Логические основы и элементы ЭВМ

1. **Структурные единицы ЭВМ**
2. **Логические операции и базовые элементы компьютера**
3. **Основные цифровые логические схемы**
4. **Арифметико-логические устройства**
5. **Тактовые генераторы**
6. **Память. Триггеры**

Структурные единицы ЭВМ

Структурные единицы ЭВМ

В ЭВМ различают структурные единицы (с точки зрения электроники):

- ***Элементы*** – обработка одной единицы электронных сигналов (битов информации). Часть электронной схемы, которая реализует элементарную логическую функцию, называется ***логический элемент компьютера***.
- ***Узл*ы** – одновременная обработка группы электронных сигналов (информационных слов)
- ***Блоки*** – некоторая последовательность в обработке информационных слов
- ***Устройства*** – выполнение отдельных машинных операций и их последовательности

Цифровые схемы

Цифровой сигнал — это сигнал, принимающий только ***два логических значения,*** которые можно закодировать цифрами 0 и 1.

Цифровая схема – это схема, в которой циркулируют цифровые сигналы.

Обычно электрический сигнал с напряжением от 0 до 1 В представляет одно значение (например, 0), а сигнал с напряжением от 2 до 5 В — другое значение (например, 1).

Напряжение за пределами указанных величин недопустимо.

Цифровые схемы конструируются из небольшого числа простых элементов, называемых ***вентилями***, путем сочетания этих элементов в различных комбинациях.

Вентили

Вентили - крошечные электронные устройства, позволяющие получать различные функции от этих двузначных сигналов. Лежат в основе аппаратного обеспечения, на котором строятся все цифровые компьютеры.

Логически **вентиль** - цифровая схема, в которой есть только ***два логических значения***.

У каждого вентиля есть одно или несколько цифровых входных данных (сигналов, представляющих 0 или 1).

Вентиль вычисляет простые функции этих сигналов, такие как логические И или ИЛИ.

Каждый вентиль формируется из нескольких транзисторов.

Несколько вентилей формируют 1 бит памяти, который может содержать 0 или 1.

Биты памяти, объединенные в группы, например, по 16, 32 или 64, формируют **регистры** (элементы процессора).

Из вентилей также может строиться само ядро вычислительной системы.

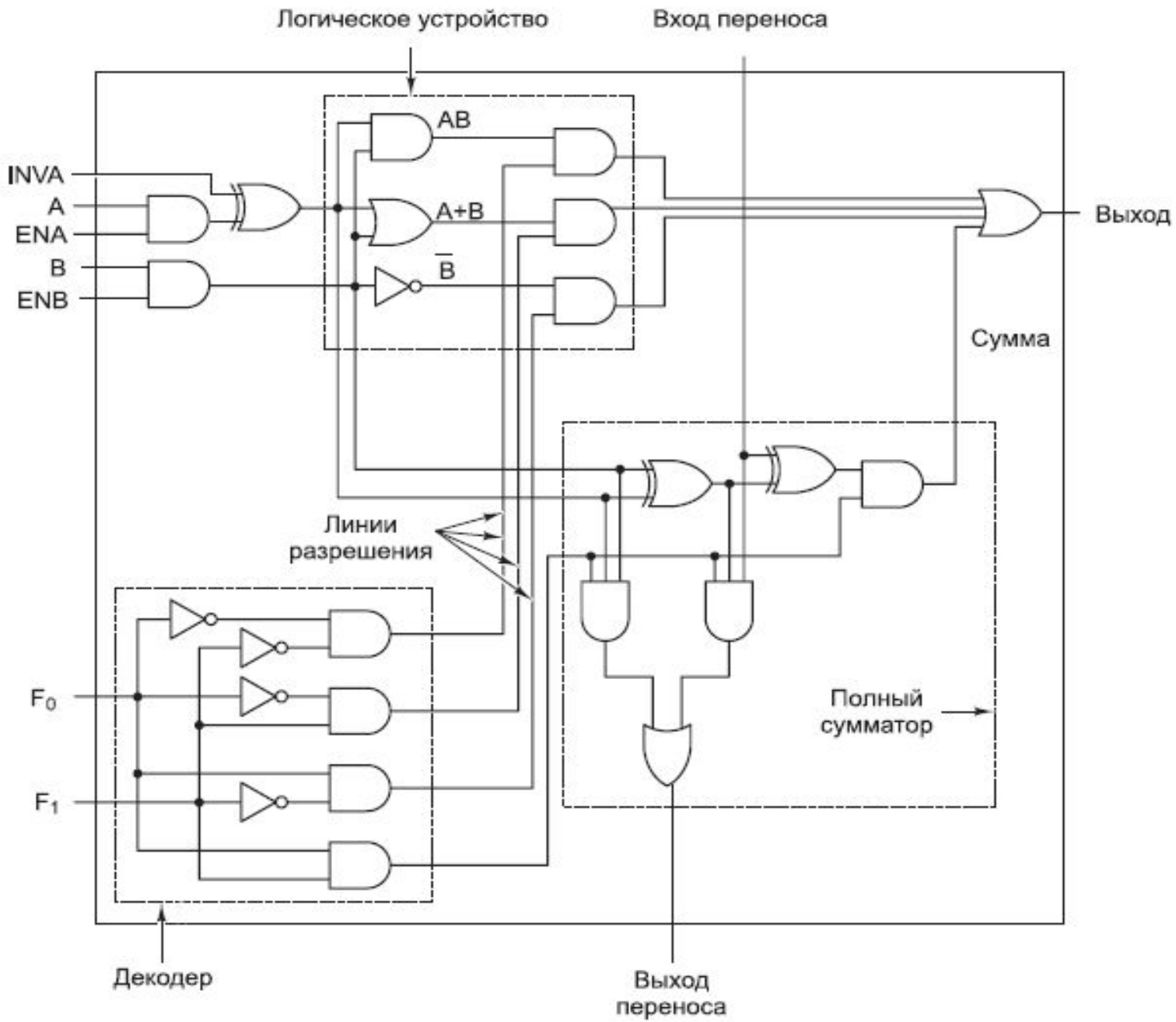


Рис. 3.17. Одноразрядное АЛУ

Логические операции и базовые элементы компьютера

Понятие булевой (логической) функции
Элементарные логические операции
Способы задания булевой функции. Понятие таблицы истинности
Реализация булевых функций в виде электронных схем

Логические операции и базовые элементы компьютера

Булевой (логической) функцией называется функция $y = f(x_1, x_2, ..., x_n)$, у которой результат *y* и аргументы x_1 x_2, ..., x_n могут принимать только два значения – 0 и 1.

С помощью булевых функций удобно описывать преобразование сигналов в цифровых схемах.

Булева функция на входе получает одну или несколько переменных и выдает результат, который зависит только от значений этих переменных.

Булевой (логической) ***переменной*** называется переменная, которая может принимать только два значения – 0 и 1. Этим значениям можно поставить в соответствие значения логических переменных «***ложно***» и «***истинно***».

У булевой функции аргументы и результат – булевы переменные (0, 1).

Булева алгебра (алгебра логики) - совокупность булевых (логических) переменных о операций над ними.

Алгебра логики используется при построении основных узлов ЭВМ (дешифратор, сумматор, шифратор).

Элементарные логические операции

Логическая операция	Обозначение
«И» - логическое умножение. Результат равен 1, если обе переменные равны 1, и 0 – в противном случае	x∧y или x·y
«ИЛИ» - логическое сложение. Результат равен 1, если хотя бы одна переменная равна 1, и 0 – в противном случае	x∨y или x+y
«НЕ» - отрицание (инверсия). Результат равен 1, если переменная равна 0, и 0 – в противном случае	$\overline{x}$ или $\neg x$
«И-НЕ», отрицание логического умножения. Результат обратен результату операции И.	$\overline{x \wedge y}$ или $\overline{x \cdot y}$
«ИЛИ-НЕ», отрицание логического сложения.	$\overline{x \vee y}$ или $\overline{x + y}$

Часть электронной схемы, которая реализует элементарную логическую функцию, называется ***логический элемент компьютера***.

Логические операции И, ИЛИ, НЕ достаточно просто реализуются с помощью электронных схем, например:

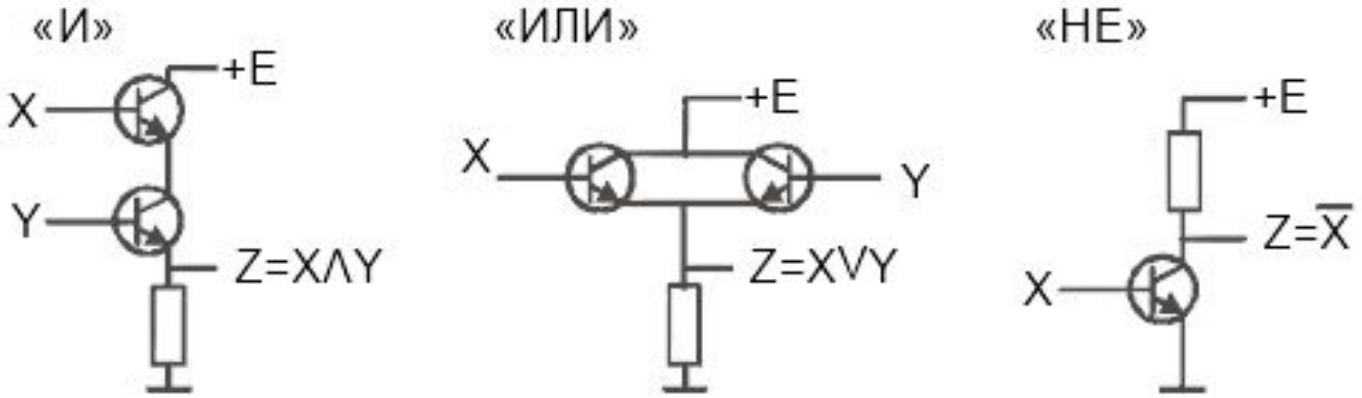


Рис.3.4

Способы задания булевой функции

Способы задания булевой функции

1) ***Аналитический***. С помощью ***логического выражения***

$y = f_1(x_1, x_2) = \mathbf{x_1 + x_2}$ **(x_1 ИЛИ x_2)**

$y = f_2(x_1, x_2, x_3) = \mathbf{x_1 + x_2 \cdot x_3}$ **(x_1 ИЛИ (x_1 И x_2))**.

