Лекция 1.

Основные типы цифровых устройств.

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. По линиям данных информация передается последовательно. На адресные входы информация поступает параллельно. 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.

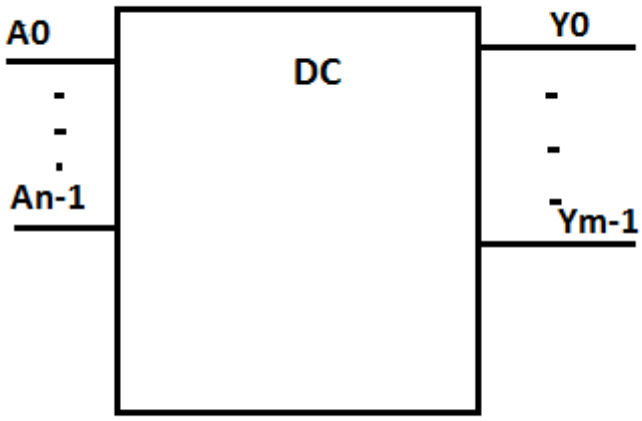
Синтез любого цифрового устройства производится согласно техническому заданию путем записи таблицы функционирования и, если это необходимо, составления по таблице системы логических уравнений в канонической форме.

Комбинационные цифровые устройства.

Дешифратор.

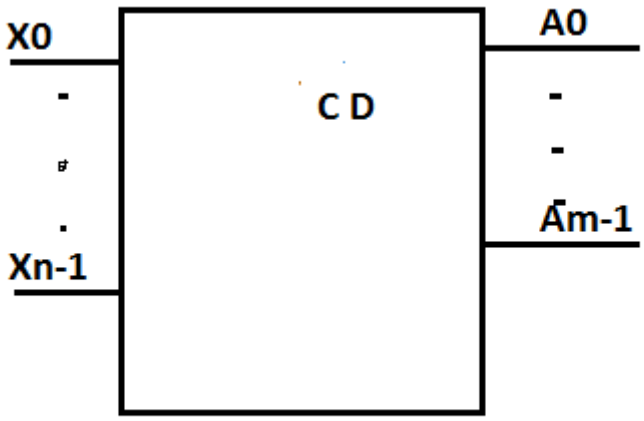
Дешифратор является одним из основных элементов вычислительных схем. Его входы подключаются к адресной шине обслуживаемого дешифратором блока, а на выходах формируются сигналы управления, позволяющие активировать в каждый момент времени один из элементов блока. **Дешифратор выдает управление по указанному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



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 $M = \log_2 N$

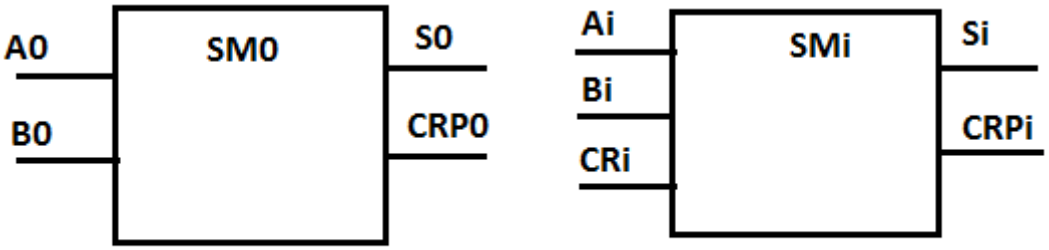


Сумматор

Сумматор – комбинационное цифровое устройство, предназначенное для получения арифметической суммы двух чисел, представленных в двоичном коде.

, для реализации N-рядного сумматора необходим один модуль полусумматора, 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));

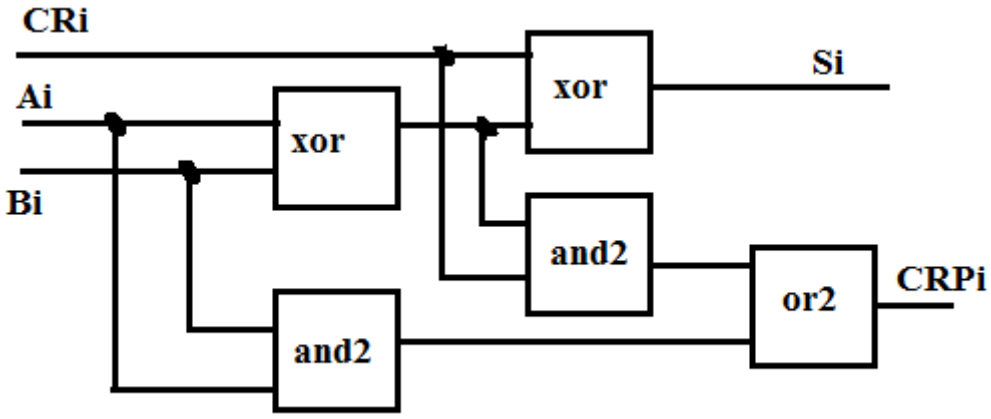
Объединим уравнения в систему

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

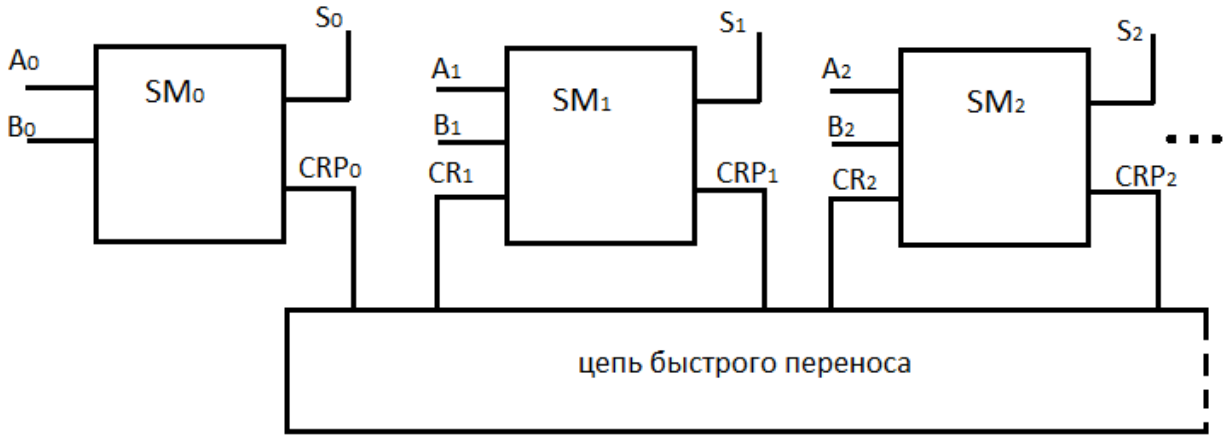
CRPi = **(Ai & Bi) \/ (CRi & (Ai xor Bi));**

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».

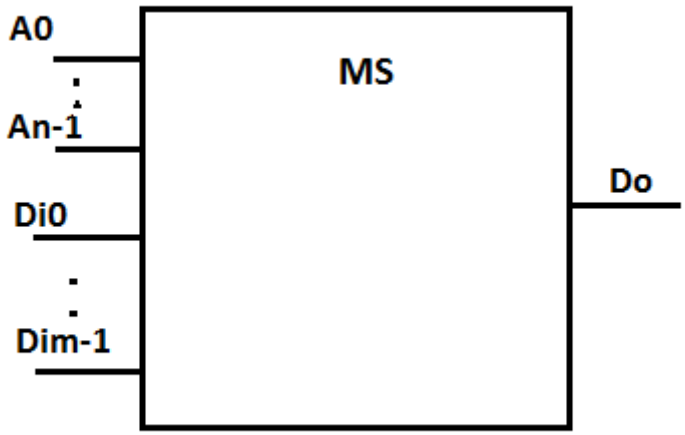


Одноразрядные сумматоры соединяются в многоразрядные путем объединения входов и выходов переноса. Для ускорения процесса используется специальная схема – цепь быстрого переноса. Возможность построения схемы быстрого переноса основана на том, что перенос используется в схеме одноразрядного сумматора на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



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

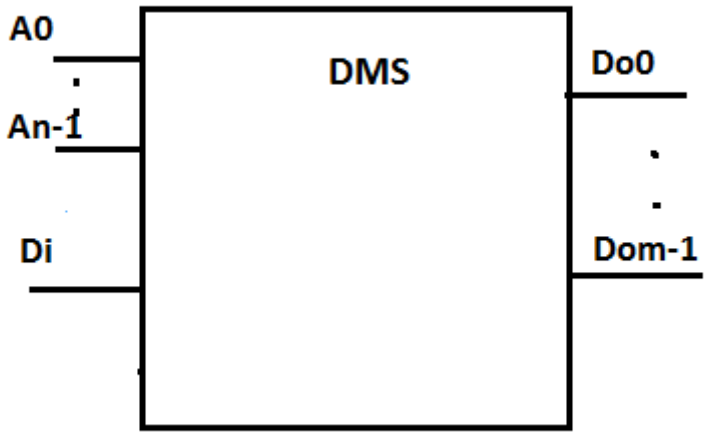
A1	A0	Di3	Di2	Di1	Di0	Do
0	0	-	-	-	0/1	0/1
0	1	-	-	0/1	-	0/1
1	0	-	0/1	-	-	0/1
1	1	0/1	-	-	-	0/1

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;

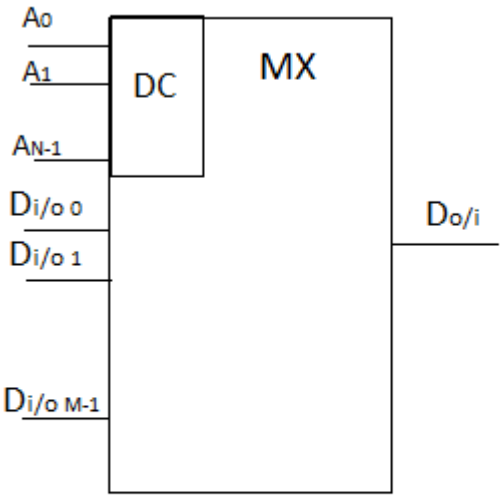
Do4 = A2 & nA1 & nA0 & Di;

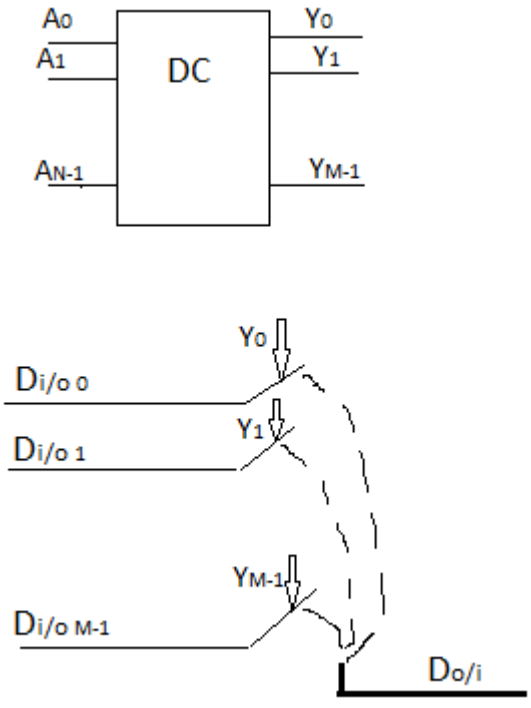
Do5 = A2 & nA1 & A0 & Di;

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.





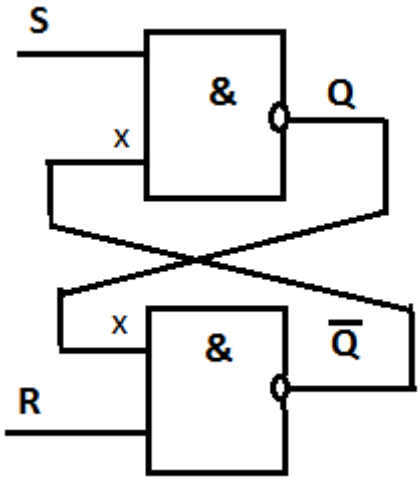
Конечные автоматы

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 - прямой 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».



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;

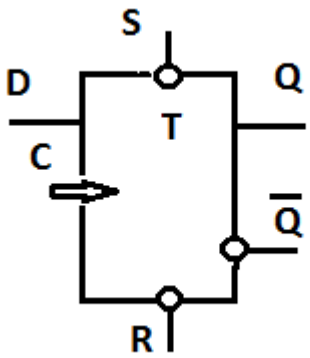
Отсюда можно заключить, что управляющим уровнем для такой структуры является «0». Действительно, при подаче «0» на один из входов такого элемента, на выходе появляется «1» независимо от состояния второго входа.

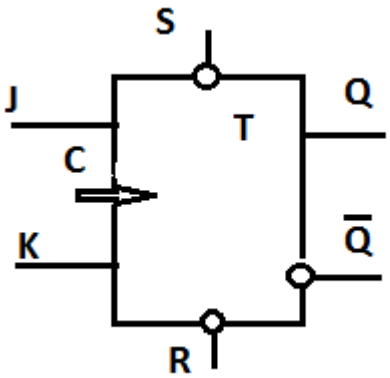
Таким образом, если на второй вход такого элемента подать состояние выхода аналогичного элемента (вход «x»), то получим:

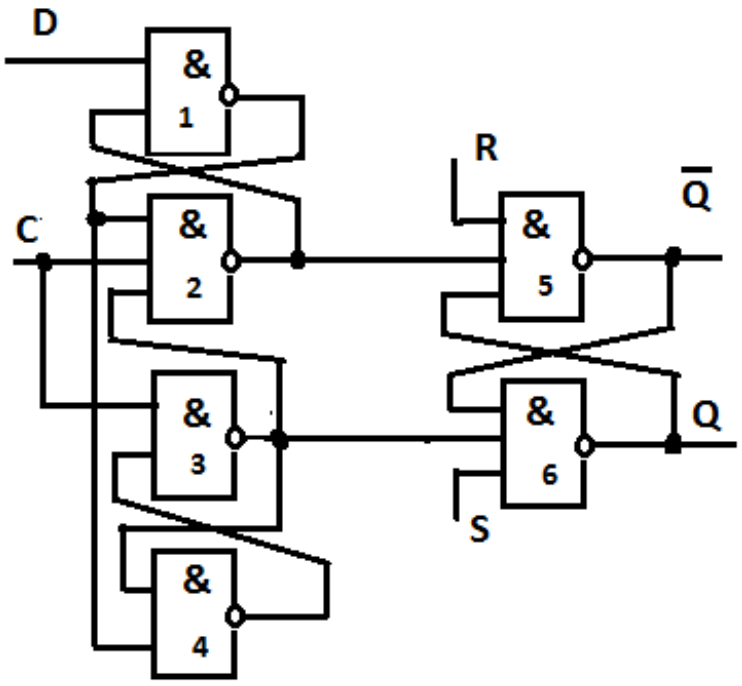
1. при отсутствии сигнала управления (S^n = 1, R^n = 1, Q^{n-1} - любое), состояние каждого из элементов триггера определяется входом “X”, т.е. петлей обратной связи. Триггер находится в режиме хранения информации (выделено желтым);
2. подача сигнала управления на R-вход приводит к установке триггера 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», обратная связь разрушается. Выход из такого состояния в **режим хранения** не определяется. Поэтому подача управления на оба входа называется запрещенной комбинацией.

-

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







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».

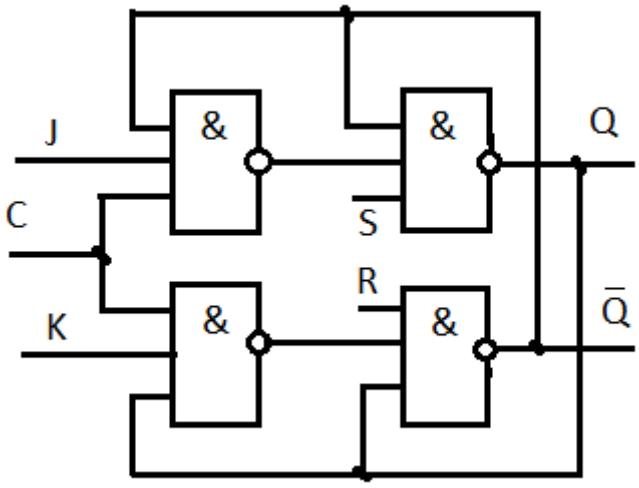
1. Момент времени, предшествующий подаче импульса синхронизации: C = 0, D = 1. Если C = 0, то выходы элементов «2» и «3» в состоянии «1», а, значит, ячейка на элементах «5» и «6» находится в режиме хранения, предположим, хранится «0». Теперь определим состояние

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 зависимости 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».

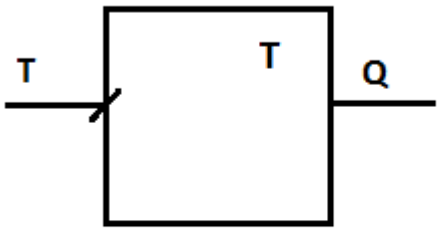


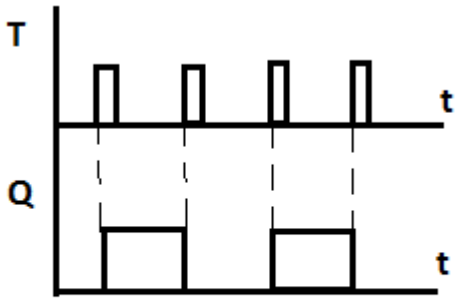
В JK-триггере информационные входы функционируют согласно следующей таблице (рассматривается в момент времени «n», т.е. момент прихода синхроимпульса).

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

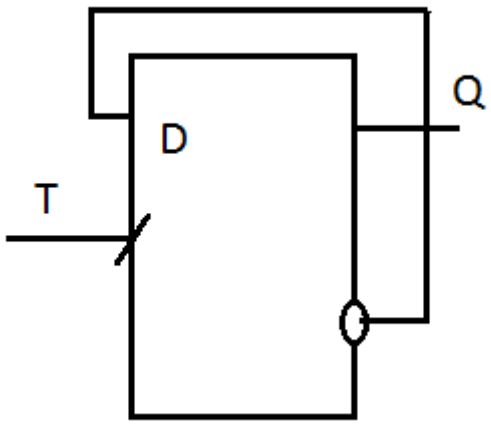
Активный уровень на информационных входах – 1. Вход J (Jump) –вход установки 1, вход K (Kill) – вход установки 0. Если оба входа неактивны триггер в режиме хранения, оба входа активны – состояние обратное предыдущему (режим Т- триггера). Значит, если объединить J и K входы и подавать на них «1», такой 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.

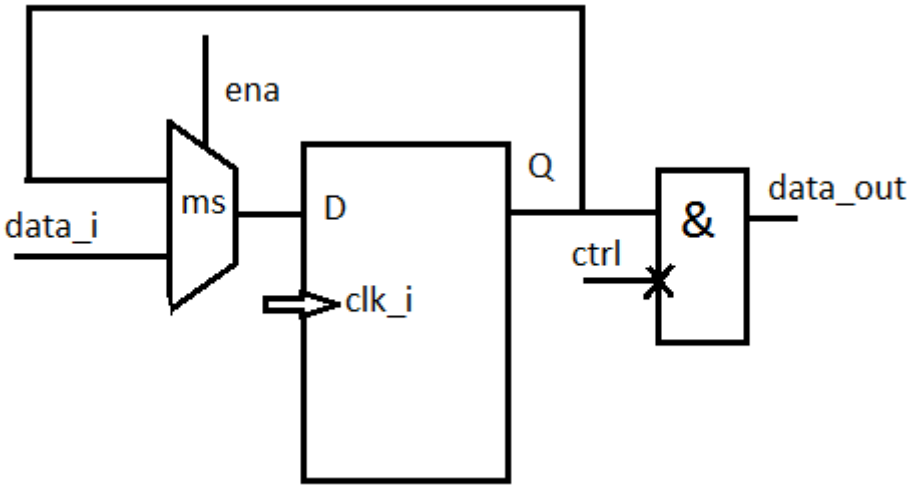
Лекция 2.

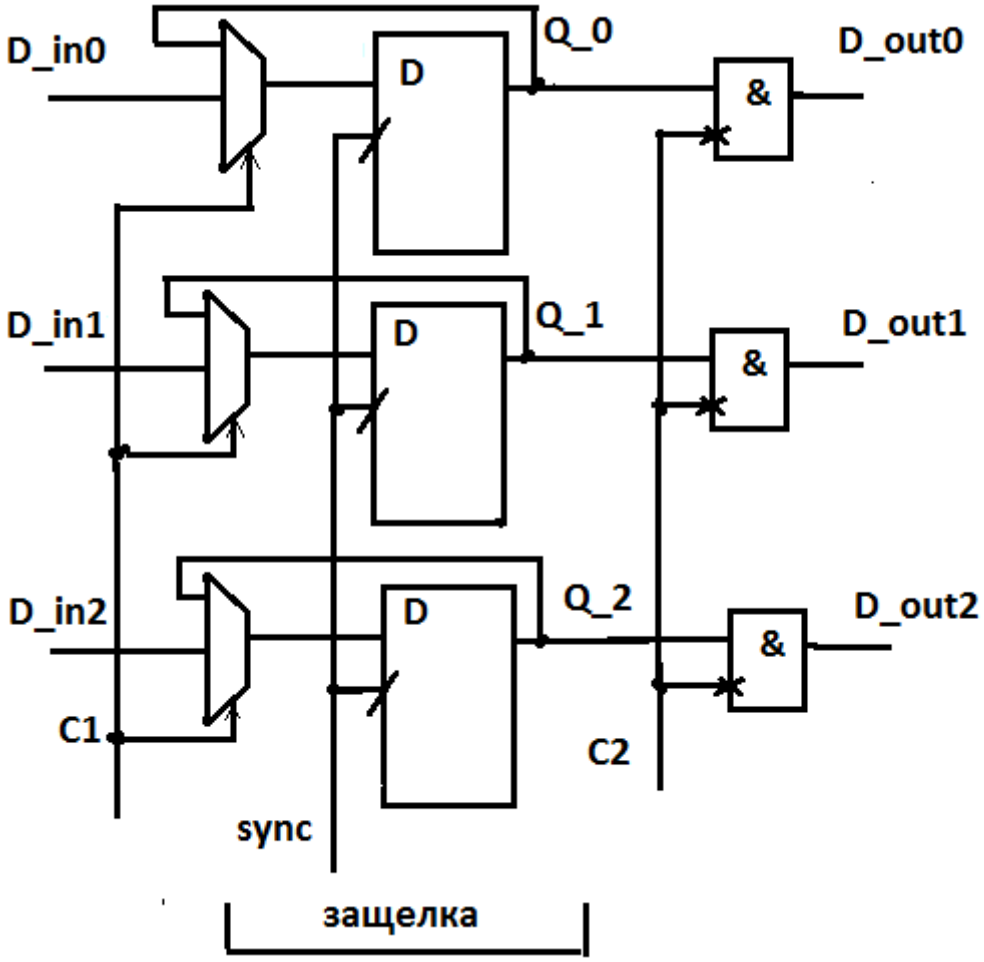
Дополнение к теме «Структуры регистров».

