Цифровая схемотехника

к.т.н., доцент

Аминев Дмитрий Андреевич

aminev.d.a@ya.ru +79067406453

Базовые понятия и определения

Схемотехника - это совокупность:
1) физически обоснованных принципов реализации схемных элементов,
2) процессов передачи сигналов в линиях связи,
3) функционально обоснованных принципов реализации структуры из схемных элементов,
4) наборов соглашений о формах представления информации и правил организации схемных (физических) интерфейсов.

Схемный элемент - электронный (или электрический) прибор или функционально законченный узел, пригодный к объединению в соответствии с правилами схемных интерфейсов. **Схемный (физический) интерфейс** - это соглашение о значениях (диапазонах значений) физического носителя информации, допускаемых при взаимных объединениях схемных элементов. **Большая интегральная схема(БИС)** - сверхминиатюрная электронная схема на полупроводниковой пластинке площадью менее 1 см^2, содержащая сотни и тысячи электронных элементов и выполняющая определенные функции.

Микропроцессор - программно-управляемое электронное устройство, предназначенное для обработки цифровой информации и построенное на одной или нескольких БИС.

Логическая ф-ция - это ф-ция, которая, как и ее аргументы, логические переменные, может принимать только два значения - "0" или "1". Лог."0" означает разомкнутую цепь, отсутствие сигнала, "ложь", низкий уровень сигнала. Лог. "1" означает замкнутую цепь, наличие сигнала, "истина", высокий уровень сигнала. В зависимости от количества входных переменных различают функции одной, двух или нескольких переменных. **Набор** - это комбинация значений логических переменных. Задать логическую функцию значит определить ее значение для всех наборов входных переменных. **Ф-ция считается полностью заданной**, если определены ее значения для всех наборов.

Пример ТИ: A,B,C – вх.перемен., X – вых. ф-ция

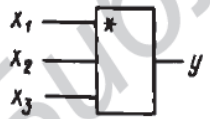
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

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

Логические элементы

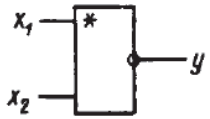
Логический элемент

$y = \varphi^*(x_1, x_2, x_3)$



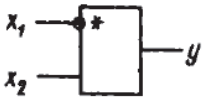
Элемент с инверсным выходом

$\overline{y = \varphi^*(x_1, x_2)}$



Элемент с инверсным входом

$y = \varphi^*(\overline{x_1}, x_2)$

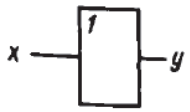


Здесь * — указатель функции, выполняемой логическим элементом.

Обозначение элементов, реализующих логические функции

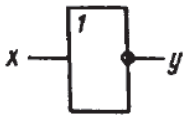
Повторитель

$y = x$



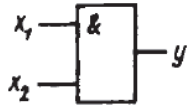
Инвертор (НЕ)

$y = \overline{x}$



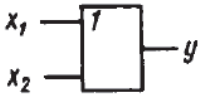
Конъюнктор (И)

$y = x_1 \cdot x_2$



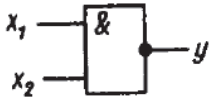
Дизъюнктор (ИЛИ)

$y = x_1 \vee x_2$



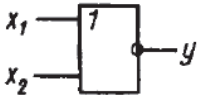
Элемент И-НЕ (элемент Шеффера)

$y = \overline{x_1 \cdot x_2} = x_1 \mid x_2$



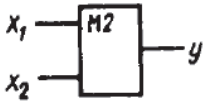
Элемент ИЛИ-НЕ (элемент Пирса)

$y = \overline{x_1 \vee x_2} = x_1 \downarrow x_2$



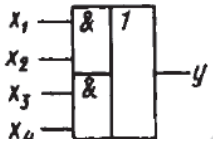
Сложение по модулю 2

$y = x_1 \cdot \overline{x}_2 \vee \overline{x}_1 \cdot x_2 = x_1 \oplus x_2$



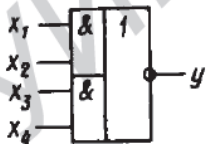
Элемент И-ИЛИ

$y = x_1 \cdot x_2 \vee x_3 \cdot x_4$



Элемент И-ИЛИ-НЕ

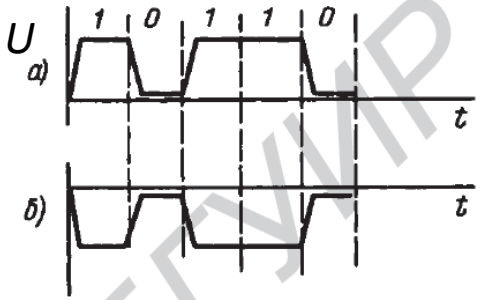
$\overline{y = x_1 \cdot x_2 \vee x_3 \cdot x_4}$



Потенциальный способ представления логических уровней:

Прямой

Инверс.

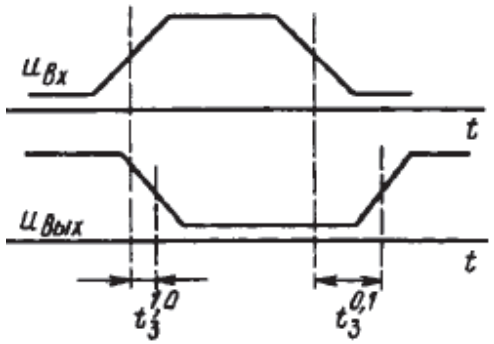


Характеристики ЛЭ

Коэф. объединения по вх. – число входов ЛЭ для подачи переменных. Нагрузочная способность (**коэф. разветвления по вых.**) – число входов других ЛЭ, которые можно подключить к выходу.

Быстродействие оценивантся задержкой распространения сигнала от входа к выходу ЛЭ.

$$t_{з\,ср} = 0{,}5\,(t_з^{0,1} + t_з^{1,0})$$



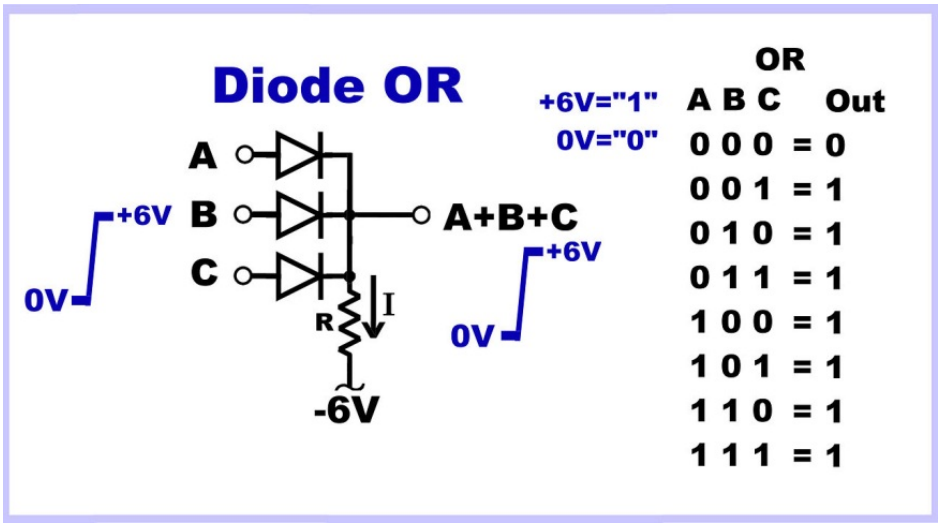
Помехоустойчивость (передаточная х-ка), Uвых(Uвх)

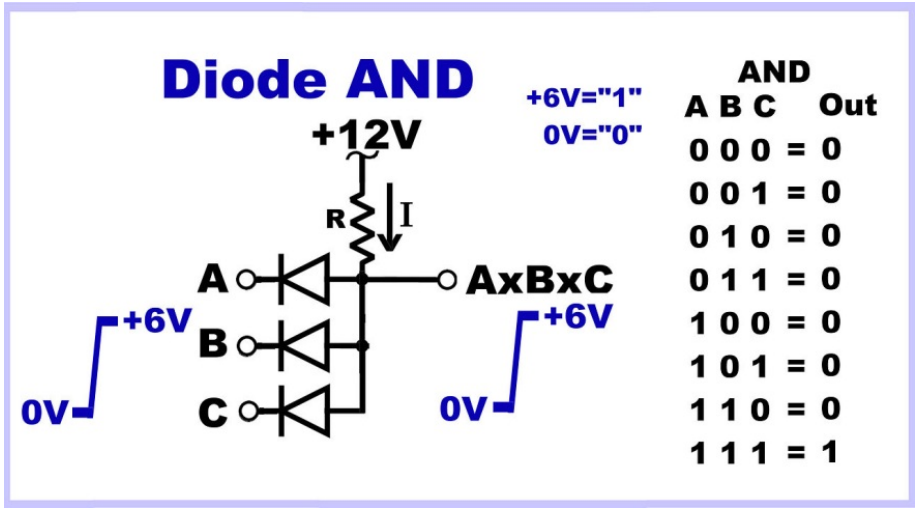
Реализация логических элементов электронике

- диодно-транзисторные ЛЭ (ДТЛ);
- транзисторно-транзисторные ЛЭ (ТТЛ);
- эмиттерно-связанные ЛЭ (ЭСЛ);
- ЛЭ на МДП- и КМДП- и КМОП транзисторах;
- ЛЭ с инжекционным питанием (интегральная инжекционная логика - $И^2Л$).

Распространёнными на сегодняшний день являются ИС, реализующие ТТЛ и её разновидности. Интегральные схемы данного типа обладают средним быстродействием (до 100 МГц) и средней потребляемой мощностью.

Диодная логика (ДЛ)

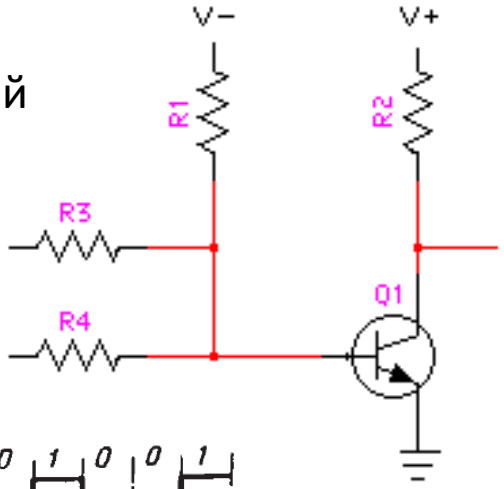




Резисторно-транзисторная логика (РТЛ)

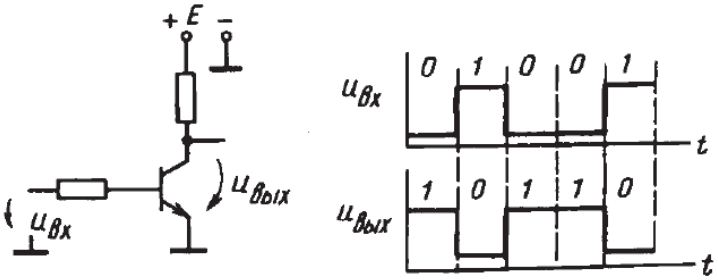
РТЛ — технология построения логических электронных схем на базе простых транзисторных ключей.

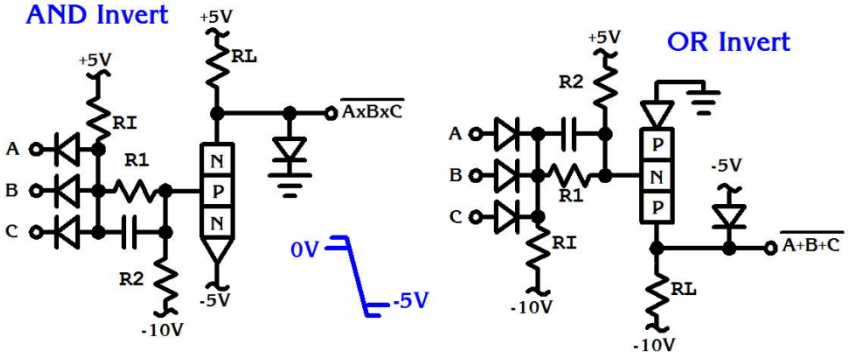
ИЛИ-НЕ в позитивной логике или И-НЕ в негативной.



Коллектор транзистора Q1 соединён через резистор R2 с шиной питания V+, а эмиттер с корпусом. К базе подключены резисторы R3,R4, входы. При отсутствии напряжения на всех входах - транзистор закрыт, и на выход через резистор поступает напряжение, близкое к напряжению питания, лог.1 на выходе при лог.0 на входе при позитивной логике. При наличии напряжения хоть на одном из входов, ключ открывается и закорачивает выход на корпус. При этом он «просаживает» выходное напряжение почти до 0.

Инвертор

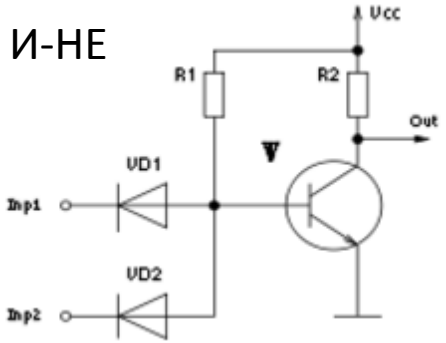




Диодно-транзисторная логика (ДТЛ)

ДТЛ - технология построения цифровых схем на основе биполярных транзисторов, диодов и резисторов. Реализации лог ф-ций (И) с помощью диодных цепей, а усиления и инверсии сигнала с помощью транзистора.

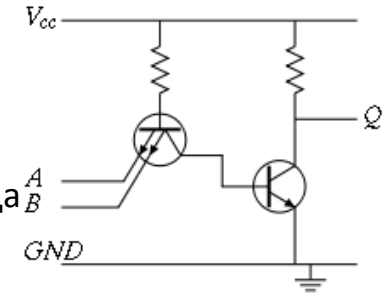
И-НЕ

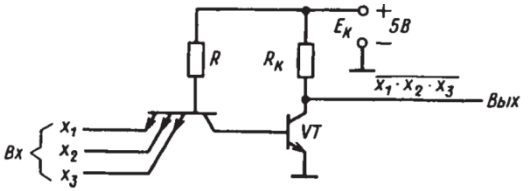


И-НЕ. Если хотя бы на одном из вх. лог.0, то ток течёт через R1 и VD во вх. цепь. На анодах диодов напряжение (0,7 В), которого недостаточно для открывания транзистора. Он закрыт. На выходе формируется лог.1. Если на все вх. поступает лог.1, ток течёт через R1 на базу транзистора, образуя на анодах VD напряжение 1,4 В. Поскольку напряжение уровня лог.1 больше этой величины, входы диодов обратно смещены и не участвуют в работе схемы. Транзистор открыт в режиме насыщения. В него втекает ток нагрузки, на выходе лог.0.

Транзисторно-транзисторная логика (ТТЛ)

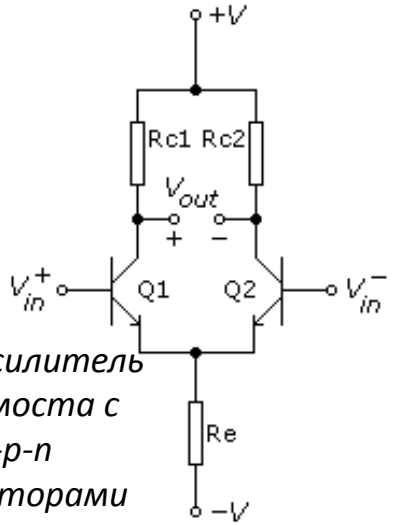
ТТЛ строится на основе биполярных транзисторов и резисторов. Транзисторы используются для выполнения логических функций (И, ИЛИ) и для усиления вых. сигнал. Простейший элемент выполняет операцию И-НЕ за счёт многоэмиттерного транзистора, объединяет свойства диода и транзисторного усилителя что позволяет увеличить быстродействие, снизить потребляемую мощность.



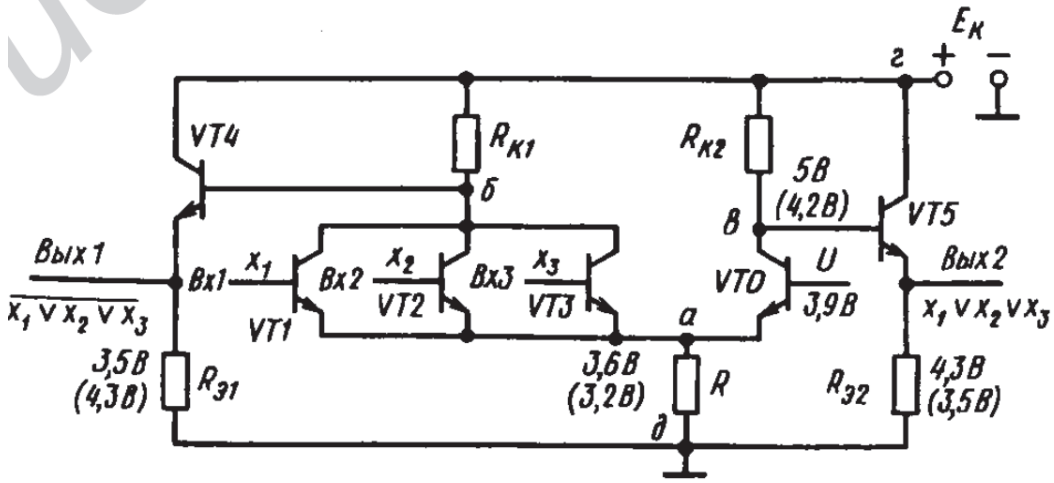


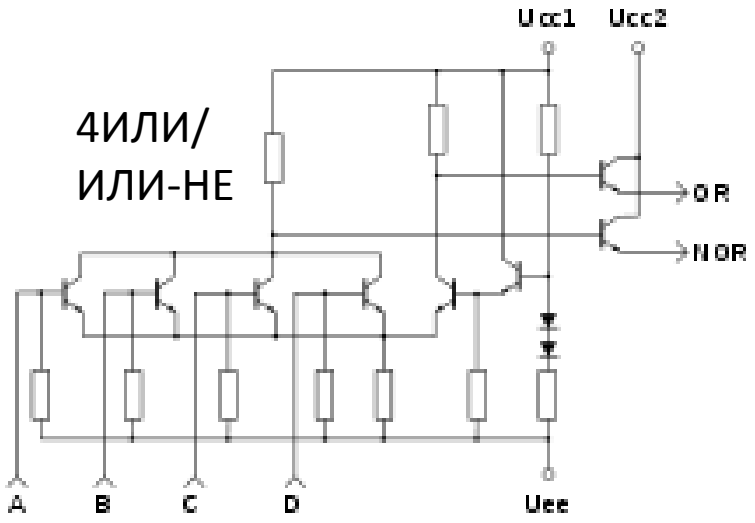
Эмиттерно-связанная логика (ЭСЛ)

ЭСЛ - на основе дифференциальных транзисторных каскадов. Транзисторы работают в линейном режиме, не переходя в насыщение, выход из которого замедлен. Низкие значения лог. перепадов в ЭСЛ снижают влияния на быстродействие паразитных ёмкостей. Транзистор, у которого напряжение на базе выше, пропускает через себя основной ток. Один транзистор в схеме сравнения подключен к опорному уровню, равному напряжению логического порога, а остальные являются входами. Выходные цепи схемы сравнения поступают на усилительные транзисторы, а с них — на выходные эмиттерные повторители. Высокое быстродействие, низкая помехоустойчивость, низкая степень интеграции, высокое энергопотребление.



Дифф. усилитель на базе моста с бипол. n-p-n транзисторами





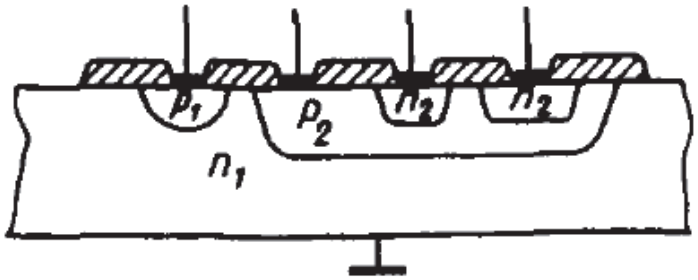
Интегральная инжекционная логика (И²Л)

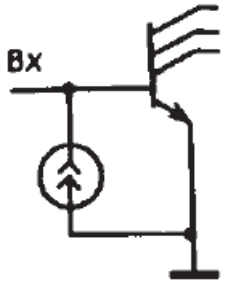
Элементы И²Л реализуются только в интегральном исполнении. Элемент состоит из двух транзисторов: горизонтальный p-n-p выполняет роль инжектора, а вертикальный многоколлекторный n-p-n работает в режиме инвертора. Общая область n-типа служит базой p-n-p, а также эмиттером n-p-n-транзистора и подключается к «заземлённой» точке. Коллектор p-n-p и база n-p-n-транзистора также являются общей областью.

Элемент имеет структуру p_1-n_1-p_2-n_2. Такую четырехслойную структуру удобно рассматривать, представив ее соединением двух обычных трехслойных транзисторных структур:

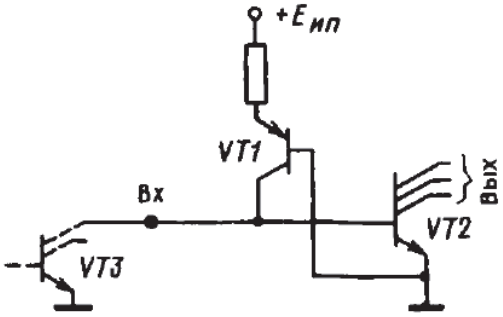
p_1-n_1-p_2

n_1-p_2-n_2.



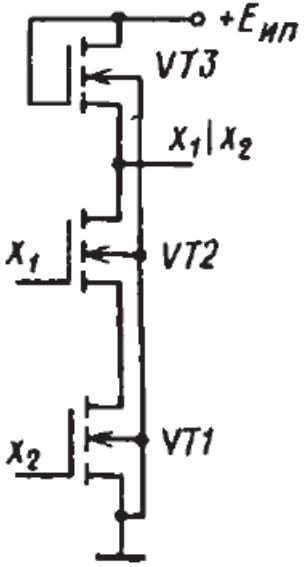


Соответствующая схема показана на рис. 1.20, *а*. Транзистор VT2 со структурой n_1-p_2-n_2 выполняет функции инвертора, имеющего несколько выходов (каждый коллектор образует отдельный выход элемента по схеме с открытым коллектором). Транзистор VT1, называемый *инжектором*, имеет структуру типа p_1-n_1-p_2. Так как область n_1 у этих транзисторов общая, эмиттер транзистора VT2 должен быть соединен с базой транзистора VT1; наличие общей области p_2 приводит к необходимости соединения базы транзистора VT2 с коллектором транзистора VT1. Так образуется соединение транзисторов VT1 и VT2.

