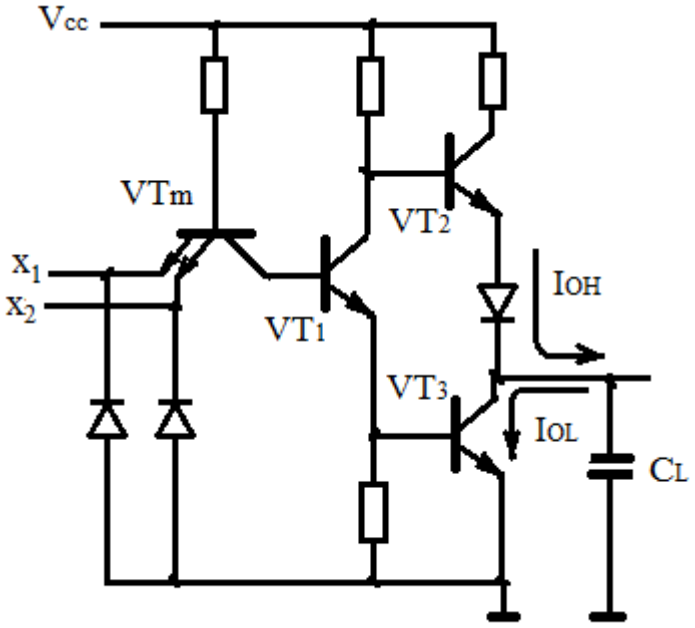
Лекция 1.
Вводная часть.

Предмет курса. Изучение основных устройств вычислительной техники, базовых элементов для построения схем, физических основ их построения и способов математического описания.

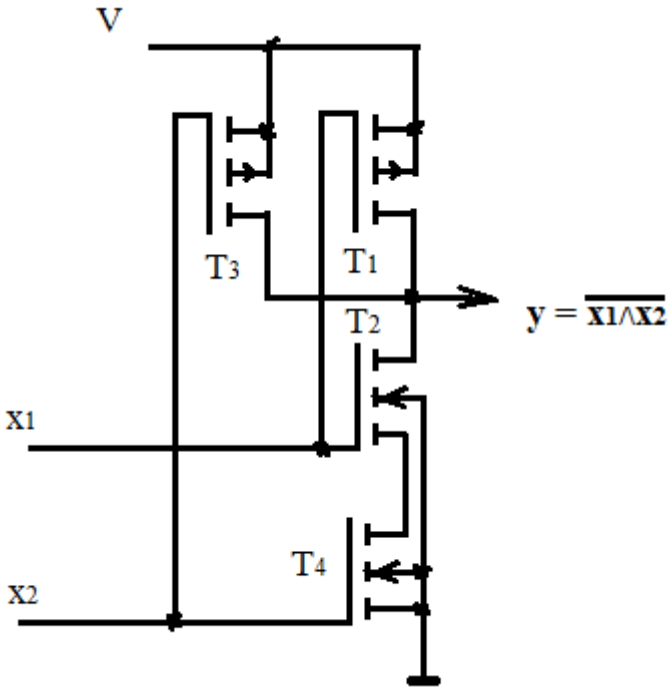
Различают аналоговые и цифровые схемы. В вычислительной технике используются, в основном, цифровые схемы. Назначение любой схемы – передача или преобразование сигнала. Цифровой сигнал имеет два уровня – уровень логической «1» и уровень логического «0». Таким образом, физическая основа построения цифровых устройств - p-n переход, а математический аппарат их описания – алгебра логики.

Алгебра логики предполагает использование следующих базовых логических функций: логического умножения (конъюнкции) – И, логического сложения (дизъюнкции) – ИЛИ и отрицания (инверсии) – НЕ. Кроме того, для логического сложения двух компонентов, кроме безальтернативного, обычного логического сложения, существует альтернативное ИЛИ (сложение по модулю 2).

Базовыми логическими элементами, содержащими наименьшее количество переходов, являются элементы И-НЕ или ИЛИ-НЕ. На рис.В.1(а,б) представлены элементы И-НЕ, построенные на основе ТТЛ – технологии и КМОП – технологии. В случае ТТЛ используются биполярные транзисторы, где основа функционирования – токи, а в случае КМОП используются полевые транзисторы, где основа функционирования – напряжение.



а)



б)

Рис. В.1(а – ТТЛ-элемент 2И-НЕ; б – КМОП-элемент 2И-НЕ)

ТТЛ-элемент, рис.В.1(а), построен на основе транзисторов типа n-p-n. Подача высоких уровней (1) на x_1 и x_2 приводит к отсутствию токов эмиттеров многоэмиттерного транзистора VT_m. Ток коллекторного перехода этого транзистора проходит эмиттерный повторитель VT_1 и создает потенциал на базе VT_3, необходимый для его насыщения. Нагрузка разряжается через открытый VT_3, на выходе снимаем низкий уровень (0). Подача низкого уровня хотя бы на один вход элемента приводит к насыщению VT_2 и заряду емкости нагрузки до состояния 1.

В КМОП-элементе, рис.В.1(б), подача высоких уровней на оба входа приводит к открытому состоянию основных транзисторов T_2 и T_4 и запиранию комплементарных - T_1 и T_3. На выходе имеем низкий уровень. Появление низкого уровня на одном из входов повлечет запирание соответствующего основного транзистора. На выходе будет фиксироваться высокий уровень.

В современных условиях любые устройства строятся на основе больших интегральных схем (БИС). Такие схемы могут иметь установленные изготовителем внутренние связи узлов и элементов, функционирующие согласно записанным программным кодам (микропроцессоры или микроконтроллеры), а также это могут быть схемы, где связи прописываются пользователем под конструируемую им схему. Такие БИС носят название программируемые логические интегральные схемы (ПЛИС).

Как было отмечено ранее, все схемы обрабатывают сигналы, представляющие информацию. При этом информацию можно хранить, передавать по направлениям, или преобразовывать. Хранение информации возможно только в схемах, содержащих в своей структуре перекрестные обратные связи. Преобразование информации возможно только в схемах с линейной структурой. Все коммутационные схемы построены на основе линейной структуры.

Таким образом, все цифровые устройства разделяются на два основных типа: комбинационные цифровые устройства (КЦУ) и конечные автоматы (или ПЦУ - последовательностные цифровые устройства). КЦУ не содержат в своей структуре обратных связей, поэтому в каждый момент времени состояние выходов таких устройств зависит только от поданной на входы комбинации. Конечные автоматы содержат в своей структуре обратные связи, таким образом, в каждый момент времени состояние выходов таких устройств зависит не только от входных воздействий, но и от состояний выходов в предыдущий момент времени.

КЦУ могут быть кодопреобразующие и коммутационные. Кодопреобразующие КЦУ имеют входы только одного типа. В каждый момент времени состояние выходов устройства зависит только от входной комбинации. КЦУ данного типа это все виды **кодопреобразователей, сумматоры, шифраторы** и **дешифраторы**. Коммутационные КЦУ имеют два типа входов: адресные входы и входы данных. От состояния адресных входов зависит, которая из линий данных будет соединяться с единственно возможной. Устройство с единственным выходом данных и множеством входов называется **мультиплексор**, устройство с единственным входом данных и множеством выходов называется **демультиплексор.**

Конечные автоматы служат для хранения информации или ее преобразования с учетом предыдущего состояния. Простейшая ячейка для хранения информации – **триггер**. Устройства, построенные на основе триггеров: **регистры, счетчики**, конечные автоматы с произвольной сменой комбинаций, и т.п. относятся к типу конечных автоматов.

Синтез любого цифрового устройства производится согласно техническому заданию путем записи таблицы функционирования и, если это необходимо, составления по таблице системы логических уравнений в канонической форме.

Комбинационные цифровые устройства.

Кодопреобразующие КЦУ.

Компаратор.

Компаратор служит для сравнения двух N-разрядных чисел, поступающих на его входы. Наиболее простым является компаратор, определяющий равенство двух чисел. Для этого необходимо лишь выявить поразрядное равенство этих чисел. Рассмотрим ситуацию для 2-разрядных чисел «a» и «b». Выход обозначим как «z». В случае равенства z = 1, в случае неравенства z = 0. Запишем таблицу только для случаев z = 1.

№	a_1	a_0	b_1	b_0	z
1	0	0	0	0	1
2	0	1	0	1	1
3	1	0	1	0	1
4	1	1	1	1	1

Из таблицы получим:

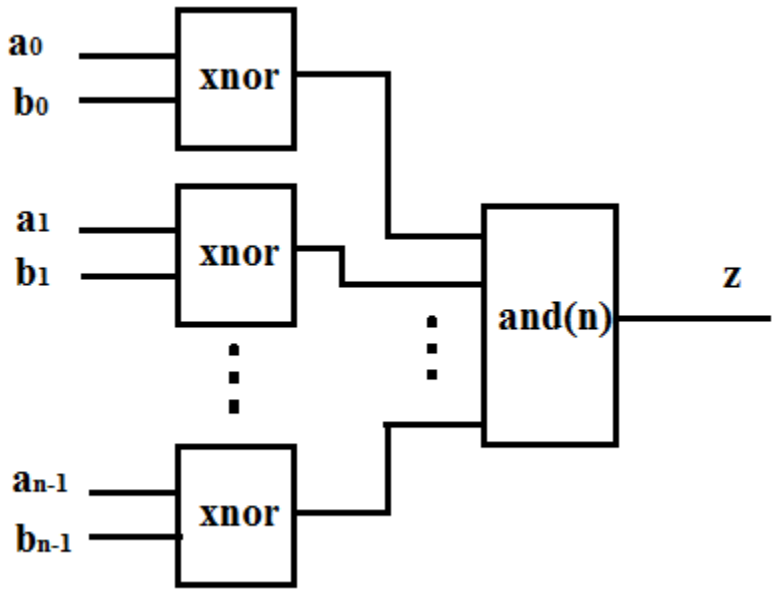
$$z = \bar{a}_1 \cdot \bar{b}_1 \cdot (\bar{a}_0 \cdot \bar{b}_0 \vee a_0 \cdot b_0) \vee a_1 \cdot b_1 \cdot (\bar{a}_0 \cdot \bar{b}_0 \vee a_0 \cdot b_0) = \overline{(a_1 \oplus b_1)} \cdot \overline{(a_0 \oplus b_0)}$$

Если заменить знак инверсии буквой «n», конъюнкцию переписать как «&», а равнозначность определить командой «xnor», то выражение примет вид:

z = **na_1&nb_1&(na_0&nb_0 \/ a_0&b_0)** \/ **a_1&b_1&(na_0&nb_0 \/ a_0&b_0)** =

(a_1 xnor **b_1)&(a_0** xnor **b_0)**

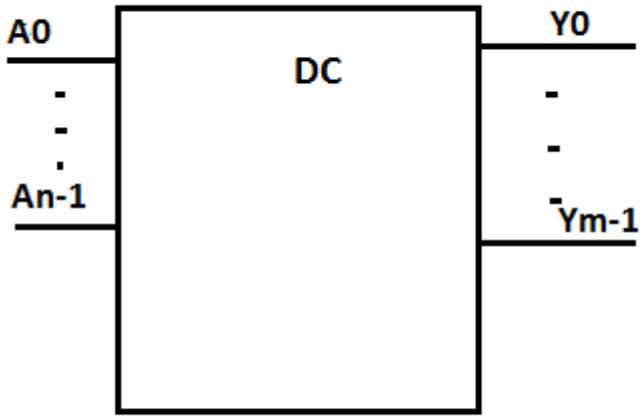
Для сравнения n-разрядных чисел схема будет выглядеть следующим образом



Дешифратор.

Дешифратор является одним из основных элементов вычислительных схем. Его входы подключаются к адресной шине обслуживаемого дешифратором блока, а на выходах формируются сигналы управления, позволяющие активировать в каждый момент времени один из элементов блока. **Дешифратор выдает управление по указанному на шине адресу.**

Таким образом, дешифратор позволяет в каждый момент времени преобразовывать полноразрядный двоичный код на N входах в унитарный двоичный код на M выходах. (Активен только тот выход, адрес которого в данный момент указан на адресных входах). Соотношения $M=2^N$ для полного дешифратора и $M<2^N$ для неполного дешифратора.



В качестве примера рассмотрим дешифратор на 3 адресных входа. Запишем таблицу функционирования

A2	A1	A0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	1 v
0	0	1	0	0	0	0	0	0	1 v	0
0	1	0	0	0	0	0	0	1 v	0	0
0	1	1	0	0	0	0	1 v	0	0	0
1	0	0	0	0	0	1 v	0	0	0	0
1	0	1	0	0	1 v	0	0	0	0	0
1	1	0	0	1 v	0	0	0	0	0	0
1	1	1	1 v	0	0	0	0	0	0	0

Система уравнений будет выглядеть следующим образом:

Y0 = nA0 & nA1 & nA2;

Y1 = A0 & nA1 & nA2;

Y2 = nA0 & A1 & nA2;

Y3 = A0 & A1 & nA2;

Y4 = nA0 & nA1 & A2;

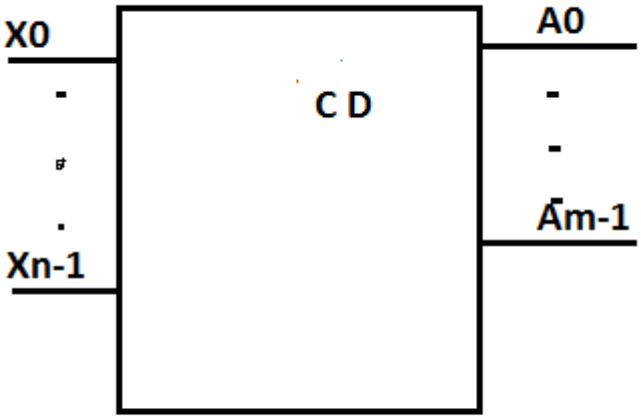
Y5 = A0 & nA1 & A2;

Y6 = nA0 & A1 & A2;

Y7 = A0 & A1 & A2;

Шифратор.

Шифратор, это устройство, определяющее адрес направления, по которому поступил запрос. В каждый момент времени в таком устройстве активный сигнал может быть только на одном входе. В противном случае состояние выходов не определено. Для шифратора с количеством входов направлений N и количеством адресных выходов M соотношение $M = \log_2 N$



Пример для шифратора на 4 входа.

Таблица функционирования

X3	X2	X1	X0	A1	A0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

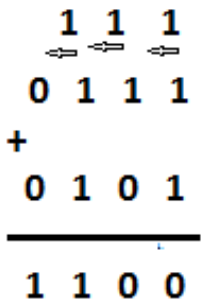
Система уравнений:

A0 = nX3 & nX2 & X1 & nX0 \/ X3 & nX2 & nX1 & nX0;

A1 = nX3 & X2 & nX1 & nX0 \/ X3 & nX2 & nX1 & nX0;

Сумматор

Сумматор – комбинационное цифровое устройство, предназначенное для получения арифметической суммы двух чисел, представленных в двоичном коде. Для примера рассмотрим сумму чисел 7 + 5. Суммируем поразрядно



Нулевой разряд. Складываем арифметически 1+1=2. Для двоичной системы это означает перенос «1» с весом 2^1 в первый разряд – $2=2^1$. Сумма для нулевого разряда принимает значение «0».

Первый разряд. Складываем разряды чисел, подаваемые на внешние входы – 1+0=1, а затем прибавляем «1» с входа переноса – 1+1=2. Не забываем, что для первого разряда вес каждой «1» это 2^1, т.е. 2. Полученный нами результат – $2x2^1$ дает перенос «1» во второй разряд и оставляет значение суммы «0». Вес переносимой «1» – 2^2.

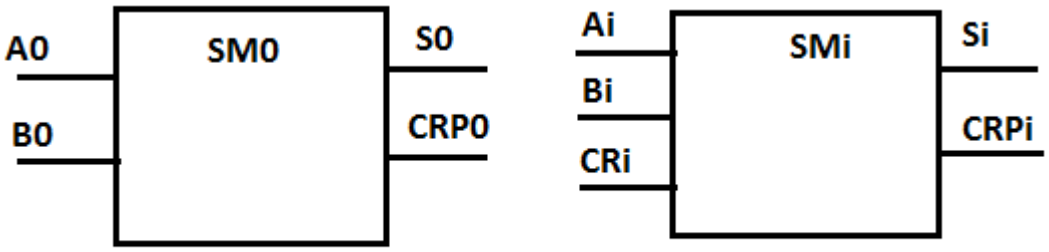
Второй разряд. Опять складываем сначала значения, поступившие с внешних разрядных входов 1+1=2. (Помним, что вес каждой 1 здесь – 2^2, т.е. результат уже 2^3). Результат, как и в прошлых сложениях, оставляет в сумме «0» и дает перенос «1» в третий разряд. Но здесь мы опять должны учесть перенос, пришедший из предыдущего разряда. Суммируем 0+1=1. Таким образом, в сумме для второго разряда остается «1».

Получаем конечный результат - 1100, что соответствует известному нам десятичному числу 12.

Замечаем, что все разрядные сумматоры, кроме нулевого разряда, имеют дополнительный вход переноса.

Итак, для реализации N-рядного сумматора необходим один модуль полу-сумматора, реализующий 0-й разряд (2 входа и 2 выхода), и N-1 модуль

полного одноразрядного сумматора (3 входа и 2 выхода).



Запишем таблицу функционирования для модуля i-го разряда.

CRi	Ai	Bi	Si	CRPi
0	0	0	0	0
0	0	1	1 v	0
0	1	0	1 v	0
0	1	1	0	1 v
1	0	0	1 v	0
1	0	1	0	1 v
1	1	0	0	1 v
1	1	1	1 v	1 v

Теперь запишем уравнения для выходов Si и CRPi, отмечая инверсные значения буквой n. Для каждого выхода рассматриваем все случаи обращения функции в 1.

Для суммы - Si разделим отклики 1 в таблице на две части: в случае Cri = 0 -(nCRi) и в случае Cri = 1. В первом случае функция обращается в 1, если Ai и Bi имеют разные знаки (((nAi) & Bi) \/ (Ai & (nBi))), а во втором случае она обращается в 1, если Ai и Bi имеют одинаковые знаки (((nAi) & (nBi)) \/ (Ai & Bi)).

Функция неравнозначности в алгебре логики имеет обозначение xor(exclusive or), а равнозначность, соответственно, обозначается xnor. Получаем уравнение для Si:

Si=nCRi & (((nAi) & Bi) \/ (Ai & nBi)) \/ CRi & (((nAi) & nBi) \/ (Ai & Bi)) =

(nCRi & (Ai xor Bi)) \/ (CRi & (Ai xnor Bi)) = (**CRi xor (Ai xor Bi));**

Для выхода переноса – CRPi при записи функции объединяем первое и третье слагаемое, выносим за скобки Ai & Bi, содержимое скобок (nCRi \/ CRi)=1. Получаем уравнение для CRPi:

CRPi = ((nCRi) & Ai & Bi) \/ (CRi & (Ai xor Bi)) \/ (CRi & Ai & Bi) =

(Ai & Bi) \/ (CRi & (Ai xor Bi));

Объединим уравнения в систему

Si = (**CRi xor (Ai xor Bi));**

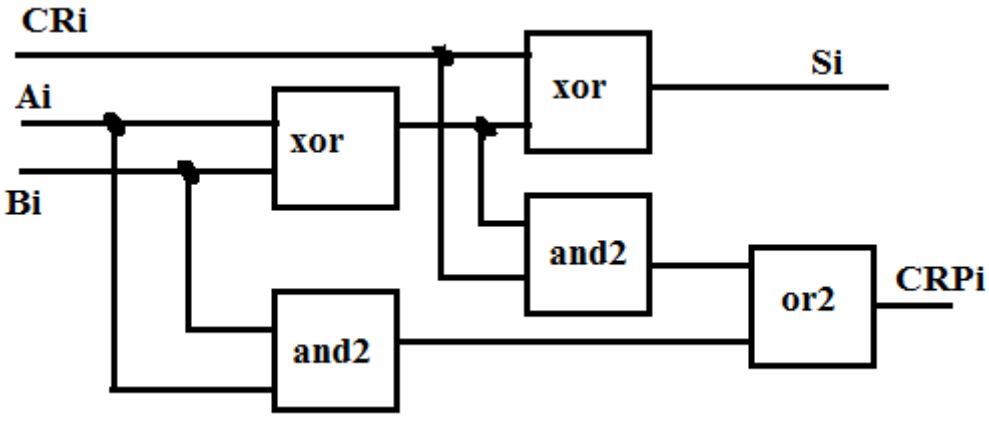
CRPi = **(Ai & Bi) \/ (CRi & (Ai xor Bi));**

Действительно, внутри одного двоичного разряда сумма обращается в единицу или в случае отсутствия «1» в переносе и разных значениях слагаемых, или в случае переноса «1» из предыдущего разряда и одинаковых значениях слагаемых.

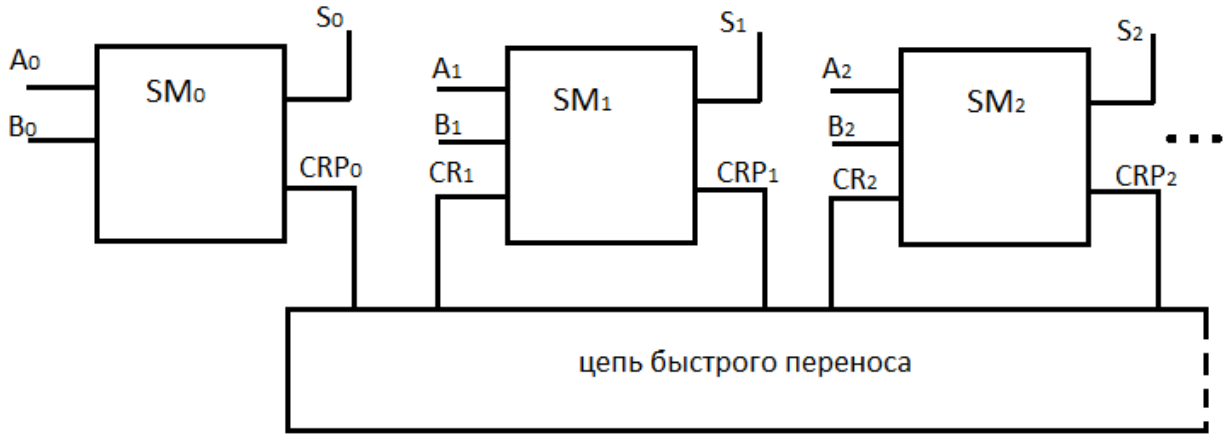
Перенос же «1» в следующий разряд происходит или в случае прихода единиц с входов слагаемых Ai и Bi, или при поступлении переноса из предыдущего разряда, когда хотя бы одно из слагаемых отлично от «0».