2) ***Таблица истинности*** задает значения логической (булевой функции) при различных всевозможных значениях аргументов (входных переменных).

Z=x + y

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Z=x· y

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Z =$\bar{x}$

X	Z
0	1
1	0

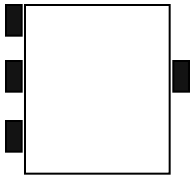
C=x· y, Z= $\bar{x} \cdot y + x \cdot \bar{y}$

X	Y	C	Z
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$C_{вх}$	X	Y	$C_{вых}$	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Электронная схема

Свойства ***электронных схем*** удобно задавать с помощью таблиц истинности, которые определяют требуемые выходные значения сигналов при различных значениях входных сигналов (ф-я большинства)



A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Рис. Условное обозначение элемента и его таблица истинности (возможная)

Правило построения логического выраже*ния*

Встает задача, как по заданной таблице истинности построить ее логическое выражение.

Правило построения логического выражения

1. Каждой строке, где выход равен 1 поставить в соответствие произведение входных переменных, причем, если значение входной переменной в данной строке = 1, то переменная берется сама по себе, а если значение входной переменной = 0, то берется ее инверсия (отрицание), т. е. или x или $\bar{x}$.
2. Полученные произведения суммируются (логически)

Пример с рис.

$$M = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC.$$

Пример.

X	Y	C	Z
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$C = x \cdot y,$

$Z = \bar{x} \cdot y + x \cdot \bar{y}$

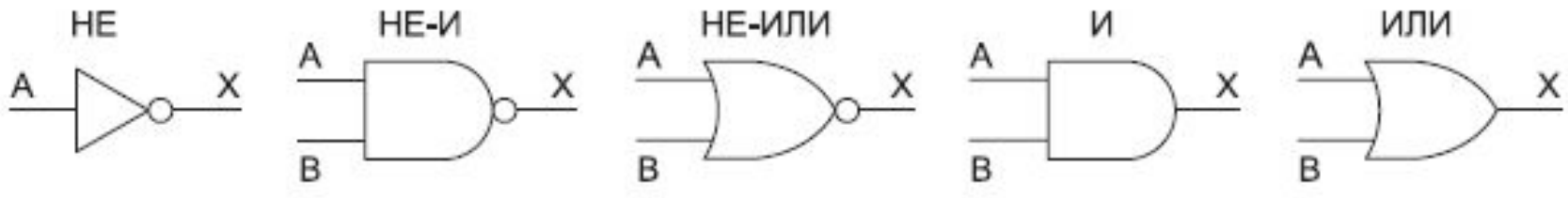
На схемах каждому произведению соответствует вентиль И, а каждой сумме вентиль ИЛИ, (у которых число входов может быть больше 2).

Правило построения схем

На схемах каждой логической операции соответствует свой вентиль:

каждому произведению соответствует вентиль И, а каждой сумме вентиль ИЛИ, (у которых число входов может быть больше 2).

Обозначения вентилей на схемах



A	X
0	1
1	0

а

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

б

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

в

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

г

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

д

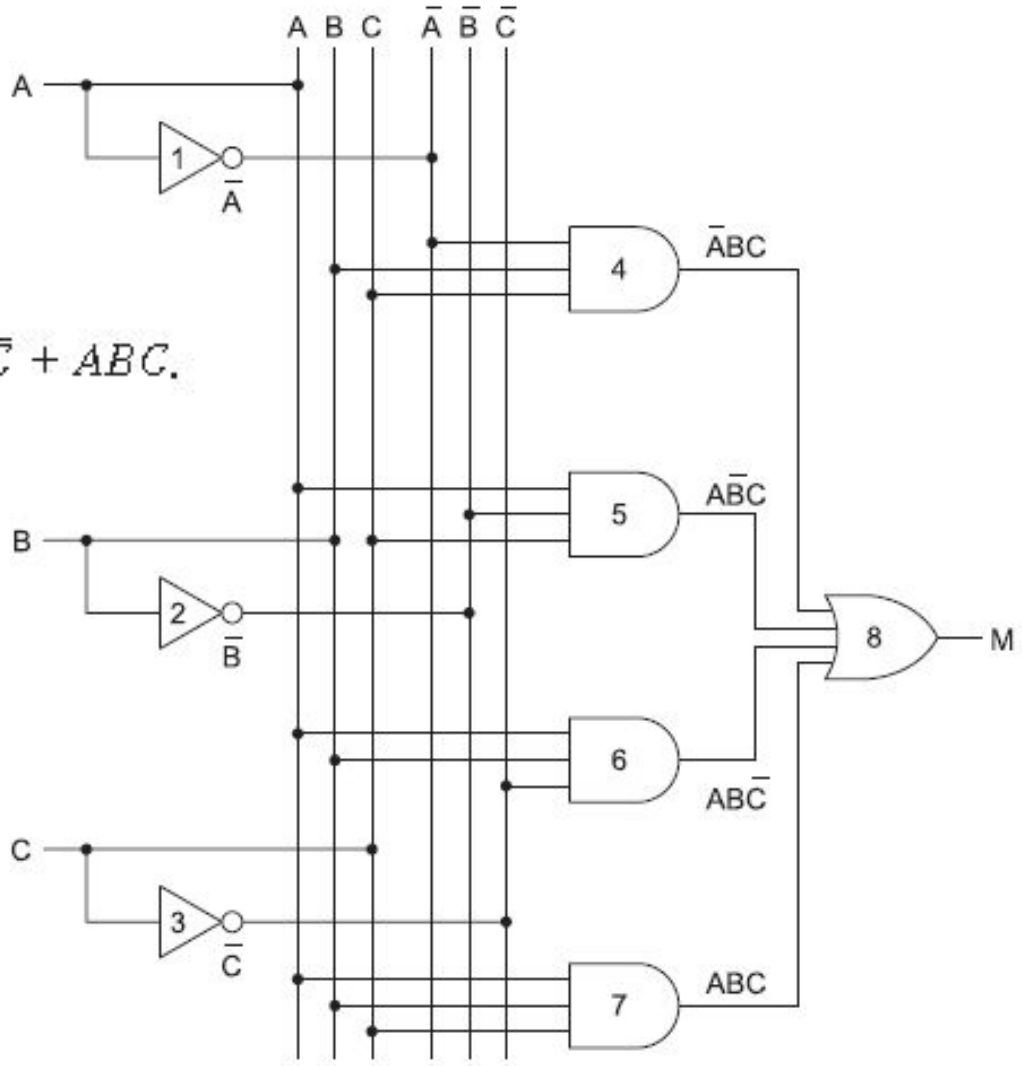
Рис. 3.2. Значки для изображения пяти основных вентилей. Режимы работы функции для каждого вентиля

Пример. Функция большинства. Построение электронной схемы

$$M = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC.$$

A	B	C	M
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

а



б

Рис. 3.3. Таблица истинности для функции большинства от трех переменных (*а*); схема реализации этой функции (*б*)

Правило построения электронной схемы булевой функции:

Правило построения электронной схемы булевой функции:
1. Составить таблицу истинности для данной функции.
2. Включить в схему инверторы, чтобы иметь возможность инверсии каждого входного сигнала.
3. Нарисовать вентиль И для каждой строки таблицы истинности с результатом 1.
4. Соединить вентили И с соответствующими входными сигналами.
5. Вывести выходы всех вентилей И и направить их на вход вентиля ИЛИ.

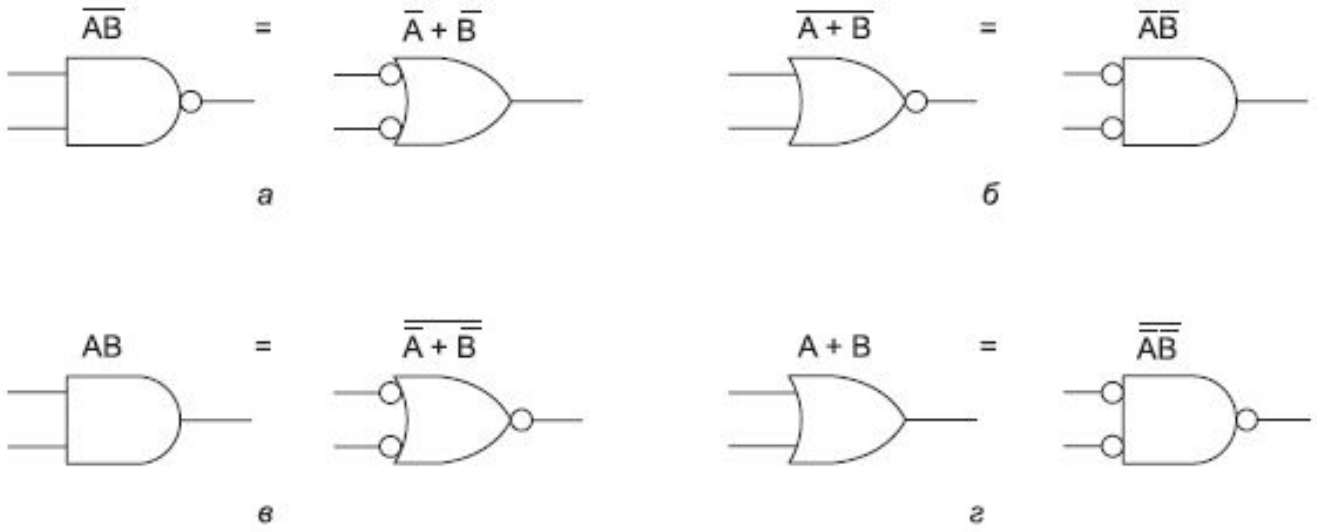


Рис. 3.6. Альтернативные представления некоторых вентилей: НЕ-И (*а*); НЕ-ИЛИ (*б*); И (*в*); ИЛИ (*г*)

Основные цифровые логические схемы

Комбинаторные схемы: мультиплексоры, декодеры, компараторы

Арифметические схемы: схемы сдвига, сумматоры

Понятие интегральной схемы

Вентили производятся и продаются не по отдельности, а в модулях, которые называются **интегральными схемами (ИС)**, или **микросхемами**.

