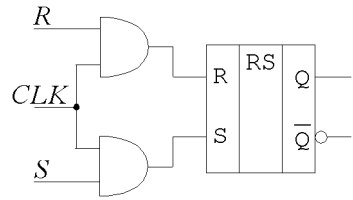




Подсоединить входы к кнопкам, выходы к светодиодам. Перебирая состояния входов, попытаться добиться от устройств работы в режимах установки, сброса, хранения. Определить, какие устройства являются триггерами.

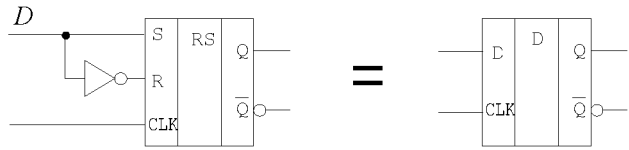
2. Триггер типа ведущий-ведомый

- Найти в наборе модуль с *RS*-триггерами (NOR RS-latch).
- По схеме



собрать *два* синхронных триггера.

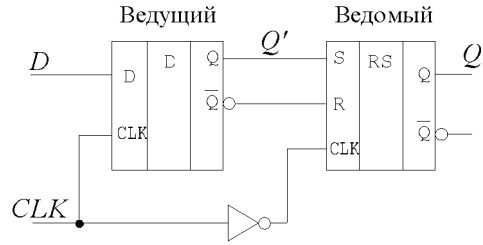
- По схеме



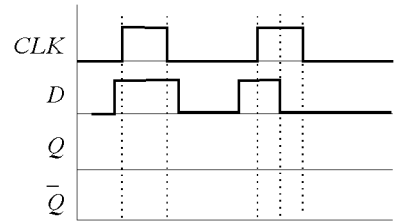
один из синхронных *RS*-триггеров превратить в *D*-триггер.

- Проверить работу *D*-триггера. Для этого подключить входы к тумблерам, выходы к светодиодам. Подать на вход *CLK* единицу (открыть триггер). Подать на вход *D* единицу. Подать на вход *CLK* ноль (закрыть триггер). При этом в триггер запишется 1. Похожим образом записать 0. Повторить процедуру, однако сигнал на *D* выставлять раньше, чем открывать триггер. Зарисовать временные диаграммы проделанных операций.

- По схеме



собрать триггер типа "ведущий—ведомый". Подсоединить входы к кнопкам, выход Q и промежуточную точку Q' подключить к светодиодам. Экспериментально заполнить временную диаграмму



3. JK-триггер

- Найти в наборе модуль с JK-триггерами (JK flip-flop).
- Подключить входы одного из триггеров к тумблерам, выходы к светодиодам.
- Проверить работу асинхронных входов R и S. Для этого подать на входы CLK, J и K нули. Выполнить установку и сброс триггера с помощью входов S и R (как у обычного RS-триггера).
- Подать на входы R и S нули. Исследовать поведение синхронных входов. Для этого установить $J = 1$, $K = 0$ и подать короткий импульс CLK. Установить $J = 0$, $K = 0$ и подать короткий импульс CLK. Установить $J = 0$, $K = 1$ и подать короткий импульс CLK. Установить $J = 0$, $K = 0$ и подать короткий импульс CLK.
- Исследовать работу триггера в счетном режиме. Для этого Установить $J = 1$, $K = 1$ и подать 5 импульсов CLK.

Практическая работа 8

Регистр-защелка; сдвиговый регистр

Данная работа соответствует материалу занятия 16.