Лекция 2

Схема одноразрядного сумматора

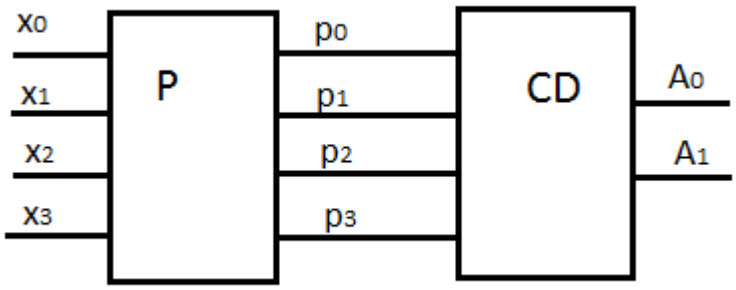


Одноразрядные сумматоры соединяются в многоразрядные путем объединения входов и выходов переноса. Для ускорения процесса используется специальная схема – цепь быстрого переноса. Возможность построения схемы быстрого переноса основана на том, что перенос используется в схеме одноразрядного сумматора на втором этапе работы.



Итак, отмечаем, что все кодопреобразующие КЦУ синтезируются по одинаковому принципу. По основному определению устройства (заданию на построение) записывается таблица функционирования, т.е. на каждую комбинацию на входах записывается требуемый отклик на выходах, а затем производится запись логических уравнений для каждого выхода в отдельности.

В качестве примера синтеза кодопреобразователя вернемся к рассмотренной ранее схеме шифратора. В реальных устройствах запросы могут поступать одновременно на несколько входов, но необходимо выбрать всегда один. Для выделения только одного запроса в каждый момент времени на входе шифратора можно поставить схему определения приоритета. Представим ее, как простейший кодопреобразователь. Исходим из условия, что наивысший приоритет имеет вход x_0, а на самый низкий приоритет у входа x_3.



Запишем таблицу истинности для P

X_3	X_2	X_1	X_0	P_3	P_2	P_1	P_0
0	0	0	0				
0	0	0	1				1
0	0	1	0			1	
0	0	1	1				1
0	1	0	0		1		
0	1	0	1				1
0	1	1	0			1	
0	1	1	1				1
1	0	0	0	1			
1	0	0	1				1
1	0	1	0			1	
1	0	1	1				1
1	1	0	0		1		
1	1	0	1				1
1	1	1	0			1	
1	1	1	1				1

Из таблицы можно записать

$P_0 = x_{0;}$

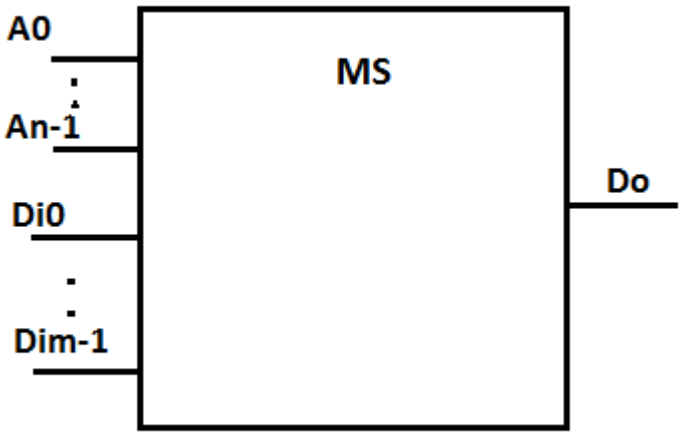
$P_1 = x_1\&(nx_0);$

$P_2 = x_2\&(nx_1)\&(nx_0);$

$P_3 = x_3\&(nx_2)\&(nx_1)\&(nx_0);$

КЦУ коммутационного типа. Мультиплексор.

Мультиплексор – это устройство, коммутирующее на единственный выход тот из входов данных, адрес которого указан на адресных входах. Устройство собирает информацию с разных входов на одну линию. В устройствах коммутационного типа на адресные входы в каждый момент времени информация поступает параллельно, а по входам данных следует последовательно, входы данных в этом случае не представляют собой шину.Так как в описании устройства в виде таблицы будут фигурировать два типа входов, и на каждую комбинацию адресов необходимо рассматривать два состояния соответствующего входа данных: «0» и «1». Функцию выхода записываем при Do=1 для каждого адреса.



Для n адресных входов и m входов данных соотношение $m=2^n$ ($m<2^n$).

Таблица функционирования для 2-х адресных входов

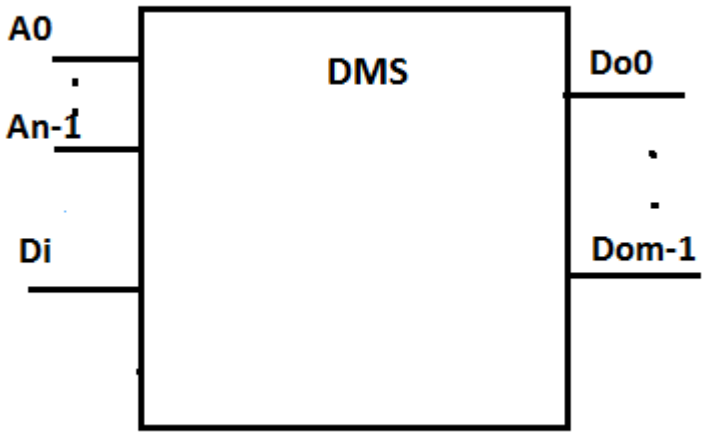
A1	A0	Di3	Di2	Di1	Di0	Do
0	0	-	-	-	0/1	0/1
0	1	-	-	0/1	-	0/1
1	0	-	0/1	-	-	0/1
1	1	0/1	-	-	-	0/1

Функция выхода:

Do = nA1 & nA0 & Di0 \/ nA1 & A0 & Di1 \/ A1 & nA0 & Di2 \/ A1 & A0 & Di3;

Демультиплексор.

Демультиплексор - это устройство, коммутирующее единственный вход данных с тем из выходов данных, адрес которого указан на адресных входах. Устройство распределяет информацию с одной линии на разные направления. Таким образом, на каждую адресную комбинацию учитывается состояние «0» или «1» единственного входа данных, при этом активным может оказаться только один выход, адрес которого указан на адресных входах, и информация на этом выходе должна совпадать с информацией на единственном входе.



Соотношение между m и n аналогично соотношению для мультиплексора.

Таблица функционирования демультиплексора на 3 адресных входа.

A2	A1	A0	Di	Do7	Do6	Do5	Do4	Do3	Do2	Do1	Do0
0	0	0	0/1	-	-	-	-	-	-	-	0/1
0	0	1	0/1	-	-	-	-	-	-	0/1	-
0	1	0	0/1	-	-	-	-	-	0/1	-	-
0	1	1	0/1	-	-	-	-	0/1	-	-	-
1	0	0	0/1	-	-	-	0/1	-	-	-	-
1	0	1	0/1	-	-	0/1	-	-	-	-	-
1	1	0	0/1	-	0/1	-	-	-	-	-	-
1	1	1	0/1	0/1	-	-	-	-	-	-	-

При этом система выходных функций выглядит следующим образом:

Do0 = nA2 & nA1 & nA0 & Di;

Do1 = nA2 & nA1 & A0 & Di;

Do2 = nA2 & A1 & nA0 & Di;

Do3 = nA2 & A1 & A0 & Di;

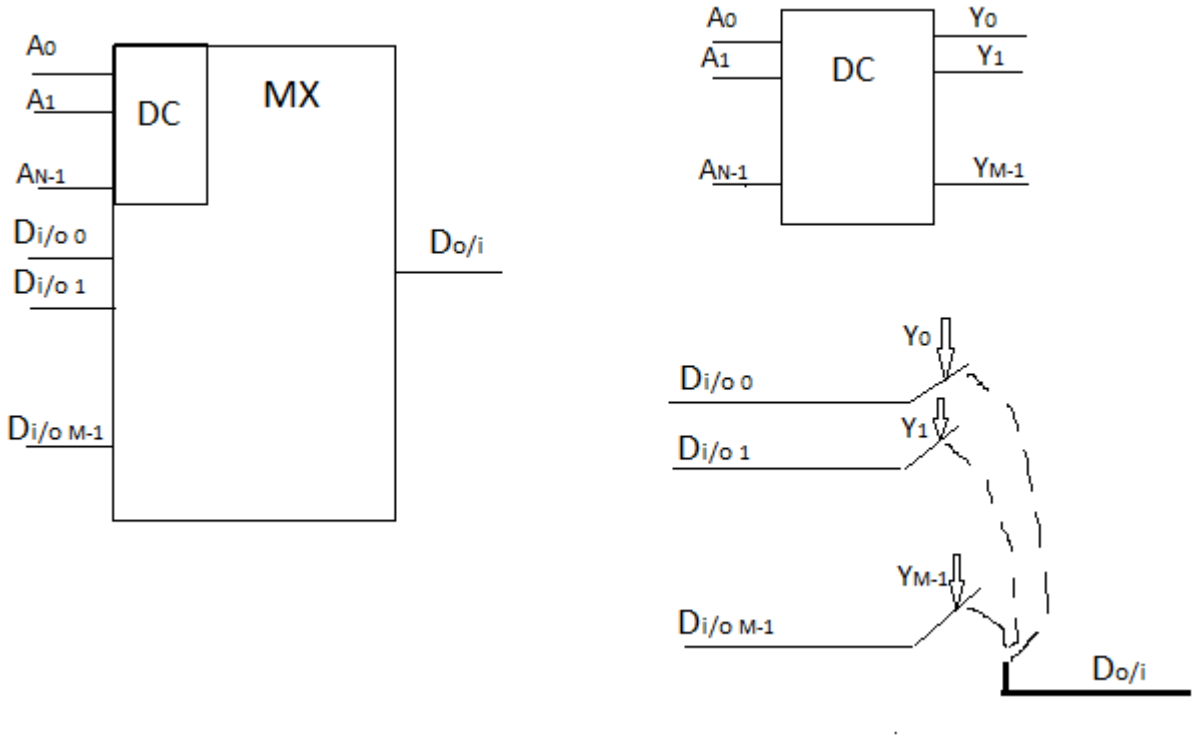
Do4 = A2 & nA1 & nA0 & Di;

Do5 = A2 & nA1 & A0 & Di;

Do6 = A2 & A1 & nA0 & Di;

Do7 = A2 & A1 & A0 & Di;

Универсальный коммутатор. Устройство, которое может функционировать как мультиплексор или как демультиплексор в зависимости от ситуации. Адресные входы включены в дешифратор, управляющий ключами направлений.



Лекция 3.

Конечные автоматы.

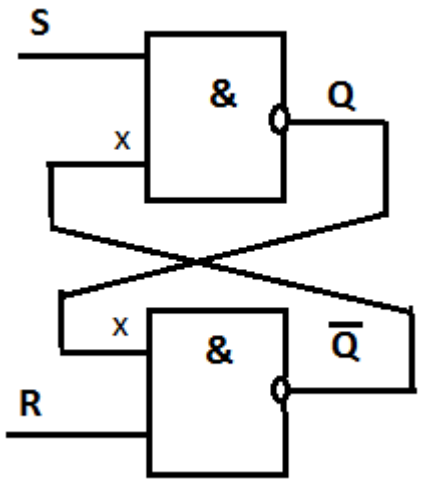
Конечные автоматы – цифровые устройства, содержащие в своей структуре обратные связи, таким образом, в каждый момент времени состояние выходов таких устройств зависит не только от входных воздействий, но и от состояний выходов в предыдущий момент времени.

Конечные автоматы. Триггер.

Простейшим конечным автоматом является триггер. Триггер – устройство, имеющее 2 устойчивых состояния (уровень 0 и уровень 1). Различают триггеры переключательного типа и триггеры установочного типа.

В основном, устройства вычислительной техники строятся на основе триггеров установочного типа. Простейшей ячейкой для любого типа триггера является асинхронный RS-триггер. Такой триггер имеет два выхода - пря-

мой и инверсный (устойчивым состоянием триггера всегда считается состояние прямого выхода Q, подтвержденное своей инверсией), и два входа S, Set – вход установки «1» и R, Reset – вход установки «0».



Рассмотрим структуру триггера, основанную на элементах 2И-НЕ.

Вспомним таблицу истинности для элемента 2И-НЕ:

Вход 1	Вход 2	Выход для «И»	Выход для «И-НЕ»
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

На основании этой таблицы можно записать тождества:

1 & X = not(X);

0 & X = 1;

Отсюда можно заключить, что управляющим уровнем для такой структуры является «0». Действительно, при подаче «0» на один из входов такого

Запишем таблицу функционирования триггера на момент времени n: входы S^n, R^n и предыдущее состояние выхода - Q^{n-1}. Выходы – прямой Q^n и инверсный - nQ^n

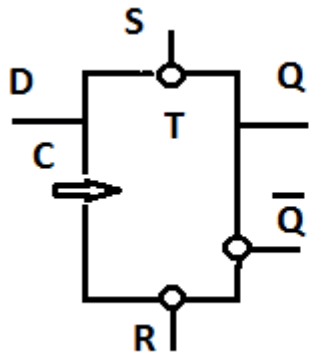
S^n	**R^n**	**Q^{n-1}**	**Q^n**	**nQ^n**
0	0	0	1	1
0	0	1	1	1
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1

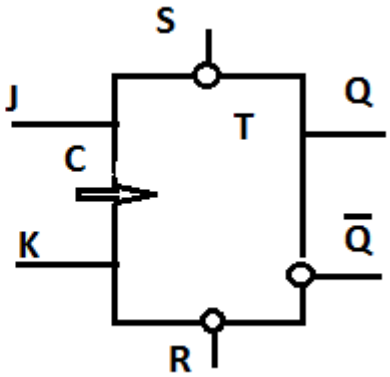
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

Анализируя таблицу истинности RS-триггера отметим, что:

1. при отсутствии сигнала управления ($S^n = 1$, $R^n = 1$, Q^{n-1} - любое), состояние каждого из элементов триггера определяется входом “X”, т.е. петлей обратной связи. Триггер находится в режиме хранения информации (выделено желтым);
2. подача сигнала управления на R-вход приводит к установке триггера в «0»
3. подача сигнала управления на S-вход приводит к установке «1»);
4. подача сигналов управления на оба входа одновременно приводит к неопределенности, на прямом и инверсном выходах устанавливается уровень «1». Выход из такого состояния в **режим хранения** не определяется. Поэтому подача управления на оба входа называется запрещенной комбинацией.

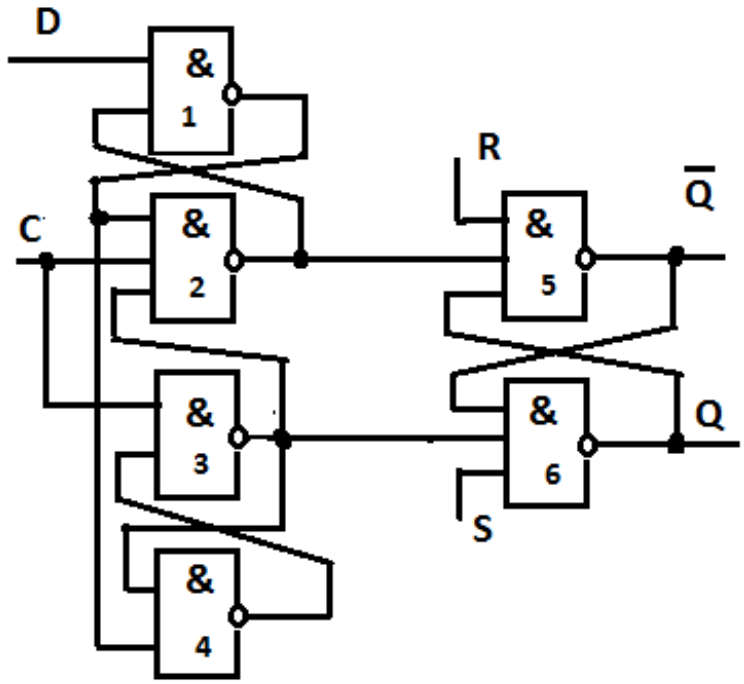
Асинхронная RS-ячейка – основа построения всех триггеров установочного типа. К этому типу относятся синхронные D-триггер и JK-триггер.





Триггеры имеют три типа входов: информационные – D (JK), вход синхронизации – C и установочные асинхронные входы S и R. По информационным входам в триггер поступает информация. Вход синхронизации определяет время записи информации в триггер. **По установочным входам триггер устанавливается в предписанное входом состояние независимо от состояния входа синхронизации и информации**. На время действия установочного сигнала вход синхронизации блокируется.

В качестве примера рассмотрим схему **D-триггера**.



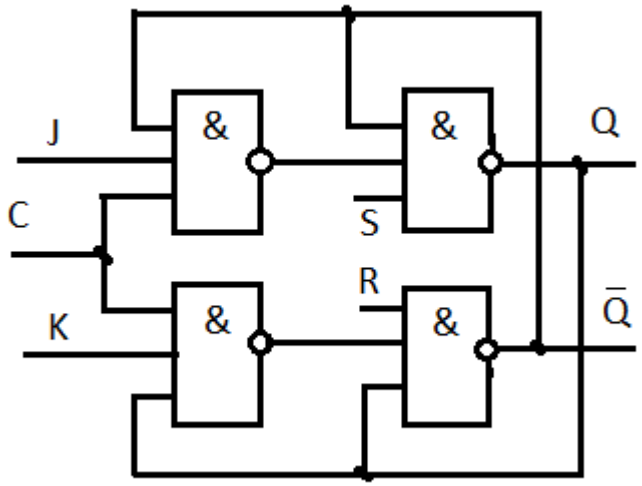
Схема, позволяющая фиксировать информацию в момент прихода фронта синхроимпульса, называется схемой 3-х триггеров. Она построена на элементах 1 – 4. Основная ячейка, в которой хранится информация, построена на элементах 5 – 6. Элементы, формирующие эту ячейку, имеют 3 входа. Внешними для нее являются установочные R и S входы. Для рассмотрения функционирования **динамического входа синхронизации** и D – входа считаем, что на установочные входы поступают уровни «1».

1. Момент времени, предшествующий подаче импульса синхронизации: C = 0, D = 1. Если C = 0, то выходы элементов «2» и «3» в состоянии «1», а, значит, ячейка на элементах «5» и «6» находится в режиме хранения, предположим, хранится «0». Теперь определим состояние выходов элементов «1» и «4». Т.к. вход D = 1 и выход элемента «2» в состоянии «1», то выход элемента «1» находится в состоянии «0». Этот уровень «0» будет удерживать выход элемента «4» в состоянии «1».
2. Следующий момент времени, D = 1, C = 1. Выход элемента «1» продолжает оставаться в состоянии «0», его состояние поступает на вход элемент «2», таким образом, на выходе элемента «2» продолжает удерживаться в «1». На вход элемента «3» теперь поступают «1» с входа C и от связи с выходом элемента «4». Теперь на выходе элемента «3» устанавливается «0», что приводит к установке в «1» основной ячейки – Q = 1, nQ = 0. Состояние «0» выхода элемента «3» позволяет удерживать в «1» выходы элементов «2» и «4» вне за-

висимости от состояния выхода элемента «1». Т.е. изменение информации на входе D теперь не влияет на состояние триггера.

JK – триггер.

Схема, иллюстрирующая информационные связи, образующие JK-триггер, представлена ниже. Из нее понятно, что управляющим уровнем на информационных входах является «1».

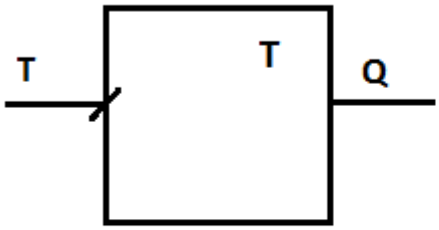


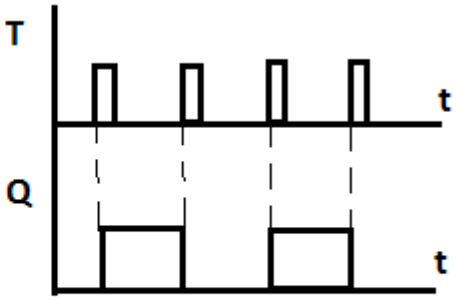
В JK-триггере информационные входы функционируют согласно следующей таблице (рассматривается в момент времени «n», т.е. момент прихода синхроимпульса).

J^n	K^n	Q^n
0	0	Q^{n-1}
1	0	1
0	1	0
1	1	nQ^{n-1}

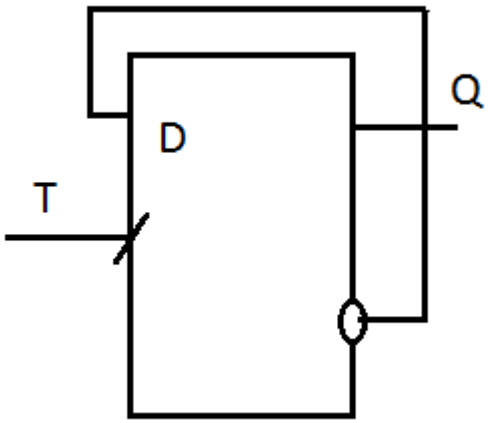
Активный уровень на информационных входах – 1. Вход J (Jump) –вход установки 1, вход K (Kill) – вход установки 0. Если оба входа неактивны триггер в режиме хранения, оба входа активны – состояние обратное предыдущему (режим Т- триггера). Значит, если объединить J и K входы и подавать на них «1», такой триггер будет работать в режиме Т-триггера, триггера переключательного типа. Поэтому JK-триггер называется универсальным (он имеет режимы работы триггера установочного типа и режим работы триггера переключательного типа). Такие свойства JK-триггера делают его удобным для построения на его основе переключательных схем, таких как, например, счетчики.

Т-триггер - триггер переключательного типа. Этот триггер имеет тактовый вход Т и единственный выход Q. Т-триггер изменяет свое состояние на противоположное в момент поступления на вход импульса с уровнем «1».





Внутренняя структура Т –триггера строится на основе D-триггера с динамическим входом синхронизации. В этом случае схема стабильна.



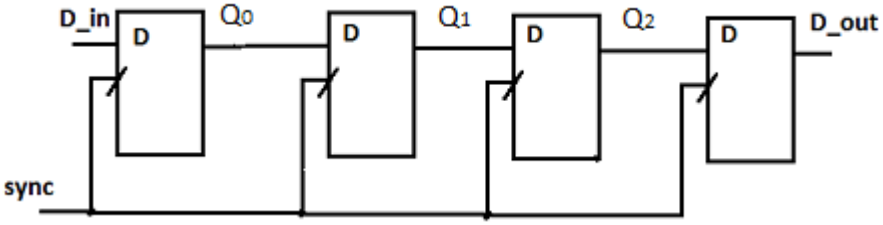
Триггеры переключательного типа имеют ограниченную область применения. Их схема не позволяет сделать предустановку устройства.