Регистры могут быть использованы для сдвига информации (последовательные) или для ее хранения (параллельные).

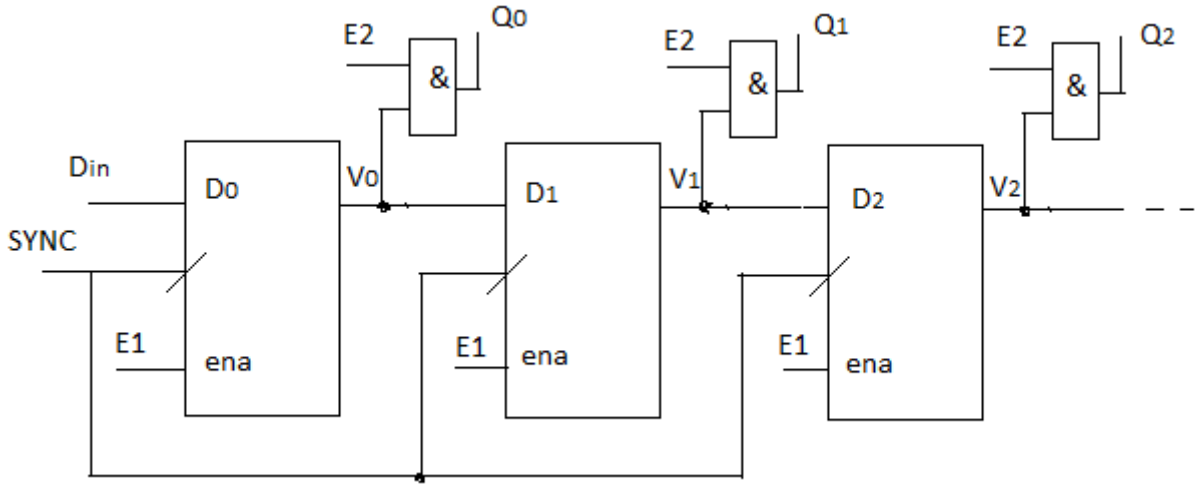
Параллельные регистры, в свою очередь делятся на регистры-защелки и буферные. В защелках импульс синхронизации может формироваться локально, как управляющий сигнал. В буферном регистре сигнал синхронизации всегда глобальный, т.е. следует в общем синхропотоке. Переписать состояние такого регистра можно с помощью управления буфером записи. На рисунке представлен один разряд буферного регистра.

Управление буфером записи производится по входу **ena**, управление буфером чтения – по входу **ctrl**. В данном случае буфер чтения бестоковый. Для варианта токового буфера используется обычный конъюнктор. Следует напомнить, что запись в регистр **всегда** происходит в момент подачи фронта импульса синхронизации.

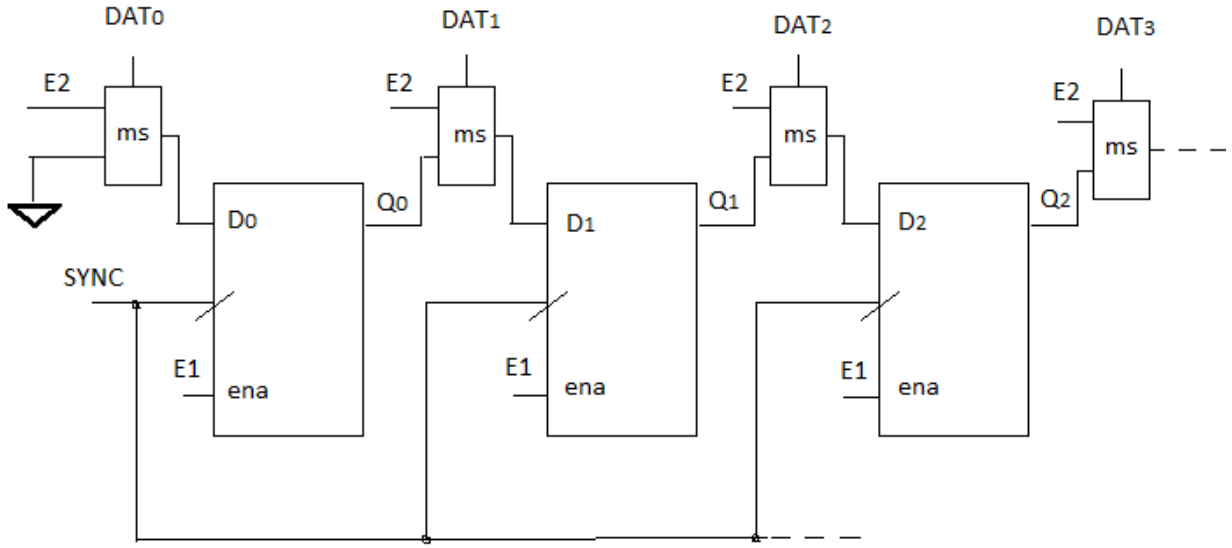




Для целей приема и передачи информации используют комбинированные регистры (последовательно-параллельные и параллельно-последовательные).



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.



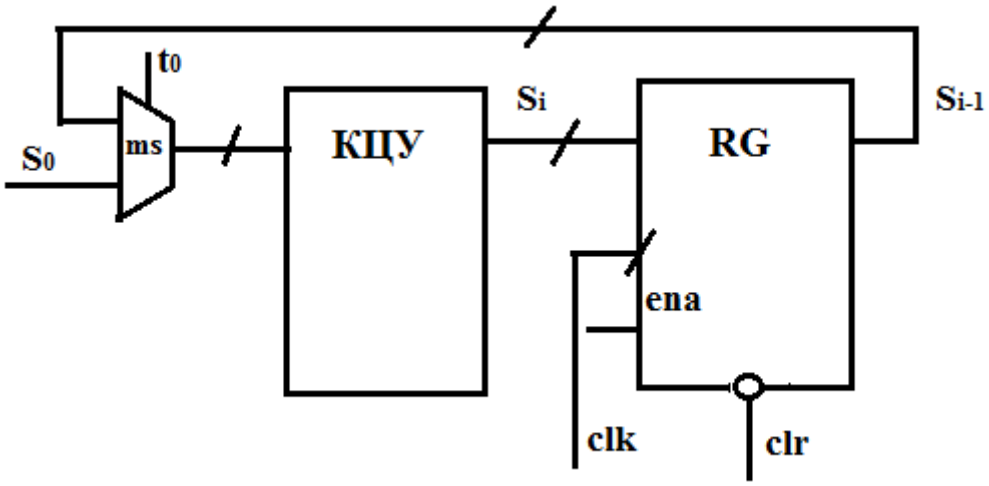
В **параллельно-последовательном** регистре запись информации производится параллельным способом при подаче одного импульса синхронизации и условии **E1=1** и **E2=1**. Считывание происходит путем сдвига информации и заполнения освобождающихся разрядов «0». Условие такого действия **E2=0**, а **E1=1**, количество подаваемых синхроимпульсов

соответствует разрядности регистра. После выдачи информации в линию возможна следующая запись. Весь процесс также регулируется с помощью счетчика.

Дополнение к теме «Счетчики».

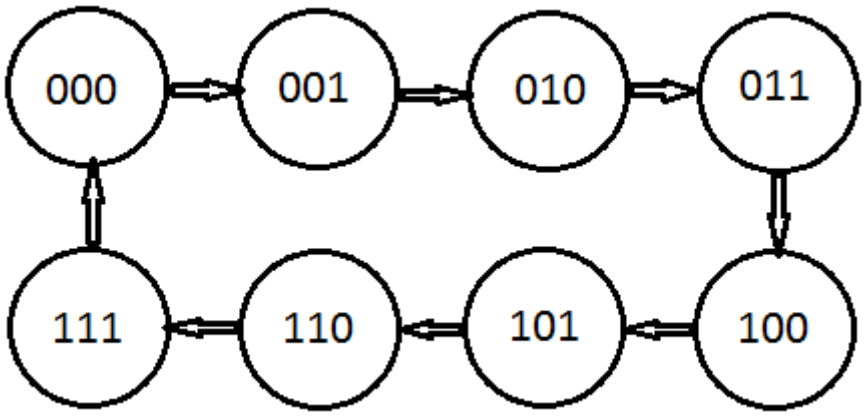
Счетчики, устройства для подсчета количества поступивших импульсов синхронизации, строятся на основе триггерных ячеек. Мы знаем, что любой триггер в момент подачи синхроимпульса записывает информацию, которая поступила к этому моменту на информационные входы. То есть, к моменту подачи синхроимпульса, на разрядные входы триггеров счетчика необходимо подать информацию, полученную путем преобразования предыдущего состояния выходов устройства. Значит, выходы разрядных триггеров надо подключить к входам КЦУ, которое будет производить соответствующее преобразование. Кроме того, все триггеры установочного типа имеют асинхронные входы S и R, воздействуя на которые возможно производить асинхронную установку состояния счетчика.

Таким образом, для построения любой пересчетной схемы, мы должны создать еще одну, внешнюю, цепь обратной связи и включить в нее КЦУ собственно для целей преобразования предыдущей информации. Схема такого конечного автомата будет выглядеть следующим образом

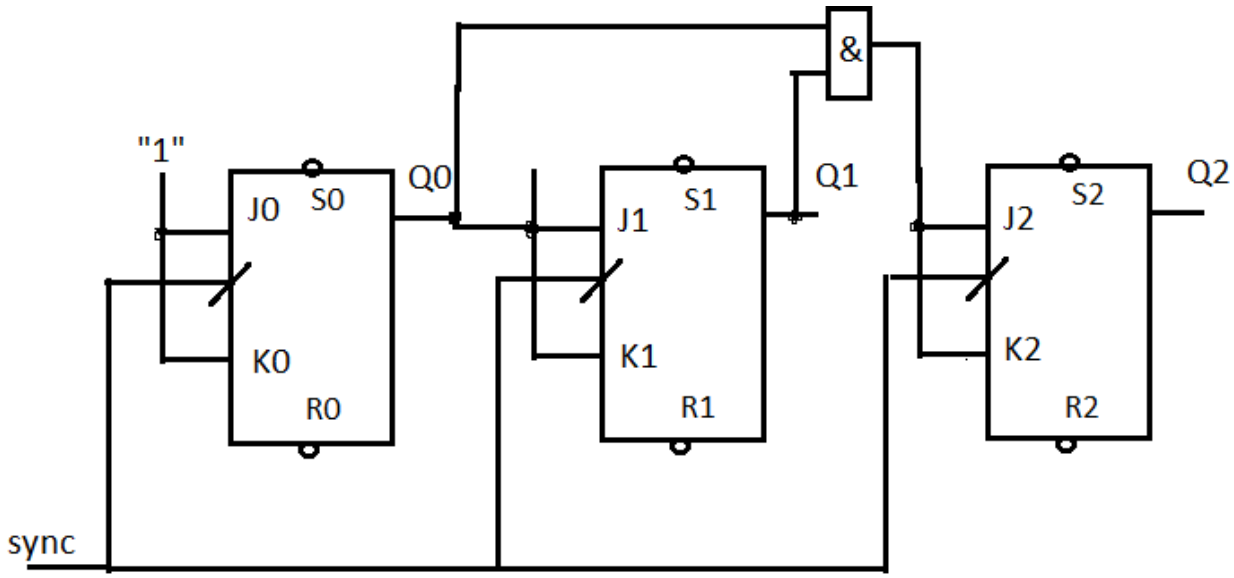


Если мы строим схему счетчика на дискретных компонентах, то удобнее использовать JK-триггеры, которые при подаче «1» на оба информационных

входа будут работать в режиме триггера переключательного типа. В этом случае для синтеза можно просто рассмотреть граф переходов для трехразрядного суммирующего счетчика,



из анализа которого очевидно, что нулевой разряд переключается при каждом фронте синхроимпульса, первый разряд – после появления «1» на выходе нулевого, второй – после появления «1» на двух предыдущих выходах, и т.д., если увеличивать разрядность. Известная вам схема такого счетчика выглядит следующим образом



а уравнение связей записывается

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

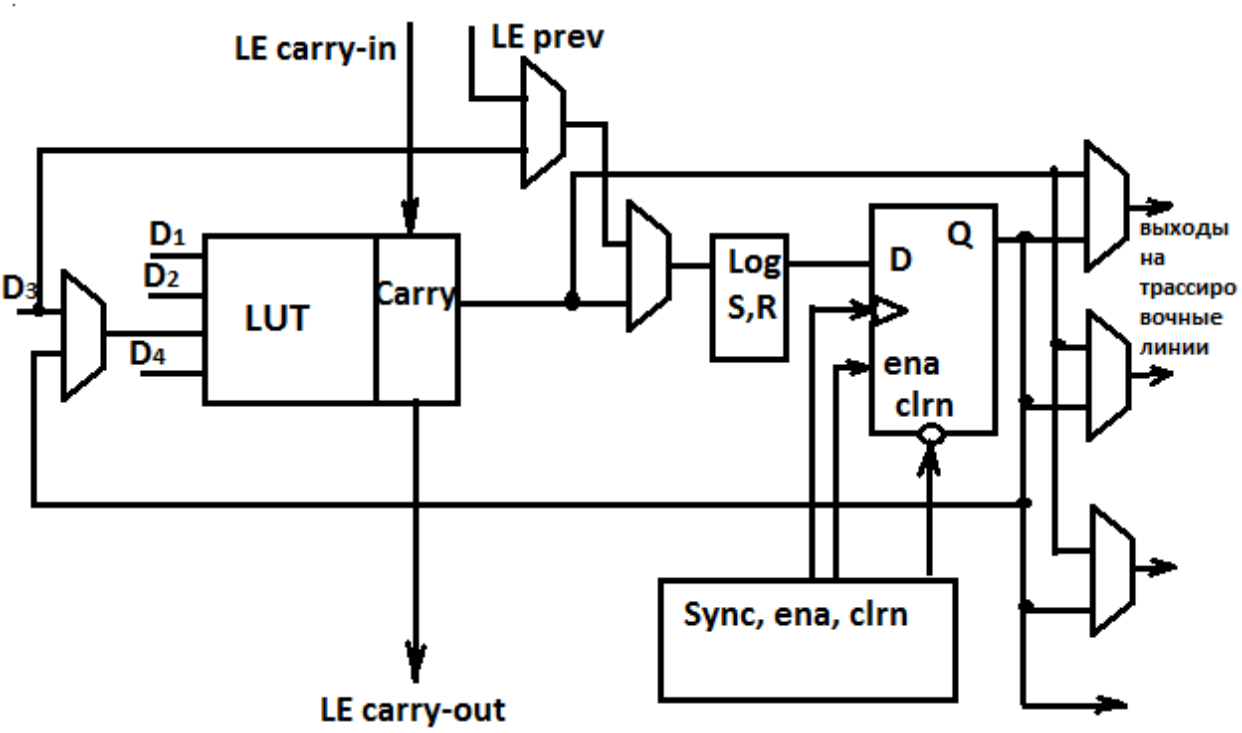
Соответственно, для вычитающего счетчика можно рассмотреть тот же граф переключений, но в обратном направлении. Тогда уравнение связей будет выглядеть так

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

А если добавить вход реверс, то, с помощью управления по этому входу, изменяется структура связей с прямых выходов при отсутствии реверса, на инверсные при наличии реверса. Уравнение связей для реверсивного счетчика

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

Если же производится проектирование под программирование в кристалл, то записываемая программа отражает следующую структуру: для n-разрядного счетчика в таблицы заносятся состояния i-х выходов в зависимости от (i-1)-х, по каждому синхроимпульсу состояния, полученные от всех n таблиц, сохраняются в n-разрядном регистре. При запрете счета вход ena позволяет перезаписывать предыдущее состояние при каждом новом импульсе синхронизации. При синхронном сбросе или начальной установке эти состояния подаются перед приходом фронта синхронизации, при асинхронном - формируются вне зависимости от синхронизации.



Недвоичные счетчики.

В недвоичных счетчиках коэффициент счета (количество состояний в цикле счета) всегда $<2^n$, где n - количество разрядов счетчика. Проще говоря, в таких счетчиках не присутствует часть возможных состояний. Отсутствующие состояния можно «вырезать» асинхронно, если подключить КЦУ к установочным входам. Тогда состояние, в которое счетчик приходит после последнего допустимого, даст установку в начальное состояние. Но при этом длительность удержания начального состояния будет меньше, чем длительность любого из остальных. Поэтому предпочтительнее строить счетчики с синхронной установкой начального состояния. Такой синтез сводится к синтезу КЦУ цепи обратной связи. Он аналогичен синтезу любого пересчетного конечного автомата. 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.

Лекция 3, 4.

Введение.

В настоящем курсе рассматриваются структуры основных узлов цифровых устройств, основанных на базовых цифровых устройствах, относящихся к классу КЦУ или ПЦУ (конечные автоматы). При изучении конструкций таких схем будет представлена возможность проектирования и программирования их в кристалл программируемой логики. Соответственно, для начала необходимо понять, что же представляет собой кристалл программируемой логики, а затем уже перейти непосредственно к конструкциям проектируемых узлов.

Программируемые логические интегральные схемы (ПЛИС).

Если вспомнить этапы синтеза цифрового устройства, то можно выделить два основных момента: запись таблицы функционирования (истинности) и составления на ее основе системы логических уравнений в канонической форме. 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:

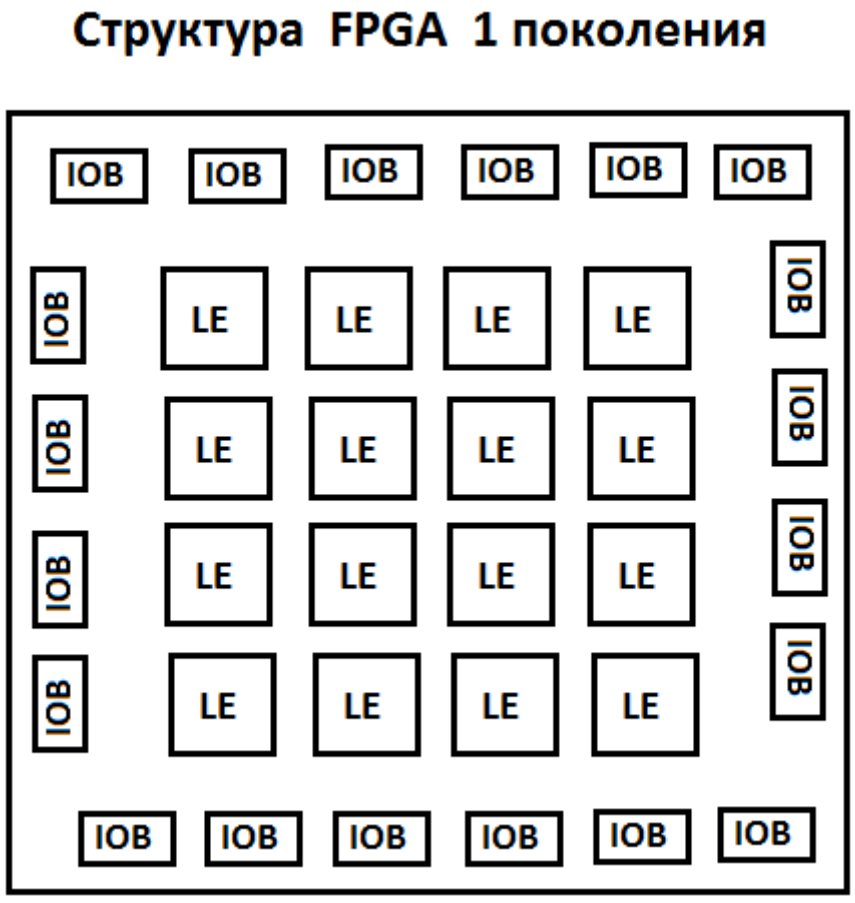
CPLD – Complex Programmable Logic Devices и FPGA – Field Programmable Gate Array.

Конструктивно любая 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 FPGA.

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

Логический блок основан на табличном ЗУ – LUT.