Интегральная схема представляет собой квадратный кусок кремния, размер которого зависит от количества вентилей, необходимых для реализации компонентов. Размеры маленьких интегральных схем обычно составляют около 2 х 2 мм, большие микросхемы могут иметь размеры около 18 х 18 мм. Микросхемы обычно помещаются в прямоугольные пластиковые или керамические корпуса значительно большего размера, если для обмена данными с внешним миром микросхеме требуется много выводов. Каждый вывод соединяется с входом или выходом какого-нибудь вентиля, с источником питания или с «землей». 1 микросхема – до 1 млрд транзисторов.

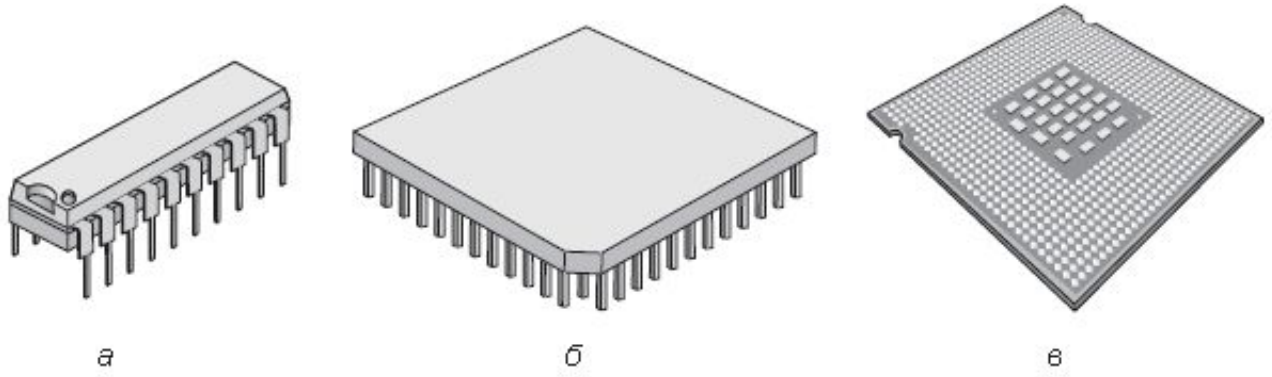


Рис. 3.9. Основные типы корпусов интегральных микросхем : DIP (*а*), PGA (*б*) и LGA (*в*)

Комбинаторные схемы

Многие применения цифровой логики требуют наличия схем с несколькими входами и несколькими выходами, в которых выходные сигналы определяются текущими входными сигналами. Такая схема называется **комбинаторной.**

Мультиплексоры

На цифровом логическом уровне **мультиплексор** представляет собой схему с 2^n входами, одним выходом и n линиями управления, которые позволяют выбрать один из входов. Выбранный вход соединяется с выходом.

На рис. 3.10 изображена схема восьмивходового мультиплексора. Три линии управления A, B и C кодируют 3-разрядное число, которое указывает, какая из восьми входных линий должна соединяться с вентилем ИЛИ и, следовательно, с выходом.

Вне зависимости от того, какое значение окажется на линиях управления, семь вентилей И всегда будут выдавать на выходе 0, а оставшийся может выдавать 0 или 1 в зависимости от значения выбранной линии входа. Каждый вентиль И запускается определенной комбинацией сигналов на линиях управления. Схема мультиплексора показана на рис. 3.10.

Противоположностью мультиплексора является **демультиплексор**, который соединяет единственный входной сигнал с одним из 2^n выходов в зависимости от значений сигналов в n линиях управления. Если бинарное значение линий управления равно k, то выбирается выход k.

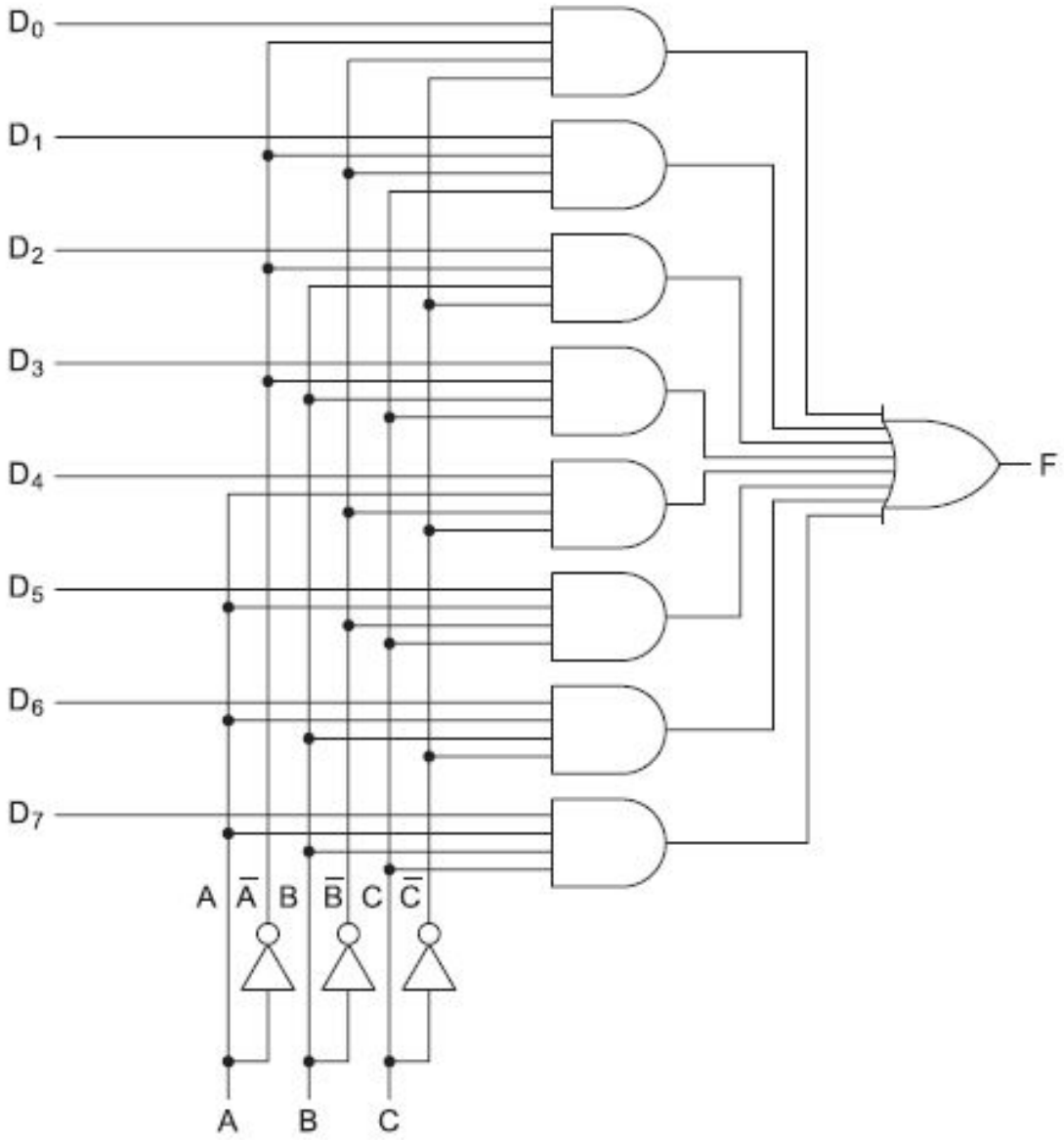


Рис. 3.10. Схема восьмивходового мультиплексора

Декодеры

В качестве второго примера рассмотрим схему, которая получает на входе *n*-разрядное число и использует его для того, чтобы выбрать (то есть установить в значение 1) одну из 2^n выходных линий. Такая схема называется **декодером**. Пример декодера для $n = 3$ показан на рис. 3.12.