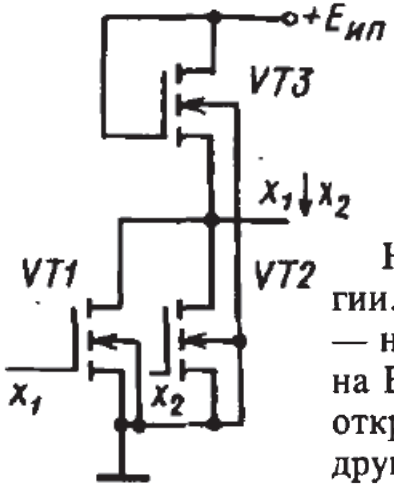


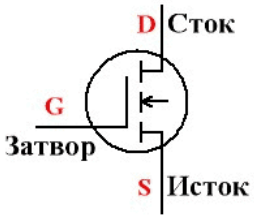
МОП и КМОП логика

а)И-НЕ

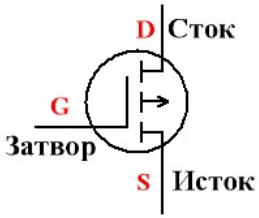


б) ИЛИ-НЕ





ПТ с n-каналом

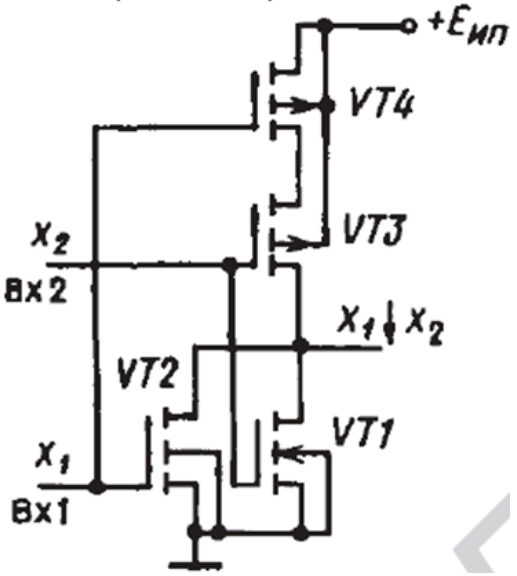


ПТ с p-каналом

На рис. 1.18, *в* приведена схема элемента ИЛИ-НЕ КМОП-технологии. В ней транзисторы VT1 и VT2 — основные, транзисторы VT3 и VT4 — нагрузочные. Пусть на одном из входов этого элемента (например, на Вх1) действует высокое напряжение U^1. При этом транзистор VT2 открыт, транзистор VT1 закрыт и независимо от уровня напряжения на другом входе и состояния остальных транзисторов на выходе устанавливается низкое напряжение U^0. Элемент реализует логическую операцию ИЛИ-НЕ.

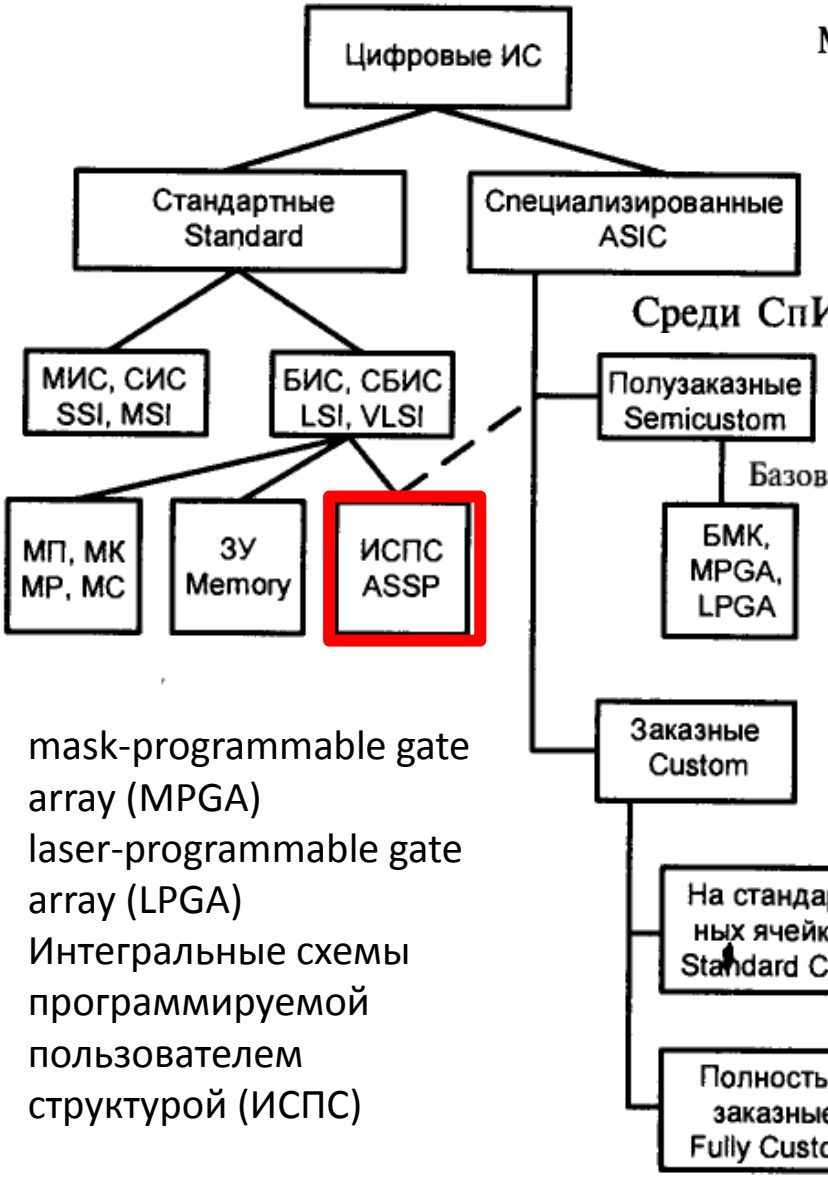
На рис. 1.18,*а* показана схема логического элемента И-НЕ на однотипных МОП-транзисторах с индуцированным каналом типа *n* (так называемая *n* МОП-технология) . Основные транзисторы VT1 и VT2 включены последовательно, транзистор VT3 выполняет роль нагрузки. В случае, когда на обоих входах элемента действует высокое напряжение U^1 ($x_1 = 1, x_2 = 1$), оба транзистора VT1 и VT2 оказываются открытыми и на выходе устанавливается низкое напряжение U^0. Во всех остальных случаях хотя бы один из транзисторов VT1 или VT2 закрыт и на выходе устанавливается напряжение U^1. Таким образом, элемент выполняет логическую функцию И-НЕ.

На рис. 1.18,*б* приведена схема элемента ИЛИ-НЕ. На его выходе устанавливается низкое напряжение U^0, если хотя бы на одном из входов действует высокое напряжение U^1 открывающее один из основных транзисторов VT1 или VT2.



в)ИЛИ-НЕ (КМОП)

Классификация цифровых интегральных схем



Микросхемы малого и среднего уровней интеграции МИС и СИС

Проектируются под конкретного заказчика

Среди СпИС (ASIC) различают классы *полузаказных* и *заказных* ИС.

Базовые матричные кристаллы называют также *вентильными матрицами*

БМК — кристалл, на прямоугольной поверхности которого размещены *внутренняя и периферийная области* (ВО и ПО). Во внутренней области по строкам и столбцам (в виде матрицы) расположены *базовые ячейки* — группы нескоммутированных схемных элементов (транзисторов, резисторов). Элементный состав базовой ячейки при разных вариантах межсоединений элементов допускает реализацию некоторого множества схем определенного класса, каждая из которых соответствует определенной *функциональной ячейке* (ФЯ). Для выпускаемого в продажу БМК создается *библиотека функциональных ячеек*, т. е., в сущности, рисунков межсоединений, дающих ту или иную схему. Библиотеки функциональных ячеек БМК насчитывают обычно десятки или сотни типовых узлов, реализованных на одной или нескольких базовых ячейках.

mask-programmable gate array (MPGA)
laser-programmable gate array (LPGA)
Интегральные схемы программируемой пользователем структурой (ИСПС)

Интегральные схемы с программируемой пользователем структурой (ИСПС) существуют уже около 25 лет и к настоящему времени представлены множеством разнообразных семейств. Программирование структур вначале было применено в программируемых логических матрицах (ПЛМ), программируемой матричной логике (ПМЛ) и базовых матричных кристаллах (БМК). Вслед за ними возникли новые классы более сложных ИСПС, продолжающих линии развития матричной логики и базовых матричных кристаллов: CPLD и FPGA, соответственно. Затем были реализованы ИСПС комбинированной (смешанной) архитектуры, сочетавшие признаки CPLD и FPGA. Позднее удалось разработать ИСПС с аналоговыми и аналого-цифровыми элементами, которые можно обозначить как ПАИС (программируемые аналоговые интегральные схемы).

ПЛИС либо ЦИСПС, т. е. *"программируемые логические интегральные схемы"* либо *"цифровые интегральные схемы с программируемой структурой"*.

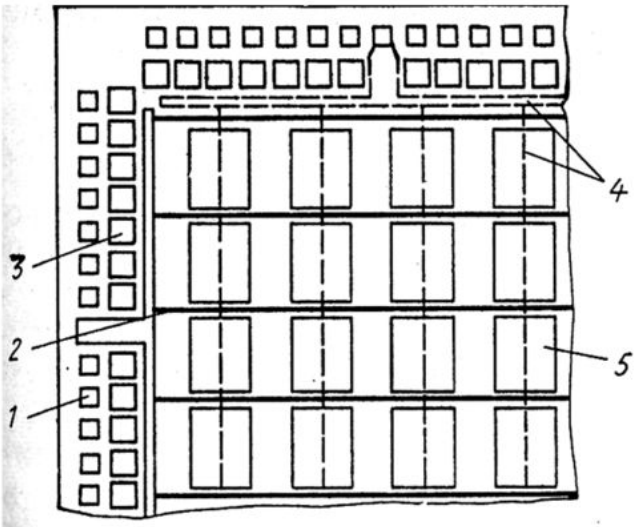
Классификация интегральных схем. БМК и ПЛМ

БМК

Кристалл МаБИС – БМК, представляет собой многослойную пластину, на которой реализованы несоединенные активные и пассивные компоненты, сгруппированные, как правило, в топологические ячейки (ТЯ).

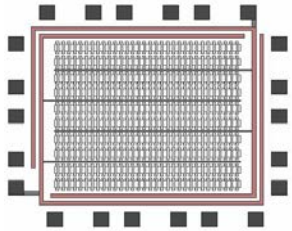
Определена библиотека функциональных элементов (БФЭ).

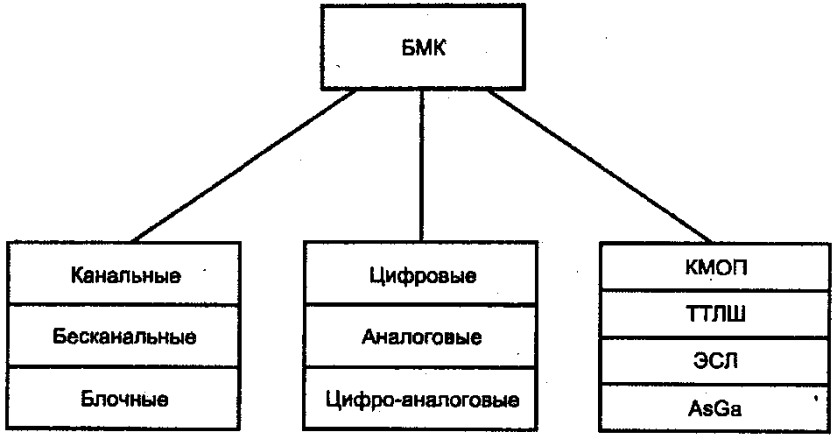
Для ее элементов существуют фотошаблоны – эталоны металлизации соединения компонентов.

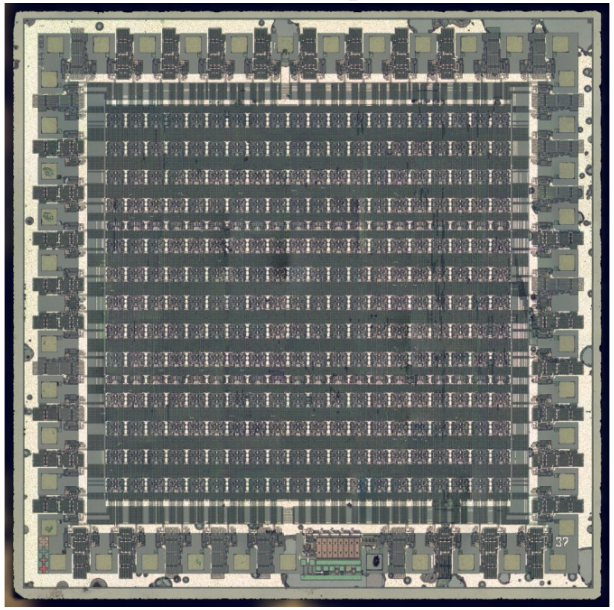


1 — внешний вывод (контактная площадка); *2* — шина питания (первый слой); *3* — периферийная ячейка; *4* — шина «земли» (второй слой); *5* — топологическая (макро) ячейка

MPGA







Программируемые логические матрицы (ПЛМ)

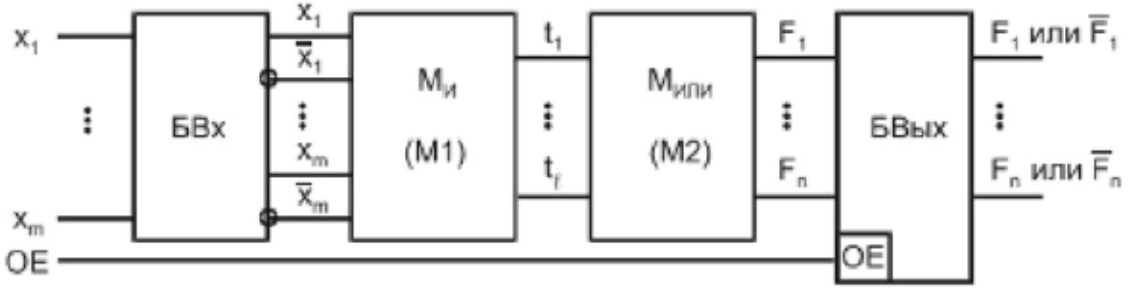


Рис. 2 Базовая структура ПЛМ

БВх – входные буферы, преобразующие однофазные входные сигналы в парафазные.

Ми – матрица элементов И (конъюнкторов), на выходе которой формируются λ термов.

Мили – матрица элементов ИЛИ (дизъюнкторы), формирующая *n* выходных функций.

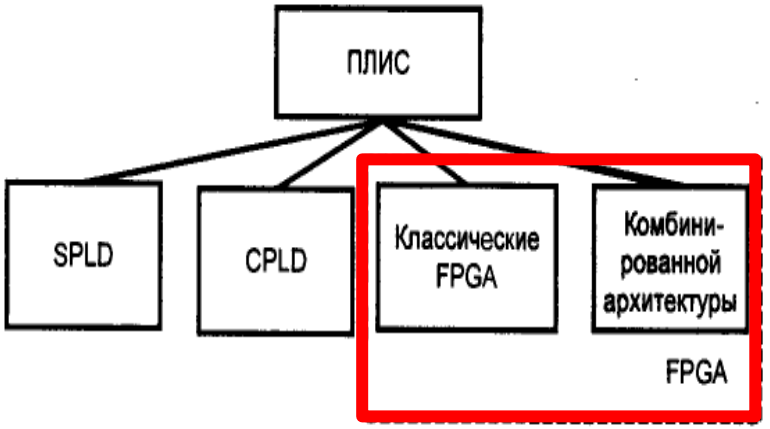
Классификация ПЛИС

- уровню интеграции и связанной с ним логической сложности;
- архитектуре (типу функциональных блоков, характеру системы межсоединений);
- числу допустимых циклов программирования;
- типу памяти конфигурации (*"теневой"* памяти);
- степени зависимости задержек сигналов от путей их распространения;
- системным свойствам;
- схемотехнологии (КМОП, ТТЛШ и др.);
- однородности или гибридности (по признаку наличия или отсутствия в микросхеме областей с различными по методам проектирования схемами, такими как ПЛИС, БМК, схемы на стандартных ячейках).

Все перечисленные признаки имеют значение и отображают ту или иную сторону возможных классификаций. Выделяя основные признаки и укрупняя их, рассмотрим классификацию по трем, в том числе двум комплексным, признакам:

- архитектуре;
- уровню интеграции и однородности/гибридности;
- числу допустимых циклов программирования и связанному с этим типу памяти конфигурации.

Классификация ПЛИС по архитектурным признакам



В сложных программируемых логических схемах CPLD (Complex Programmable Logic Devices) несколько блоков, подобных ПМЛ, объединяются средствами программируемой коммутационной матрицы. В CPLD могут входить сотни блоков и десятки тысяч эквивалентных вентилей. Архитектуры CPLD разрабатываются фирмами Altera, Atmel, Lattice Semiconductor, Cypress Semiconductor, Xilinx и др. Воздействуя на программируемые соединения коммутационной матрицы и ПМЛ, входящих в состав CPLD, можно реализовать требуемую схему.

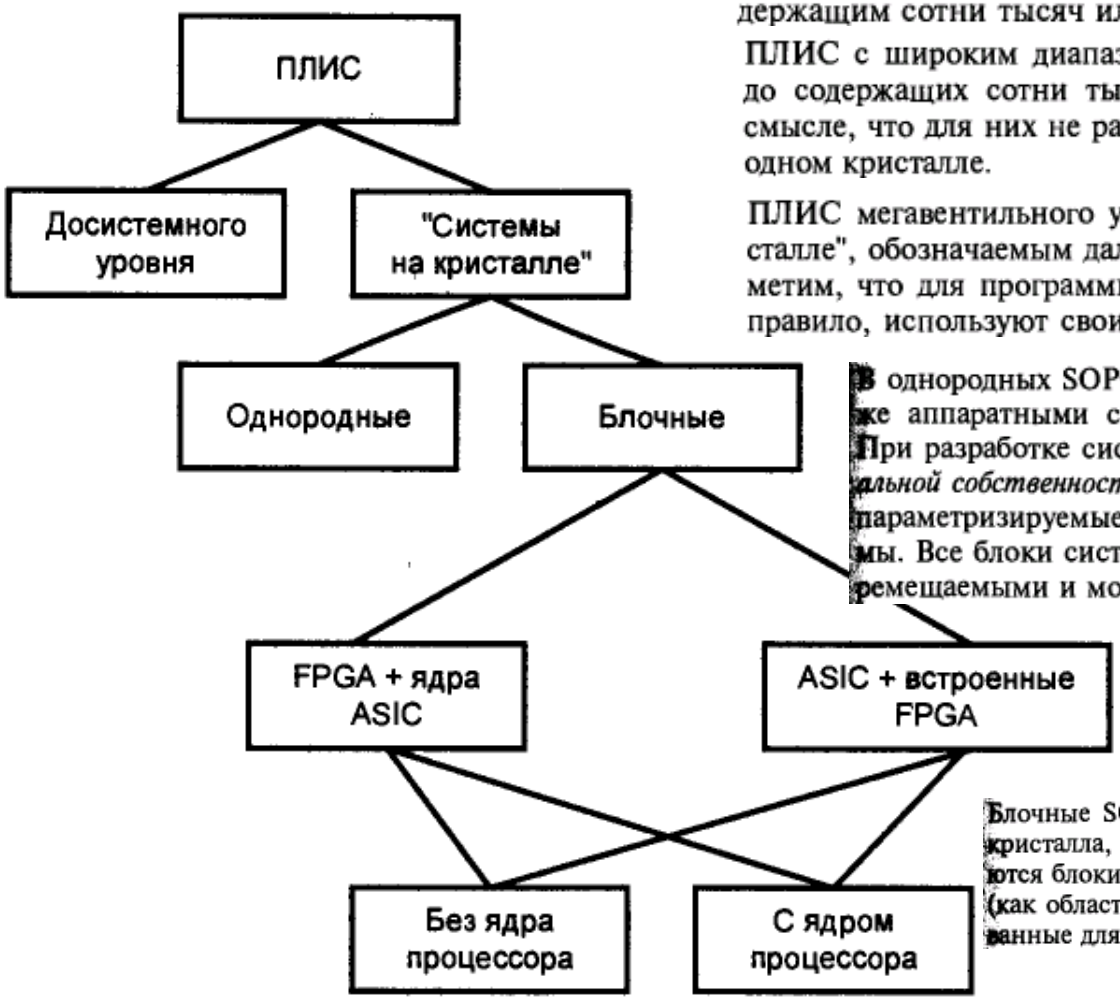
Первый из классов — SPLD, Simple Programmable Logic Devices, т. е. простые программируемые логические устройства. По архитектуре эти ПЛИС делятся на подклассы программируемых логических матриц ПЛМ (PLA, Programmable Logic Arrays) и программируемой матричной логики ПМЛ (PAL, Programmable Arrays Logic, или GAL, Generic Array Logic).

Оба эти подкласса микросхем реализуют дизъюнктивные нормальные формы (ДНФ) переключательных функций, а их основными блоками являются две матрицы: матрица элементов И и матрица элементов ИЛИ, включенные последовательно. Такова структурная модель ПЛМ и ПМЛ. Технически они могут быть выполнены и как последовательность двух матриц элементов ИЛИ-НЕ, но варианты с последовательностью матриц И-ИЛИ и с последовательностью матриц ИЛИ-НЕ — ИЛИ-НЕ функционально эквивалентны, т. к. второй вариант согласно правилу де Моргана тоже реализует ДНФ, но для инверсных значений переменных.

Микросхемы программируемых пользователями вентильных матриц FPGA (Field Programmable Gate Arrays) в своей основе состоят из большого числа конфигурируемых логических блоков ЛБ, расположенных по строкам и столбцам в виде матрицы, и трассировочных ресурсов, обеспечивающих их межсоединения. В архитектуре FPGA явно прослеживается большое сходство с архитектурой MPGA. Разница в том, что FPGA, поступающая в распоряжение потребителя, имеет уже готовые, стандартные, хотя и не запрограммированные, трассировочные ресурсы, не зависящие от конкретного потребителя. Получение конкретного проекта на базе FPGA, как и на основе других ПЛИС, реализуется воздействием на программируемые межсоединения, в ходе которого обеспечивается замкнутое состояние одних участков и разомкнутое — других. Обращаться к изготовителю FPGA при этом не требуется.