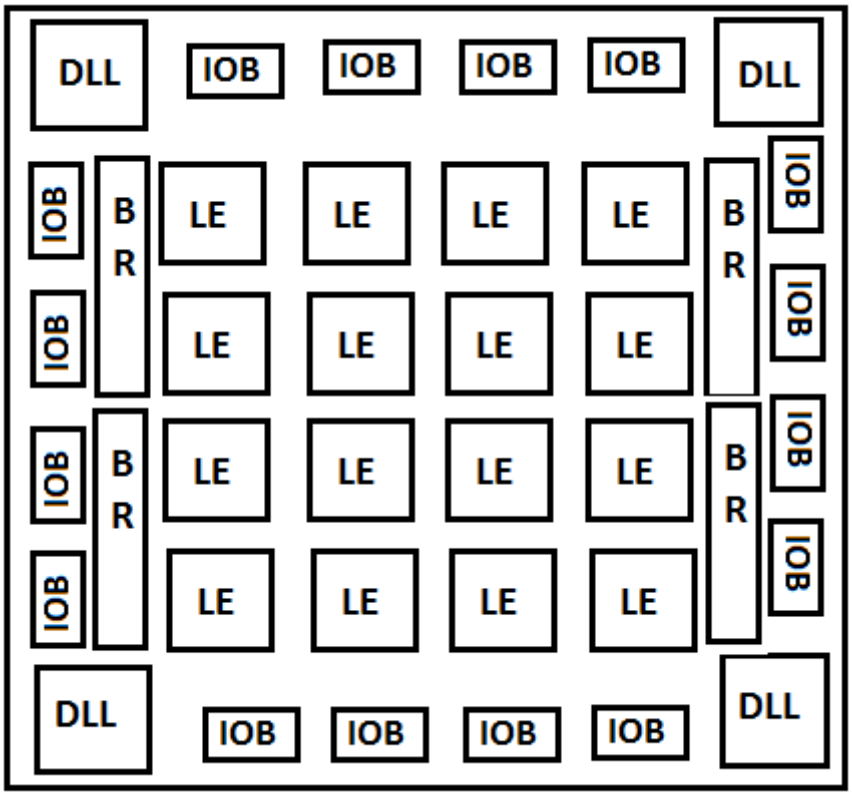


Основа системы межсоединений – выделенные линии каналов. При такой системе невозможно предсказать путь соединения логических блоков,

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.

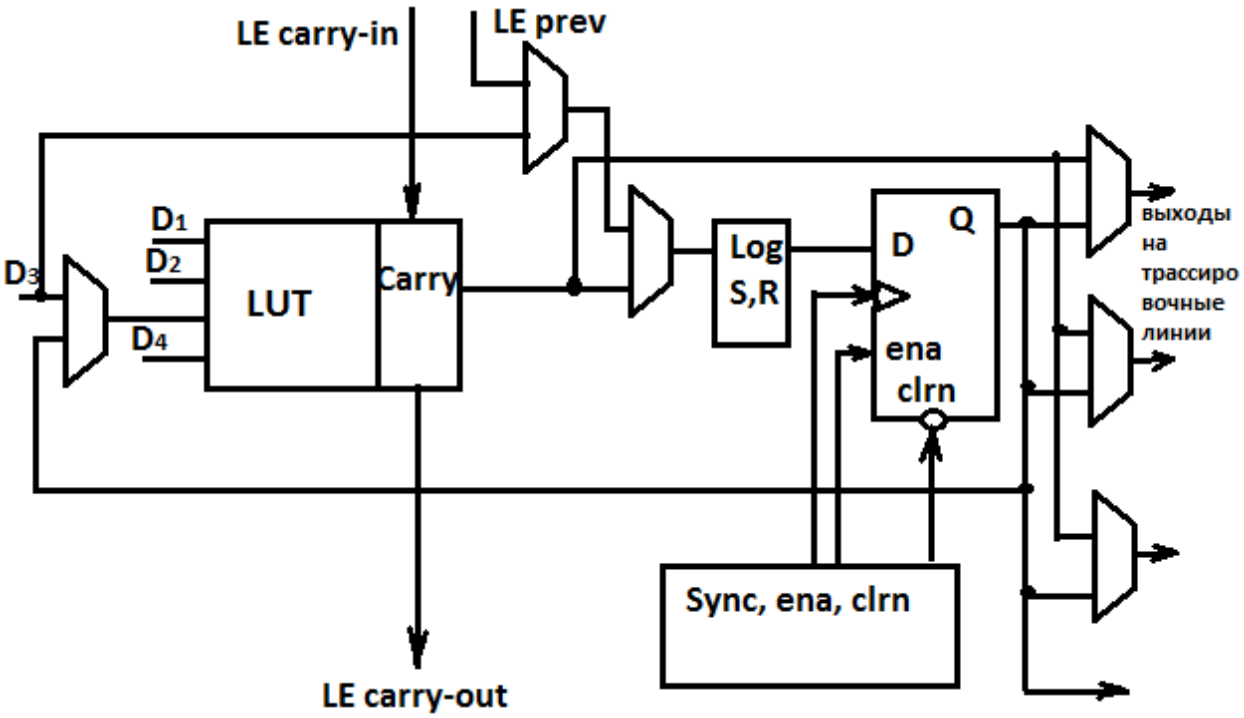
Для устранения подобных недостатков в FPGA последующих поколений (2, 3) стали вводить дополнительные логические связи – локальную матрицу соединений, позволяющую объединять отдельные ячейки, сформированные на основе LUT, в логический блок. Для компенсации фазовых сдвигов в структуру кристалла вошли блоки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

FPGA 3 поколения



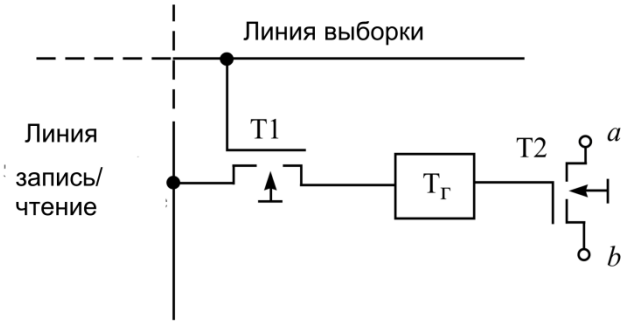
Логический блок строится на основе небольших блоков запоминающих устройств, LUT (Look-up-tables) на16 ячеек (4 входа). Для коммутаций в структуре блока присутствуют управляемые мультиплексоры, для сохранения состояния предыдущего такта – триггер. В дальнейшем, в 4-м и 5-м поколениях FPGA объем памяти LUT возрастает до 64 ячеек.

Фрагмент структуры конфигурируемого логического блока (логического элемента).

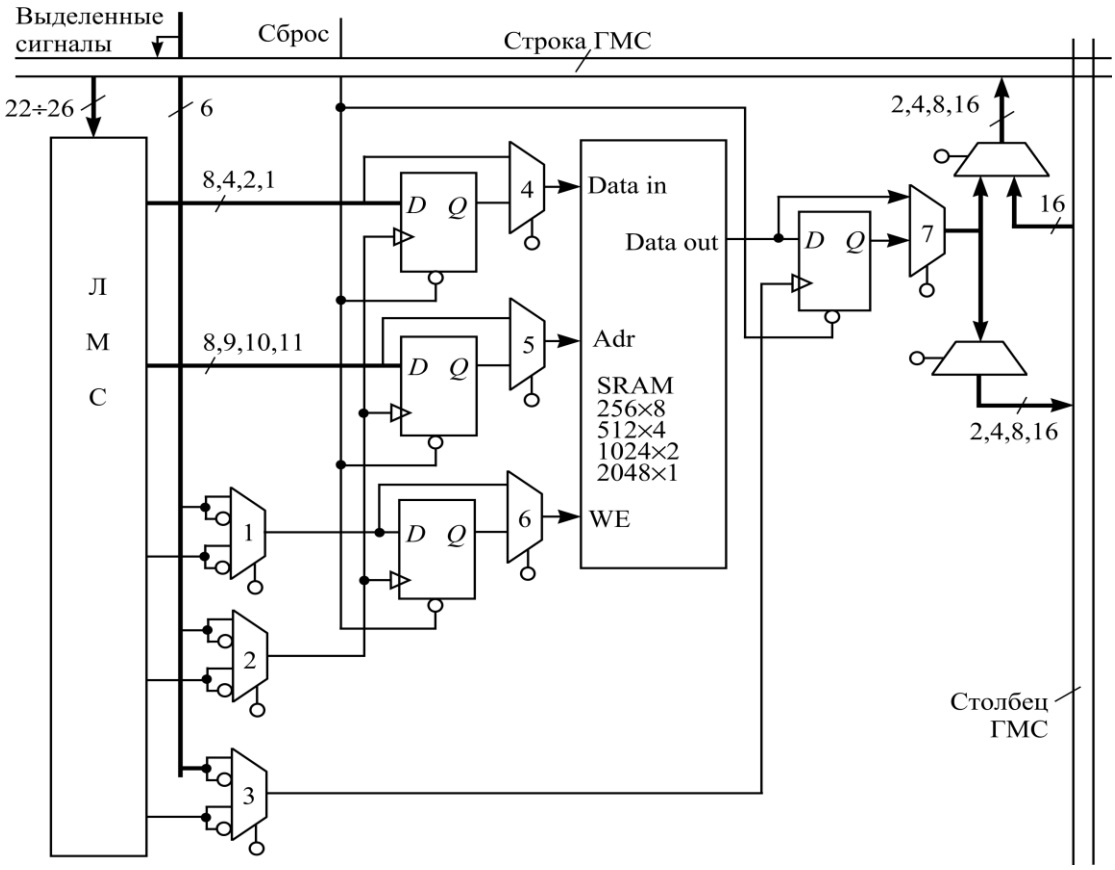


В FPGA 4 и 5 поколений архитектура основана на матрице мегаблоков, включающих в себя некоторое количество конфигурируемых логических блоков, объединенных посредством локальной матрицы соединений. Элементами такой матрицы служат правые части каждого логического блока, представленные простой логикой. Левую 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 конфигурируются по длине ячейки. Блоки двухбуферные, в начальных вариантах небольшой общей емкости (2К). В последних поколениях схем такие блоки имеют существенно больший объем.



не

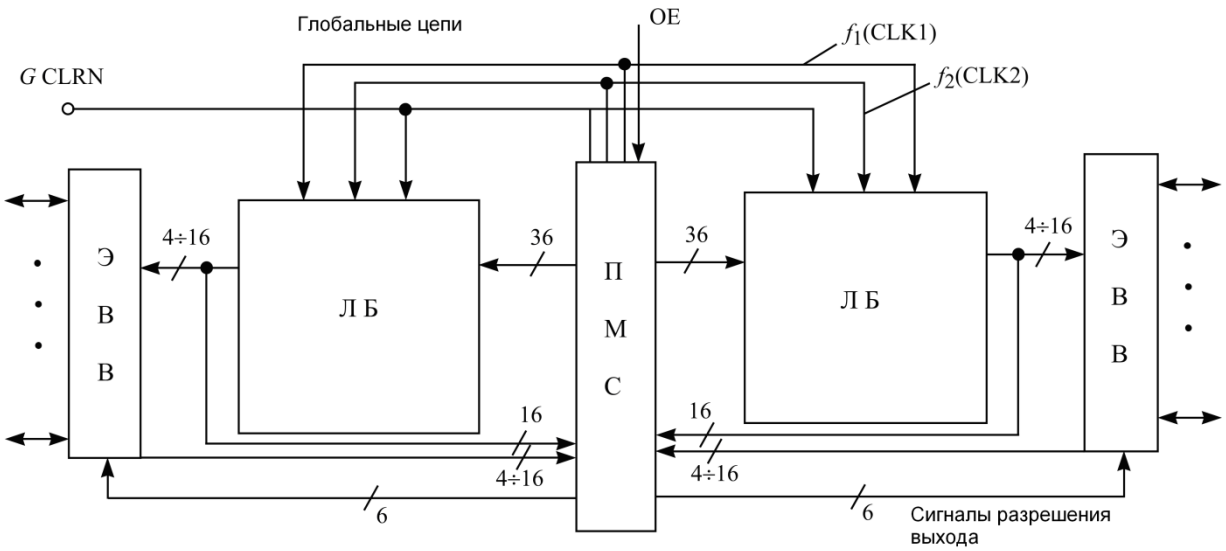
Память конфигурации на ячейках SRAM не позволяет сохранять конфигурацию схемы в кристалле при отключенном питании. Для восстановления конфигурации на плате к схеме присоединяют

конфигурационную FLASH, где коды сохраняются при выключенном питании.

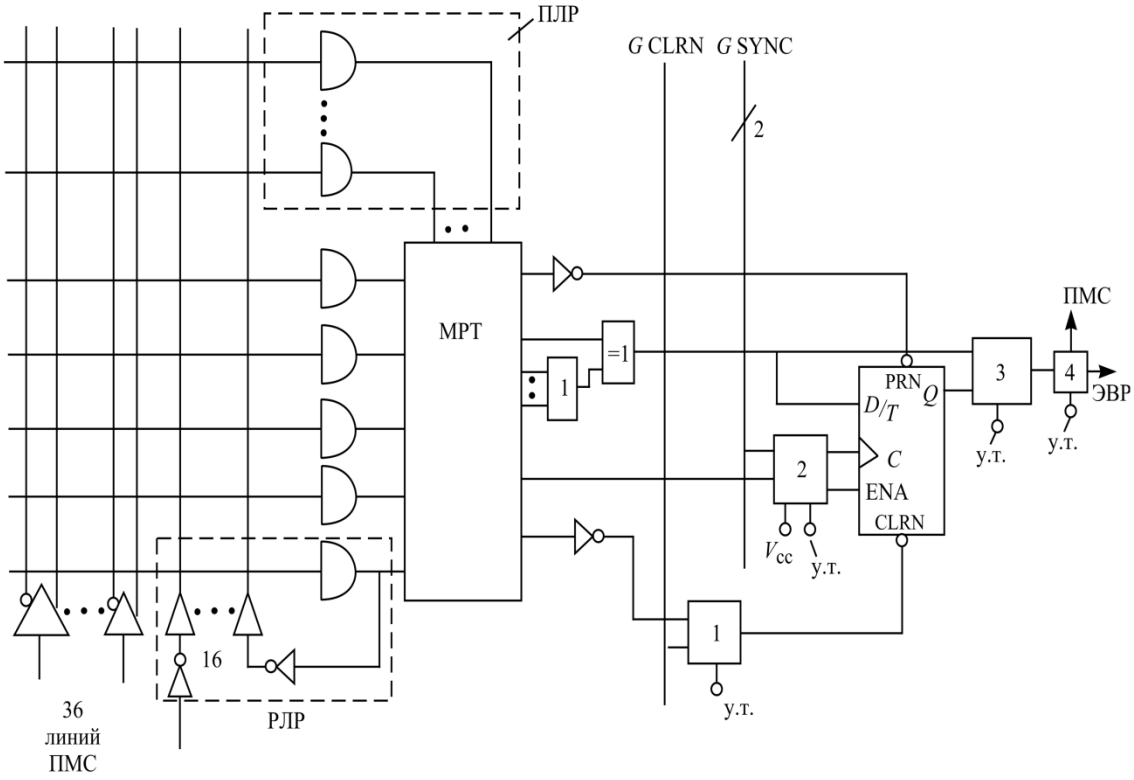
Программирование под FPGA всегда следует производить с использованием разветвленных алгоритмов, т.к. в таблицах все ячейки должны быть заполнены.

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.

Лекция 5.

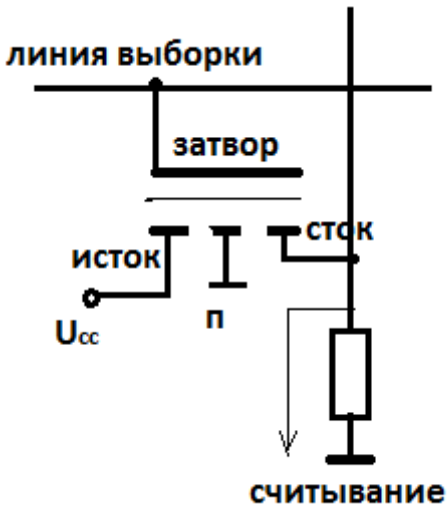
Устройства памяти.

Возможна различная классификация устройств памяти. Прежде всего, для процессорных систем различают внутреннюю память и внешнюю память. В нашем курсе мы будем касаться только внутренней памяти процессорных систем.

Сначала рассмотрим варианты такой памяти с точки зрения физического строения ячеек накопителя.

Здесь можно выделить ячейки накопителя, построенные на триггерах и ячейки накопителя, построенные на транзисторах. Причем последние также разделяются на категории: ячейки, использующие переход как емкость, и ячейки, использующие непосредственно регулируемый канал.

Регулируемый канал используется в постоянных запоминающих устройствах, матрицы накопителей которых построены на полевых транзисторах. Такие схемы 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.

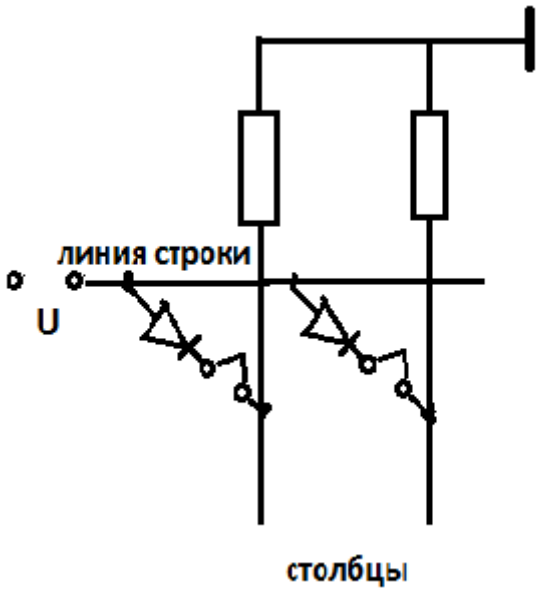
Плавающий затвор: между затвором и каналом вводится дополнительная область, вызывающая стекание в нее заряда. При снятии напряжения с затвора заряд сохраняется и удерживает транзистор в запертом состоянии. Пороговое напряжение настолько велико, что поле не создается. В схемах,

построенных по такой технологии стирание возможно только со всей поверхности кристалла (со всей матрицы накопителя). Прежде так выполнялись схемы с ультрафиолетовым стиранием информации.

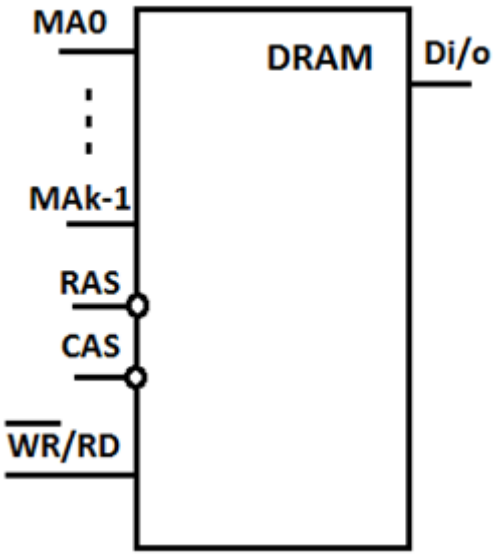
В настоящее время ультрафиолетовое стирание практически не применяется, термин EPROM используют для схем на основе плавающего затвора, которые допускают только полное стирание информации, и имеют значительно меньший ресурс для перезаписи, чем схемы на основе двойного затвора.

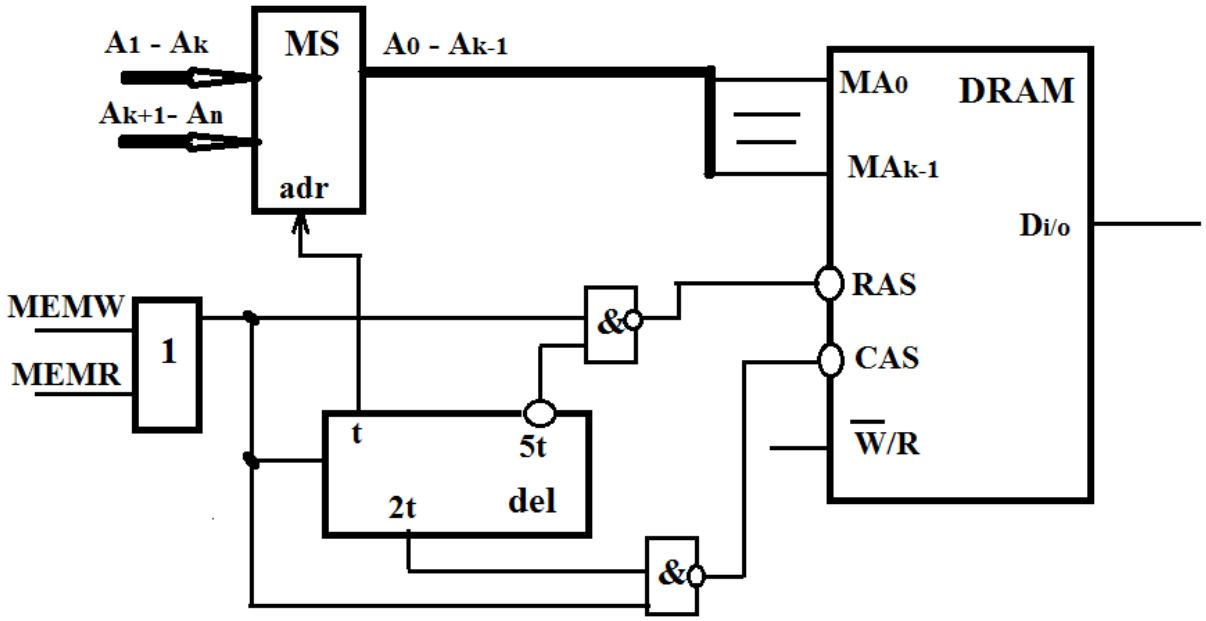
Двойной затвор получают встраиванием дополнительного электрода. В этом случае появляется возможность стирания определенных участков информации. На основе такой технологии строятся схемы FLASH.

Следует также упомянуть о схемах ПЗУ, использующих простую диодную матрицу. Это может быть или готовая матрица, имеющая диоды только в ячейках с записью «1» - ROM, или однократно программируемая пользователем матрица, узлы которой представлены диодами с перемычками(«1») или диодами, включенными встречно(«0») -PROM.