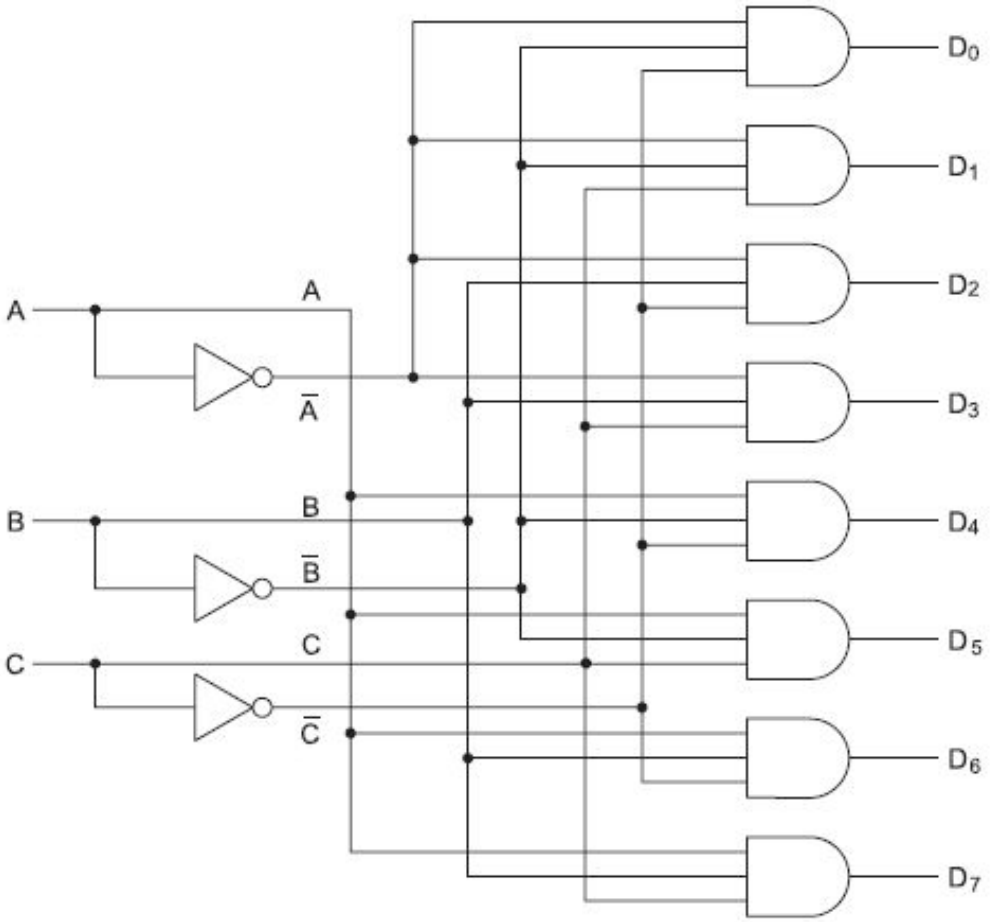


Рис. 3.12. Схема декодера, содержащего 3 входа и 8 выходов

Пример применения декодера

Чтобы понять, зачем нужен декодер, представим себе память, состоящую из восьми микросхем, каждая из которых содержит 256 Мбайт. Микросхема 0 имеет адреса от 0 до 256 Мбайт, микросхема 1 — адреса от 256 до 512 Мбайт и т. д.

Три старших двоичных разряда адреса используются для выбора одной из восьми микросхем. На рис. 3.12 эти три бита — три входа *A*, *B* и *C*.

В зависимости от входных сигналов ровно одна из восьми выходных линий (*D*0, ..., *D*7) принимает значение 1; остальные линии принимают значение 0. Каждая выходная линия активизирует одну из восьми микросхем памяти. Поскольку только одна линия принимает значение 1, активизируется только одна микросхема.

Компараторы

Еще одна полезная схема — ***компаратор***. Компаратор сравнивает два слова, которые поступают на вход. Компаратор, изображенный на рис. 3.13, принимает два входных сигнала *A* и *B* по 4 бита каждый и выдает 1, если они равны, и 0, если они не равны

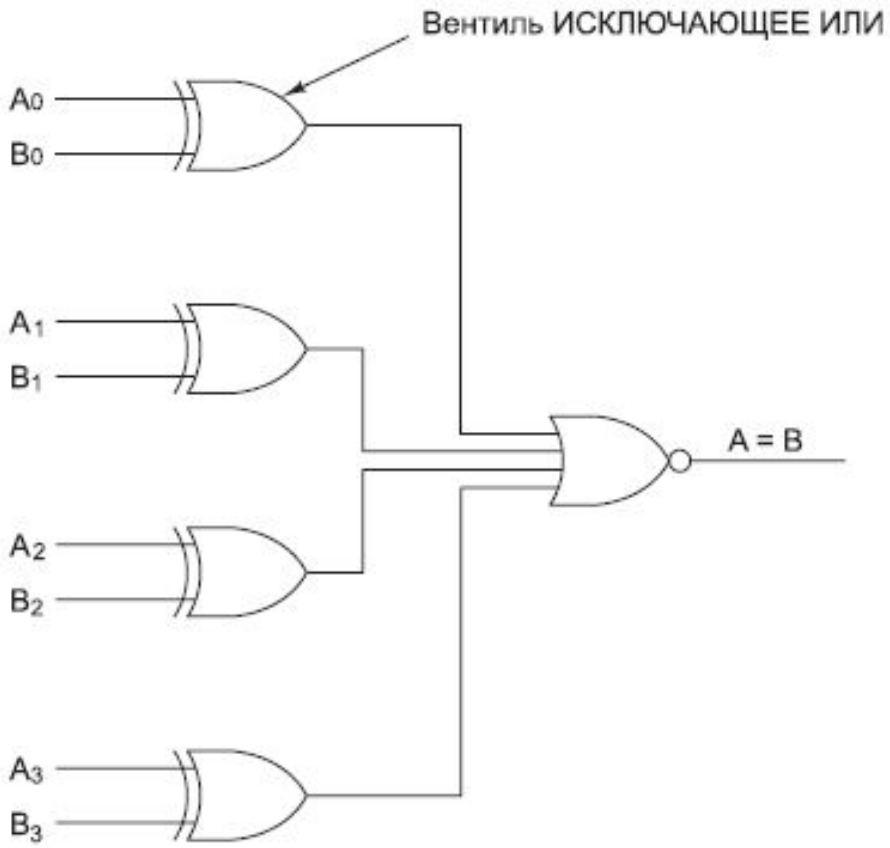


Рис. 3.13. Простой 4-разрядный компаратор

Арифметические схемы

Схемы, используемые для выполнения арифметических операций не является функцией состояния только входных сигналов

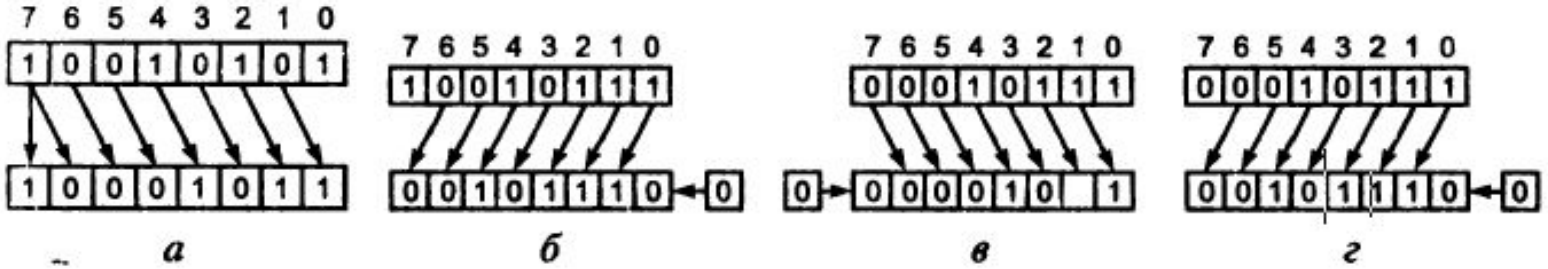
Схемы сдвига

Битовые сдвиги (bit shifts). Эти операции также осуществляются на битовом уровне и являются унарными.

В этих операциях биты сдвигаются в регистре (влево или вправо). Освобождающиеся позиции должны чем-то заполняться.

Виды сдвигов: арифметический, логический и циклический.

Арифметический. Выдвинутые биты аннулируются. При сдвиге влево с правой стороны вдвигаются нули, а при сдвиге вправо знаковый разряд остается в крайнем правом разряде, сохраняя знак операнда (рис. 1.9, а, б).



Арифметический сдвиг влево на *n* разрядов эквивалентен умножению на 2^n (если не происходит переполнения), в то время как арифметический сдвиг вправо на *n* эквивалентен делению на 2^n.

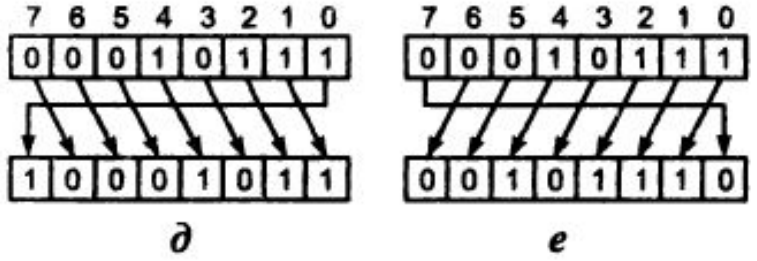
Логический. Выдвинутые биты аннулируются и их место занимают нули.

Логический сдвиг является более подходящим для двоичных чисел без знака, а арифметический сдвиг — для двоичных со знаком.

Схемы сдвига

Циклический (разрядное вращение)**.** Биты, «выдвигаемые» из регистра в любую сторону, «вдвигаются» в регистр с другой его стороны**.**

Эта операция используется, если необходимо сохранить все существующие биты, и часто применяется в цифровой криптографии.



Пример. Выходные данные, которые представляют собой входные данные, сдвинутые на один бит, поступают на линии S_0, ..., S_7. Линия управления C определяет направление сдвига: 0 — влево, 1 — вправо. При сдвиге влево в бит 7 вставляется Аналогичным образом при сдвиге вправо в бит 0 вставляется значение 1.