Классификация ПЛИС

Классификация ПЛИС по уровню интеграции



Термин SOPC (System On Programmable Chip), т. е. *"система на программируемом кристалле"* относится к ПЛИС наибольшего уровня интеграции, содержащим сотни тысяч или даже миллионы эквивалентных вентилей. Такой

ПЛИС с широким диапазоном изменения уровня интеграции (от простых до содержащих сотни тысяч вентилей) отнесены к "досистемным" в том смысле, что для них не рассматривались вопросы создания целых систем на одном кристалле.

ПЛИС мегавентильного уровня интеграции отнесены к "системам на кристалле", обозначаемым далее как SOPC (Systems On Programmable Chip). Заметим, что для программируемых систем на кристалле разные фирмы, как правило, используют свои обозначения (PSOC, CSOC, FIPSOC и т. д.), ре-

В однородных SOPC различные блоки системы реализуются одними и теми же аппаратными средствами, благодаря программируемости этих средств. При разработке систем используются так называемые *"единицы интеллектуальной собственности"* IP (Intellectual Properties), т. е. заранее реализованные параметризируемые мегафункции для создания тех или иных частей системы. Все блоки системы при этом являются полностью синтезируемыми, перемещаемыми и могут располагаться в разных областях кристалла. Создание

Блочные SOPC имеют *аппаратные ядра*, т. е. специализированные области кристалла, выделенные для определенных функций. В этих областях создаются блоки неизменной структуры, спроектированные по методологии ASIC (как области типа БМК или схем со стандартными ячейками), оптимизированные для заданной функции и не имеющие средств ее программирования.

содержат полный комплект блоков, характерных для микропроцессорной

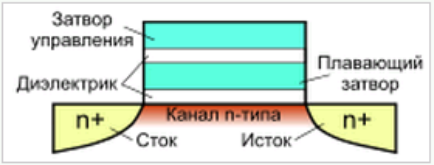
ориентированы на те или иные конкретные приложения

Классификация ПЛИС

Классификация ПЛИС по признакам кратности программирования
(по типу теневой памяти)

В однократно программируемых ПЛИС используются элементы с необратимыми изменениями состояний — специальные *перемычки* или *ЛИЗМОП-транзисторы*. ЛИЗМОП-транзисторы имеют заряжаемые *"плавающие"* затворы, которые, в общем случае, могут как заряжаться, так и разряжаться. Для однократно программируемых ПЛИС возможности ЛИЗМОП-транзисторов используются лишь частично: для них применяются такие конструкции, в которых отсутствуют возможности стирания записанной информации.

В простых ПЛИС первых поколений применялись *плавкие перемычки* типа fuse. В таких ПЛИС в исходном состоянии имеются все возможные соединения, а для получения требуемой конфигурации схемы часть перемычек разрушается (пережигается). При программировании плавких перемычек возникает определенный процент брака, кроме того, со временем проводимость разрушенной перемычки может восстановиться из-за явления электромиграции в материалах. В течение многих лет велась большая работа по



Конструкция транзистора с плавающим дополнительным затвором.

ЛИЗМОП (FAMOS — Floating gate Avalanche injection Metal Oxide Semiconductor) — полевой МОП-транзистор с лавинной инжекцией заряда

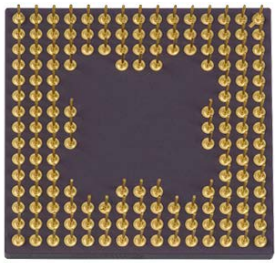
В однократно программируемых FPGA нашли применение *пробиваемые перемычки* типа antifuse. В исходном состоянии сопротивления перемычек чрезвычайно велики, а в пробитом достаточно малы. Перемычки очень компактны — их площадь близка к площади пересечения двух дорожек межсоединений. Паразитные емкости перемычек также очень малы. Боль-

В третьем варианте (с плавающими затворами) роль программируемых элементов играют *однозатворные ЛИЗМОП-транзисторы*, а кристаллы микросхем размещаются в дешевых корпусах, не имеющих специальных окошек для стирания информации (зарядов в плавающих затворах). Для транзисторов с одним (плавающим) затвором и каналом p-типа до программирования затвор не имеет заряда, и транзистор заперт. Введение в затвор заряда электронов приводит к возникновению в транзисторе проводящего канала. Заряд в плавающем затворе сохраняется в течение десятков лет. Память конфигурации с элементами описанного типа называют EPROM-OTP (Electrically Programmable Read-Only Memory — One Time Programmable). Однозатворные

К памяти типа EEPROM близка память конфигурации типа Flash.

Классификация ПЛИС

В репрограммируемых ПЛИС с памятью конфигурации типа EEPROM (Electrically Erasable Programmable Read-Only Memory) стирание старых данных осуществляется электрическими сигналами. Используются двухзатворные ЛИЗМОП-транзисторы. Управление процессами в транзисторе производится с помощью двух затворов — обычного и плавающего. При определенных сочетаниях программирующих напряжений на внешних выводах транзистора (плавающий затвор внешнего вывода не имеет) создаются режимы как заряда плавающих затворов, так и их разряда. В русской терминологии память типа EEPROM называют РПЗУ-ЭС (репрограммируемые запоминающие устройства с электрическим стиранием). Электрическое стирание содержимого памяти не требует извлечения микросхем из устройства



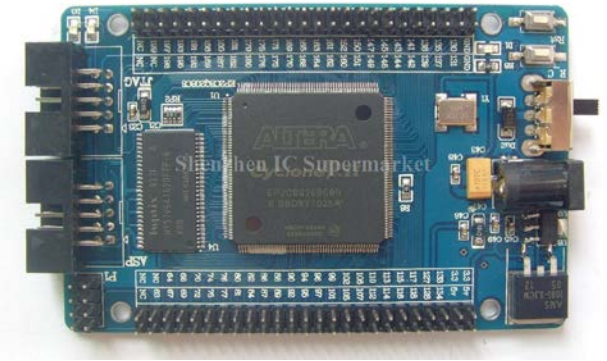
Последний класс ПЛИС по второму признаку принятой классификации — *оперативно репрограммируемые*. В таких ПЛИС конфигурация задается с помощью загрузки файла в "теневую" триггерную память, т. е. операций, не имеющих какого-либо специального характера. В противоположность предыдущим вариантам для программирования не нужны ни специальные программаторы, ни специальные режимы с повышенными напряжениями и длительностями воздействий на элементы памяти. Память конфигурации — обычная статическая (триггерная), т. е. типа SRAM, Static Random Access Memory. Загрузка памяти производится с высокой скоростью, свойственной статической триггерной памяти, последовательным потоком битов или байтов. Элементом с программируемой проводимостью (режимом "замкнуто-разомкнуто") служит обычный МОП-транзистор, управляемый триггером памяти конфигурации (теневой памяти). Состояние триггера задает режим ключевому транзистору. *Программирование соединения сводится к установке*

ригтерная память не является *энергонезависимой*, и *выключение питания ве*
ет к разрушению конфигурации ПЛИС, поэтому при очередном его включе
ии нужно ее восстановить, загрузив в триггеры теневой памяти файл кон
игурации из какой-либо энергонезависимой памяти. Загрузка производит

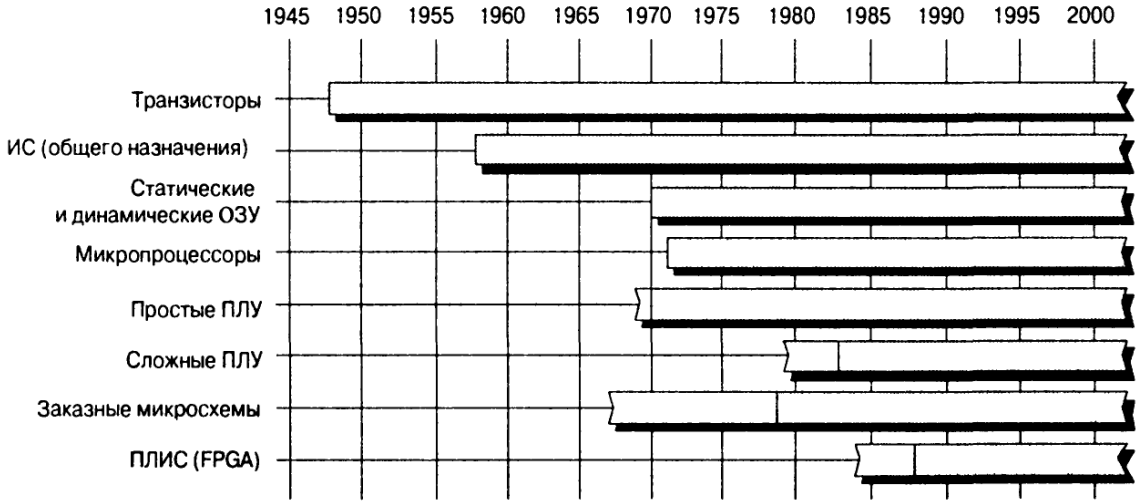
озможности оперативной реконфигурации, свойственные ПЛИС с триг-
рной памятью, получили дальнейшее развитие в *архитектурах с динамиче-
ким репрограммированием*. В ПЛИС с динамическим репрограммированием
онфигурация может быть изменена чрезвычайно быстро. Переход от одной
онфигурации к другой не требует ввода извне нового файла конфигурации

Свойства и преимущества ПЛИС

- Универсальность и связанный с нею высокий спрос со стороны потребителей, что обеспечивает массовое производство.
- Низкая стоимость, обусловленная массовым производством и высоким процентом выхода годных микросхем при их производстве вследствие достаточно регулярной структуры.
- Высокое быстродействие и надежность как следствие реализации на базе передовых технологий и интеграции сложных устройств на одном кристалле.
- Разнообразие конструктивного исполнения, поскольку обычно одни и те же кристаллы поставляются в разных корпусах.
- Разнообразие в выборе напряжений питания и параметров сигналов ввода/вывода, а также режимов снижения мощности, что особенно важно для портативной аппаратуры с автономным питанием.
- Наличие разнообразных, хорошо развитых и эффективных программных средств автоматизированного проектирования, малое время проектирования и отладки проектов, а также выхода продукции на рынок.
- Простота модификации проектов на любых стадиях их разработки.



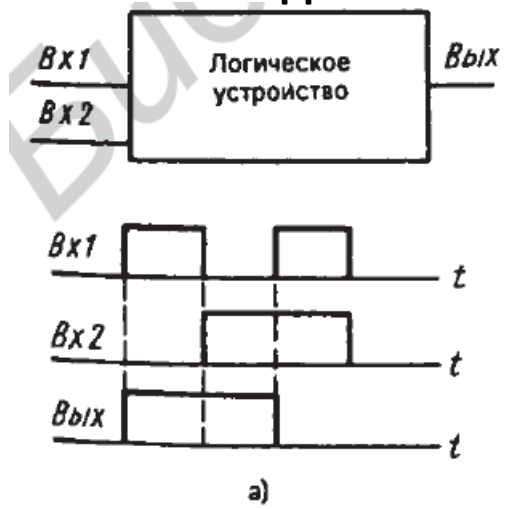
ИСТОРИЯ РАЗВИТИЯ ПЛИС



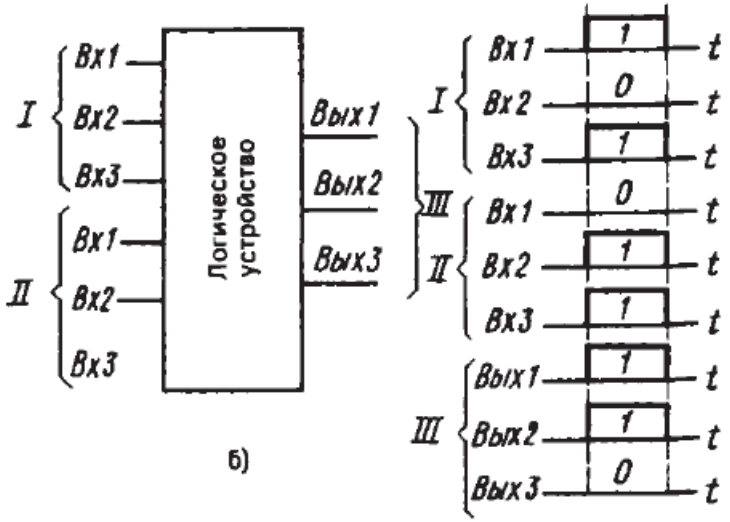
Логические или цифровые устройства

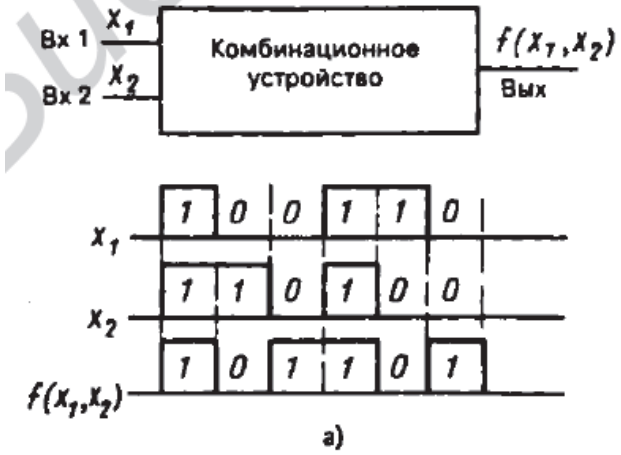
Устройства, предназначенные для формирования функций алгебры логики, называются *логическими устройствами* или *цифровыми устройствами*. Цифровые устройства (либо их узлы) можно делить на типы по различным признакам.

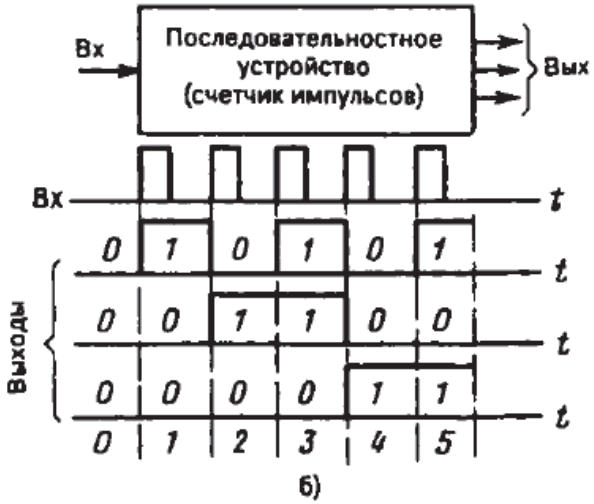
Последовательные



Параллельные

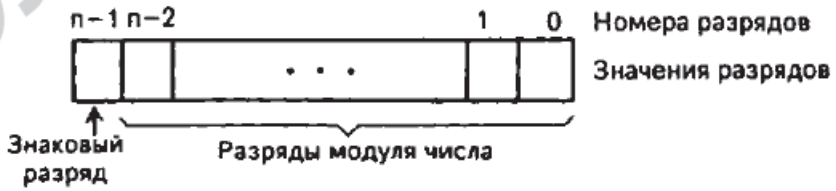


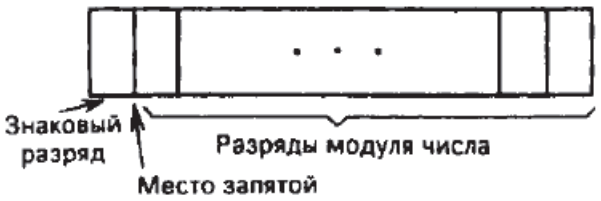


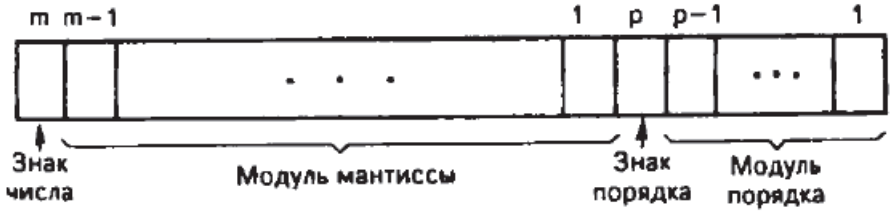


Представление чисел в цифровых устройствах

Числа в цифровых устройствах могут представляться в форме целых чисел, чисел с фиксированной запятой и чисел с плавающей запятой.

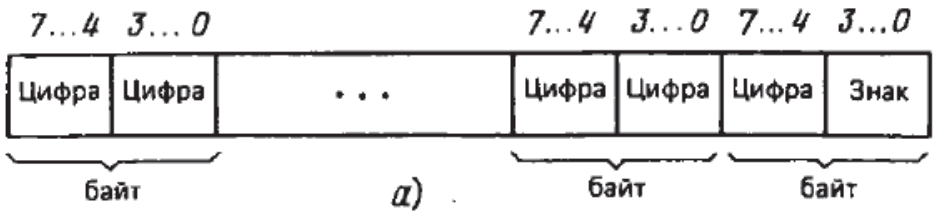


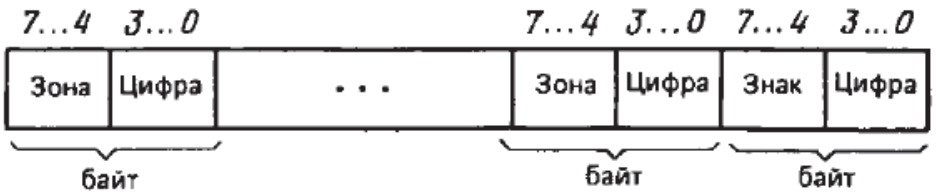




$N = M \cdot 2^{П}$, где M и $П$ — мантисса и порядок числа

При использовании упакованного формата каждый *байт* (8 разрядов двоичного числа) содержит две десятичные цифры (рис. 2.4,*а*).





Триггеры

Основные обозначения. Триггер имеет два выхода: *прямой* Q и *инверсный* $\overline{Q}$. Состояние, в котором находится триггер, определяется уровнями напряжения на этих выходах: если напряжение на выходе Q соответствует уровню *лог.0* ($Q = 0$), то принимается, что триггер находится в состоянии 0, при $Q = 1$ триггер находится в состоянии 1. Логический уровень на инверсном выходе $\overline{Q}$ представляет собой инверсию состояния триггера (в состоянии 0 $Q = 1$, и наоборот).

x_1	x_2	$x_1 \mid x_2$	$x_1 \downarrow x_2$
0	0	1	1
0	1	1	0
1	0	1	0
1	1	0	0

Триггеры имеют различные типы входов. Приведем их обозначения и назначения:

S	R	Q
0	0	Q_0
0	1	0
1	0	1
1	1	*

J	K	Q
0	0	Q_0
0	1	0
1	0	1
1	1	$\overline{Q}_0$

D	Q
0	0
1	1

T	Q
0	Q_0
1	$\overline{Q}_0$

R (от англ. Reset) — *раздельный вход установки в состояние 0*;

S (от англ. Set) — *раздельный вход установки в состояние 1*;

K — *вход установки универсального триггера в состояние 0*;

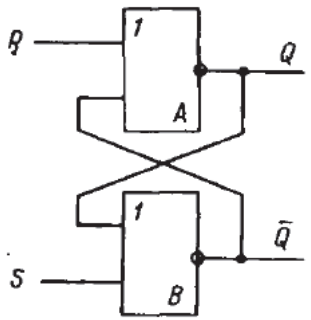
J — *вход установки универсального триггера в состояние 1*;

T — счетный вход;

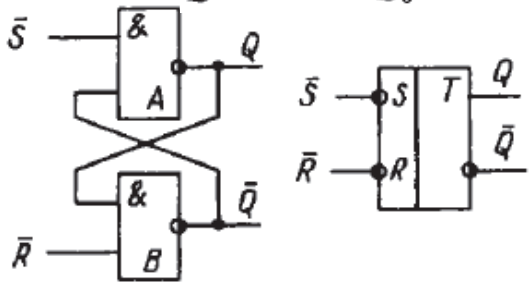
D (от англ. Delay) — *информационный вход установки триггера в состояние, соответствующее логическому уровню на этом входе*;

C — *управляющий (синхронизирующий) вход.*

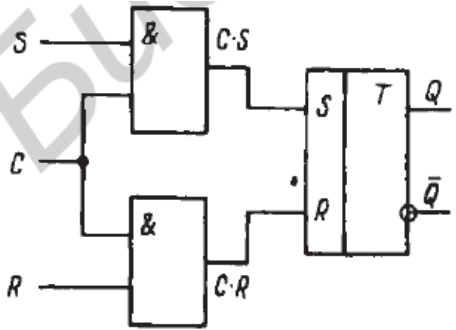
Наименование триггера определяется типами его входов. Например, RS-триггер — триггер, имеющий входы типов R и S.