Лекция 4.

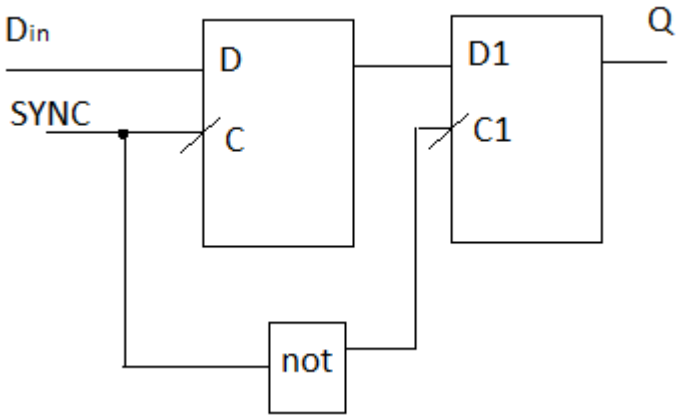
Регистры.

Регистры – конечные автоматы, служащие для сдвига информации (последовательные регистры) и для хранения информации (параллельные регистры).

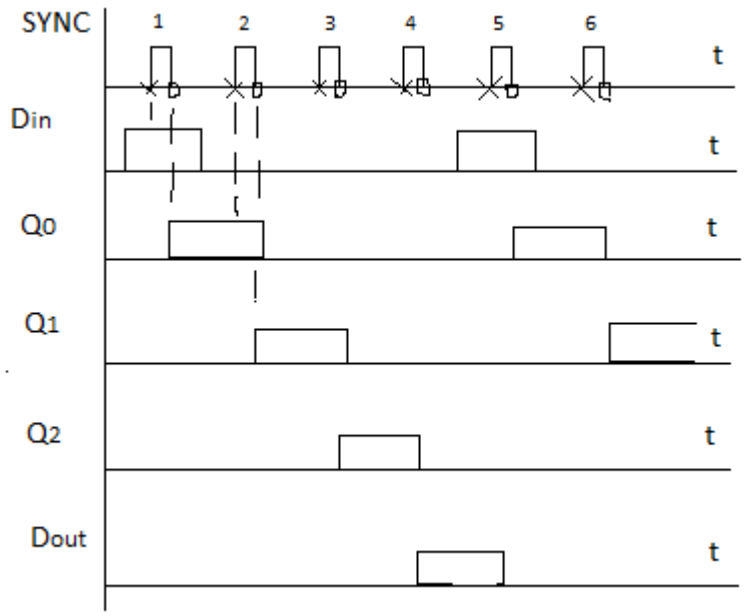
Последовательный регистр имеет вход данных и один выход данных.



По каждому поступающему на вход импульсу синхронизации информация, находящаяся на входе D, записывается в разрядный триггер. Но в случае, если задержка срабатывания триггера оказывается сопоставимой с длительностью фронта синхроимпульса, одна и та же информация может записаться в два (или более) триггера подряд. Во избежание таких сбоев между триггерами ставят линии задержки, или триггеры строятся по двухтактной схеме.



Как видно на рисунке, в двухтактной схеме информация записывается в первый триггер по фронту синхроимпульса, а появляется на выходе Q на спаде синхроимпульса. Изобразим временную диаграмму работы 4-разрядного регистра сдвига, обозначим фронт синхроимпульса «х», а спад синхроимпульса «0».



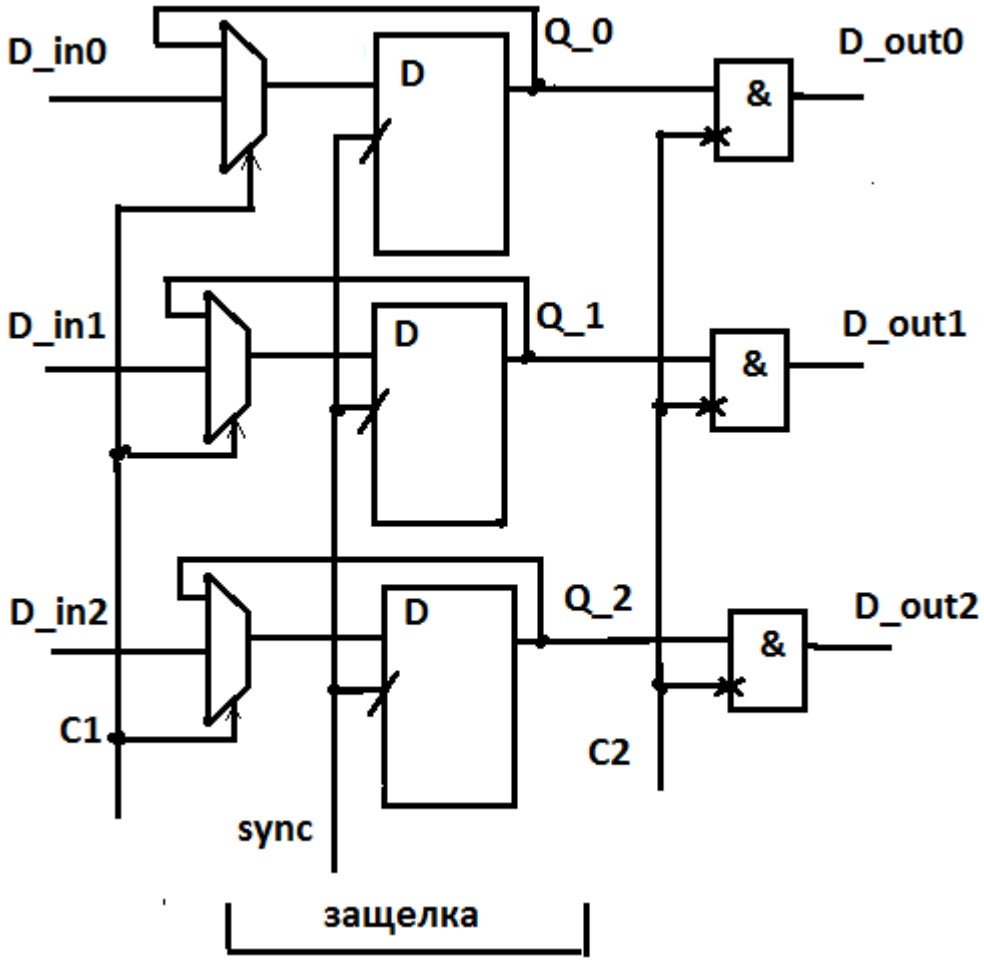
Как очевидно из графика, информация сдвигается по разрядам регистра. Количество подаваемых импульсов синхронизации всегда соответствует числу разрядов, на которое сдвигается информация.

Параллельный регистр имеет количество входов и выходов данных, обеспечивающих разрядность подключенной шины.

Регистр, входы и выходы которого всегда подключены к шинам, называется регистр-защелка. Регистр, входы и выходы которого подключаются к шине по управляющему сигналу – буферный регистр.

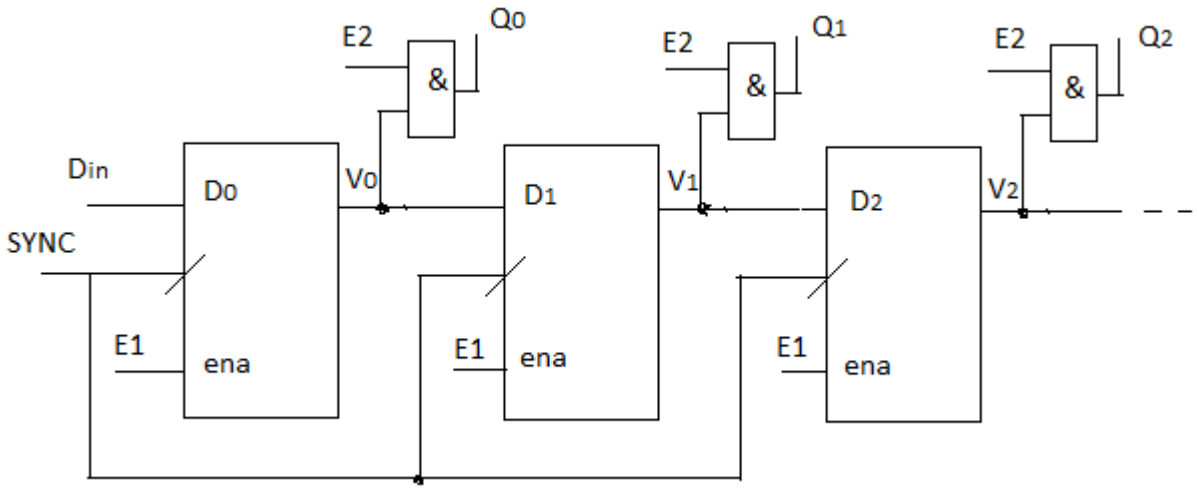
Запись производится при поступлении фронта синхросигнала. На рисунке C1 – сигнал управления буфером записи, а C2 – сигнал управления буфером считывания. Буфер записи представлен мультиплексорами, позволяющими пропускать на входы информацию с шины, или сохранять ранее записанную. Буфер чтения позволяет «отрывать» выходы от шины или подключать к шине. Иногда в качестве буфера чтения используются просто элементы И. Тогда на выходе неподключенного регистра читают 0.

.

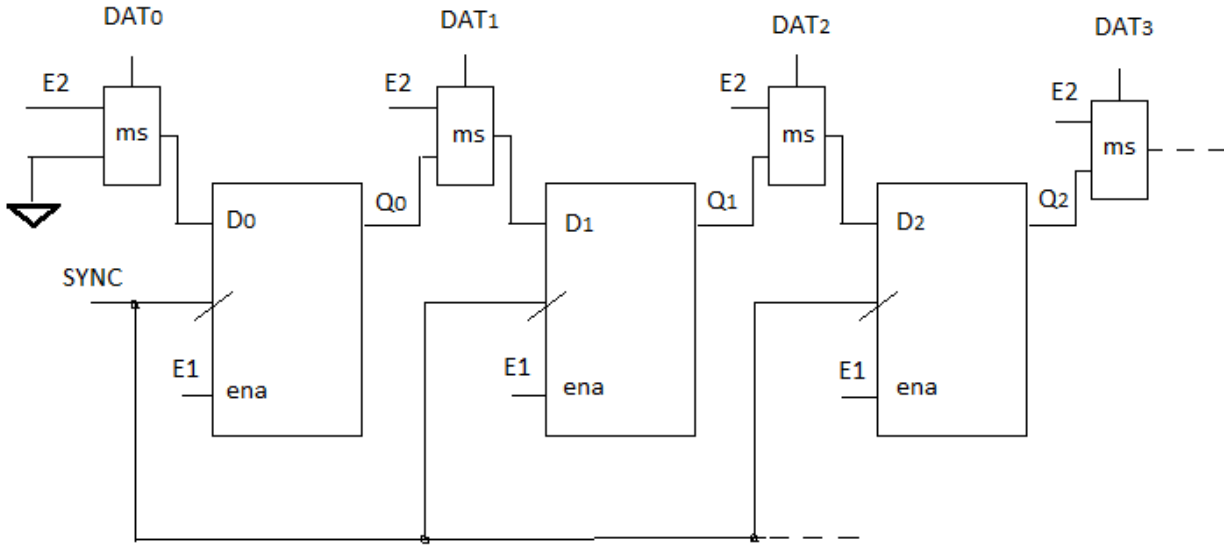


Очевидно, что запись возможна на синхроимпульсе при условии подключения входов к шине (**sync + C1=1**), а считывание – в любое время, когда выходы подключены к шине (**C2=1**)

По способу записи-считывания различают также **последовательно-параллельный** и **параллельно-последовательный** регистры.



В **последовательно-параллельном** регистре набор информации происходит в последовательном n-разрядном регистре. В течение времени подачи n синхроимпульсов условием набора является **E1=1, E2=0**, затем набранная информация должна переписаться в параллельный буфер. При этом должно быть условие **E1=0, E2=1**, для записи в буфер достаточно одного синхроимпульса. Весь процесс регулируется с помощью счетчика.



В **параллельно-последовательном** регистре запись информации производится параллельным способом при подаче одного импульса синхронизации и условии **E1=1** и **E2=1**. Считывание происходит путем сдвига информации и заполнения освобождающихся разрядов «0». Условие такого действия **E2=0**, а **E1=1**, количество подаваемых синхроимпульсов соответствует разрядности регистра. После выдачи информации в линию возможна следующая запись. Весь процесс также регулируется с помощью счетчика.

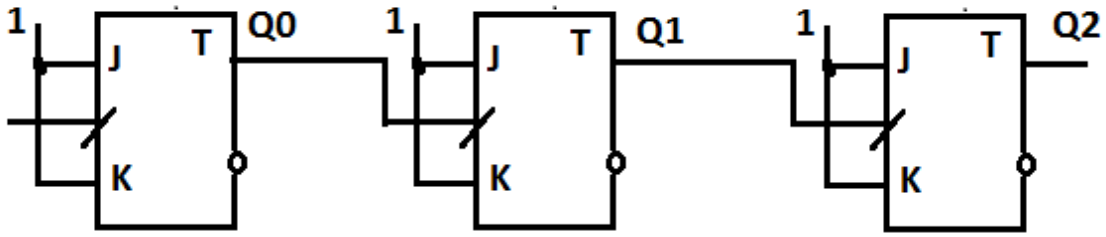
Лекция 5.

Счетчики.

Счетчики – конечные автоматы, каждое последующее состояние которых на 1 отличается от предыдущего. Счетчики всегда подсчитывают количество поступающих на них синхроимпульсов. Кроме того, счетчики могут использоваться для деления частоты входного сигнала синхронизации.

Счетчики могут классифицироваться по порядку счета (суммирующие, вычитающие и реверсивные), по способу подачи синхроимпульса (асинхронные и синхронные) и по количеству состояний в цикле счета (двоичные – $K = 2^n$ и недвоичные – $K < 2^n$, где n – количество разрядных триггеров).

В асинхронных счетчиках порядок счета определяется связями между выходами разрядных триггеров и входами синхронизации. Связь с прямого выхода используется для построения вычитающего счетчика, связь с инверсного выхода – для построения суммирующего.



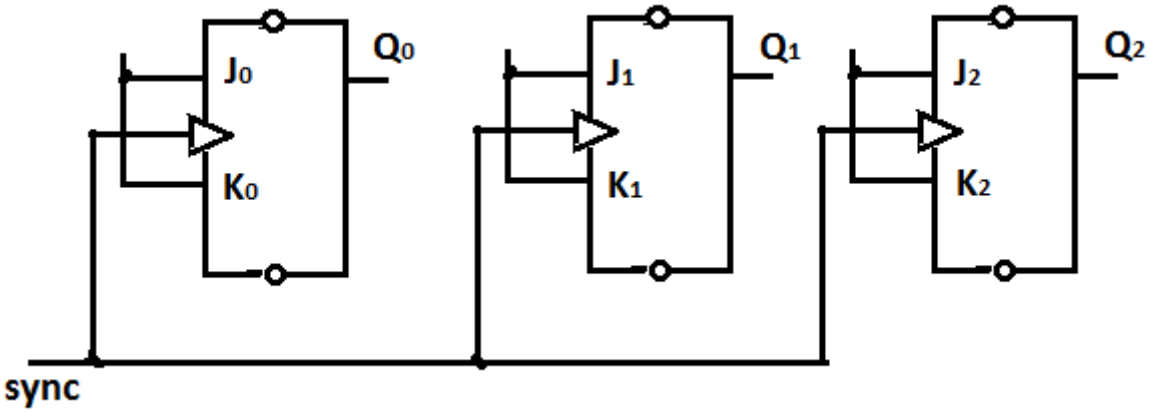
Перед вами схема асинхронного счетчика, построенного на основе JK-триггеров со связями входов синхронизации с прямыми выходами разрядных триггеров. Каждый разрядный триггер будет менять свое состояние по положительному фронту импульса синхронизации, т.к. входы JK установлены в 1. Внешний импульс синхронизации подается только на триггер нулевого разряда. Считаем, что до подачи импульса синхронизации счетчик показывает состояние «0» - Q0 = 0, Q1 = 0, Q2 = 0. Подаем первый синхроимпульс. Состояние триггера нулевого разряда изменяется, теперь Q0 = 1, значит, на входе триггера первого разряда возникает положительный перепад, который обозначим «/». Этот перепад в свою очередь изменяет состояние триггера первого разряда, Q1 = 1. Теперь и на входе триггера второго разряда возникает положительный перепад «/», таким образом, состояние этого триггера также изменяется Q2 = 1. Подаем второй синхроимпульс. Состояние триггера нулевого разряда опять меняется на противоположное, но теперь Q0 = 0. Теперь на входе триггера первого разряда возникает уже отрицательный перепад, который обозначим «\». Но такой перепад не приводит к изменению состояния триггера первого разряда и по-прежнему остается Q1 = 1. Соответственно, не меняется и состояние триггера второго разряда – Q2 = 1. Составим таблицу переключений такого счетчика

№ п/п	Q0	перепад	Q1	перепад	Q2	Десятичное значение
0	0	-	0	-	0	0
1	1	«/»	1	«/»	1	7
2	0	«\»	1	-	1	6
3	1	«/»	0	«\»	1	5
4	0	«\»	0	-	1	4
5	1	«/»	1	«/»	0	3
6	0	«\»	1	-	0	2
7	1	«/»	0	«\»	0	1
8	0	«\»	0	-	0	0

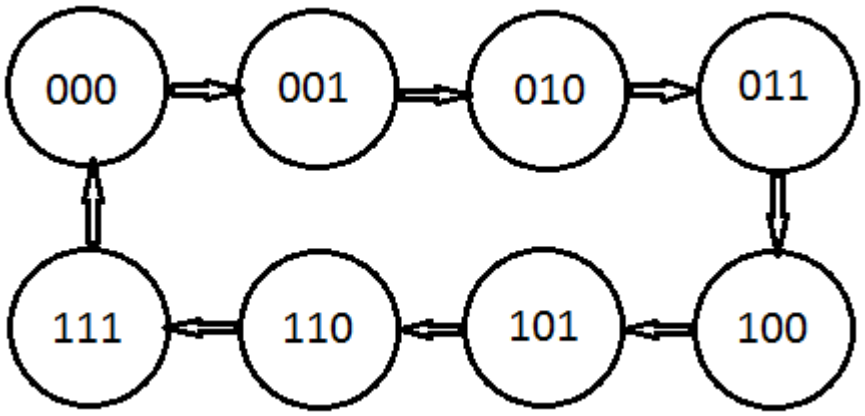
Из таблицы видно, что построенный по такой схеме счетчик будет вычитающим. Для построения суммирующего асинхронного счетчика необходимо использовать инверсные выходы разрядных триггеров.

Нетрудно убедиться в том, что асинхронные счетчики имеют большую задержку установки состояния, так как задержка счетчика накапливается из задержек триггеров. По этой причине используются, в основном, синхронные счетчики.

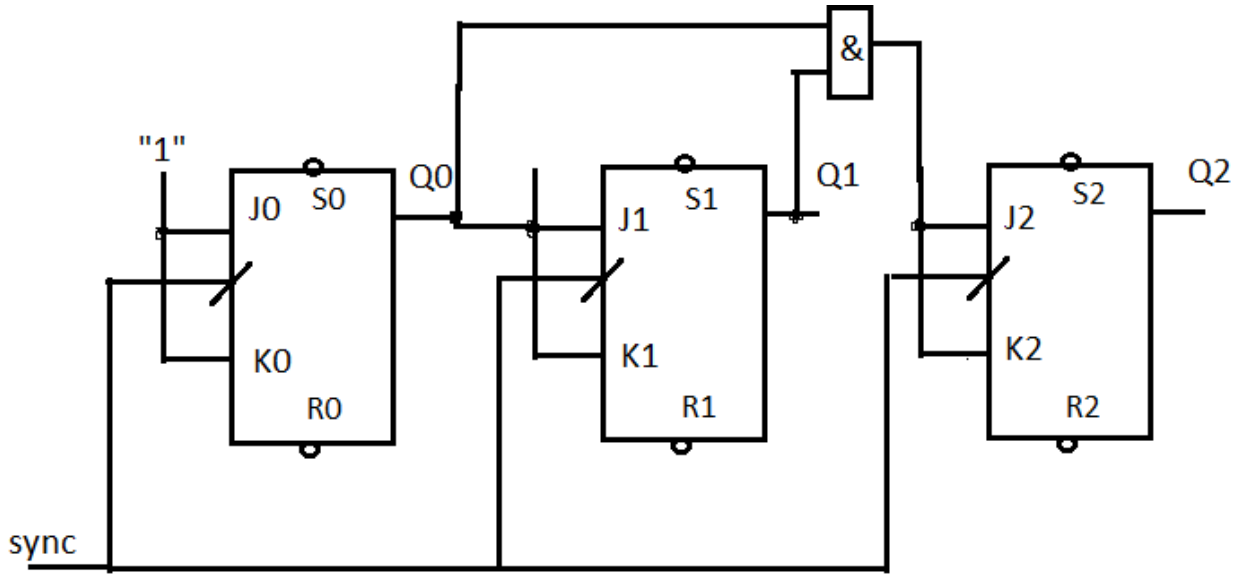
В синхронных счетчиках порядок счета устанавливается связями между выходами триггеров и их информационными входами. Рассмотрим построение суммирующего трехразрядного синхронного счетчика. Счетчик будем строить на основе JK-триггеров. Счетчик будет иметь все возможные состояния (двоичный), поэтому можем объединить входы J и K для каждого разряда и посмотреть, каким образом соединить их с выходами разрядных триггеров**.**



Сначала запишем граф переключений такого счетчика



В обозначениях считаем крайний правый разряд – Q0, затем справа-налево Q1 и Q2. **Импульс синхронизации поступает на все триггеры одновременно! В каждом круге записано состояние счетчика, полученное после прихода очередного импульса синхронизации.** Заметим, что Q0 изменяется на каждый приход импульса синхронизации, можем постоянно подавать на объединенный вход J_0K_0 единицу. Состояние первого разряда изменяется в каждом такте, следующем за появлением «1» в нулевом разряде. Так, после состояния 001, где Q0 = 1, возникает состояние 010, где уже Q1 = 1, а после состояния 011 (Q0 = 1) возникает состояние 100, где Q1 = 0. Если мы соединим выход Q0 с входами J_1K_1, то накопленная «1» в нулевом разряде станет «подталкивать» счет в первом. Теперь нетрудно заметить, что состояние второго разряда изменяется в такте, последующем за накоплением единиц в предыдущих двух. За состоянием 001(3) следует 100(4), а за состоянием 111(7) всегда последует 1000(8), что для 3-х разрядов рассматривается как 000. Поэтому, логически умножив состояния предыдущих выходов разрядных триггеров, мы подаем результат этого умножения на информационные входы. Для второго разряда – на входы J_2K_2. Полученная схема будет выглядеть следующим образом