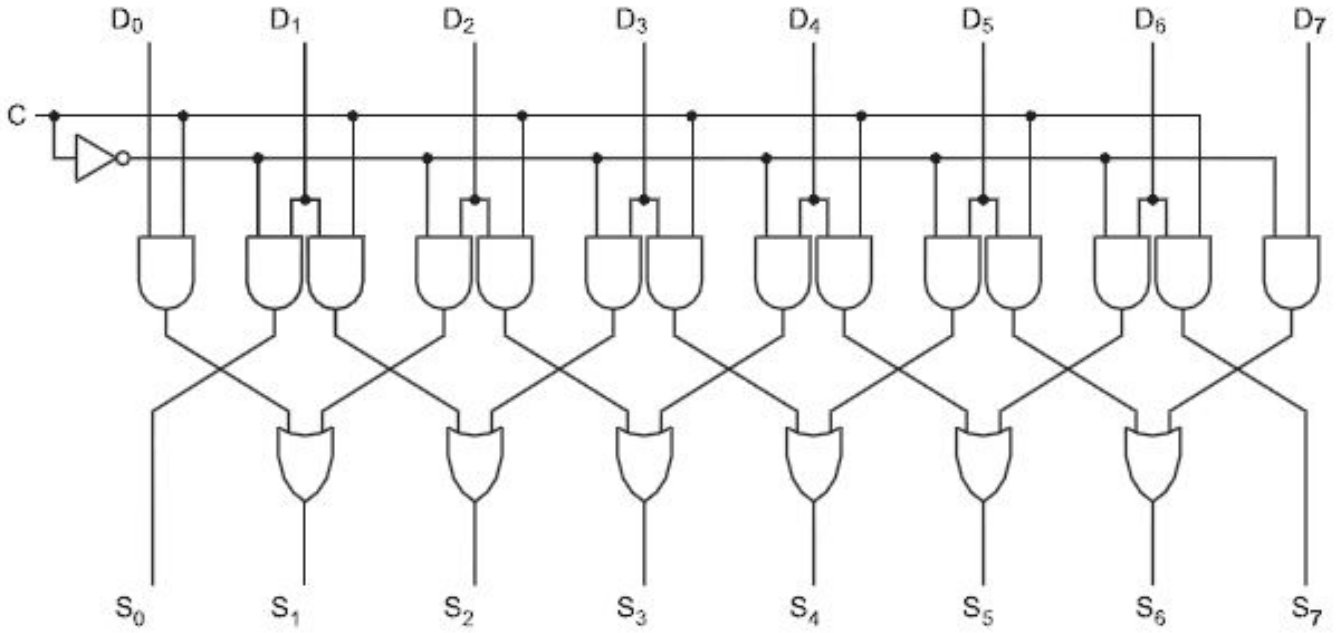


Рис. 3.14. Схема сдвига

Если $C = 1$, правый член каждой пары включается, пропуская через себя соответствующий бит. Так как правый вентиль И соединен с входом вентиля ИЛИ, который расположен справа от этого вентиля И, происходит сдвиг вправо. Если $C = 0$, включается левый вентиль И из пары, и тогда происходит сдвиг влево.

Сумматоры. Полусумматор

Схема выполнения операций сложения является существенной частью любого процессора. Таблица истинности для сложения одноразрядных целых чисел показана на рис. 3.15, *а*. Здесь имеется два результата: сумма входных переменных *A* и *B* и перенос на следующую (левую) позицию.

$1_2 + 1_2 = 10_2$

A	B	Сумма	Перенос
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

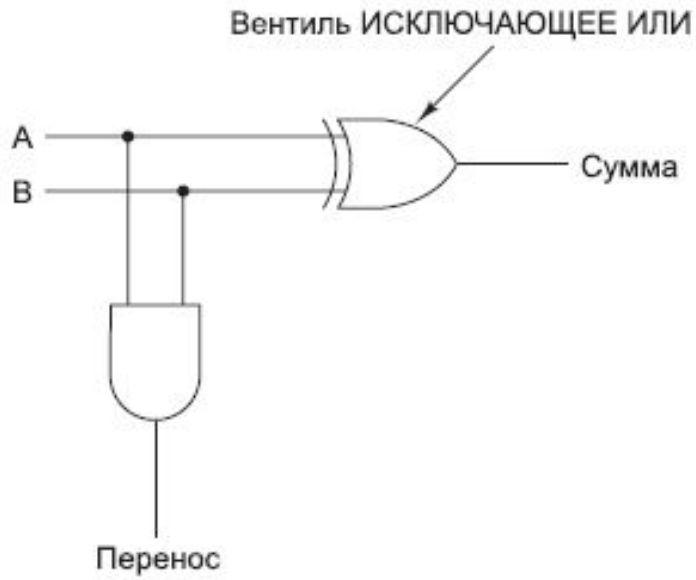


Рис. 3.15. Таблица истинности для сложения одноразрядных чисел (*а*); схема полусумматора (*б*)

Схема для вычисления бита суммы и бита переноса показана на рис. 3.15, *б*. Такая схема обычно называется **полусумматором**.

Полный сумматор

Полусумматор подходит для сложения битов нижних разрядов двух многобитовых слов. Однако он не годится для сложения битов в середине слова, потому что не может осуществлять перенос в эту позицию. Поэтому необходим **полный сумматор** (рис. 3.16).

Из схемы должно быть ясно, что полный сумматор состоит из двух полусумматоров.

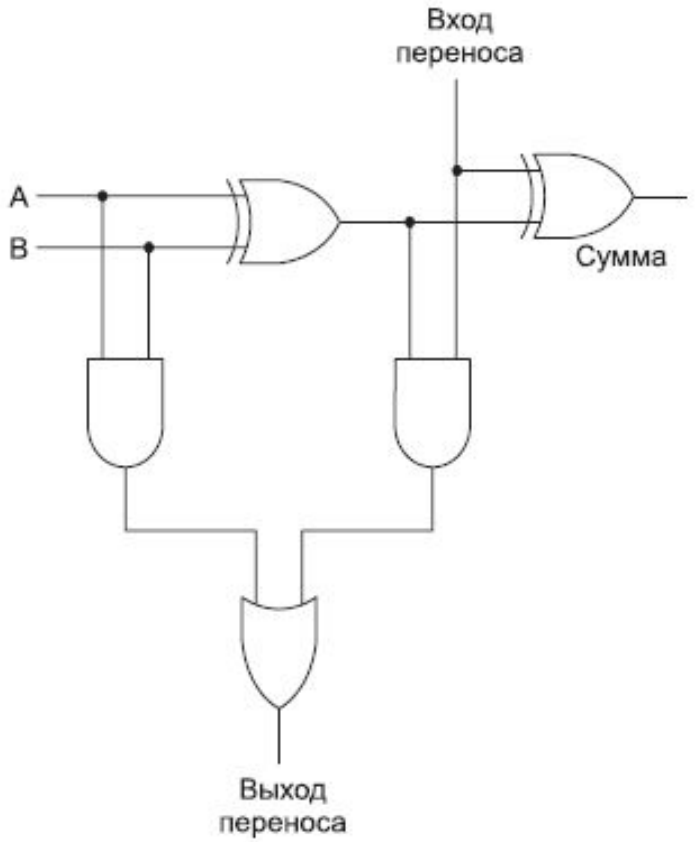
Сумма равна 1, если нечетное число переменных *A*, *B* и *вход переноса* принимает значение 1 (то есть если единице равна или одна из переменных или все три).

Выход переноса принимает значение 1, если либо *A* и *B* одновременно равны 1 (левый вход в вентиль ИЛИ), либо один из них равен 1 и *вход переноса* также равен 1.

Два полусумматора порождают и биты суммы, и биты переноса.

A	B	Вход переноса	Сумма	Выход переноса
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

а



б

Рис. 3.16. Таблица истинности для полного сумматора (*а*); схема для полного сумматора (*б*)

Арифметико-логические устройства

Арифметико-логические устройства

Большинство компьютеров содержат одну схему для выполнения над двумя машинными словами операций И, ИЛИ и сложения. Обычно эта схема для *n*-разрядных слов состоит из *n* идентичных схем — по одной для каждой битовой позиции.

На рис. 3.17 изображена такая схема, которая называется **одноразрядным арифметико-логическим устройством (АЛУ)**. Это устройство может вычислять одну из 4-х следующих функций:

A И *B*, *A* ИЛИ *B*, НЕ *B*, *A* + *B*.

Выбор функции зависит от того, какие сигналы поступают на линии *F*0 и *F*1:

00, 01, 10 или 11 (в двоичной системе счисления).

Отметим, что здесь *A* + *B* означает арифметическую сумму *A* и *B*, а не логическую операцию И.