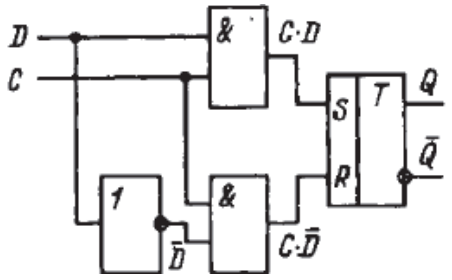
$$Q = S \vee \overline{R} \cdot Q_0$$

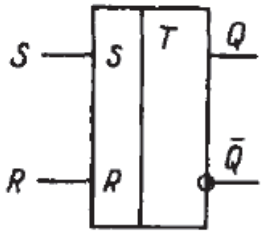


$$Q = \overline{C} \cdot Q_0 \vee C \cdot (S \vee \overline{R} \cdot Q_0)$$



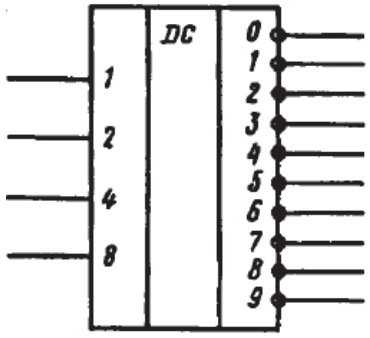
$$Q = \overline{C} \cdot Q_0 \vee C \cdot D$$

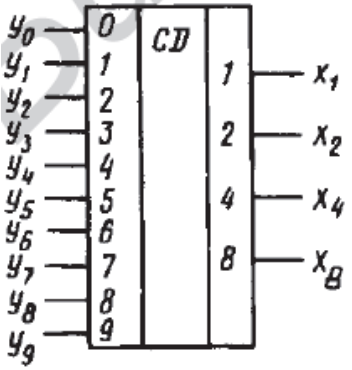




Шифраторы, дешифраторы, мультиплексоры, демультиплексоры

Шифратор (называемый также *кодером*) осуществляет преобразование десятичных чисел в двоичную систему счисления. Пусть в шифрато-





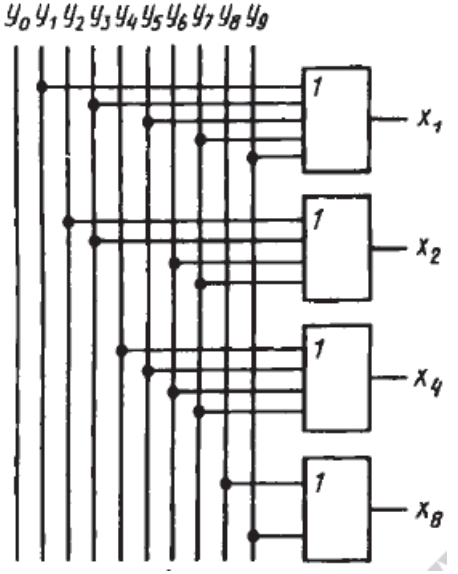
$$x_1 = y_1 \vee y_3 \vee y_7 \vee y_9.$$

Для остальных выходов

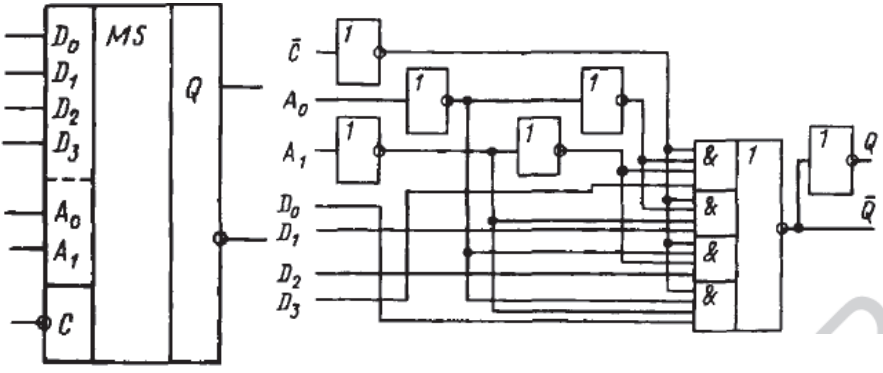
$$x_2 = y_2 \vee y_3 \vee y_6 \vee y_7,$$

$$x_4 = y_4 \vee y_5 \vee y_6 \vee y_7,$$

$$x_8 = y_8 \vee y_9.$$



Назначение и принцип работы. Устройство, которое осуществляет выборку одного из нескольких входов и подключает его к своему выходу, называется *мультиплексором*. Мультиплексор имеет несколько информационных входов (D_0, D_1, ...), адресные входы (A_0, A_1, ...), вход для подачи стробирующего сигнала C и один выход Q. На рис. 3.23, *а* показано символическое изображение мультиплексора с четырьмя информационными входами.

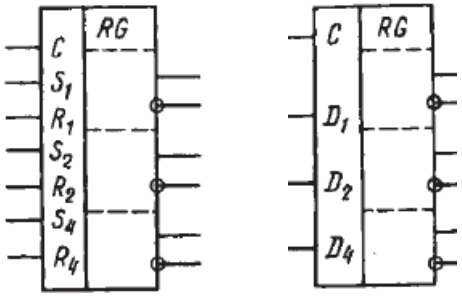


$$Q = (D_0 \cdot \overline{A}_1 \cdot \overline{A}_0 \vee D_1 \cdot \overline{A}_1 \cdot A_0 \vee D_2 \cdot A_1 \cdot \overline{A}_0 \vee D_3 \cdot A_1 \cdot A_0) \cdot C.$$

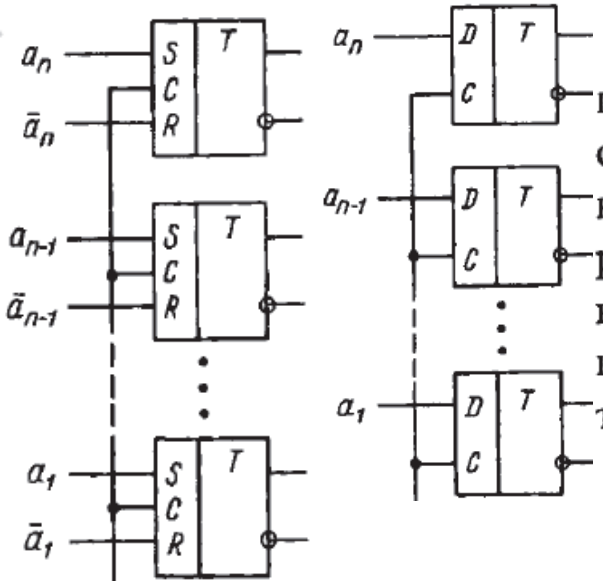
Адресные входы		Стробирующий сигнал	Выход
A_1	A_0	C	Q
×	×	0	0
0	0	1	D_0
0	1	1	D_1
1	0	1	D_2
1	1	1	D_3

Демультиплексор имеет один информационный вход и несколько выходов и осуществляет коммутацию входа к одному из выходов, имеющему заданный адрес (номер). На рис. 3.25 показана структура демуль-

Регистры

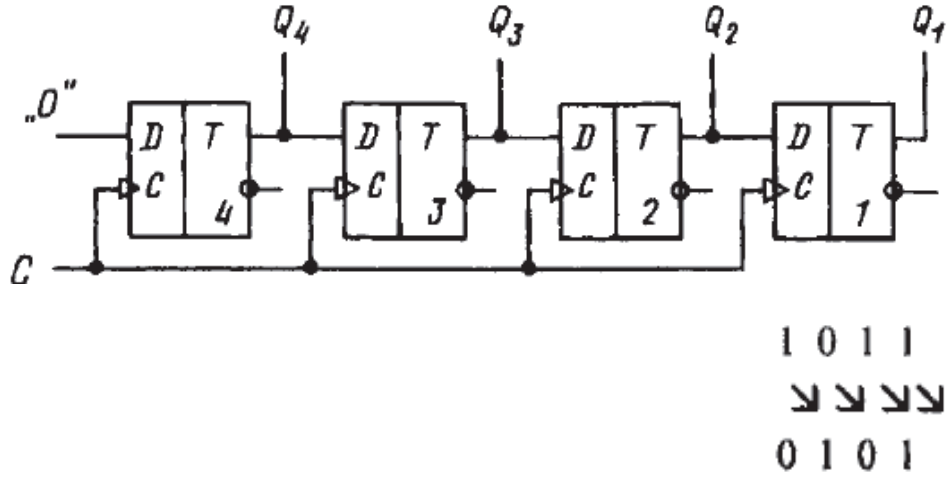


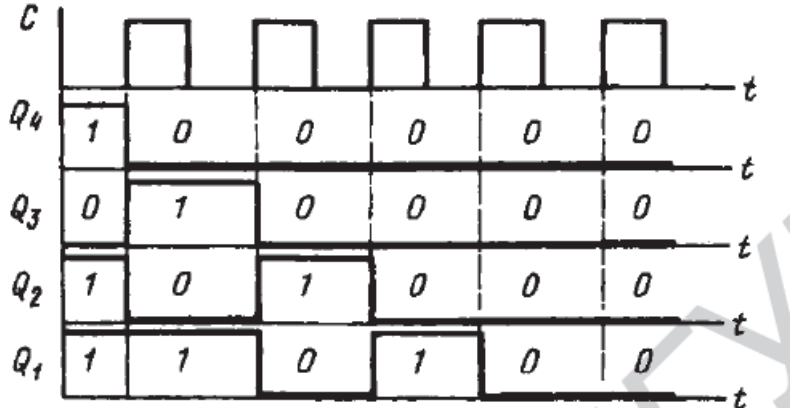
Основная функция регистров — хранение одного многоразрядного числа. При этом число должно быть представлено в двоичной системе счисления или в любой другой системе, но с двоичным представлением цифр разрядов (т.е. в любой двоично-кодированной системе счисления).



В зависимости от формы представления числа (параллельной или последовательной), вводимого в регистр, различают два типа регистров: *параллельные и последовательные*. В параллельный регистр предназначенное для хранения число подается одновременно всеми разрядами, т.е. в параллельной форме. В последовательный регистр ввод числа производится путем последовательной во времени подачи цифр отдельных разрядов (обычно начиная с цифры младшего разряда), т.е. в последовательной форме.

Сдвиговый регистр

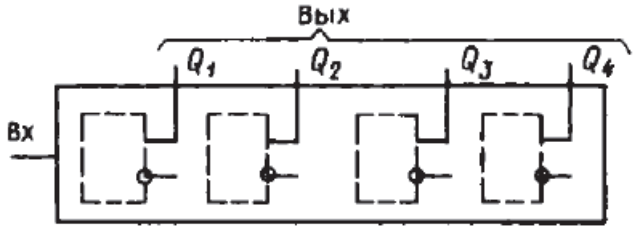




Счетчики, сумматоры

Счетчик — это цифровое устройство, определяющее, сколько раз на его входе появился некоторый определенный логический уровень. В

$$N = 2^n - 1$$



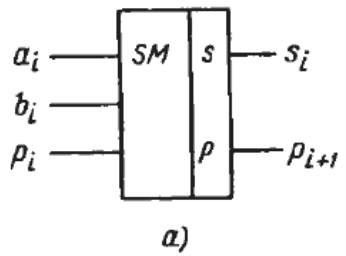
Делитель частоты — устройство, которое при подаче на его вход периодической последовательности импульсов формирует на выходе такую же последовательность, но имеющую частоту повторения импульсов, в некоторое число раз меньшую, чем частота импульсов входной последовательности.

Число поступивших импульсов	Состояние триггеров				Число поступивших импульсов	Состояние триггеров			
	Q_4	Q_3	Q_2	Q_1		Q_4	Q_3	Q_2	Q_1
0	0	0	0	0	8	1	0	0	0
1	0	0	0	1	9	1	0	0	1
2	0	0	1	0	10	1	0	1	0
3	0	0	1	1	11	1	0	1	1
4	0	1	0	0	12	1	1	0	0
5	0	1	0	1	13	1	1	0	1
6	0	1	1	0	14	1	1	1	0
7	0	1	1	1	15	1	1	1	1

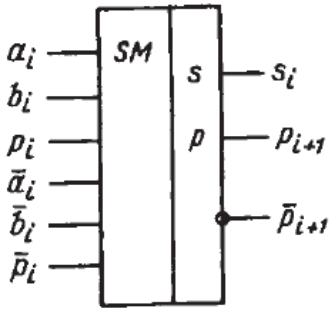
Одноразрядный двоичный сумматор. В каждом из разрядов определяется цифра суммы путем сложения по модулю 2 цифр слагаемых и поступающего в данный разряд переноса и формируется перенос, передаваемый в следующий разряд.

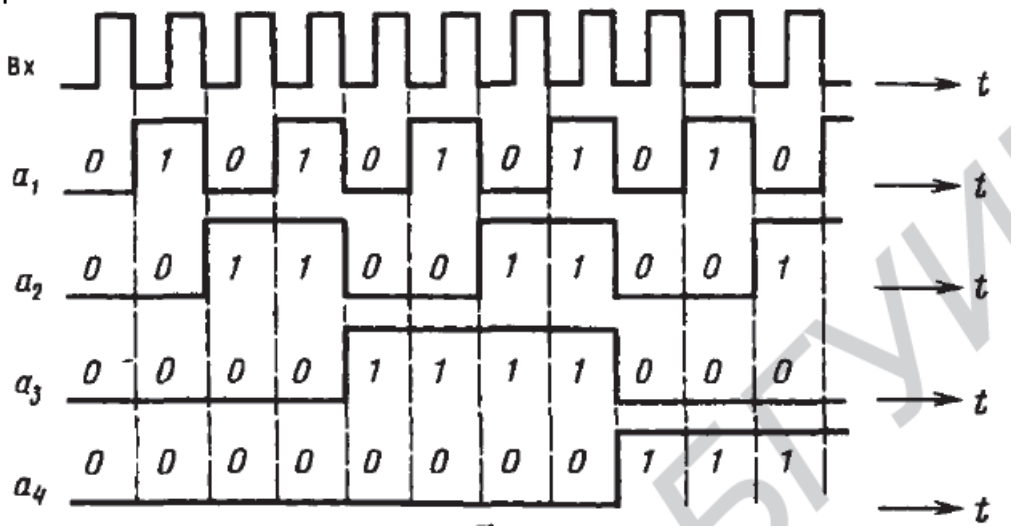
$$s_i = \overline{\bar{a}_i \cdot \bar{b}_i \cdot \bar{p}_i \vee a_i \cdot b_i \cdot \bar{p}_i \vee \bar{a}_i \cdot b_i \cdot p_i \vee a_i \cdot \bar{b}_i \cdot p_i}$$

$$p_{i+1} = \overline{\bar{a}_i \cdot \bar{b}_i \vee \bar{a}_i \cdot \bar{p}_i \vee \bar{b}_i \cdot \bar{p}_i}.$$

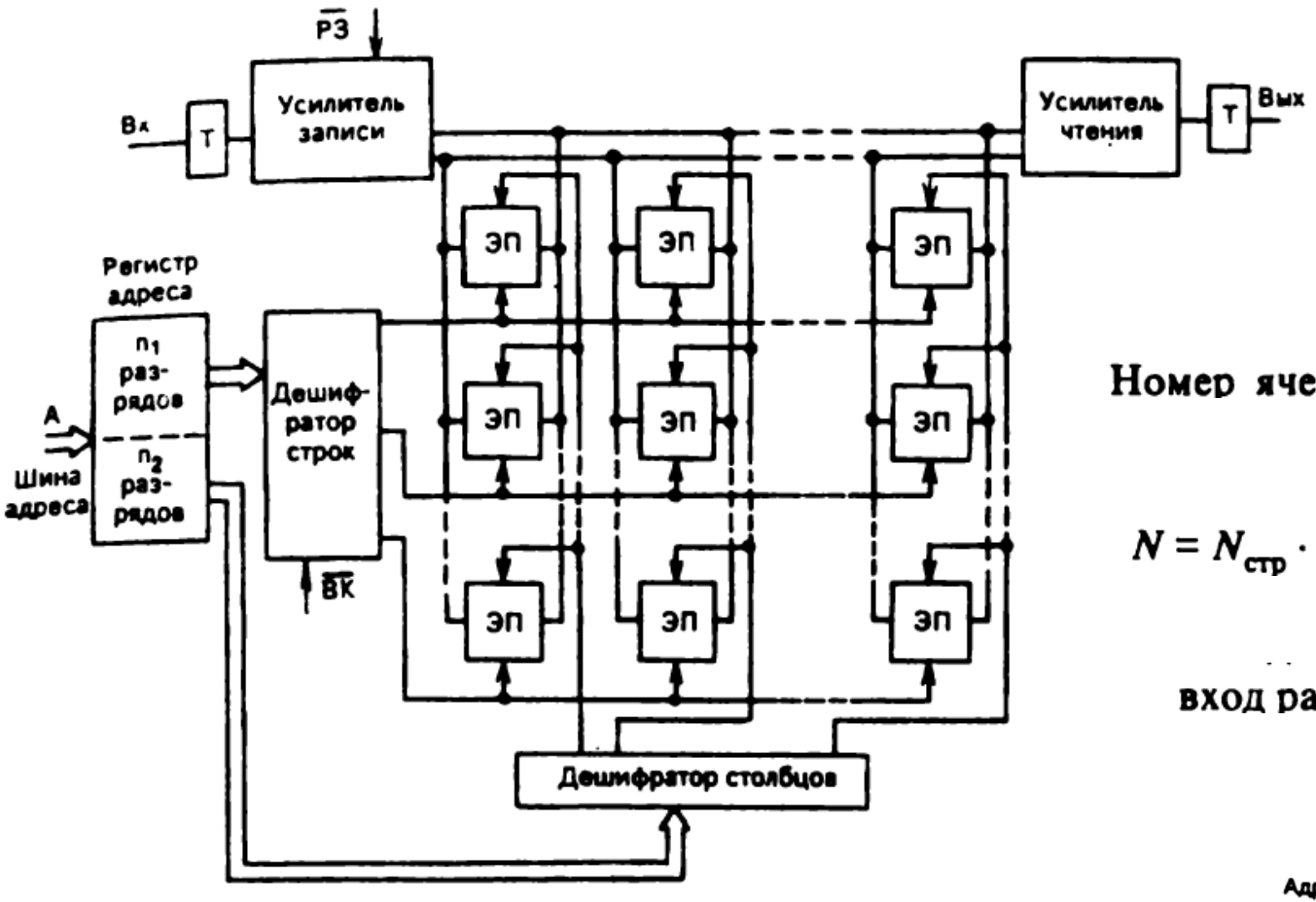


а)





Память



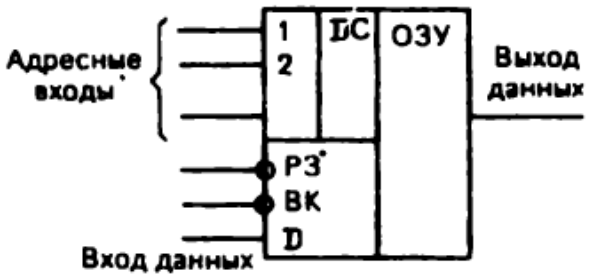
$$N = 2^m$$

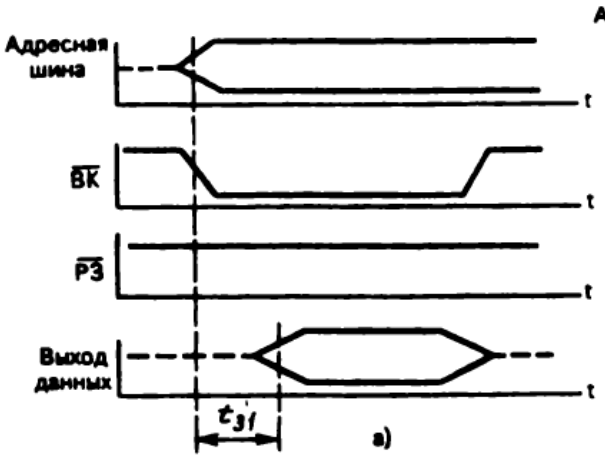
$$M = N \cdot n \text{ бит}$$

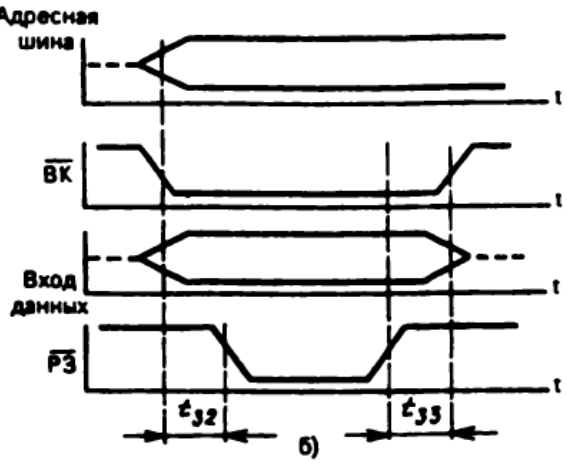
Номер ячейки называется *адресом*

$$N = N_{стр} \cdot N_{ст} = 2^{n_1} \cdot 2^{n_2} = 2^{n_1 + n_2} = 2^n$$

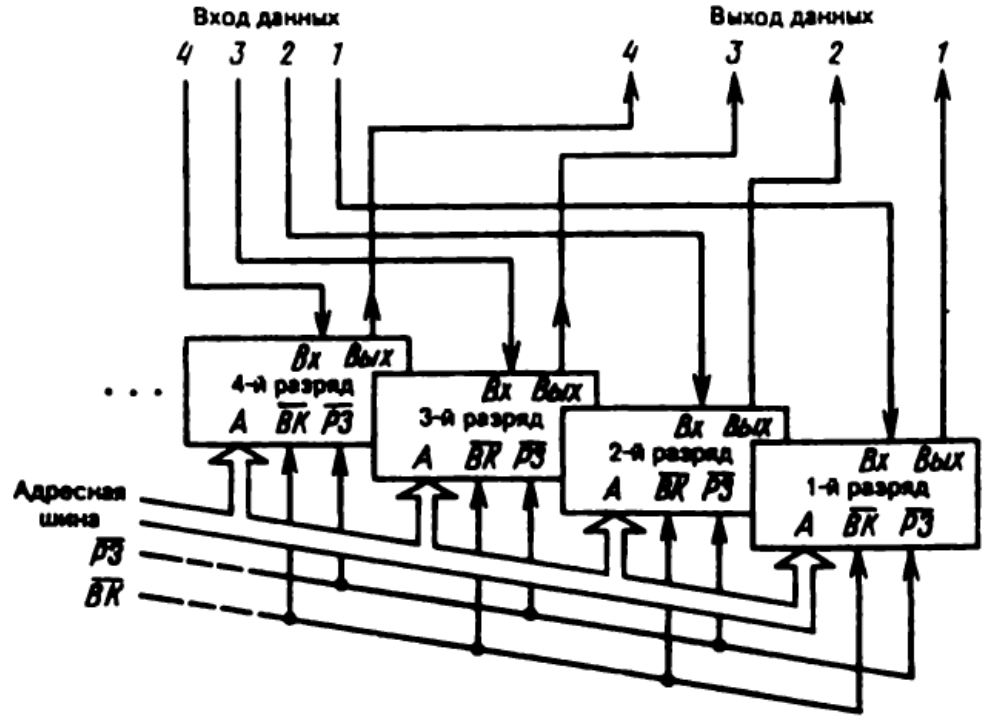
вход разрешения записи ($\overline{РЗ}$)





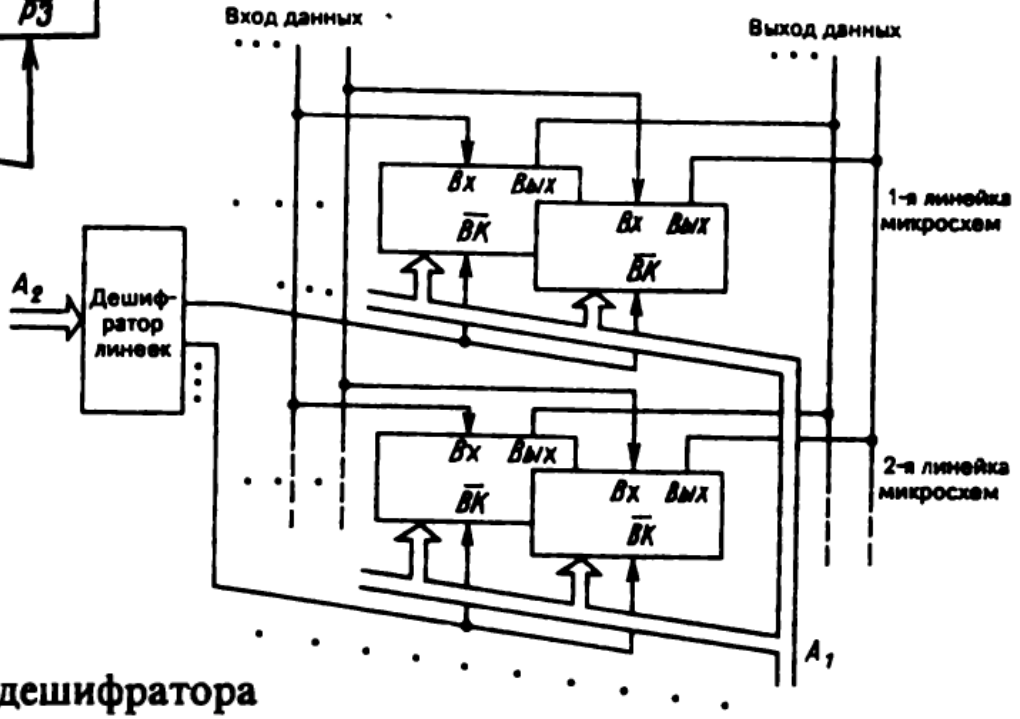


Наращивание памяти



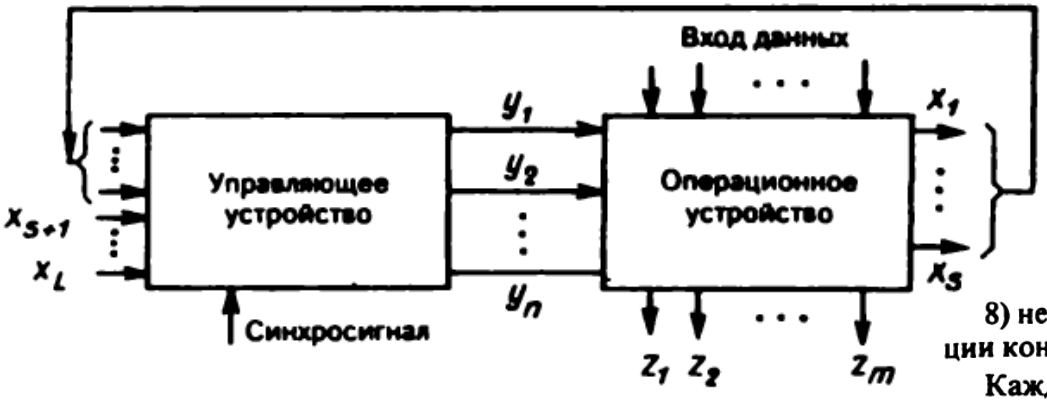
Группа разрядов A_2 номер линейки

A_1 — номер ячейки в выбранной линейке



Выбор линейки осуществляется с помощью дешифратора

Структура процессора (обобщенная)



Микрооперации

1) установка регистра в некоторое состояние
2) инвертирование
3) пересылка содержимого одного узла в другой
4) сдвиг 5) счет 6) сложение 7) сравнение

8) некоторые логические действия (поразрядно выполняемые операции конъюнкции, дизъюнкции и др.).