В суммирующем счетчике, где изменение каждого последующего разряда возможно лишь при установке предыдущих в 1, уравнение связей записывается как

$J_iK_i = Q_0 \& Q_1 \& \ldots \& Q_{i-1}$;

Если рассматривать вычитающий счетчик, то очевидно, что изменение каждого последующего разряда будет происходить после списания предыдущих разрядов в «0». Таким образом, необходимо соединять информационные входы разрядных триггеров уже не с прямыми, а с инверсными выходами. В этом случае уравнение связей будет записываться как

$J_iK_i = nQ_0 \& nQ_1 \& \ldots \& nQ_{i-1}$;

В реверсивном счетчике, где вводится дополнительный сигнал – реверс (REV), позволяющий производить вычитание при значении REV=1 и сложение при значении REV=0 уравнение связи можно записать следующим образом:

$J_iK_i = (Q_0 \& nREV \vee nQ_0 \& REV) \& (Q_1 \& nREV \vee nQ_1 \& REV) \& \ldots$

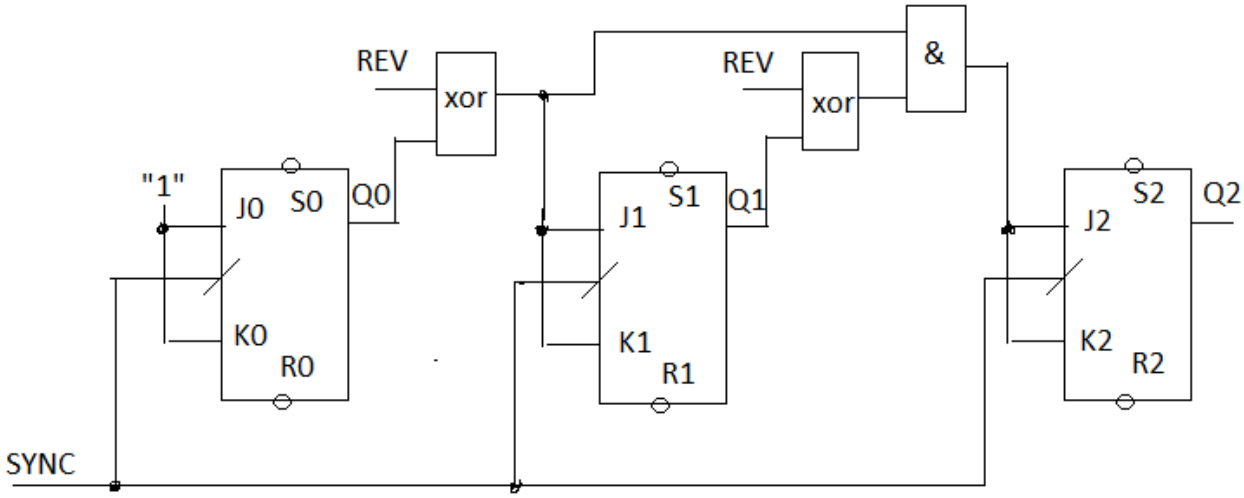
$\& (Q_{i-1} \& nREV \vee nQ_{i-1} \& REV)$;

Выражение в скобках не что иное, как неравнозначность между состоянием предыдущего выхода и значением реверса, поэтому запишем окончательное уравнение

$J_iK_i = (Q_0 \text{ xor } REV) \& (Q_1 \text{ xor } REV) \& \ldots \& (Q_{i-1} \text{ xor } REV)$;

Очевидно, что сигнал REV позволяет переключать связи, определяющие порядок счета.

Схема реверсивного синхронного счетчика выглядит следующим образом



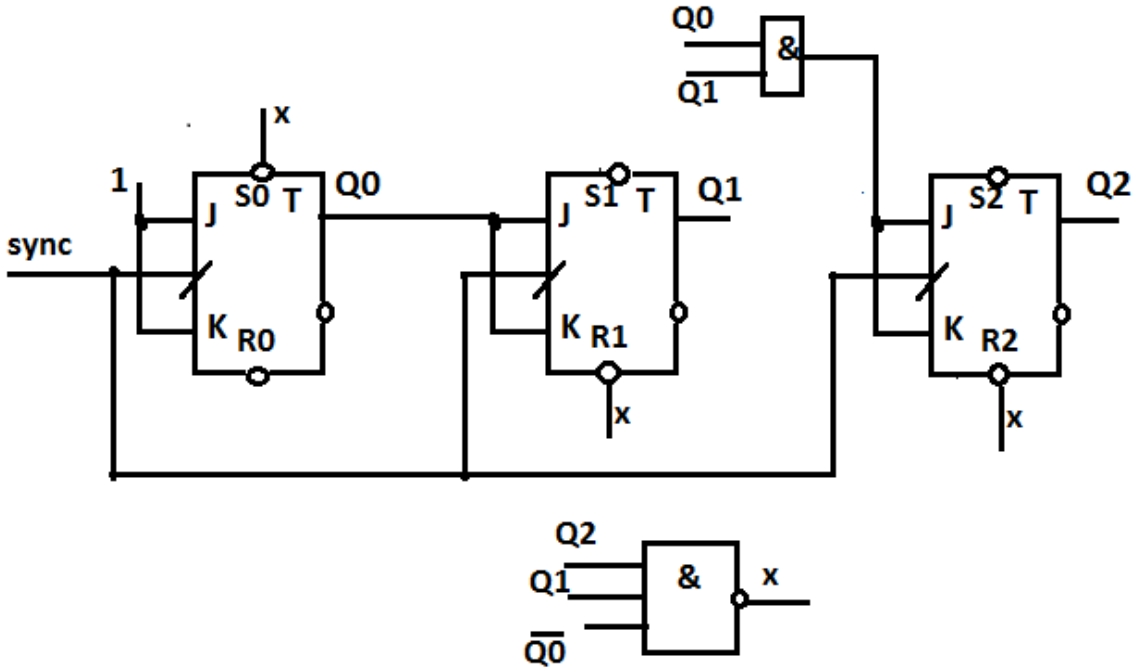
Следует заметить, что реверсивный счетчик корректно работает только по синхронной схеме. При построении асинхронного счетчика все связи прописываются на входы синхронизации, и изменение состояния реверса может оказать воздействие на триггер, как ложный синхроимпульс. В этом случае счетчик будет вычитать не из того значения, на котором остановился при суммировании.

Недвоичные счетчики. Такие счетчики имеют неполный цикл счета, причем переход в начальное состояние может совершаться по воздействию на установочные входы (асинхронная установка) или во время подачи импульса синхронизации (синхронная установка).

Схема недвоичного счетчика с асинхронной установкой имеет те же связи выходов и информационных входов разрядных триггеров, как в двоичном счетчике. По каждому синхроимпульсу счетчик переходит в следующее состояние, но состояние, последующее за конечным, позволяет получить на выходе управляющей логики сигнал установки начального состояния. Рассмотрим пример:

Необходимо построить суммирующий счетчик, имеющий состояния от “1” до “5”. Его можно получить из 3- разрядного двоичного суммирующего счетчика с начальной установкой в «1» Q_0=1, Q_1=0, Q_2=0 (воздействие производим сигналом «х» по входам S0, R1, R2). Сигнал «х» уровня «0» формируем из состояния, следующего за конечным состоянием “5”.По следующему импульсу синхронизации счетчик перейдет в состояние “6” - Q_0=0, Q_1=1, Q_2=1. В ответ на эту комбинацию элемент 3И-НЕ выдаст на выходе «х» уровень «0». Соответственно при подаче этого уровня на входы S0, R1, R2 счетчик установится в начальное состояние “1”. Однако, при асинхронной установке мы имеем более короткое время существова-

ния начального состояния по отношению к остальным, так как в этом же такте возникает и состояние, следующее за конечным. Длительность удержания последнего равна задержке элемента 3И-НЕ плюс задержка установки триггера.



Лекция 6.

Недвоичные счетчики (продолжение).

В случае синхронной установки все состояния счетчика удерживаются в течение такта (периода синхроимпульса), то есть одинаковы по времени. Рассмотрим тот же пример, счет от “1” до “5”, но для случая с синхронной установкой. Для начала запишем таблицу переключений такого счетчика:

Q_2^{n-1}	Q_1^{n-1}	Q_0^{n-1}	Q_2^{n}	Q_1^{n}	Q_0^{n}
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	0	0	1

В таблице слева записано поразрядно предыдущее состояние счетчика, а справа, также поразрядно, состояние счетчика после прихода импульса синхронизации в момент времени “n”. Теперь выделим все переходы для

каждого разряда и запишем, какие уровни надо подать на входы J и K к моменту времени “n”, чтобы такие переходы осуществились.

Q_2^{n-1}	Q_2^{n}	J_2^{n}	K_2^{n}
0	0	0	X
0	0	0	X
0	1	1	X
1	1	X	0
1	0	X	1

Поясним таблицу. Для этого вспомним таблицу переходов JK-триггера

$\mathbf{J^n}$	$\mathbf{K^n}$	$\mathbf{Q^n}$
0	0	Q^{n-1}
1	0	1
0	1	0
1	1	nQ^{n-1}

Если триггер второго разряда после подачи синхроимпульса сохраняет состояние «0», то из таблицы переключений триггера видно, что это возможно для первой и третьей строк, когда $Q^n=Q^{n-1}$ или $Q^n=0$. То есть при любом значении на входе K (обозначим как X) состояние входа J=0. Если триггер второго разряда после подачи синхроимпульса переходит в «1», то нас интересуют уже вторая и четвертая строки таблицы переключений триггера, когда $Q^n=1$ или $Q^n=nQ^{n-1}$. То есть теперь при любом значении на входе K (обозначим как X) состояние входа J=1. Аналогично можно рассмотреть сохранение на выходе состояния «1» - первая и вторая строки таблицы, когда при любом J K=0, или переход в состояние «0» - третья и четвертая строки таблицы, когда при любом J K=1.

Теперь по такому же принципу запишем таблицы воздействий для первого и нулевого разрядов

Q_1^{n-1}	Q_1^{n}	J_1^{n}	K_1^{n}
0	1	1	X
1	1	X	0
1	0	X	1
0	0	0	X
0	0	0	X

Q_0^{n-1}	Q_0^{n}	J_0^{n}	K_0^{n}
1	0	X	1
0	1	1	X

1	0	X	1
0	1	1	X
1	1	X	0

Запишем полную таблицу воздействий для всех разрядов счетчика

Q_2^{n-1}	Q_1^{n-1}	Q_0^{n-1}	Q_2^{n}	Q_1^{n}	Q_0^{n}	J_2^{n}	K_2^{n}	J_1^{n}	K_1^{n}	J_0^{n}	K_0^{n}
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	0	0	1	X	1	0	X	X	0

Теперь запишем уравнения связей между состояниями выходов в момент времени, предшествующий приходу синхроимпульса (выделено красным) и каждым входом.

1. K_0^{n}. Если проследить изменения этой функции, принимая X за любое значение можно сказать, что она повторяет инверсные значения Q_2^{n-1}. Запишем $K_0^{n} = nQ_2^{n-1}$;
2. J_0^{n}. Эта функция никогда не принимает значения «0», поэтому запишем $J_0^{n} = 1$;
3. K_1^{n}. Изменения этой функции повторяют значения Q_0^{n-1}. (Помним, что значение X можно выбирать любое). Запишем $K_1^{n} = Q_0^{n-1}$;
4. J_1^{n}. Эта функция также повторяет инверсные значения Q_2^{n-1}. Поэтому снова записываем: $J_1^{n} = nQ_2^{n-1}$;
5. K_2^{n}. Здесь снова видим повторение Q_0^{n-1}. Записываем: $K_2^{n} = Q_0^{n-1}$;
6. J_2^{n}. В этом случае функция не содержит прямых повторений состояний предыдущих выходов, поэтому пишем выражение, при котором функция обращается в «1»: $J_2^{n} =(nQ_2^{n-1})\&(Q_1^{n-1})\&(Q_0^{n-1})$;

Q_2^{n-1}	Q_1^{n-1}	Q_0^{n-1}	J_2^{n}	K_2^{n}	J_1^{n}	K_1^{n}	J_0^{n}	K_0^{n}
0	0	1	0	X	1	X	X	1
0	1	0	0	X	X	0	1	X
0	1	1	1	X	X	1	X	1
1	0	0	X	0	0	X	1	X
1	0	1	X	1	0	X	X	0

Теперь запишем систему уравнений полностью:

$K_0^{n} = nQ_2^{n-1}$;

$J_0^n = 1$;

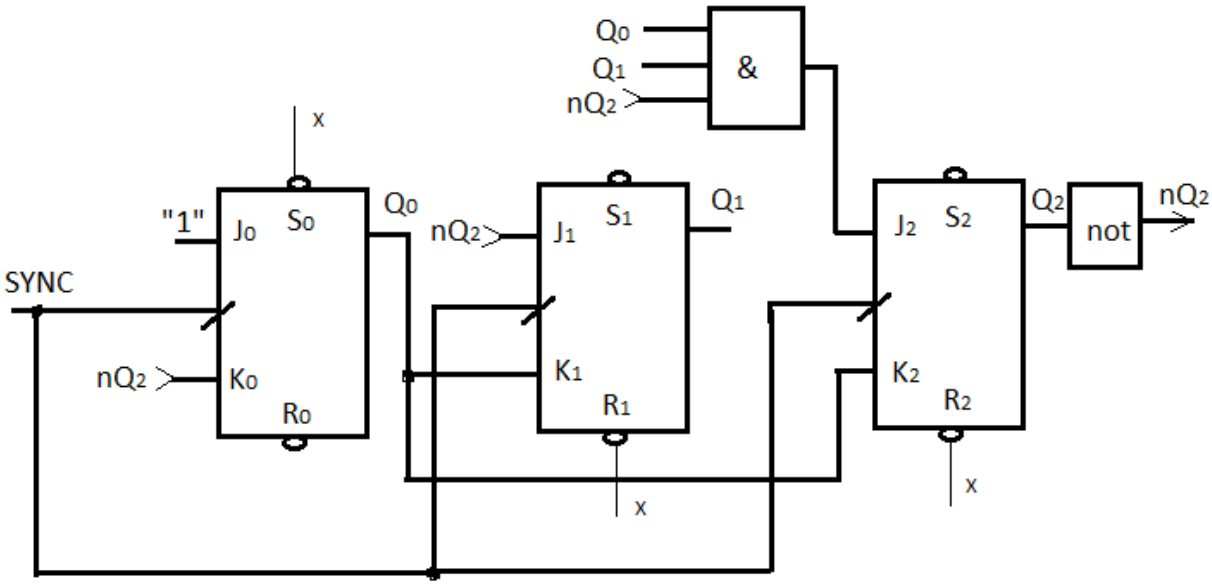
$K_1^n = Q_0^{n-1}$;

$J_1^n = nQ_2^{n-1}$;

$K_2^n = Q_0^{n-1}$;

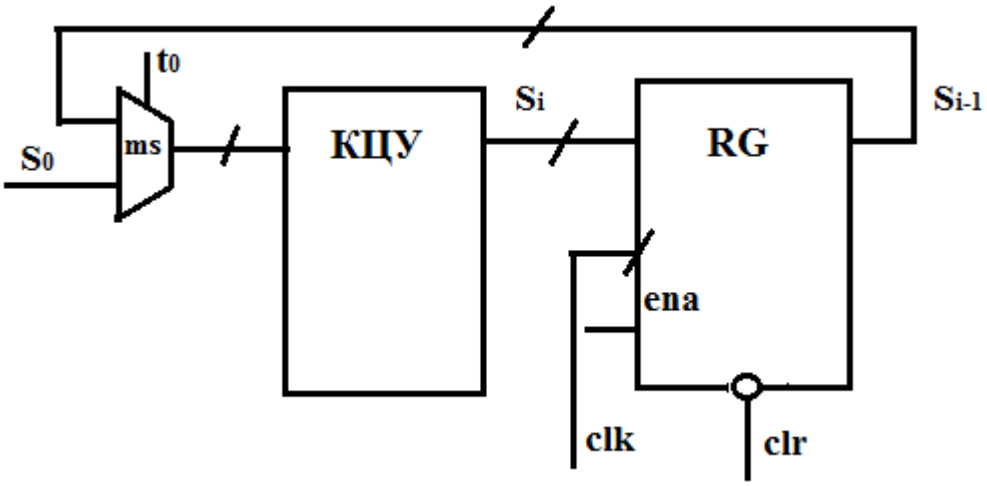
$J_2^n = (nQ_2^{n-1})\&(Q_1^{n-1})\&(Q_0^{n-1})$;

Этой системой мы описали связи информационных входов с выходами разрядных триггеров, необходимые для поддержания заданного порядка переключений. При этом схема счетчика будет выглядеть следующим образом



Перед началом счета счетчик устанавливается в начальное состояние асинхронно, с помощью воздействия сигналом «x» на установочные входы.

Из рассмотренных схем счетчиков можно выделить общую конструкцию конечного автомата, в котором информация может преобразовываться в каждом такте на основе предыдущей. Эта конструкция всегда состоит из двух частей: регистра, для сохранения информации и кодопреобразующего КЦУ.

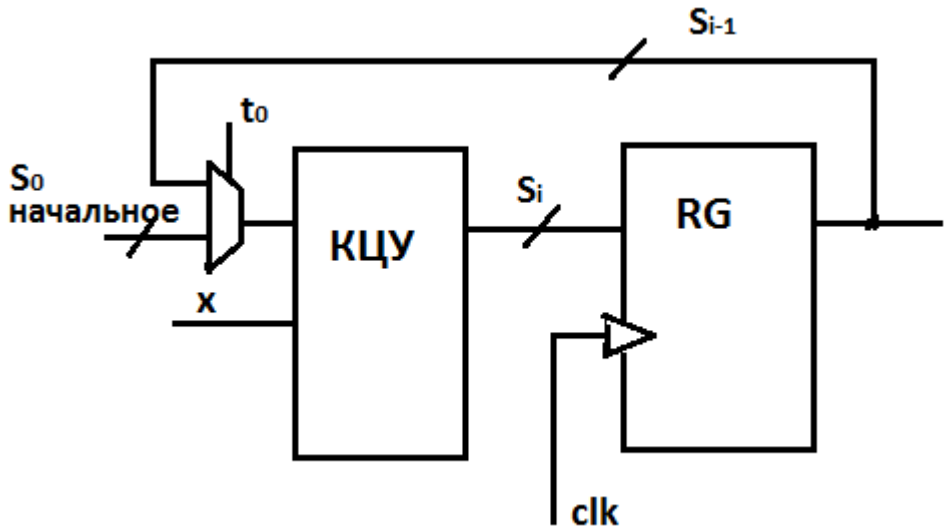


Процесс синтеза недвоичных счетчиков с синхронной установкой является частным случаем синтеза конечных автоматов с произвольной сменой состояний. Существуют два типа таких автоматов: автомат Мура и автомат Мили. В любом типе автоматов направление перехода из одного состояния в другое всегда зависит от внешнего воздействия. Но в автомате Мура каждому состоянию соответствует только одно событие, а в автомате Мили такое соответствие также зависит от внешнего воздействия. В любом случае для такого синтеза необходимо определить уравнения связей информационных входов с выходами разрядных триггеров. Для этого по имеющейся в задании таблице переключений строится таблица воздействий, с помощью которой возможна запись уравнений связей.

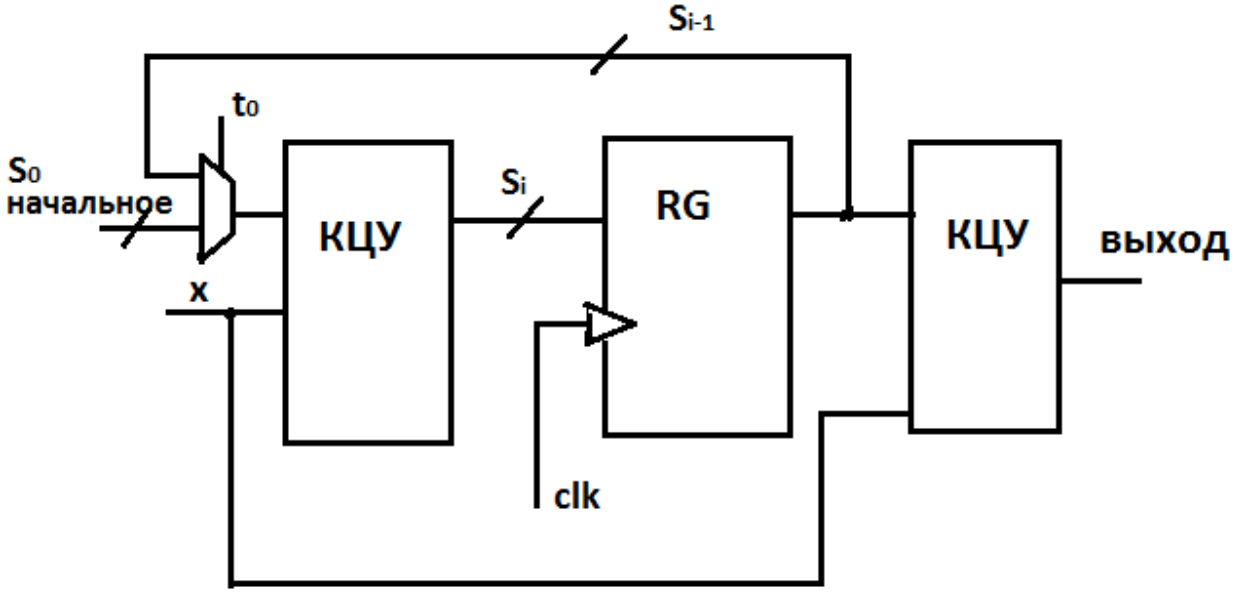
Таким образом, произведя синтез недвоичного счетчика, мы рассмотрели частный случай автомата Мура.

Рассмотрим теперь синтез автомата Мили.

Структура автомата может быть представлена следующим образом



Автомат Мура.



Автомат Мили.