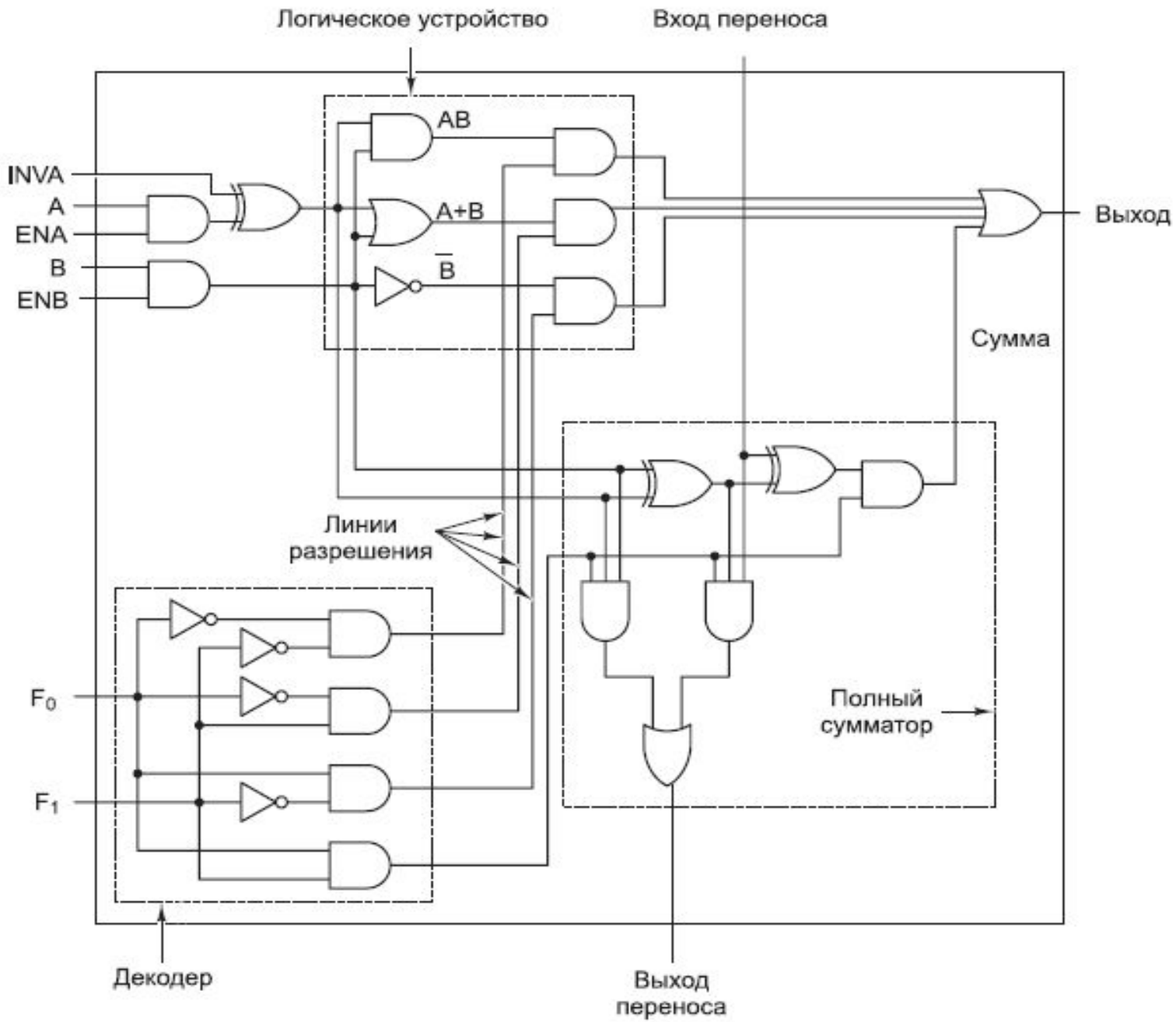


Рис. 3.17. Одноразрядное АЛУ

8-разрядное АЛУ

Одноразрядные схемы называются ***разрядными микропроцессорными секциями***.

Они позволяют разработчику строить АЛУ любой разрядности.

На рис. показана схема 8-разрядного АЛУ, составленного из восьми ***одноразрядных секций***.

Сигнал INC (увеличение на единицу) нужен только для операций сложения. Он дает возможность вычислять такие суммы, как $A + 1$ и $A + B + 1$.

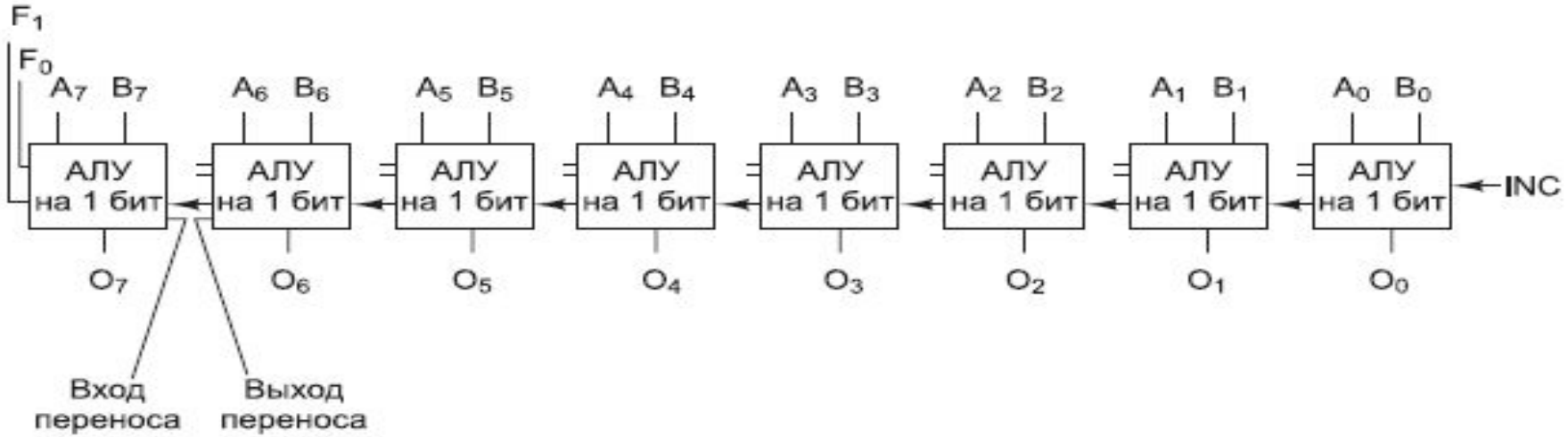


Рис. 3.18. Восемь одноразрядных секций, соединенных в 8-разрядное АЛУ. Для упрощения схемы сигналы разрешения и инверсии не показаны

Тактовые генераторы

Тактовые генераторы

Во многих цифровых схемах важно соблюдать порядок выполнения операций. Иногда одна операция должна предшествовать другой, иногда две операции должны происходить одновременно. Для контроля временных параметров в цифровые схемы встраиваются ***тактовые генераторы***, позволяющие обеспечить синхронизацию.

Тактовый генератор — это схема, которая вызывает серию импульсов. Все импульсы одинаковы по длительности. Интервалы между последовательными импульсами также одинаковы. Временной интервал между началом одного импульса и началом следующего называется **временем такта**. Частота импульсов обычно составляет от 100 МГц до 4 ГГц, что соответствует времени такта от 10 до 250 пс. Частота тактового генератора обычно контролируется высокоточным кварцевым генератором.

В компьютере за один такт может произойти множество событий. Если они должны осуществляться в определенном порядке, то такт следует разделить на подтакты.

Если сделать ответвление от задающей линии тактового генератора и вставить схему с определенным временем задержки, то будет порожден вторичный сигнал, сдвинутый по фазе относительно первичного (рис. 3.19, *а*).

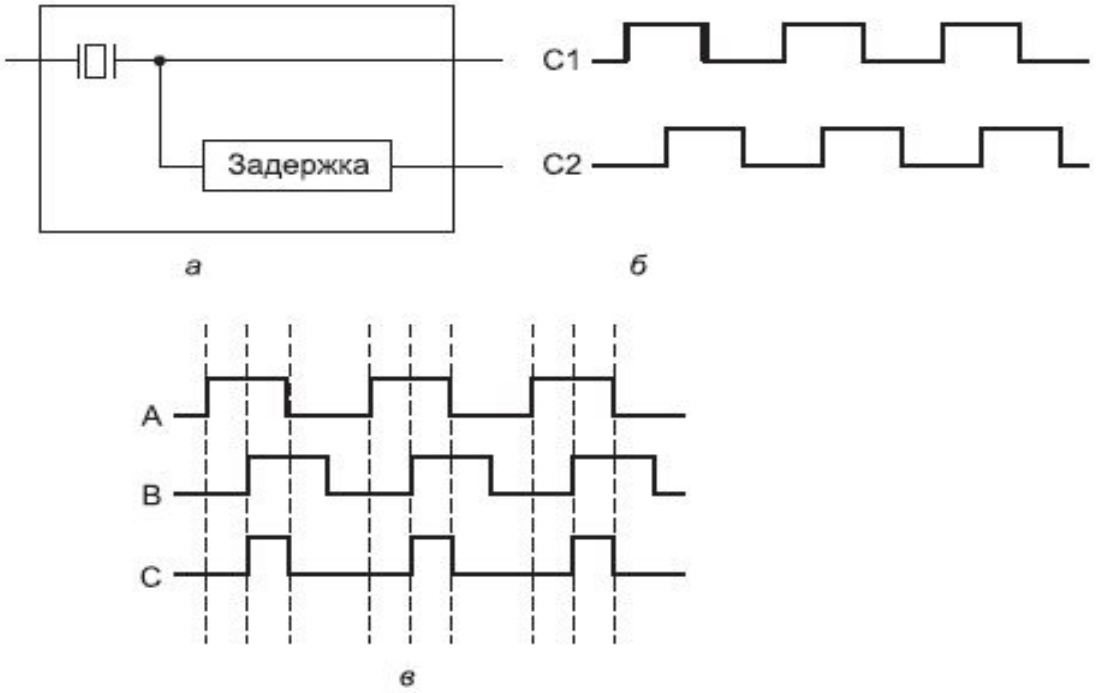


Рис. 3.19. Тактовый генератор (*а*); временна́я диаграмма тактового генератора (*б*); порождение асинхронных тактовых импульсов (*в*)

Временная диаграмма, показанная на рис. 3.19, *б*, предлагает четыре точки начала отсчета времени для дискретных событий:

1. Фронт *C*1.
2. Спад *C*1.
3. Фронт *C*2.
4. Спад *C*2.

Связав различные события с разными перепадами (фронтами и спадами), можно достичь требуемой последовательности выполнения действий.

Память

Память является необходимым компонентом любого компьютера. Без памяти не было бы компьютеров, по крайней мере таких, какие есть сейчас. Память используется для хранения как команд, так и данных. Память строится на таких элементах, как защелки и триггеры.