Каждое такое элементарное действие, выполняемое в одном из узлов ОУ в течение одного тактового периода, называется *микрооперацией*.

Операционное устройство (ОУ) — устройство, в котором выполняются операции. Оно включает в качестве узлов регистры, сумматоры, каналы передачи информации, мультиплексоры для коммутации каналов, шифраторы, дешифраторы и т.д. ***Управляющее устройство*** (УУ) координирует действия узлов операционного устройства; оно вырабатывает в некоторой временной последовательности управляющие сигналы, под действием которых в узлах операционного устройства выполняются требуемые действия.

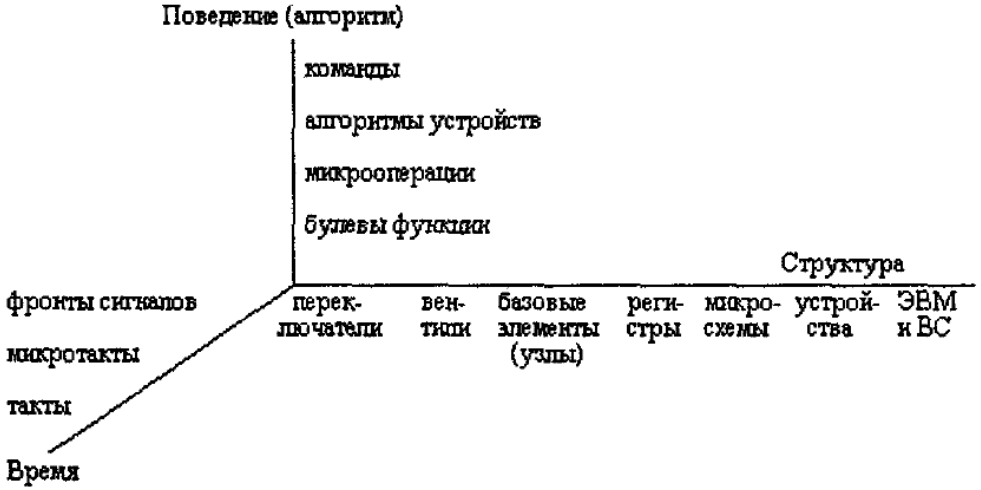
совокупность одновременно выполняемых микроопераций называется *микрокомандой*, а весь набор микрокоманд, предназначенный для решения определенной задачи, — *микропрограммой*.

Формирование управляющих сигналов $y_1, \ldots, y_n$ для выполнения микрокоманд может происходить в зависимости от состояния узлов операционного устройства, определяемого сигналами $x_1, \ldots, x_s$, которые подаются с соответствующих выходов операционного устройства на входы управляющего устройства. Управляющие сигналы $y_1, \ldots, y_n$ могут также зависеть от внешних сигналов $x_{s+1}, \ldots, x_L$.

HDL общие сведения

HDL (Hardware Description Language) - язык описания аппаратуры. VHDL и Verilog были разработаны в 1983 г.

VHDL (VHSIC (Very high speed integrated circuits) Hardware Description Language) — язык описания аппаратуры интегральных схем. Язык проектирования VHDL является базовым языком при разработке аппаратуры современных вычислительных систем. Разработан с целью формального описания логических схем для всех этапов разработки электронных систем, начиная модулями микросхем и заканчивая крупными вычислительными системами.



Возможности HDL в области отображения основных характеристик (структурный, временной, функциональный аспект) аппаратуры

Verilog, Verilog HDL (англ. Verilog Hardware Description Language) —язык описания аппаратуры, используемый для описания и моделирования электронных систем. Verilog HDL, наиболее часто используется в проектировании, верификации и реализации (например, в виде СБИС) аналоговых, цифровых и смешанных электронных систем на различных уровнях абстракции. Синтаксис похож на C (конструкции «if», «while»)

HDL модульность

Полное HDL-описание каждого объекта проекта (в дальнейшем для краткости — объекта — entity, модуля — module) состоит из двух частей: *описания интерфейса объекта* и *описания тела объекта* (описание архитектуры — architecture в терминологии VHDL).

Интерфейс объекта проекта (module, entity) определяет его имя, входы-выходы и параметры. Входы-выходы — это состав портов (port), их имена, направленность (входы — in, input, выходы — out, output, двунаправленные — inout), разрядность (один бит — bit или вектор — bit_vector), алфавит кодирования (0, 1 — один из стандартов для VHDL или 0, 1, Z, X — стандарт для VERILOG) сигналов, способ представления их значений (целый — integer, вещественный — real) и т. д.

Например, у объекта проекта по имени SM (рис. 1.2) три входных (input, in) порта: A, B, C и один выход (output, out): S, на которые могут поступать сигналы, имеющие двоичные (bit) значения 0 или 1.

VHDL

```
entity SM is
generic(TZ: time:=0 ns);
  port (A, B, C: in bit;
        S: out bit);
end SM;
```

VERILOG

```
`timescale 1 ns/100 ps
module SM (A, B, C, S);
  parameter TZ=0;
  input   A, B, C;
  output  S;
```

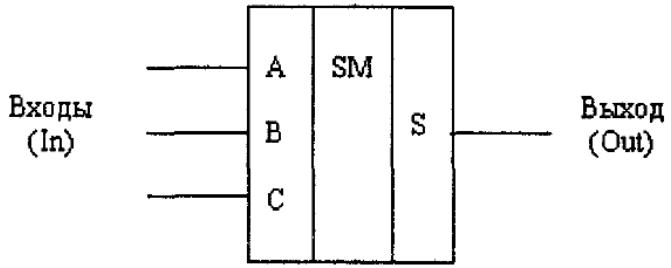


Рис. 1.2. Интерфейс объекта проекта SM

VHDL

```
D<=C after 10 ns;
C<=B after 10 ns;
B<=A after 10 ns;
```

VERILOG

```
`timescale 1 ns/1 ns
assign  #10 D=C;
assign  #10 C=B;
assign  #10 B=A;
```

Модель без задержек (бесконечно малая дельта-задержка подразумевается):

VHDL

```
D<=C;
C<=B;
B<=A;
```

VERILOG

```
assign  D=C;
assign  C=B;
assign  B=A;
```

HDL модульность

Тело объекта проекта специфицирует его структуру и функцию (**поведение,** алгоритм). Его описание в HDL VERILOG следует за описанием интерфейса **модуля**, а в языке VHDL выделяется в отдельную часть и содержится в *описании архитектуры объекта* (architecture) (рис. 1.3).

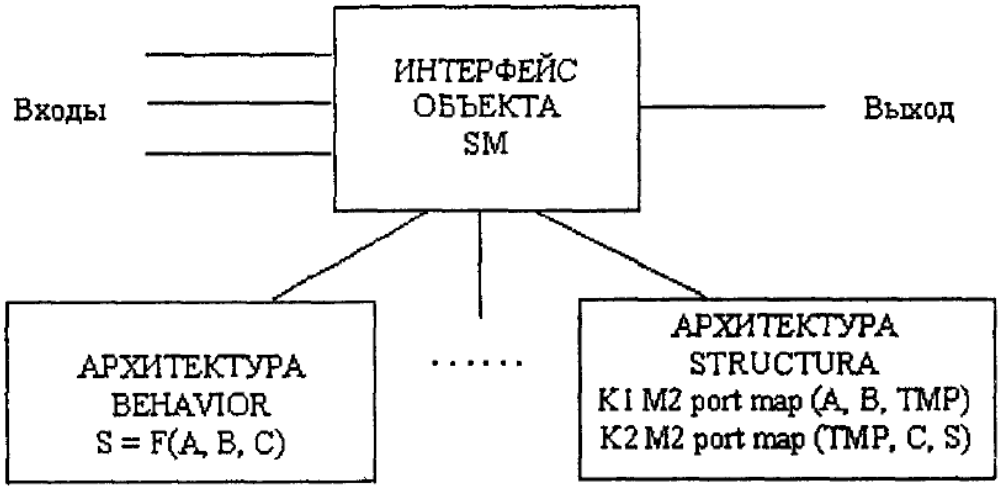


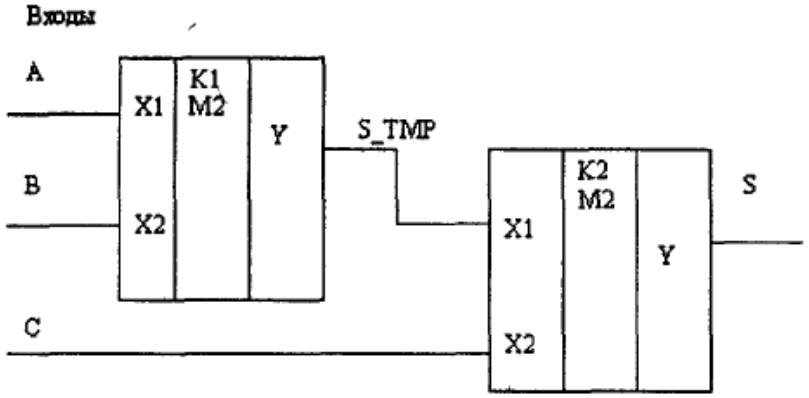
Рис. 1.3. Пример соответствия множества архитектур одному интерфейсу в VHDL

Средства HDL для отображения структур цифровых систем базируются на представлении о том, что *описываемый объект проекта* (entity, module) представляет собой структуру из более простых объектов-*компонентов* (component), соединяемых друг с другом *линиями связи* (проводами — wire (VERILOG) или сигналами — signal (VHDL). Каждый компонент, в свою очередь, является объектом и может состоять из компонент низшего уровня (иерархия объектов). Взаимодействуют объекты путем передачи *сигналов* (signal) по линиям связи. Линии связи (wire — в VERILOG) или отождествляемые с ними сигналы (signal — в VHDL) подключаются к входным (in, input) и выходным (out, output) портам (port) связываемых компонентов.

Например, компонент M2 (рис. 1.4) объекта SM имеет два входных порта — X1, X2 и один выходной — Y.

Экземпляры однотипных компонент, как уже отмечалось, могут различаться параметрами настройки (generic в VHDL, parameter в VERILOG), например задержкой сигналов.

HDL модульность



Структурный аспект объекта проекта SM, состоящего из двух экземпляров компонентов типа M2 (схема объекта SM)

--VHDL

```
architecture STRUCTURA of SM is

  signal S_TMP: bit;-- промежуточный

  begin -- для K1 позиционное
     K1: entity M2 generic map (15 ns)
          port map (A, B, S_TMP);
     -- для K2 - ключевое
     -- соответствие
     K2: entity M2  port  map
       (X1=>S_TMP,
              Y=> S,
             X2=> C);
end STRUCTURA;
```

VERILOG

```
// описание интерфейса SM было
// выше
   wire S_TMP;//промежуточный

//для K1 позиционное соответствие
        M2  #(15)
            K1 (A,B,S_TMP);
//для K2 - ключевое
//соответствие и порядок пар
//порт-сигнал произвольный
     M2  K2 (.X1(S_TMP),
             .Y(S),
              .X2(C));
endmodule //SM
```

— K1, K2 — имена экземпляров компонента M2;
— M2 — тип компонента;
— 15 ns задержка для конкретизации K1; конкретизация K2 имеет задержку 10 ns по умолчанию, VERILOG — описание SM предполагает масштаб времени 1 ns (timescale — см. ниже также описание объекта проекта M2) с точностью 100 ps;
— A, B, C, S — имена внешних сигналов, связанных с портами;
— S_TMP — имя промежуточного сигнала.

HDL модульность

Поведение объекта проекта

Компонент представляет объект проекта низшего ранга, функция которого может быть раскрыта, или он может быть, в свою очередь, описан структурно. Так, описание функции объекта проекта M2 в случае, если он реализует операцию сложения по модулю 2 (хог — исключающее ИЛИ), может быть таким:

VHDL

```
entity M2 is
generic (TDEL: time:=10 ns);
port (X1,X2 : in bit;
      Y : out bit);
end;

architecture BEH of M2 is
begin
   Y<=  (X1 xor X2) after TDEL;
end;
```

VERILOG

```
`timescale 1 ns /100 ps
module M2 (X1,X2,Y);
    parameter TDEL=10;
    input X1,X2;
    output Y;
  // конец интерфейса

   //ниже тело модуля M2

   assign #(TDEL) Y= (X1 ^ X2);
endmodule
```

Разнообразие стилей описаний архитектур

HDL допускает большое разнообразие стилей описаний объектов проекта, и, например, объект проекта SM может быть представлен чисто поведенческим способом (как «черный ящик»).

VHDL

```
entity SM_BEH is
   generic (TDEL: time:=25 ns);
   port (A, B, C: in bit;
         S: out bit);
end SM_BEH ;--конец интерфейса
arhitecture BEHAVIOUR of SM_BEH is
begin
  S<=  (A xor B xor C) after TDEL ;
end;
```

VERILOG

```
`timescale 1  ns/100 ps
module SM_BEH (A, B, C ,S);

   input   A, B, C;
   output  S;
   parameter TDEL=25;


    assign #(TDEL) S= (A ^ B ^ C);
endmodule // SM_BEH
```

Одноразрядный сумматор. Дадим пример использования объекта проекта SM-полусумматора как компонента одноразрядного сумматора adder, который помимо суммы sum формирует перенос cout. Иллюстрируется смешанный структурный — для суммы — sum и потоковый (вход регистра, преобразуясь, проходит на его выход) для переноса — cout стиль описания объекта adder.

VHDL

```
entity adder is
 generic(T_SM: time:=25 ns);
 port (a : in bit;
    b : in bit;
      cin  : in bit;
      sum  : out bit;
      cout : out bit);
end adder;
architecture mix of adder is
begin
  summa: entity SM port map
       (a,b,cin,sum);
  cout <= (a and b) or
       (cin and a) or
       (cin and b)after T_SM;
end;
```

VERILOG

```
`timescale 1ns/10 ps
module adder

(a,b,cin,sum,cout);
input a,b,cin;
output sum,cout;
parameter T_SM= 25;


SM summa
       (a,b,cin,sum);
assign #(T_SM)cout = (a & b)|
             ( cin & a) |
            (cin & b);
endmodule
```

HDL тестирующая программа

Тестирующая программа. Здесь приведен пример программы, тестирующей одноразрядный сумматор путем подачи двух входных наборов: a, b, c = 000,111.

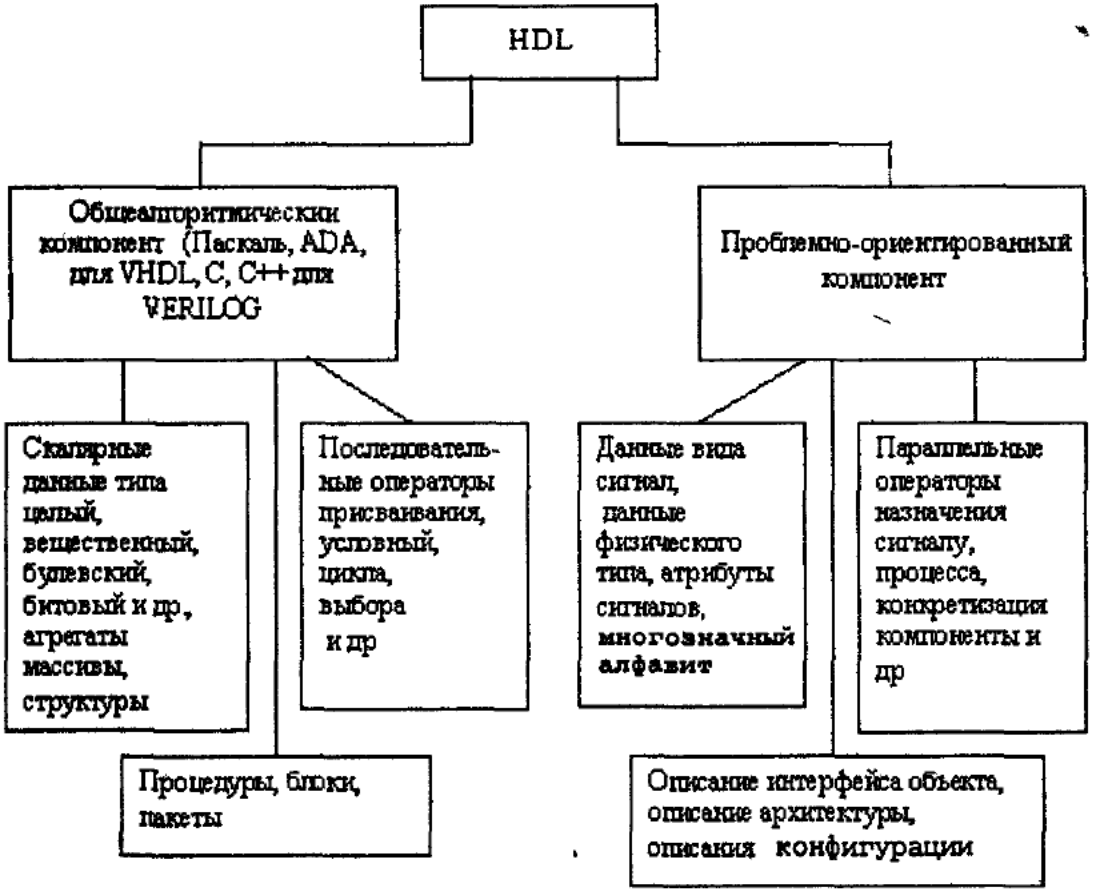
VHDL

```
entity adder_tb is end;
architecture BEH of adder_tb is
        component adder
        port(
                a, b, cin : in bit;
                sum : out bit;
                cout : out bit);
  end component;
  -- Stimulus signals
  signal a : bit;
  signal b : bit;
  signal c : bit;
  -- Observed signals
  signal sum : bit;
 signal cout : bit;
begin
-- ниже тестовые векторы
          process begin
          a<='0';b<='0';c<='0';
          wait for 100 ns;
          a<='1';b<='1';c<='1';
          wait for 100 ns;
        end process;
        -- Unit Under Test port map
        UUT : adder
              port map (
                    a => a,
                    b => b,
                    cin => c,
                    sum => sum,
                    cout => cout
              );
      end BEH;
```

VERILOG

```
    `timescale 1 ns/100 ps
  module adder_tb;

  reg a;
  reg b;
  reg c;

  wire sum;
  wire cout;

  initial begin
  a<=0;b<=0;c<=0;
     #100;
  a<=1;b<=1;c<=1;
  #100; $finish;
  end
  //тестируется adder
       adder  UUT
            (
              .a(a),
              .b(b),
              .cin(c),
              .sum(sum),
              .cout(cout)
            );
endmodule //adder_tb
```

HDL программирование



Два компонента HDL: общеалгоритмический и проблемно-ориентированный должны быть объявлены. При описании указываются: .

— **тип объекта,** задающий диапазон возможных значений и операций (например, данные целого типа — integer, вещественного — real);

— **вид или класс объекта**, задающий дополнительные свойства, например состав применимых к объекту операторов присваивания (например, в VHDL — это **виды: константа**, **сигнал** и **переменная** — constant, signal, variable; в VERILOG — это **параметр**, **переменная** и **соединение** — parameter, reg, wire). Здесь перечислены типы и **виды** объектов (данных) VHDL и VERILOG.