УПРАЖНЕНИЯ

1. Регистр-защелка

- Найти в наборе модуль с регистром-защелкой. Он называется 8 bit D-type register. Этот регистр *срабатывает по переднему фронту*, т.е. запись информации в него происходит в момент перепада из низкого уровня сигнала CLK в высокий.
- Подключить входы $D0 \ldots D3$ к тумблерам, выходы $Q0 \ldots Q7$ к светодиодам. На входы $D4 \ldots D7$ подать 0. Вход $\overline{OE}$ не подключать. Вход CLK подключить к кнопке.
- Записать в регистр число 5. Для этого: выставить двоичное число 5 на тумблерах, затем нажать и отпустить кнопку. Заметить момент, в который изменится состояние выходов.
- Проверить, срабатывает ли регистр по спаду. Для этого: Нажать кнопку (пока не отпускать), выставить число F h на тумблерах, отпустить кнопку. Нажать и отпустить кнопку еще раз. Заметить, когда изменилось состояние выходов.

2. Сдвиговый регистр из регистра-защелки

- Изменить подключение регистра, который исследовался в предыдущем упражнении. Подключить вход D_0 к тумблеру, вход $D1$ к выходу $Q0$, вход $D2$ к выходу $Q1,\ldots$, вход $D7$ к выходу $Q6$. Все выходы оставить подключенными к светодиодам. Вход CLK оставить подключенным к кнопке.
- Выставить на тумблере 0. Нажать кнопку 8 раз. Пронаблюдать состояние выходов.
- Выставить на тумблере 1. Нажать кнопку 8 раз. Пронаблюдать состояние выходов.
- Манипулируя тумблером и кнопкой, добиться того, чтобы в регистре было записано число A5 h. Нарисовать временную диаграмму сигналов $D0$ и CLK в этом случае.

3. Сдвиговый регистр

- Найти в наборе сдвиговый регистр. Он называется 4-bit shift register.
- Подключить вход CLK к кнопке, входы $R/\overline{L}$, PR, $P0$, $P1$, $P2$, $P3$, DSR, DSL к тумблерам. Выходы $Q0 \ldots Q3$ подключить к светодиодам.
- Экспериментально выяснить, какие входы определяют: направление сдвига, предустановку, вход в младший триггер, вход в старший триггер.
- Научиться использовать данный регистр как регистр-защелку. Обратите внимание на то, что вход PR (предустановка) данного регистра является *синхронным*, т.е. для предустановки надо подать уровень 1 на вход PR, а затем фронт сигнала CLK.
- Научиться записывать в регистр число со стороны младшего разряда. Научиться записывать в регистр число со стороны старшего разряда.

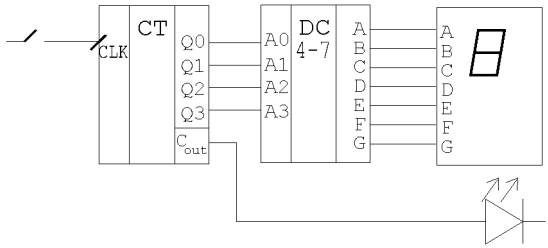
Практическая работа 9

Счетчики

УПРАЖНЕНИЯ

1. Инкрементный и декрементный счет от 0 до 15

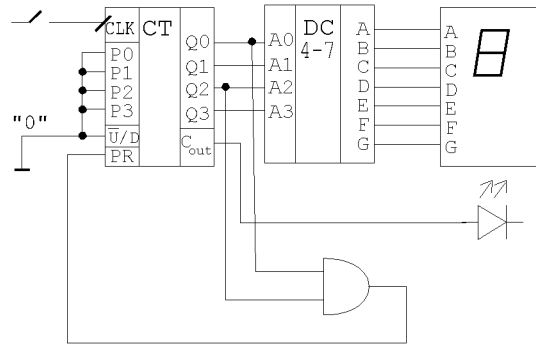
- Найти в наборе двоичный счетчик. Он называется binary up/down counter.
- Подключить вход CLK к кнопке, входы PR, $P0$, $P1$, $P2$, $P3$, RES, $\overline{U}/D$ к тумблерам, на вход C_{in} подать 0. Выход C_{out} подключить к светодиоду. Выходы $Q0 \ldots Q3$ подключить к дешифратору $2 \rightarrow 7$. Выходы дешифратора подать на 7-сегментный индикатор:



- Подать на счетчик последовательно 16 импульсов CLK. Пронаблюдать состояние индикатора. Отметить, в какие моменты изменяется состояние выхода C_{out}.
- Подать на вход $\overline{U}/D$ лог. 1. Подать 16 импульсов CLK. Пронаблюдать состояние индикатора и выхода C_{out}.