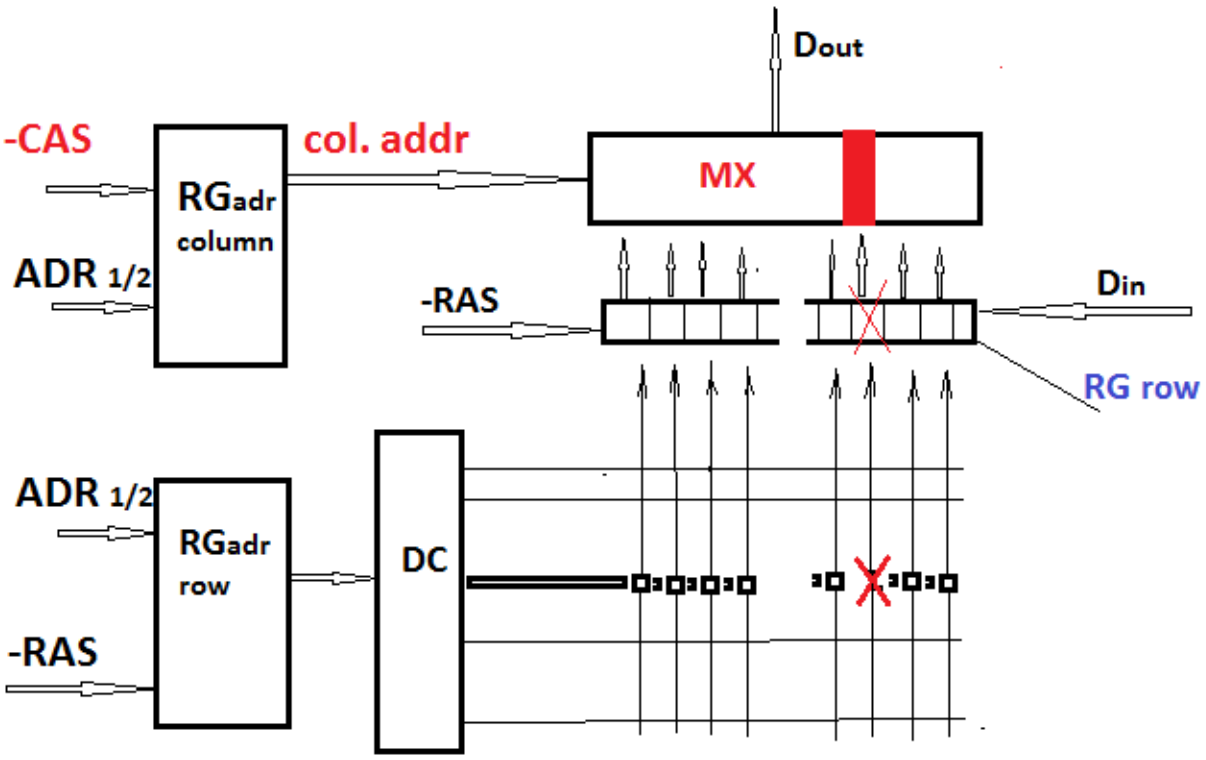


Матрица накопителя динамического ОЗУ построена на ячейках, использующих **емкость перехода**. Их основа – увеличенная емкость перехода затвор-исток полевого транзистора.





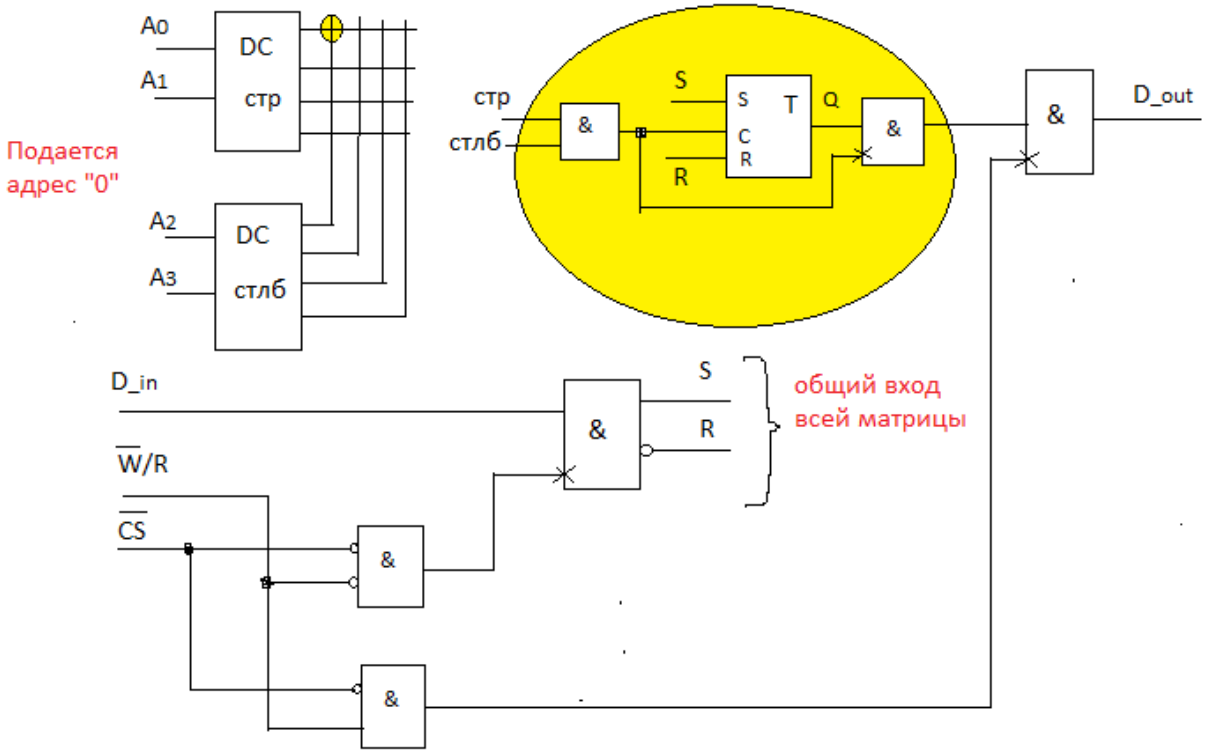
В DRAM адресные входы подключаются к шине через мультиплексоры, выделяющие адреса строки и столбца. K=n/2. Вход RAS (row address strobe) (строб строки) активен, когда подается адрес строки, вход CAS (column address strobe) (строб столбца) активен после подачи адреса столбца.



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.

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.

Классификация по способу доступа к ячейке памяти.

По способу и порядку доступа к ячейке внутреннюю память можно разделить на адресную, память с последовательным доступом и ассоциативную.

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.

Лекции 6, 7.

Устройства памяти. Ассоциативная память.

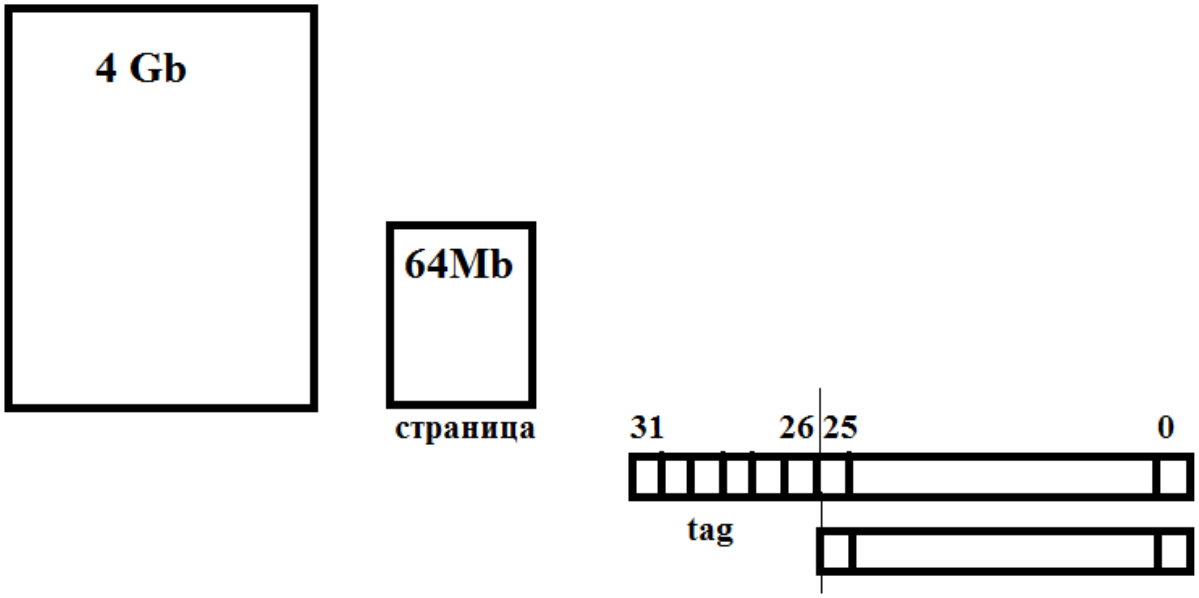
Ассоциативная память всегда является дополнительной к основной и не увеличивает общего объема памяти системы. Такая память известна как Cache – память. В Cache всегда есть два блока: адресный и данных. В адресном блоке хранятся сведения об адресах участков информации, скопированных из основной памяти, а в блоке данных – собственно, скопированная информация.

Смысл появления Cache в том, что основная память системы построена на основе динамических ячеек, которые не могут отрабатывать мгновенно. Преимущество динамической памяти только в ее низкой себестоимости. Для увеличения скорости работы памяти параллельно большой динамической ставятся блоки статической памяти существенно меньшего объема.

Условно весь объем памяти системы можно разделить на страницы, каждая из которых имеет объем Cache. В приведенном примере получаем:

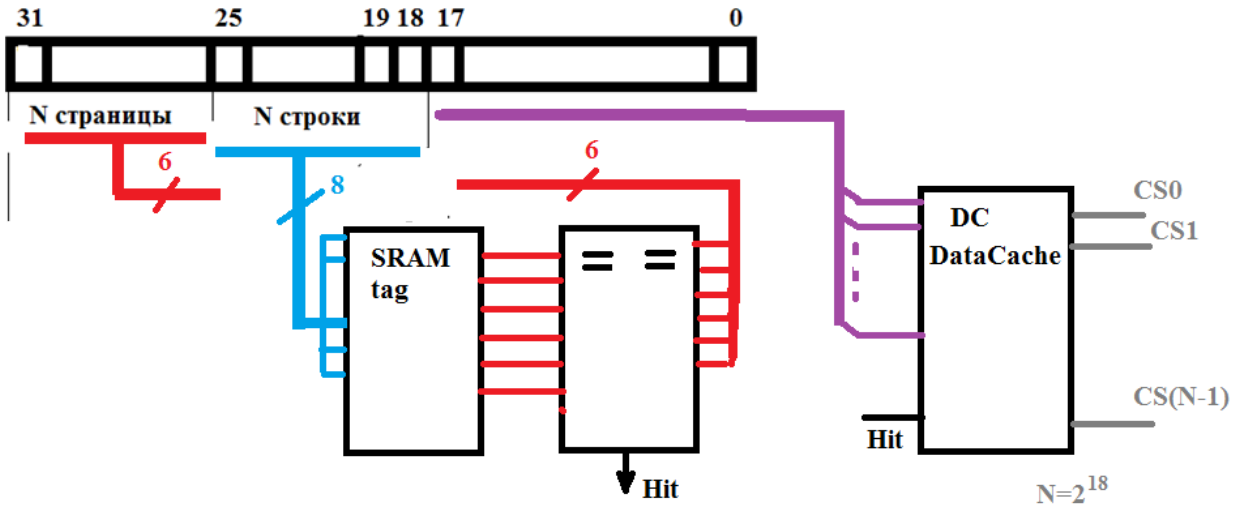
$$2^{32} : 2^{26} = 2^{6}$$

страниц Cache укладывается на основной памяти.

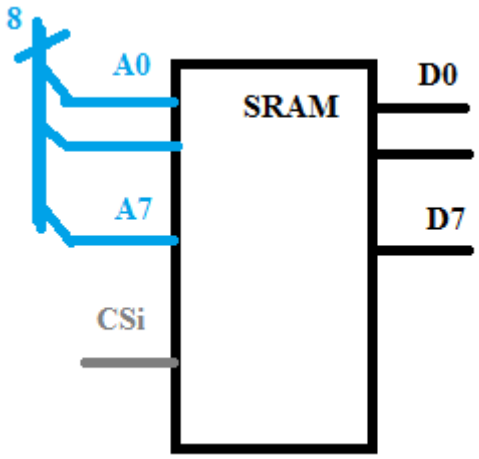


Таким образом, если копировать из основной памяти в Cache подобный объем информации, записанной в подряд идущих ячейках, то адрес этого участка будет записываться на разрядах шины с 26 по 31 соответственно. Этот адрес называется признаком скопированной информации, или тегом(tag). Cache, в которую информация копируется постранично, называется Cache прямого отображения.

При обращении к ячейке памяти ее адрес выставляется на шине, и участок, содержащий тег, сравнивается с сохраненным в адресной Cache тегом. Для этого номер страницы поступает на первые входы схемы сравнения, а номер строки - на SRAM тегов. Выход SRAM тегов поступает на вторые входы схемы сравнения. Если совпадение есть, вырабатывается сигнал Hit=1, открывающий Cache данных.



На рисунке видно, что в случае Cache прямого отображения получается небольшая адресная часть для сравнения 6-разрядного тега для каждой из 256 строк. В Cache данных входит 2^{18} структур, отображающих символы в каждой строке. Таким образом, объем каждой структуры 256 ячеек.

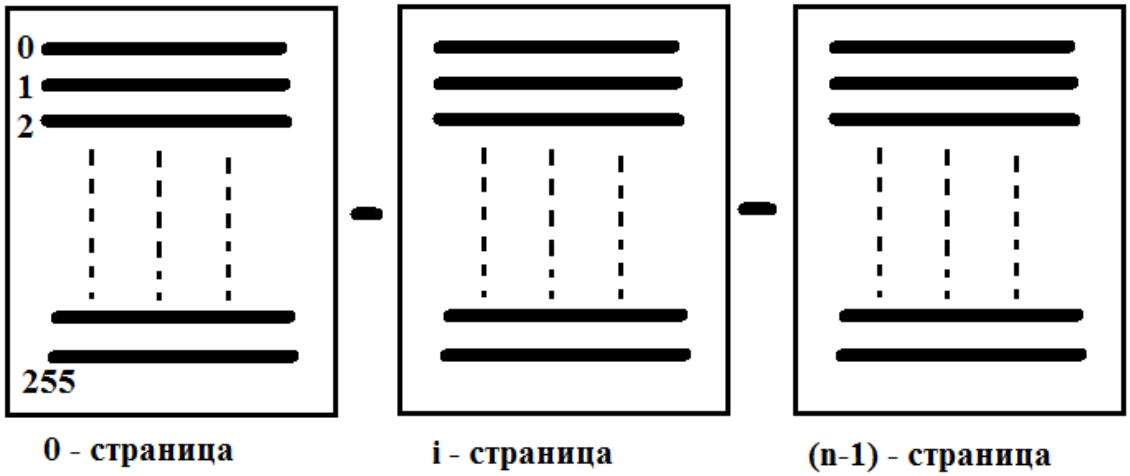


Если символы будут не байты, а слова, то количество структур – 2^{16}, обслуживающих 32 разряда шины данных.

Недостатком Cache прямого отображения является трудность работы с разветвляющимися программами, когда скопированные участки памяти оказываются невостребованными.

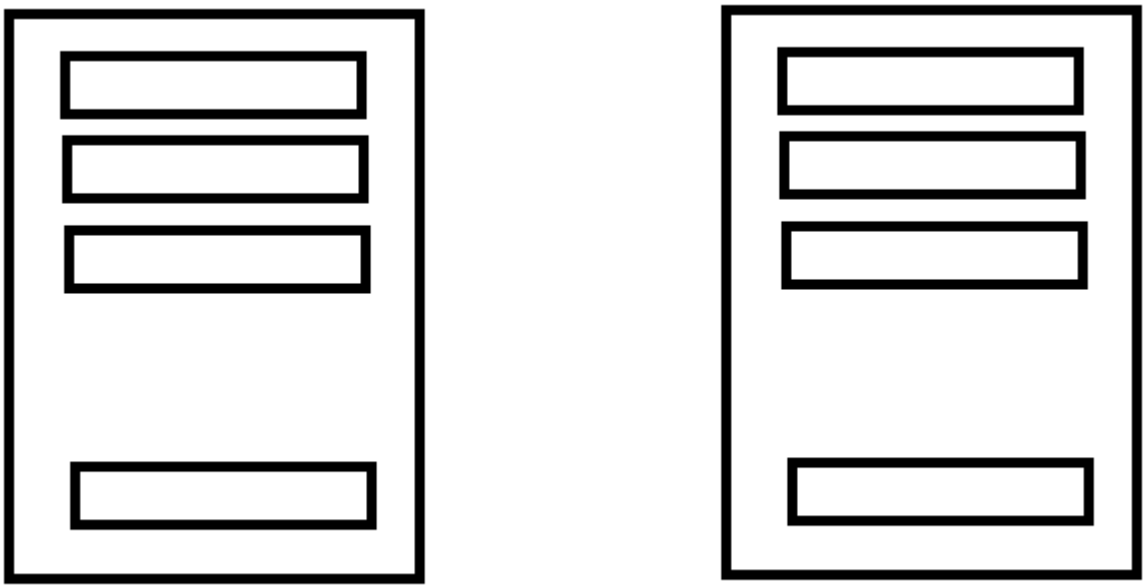
В этом отношении удобнее может показаться полностью ассоциативная Cache, где возможен набор страницы из разных строк с разных страниц.

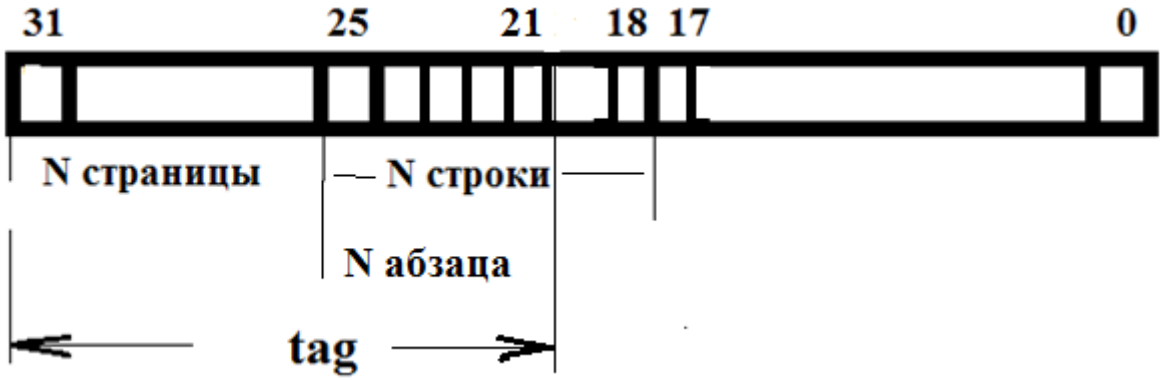
В этом случае тег включает в себя номер страницы и номер строки.



В этом случае сильно увеличивается разрядность схемы сравнения, и появляются неиспользуемые участки в случае длинных кодов до разветвления.

Поэтому наиболее удобным считается наборно-ассоциативная Cache, где страница разбивается на абзацы.





В этом случае в тег входит номер страницы и номер абзаца, что сокращает разрядность компаратора.

Cache данных у всех типов построена аналогично.

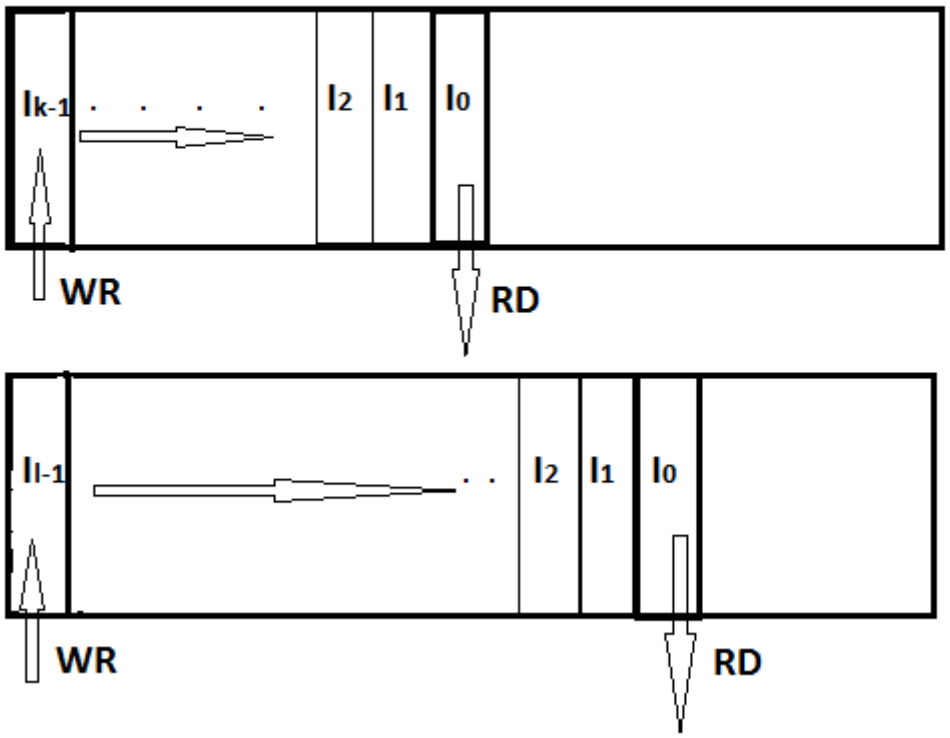
Память с последовательным доступом.

В памяти с последовательным доступом обращение к любой, выбранной по адресу, ячейке невозможно. Просмотр информации возможен лишь в ячейке, принятой за «вершину» такой памяти.