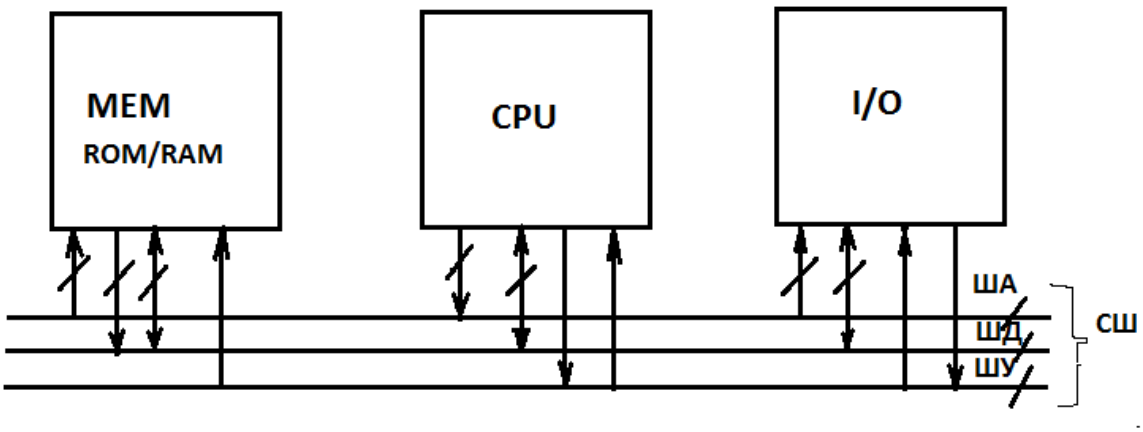
Лекция 7.

Микропроцессоры.

Структура микропроцессорной системы.

Любая микропроцессорная система состоит из трех основных компонентов: собственно микропроцессора, производящего операции над данными, области памяти, где хранятся коды программ работы процессора и обрабатываемые массивы данных, и области устройств ввода-вывода, состоящую

из схем, адаптирующих процессорную систему к внешним устройствам. Соединение блоков микропроцессорной системы производится по системной шине (СШ).



Системная шина состоит из трх групп шин:

-шины адреса, на которую процессор выставляет адрес устройства в пространстве памяти или пространстве ввода-вывода, с которым будет поизводиться обмен информацией;

-шины данных, по которой производится обмен информацией;

-шины управления, по которой процессор посылает управляющие сигналы и получает запросы от устройств ввода-вывода.

Шина адреса однонаправлена, информация, следующая по ней, многоразрядна. Шина данных двунаправлена и информация, следующая по ней, также многоразрядна. Шина управления состоит из отдельных проводников, каждый из которых передает определенный сигнал управления. Формировать управляющие сигналы может процессорный блок и блок устройств ввода-вывода (запросы на процессорный блок).

Для полноценного обмена процессора по шине данных важны три момента: направление обмена (чтение/запись), обьект обмена (память/устройства ввода-вывода) и, наконец, структура информации (данные/код).

Устройства памяти.

Внутренняя память микропроцессорной системы по способу доступа к ячейкам накопителя делится на 3 типа: адресная память, память с последовательным доступом и ассоциативная память.

В адресной памяти доступ к любой ячейке накопителя возможен по любому выставленному на шине адресу, независимо от предыдущего обращения.

В памяти с последовательным доступом порядок обращения к ячейкам задается счетчиком адресов, который невозможно переустановить в процессе работы с памятью. Таким образом, адрес обращения к каждой последующей ячейке отличается от предыдущего всегда на определенную величину.

В ассоциативной памяти ячейка накопителя содержит информацию, скопированную из основной адресной памяти. При этом некоторая часть адреса этой основной памяти, так называемый тег, или признак, также сохраняется в ассоциативной памяти. И, если вызываемый процессором адрес содержит такой тег, информация извлекается из накопителя ассоциативной памяти.

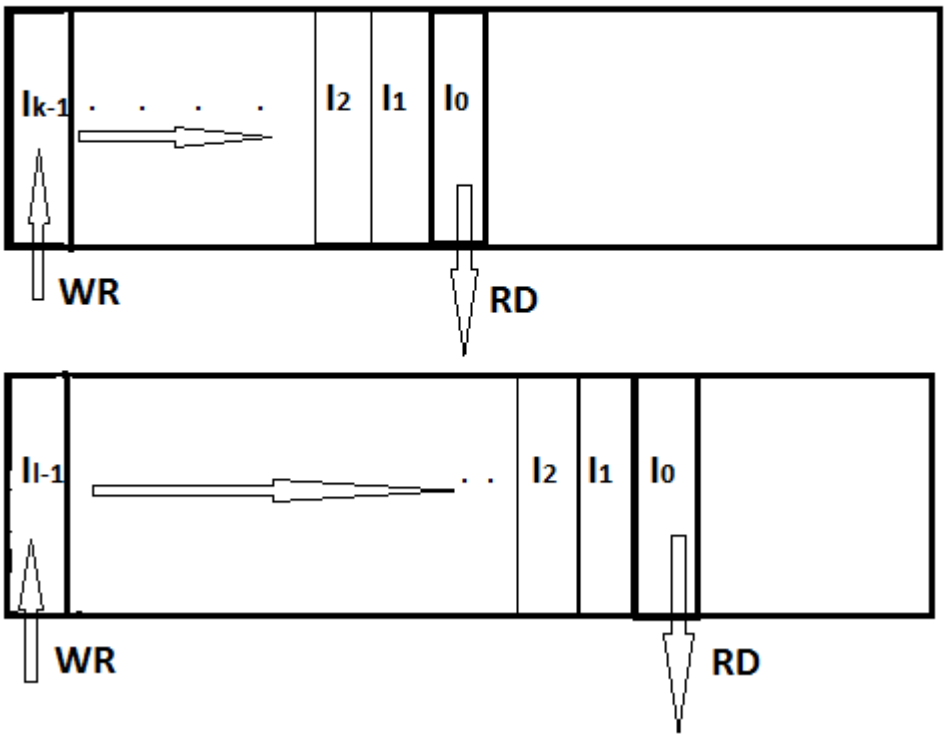
Пример ассоциативной памяти CACH-память. Данная структура в настоящем курсе не рассматривается.

Память с последовательным доступом.

Примеры памяти с последовательным доступом: память **FIFO(first input, first output)** и память **LIFO(last input, first output),** или **stack**.

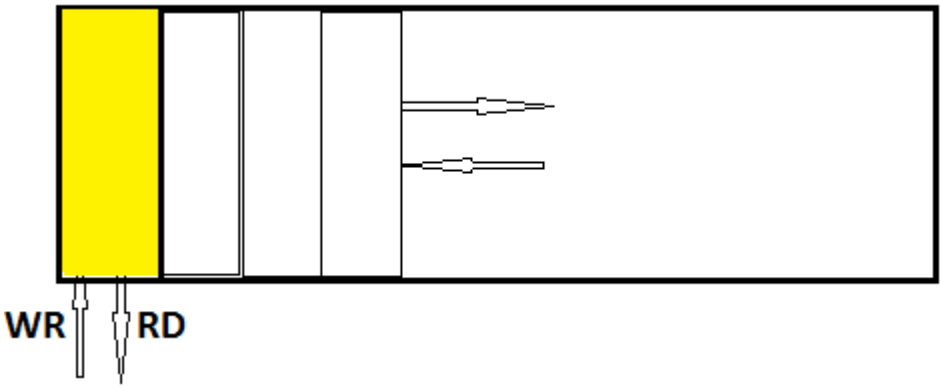
В FIFO процессы записи и считывания обсчитываются двумя счетчиками адресов. Оба счетчика суммирующие. Для исключения ложного считывания (чтения ячейки, куда не производилась запись, счетчик записи имеет возможность реверса).

FIFO на n ячеек с записью k и l единиц информации



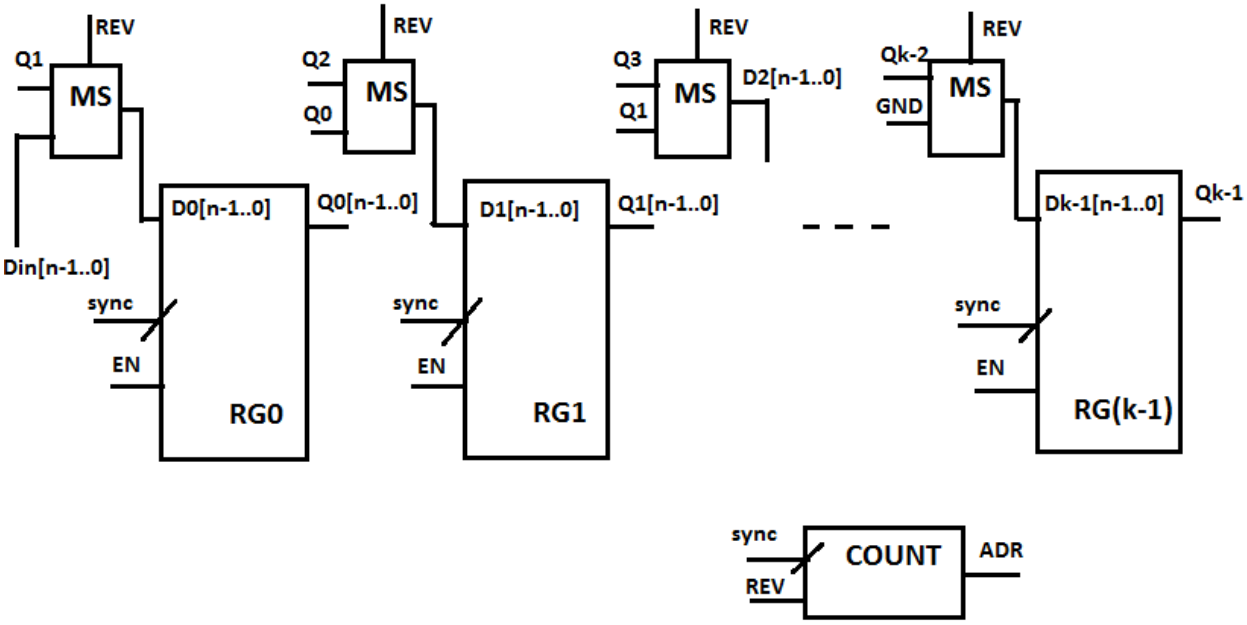
Внешние стрелки показывают информацию, доступную для просмотра.

В LIFO доступна для просмотра только одна точка, поэтому адреса записи и считывания обсчитываются одним реверсивным счетчиком.



Адресуемая ячейка, через которую производится запись и считывание информации называется вершиной стека.

На схеме показана структура n-разрядного стека глубиной k. Вершина стека – RG0.

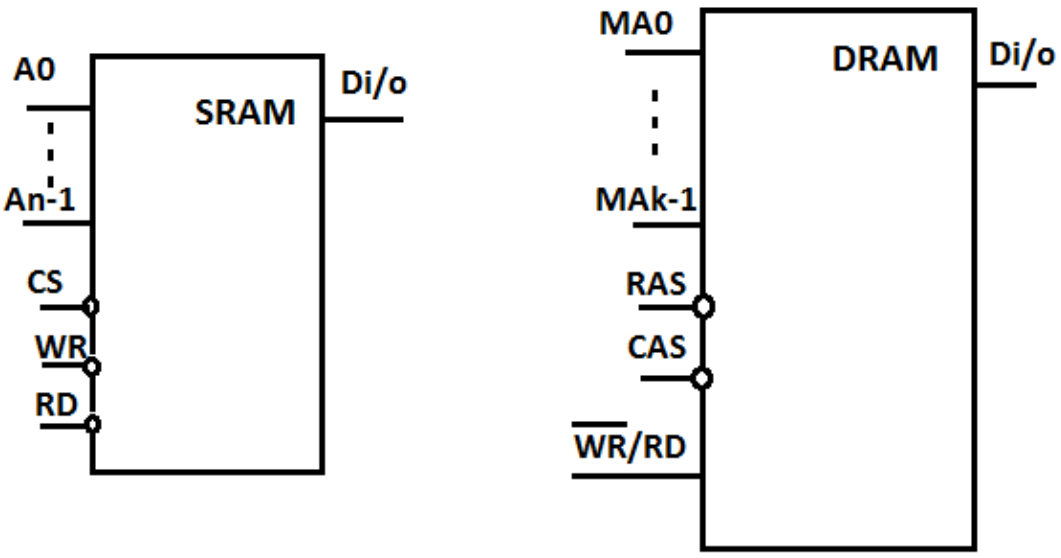


Адресная память.

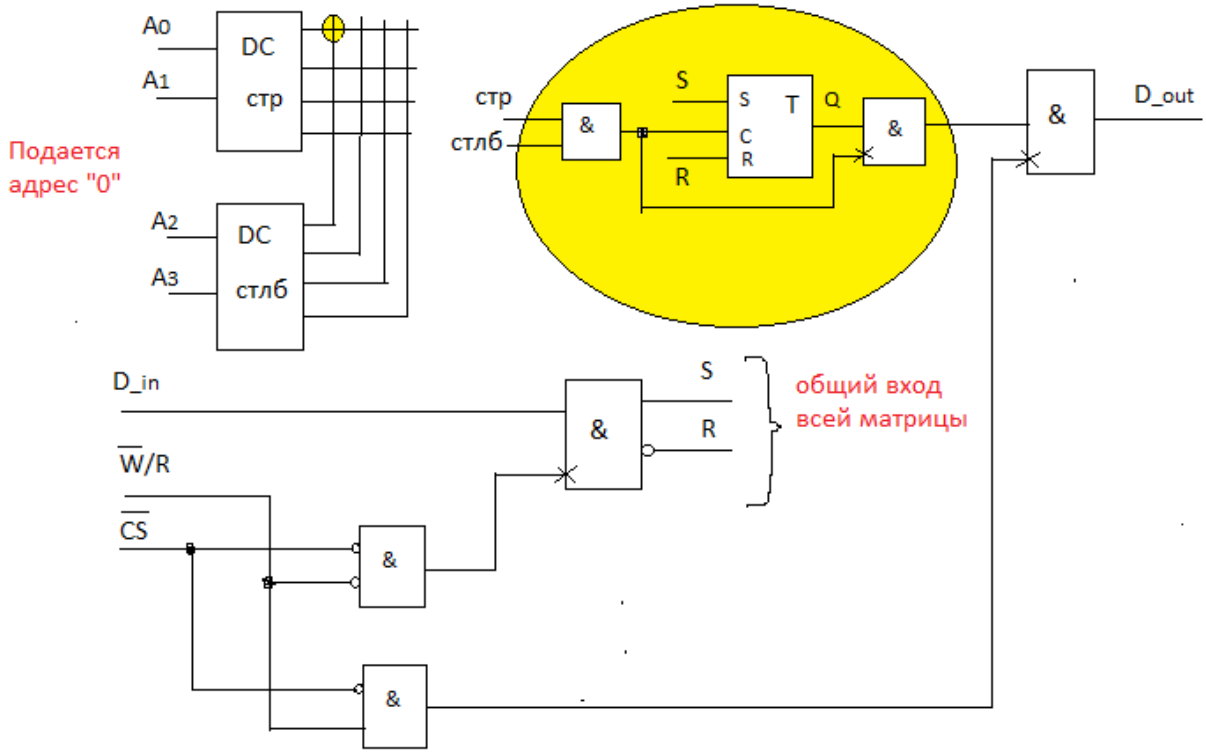
Примеры адресной памяти: постоянные запоминающие устройства ROM и оперативные запоминающие устройства RAM.

Оперативные запоминающие устройства.

Оперативные запоминающие устройства делятся на 2 класса по структуре матриц накопителя. Это динамические ОЗУ (DRAM), имеющие ячейки накопителя емкостного типа, и статические ОЗУ (SRAM), в которых накопители строятся на основе триггерных ячеек. Обращение к ячейкам накопителя DRAM не может совершаться одновременно по строкам и столбцам из-за инерционности емкости. При считывании с DRAM информация стирается (емкость разряжается) и необходимо время на восстановление информации в ячейке. Кроме того, емкостным ячейкам, не участвующим в процессе обмена, необходима регенерация (поддержание заряда). Такая память обладает низким быстродействием, но имеет низкую себестоимость производства.



Память SRAM, имеющая триггерную матрицу накопителя, наоборот, обладает сверхвысоким быстродействием. Если рассматривать матрицу накопителя, имеющую 2^n ячеек, то в SRAM имеется n адресных входов, распределенных на дешифраторы строки и столбца, вход CS (chip select), позволяющий подключить именно данный кристалл к шине и входы управления буферами записи и считывания.

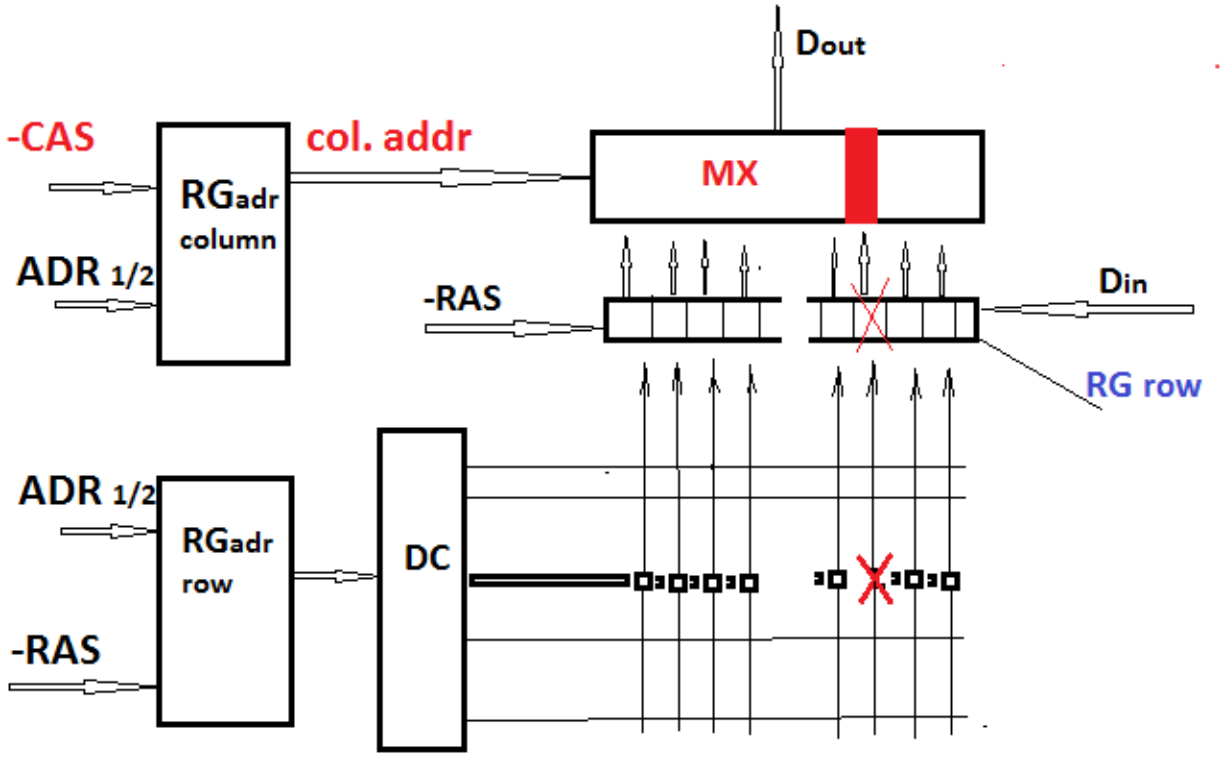


На рисунке изображена внутренняя структура схемы SRAM на 4 адресных входа, на которые подаем адрес «0»(0000). Пересечение строки и столбца матрицы накопителя, на которых при этом будут уровни «1» выделено желтым. Видно, что будет активна ячейка синхронного RS-триггера, на вход С которой поступит «1». В ячейку будет записана информация, поступившая на входы S и R всей матрицы. На рисунке видно, как работают входы CS(активный уровень «0») и W/R(для записи активный уровень «0», а для чтения «1»). Получаемые на выходах элементов 2И уровни «1» открывают буфер записи или буфер чтения. Считывание с ячейки накопителя также управляется с элемента 2И на линиях выходов дешифраторов строк и столбцов.

Лекция 8.

Оперативные запоминающие устройства. DRAM.

В DRAM адресные входы подключаются к шине через мультиплексоры, выделяющие адреса строки и столбца. K=n/2. Вход RAS (row address strobe) (строб строки) активен, когда подается адрес строки, вход CAS (column address strobe) (строб столбца) активен после подачи адреса столбца.



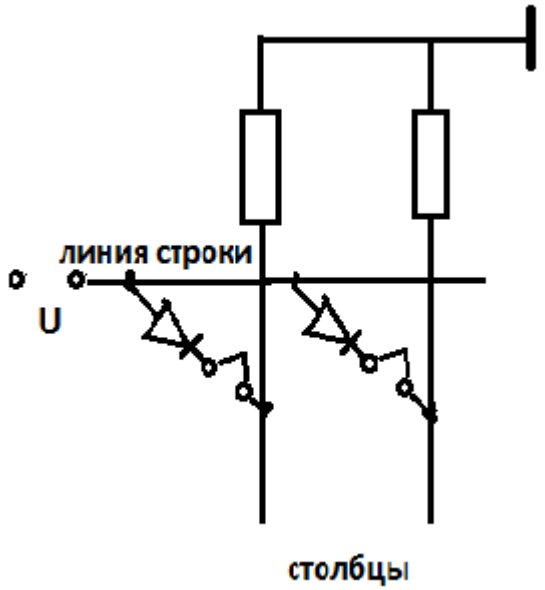
На рисунке изображен процесс поиска ячейки и считывания информации.

При записи строки в защелку строки информация в ячейках разрушается, поэтому во время выдачи информации одновременно идет восстановление строки из защелки. Строки, не выбранные дешифратором при активном RAS, подвергаются регенерации.

Постоянные запоминающие устройства.

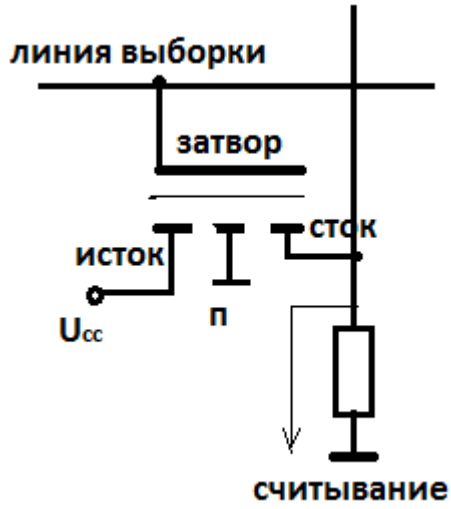
В матрице накопителя постоянного запоминающего устройства строки подключены к выходам дешифратора адреса, а столбцы – к шине данных.

Постоянные запоминающие устройства, имеющие диодные накопители, могут программироваться однократно (ROM, PROM).



Фрагмент матрицы накопителя PROM. В исходном состоянии все узлы в состоянии «1».

Постоянные запоминающие устройства, матрицы накопителей которых построены на полевых транзисторах, возможно многократно перепрограммировать. Это могут быть схемы с ультрафиолетовым стиранием информации (EPROM), или с электрическим стиранием (EEPROM).