2. Инкрементый счет до заданного числа

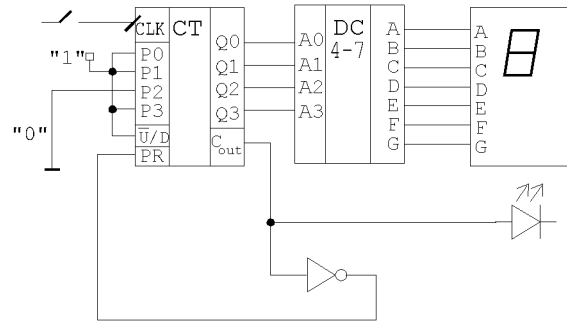
- Использовать схему, собранную в предыдущем упражнении. Подать на вход PR сигнал со схемы "И" как показано на рисунке. На вход схемы "И" подать сигналы, соответствующие единицам в числе 5.



- Подать 16 импульсов CLK. Пронаблюдать состояние индикатора и выхода C_{out}.
- Построить счетчик, который перебирает числа от 0 до 11.

3. Декрементный счет с предзагрузкой

- Изменить схему в соответствии с рисунком:

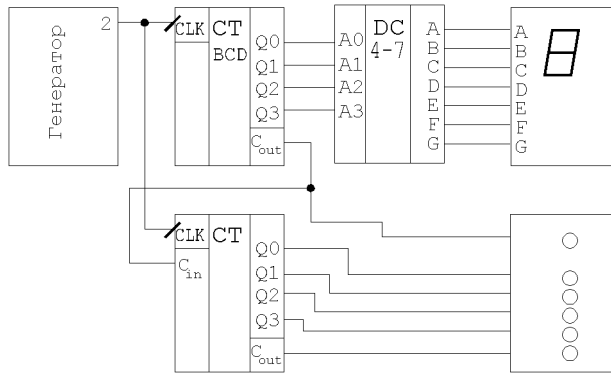


- Подать 16 импульсов CLK. Пронаблюдать состояние индикатора и выхода C_{out}.
- Построить счетчик, который перебирает числа от 9 до 0.

4. Последовательное включение счетчиков

- Найти в наборе двоично-десятичный счетчик. Он называется BCD up/down counter.

- Собрать схему, как показано на рисунке.



Выходы двоично-десятичного счетчика подать на индикатор, выходы двоичного — на светодиоды. Входы CLK и $\overline{U}/D$ счетчиков объединить. На выходы CLK подать сигнал с генератора; на входы $\overline{U}/D$ подать 0. Соединить выход C_{out} (переполнение) двоично-десятичного счетчика со входом C_{in} двоичного. Эти входы и выходы делаются специально для каскадного соединения счетчиков.

- Подать на вход CLK сигнал с выхода “2” генератора. Проанблюдать работу пары счетчиков.

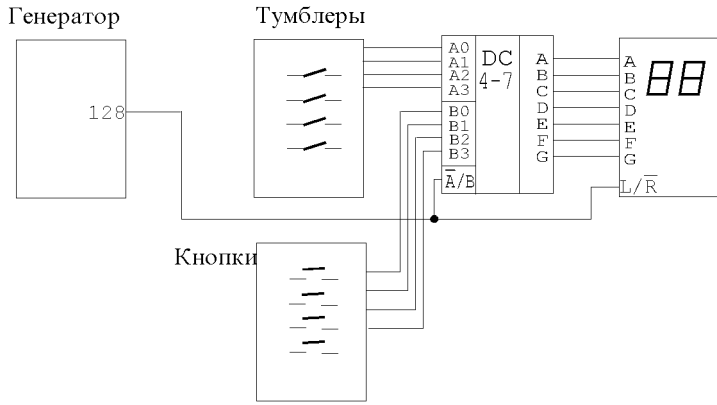
Практическая работа 10

Мультиплексирование при отображении данных

В данной работе используется материал занятия 17.