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

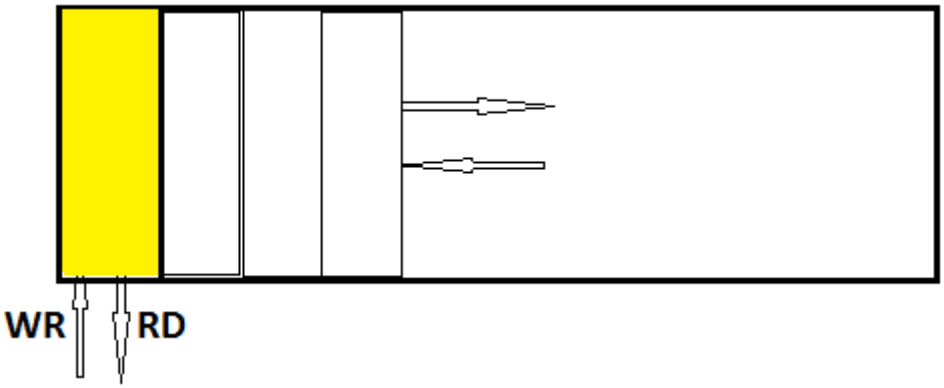
Память FIFO имеет две «вершины»: при записи, и при считывании. В FIFO процессы записи и считывания обсчитываются двумя счетчиками адресов. Оба счетчика суммирующие. Для исключения ложного считывания (чтения ячейки, куда не производилась запись, счетчик записи имеет возможность реверса). Так как глубина FIFO, равно как и глубина стека обсчитываются через указатели, формат которых стандартен для регистров системы, то и глубина памяти с последовательным доступом не может превышать возможности указателя.

FIFO на n ячеек с записью k и l единиц информации

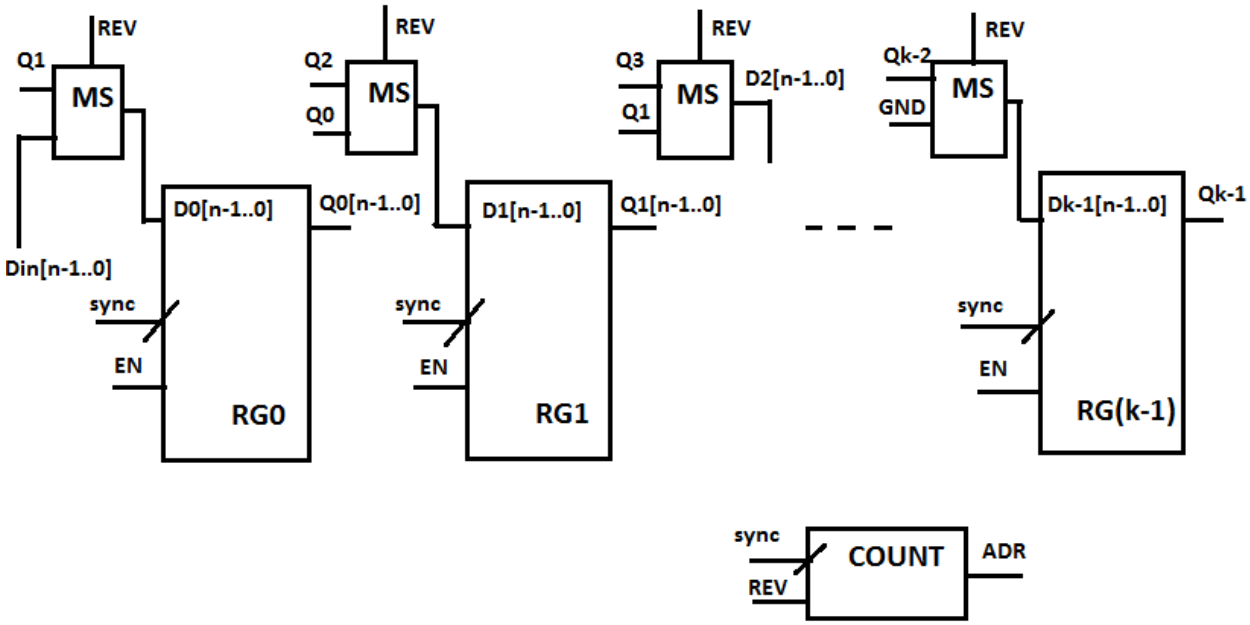


Внешние стрелки показывают информацию, доступную для просмотра.

В LIFO доступна для просмотра только одна точка, называемая «вершиной» стека поэтому адреса записи и считывания обсчитываются одним реверсивным счетчиком.



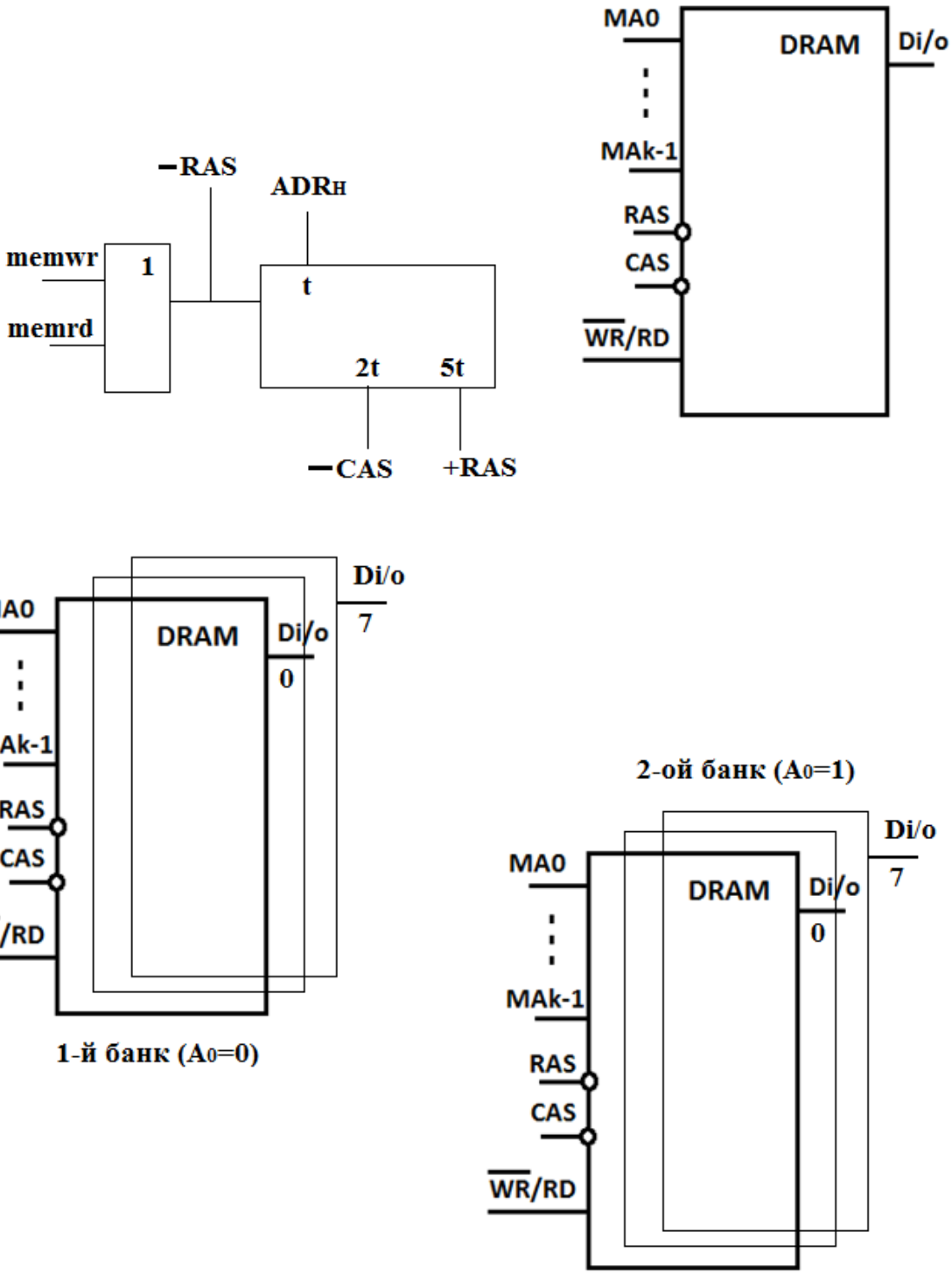
На схеме показана структура n-разрядного стека глубиной k. Вершина стека – RG0.



Адресная память.

В адресной памяти доступ к любой ячейке накопителя возможен по любому выставленному на шине адресу, независимо от предыдущего обращения. Примеры адресной памяти: постоянные запоминающие устройства ROM и оперативные запоминающие устройства RAM.

Динамическая оперативная память.



SDRAM

SDRAM (Synchronous Dynamic Random Access Memory) – это синхронизированная динамическая память с произвольным порядком выборки (синхронная динамическая оперативная память). Появилась она в 1997 году, и все наборы системной логики поддерживают этот тип памяти.

SDRAM состоит из физических ячеек, которые собраны в страницы. Размер страницы может быть от 512 байт до нескольких килобайт. Каждая страница разбита на два банка: в одном банке ячейки с нечетными адресами строк, а в другом – с четными (для ускорения работы памяти). Каждая ячейка имеет свой адрес, состоящий из номера (адреса) строки и номера (адреса) столбца. Сначала передается номер строки, затем номер столбца. В страничном режиме, передав номер строки, можно получить доступ ко всем ячейкам с разными номерами столбцов, то есть не надо для каждой из них передавать номер строки, достаточно только номера столбца – экономится цикл. Такой режим называется Fast Page Mode. Строки делятся на четные и нечетные - получается два банка: один – с четными строками, а другой – с нечетными. Во время обращение к одному банку в другом производится выборка адреса - такой режим иногда называют расслоением.

Для того чтобы увеличить скорость доступа к памяти, используется пакетный режим (burst) доступа: после установки строки и столбца ячейки происходит обращение к следующим трем смежным адресам без дополнительных состояний ожидания. Схема пакетного режима будет выглядеть так: x-y-y-y, где x – время выполнения первой операции доступа состоящей из продолжительности цикла и времени ожидания, а y – это число циклов, необходимое для выполнения каждой последующей операции. Для SDRAM схема будет выглядеть так: 5-1-1-1.

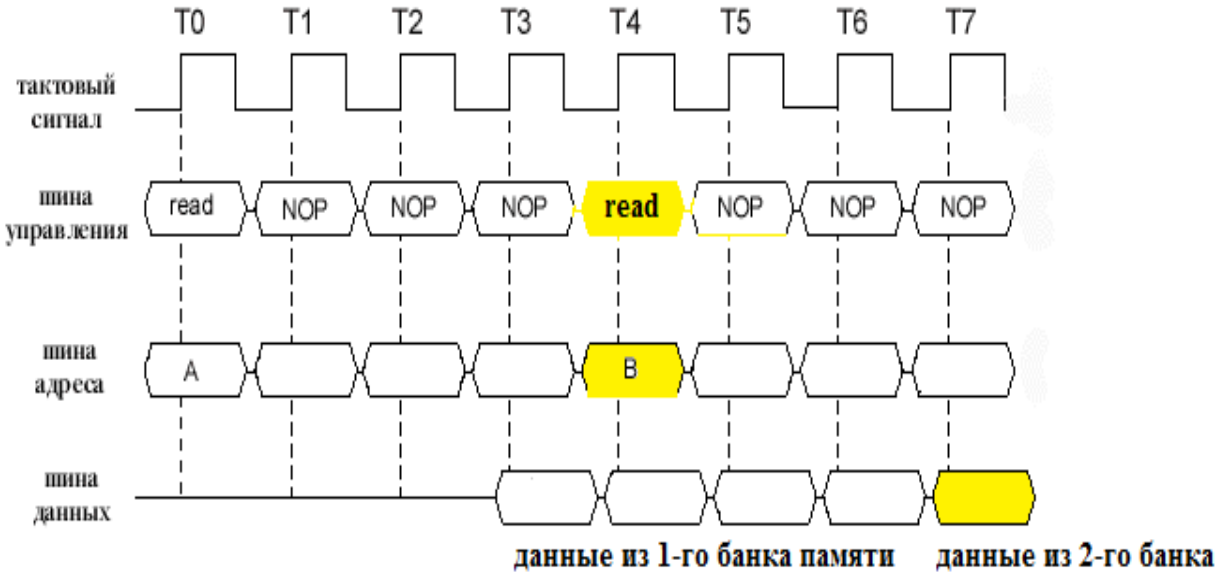


Диаграмма работы SDRAM

Основное отличие SDRAM от предшествующих типов памяти заключается в том, что сигналы ее синхронизированы с тактовым генератором системной платы.

DDR SDRAM

Полное название памяти DDR – DDR SDRAM (Double Data Rate Synchronous Dynamic Random Access Memory – динамическая синхронизированная память с произвольным порядком выборки и удвоенной передачей данных). Этот тип памяти сочетает в себе приемлемую скорость и относительную дешевизну.

Принцип работы DDR SDRAM подобен обычной SDRAM. Память разбита на страницы, каждая страница разбита на банки. Работа памяти синхронизирована с тактовым генератором системной платы. Основное отличие заключается в том, что за один цикл происходит два обращения к данным: по фронту и срезу импульса тактового сигнала системной шины - чтение/запись происходит два раза за один такт.

DDR SDRAM управляется инверсными тактовыми сигналами. Управляющие и адресные сигналы регистрируются по положительному фронту тактового сигнала, а данные передаются по обоим фронтам сигнала. Такая схема работы требует более четкой синхронизации. Для этого введен дополнительный строб-сигнал чтения/записи .

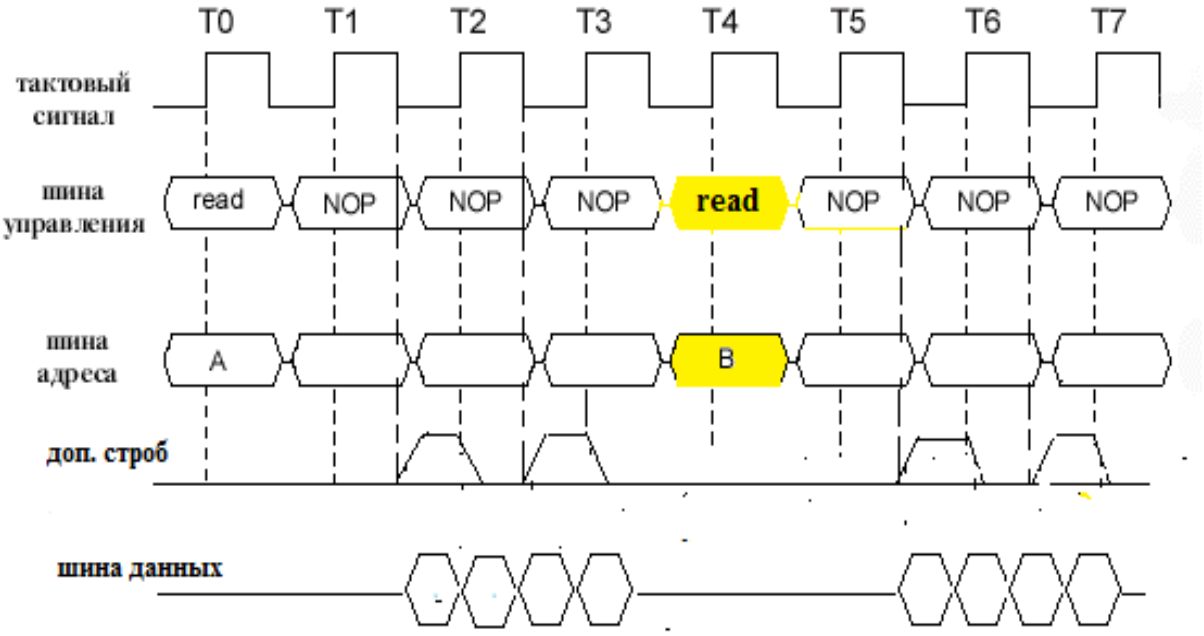


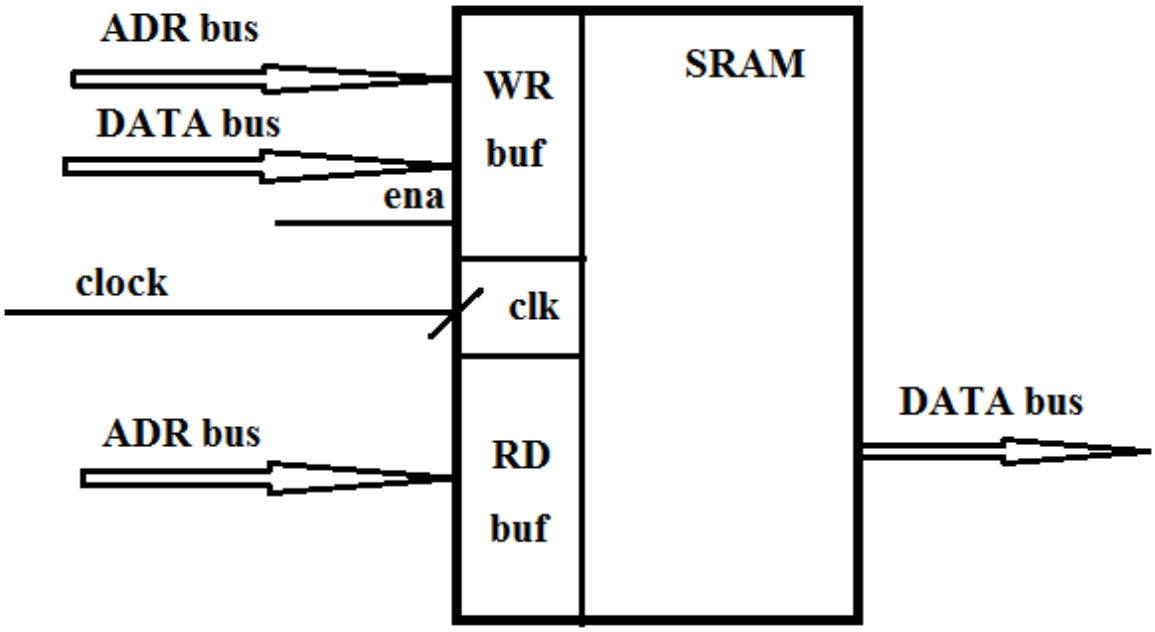
Диаграмма работы DDR SDRAM

Статическая память

SRAM

Статическая память построена на основе триггерных ячеек.

Буферизированная память



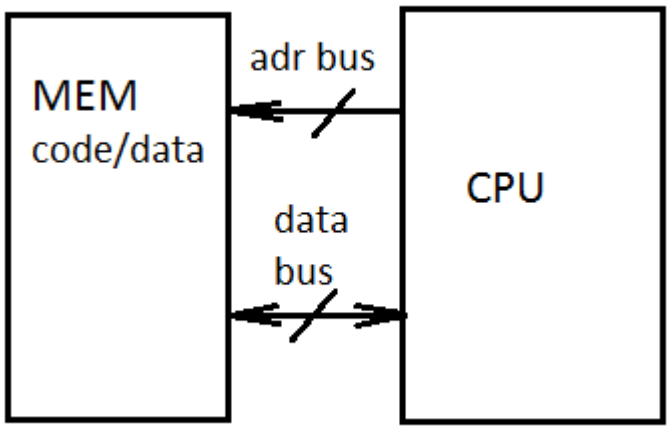
Такая память может быть однобуферной или двухбуферной.

В однобуферной памяти возможно только раздельное обращение к буферам записи и считывания.

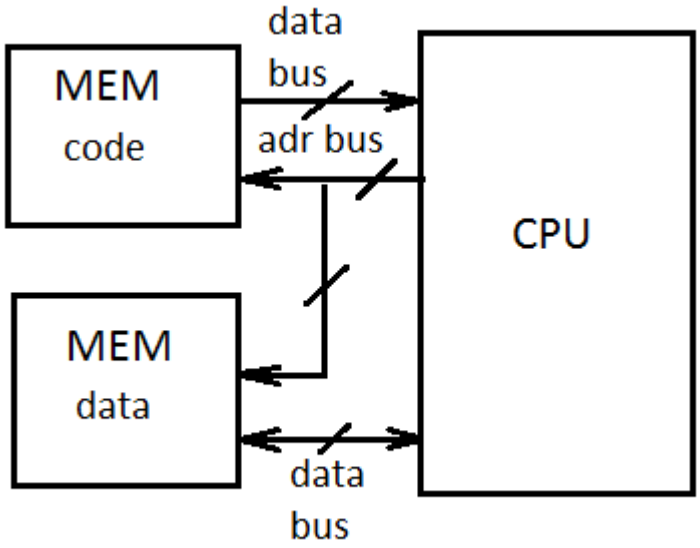
В двухбуферной памяти возможно одновременное обращение к буферу записи и буферу считывания, но по **разным** адресам.

Типы постоянной памяти (ROM, PROM, EPROM и EEPROM) рассматривались в предыдущей лекции. Остается добавить, что в любой постоянной памяти матрица накопителя строками выходит на адресную шину, а столбцами – на шину данных. В любой оперативной памяти строка и столбец матрицы определяются по адресу, выставленному на адресной шине.

В зависимости от типа обмена процессор-память различают два типа архитектуры микропроцессорной системы: Принстонская, или архитектура фон-Неймана и Гарвардская.



Архитектура фон Неймана



Гарвардская архитектура

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,

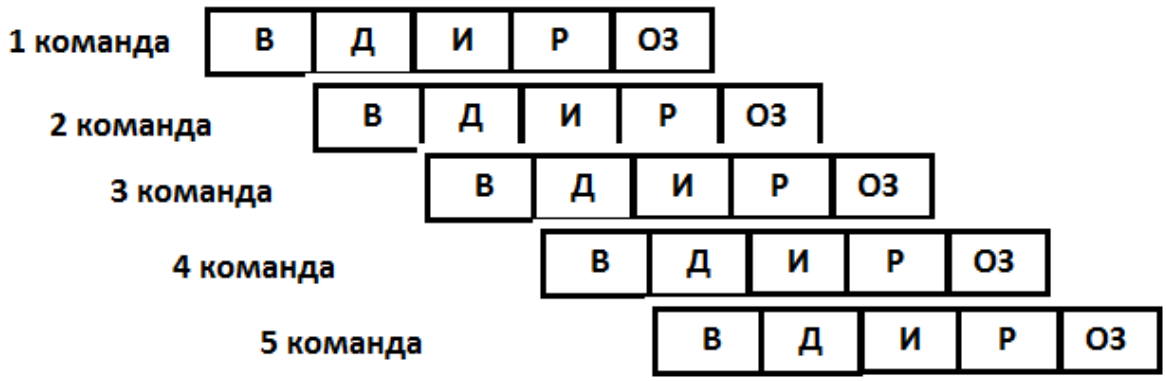
2 – дешифрация кода,

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 8, 9

Структура микропроцессора. Регистры, структура команды.