VHDL

Типы	Виды
Скалярные типы:	
перечислимый (enumerated)	переменная (variable)
целый (integer)	сигнал (signal)
физический	константа (constant)
плавающий (float)	
файл (file)	
ссылочный (access)	
Агрегатные типы:	
Индексируемый (array)	
структурный (record)	

VERILOG

Типы	Виды
reg	переменная
nteger	
time	
real	
realtime	
Подвиды	
wire, wand	соединение
wor, triand	(нет)
trior	
tri0,tri1	
supply0, supply	
событие (event)	событие
integer,real	параметр
ASCII строка	(parameter)
reg,time	

HDL операторы

Оператор	Пример VHDL	Пример VERILOG
1. **Ожидания**		
условия	wait until A1=B1;	wait A1==B1;
задержки	wait for 10 ns;	#10;
события	wait on A,B;	@(A or B);
2. **Оператор присваивания** переменной (VHDL), процедурного блокирующего присваивания (VERILOG) переменной	V := V1+V2; --variable V	V = V1+V2; // reg V
3. **Последовательный оператор** назначения сигнала (VHDL) процедурного неблокирующего присваивания (VERILOG) переменной	--signal S S1<=S2;	//reg S S1<=S2;

Операция	Пример		
	VHDL	VERILOG	Результат
Арифметические операции (над целыми и вещественными — VHDL, всеми типами — VERILOG)			
Сложение	2+3	2+3	5
Вычитание	2-3	2-3	-1
Умножение	2*3	2*3	6
Деление цел.	2/3	2/3	0
Модуль	2 mod 3	2 % 3	2
Остаток	-2 rem 3	—	-2
Абсолютное	abs (-1)		1
Степень (VERILOG-2000)	2 ** 3	—	8
Унарные арифметические			
Плюс	+1	+1	1
Минус	-1	-1	-1
Минус		- 4'b1011	4'b0101

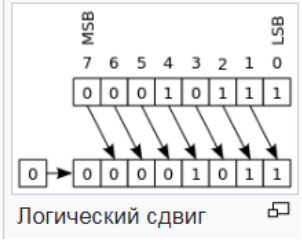
4. **Условный оператор**	if A=B then
	S1<=S2;
	end if;
		if (A==B)
		S1<=S2;

5. **Оператор выбора**	case S is
	when A => B:=Y;
	when D=>B:=notY;
	when others=>B:='1';
	end case;
		case (S)
		A: B=Y;
		D: B=~Y
		default:B=1;
		endcase

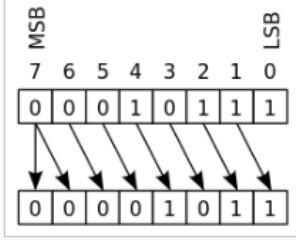
6. **Оператор цикла**	for i in 1 to 4 loop M(i):=0; end loop; while A<B loop A:=A+1; end loop;	for (i=1;i<=4;i=i+1) M[i]=0; while A<B A=A+1;
7. **Оператор возврата**	return F;	//нет аналога
8. **Оператор вызова процедуры**	P(X1,X2);	P(X1,X2);
9. **Оператор выхода** из цикла (VHDL), из группы операторов (VERILOG)	exit;	disable GG;
10. **Оператор перехода к следующей итерации** в цикле	next;	//нет аналога
11. **Оператор вывода**	report "A=B";	$display ("A=B");
12. **Пустой оператор**	null;	//нет аналога
13. **Последовательный оператор утверждения**	assert A=B report "XOXO";	//нет аналога

Verilog операторы

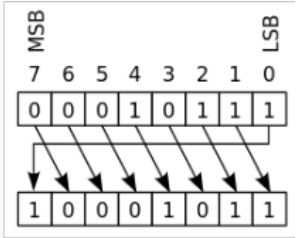
Тип	Символы	Выполняемая операция
Побитовые	~	Инверсия
	&	Побитовое AND
	|	Побитовое OR
	^	Побитовое XOR
	~^ или ^~	Побитовое XNOR
Логические	!	NOT
	&&	AND
	||	OR
Редукция	&	Редуцированное AND
	~&	Редуцированное NAND
	|	Редуцированное OR
	~|	Редуцированное NOR
	^	Редуцированное XOR
	~^ или ^~	Редуцированное XNOR
Арифметические	+	Сложение
	-	Вычитание
	-	2's complement
	*	Умножение
	/	Деление
	**	Экспонента (*Verilog-2001)
Отношение	>	Больше
	<	Меньше
	>=	Больше либо равно
	<=	Меньше либо равно
	==	Логическое равенство
	!=	Логическое неравно
	===	4-state логическое равенство
	!==	4-state логическое неравно
Сдвиг	>>	Логический сдвиг вправо
	<<	Логический сдвиг влево
	>>>	Арифметический сдвиг вправо (*Verilog-2001)
	<<<	Арифметический сдвиг влево (*Verilog-2001)
Сцепление	{ , }	Сцепление
Копирование	{n{m}}	Копирует m значение n раз
Условие	? :	Условие



Логический сдвиг



Арифметический сдвиг (правый)



Циклический сдвиг

Побитовое

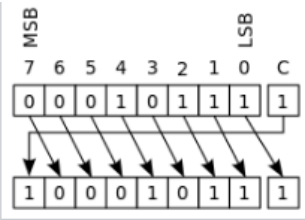
НЕ	01				
	10				
И	0011	ИЛИ	0011	Искл. ИЛИ	0011
	0101		0101		0101
	0001		0111		0110

Редукция

Вычисляется операция м-у 0-м и 1-м, между предыдущим рез-том и 2-м битом и т.д

Разряды операнда	&	|	~&	~|
Все в лог.0	0	0	1	1
Все в лог.1	1	1	0	0
Часть в лог.0, часть в лог.1	0	1	1	0

Разряды операнда	^	~^
Нечетное число бит в лог.1	1	0
Четное число бит в лог.1 (или ни одного)	0	1



Циклический сдвиг через перенос

HDL Триггер

D-триггер с информационным входом DIN и тактовым сигналом CLK. Триггер принимает информацию по фронту CLK.

Иллюстрируются средства описания фронтов сигналов.

VHDL

```
entity dff is
 port (CLK,DIN: in bit;
       DOUT: out bit);
end;
architecture beh of dff is
 begin
process (CLK)
begin
  if (CLK'event and CLK='1') then
    DOUT <= DIN ;
  end if;
end process;
end;
```

VERILOG

```
module dff( clk,din,dout);
  input clk,din;
  output dout;
  reg dout;


always @(posedge clk)
    begin

     dout <=  din;
    end
endmodule //dff
```

D-триггер с информационным входом DIN, асинхронным инверсным сбросом RST_N и тактовым сигналом CLK. Триггер принимает информацию по фронту CLK. VERILOG-описание выполнено в стиле VERILOG-2000, VHDL использует пакет STD_LOGIC_1164. Задержка выхода — 6 ns.

VHDL

```
Library IEEE;
Use IEEE.std_logic_1164.all;
entity dffr is
  port (CLK,DIN,RST_N
        : in STD_LOGIC;
        DOUT: out STD_LOGIC);
end;
architecture beh of dffr is
 begin
process (CLK, RST_N)

begin
 if  RST_N='0' then
     DOUT <= '0' after 6 ns;
elsif  (CLK'event and CLK='1') then
     DOUT <= DIN after 6 ns;
  end if;
end process;
end;
```

VERILOG-2000

```
timescale 1 ns/100 ps
  module dffr
    (input wire clk,
     din,RST_N,
    output reg dout);


  always @(posedge clk,
        negedge RST_N)
  begin
    if(!RST_N)
     dout <= #(6)  0;
     else
      dout <= #(6) din;
  end
endmodule //dffr
```

Триггер с информационным входом Din, асинхронным инверсным сбросом RST_N и тактовым сигналом CLK. Триггер принимает информацию по фронту CLK. Описание в потоковом стиле. Задержка каждого вентиля равна 1 ns. Выход триггера в VHDL-описании объявлен как INOUT, так как с него происходится считывание внутри архитектуры. В VERILOG-описании для последних двух операторов непрерывного присваивания применена сокращенная форма записи.

VHDL

```
entity dffr_c is
  port (CLK,DIN,rst_n: in bit;
        DOUT: inout bit);
end;
architecture dffr_c of dffr_c is
 signal n1,n2,n3,n4,n5,n6,
   q_n,c_n,d_n: bit;
   begin
    n2<= d_n and c_n after 1 ns;
    n3<= not(n1 and n4)after 1 ns;
   dout<= not(n5  and q_n) after 1 ns;
    n1<= not(dout and c_n and rst_n)
              after 1 ns;
      n4<= not(n2 and n3 and rst_n)
                after 1 ns;
      n5<= not(n3 and clk)after 1 ns;
      n6<= not(n4 and clk)after 1 ns;
      q_n<= not(n6 and rst_n and dout)
                    after 1 ns;
      c_n<= not(clk)after 1 ns;
      d_n<= not(din)after 1 ns;
 end;
```

VERILOG

```
   timescale 1 ns/100 ps
module dffr_c(clk,din,rst_n,dout);
    input clk,rst_n,din;
   output dout;
    reg dout;

wire n1,n2,n3,n4,n5,n6,
     q_n,c_n,d_n;

assign #(1)n2= d_n & c_n;
assign #(1)n3=~( n1 & n4);
assign #(1)dout=~( n5 & q_n);

assign #(1)n1=~(dout & c_n & rst_n);

assign #(1)n4=~(n2 & n3 & rst_n);
assign #(1)n5=~(n3 & clk);
assign #(1)n6=~(n4 & clk);

assign #(1)q_n=~(n6 & rst_n & dout);
assign #(1)c_n=~(clk),
                d_n=~(din);
endmodule //dffr_c
```

Модель RS-триггера-защелки

VHDL

```
entity RST is
   port(S, R: in bit; Q: out bit);
  -- ниже параллельный
 -- оператор утверждения
   begin postponed
    assert not((R='0') and (S='0'))
    report "RS - возможны гонки"
    severity warning;
 end RST;
 architecture POVED of RST is
  begin process (R, S)
    begin
     if (R='1') and (S='0') then
        Q<='1' after 17 ns; -- в 1
     elsif (R='0') and (S='1') then
        Q<='0' after 13 ns;
     end if;
  end process;
 end;
```

VERILOG

```
timescale 1 ns/100 ps
  module RST(S,R,Q);
    input S,R;
    output Q; reg Q;
   always  @(R or S) begin

    if  ( ((R==0) && (S==0)))
    $display("RS - возможны гонки");


          if ((R==1) && (S==0))
              Q<=#17 1;
         else  if((R==0) && (S==1))
              Q<=#13 0;


    end
endmodule
```

Присваивание с дельта-задержкой

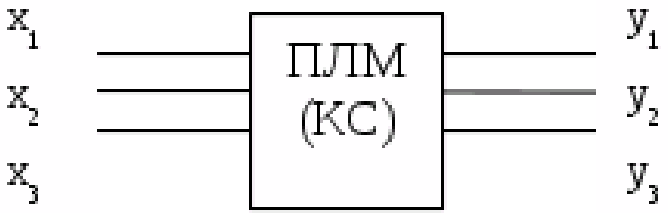
VERILOG

```
module SM_NODEL (A, B, C ,S)
 input   A, B, C;
 output  S;reg S,S_TMP;
 always @(A or B or C or S_TMP)
    begin
       S_TMP<=A ^ B;
       S<=S_TMP ^ C;
    end
    endmodule
```

Программируемые логические матрицы (ПЛМ)

ПЛМ и ПЛИС – это микросхемы, содержащие много (>тысячи) логических элементов (ЛЭ) и других компонентов, входят в довольно многочисленную группу программируемых логических приборов (ПЛП). Выпущенные изготовителем «стандартные полуфабрикаты» не могут выполнять никаких операций. ЛЭ и компоненты расположены в них в определенном порядке – матрицами, блоками, группами и др. Чтобы они могли выполнять необходимые логические операции, нужно провести заключительную операцию – программирование, которое осуществляется пользователем, без участия изготовителя. При программировании имеющиеся в «полуфабрикатах» ЛЭ организуются в специальные логические структуры, выполняющие заданные логические функции. *Изготавливать специализированную СБИС целесообразно при большом объеме выпуска (более 10 000 штук в год). В отладочных и мелкосерийных партиях выгодно ПЛП.*

ПЛМ – комбинационное устройство, включающее в себя две матрицы ЛЭ (или одну), расположенных на кристалле микросхемы. Соединение этих ЛЭ в определенные логические схемы, выполняющие заданный набор логических функций, производится разработчиком аппаратуры. Программирование превращает «полуфабрикат» в законченное функциональное изделие.

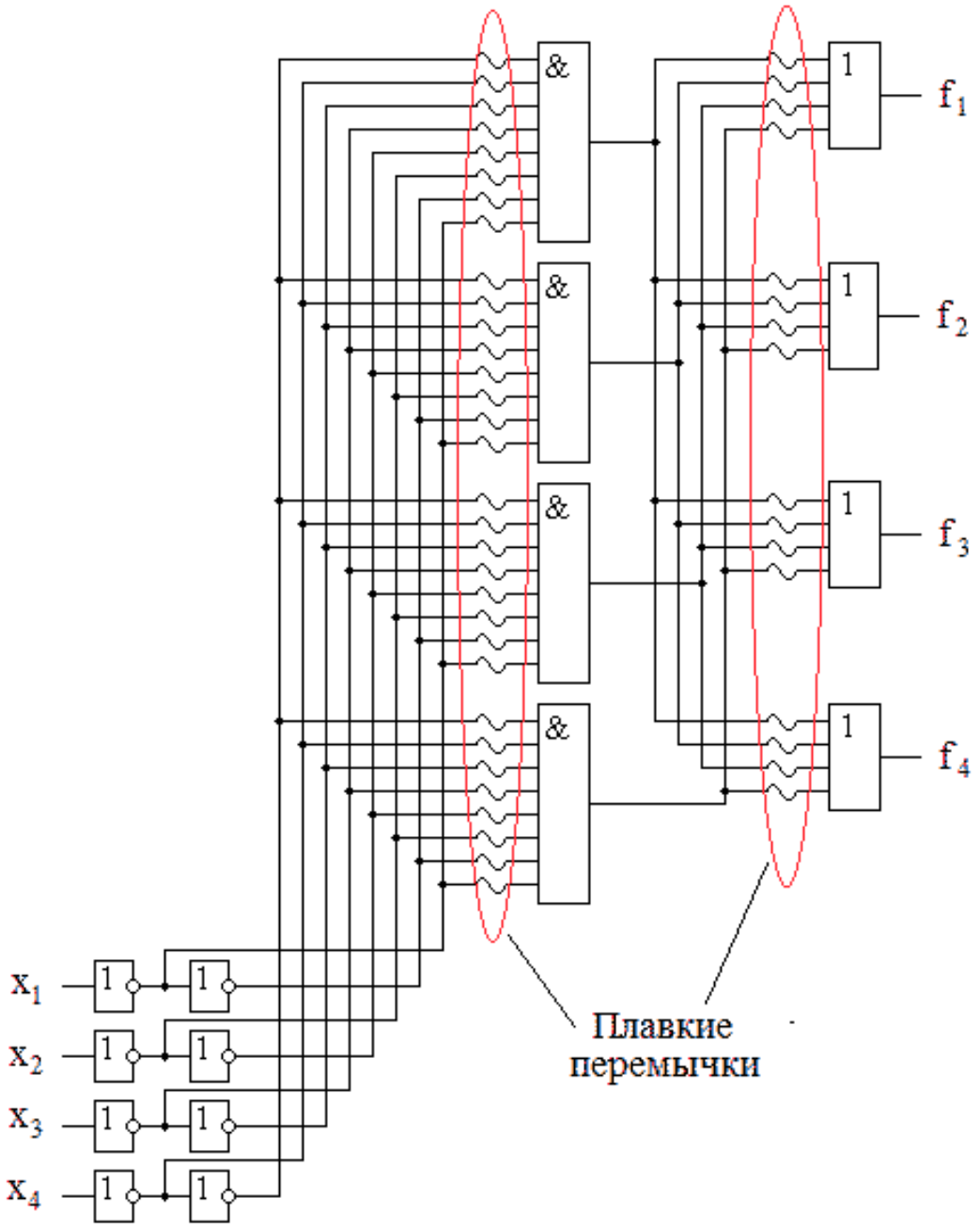


Основу ПЛМ составляют две ступени ЛЭ и входные ячейки (инверторы-повторители). 1-я ступень представляет собой матрицу ЛЭ типа И (конъюнкторов), 2-я– матрицу элементов ИЛИ (дизъюнкторов). Выходные функции задаются потребителем в виде ДНФ.

Структура ПЛМ

ПЛМ (PLA — Programmable logic Array)
заключается в реализации логической функции, представленной в СДНФ.
"И" способны реализовать любой минтерм СДНФ, "ИЛИ" осуществляют суммирование термов по логическому выражению СДНФ.

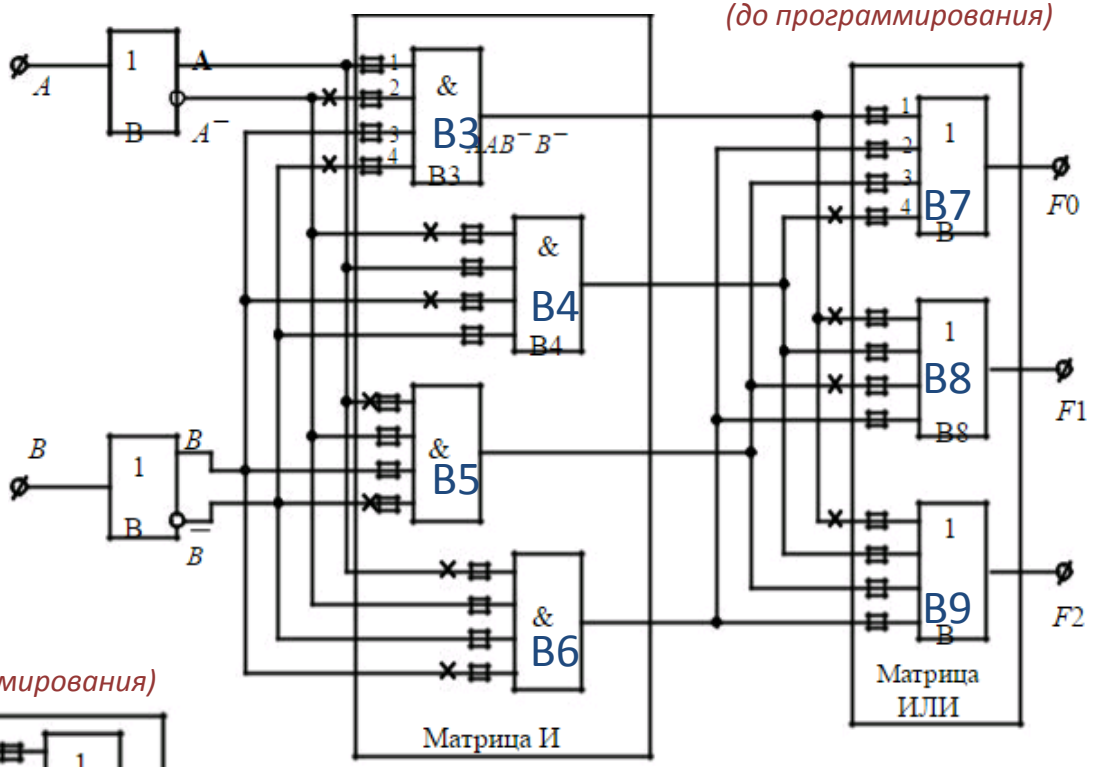
В схеме ПЛМ ранг терма ограничен количеством входов и равен 4, количество термов тоже равно четырем.
В выпускавшихся микросхемах ПЛМ количество входов 16 (максимальный ранг минтерма 16), количество термов 32 и количество выходов микросхемы 8.



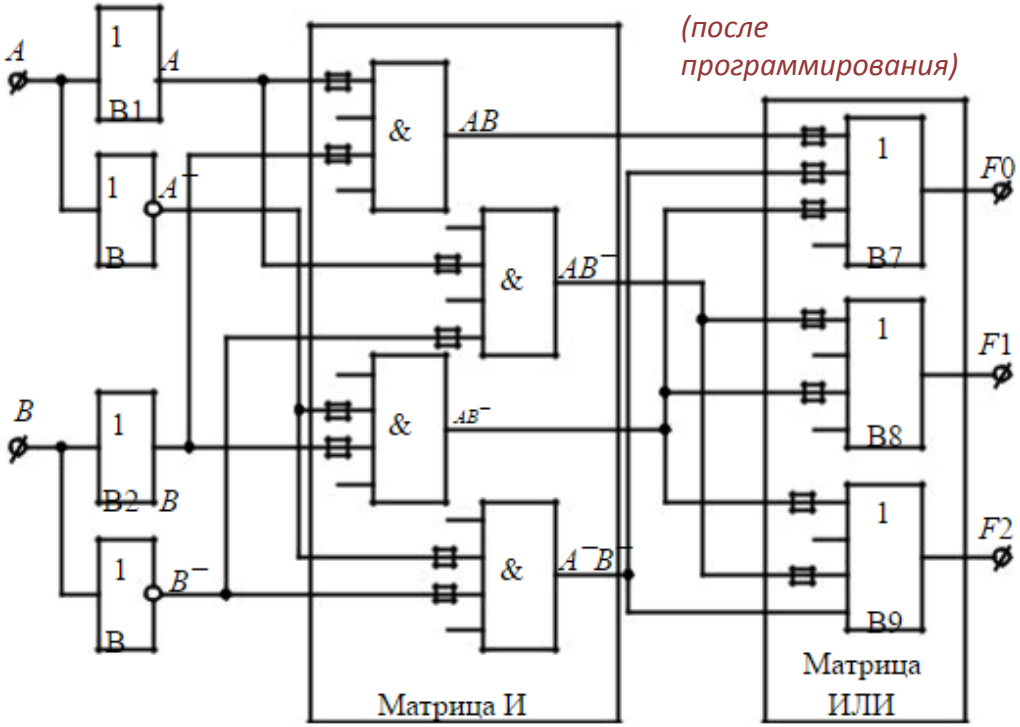
ПЛМ на плавких перемычках

Каждый из вх. (A,B) и их инверсий соединяется с одним из вх. всех схем И через плавкие перемычки (ПП). Вых. каждого из конъюнкторов (В3-В6) соединяются с входами всех схем ИЛИ (В7-В9) через ПП. Без пережигания ПП на всех вых. И образуются одинаковые лог. произведения (термы) A ~A B ~B , а на всех вых. ИЛИ – одинаковые ф-ции Fi из 4-х одинаковых термов . Этот терм равен нулю (A ~A = 0, B ~B = 0). Программирование ПЛМ производится пережиганием тех перемычек, которые окажутся ненужными для выполнения заданных ф-ций Fi. В результате программирования часть ЛЭ может быть исключена.