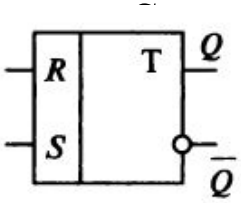
Защелка

Защелка - электронная схема, применяемая в регистрах компьютера для запоминания одного разряда двоичного кода (для создания 1 бита памяти). Защелка имеет два устойчивых состояния, одно из которых соответствует двоичной единице, а другое — двоичному нулю.

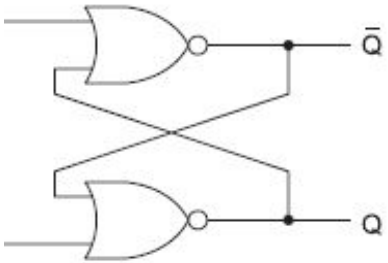
Самый распространенный тип защелки — так называемый **RS**-защелка (*S* и *R*, соответственно, от ***англ.*** set — установка и reset — сброс). Она имеет два симметричных входа *S* и *R* и два симметричных выхода *Q* и $\bar{Q}$.

Выходной сигнал $\bar{Q}$ является логическим отрицанием сигнала *Q*.

Условное обозначение и схема



R



Если *S* принимает значение 1, то *Q* принимает значение 1 независимо от предыдущего состояния защелки.

Сходным образом переход *R* в значение 1 вызывает $Q = 0$.

Схема «запоминает», какой сигнал был последним: *S* или *R*. Используя это свойство, мы можем строить компьютерную память.

Синхронные SR-защелки

Синхронные SR-защелки позволяют изменять состояние защелки только в определенные моменты (рис 3.21).

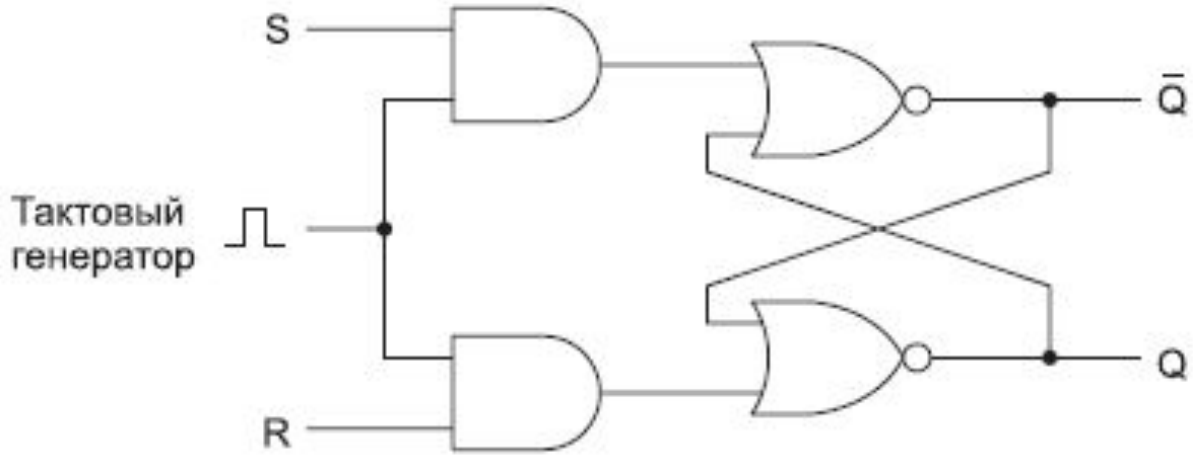


Рис. 3.21. Синхронная SR-защелка

Эта схема имеет дополнительный синхронизирующий вход, который по большей части равен 0. При этом выходы обоих вентилей И равны 0, и независимо от значений *S* и *R* защелка не меняет свое состояние.

Когда значение синхронизирующего входа равно 1, действие вентилей И прекращается, и состояние защелки становится зависимым от *S* и *R*.

Синхронные D-защелки

Синхронные D-защелки - защелки с одним входом

Сигнал R формируется как инверсия сигнала S.

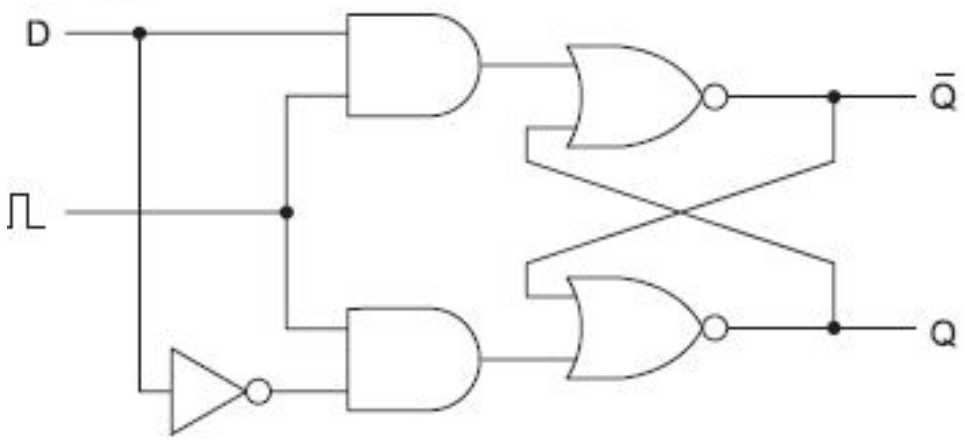


Рис. 3.22. Синхронная D-защелка

Такая схема требует 11 транзисторов.

Схема остается в устойчивом состоянии до тех пор, пока на нее подается питание (на рисунке не обозначено).

Триггеры

В такой схеме, которая называется **триггером** (flip-flop), смена состояния происходит при переходе синхронизирующего сигнала с 0 на 1 (фронт) или с 1 на 0 (спад).

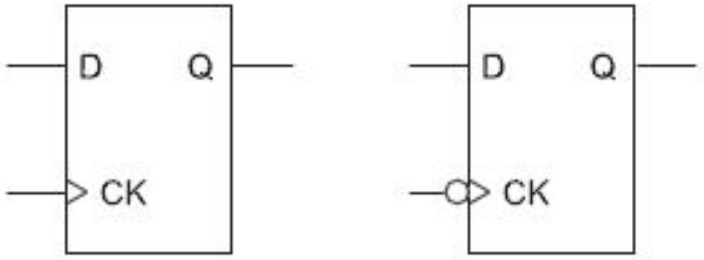
Т. е. длина синхронизирующего импульса не имеет значения, поскольку переходы происходят быстро.

Отличие между триггером и защелкой.

Триггер запускается перепадом сигнала, а защелка запускается уровнем сигнала.

Многие (хотя не все) защелки и триггеры также имеют выход $\bar{Q}$.

Обозначения на схемах: защелка (слева) и триггер (справа)



Регистры

Регистр — узел ЭВМ, предназначенный для хранения двоичных слов и выполнения над ними некоторых логических операций.

Регистр представляет собой совокупность ***триггеров*** по числу разрядов в слове, и вспомогательных схем.

Эти схемы обеспечивают выполнение таких операций, как:

- прием слова;
- выдача слова;
- сдвиг слова влево или вправо на требуемое количество разрядов;
- преобразование последовательного кода в параллельный и наоборот;
- разрядные логические операции.

На рис. 3.26 показано, как восемь триггеров объединяются для формирования 8-разрядного регистра.

Регистры

Регистр получает 8-разрядное входное значение (*I*0 –*I* 7) при изменении синхронизирующего сигнала *CK*, с которым связаны все ***синхронизирующие*** линии. Инвертирующие входы аннулируются инвертором, связанным с *CK*, поэтому триггеры запускаются при переходе от 0 к 1.

Все восемь сигналов ***очистки*** тоже объединены, поэтому когда сигнал сброса *CLR* переходит в состояние 0, все триггеры переходят в состояние 0.

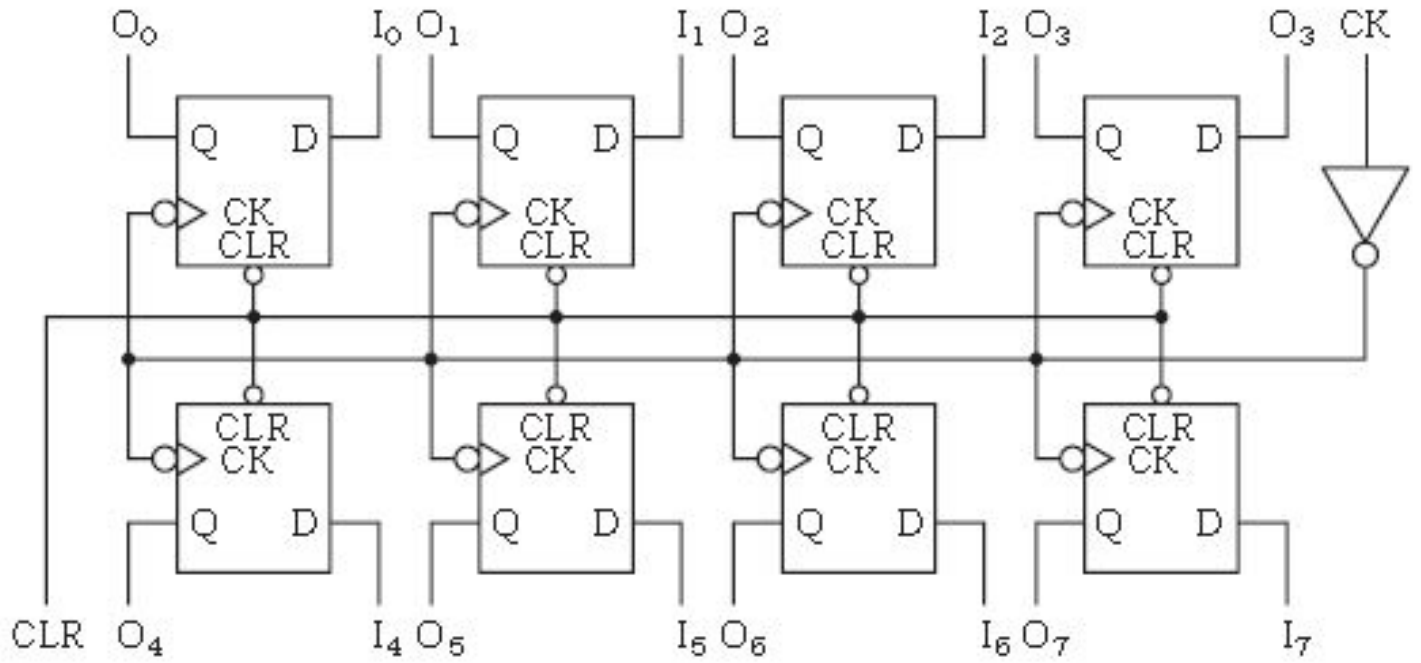


Рис. 3.26. 8-разрядный регистр, построенный из одноразрядных триггеров

8-разрядный регистр может использоваться в качестве структурного элемента для построения регистров большей разрядности.