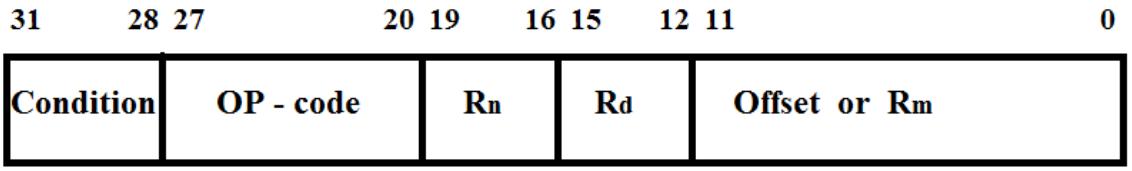
Ядро любого микропроцессора состоит из нескольких блоков. Условно их можно назвать:

1. – блок очереди команд,
2. – блок дешифрации команд,
3. – блок регистров.

4. – блок производства операций, включающий: АЛУ, умножитель, делитель, сдвиговый регистр.
5. –блок управления (системные регистры).

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, этот формат является оптимальным, так как программы обычно построены по разветвленным алгоритмам.

Команды RISC- процессора имеют определенный формат (кратный формату минимальной ячейки памяти). Этот формат разделяется на поля в зависимости от принятого типа кодирования. Например, формат команды прямой загрузки для стандартного Ассемблера ARM-7 будет выглядеть таким образом

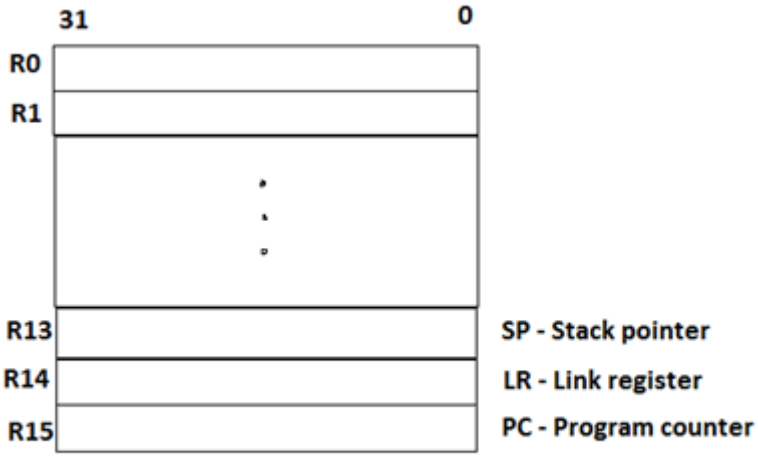


Мы еще вернемся к пояснению смысла записей в полях этого формата.

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

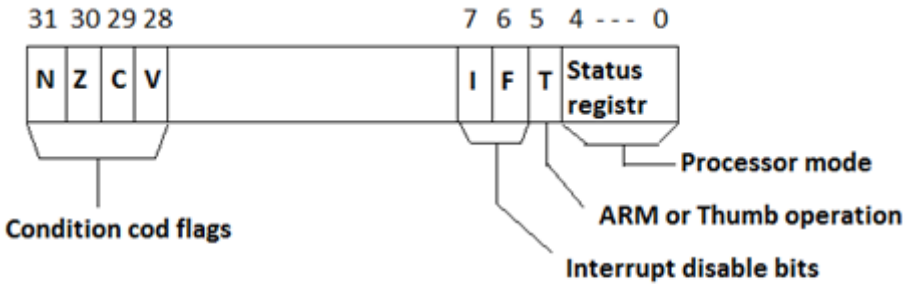
Структура ядра ARM-процессора предполагает блок из 16-ти 32-разрядных регистров общего назначения, 13 из которых не имеют дополнительных специальных функций. Это регистры от R0 до R12. Кроме того, в общее число регистров общего назначения входят указатель стека, указатель связей и программный счетчик. 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. Программный счетчик указывает на адрес

вызываемой из памяти команды.0-й разряд PC всегда установлен в 0.



С помощью команд прямой и обратной загрузки происходит обмен между регистрами общего назначения и памятью данных, а также подключенными устройствами ввода-вывода (device memory).

Регистр текущего статуса (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).

В некоторых типах процессоров существует флажок, называемый **Q** или **SAT**, который фиксирует переполнение, но не копирует в регистр полученный результат. В регистре остается значение верхней границы области используемых чисел - **0x7FFFFFFF** или **0x80000000**.

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

Флаги состояний используются командами условных действий.

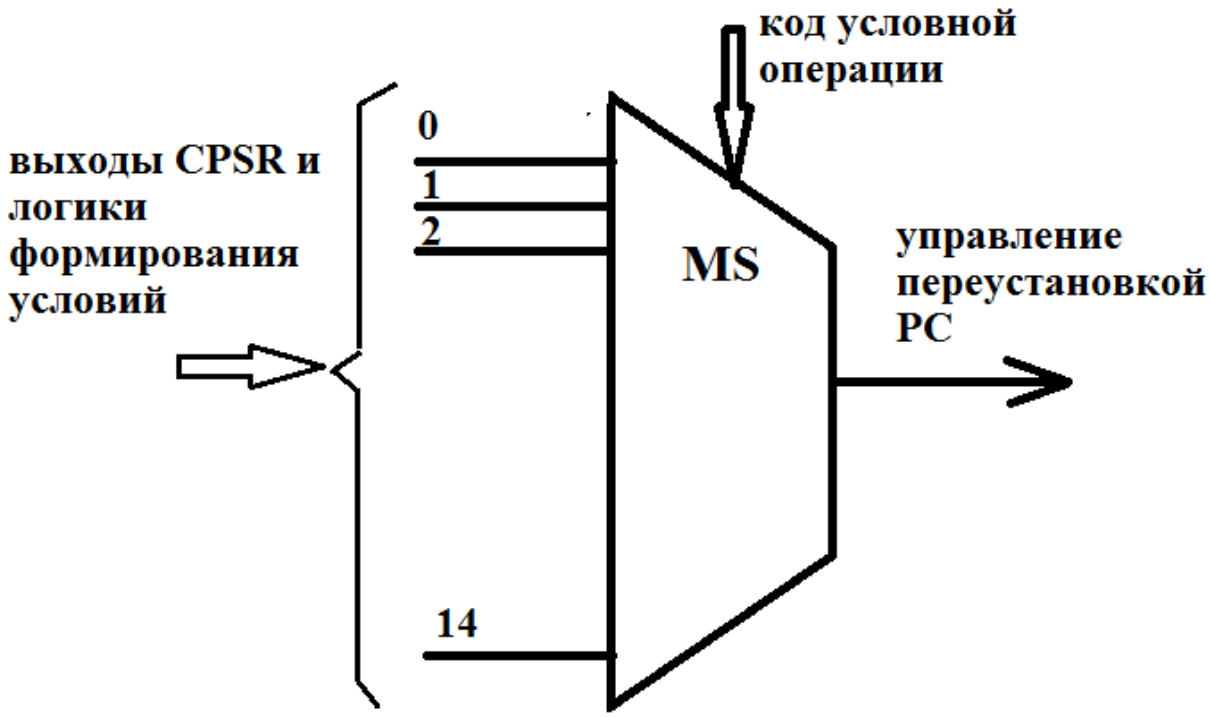
Таблица условий.

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 (higher)	Беззнаковое больше	**nC v Z = 0**
9	LS (lower or same)	Беззнаковое меньше или такое же	**nC v Z = 1**
10	GE (greater than or equal)	Знаковое больше, чем, или равно	$\mathbf{N \oplus V = 0}$
11	LT (less than)	Знаковое меньше, чем	$\mathbf{N \oplus V = 1}$
12	GT (greater than)	Знаковое больше, чем	$\mathbf{Z \vee (N \oplus V) = 0}$

13	LE (less than or equal)	Знаковое меньше, чем или равно	$Z \vee (N \oplus V) = 1$
14	AL (always)	Всегда	

Последнее условие указывается для всех безусловно выполняемых команд.

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



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.

Примеры команд Ассемблера. Форма записи.

Команды прямой и обратной записи.

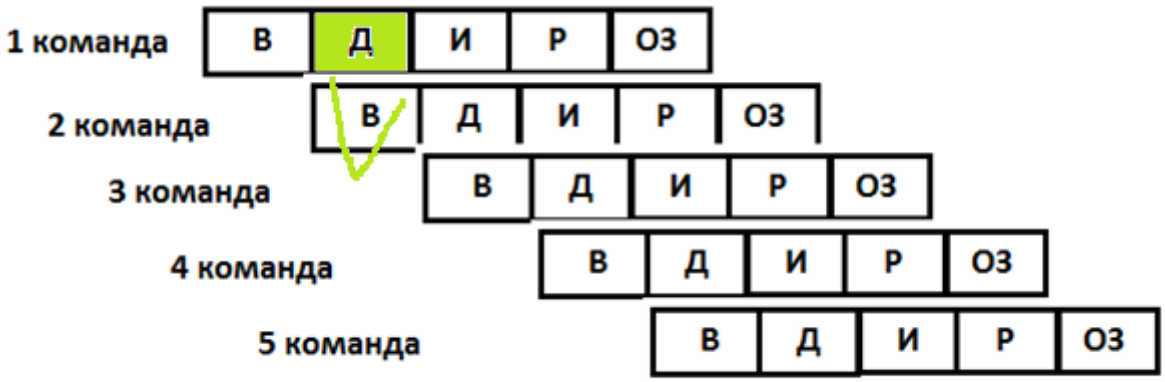
Мнемоника операции прямой загрузки: **LDR – Load Register**, мнемоника операции обратной загрузки: **STR – Store Register.** Формат данных может быть: слово, полуслово или байт. Соответствующие мнемоники:

LDRB (Load Register Byte)
LDRSB (Load Register Signed Byte)
LDRH (Load Register Halfword)
LDRSH (Load Register Signed Halfword)

Отличие загрузки знакового или беззнакового операнда, формат которого меньше, чем слово, в расширении (знаком или нулем).
Соответственно, для обратной загрузки

STRB (Store Register Byte)
STRH (Store Register Halfword)

Операнды: – возвращаемся к иллюстрации формата команды.
Регистр процессора (R_d).
Ячейка памяти. Адрес ячейки формируется на основе базового, содержащегося в регистре (R_n). Смещение (offset), прибавляемое к базовому адресу, может записываться или непосредственно, или как содержимое индексного регистра (R_m). Причем содержимое индексного регистра может задаваться со сдвигом вправо или влево на указанное число позиций (разрядов).
Условие в формате команды всегда должно быть записано как 1110 – AL(см. таблицу условий).
Надо понимать, что в качестве R_d , R_n и R_m могут использоваться любые регистры
общего назначения. Использование программного счетчика для задания адреса возможно лишь в качестве базового. При этом к текущему состоянию PC добавляется 8, т.к. команда, идущая по следующему адресу, уже поступила на конвейер



Примеры:

LDR R2, [R6, # -20] – загрузка в R2 по адресу, записанному в R6, минус 20
LDR R2, [R6, R7]
LDR R2, [R6, R8, LSL# 2]
(LSL – Logic Shift Left, LSR – Logic Shift Right)
LDR R5, MEMLOC === LDR R5, [R15, # offset]
STR R5, [R2].

Для сравнения, обращение к памяти в Ассемблере 8086/88 делалось с помощью команд пересылки MOV, но с различными способами адресации

MOV AX, [BX] – один байт кода операции;

MOV AX, [BX + disp] – в зависимости от смещения – 2 или 3 байта.

При работе команд, использующих АЛУ (арифметико-логическое устройство), операнды поступают только из внутренней памяти ядра, что экономит время обработки. Для команд RISC-процессора, выполняющихся за один такт, все КЦУ, участвующие в процессе обработки, как коммутирующие, так и кодопреобразующие, должны образовывать цепочки, имеющие общую задержку, не превышающую время такта.

Команды, исполняющиеся в АЛУ, имеют следующую форму записи:

OP R_d, R_n, Operand 2

В качестве второго операнда может быть число определенной разрядности, содержимое регистра, или содержимое регистра, сдвинутое на заданное количество разрядов.

OP R_d, R_n, #op2
OP R_d, R_n, R_m
OP R_d, R_n, R_m, LSL #3
OP R_d, R_n, R_m, LSL R5

ADD R_d, R_n, R_m , ADD R_d, R_n,#20

MOV R_d, R_n, MOV R_d, #54

Для сравнения, в Ассемблере 8086 такие команды могли использовать операнд из пространства памяти.

ADD AX, BX и ADD AX, [BX]

Лекция 10,11

Примеры команд (продолжение)

Команды, формирующие условия и команды условных действий.

Формируют условия (устанавливают флаги) команды арифметических действий первой ступени и логических действий, если это допускается синтаксисом (ADD или ADDS, SUB или SUBS, AND или ANDS) и команды пересылок с тем же суффиксом S - MOVS.

ADD R2, R3, R1

ADDS R2, R3, R1

SUB R1, R1, #1

SUBS R1, R1, #1; используется как команда декремента в организации циклов.

Команды сдвигов при необходимости тоже могут выставлять флаги.

Кроме того, существуют команды сравнения: арифметического – CMP и логического TEQ, на основе поразрядного сложения по модулю 2. А также

команда, позволяющая тестировать выделенные разряды на наличие 0 с установкой флага – TST.

CMP R3, R0

CMP R1, #0

TEQ R3, R4

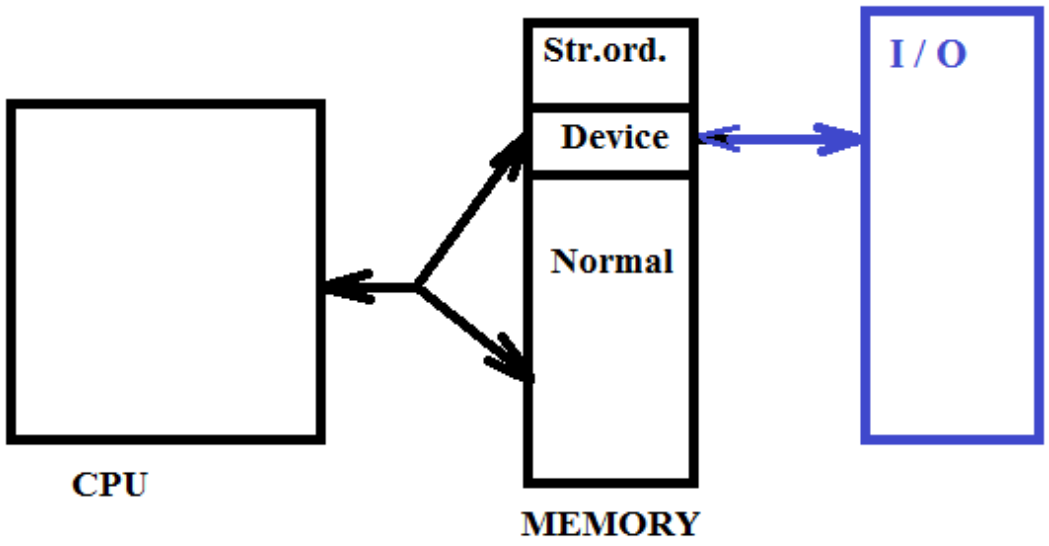
TST R5, #0x00000040

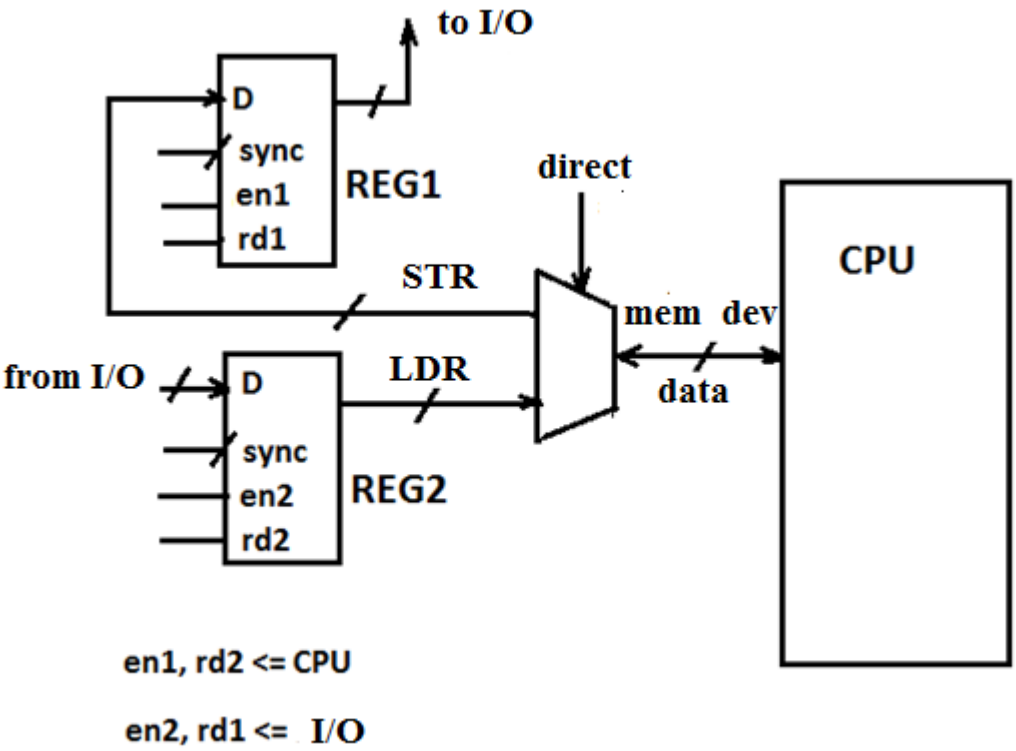
Используют условия команды условных действий: условные переходы, условные перемещения или операции арифметики первой ступени, выполняемые только при наличии указанного условия.

Например: ADDEQ R_d, R_n, Operand 2.

Устройства ввода-вывода. Последовательный обмен.

Выход на устройства ввода-вывода производится через область памяти устройств. В этой области находятся параллельные порты, через которые подключаются регистры периферийных портов, часть этой области сконфигурирована как FIFO. Обмен производится с помощью команд прямой и обратной загрузки.





Передача данных по внешней линии всегда производится последовательно. Поэтому в структуре устройств ввода-вывода необходимо преобразование информации для приема последовательно-параллельное, для передачи – параллельно-последовательное. Эти процессы могут обрабатываться или с помощью комбинированных регистров, или используя коммутационные схемы: мультиплексоры и демультиплексоры (сериализация и десериализация).

UART

Последовательный порт, функционирующий с небольшими изменениями более 50 лет. С 1962 года он работал по физическому протоколу RS232. В настоящее время существуют как варианты отдельных микросхем, так и встроенные в микроконтроллеры.

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 (битах в секунду). Существует общепринятый ряд стандартных скоростей: 300; 600; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200; 230400; 460800; 921600 бод.

Основные регистры данных последовательные регистр приема и регистр передачи.

Кроме регистров приема и передачи имеет два адресуемых делителя частоты (старший и младший байты адресуются отдельно), регистр управления линией, регистр состояния линии, регистр разрешения прерываний и регистр идентификатора 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 (постоянная передача 0). 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

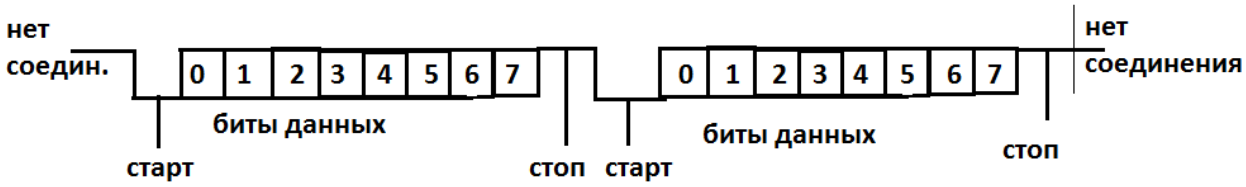
Многие реализации UART имеют возможность автоматически контролировать целостность данных методом контроля битовой чётности. Когда эта функция включена, последний бит данных в минимальной посылке («бит чётности») контролируется логикой UART и содержит информацию о чётности количества единичных бит в этой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

(*Even parity*), когда сумма количества единичных бит в посылке является четным числом, и контроль на нечетность (*Odd parity*), когда эта сумма нечетна. При приеме такой посылки UART может автоматически контролировать бит четности и выставлять соответствующие признаки правильного или ошибочного приема. Бит четности получают через побитовый XOR информационных бит в посылке.

Контроль чётности (пример для 7-битной посылки)

данные	**Количество единиц**	**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. р2 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, если предполагается проверка на четность, и 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).

Для формирования временных интервалов передающий и приёмный UART имеют **источник точного времени** (тактирования). Точность этого источника должна быть такой, чтобы сумма погрешностей (приёмника и передатчика) установки временного интервала от начала стартового импульса до середины стопового импульса не превышала половины (а лучше хотя бы четверти) битового интервала.

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 загруженности регистров передачи: буферного и регистра сдвига (загружен/пуст), о прерывании отсоединения линии, об ошибках приема (ошибка паритета или не принята стоповая посылка), о переполнении регистра приема и загруженности буфера приема. Регистр состояния доступен только для чтения.

Регистр идентификатора прерываний хранит код условия прерывания (маску) и признак запроса прерывания. Прерывания могут возникать из-за ошибок приема (ошибка четности или не принят стоп), из-за заполненности регистра приема или отсутствия данных в регистре передачи.

Регистр разрешения прерывания управляет формированием запроса в соответствии с кодом идентификации.

UART (USART) как автономный интерфейс практически не применяется.

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.