УПРАЖНЕНИЯ

1. • Собрать схему по рисунку



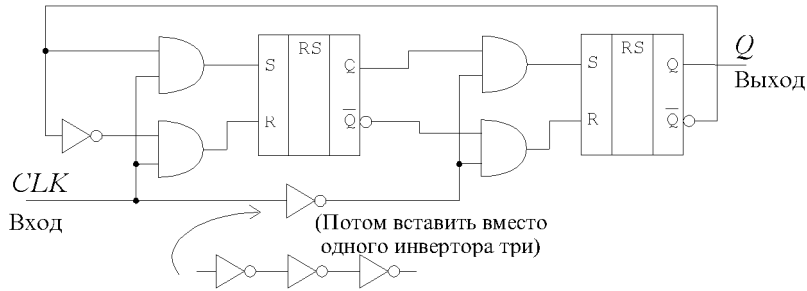
Вход $\overline{A}/B$ дешифратора определяет, какое число подвергается дешифрации. Если на этом входе 0, то обрабатывается число со входов $A0 \ldots A3$, если 1, то со входов $B0 \ldots B3$. Вход $L/\overline{R}$ индикатора определяет, на каком индикаторе отображается число. Если на этом входе 0, то число отображается на правом индикаторе, если 1, то на левом.

- Подать на схему сигнал с выхода 2 Гц генератора. Набирая различные числа тумблерами и кнопками, проследить, как происходит отображение.
- Подать на схему сигнал с выхода 8 Гц генератора, затем 16 Гц, 32 Гц и 128 Гц. Проследить за тем, как изменяется поведение индикатора.

Практическая работа 11

Влияние задержек на работу логических схем

Рассмотрим работу такого устройства:



Устройство представляет собой счетный триггер. Его "происхождение" таково. Вначале собран RS-триггер типа ведущий-ведомый, тактируемый по спаду сигнала CLK. Затем он превращен в D-триггер. Наконец, его выход $\overline{Q}$ соединили со входом, превратив тем самым в счетный триггер.

При нормальной работе состояние выхода Q должно меняться каждый раз при прохождении спада сигнала CLK.

С работой этого устройства связано понятие **логических гонок**. Триггер работает нормально, если задержка сигнала на ведущем RS-триггере *больше*, чем задержка на инверторе в цепи CLK. В противном случае, триггер на очень короткое время (а именно, когда ведущий RS-триггер уже сработал, а инвертор еще нет) становится прозрачным. В большом числе схем это пройдет незамеченным, однако в цепи счетного триггера сигнал успеет "проскочить" оба триггера и выход переключится несколько раз. При этом работа схемы станет неустойчивой или вообще не будет иметь ничего общего с работой счетного триггера.

Для того, чтобы пронаблюдать этот эффект, можно искусственно увеличить задержку на инверторе, вставив три или пять инверторов вместо одного.

УПРАЖНЕНИЯ

1. • Собрать схему, изображенную на рисунке. Воспользоваться указаниями к работе 7 (Триггеры).

- Подключить вход CLK к кнопке, выход Q к светодиоду.
- Несколько раз нажать кнопку. Убедиться в том, что счетный триггер работает нормально.
- Заменить один инвертор в цепи CLK на три. Нажать кнопку несколько раз. Исследовать работу схемы.
- Заменить три инвертора в цепи CLK на пять. Нажать кнопку несколько раз. Исследовать работу схемы.

Сдано в набор 1.10.98.
Подписано в печать 19.11.98
Формат 60 × 90/16.
Бумага типографская N1.
Офсетная печать.
Тир. 500 экз.

ЗАО “РИК Русанова”
103006 Москва, ул. Садовая-Триумфальная, 12-14
Лицензия N065590 от 25.12.97 г.