(до программирования)



(после программирования)



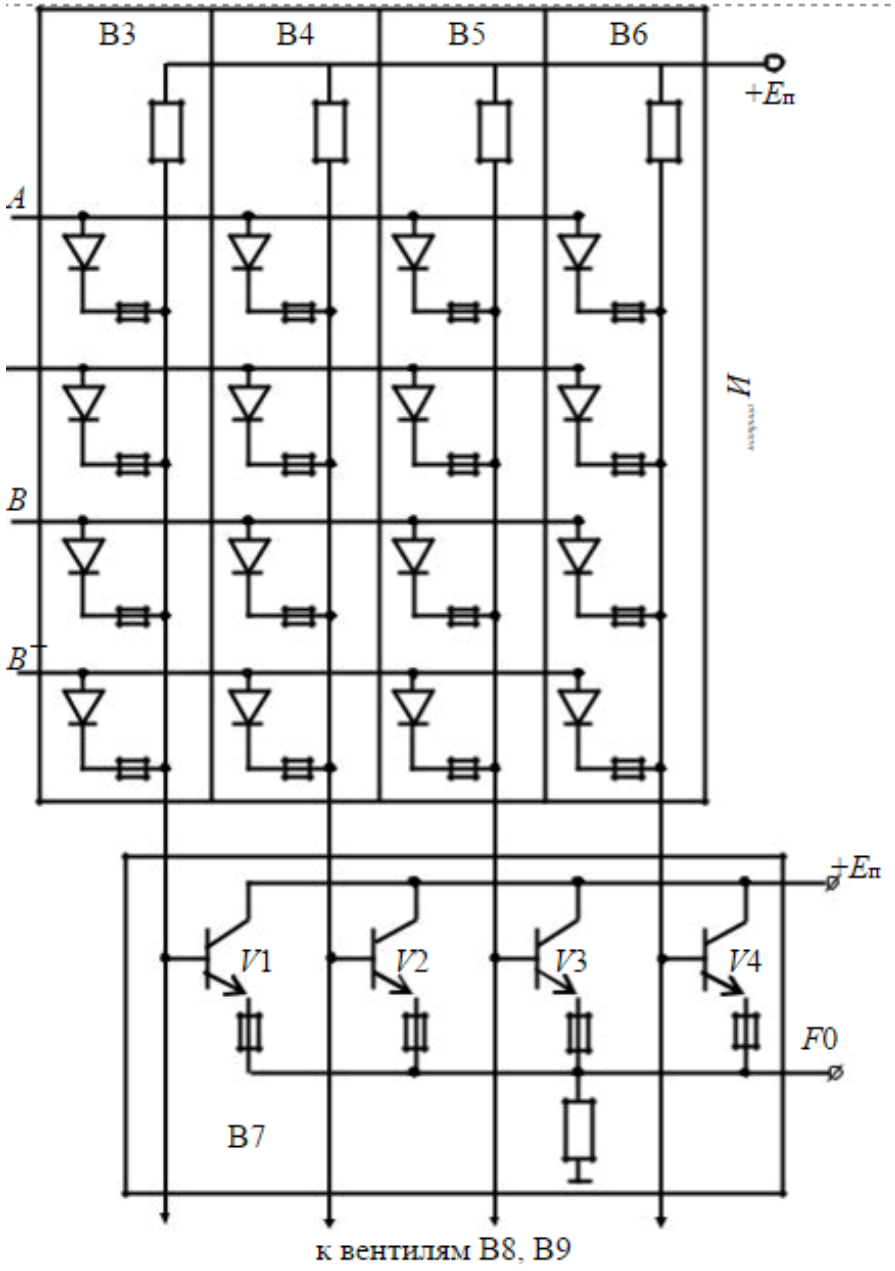
Например, при программировании пережжены ПП, которые указаны значком **x** (вх. 2, 4 вентиля В3 и др.). Разорванные перегоревшими ПП соединения на схеме не показаны, от них остались свободные выводы. На выходах конъюнкторов приведены образуемые ими термы AB,A~B ,~AB, ~A~B. На вых. ПЛМ (выходы вентилей В7, В8, В9) образуются ЛФ:

F0= AB+~AB+~A~B

F1=A~B+A~B

F2=A~B+~A~B+~A~B

ПЛМ, разновидности



Некоторые ПЛМ включают в себя до 10000 эквивалентных вентилей (двухвходовых И-НЕ или ИЛИ-НЕ). Число вых. функций (Fi) и вх. сигналов (A,B,C, …) достигает десятков, а число термов (конъюнктивных членов) – сотен.

Разновидности ПЛМ:

1. Обе матрицы могут быть выполнены на однотипных ЛЭ, например на базовых ТТЛ элементах И-НЕ. Тогда вентили второй ступени (B7, …, B9) будут выполнять функции ИЛИ.
2. Программируемой может быть только одна матрица И (И-НЕ),а матрица ИЛИ при этом имеет фиксированную (непрограммируемую) структуру (программируемая матричная логика).
3. «Полуфабрикат» ПЛМ может состоять из одних матриц ЛЭ без линий соединения. При программировании таких матриц получают специализированный фотошаблон, который используется для нанесения металлизированных соединений.
4. ПЛМ может быть репрограммируемой, т.е. можно стирать старую информацию (систему соединений ЛЭ) и производить новое программирование (ПЛМ с плавкими перемычками – нерепрограммируемая).

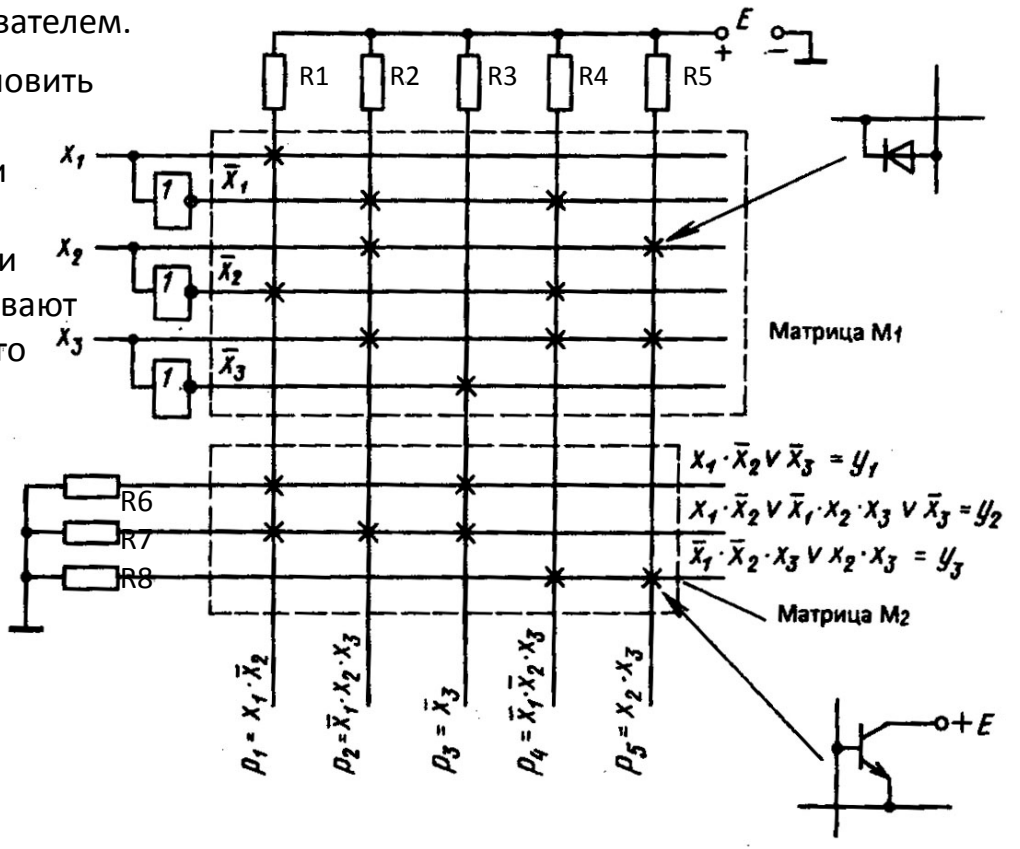
Программируемые логические матрицы

ПЛМ – это универсальная структура, позволяющая запрограммировать систему булевых функций путем организации связи между вертикальными и горизонтальными шинами. Набор этой связи программируется, и в результате ПЛМ реализует заданную систему выражений. ПЛМ может быть настроена на выполнение любой ЛФ определенной сложности. ПЛМ может осуществляться на заводе в процессе изготовления микросхемы на этапе формирования элементов в узлах матриц или пользователем.

M_1 реализует необходимые &, если необходимо установить связь в M_1, на пересечении устанавливается диод для гальванической связи между горизонтальной шиной и вертикальной шиной. Если такой связи не надо, диод прожигается и переменная не участвует в образовании конъюнкции. Сопротивления R1,R2,R3,R4,R5 обеспечивают протекание тока через диод на базис соответствующего транзистора.

M_2 программируется для организации ДНФ. На пересечении программируемые транзисторы. В цепи эмиттера имеется сопротивление, которое при программировании прожигается. В результате связь между горизонтальной и вертикальной шиной теряется.

При программировании связи **X** в M_2 при срабатывании транзистора, соответственно на R6, R7 или R8 появляется *1* (функции y_1-y_3).
Т.к. любая ЛФ м.б. представлена в СДНФ, которая минимизируется, программирование ПЛМ позволяет осуществить построение матрицы любой комб.схемы.
Схема работает только при наличии вх. сигналов и не запоминает предыдущее состояние.



	x_1	x_2	x_3	y_1	y_2	y_3
p_1	1	0	—	1	1	.
p_2	0	1	1	.	1	.
p_3	—	—	0	1	1	.
p_4	0	0	1	.	.	1
p_5	—	1	1	.	.	1

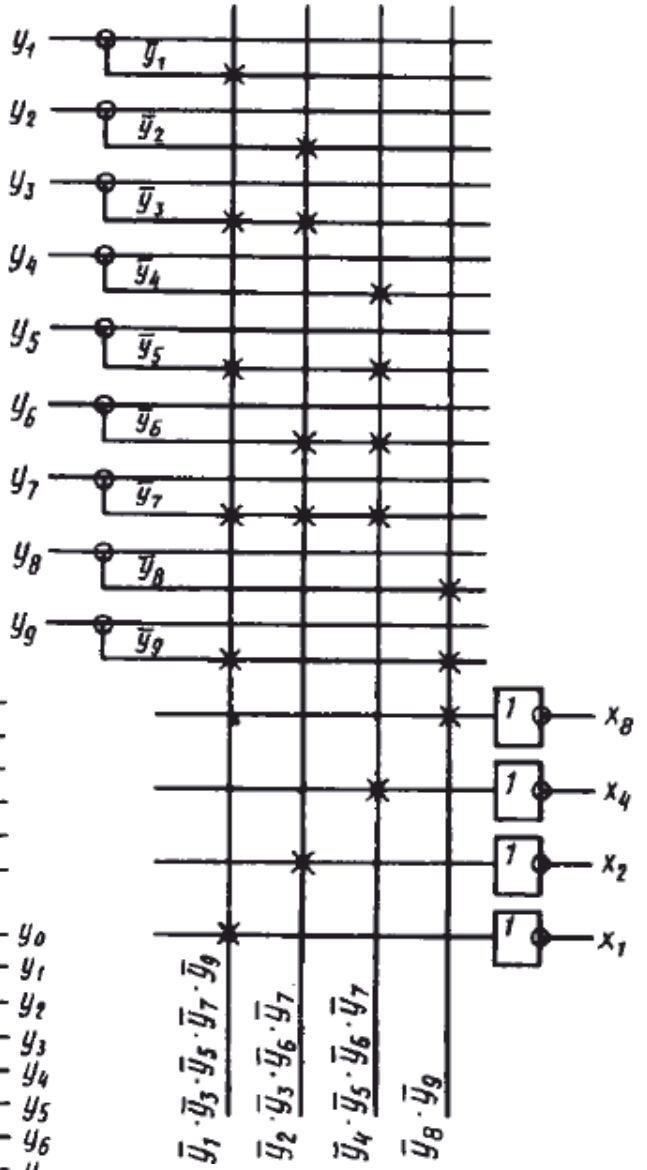
ПЛМ. Пример реализации шифратора/дешифратора

Шифратор. Рассмотрим построение шифратора, преобразующего унитарный десятичный код (с отображением десятичной цифры уровнем *лог.1* на одной из десяти цепей) в двоичный *код 8421*. Воспользуемся полученными в § 3.5 для дешифратора логическими выражениями

$$x_1 = \bar{y}_1 \mid \bar{y}_3 \mid \bar{y}_5 \mid \bar{y}_7 \mid \bar{y}_9,\ x_2 = \bar{y}_2 \mid \bar{y}_3 \mid \bar{y}_6 \mid \bar{y}_7,$$

$$x_4 = \bar{y}_4 \mid \bar{y}_5 \mid \bar{y}_6 \mid \bar{y}_7,\ x_8 = \bar{y}_8 \mid \bar{y}_9,$$

где y_i — входные сигналы; x_j — выходные сигналы (значения разрядов *кода 8421*).



Дешифратор. Реализацию на ПЛМ дешифратора рассмотрим на примере дешифратора, преобразующего трехразрядный двоичный код (x_1, x_2, x_4), подаваемый на вход, в унитарный 8-разрядный код на выходе (y_0, ... , y_7).

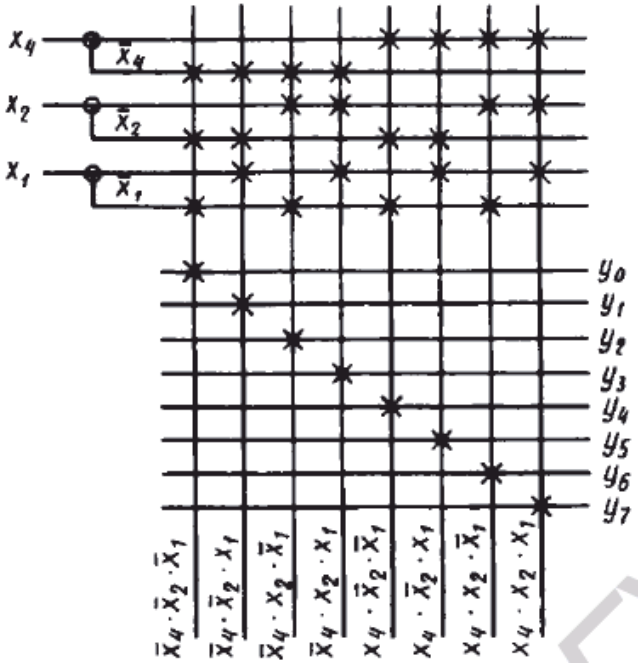
Функционирование такого дешифратора определяется следующими логическими выражениями:

$$y_0 = \bar{x}_4 \cdot \bar{x}_2 \cdot \bar{x}_1,\ y_1 = \bar{x}_4 \cdot \bar{x}_2 \cdot x_1,$$

$$y_2 = \bar{x}_4 \cdot x_2 \cdot \bar{x}_1,\ y_3 = \bar{x}_4 \cdot x_2 \cdot x_1,$$

$$y_4 = x_4 \cdot \bar{x}_2 \cdot \bar{x}_1,\ y_5 = x_4 \cdot \bar{x}_2 \cdot x_1,$$

$$y_6 = x_4 \cdot x_2 \cdot \bar{x}_1,\ y_7 = x_4 \cdot x_2 \cdot x_1.$$



ПЛМ. Пример реализации мультиплексора/демультиплексора

схема мультиплексора с четырьмя входами (D_3, D_2, D_1, D_0). Здесь A_1, A_0 — адресные входы; C — вход для подачи сигнала разрешения выдачи; y — выход.

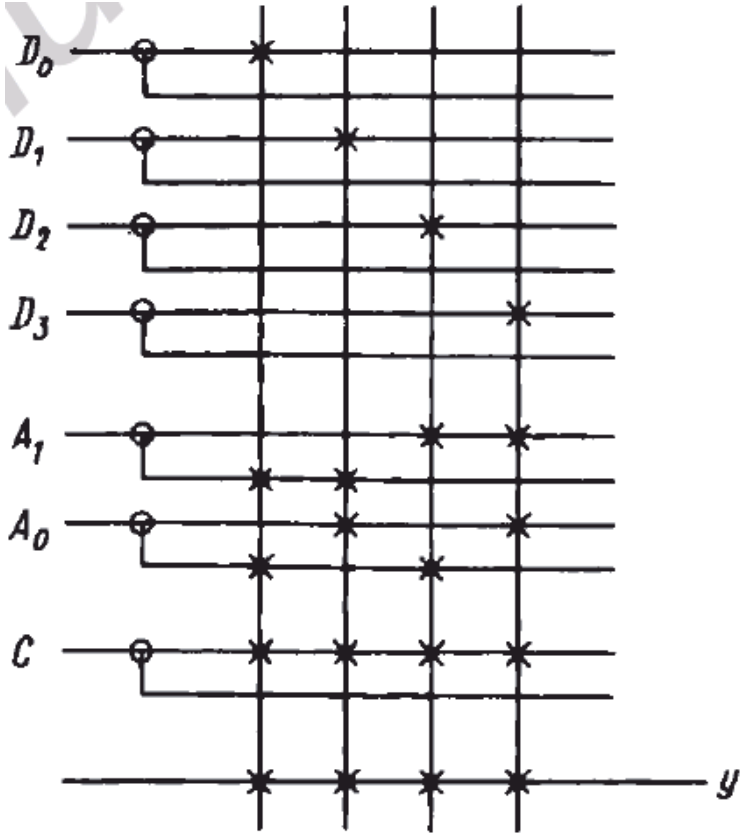
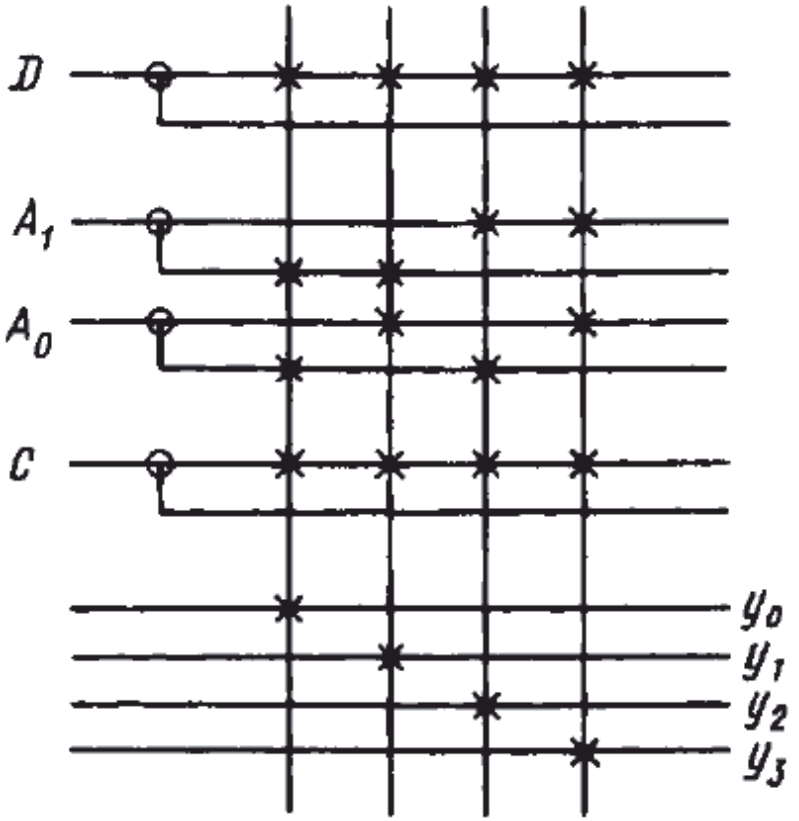


Схема де-мультиплексора с четырьмя выходами (y_3, y_2, y_1, y_0). Здесь D — вход; A_1, A_0 — адресные входы; C — вход сигнала разрешения выдачи.