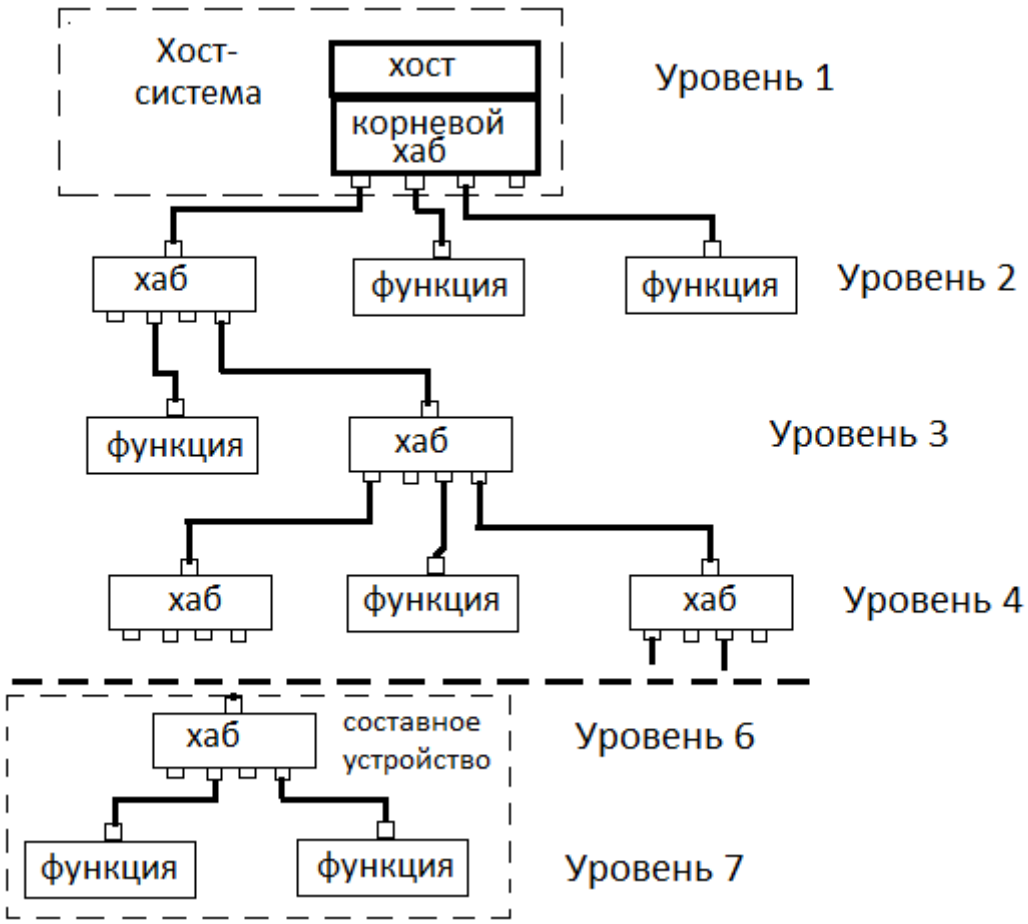
Структура USB

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.

Функция не может сама инициировать обмен, весь обмен управляется хостом. Контроллер хоста периодически опрашивает подключенные и резервные устройства на предмет обмена, рассылая пакет начала кадра и составляет списки заявок. Если устройство готово к обмену, между ним и хостом организуется канал,

т.е. программно поддерживаемое соединение. Так как функции представляют собой сложные схемы, в которых выдачей и приемом информации могут заниматься различные узлы, то внутри функции выделяются конечные точки. Таким образом, канал организуется между хостом и конечной точкой. Обмен всегда производится между хостом и конечной точкой при поддержке контроллера хаба.

Данные по USB шине всегда передаются младшими битами вперед. USB пакет состоит из нескольких полей.

Любой тип пакета должен содержать поле синхронизации, поле идентификатора пакета и поле конца пакета.

Пакет-признак кроме этих полей должен содержать поле адреса устройства, поле адреса конечной точки и поле контроля (циклический контроль по избыточности).

Пакет с данными должен содержать поле данных и поле контроля.

Пакет согласования содержит только три основных поля.

Пакет начала кадра должен содержать поле номера кадра и поле контроля.

- **Поле синхронизации (Sync Field)**

Все пакеты должны начинаться с поля синхронизации. Поле синхронизации имеет размер 8 бит для низкоскоростных и полноскоростных устройств или 32 бита для высокоскоростных устройств и используется для подсинхронизации тактового генератора, встроенного в USB контроллер. Последние два бита поля синхронизации являются маркером, который используется для идентификации конца области синхронизации и начала PID поля.

- **Поле идентификатора пакета (PID Field)**

Идентификатор пакета следует непосредственно после поля синхронизации в каждом передаваемом USB пакете. PID состоит из 4-битного поля типа пакета, следующего за 4-битным собственным проверочным полем .

- **Поле адреса устройства (Addr Field)**

Поле адреса используется для идентификации направления к устройству, в которое направлен текущий пакет. Размерность поля - 7 бит, что позволяет адресовать 127 уникальных USB устройств. После сброса или включения питания, адрес устройства устанавливается по умолчанию в 0 и должен быть запрограммирован хостом в ходе процесса присвоения номера. Адрес 0 (заданный по умолчанию) зарезервирован для вновь подключаемых устройств и не может быть назначен для нормальной работы.

- **Поле адреса конечной точки (Endpoint Field)**

Поле адреса конечной точки имеет размерность 4 бита и позволяет размещать в устройстве до 16 конечных точек. Конечная точка это логический адрес, образованный внутри устройства. Связь хоста и конечной точки производится по каналу, представляющему собой логическое образование.

· **Поле номера кадра (Frame Number Field)**

Поле номера кадра представляет собой 11-битное поле, которое инкрементируется хостом при инициировании нового кадра. Поле номера кадра начинается заново с нуля при достижении максимального значения 0x7FF, и используется только для SOF маркеров в каждом начале кадра.

· **Поле циклического контроля по избыточности (CRC)**

Циклический контроль по избыточности (CRC) используются для защиты всех полей кроме PID в маркерах и пакетах данных. Защита Маркера и пакета данных обеспечивает 100% нахождение всех одиночных и двойных битовых ошибок. Для маркеров предусмотрено 5-битное поле CRC, которое используется для защиты полей ADDR и ENDP пакетов IN, SETUP,OUT или поле отметки времени маркера SOF. Для пакета данных используется 16-битный полином, кодирующий всё поле данных пакета.

· **Поле конца пакета (EOP)**

Поле конца пакета представляет собой сигнал окончания пакета и устанавливается на шине путем выставления сигнала уровня 0 "Single Ended Zero " SE0 в течение двух битовых интервалов следующего за 1-м состоянием, установленным на время одного битового интервала.

Описание типов USB пакетов

Для USB шины позиционируется четыре типа пакетов. Пакеты-признаки обозначают, какого типа транзакция начинается на шине, пакет с данными включает в себя передаваемую структуру данных, пакеты согласования используются для информирования об успешно выполненной транзакции или ошибках произошедших при передаче, и пакеты начала кадра инициируются при генерации нового кадра на шине.

· **Пакеты-признаки (Token Packets)**

Существуют три типа таких пакетов:

1. **In** - информируют USB устройство, что хост хочет читать данные из устройства
2. **Out** - информирует USB устройство, что хост хочет передавать данные в устройство
3. **Setup** - используются для обозначения начала управляющего (Control Transfer) типа передачи данных

Все пакеты-признаки имеют следующий формат:

SYNC PID ADDR ENDP CRC5 EOP

· **Пакеты данных (Token Packets)**

Существуют два типа пакетов данных, каждый из которых способен содержать до 1024 байтов данных.

1.DATA0
2.DATA1

У высокоскоростных устройств пакеты данных имеют два других идентификатора: DATA2 и MDATA. Пакет данных имеет следующий формат:

SYNC PID DATA CRC16 EOP

- Максимальная полезная загрузка для низкоскоростных устройств составляет 8 байт.

- Максимальная полезная загрузка для полноскоростных устройств составляет 1023 байта.

- Максимальная полезная загрузка для высокоскоростных устройств составляет 1024 байт.

- Данные всегда посылаются целым числом байт.

· **Пакеты согласования (Handshake Packets)**

Существуют три типа пакетов согласования, структура которых имеет только PID поле:

- ACK - подтверждение того, что пакет был успешно принят

- NAK - информирует, что устройство в данный момент не может принимать либо отправлять данные. Также используется в Interrupt транзакциях для информирования хоста, что устройство не имеет новых данных для передачи. Хост никогда не может выдавать ответ NAK. NAK используется в целях управления потоком данных.

- STALL - указывает, что устройство неспособно передавать или получать данные, и требуется вмешательство хоста для снятий состояния останова. Как только конечная точка устройства остановлена, устройство должно продолжить возвращать STALL, пока условие, вызвавшее останов не будет удалено с помощью вмешательства хоста. Хосту запрещено возвращать STALL.

Пакет подтверждения имеет следующий формат:

SYNC PID EOP

· **Пакеты начала кадра (Start of Frame Packets)**

SOF пакет состоит из 11-ти битного номера кадра и генерируется хостом каждую 1ms ± 500ns для полноскоростной шины и каждые 125 μs ± 0.0625 μs для высокоскоростной шины.

Пакет начала кадра имеет следующий формат:

SYNC PID Frame Number CRC16 EOP

Типы каналов.

· Каналы сообщений. Являются двунаправленными каналами. Возникает канал при отсылке хостом запроса в устройства, и управляет передачей только хост. Каналы сообщений используется для передач только управляющего типа.

· Потоковые каналы. Являются однонаправленными. Эти передачи могут контролироваться не только хостом, но и устройством. Используется для передач данных типа прерывание, групповая пересылка, изохронная.

В зависимости от типа передаваемых данных, предъявляемых требований к скорости обработки, задержки доставки и т.п. определены следующие типы передач.

· Управляющие передачи. Используются для конфигурирования устройств во время подключения и выполнения других специфических функций над устройством, включая организацию новых каналов.

· Прерывания. Используются для спонтанных, но гарантированных передач с гарантированными скоростями и задержками. Используются обычно для передачи введенных данных от клавиатуры или сведений об изменении положения указателя мыши, в устройствах обратной связи, и т.д.

· Групповая пересылка. Используется для гарантированной передачи данных больших объемов без предъявленных требований к скоростям и задержкам. Занимает под себя всю свободную пропускную способность шины. В любой момент доступная полоса может быть урезана при необходимости осуществления передач других видов с более высоким приоритетом, или добавлена, при освобождении другими устройствами. Обычно такие передачи используется между принтерами, сканерами, накопителями и др.

· Изохронные передачи. Используются для потоковых передач данных в реальном времени. Резервируют определенную полосу пропускания шины, гарантируют определенные величины задержек доставки, но не гарантируют доставку (в случае обнаружения ошибки повторной передачи не происходит). Передачи этого вида используются для передачи аудио и видео трафика.

Приоритеты передач по USB-шине

Все операции по передаче данных инициируются хост-системой независимо от того, принимает ли она данные или пересылает в периферийное устройство. Все операции, которые необходимо выполнить, хранятся в виде четырех списков по типам передач:

- изохронные передачи;

- передачи прерываний;

- передачи управляющих команд;

- передачи данных больших объемов.

Списки постоянно обновляются новыми запросами. Планирование операций по передаче информации в соответствии с упорядоченными в виде списков запросами выполняется хостом с 1-миллисекундным интервалом. В начале каждого такого интервала хост посылает по шине пакет SOF (StartOf Frame - начало кадра), после чего начинается обслуживание запросов из списка изохронных передач (т.к. они имеют наивысший приоритет).

После того, как все запросы из этого списка будут обслужены, хост-система переходит к списку операций передачи прерываний, затем к списку запросов на передачу данных большого объема.

По истечении 90% указанного 1-миллисекундного интервала хост автоматически переходит к обслуживанию запросов на передачу управляющих команд независимо от того, успел ли он полностью обслужить другие три списка или нет.

Тем самым гарантируется, что управляющим передачам всегда будет выделено не менее 10% пропускной способности шины. Если передача всех управляющих пакетов будет завершена до истечения выделенной для них доли интервала планирования, то оставшееся время будет использовано хостом для передачи данных большого объема (до конца указанного 1-миллисекундного интервала).

Таким образом:

- изохронные передачи гарантированно получают 90% пропускной способности шины;

- передачи прерываний занимают оставшуюся часть изохронных операций часть этой 90-процентной доли;

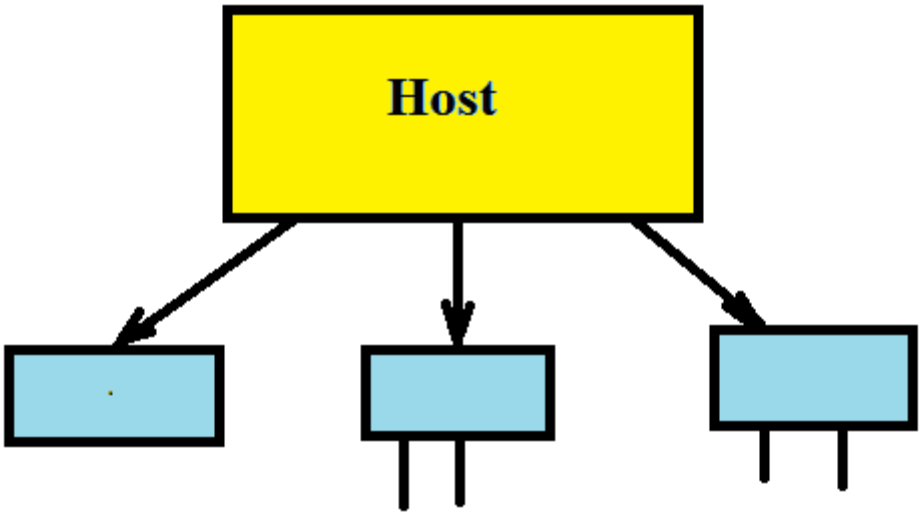
- под передачу данных большого объема выделяется все время, оставшееся после изохронных передач и передач прерываний (по-прежнему в рамках 90% доли пропускной способности);

- управляющим передачам гарантируется 10% пропускной способности;

- если передача всех управляющих пакетов будет завершена до истечения выделенного для них 10% интервала, то оставшееся время будет использовано для передачи данных большого объема.

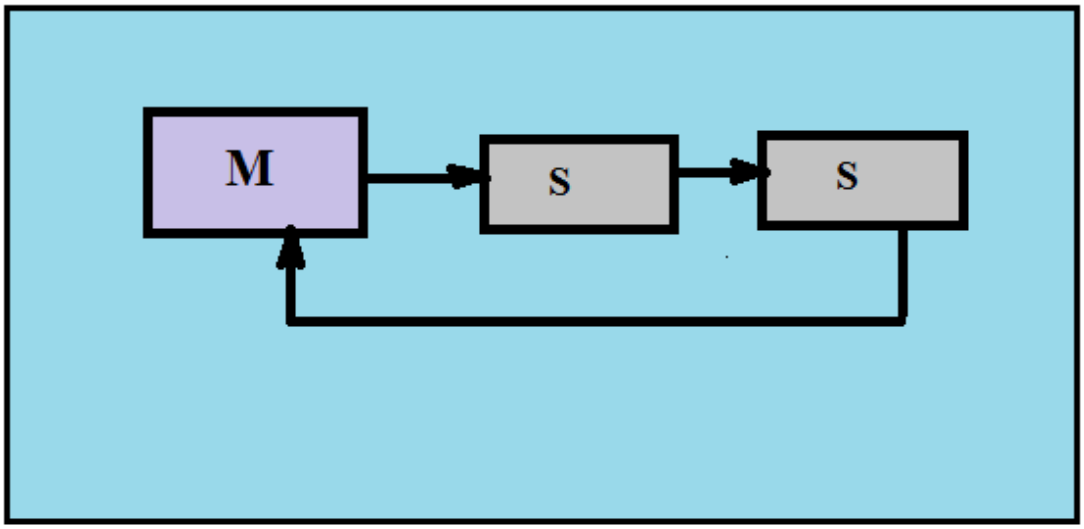
Лекция 12.

Мы рассмотрели шину USB, используемую для связи различных цифровых устройств.



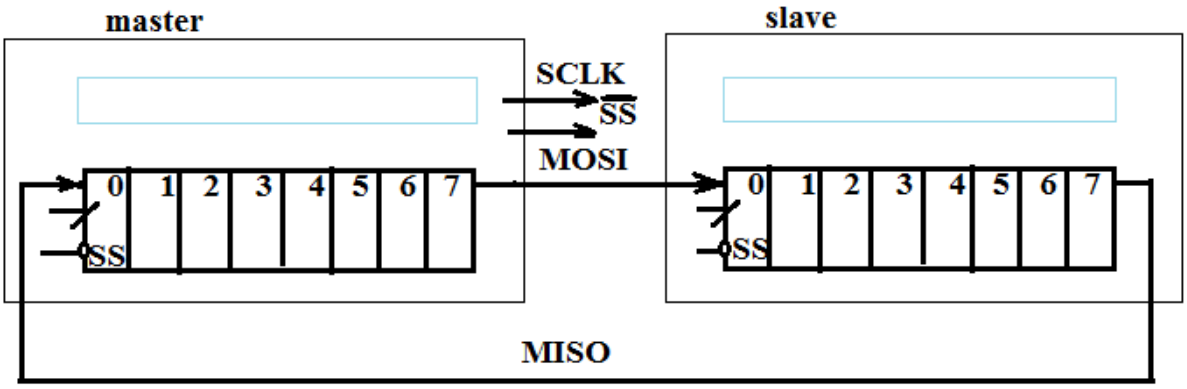
Определяя понятие «канал», мы дали определение конечной точки как блока функции, который будет участвовать в обмене информацией. Таким образом, мы объявили функцию сложным устройством, внутри которого необходимо также регулировать обмен информацией. Действительно, любая современная цифровая система нуждается в устройстве обмена между ее составляющими.

К устройствам, поддерживающим такой обмен, относятся шины SPI и I2C.



Интерфейс (шина) SPI.

Последовательный периферийный интерфейс (шина) служит для обмена информацией между микросхемами. Построение шины основано на строгой иерархии «ведущий – подчиненные». В качестве ведущего обычно выступает микроконтроллер, а подчиненными могут быть различные схемы памяти, таймеры, АЦП и ЦАП и т.п. Организация представляет собой цепь из последовательных регистров, выделяемых в пространстве объединяемых микросхем.

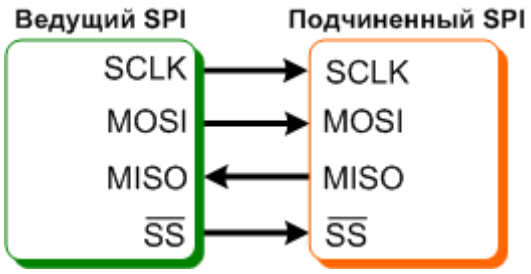


Для выбора ведомого ведущее устройство посылает в его направлении низкий уровень по линии slave select (SS).

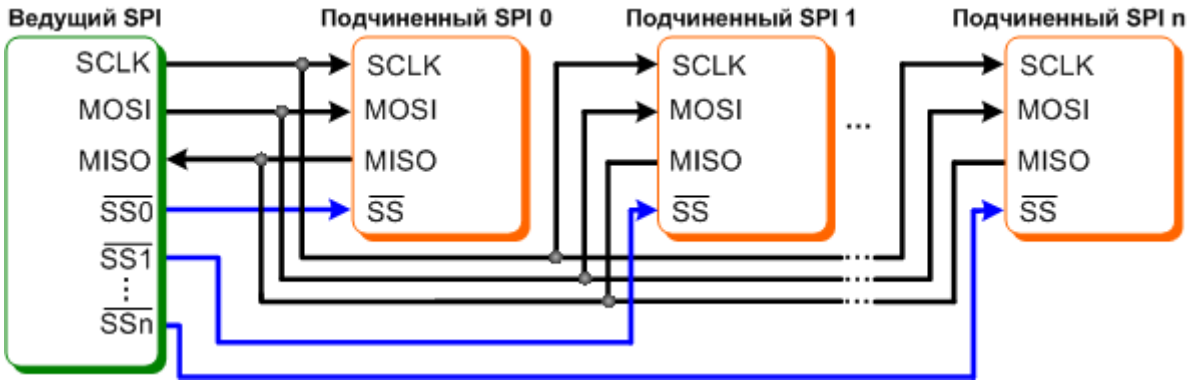
Передача и прием ведутся одновременно, пакетами. Чаще всего длина пакета составляет 8 бит, но это не является обязательным условием. Сдвиг производится по тактовой частоте, генерируемой ведущим устройством. Ведомые устройства используют синхросигнал для определения моментов изменения битов на линии данных, при этом ведомые устройства никак не могут влиять на частоту следования битовых интервалов. Как в ведущем устройстве, так и в ведомом устройстве имеется счетчик импульсов синхронизации (битов). Счетчик в ведомом устройстве позволяет последнему определить момент окончания передачи пакета. Счетчик сбрасывается при выключении подсистемы SPI, такая возможность всегда имеется в ведущем устройстве. В ведомом устройстве счетчик обычно сбрасывается деактивацией интерфейсного сигнала SS.

Так как действия ведущего и ведомого устройства тактируются одним и тем же сигналом, то к стабильности этого сигнала не предъявляется никаких требований, за исключением ограничения на длительность полупериодов, которая определяется максимальной рабочей частотой более медленного устройства. Это позволяет использовать SPI в системах с низкостабильной тактовой частотой, а также облегчает программную эмуляцию ведущего устройства.

Самым простым будет подключение только одного ведомого устройства.