Регистры

Регистр получает 8-разрядное входное значение (*I*0 –*I* 7) при изменении синхронизирующего сигнала *CK*, с которым связаны все ***синхронизирующие*** линии. Инвертирующие входы аннулируются инвертором, связанным с *CK*, поэтому триггеры запускаются при переходе от 0 к 1.

Все восемь сигналов ***очистки*** тоже объединены, поэтому когда сигнал сброса *CLR* переходит в состояние 0, все триггеры переходят в состояние 0.

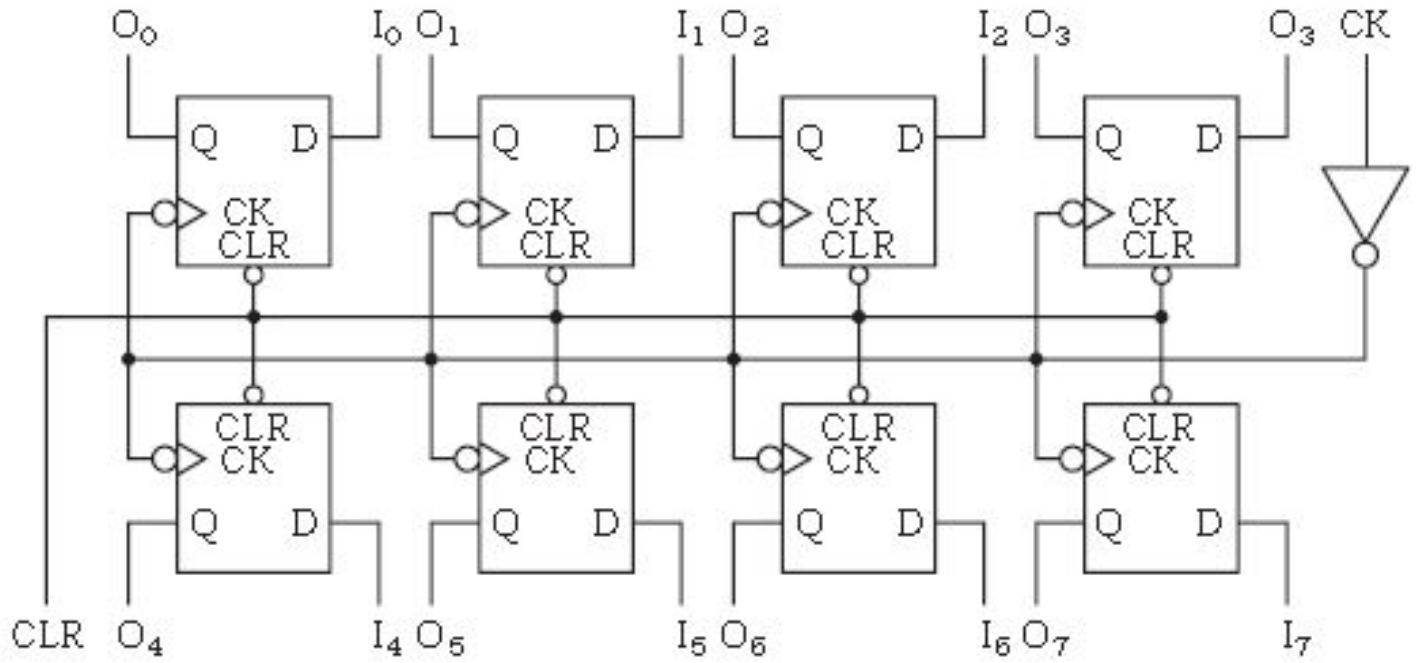


Рис. 3.26. 8-разрядный регистр, построенный из одноразрядных триггеров

8-разрядный регистр может использоваться в качестве структурного элемента для построения регистров большей разрядности.

Виды памяти компьютера

Непосредственно в компьютере используются два вида электронной памяти ОЗУ и ПЗУ.

Память **ОЗУ** (оперативное запоминающее устройство), или **RAM** (Random Access Memory — **оперативная память**) позволяет и записывать, и считывать информацию. Ее назначение хранить программу и данные в процессе выполнения программы.

ПЗУ (постоянное запоминающее устройство), или **ROM** (Read-Only Memory — постоянная память).

Существует два типа ОЗУ: статическое и динамическое.

ОЗУ

Существует два типа ОЗУ: статическое и динамическое.

Статическое ОЗУ (Static RAM, **SRAM**) конструируется с использованием D-триггеров. Информация в ОЗУ сохраняется на протяжении всего времени, пока к нему подается питание: секунды, минуты, часы и даже дни. Статическое ОЗУ работает очень быстро. Обычно время доступа составляет несколько наносекунд. По этой причине статическое ОЗУ часто используется в качестве кэш-памяти второго уровня.

В **динамическом ОЗУ** (Dynamic RAM, **DRAM**), напротив, триггеры не используются. Динамическое ОЗУ представляет собой массив ячеек, каждая из которых содержит транзистор и крошечный конденсатор. Конденсаторы могут быть заряженными и разряженными, что позволяет хранить нули и единицы. Поскольку электрический заряд имеет тенденцию исчезать, каждый бит в динамическом ОЗУ должен **обновляться** (перезаряжаться) каждые несколько миллисекунд, чтобы предотвратить утечку данных.

В современных компьютерах сочетание ***кэшпамяти*** на основе статического ОЗУ и ***основной памяти*** на основе динамического ОЗУ соединяет в себе преимущества обоих устройств.

ПЗУ

Первоначально ПЗУ не позволяли изменять и стирать хранящуюся в них информацию (ни умышленно, ни случайно). Данные записываются в ПЗУ в процессе производства. Для этого изготавливается трафарет с определенным набором битов, который накладывается на фоточувствительный материал, а затем открытые (или закрытые) части поверхности вытравливаются.

Единственный способ изменить программу в ПЗУ — поменять всю микросхему.

Флэш-память

Флэш-память - современный тип электронно перепрограммируемого ПЗУ.

Флэш-память образуется из множества твердотельных ячеек, состоящих из одного специального **флэш-транзистора**.

Флэш-память стирается и записывается блоками.

Они используются для хранения изображений в цифровых камерах и для других целей.

В настоящее время флэш-память начинает вытеснять магнитные диски в качестве внешнего накопителя, учитывая время доступа в 50 нс.

Флэш-память обеспечивает лучшее время доступа при более низком энергопотреблении; с другой стороны, стоимость одного бита флэш-памяти существенно выше, чем у дисков.

Резюме

Компьютеры собираются из интегральных схем, содержащих крошечные переключатели, которые называются вентилями. Обычно используются вентили И, ИЛИ, НЕ-И, НЕ-ИЛИ и НЕ. Комбинируя отдельные вентили, можно строить простые схемы.

Более сложными схемами являются мультиплексоры, демультиплексоры, кодеры, декодеры, схемы сдвига и АЛУ. С помощью программируемой вентильной матрицы (FPGA) можно запрограммировать произвольные булевы функции. Если требуется много булевых функций, программируемые логические матрицы обычно более эффективны, чем другие средства. Для преобразования схем из одной формы в другую используются законы булевой алгебры. Во многих случаях это позволяет создать более экономичные схемы.

Арифметические действия в компьютерах осуществляются сумматорами. Одноразрядный полный сумматор можно сконструировать из двух полусумматоров. Чтобы построить сумматор для многоразрядных слов, полные сумматоры соединяются таким образом, чтобы выходной сигнал переноса каждого сумматора передавался его левому соседу.

Резюме

Статическая память состоит из защелок и триггеров, каждый из которых может хранить один бит информации. Их можно объединять, получая восьмиразрядные триггеры и защелки или готовую память для хранения слов. Существуют различные типы памяти: ОЗУ, ПЗУ, флэш-память.

Статическое ОЗУ не нужно обновлять: оно хранит информацию, пока включен компьютер. Динамическое ОЗУ, напротив, нужно периодически обновлять, чтобы предотвратить потерю информации.

Контрольные вопросы

1. Структурные единицы ЭВМ
2. Понятие цифрового сигнала и цифровой схемы
3. Понятие вентиля
4. Понятие булевой (логической) переменной и булевой функции
5. Способы задания булевой функции
6. Правило построения электронной схемы булевой функции:
7. Понятие мультиплексора, декодера, компаратора
8. Схемы полусумматора и полного сумматора
9. Операции, выполняемые алу
10. Понятие тактового генератора
11. Назначение защелки и триггера
12. Различие между защелкой и триггером
13. Понятие регистра, как он реализуется
14. Виды электронной памяти
15. Виды ОЗУ и их реализация

Папка для Колледжа
https://drive.google.com/open?id=1zI21f1EIKK908VDIxMzVjI63O6xKK_P1