Узел матрицы накопителя на основе полевого транзистора.

Основа – МОП-структура с плавающим или двойным затвором.

Плавающий затвор: между затвором и каналом вводится дополнительная область, вызывающая стекание в нее заряда. При снятии напряжения с затвора заряд сохраняется и удерживает транзистор в запертом состоянии. Пороговое напряжение настолько велико, что поле не создается.

В настоящее время ультрафиолетовое стирание практически не применяется, термин EPROM используют для схем на основе плавающего затвора, которые допускают только полное стирание информации, и имеют значительно меньший ресурс для перезаписи, чем схемы на основе двойного затвора.

CPU. Производительность процессора. Архитектура процессоров.

Если процессор работает с тактовой частоой F, то время T=1/F называется тактом. Время выполнения тестовой задачи можно рассчитать через такт

T x C x I,

где C – количество тактов на инструкцию, а I – количество инструкций на задачу.

Соответственно, чем меньше времени затрачивается на решение тестовой задачи, тем производительность процессора выше. В указанном выше выражении уменьшение T ограничено свойствами структуры, поэтому изменение производительности можно достичь изменением I или C.

Рассмотрим две основные архитектуры процессорного ядра. RISC – процессоры (Reduced Instruction Set Computer) и CISC - процессоры (Complete Instruction Set Computer).

Любой тип процессора выполняет инструкции, непрерывным потоком поступающие из памяти по шине данных. Выполнение инструкции можно разбить на 5 этапов:

1 – выборка кода из памяти по выставленному на адресной шине адресу,

2 – дешифрация кода,

3 – исполнение,

4 – получение результата,

5 – обратная загрузка результата.

Для ускорения процесса работа производится **конвейерным** способом, т.е. в каждый момент времени одновременно выполняются разные этапы следующих подряд команд.

Рассмотренный выше случай – пятиступенчатый конвейер, но для разных процессоров возможно объединение 4 и 5 или 3, 4 и 5 этапов, в этих случаях мы имеем четырех- или трехступенчатый конвейер.

Для CISC – процессоров характерны сложные многотактовые инструкции, производители этих процессоров старались увеличить производительность засчет уменьшения I. Но это приводило к приостановке конвейера, а, следовательно, снова снижало производительность процессора.

RISC – процессоры выполняют простые однотактовые операции. Они, в отличие от CISC не могут выполнять сложные задачи, зато для них C = 1,а так как операции обмена с пространством памяти в RISC выделены в отдельную группу, конвейер работает практически безостановочно и производительность высока.

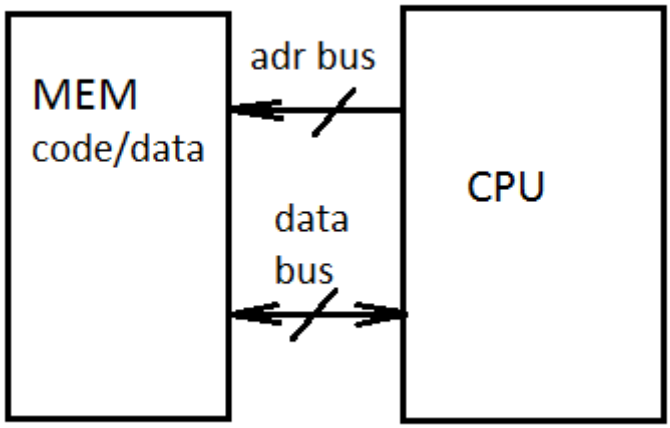
В настоящее время классические структуры в их первоначальном виде уже не используются. Процессорные системы строятся, в основном, на основе модифицированных RISC-ядер.

Лекция 9.

Типы архитектуры микропроцессорных систем.

В настоящее время существуют два типа архитектуры микропроцессорных систем – Принстонская, или архитектура фон-Неймана и Гарвардская.

В 1945 г. американский математик Джон фон Нейман сформулировал основные принципы работы современных компьютеров. Им была предложена архитектура, получившая его имя (von Neumann architecture) и предполагающая хранение программ и данных в общей памяти (1946 г.).

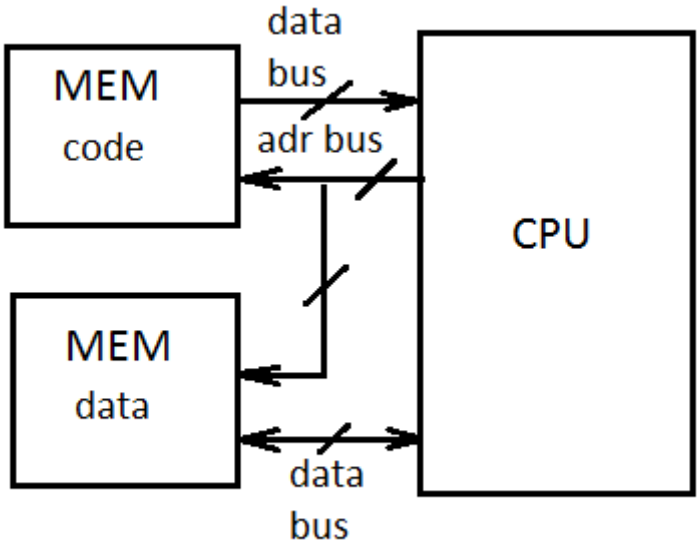


Такая архитектура наиболее характерна для микропроцессоров, ориентированных на использование в компьютерах. Примером могут служить микропроцессоры семейства x86. Эти микропроцессоры относятся к CISC-процессорам.

Структура ядра CISC-процессора предполагает наличие малого количества регистров общего назначения, к тому же имеющих строго определенные функции. Эти функции обусловлены наличием большого количества указателей и счетчиков, входящих в состав ядра и позволяющих выполнять циклические операции, записанные в одной инструкции. При выполнении операций в АЛУ процессор может пользоваться операндами как хранящи-

мися в регистрах общего назначения, так и в пространстве памяти, выделенном под данные. Таким образом, для написания программ под CISC=процессор используется большее количество адресаций данных, чем при программировании под RISC-процессор. Команды CISC-процессора выполняются за несколько тактов, что позволяет не выстраивать коммутационные КЦУ в цепочку, а использовать один коммутатор и служебный регистр (например, в командах пересылок).

Архитектура, предполагающая раздельное использование памяти программ и данных, носит название гарвардской (Harvard architecture). Гарвардская архитектура позволяет центральному процессору работать одновременно как с памятью программ, так и с памятью данных, что существенно увеличивает производительность.



Такая архитектура была создана под использование RISC- процессоров.

Работа микропроцессорной системы происходит под действием команд языка низкого уровня (Ассемблера). Структура команды представлена адресами блоков, задействованных в операции, адресами источников и приемников информации. Кроме того, в команде могут содержаться операнды. Операнды могут быть данными, необходимыми для загрузки в регистр, или адресами (прямыми, или смещениями).

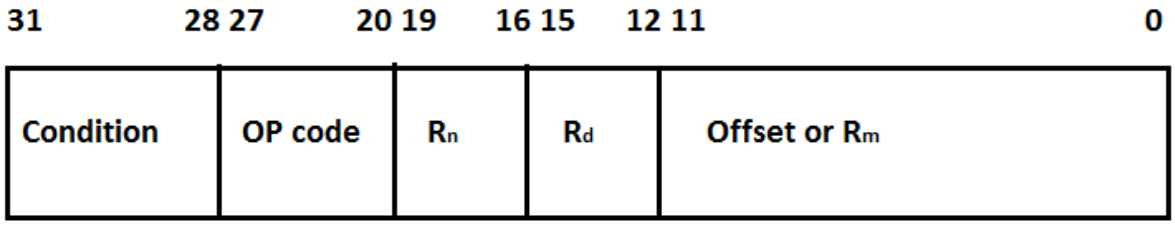
Любой микропроцессор состоит из нескольких блоков. Условно их можно назвать:

1. – блок очереди команд,
2. – блок дешифрации команд,
3. – блок регистров (внутренняя память).

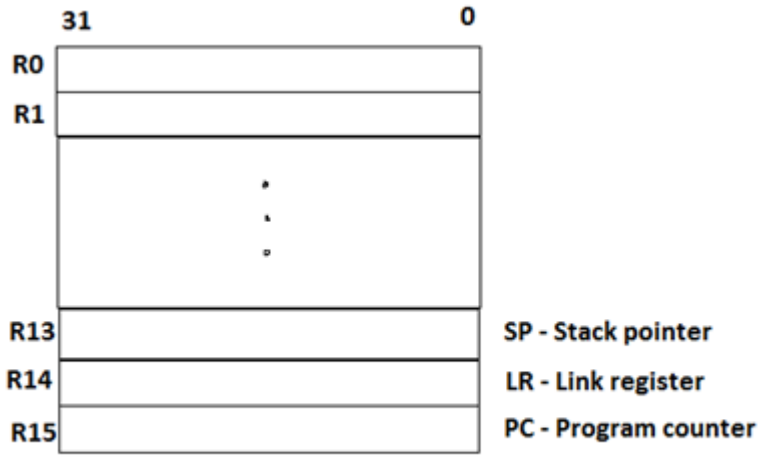
4. – блок производства операций, включающий: АЛУ, умножитель, делитель, сдвиговый регистр.

Как мы уже рассматривали, из блока памяти команды поступают в процессор непрерывным потоком. На входе в процессор с шины данных команды записываются в регистр очереди. Такой регистр содержит 3 команды одновременно, такой формат является оптимальным, так как программы обычно построены по разветвленным алгоритмам.

При считывании с регистра очереди команды дешифрируются, т.е. адресная информация, содержащаяся в каждом поле формата, поступает на свой дешифратор для выработки сигналов управления.



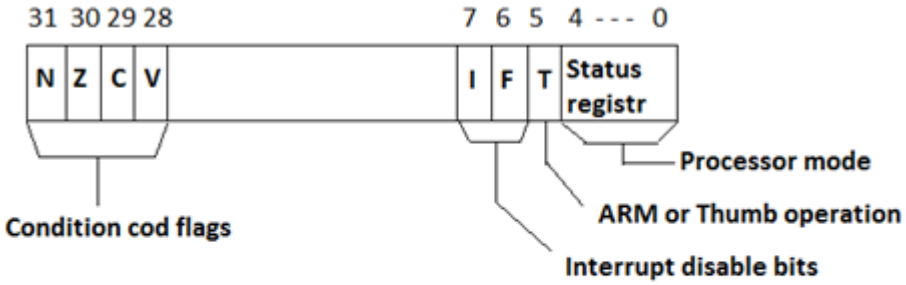
Структура ядра RISC-процессора предполагает наличие большой внутренней памяти, состоящей из регистров общего назначения, не имеющих дополнительных специальных функций. В некоторых семействах микропроцессоров исключение составляет так называемый «регистр нуля». Этот регистр не предназначен для записи информации, в нем хранится «0». Кроме того, в общее число регистров общего назначения входят указатель стека указатель связей и программный счетчик. В этих регистрах всегда записываются адреса: в указатель стека – адрес вершины стека в исполняемой программе, а в указатель связей – адрес выхода из основной программы в вызванную область для возможности возврата.



С помощью команд прямой и обратной загрузки происходит обмен между регистрами общего назначения и памятью данных.

АЛУ – арифметико-логическое устройство. Содержит сумматор и простую комбинаторную логику. Связано с регистром флагов.

Регистр флагов ARM-процессора



CPSR - Current Program Status Register

Различают флаги состояний и системные флаги.

Флаги состояний характеризуют состояние результата операции.

N(S) – знак.

Z – ноль,

C – перенос,

V – переполнение (overflow).

Флаги состояний устанавливаются командами, выполняемыми в операционном блоке (АЛУ), но только теми, которые для этого определены.

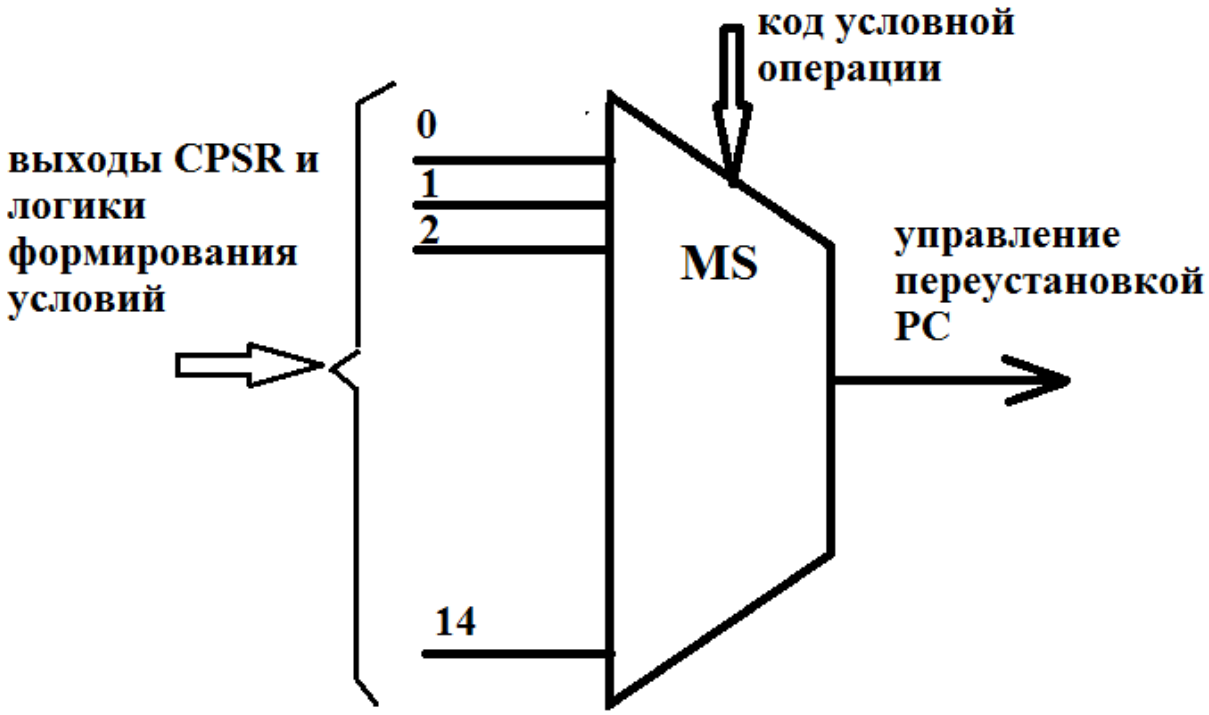
Лекция 10.

Флаги состояний используются командами условных действий.

Таблица условий.

Код условия	Обозначение условия	Наименование условия	Комбинация флагов
0	EQ	Равно (ноль)	Z = 1
1	NE	Не равно (не ноль)	Z = 0
2	CS/HS	Перенос (беззнаковое больше или такое же)	C = 1
3	CC/LO	Нет переноса (беззнаковое меньше)	C = 0
4	MI	Знак минус	N = 1
5	PL	Знак плюс	N = 0
6	VS	Переполнение	V = 1
7	VC	Нет переполнения	V = 0
8	HI	Беззнаковое больше	nC v Z = 0
9	LS	Беззнаковое меньше или такое же	nC v Z = 1
10	GE	Знаковое больше, чем или равно	N + V = 0
11	LT	Знаковое меньше, чем	N + V = 1
12	GT	Знаковое больше, чем	Z v (N + V) = 0
13	LE	Знаковое меньше, чем или равно	Z v (N + V) = 1
14	AL	Все условия	

Очевидно, что выходы регистра флагов (условий) через логические цепочки подключаются к входам мультиплексора, а команда условного действия в своем коде содержит адрес необходимого входа.



При работе команд, использующих АЛУ (арифметико-логическое устройство), операнды поступают только из внутренней памяти ядра, что экономит время обработки. Команды RISC-процессора, в основном, выполняются за один такт, все КЦУ, участвующие в процессе обработки, как коммутирующие, так и кодопреобразующие образуют цепочки и имеют общую задержку, не превышающую время такта. Команды пересылок между регистрами выполняются как арифметическая операция сложения с 0.

Системные флаги описывают характеристики системы. Это может быть режим работы системы, возможность совершения прерывания, или указание формата используемых команд.

Разряды 7 и 6 CPSR указывают на возможность прерываний от внешних источников (IRQ – по обычным линиям запроса, FIQ – по скоростным).

Установка «1» при невозможности обслуживания запроса.

5-й разряд указывает, какое кодирование применяется в исполняемых операциях. Возможно ARM-кодирование с 32-разрядными инструкциями и

Thumb – кодирование с 16-разрядными. Наибольшее применение получил синтаксис Thumb-2. В нем смешанный формат инструкций.

Код, записанный в младших разрядах, определяет способ функционирования процессорной системы.

	$CPRS_{4-0}$	Operating Mode
1.	10000	User
7.	10001	FIQ
6.	10010	IRQ
3.	10011	Supervisor
4.	10111	Abort
5.	11011	Undefined
2.	11111	System

1. User – основной способ работы для прикладных программ. Способ непривилегированный, имеет ограниченный обмен с системными ресурсами.
2. System – открывает полный доступ к системным ресурсам.
3. Supervisor – включается, когда программное прерывание делает невозможным дальнейшее исполнение программы (аварийное завершение). Кроме того, включается при сбросе или выключении питания.
4. Abort – включается при попытке программы осуществить обмен с запрещенным участком памяти.
5. Undefined – включается при заявлении несуществующей инструкции.
6. IRQ – режим реагирования на обычные запросы на прерывания о внешних источников.
7. FIQ – режим реагирования на запросы от внешних устройств на прерывание по скоростным линиям запросов. Используется для обслуживания запросов, требующих немедленного обслуживания.

Прерывание – это процесс, позволяющий приостановить выполнение основной программы на время выполнения подпрограммы.

Прерывания: аппаратные, программные и исключительные ситуации. Исключительные ситуации также можно отнести к программным прерываниям, но они происходят на этапе компиляции или отладки.

Первые действия процессора при поступлении сигнала о прерывании:

PC ➔LR (banked);

CPSR ➔SPSR (banked);

Changes bits 4—0 of CPSR, sets **I** and **F**;

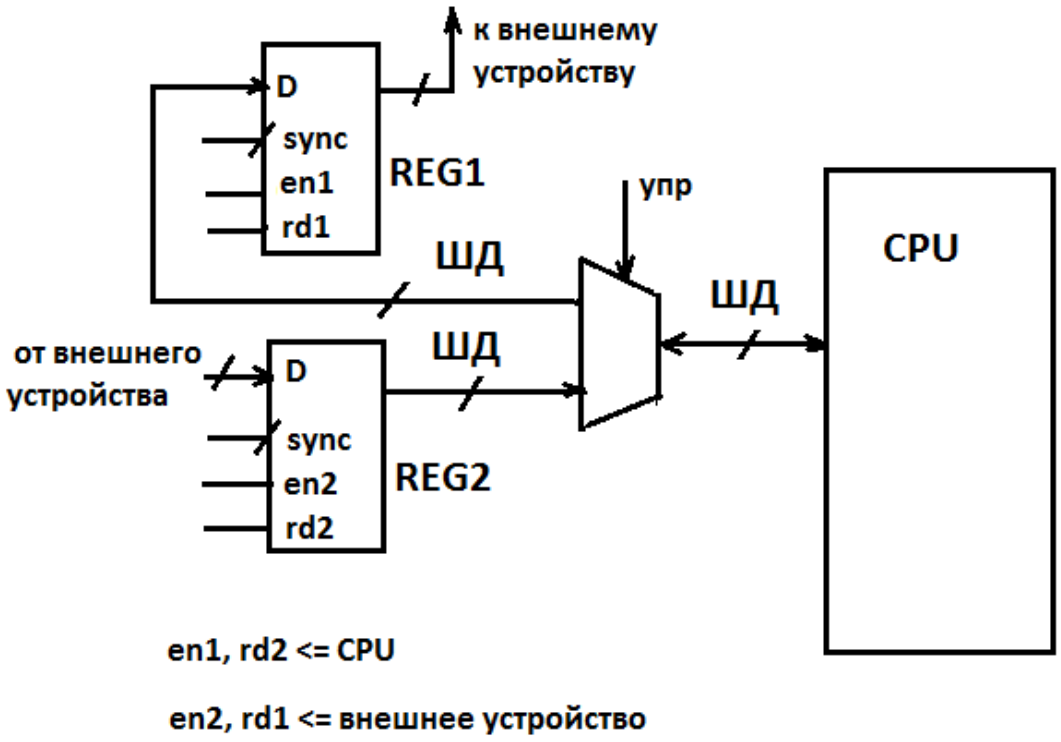
Vector address ➔ PC.

Таблица векторов прерываний

Address	Exception	Priority	Mode entered
0x000	Reset	1	Supervisor
0x004	Unimplemented instruction	6	Undefined
0x008	Software interrupt	-	Supervisor
0x00C	Instruction access violation	5	Abort
0x010	Data access violation	2	Abort
0x018	IRQ	4	IRQ
0x01C	FIQ	3	FIQ

Устройства ввода-вывода.

Основной принцип ввода и вывода информации основан на взаимодействии регистров.



Если внутри МПС передача информации производится параллельным способом, то все внешние передачи осуществляются последовательно.

Основные последовательные интерфейсы.

UART (USART), USB – высокоскоростные,

SPI, I2C - низкоскоростные.

Лекция 11.

UART (прототип RS232).

Передача данных в UART осуществляется по одному биту в равные промежутки времени. Этот временной промежуток определяется заданной **скоростью UART** и для конкретного соединения указывается в бодах (что в данном случае соответствует битам в секунду). Существует общепринятый ряд стандартных скоростей: 300; 600; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200; 230400; 460800; 921600 бод.

Основные регистры данных: регистр приема и регистр передачи.

Кроме регистров приема и передачи имеет два адресуемых делителя частоты (старший и младший байты адресуются отдельно), регистры управле-

ния линией и модемом, регистры состояния линии и модема, регистр разрешения прерываний и регистр-идентификатор прерываний.

Делители служат для хранения констант, изменяющих коэффициент деления тактовой частоты, чтобы обеспечить определенную скорость передачи.

Перед началом работы необходимо записать управляющее слово по адресу регистра управления линией. В формате управляющего слова определяется:

1) - доступ к регистрам приема/передачи или к регистрам выбора скорости;

2) - нормальная передача символов или старт (рассоединение);

3) – наличие контроля и тип контроля (паритет, непаритет);

4) - количество стоповых бит;

5) - количество разрядов в символе.