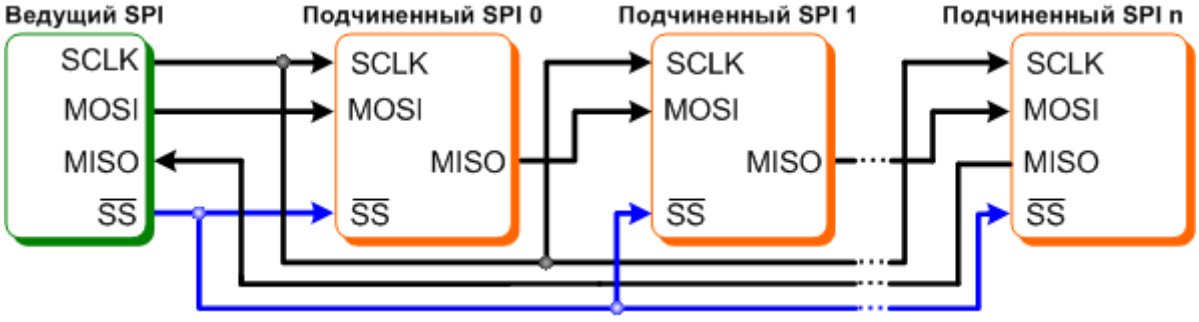


Если в структуре обмена присутствует несколько ведомых устройств, то их подключения к ведущему возможны одним из 2-х способов: независимое подключение или каскадное подключение.



Независимое подключение к шине SPI.

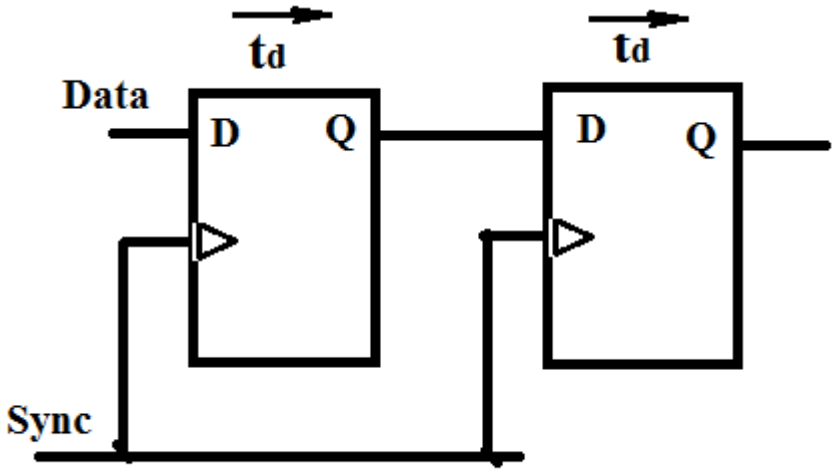
При таком подключении ведущему необходимо иметь несколько выходов для формирования сигналов выбора подключаемого ведомого. Если это нежелательно, то используется каскадное подключение.



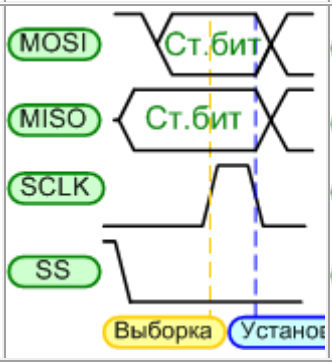
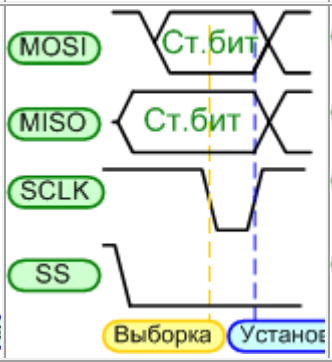
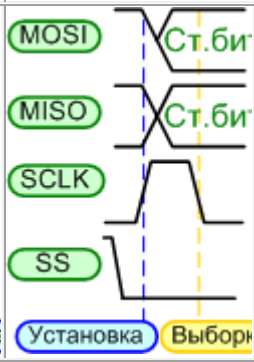
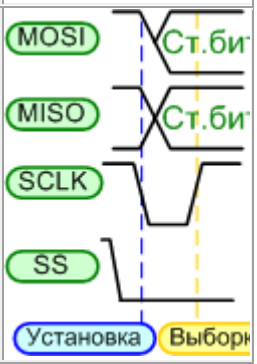
Каскадное подключение к шине SPI(возможно при совместимости ведомых устройств по режимам работы).

Режимы работы SPI

Функционирование любого регистра сдвига предполагает установку информации в каждом разряде с некоторой задержкой относительно ее выборки. Любое изменение уровня импульса синхронизации (фронт или спад) происходит за какое-то конечное время, которое не должно превышать время задержки триггера. Поэтому, для предотвращения сбоев при сдвиге информации, задержку триггера увеличивают (преимущественно за счет увеличения выходной емкости).



В регистрах шины SPI выборка и установка производятся по противоположным фронтам импульса синхронизации. Исходя из этого, SPI имеет 4 режима работы.

Режим SPI	0	1	2	3
CPOL	0	1	0	1
CPHA	0	0	1	1
Временная диаграмма первого цикла синхронизации				

CPOL – исходный уровень полярности сигнала синхронизации;

CPHA – исходная фаза цикла обмена.

Преимущества интерфейса SPI.

Полнодуплексная передача данных по умолчанию.

Возможность произвольного выбора длины пакета.

Возможность использования в системах с низкостабильной тактовой частотой.

Адрес ведомого устройства не передается в структуре пакета.

В отличие от параллельных интерфейсов имеет только 4 вывода.

Недостатки.

Ведомое устройство не может управлять потоком данных.

Нет подтверждения приема данных со стороны ведомого устройства (ведущее устройство может передавать данные «в никуда»).

Нет определенного стандартом протокола обнаружения ошибок.

Наличие множества вариантов реализации интерфейса.

Отсутствие поддержки горячего подключения устройств.

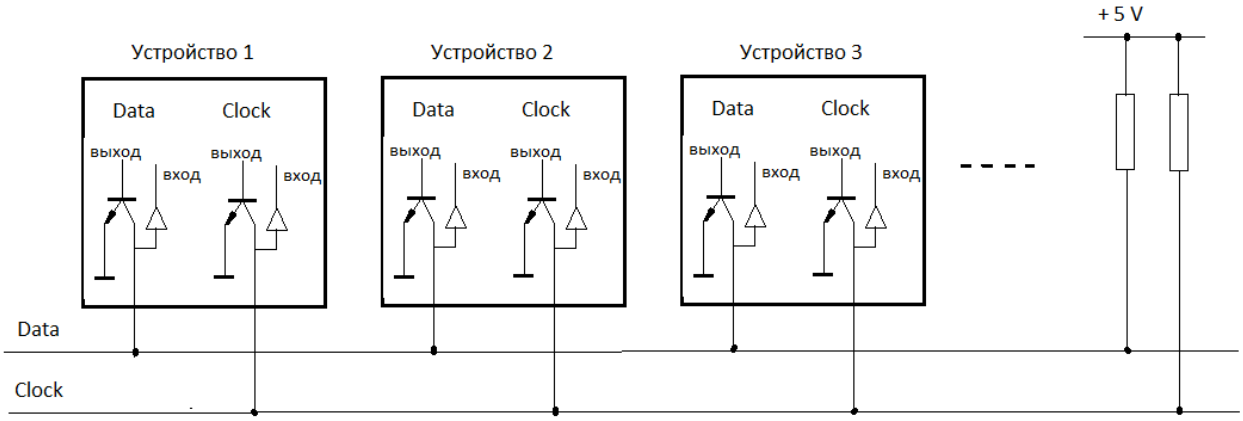
Необходимо больше выводов, чем для интерфейса I2C.

Интерфейс (шина I2C - Inter-integrated circuit bus)

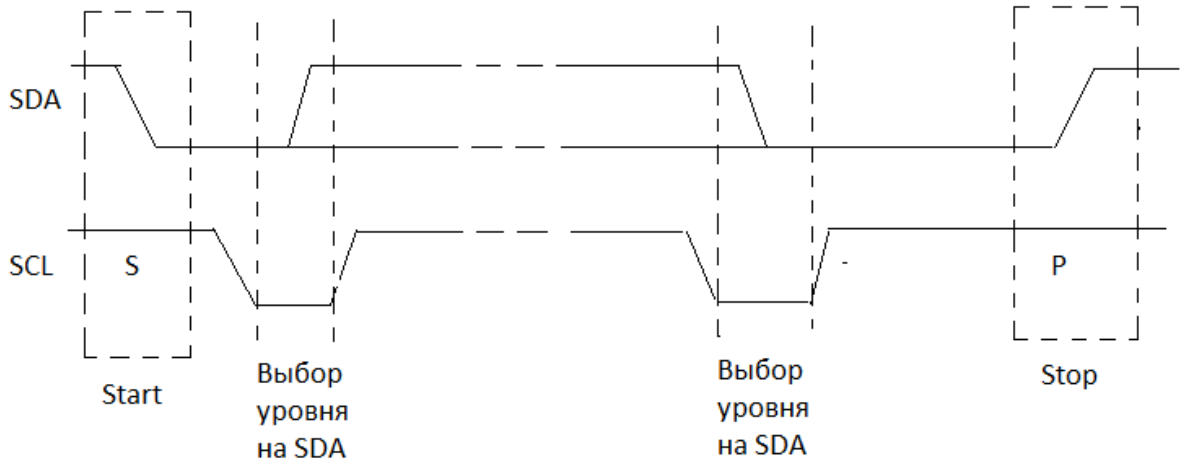
Сетевой последовательный интерфейс, осуществляющий связь ведущих и ведомых по 2-проводной схеме. К шине может быть подключено до 127 устройств, если это не превысит предельную емкость (400 пф).

По скорости передачи различают три режима работы шины: стандартный (S) - 100Кбит/сек, быстрый (F)– 400Кбит/сек и высокоскоростной режим (Hs) – до 3,4Мбит/сек. Первые два режима функционируют практически одинаково. Передача пакетная, объем пакета 8 бит.

Электрическое подключение к проводам.



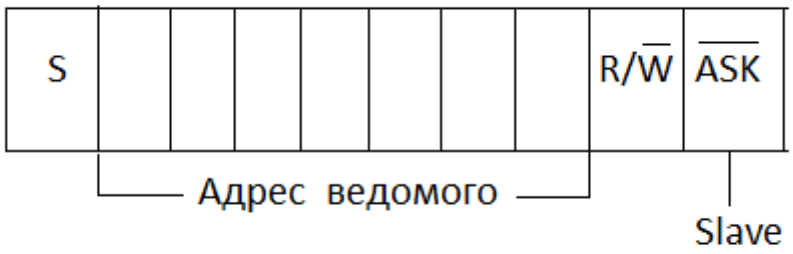
Изначально провода данных и синхронизации подтянуты к «1». Ведущее устройство, первым выставившее на линию **данных** «0» при неактивной линии **синхронизации** может начинать обмен. Синхронизацию всегда генерирует ведущее устройство. При низком уровне SCL происходит выбор данных и их передача, при высоком – считывание приемником. Первая посылка всегда идет от ведущих и содержит адреса ведомых, при этом определяется и направление передачи. Далее следует пересылка пакетов информации. По окончании передачи ведомый должен передать в позиции подтверждения «1», после чего ведущий отпускает линию.

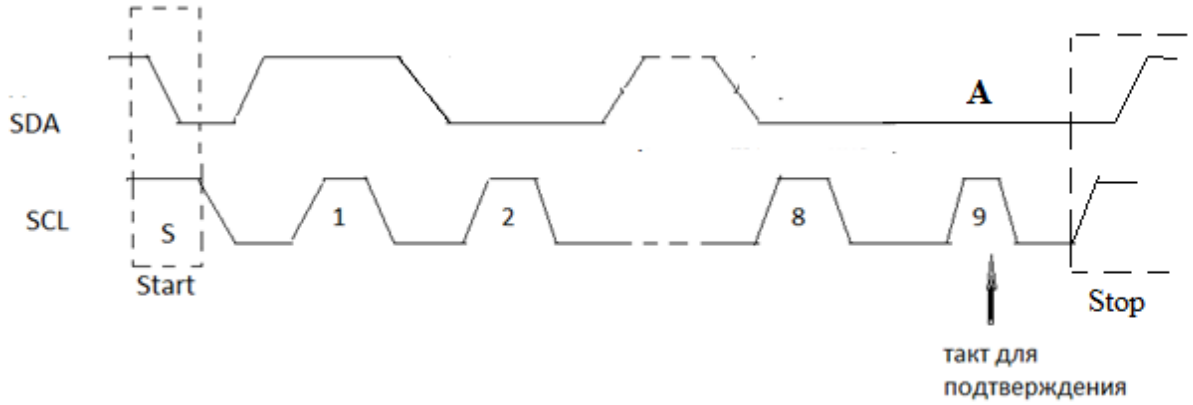


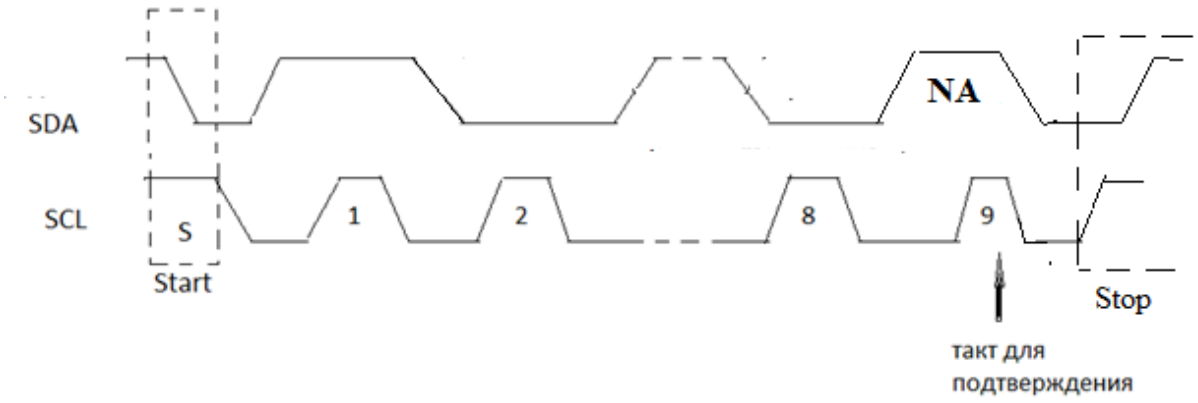
Адресация устройств на шине.

Ведущее устройство выставляет адрес ведомого и ждет подтверждения. Адрес ведомого устройства занимает 7 бит, 8-ой бит указывает дальнейшее направление передачи информации. Далее следует подтверждение. Все

устройства системы сравнивают 7 бит после старта со своими адресами, устройство, адрес которого был набран, становится ведомым и выдает сигнал подтверждения. После получения сигнала подтверждения ведущее устройство должно придержать линию синхронизации на низком уровне, пока ведомое ее не отпустит. Иначе синхронизация оборвется.







Арбитраж.

Ведущим становится устройство, выставившее низкий уровень на провод данных первым перед появлением низкого уровня на проводе синхронизации.

Синхронизацию задает ведущее устройство, но ведомое, если оно не имеет достаточного быстродействия, может замедлять обмен данными.
Таким образом, низкий уровень на проводе синхронизации появляется от устройства с самой высокой частотой, а высокий уровень появляется, когда его сформирует устройство с самой низкой частотой обмена.

Лекция 13.

Прерывания.

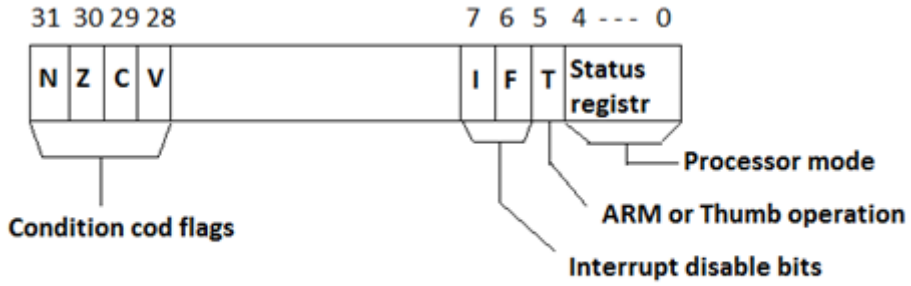
Различают два способа обмена данными между МПС и внешними устройствами: обмен по готовности и обмен по прерываниям. Готовность может быть двух видов: безусловная и программно-определяемая. Программно-определяемая готовность предполагает ввод или вывод информации только после прочтения выставленного внешним устройством бита готовности. Это приводит к непроизводительным затратам процессорного времени, поэтому в большинстве случаев используется все же обмен по прерываниям.

Прерывания – особым образом построенная операция приостановки основной программы и выполнения подпрограммы с дальнейшим возвратом в основную программу.

Прерывания могут быть программные, аппаратные и исключительные ситуации: ошибки, ловушки и аварийные завершения.

Основная часть исключительных ситуаций – отладочные прерывания.
Рассмотрим организацию прерывания в ARM-процессоре.

Обратимся к структуре CPSR

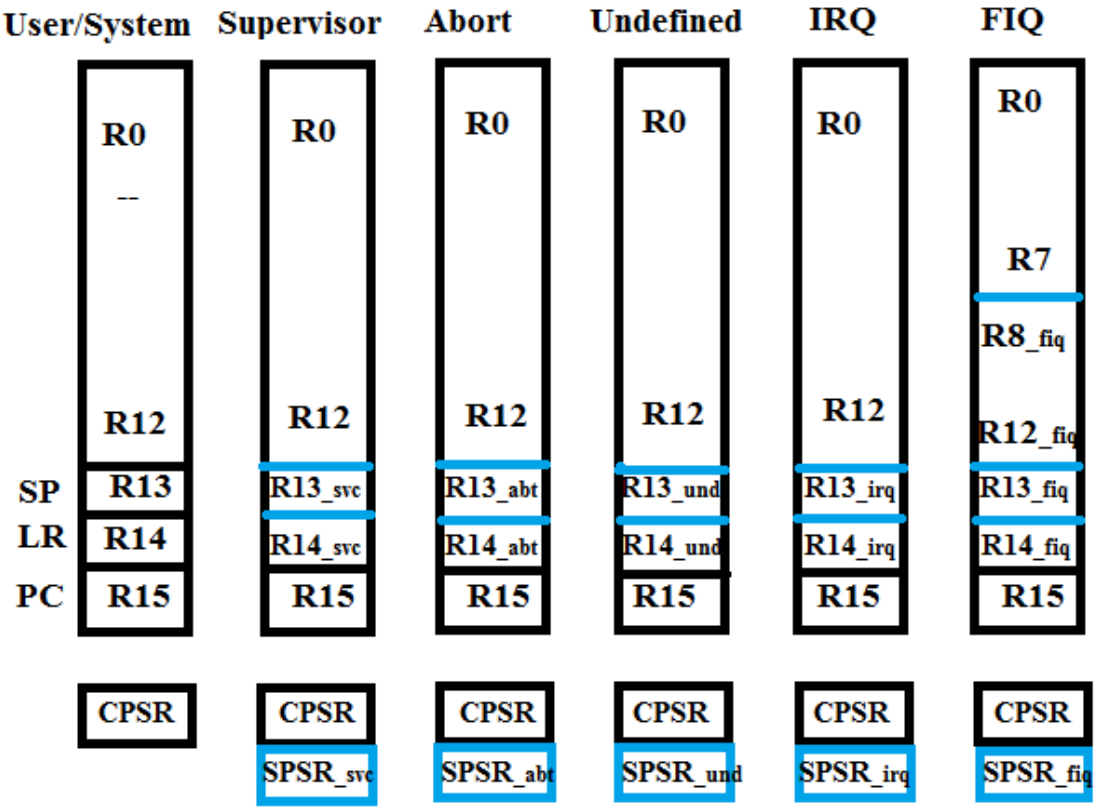


Код, записанный в младших разрядах, определяет способ функционирования процессорной системы.

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

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

Если первые два режима это текущее исполнение программы, то остальные устанавливаются при получении кода соответствующего прерывания. Поэтому для 5-ти режимов в блоке регистров необходимо производить объединения регистров и подключение дополнительных стековых областей.



В режимах работы без прерываний все регистры работают автономно.

В режимах, возникающих по причине появления прерывания, SP и LR объединяются в банки, а также добавляется банк для статуса. В режиме FIQ, где необходимо быстро обслужить запрос, выделяется специальный банк из регистров R8 – R12.

Первые действия процессора при поступлении сигнала о любом прерывании:

1. Сохранение точки выхода в основную программу - **PC ➔LR** (banked);
2. Сохранение текущего статуса из основной программы - **CPSR ➔SPSR** (banked);
3. Установка нового статуса, соответствующего типу прерывания - **Changes bits 4—0 of CPSR, sets I and F**;

Копирование статуса в регистр

MRS R_d, CPSR

Обратное копирование делать так просто нельзя.

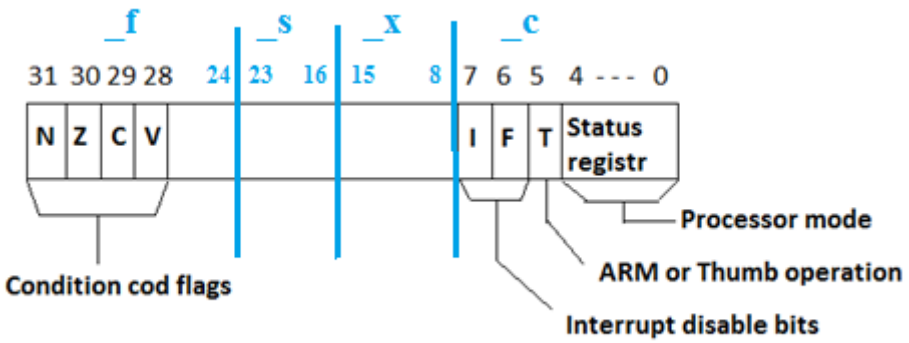
Регистр текущего статуса разделен на 4 поля:

Поле разрядов 31-24 - _f;

Поле разрядов 23-16 -_ s;

Поле разрядов 15 – 8 - _x;

Поле разрядов 7 – 0 - _c.



Например, при настройках аппаратного прерывания:

```
MOV R1, #0b11010010   // interrupts masked, MODE = IRQ
MSR CPSR_c, R1   // change to IRQ mode
```

4. В зависимости от нового статуса запись вектора в программный счетчик - **Vector address ➔ PC.**

Таблица векторов прерываний.

Адрес	Причина	Приоритет	Режим
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

5. Из таблицы векторов в программный счетчик записывается адрес выхода на подпрограмму - **Start address➔PC.**