1. Первое управляющее слово всегда имеет в старшем разряде «1», что дает обращение к регистрам-делителям. После установки необходимой скорости записывается управляющее слово, имеющее в старшем разряде «0». Оно и будет определять регламент работы приема и передачи.
2. В следующем разряде записывается «1 » при рассоединении и ожидании старта. В штатном режиме записывается «0».
3. Далее три разряда (5,4 и 3) служат для определения контроля. Если в р3 записан «0», то контроль отсутствует, и состояние остальных разрядов не имеет смысла. Если в р3 записана «1», то значение р5 и р4 будут обозначать следующее:

P5	P4	Тип контроля
0	0	Дополнение до чета
0	1	Дополнение до нечета
1	0	Контроль четности
1	1	Контроль нечетности

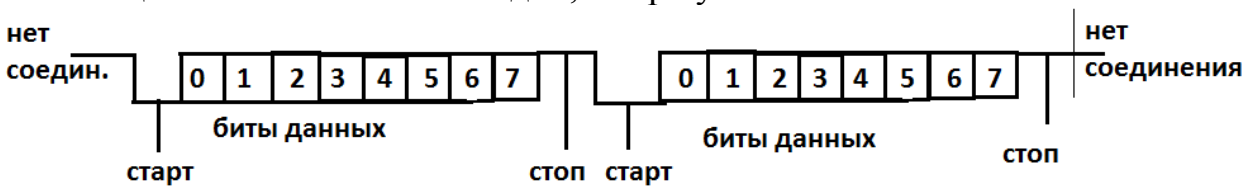
Многие реализации UART имеют возможность автоматически контролировать целостность данных методом контроля битовой чётности. Когда эта функция включена, последний бит данных в минимальной посылке («бит чётности») контролируется логикой UART и содержит информацию о чётности количества единичных бит в этой минимальной посылке. Различают контроль на четность (англ. *Even parity*), когда сумма количества единичных бит в посылке является четным числом, и контроль на нечетность

(англ. *Odd parity*), когда эта сумма нечетна. При приеме такой посылки UART может автоматически контролировать бит четности и выставлять соответствующие признаки правильного или ошибочного приема.

Контроль чётности (пример для 7-битной посылки)

данные	**количество единичных бит**	**Бит четности**	
		even	**odd**
0000000	**0**	**0**	**1**
1010001	**3**	**1**	**0**
1101001	**4**	**0**	**1**
1111111	**7**	**1**	**0**

4. Следующий разряд содержит информацию о количестве стоповых бит. Чаще всего стоповый бит один, т.е. p2 устанавливается в «0».



Помимо собственно информационного потока, UART автоматически вставляет в поток синхронизирующие метки, так называемые **стартовый и стоповый биты**. При приёме эти лишние биты удаляются из потока. Обычно стартовый и стоповый биты обрамляют один байт информации (8 бит), однако встречаются реализации UART, которые позволяют передавать по 5, 6, 7, 8 или 9 бит. Обрамленные стартом и стопом биты являются минимальной посылкой. Некоторые реализации UART позволяют вставлять два стоповых бита при передаче для уменьшения вероятности рассинхронизации приёмника и передатчика при плотном трафике. Приёмник игнорирует второй стоповый бит, воспринимая его как короткую паузу на линии.

Принято соглашение, что пассивным (в отсутствие потока данных) состоянием входа и выхода UART является логическая 1. Стартовый бит всегда логический 0, поэтому приёмник UART ждёт перепада из 1 в 0 и отсчитывает от него временной промежуток в половину длительности бита (середина передачи стартового бита). Если в этот момент на входе всё ещё 0, то запускается процесс приёма минимальной посылки. Для этого приёмник отсчитывает 9 битовых длительностей подряд (для 8-битных данных) и в каждый момент фиксирует состояние входа. Первые 8 значений являются принятыми данными, последнее значение проверочное (стоп-бит). Значение стоп-бита всегда 1, если реально принятое значение иное, UART фиксирует ошибку.

5. Разряды 0 и 1 определяют количество разрядов в символе (от 5 до 8).

Поскольку синхронизирующие биты занимают часть битового потока, то результирующая **пропускная способность** UART не равна скорости соединения. Например, для 8-битных посылок формата 8-N-1 синхронизирующие биты занимают 20 % потока, что для физической скорости 115 200 бод даёт битовую скорость данных 92 160 бит/с или 11 520 байт/с.

В регистре состояния сообщается о заполненности регистров приема и передачи, об ошибках приема, о прерывании отсоединения, о переполнении регистров приема и передачи.

Регистр разрешения прерывания позволяет определить источник разрешенного прерывания (модем или линия); разрешение, если регистр передачи пуст и разрешение, если получаемые данные не имеют доступа к регистру приема.

Интерфейс *USB* (Universal Serial Bus - Универсальный Последовательный Интерфейс)

Предназначен для подключения периферийных устройств к персональному компьютеру. Позволяет производить обмен информацией с периферийными устройствами на трех скоростях (спецификация *USB 2.0*):

- Низкая скорость (*Low Speed* - LS) - 1,5 Мбит/с;
- Полная скорость (*Full Speed* - FS) - 12 Мбит/с;
- Высокая скорость (*High Speed* - HS) - 480 Мбит/с.

Для подключения периферийных устройств используется 4-жильный кабель: питание +5 В, сигнальные провода *D*+ и *D*-, общий провод.

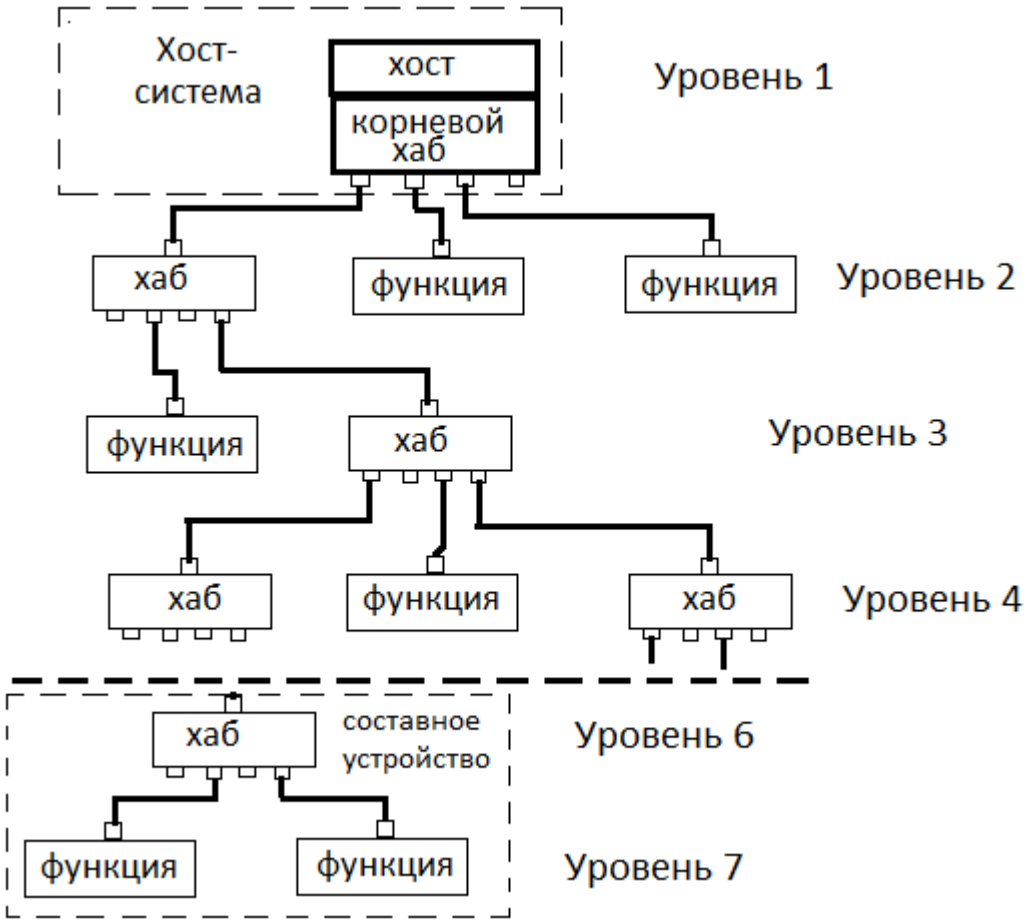
Структура USB

Хост – всегда основное ведущее устройство. (Персональный компьютер.)

Хаб – концентратор. Содержит контроллер и повторитель.

Регистры хаба-контроллера связываются с хостом и обеспечивают опознавание функции, и ее связь с хостом. Хаб-повторитель обеспечивает энергетический режим работы шины.

Функция – оконечное устройство, подключаемое к хосту. Всегда ведомое.



Порт хаба или функции, подключаемый к хабу более высокого уровня, называется восходящим портом (upstream port), а порт хаба, подключаемый к хабу более низкого уровня, или к функции, называется нисходящим портом(downstream port).

Все передачи данных по интерфейсу инициируются хостом, однако приемником или передатчиком может быть как хост, так и функция. Передача пакетная.

В интерфейсе USB испольуется несколько разновидностей пакетов:

- ***пакет-признак*** (*token paket*) описывает тип и направление передачи данных, адрес устройства и порядковый номер конечной точки (КТ - адресуемая часть USB-устройства).
- ***пакет с данными*** (*data packet*) содержит передаваемые данные;
- ***пакет согласования*** (*handshake packet*) предназначен для сообщения о результатах пересылки данных

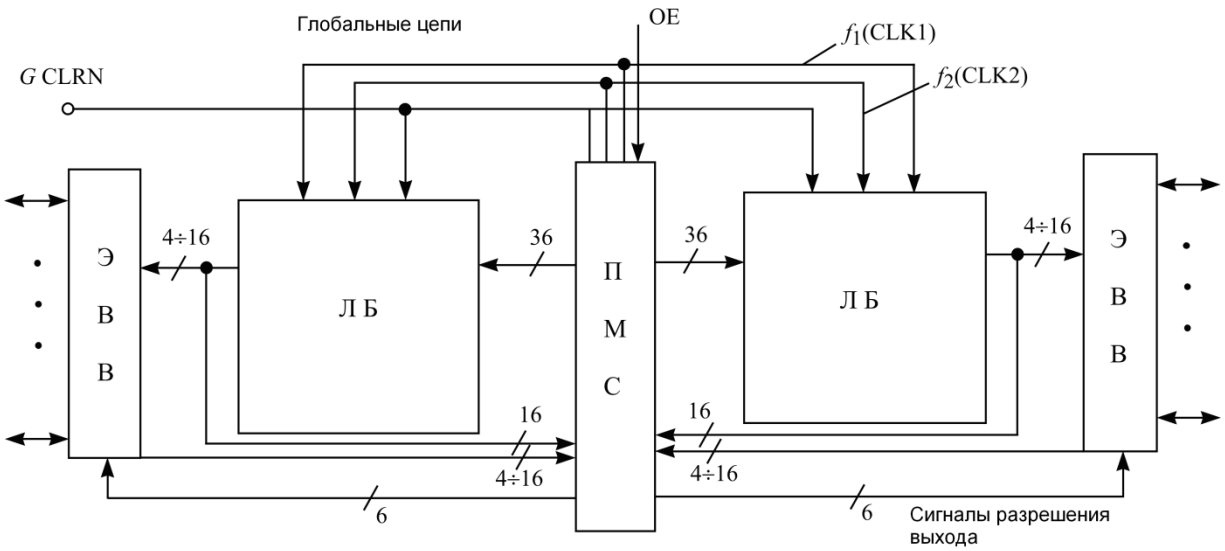
Лекция 12.

Программируемые логические интегральные схемы (ПЛИС).

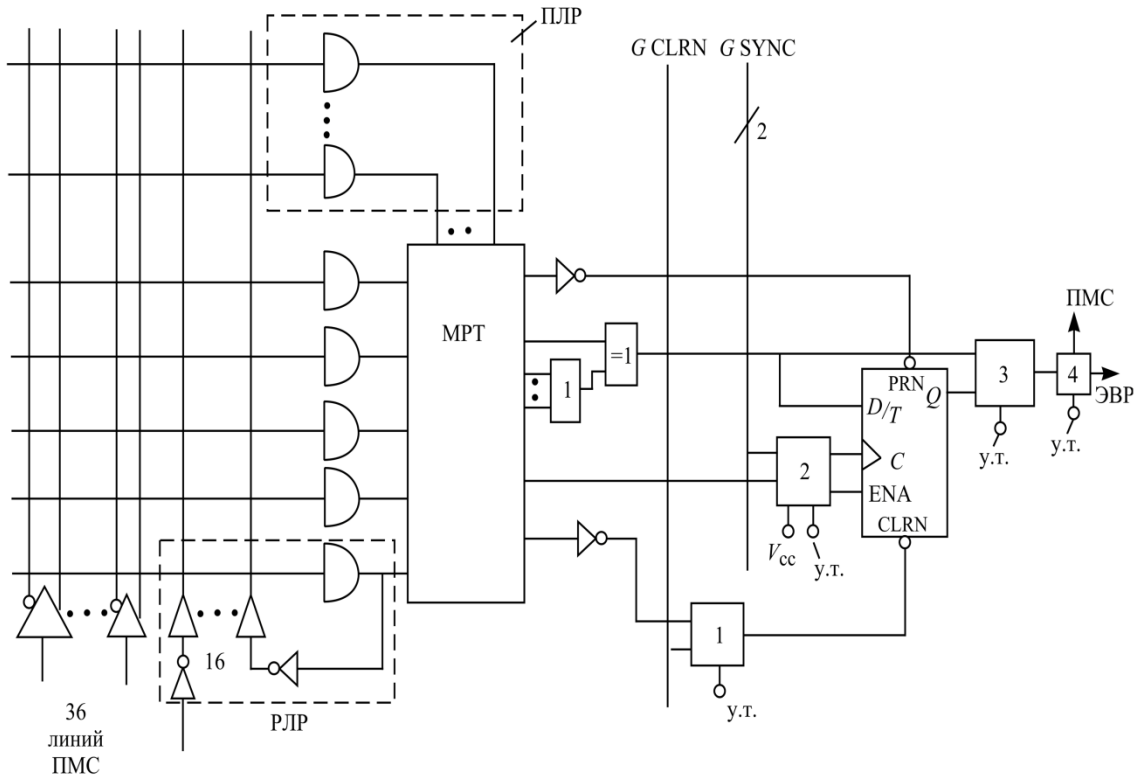
Как уже отмечалось ранее, существует тип кристалла, в котором все связи могут быть запрограммированы пользователем. Приведем общие сведения о структурах таких кристаллов. Структура любого кристалла базируется на p-n переходе, но, соответственно возможностям описания цифрового устройства, эти переходы могут группироваться для построения матриц логических элементов (И – ИЛИ), или же для построения матриц простейших таблиц функционирования. Таким образом, ПЛИС делятся на два различных класса. Конструктивно ПЛИС состоит из внешней части, содержащей буферные и различные адаптирующие элементы, и внутренней части, состоящей из логических блоков, системы межсоединений этих блоков и элементов памяти конфигурации. Рассмотрим структуру внутренней части для каждого из классов ПЛИС.

ПЛИС семейства CPLD.

Класс CPLD имеет структуру логического блока, представленную устройством ПМЛ (программируемой матричной логики) с параметрами для кристаллов фирмы Altera 36x80x16. Т.е. блок содержит 36 входов, 80 термов (элементов И) и 16 выходов (элементов ИЛИ). Матрица элементов «И» полнодоступна (несвязана), каждый вход может входить в каждый элемент «И».



Матрица «ИЛИ» для ПМЛ связана, поэтому логический блок состоит из 16 макроячеек, в каждой из которых в элемент «ИЛИ» возможно подключение 5 термов. Для увеличения количества термов, включаемых в «ИЛИ», в структуре блока содержится параллельный логический расширитель, для расширения состава терма служит разделяемый логический расширитель, подключающий инверсный выход 5 терма каждой макроячейки для доступа всем термам своего логического блока. Для сохранения информации предыдущего такта в состав макроячейки входит триггер.



Система межсоединений CPLD представлена программируемой матрицей соединений, позволяющей соединить любую макроячейку с кристалла с другой, в котором из логических блоков она бы ни находилась. Такая структура строится по принципу программируемой логической матрицы (ПЛМ), матрица ИЛИ в которой полнодоступна. Система межсоединений, построенная на основе гибкой логики, позволяет предсказать задержки в схеме.

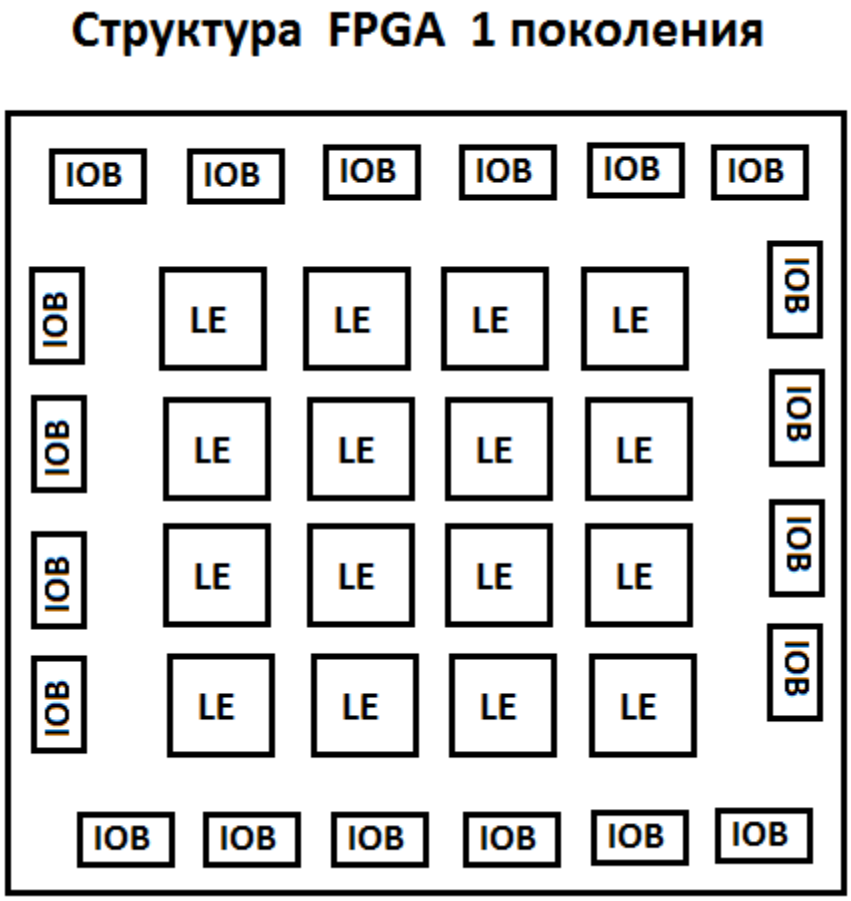
Память конфигурации CPLD распределенная, построена на базе EEPROM. Это позволяет сохранять конфигурацию схемы в кристалле при выключенном питании.

Функцию, программируемую в кристалле CPLD, желательно описывать с помощью системы логических уравнений в канонической форме с минимизацией. При программировании использовать неразветвленные алгоритмы.

Как следует из вышеизложенного, в кристаллах рассмотренного класса возможно построение схем невысокой степени сложности (КЦУ, конечные автоматы). Единственным преимуществом таких кристаллов является их энергонезависимость.

ПЛИС семейства FPGA.

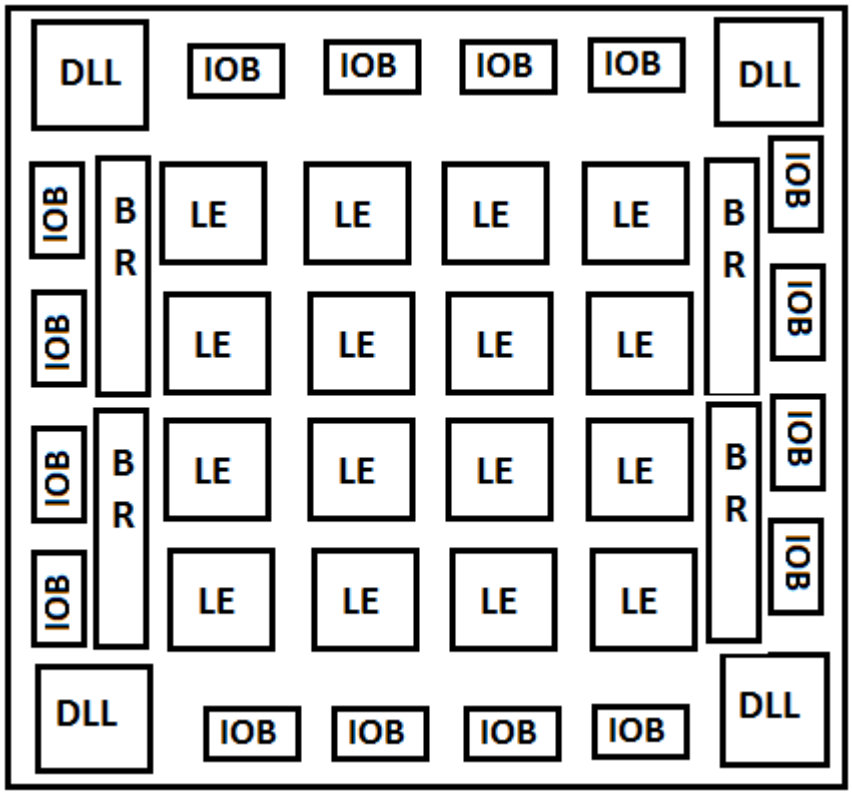
Основа архитектуры кристалла представленного класса – матрица логических блоков, соединяемых посредством выделенных линий (каналов).



Основа системы межсоединений – выделенные линии каналов. При такой системе невозможно предсказать путь соединения логических блоков, расположенных на поверхности кристалла, поэтому задержка в таких схемах непредсказуема. Непредсказуемость задержки в схемах, построенных на базе FPGA первых поколений, создавала значительные трудности при работе на высоких частотах.

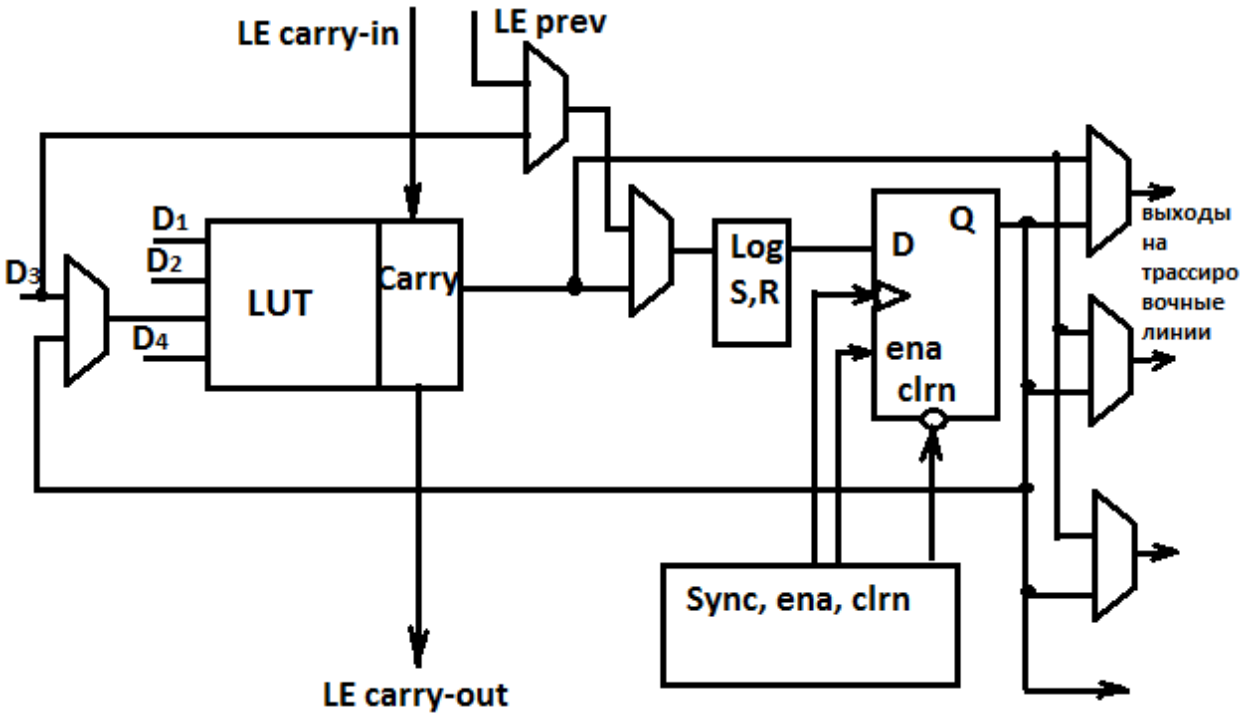
Для устранения подобных недостатков в FPGA последующих поколений (2, 3) стали вводить дополнительные логические связи – локальную матрицу соединений, позволяющую объединять отдельные ячейки, сформированные на основе LUT, в логический блок. Для компенсации фазовых сдвигов в структуру кристалла вошли блоки DLL (Delay locked loop), построенные на основе линий задержек. Логические блоки объединялись с помощью глобальной матрицы соединений канальной структуры.

FPGA 3 поколения



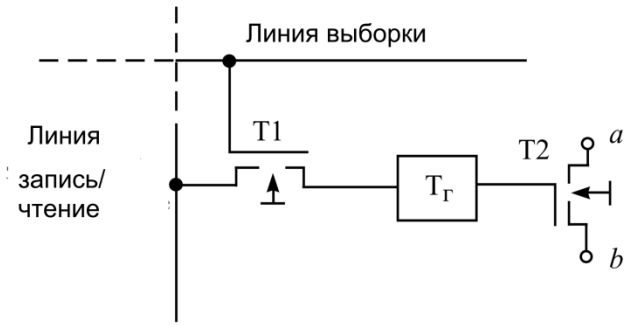
Логический блок строится на основе небольших блоков запоминающих устройств, LUT (Look-up-tables) на16 ячеек (4 входа). Для коммутаций в структуре блока присутствуют управляемые мультиплексоры, для сохранения состояния предыдущего такта – триггер.

Фрагмент структуры конфигурируемого логического блока (логического элемента).

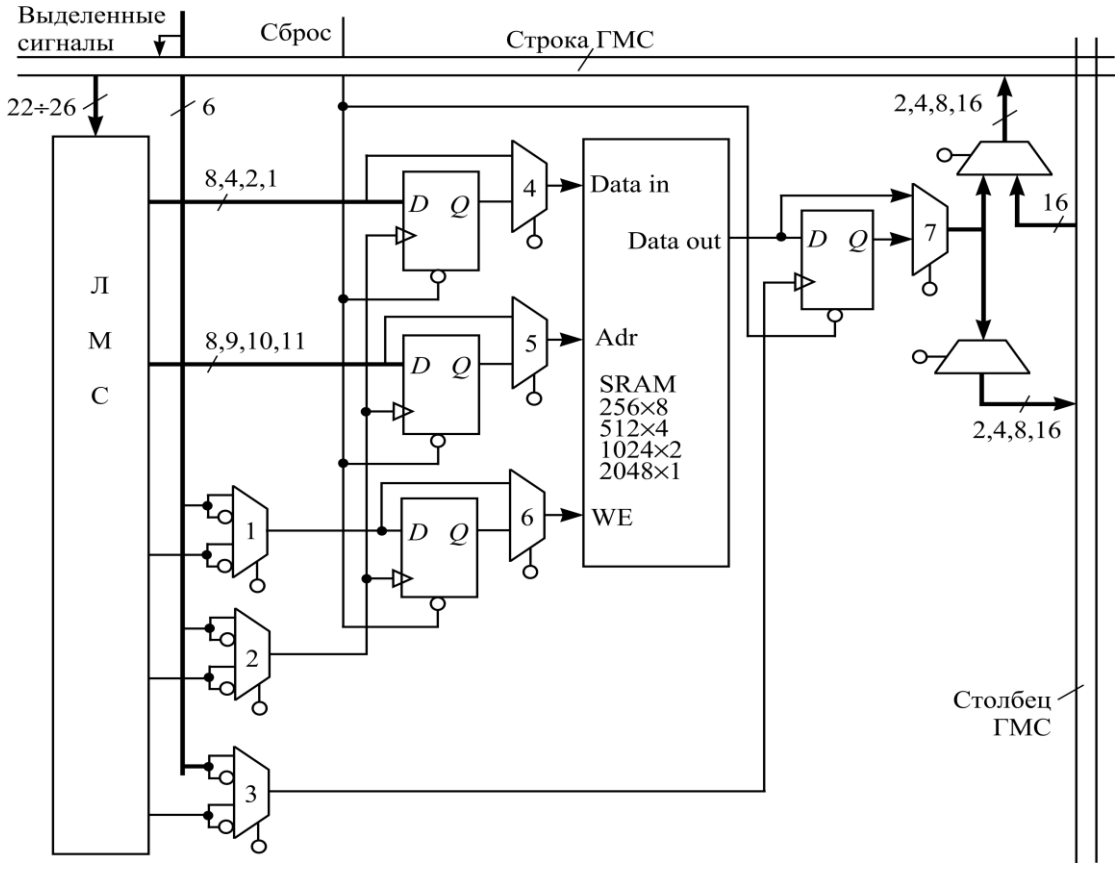


В FPGA 4 и 5 поколений архитектура основана на матрице мегаблоков, включающих в себя некоторое количество конфигурируемых логических блоков, объединенных посредством локальной матрицы соединений. Элементами такой матрицы служат правые части каждого логического блока, представленные простой логикой. Левую часть представляют LUT. Каждый мегаблок обслуживает DCM (Digital clock manager), позволяющий не только компенсировать задержки на глобальной матрице, но и изменять частоту и фазу сигнала. Кроме того, мегаблок содержит блок умножителя и встроенный блок памяти.

Память конфигурации FPGA строится на основе триггерных ячеек (SRAM) и представлена как распределенной (отдельные ячейки, обслуживающие точки связи на каналах)



так и выделенной памятью (LUT, встроенные блоки памяти). Встроенные блоки памяти, переконфигурируемые по длине ячейки, в основном, двухбуферные, небольшой общей емкости (от 2К в начальных вариантах, в дальнейших – не менее 4К).



не

Память конфигурации на ячейках SRAM не позволяет сохранять конфигурацию схемы в кристалле при отключенном питании. Для восстановления

конфигурации на плате к схеме присоединяют конфигурационную FLASH, где коды сохраняются при выключенном питании.

Программирование под FPGA всегда следует производить с использованием разветвленных алгоритмов, т.к. в таблицах все ячейки должны быть заполнены.

Лекция 13.

Протокол JTAG.

Полученный в результате компиляции ассемблерный код передается от компьютера в кристалл по протоколу JTAG.

Протокол JTAG создавался для тестирования, так как с ростом степени интеграции БИС, плотности монтажа и появлением многослойных печатных плат, методы диагностики, основанные на подключении к контрольным точкам платы и выводам микросхем, становились все более сложными в использовании и неэффективными. Программирование явилось частным случаем тестирования схемы внутри кристалла. Приведем краткую справку об используемом протоколе.

В начале 1985 года объединенными усилиями нескольких европейских компаний была создана группа для разработки решения проблем тестирования интегральных схем, цифровых устройств и систем. Эта группа получила имя: *Joint European Test Action Group (JETAG)*. Позднее, в 1988 году к ней присоединились представители североамериканских компаний, и название было изменено на *Joint Test Action Group (JTAG).*

Результатом работы этой группы явился принятый в 1990 году стандарт IEEE Std.1149.1 и его усовершенствованная версия: стандарт IEEE Std.1149.1a (1993).

Стандарт JTAG определяет:

- интерфейс, через который осуществляется обмен тестовыми инструкциями и данными между ведущим устройством и встроенными средствами тестирования (TAP – Test Access Port);

- минимальный набор средств тестирования, встраиваемых в БИС (средства поддержки метода граничного сканирования). В основу стандарта положена идея внедрения в компоненты цифрового устройства средств, обеспечивающих унифицированный подход к решению следующих задач:

тестирование связей между интегральными схемами после того, как они были смонтированы на печатной плате или другой основе;

наблюдение за работой компонент без вмешательства в их нормальную работу, или непосредственное управление одним или более компонентом;

обеспечение стандартизованного доступа к произвольным средствам самотестирования, встраиваемым в БИС.

Тестируемая плата с расположенными на ней БИС подключается через последовательный канал передачи данных (JTAG интерфейс) к некоторому ведущему устройству. Ведущее устройство, используя возможности, предоставляемые JTAG, решает задачи связанные с диагностикой тестируемого устройства, локализации неисправностей, загрузкой конфигураций PLD и т. п.

Как правило, ведущим устройством является персональный компьютер, оснащенный соответствующим программным обеспечением. Подключение к ведомому устройству осуществляется через параллельный или последовательный порт, или через плату расширения.

TAP требует 4-х внешних контактов:

- *TDI (Test Data Input)* – контакт для получения последовательных данных. На этот контакт последовательно, бит за битом подаются данные, которые затем интерпретируются схемой управления;

- *TDO (Test Data Output)* – контакт вывода последовательных данных. С этого контакта ведущее устройство последовательно считывает данные из БИС (например, результат тестовых операций);

- *TCK (Test Clock Input)* – контакт сигнала синхронизации обмена;

- *TMS (Test Mode Select)* – этот контакт управляет состоянием внутреннего автомата TAP. В частности, с помощью этого контакта определяется, что загружается: команда или данные, а также определяется начало и конец загрузки;

- *TRST (Test ReSeT)* – сброс в начальное состояние контроллера внутреннего автомата TAP (контакт не является обязательным для реализации).

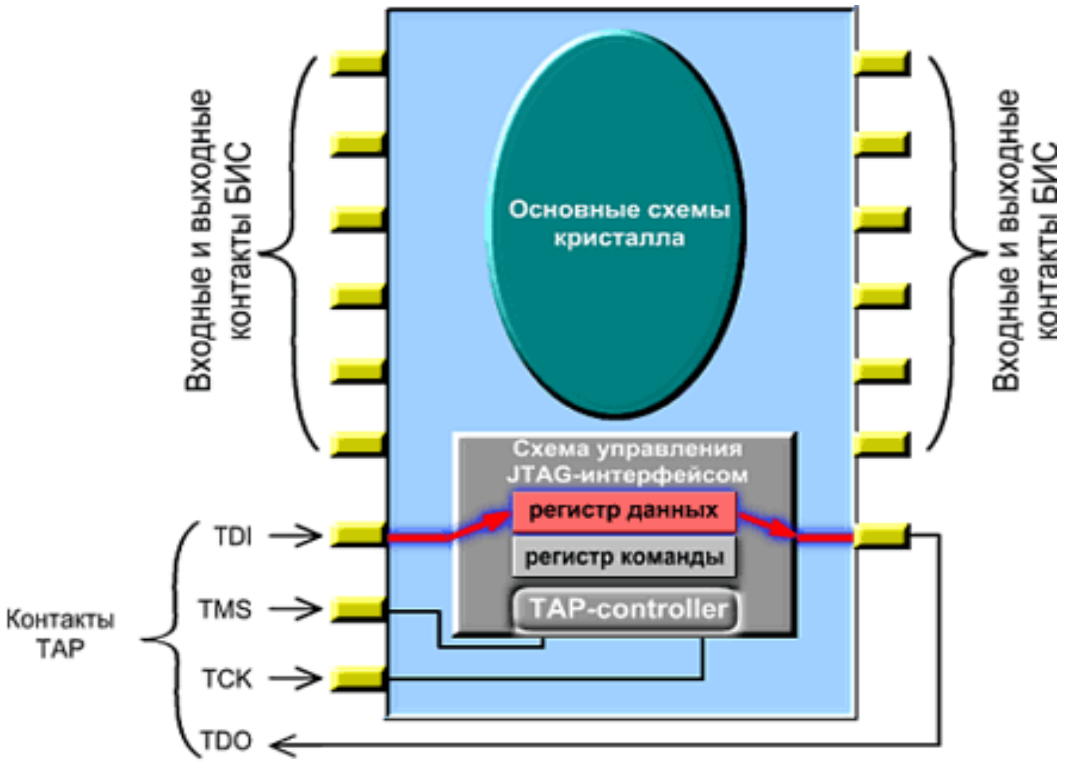


Рис. П1.2. Структура TAP

В процессе обмена информацией через TAP ведущее устройство воспринимает БИС как сдвиговый регистр, при этом
- *TDI* – вход сдвигового регистра;

- *TDO* – выход сдвигового регистра;

- *TCK* – сигнал сдвига;

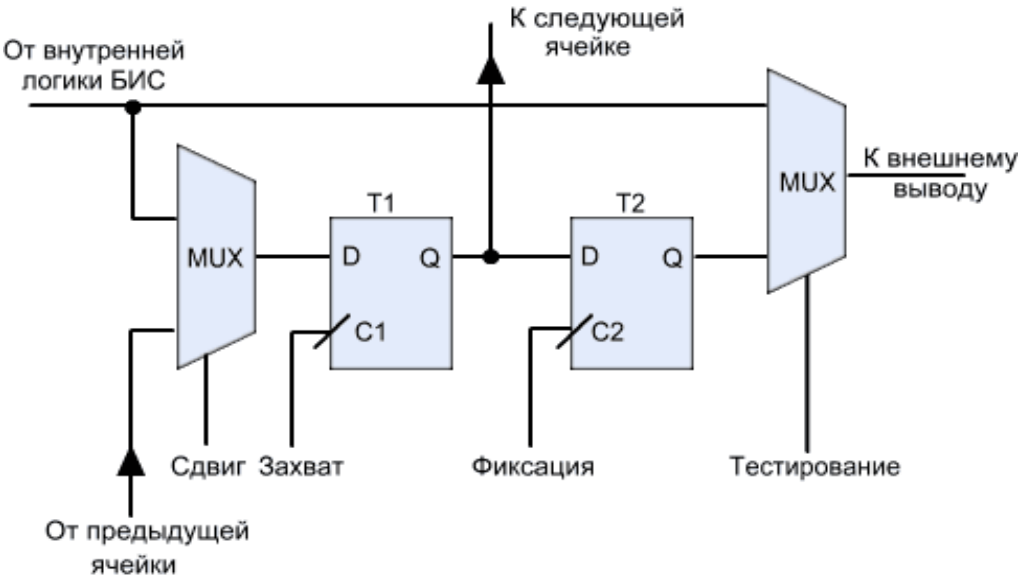
В зависимости от состояния автомата TAP-контроллера в канал может быть включен либо регистр данных, либо регистр команды.

Регистр команды в JTAG-контроллере всегда один.

Регистров данных в JTAG-контроллере может быть сколько угодно. Какой именно регистр данных будет выбран для подключения, как правило, определяется загруженной командой.
TAP-контроллер имеет граф из 16 состояний. Переход из одного состояния в другое зависит от сигнала TMS.

Регистр данных – сдвигающий регистр, состоящий из цепочки ячеек граничного сканирования – BSC (boundary scan cell).Такой регистр называется регистром граничного сканирования (Boundary Scan Register).

Вариант схемы отдельной ячейки

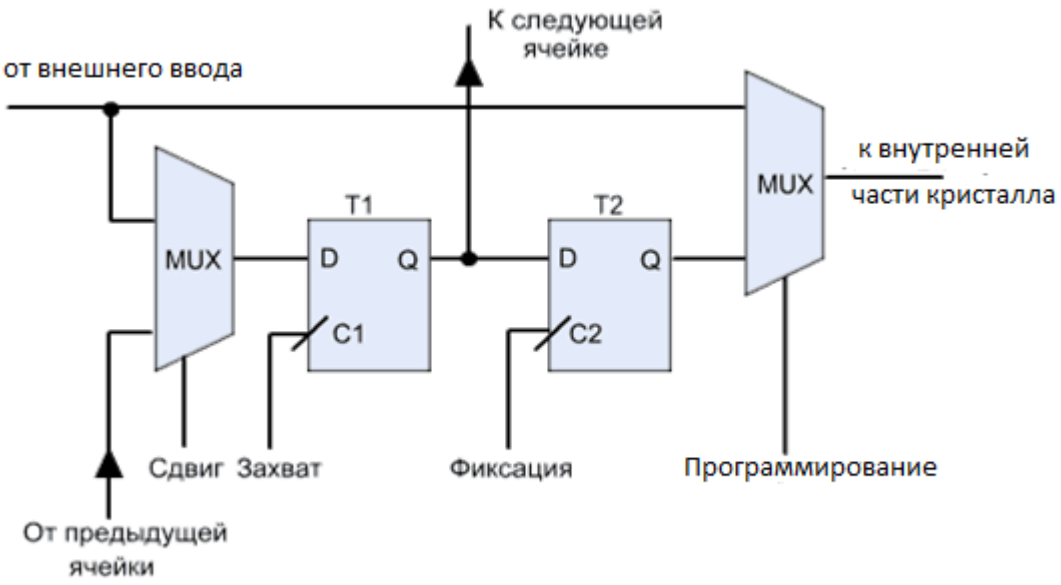


Можно выделить несколько режимов в работе ячейки:
- режим сдвига, когда в триггере Т1 по сигналу «захват» сохраняется состояние аналогичного триггера предыдущей ячейки. В этом режиме ведущее устройство последовательно выдвигает текущее состояние ячеек и вдвигает новое;

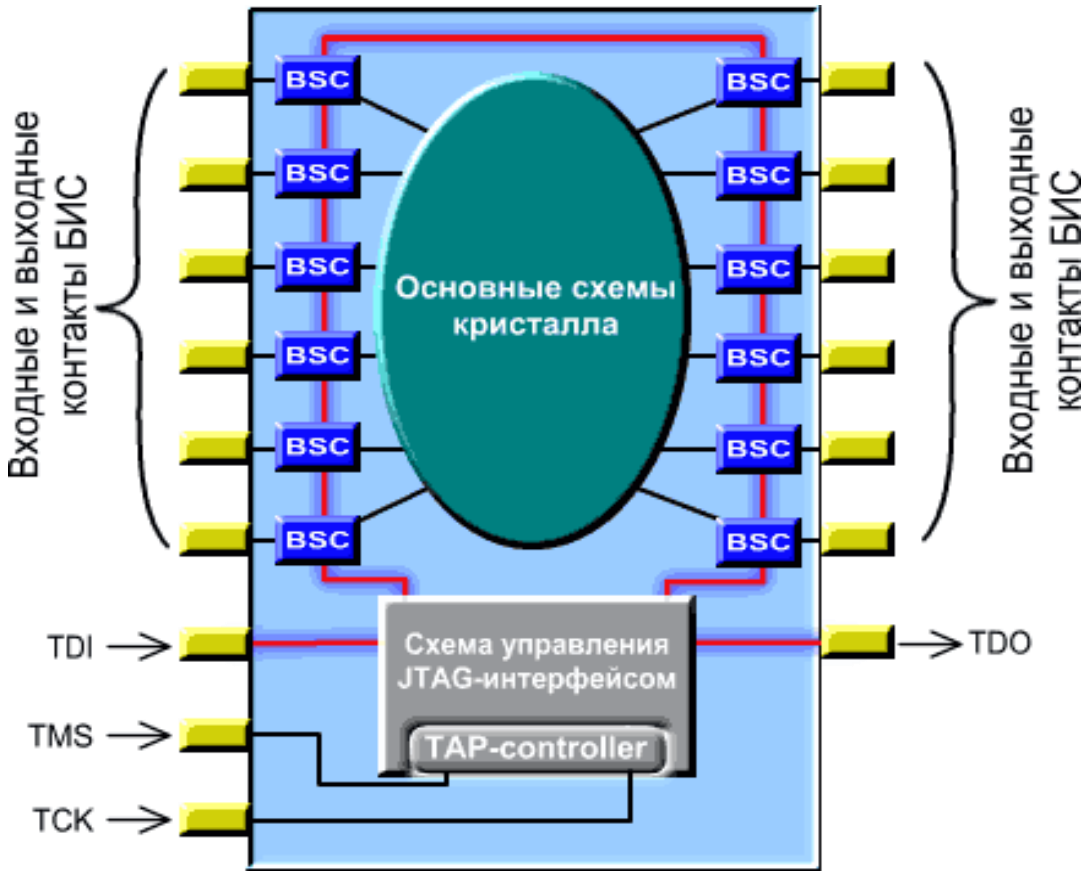
- режим наблюдения («Sample»). В этом режиме по импульсу текущее состояние вывода фиксируется в триггере и может быть потом считано ведущим устройством. При этом в процессе обмена данные, получаемые от ведущего устройства, фиксируются в триггере. При необходимости, в режиме тестирования (EXTEST) эти данные могут быть выведены на внешний вывод;

- режим тестирования (EXTEST, – Executing Test). В этом режиме на выход подается логическое значение, которое находится в триггере Т2.

Для целей программирования структура ячейки будет выглядеть следующим образом



Архитектура кристалла, поддерживающая метод граничного сканирования



Каждая ячейка граничного сканирования располагается во внешней части кристалла между контактной площадкой и внутренней частью.

Возможны следующие режимы работы:

- загрузка программ или чтение внутрисистемных ЗУ. В этом случае отключается контактная площадка, и вся информация поступает из BSC во внутреннюю часть кристалла или из кристалла в BSC;

- тестирование соединений БИС в плате или нескольких БИС между собой. Отключена внутренняя часть, остается соединение BSC с контактной площадкой;

- тестирование штатной работы. Все соединения сохраняются. Вся цепочка связей передается программе-обработчику.