ПЛМ. Пример реализации регистра

Регистр. Регистр является устройством с памятью (устройством последовательностного типа). Реализация такого типа устройства на ПЛМ требует наличия в ПЛМ элементов памяти — триггеров, образующих регистр. В такой ПЛМ матрицы M_1 и M_2 используются для построения комбинационной схемы, с помощью которой регистру придаются дополнительные свойства, кроме простого хранения кода. Одним из таких свойств может быть, например, сдвиг содержимого регистра влево или вправо (на один или несколько разрядов). Рассмотрим построение универсального регистра, обладающего следующими возможностями:

прием извне в регистр кодовой комбинации, поступающей в регистр;

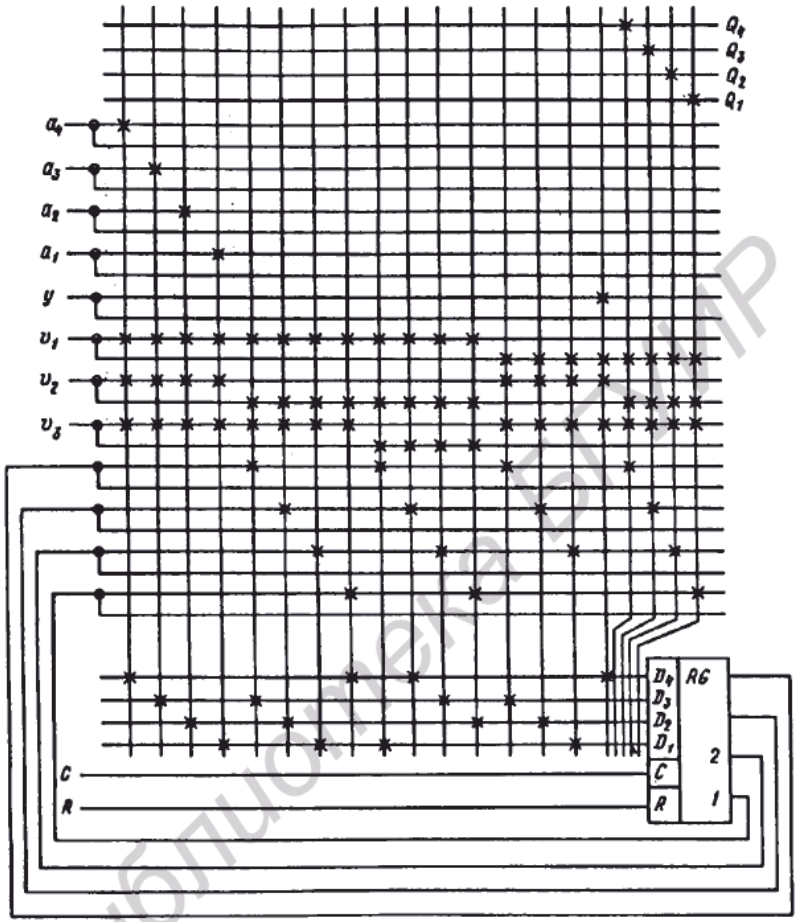
циклический сдвиг содержимого регистра влево или вправо (сдвиг с передачей выдвигаемой из регистра цифры в освобождающий при сдвиге разряд регистра);

сдвиг вправо с приемом в освобождающийся старший разряд регистра цифры, подаваемой на вход;

выдача содержимого регистра на выход.

v_1	v_2	v_3	Выполняемая функция
0	0	1	Выдача содержимого регистра
0	1	1	Сдвиг вправо с приемом y в старший разряд
1	0	0	Циклический сдвиг влево
1	0	1	Циклический сдвиг вправо
1	1	1	Прием в регистр

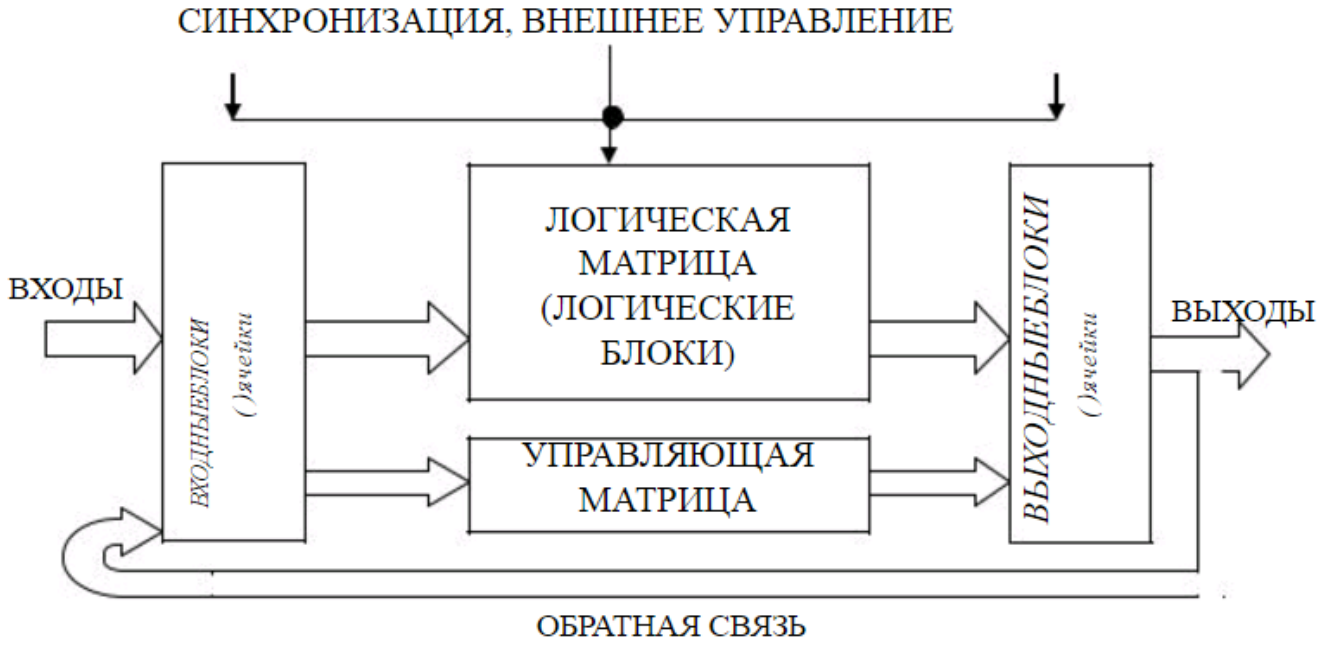
Схема четырехразрядного регистра с указанными функциями



Пусть вид выполняемой в регистре функции задается комбинацией v_1, v_2, v_3 в соответствии с табл. Число выполняемых в регистре функций ограничим пятью

Здесь a_4, a_3, a_2, a_1 — выводы для подачи входного кода; y — вывод для подачи на вход цифры, вводимой при сдвиге вправо в освобождающийся старший разряд регистра; Q_4, Q_3, Q_2, Q_1 — выводы содержимого регистра; v_1, v_2, v_3 — выводы для выбора выполняемой регистром функции

Обобщенная модель ПЛИС на основе ПЛМ



Особенности ПЛИС Xilinx (XC2018, …, XC3090):

1.ЛМ на КМОП-элементах, 100 и более ЛБ с динамически изменяемой конфигурацией, построенных на основе ячеек статических ЗУ с произвольной выборкой. 3.ЛБ имеет 4 – 5 лог. вх., 2 вых. и синхровх., реализует любую лог. ф-цию 4-5 переменных. 4. ЛБ может воспринимать на вх. и формировать на вых. сигналы положительной и отрицательной логики. 5.Вокруг ЛМ расположено несколько десятков двунаправленных БВВ с изменяемой конфигурацией. БВВ содержит вх. регистр, схему настройки вх. порогового напряжения и вых. схему с тремя состояниями (0,1, z). ЛБ и БВВ соединяются между собой. Используются соединительные элементы, программируемые потребителем: а) горизонтальные и вертикальные металлизированные линии, проложенные между ЛБ и БВВ и сегментированные; б) элементы обмена на координатных переключателях, соединяющие между собой отдельные сегменты металлизированных линий; в) программируемые соединительные точки, связывающие металлизированные линии с ЛБ и БВВ.

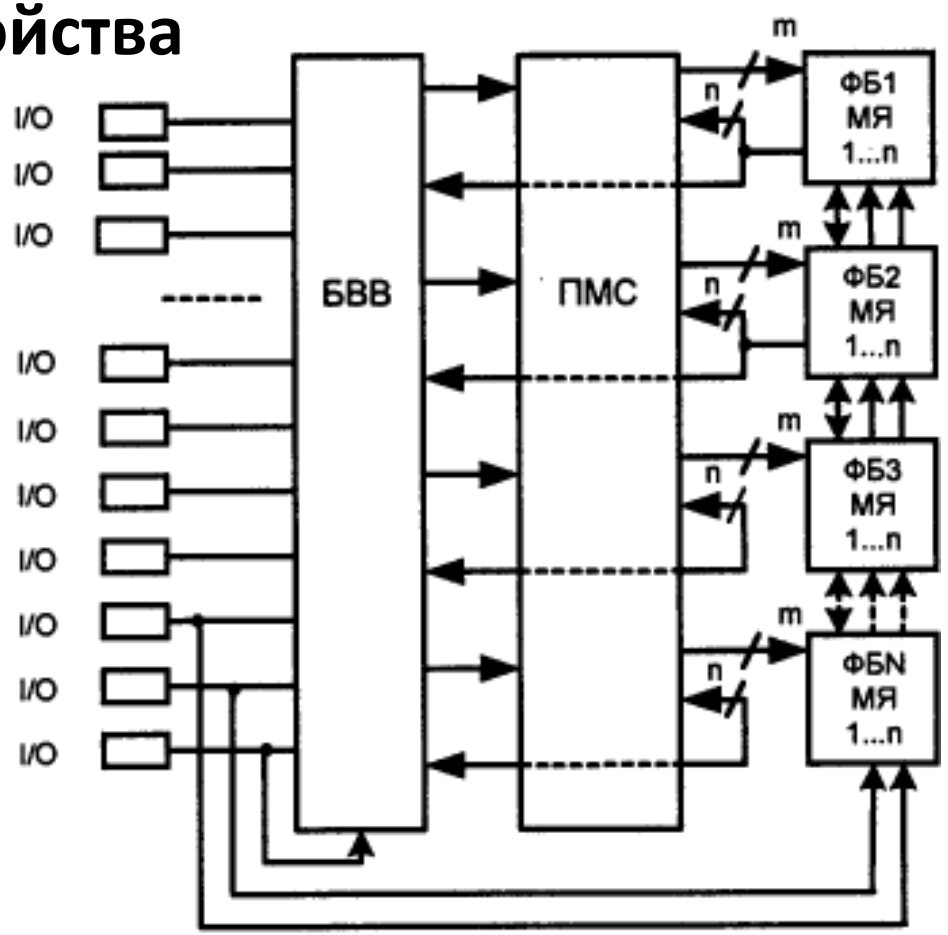
Основу ПЛИС составляет ЛМ.
Во вх. блоках формирователи парафазных сигналов (A,~A , …), и триггеры, регистры.
Вых. ячейки с управлением полярностью выхода (отрицательная или положительная логика), с запоминанием сумм произведений, с внутренней синхронизацией и т.д. Выбор функции осуществляет **управляющая матрица**.
Под действием **ОС** вых. блоки могут быть преобразованы в двунаправленные БВВ.
Синхронные узлы имеют входы синхронизации.
Сигналы внешнего управления: предустановка, сброс или загрузка регистров, разрешение (запрет) выходов и др. В составе высокоинтегрированных ПЛИС имеются макроячейки высокого уровня сложности – счётчики, мультиплексоры, дешифраторы, АЛУ и ЗУ.

CPLD – сложные программируемые логические устройства

CPLD — микросхемы высокого уровня интеграции, состоящие из:

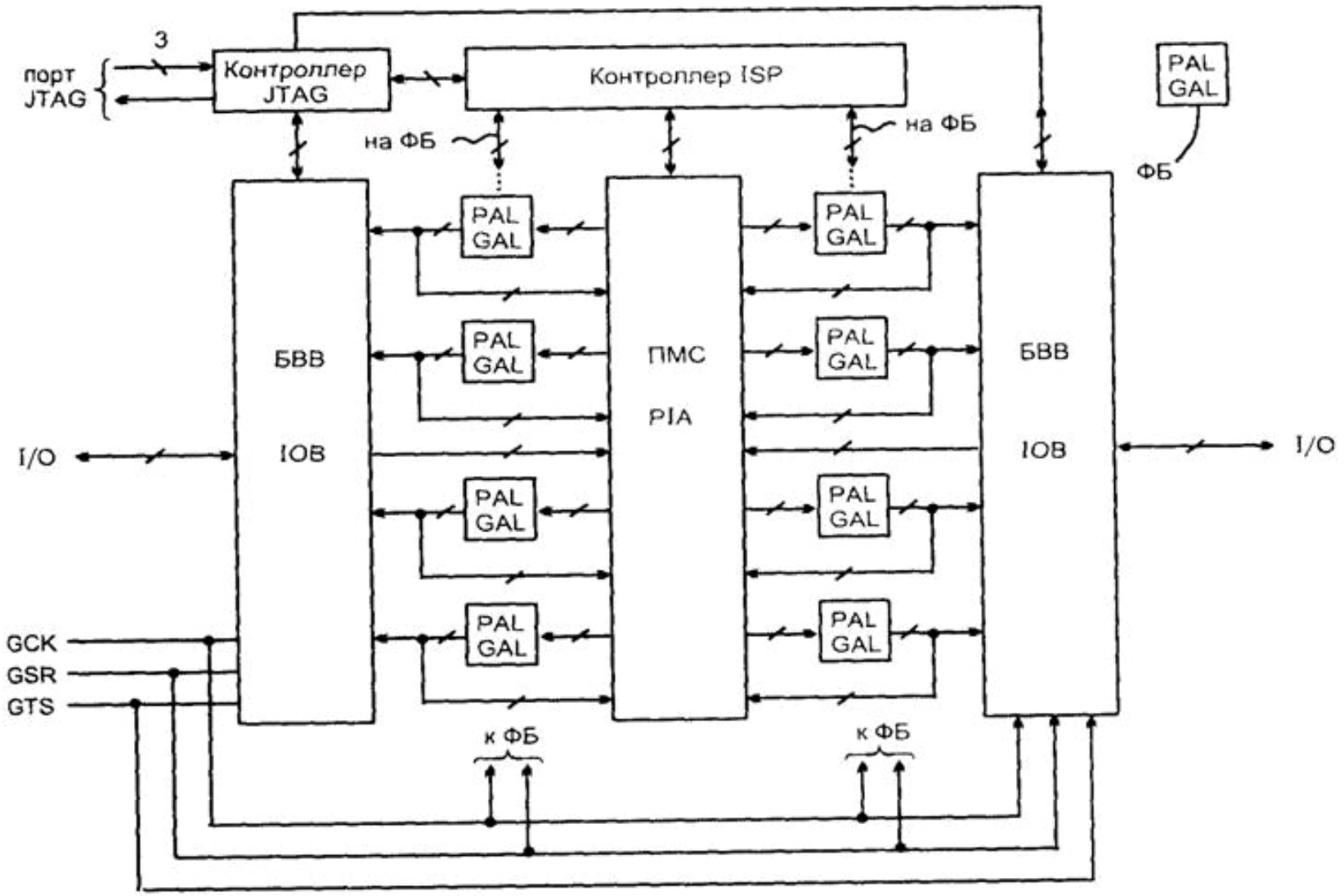
- подобных функциональных блоков (ФБ);
- системы коммутации, позволяющей объединять ФБ в единое устройство, выполненная в виде матрицы соединений;
- - блоков ввода/вывода (БВВ).

Число ФБ N зависит от уровня интеграции. В каждом ФБ n макроячеек (МЯ). ФБ получают вх. сигналы от программируемой матрицы соединений (ПМС). Число таких сигналов m. Вых. сигналы ФБ поступают как в ПМС, так и в БВВ. ПМС обеспечивает полную коммутируемость ФБ, т. е. возможность подавать сигналы с любого их вых. на любой вх.



БВВ связаны с внешними двунаправленными выводами I/O. 3 нижних вывода специализируются для подачи на матрицу ФБ сигналов GCK глобального (для всей микросхемы) тактирования, GSR глобальной установки/сброса и GTS (Global 3-state Control) глобального управления третьим состоянием вых. буферов, либо эти выводы могут быть I/O. Число контактов I/O может совпадать с числом выходов всех ФБ, или быть и меньше (часть МЯ использована для внутренних сигналов). *Кроме показанных блоков в CPLD могут быть контроллеры для управления операциями программирования непосредственно в системе (ISP, In System Programmability), JTAG и др.*

CPLD – сложные программируемые логические устройства



В CPLD используется энергонезависимая память конфигурации EEPROM или Flash, содержимое которой на самом кристалле защищается специальным битом секретности, сбросить который можно лишь при стирании всего содержимого памяти.

Блоки ввода/вывода CPLD

БВВ соединяют внешние контакты микросхемы с ее внутренними цепями. Основой БВВ служат два буфера — вх. (1) и вых. (2). Чтобы обеспечить постоянство уровней напряжения, поступающих на входной буфер, и их независимость от амплитуды входных сигналов, в схеме вырабатывается внутреннее напряжение питания VCCINT и вводится цепь из двух фиксирующих диодов. Схема программируемой общей точки ПрОТ позволяет пользователю при необходимости получать дополнительный «заземленный» вывод.

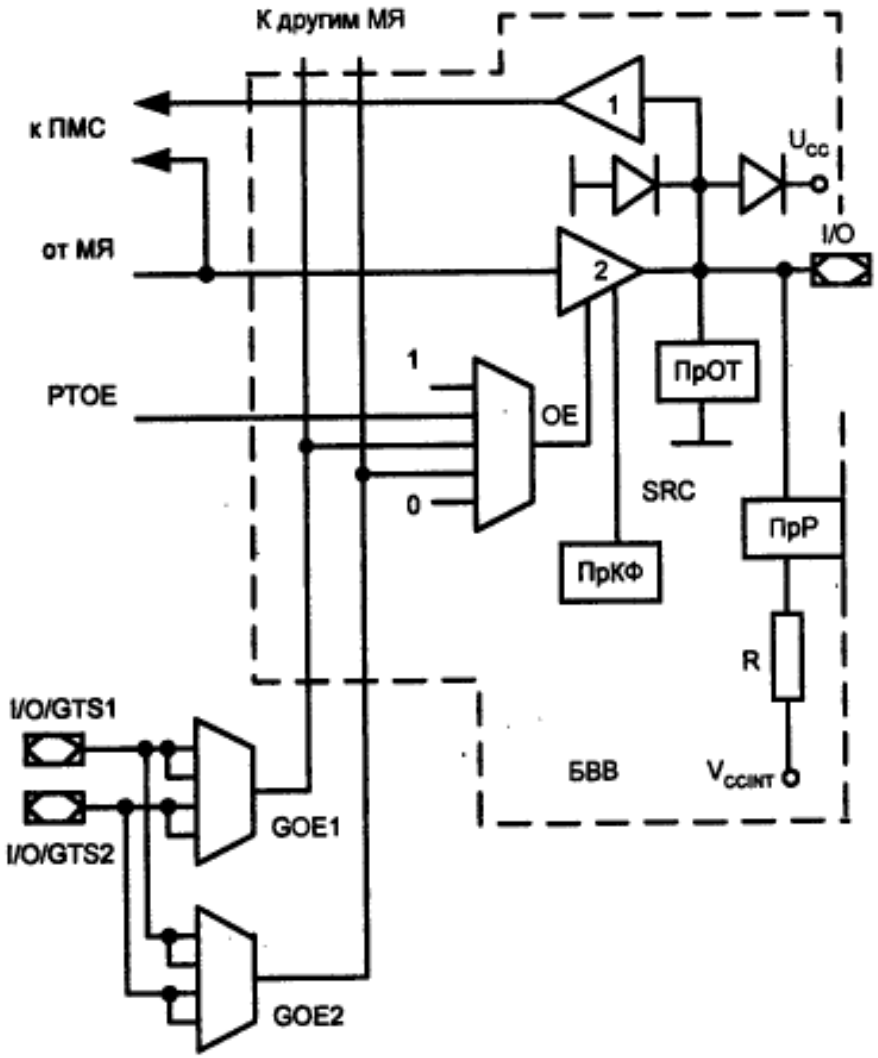


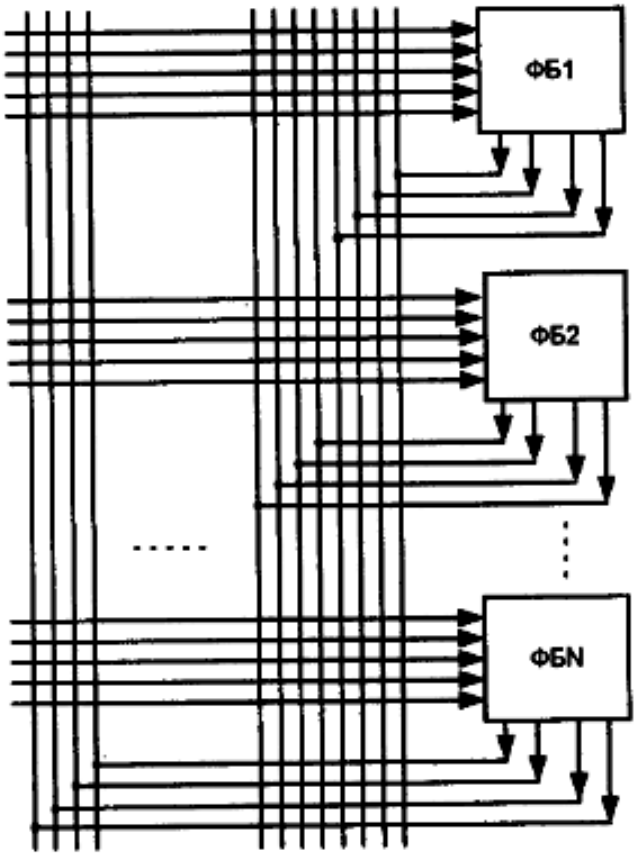
Схема программирования подключаемого резистора ПрР введена для исключения плавающих потенциалов на контактах ввода, когда они не используются в рабочем режиме. В этом случае контакту задается высокий потенциал от цепи VCCINT — R. Резистор R используется и в некоторых других режимах, а в рабочих режимах отключается. Вых. буфер 2 получает сигналы разрешения работы OE и управления крутизной фронта выходного напряжения SRC (Slew Rate Control). OE с помощью программируемого MUX3 вырабатывается : от терма PTOE, получаемого от МЯ, от любого из глобальных сигналов управления третьим состоянием (GOE1, GOE2), от константы 1 и 0. Глобальные сигналы управления третьим состоянием образуются с возможностью выбора любой полярности исходных сигналов GTS1 и GTS2. Вых. буферы конфигурируются для работы с Uпит (5 или 3,3 2,5 и 1,8 В) при подключении внешнего источника питания.

Программируемая матрица соединений

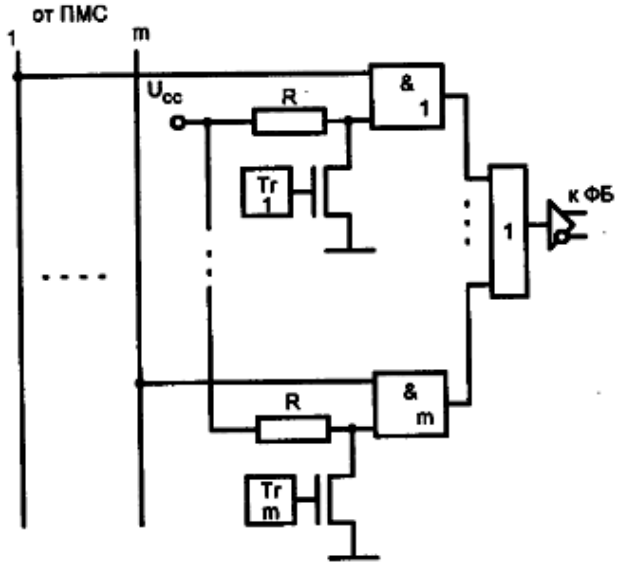
В ПМС вых. ФБ подключаются к вертикальным непрерывным (несегментированным) линиям, каждому вых. соответствует своя линия. Вх. ФБ связаны с горизонтальными линиями, пересекающими все вертикальные линии. На пересечениях горизонтальных и вертикальных линий имеются программируемые точки связи, так что любой вход ФБ может быть подключен к любому выходу. Достоинством такой ПМС является малая и предсказуемая задержка коммутируемых сигналов.

Для каждого соединения образуется идентичный всем другим канал связи с малым числом программируемых ключей или даже их отсутствием. В этом случае программируемых ключей в цепи передачи сигнала нет, программируются только напряжения на нижних входах элемента «И», и ФБ получит сигнал от i-ой вертикальной линии ПМС (i =1,2,...,m), если транзистор Ti заперт, и на втором вх. i-го элемента «И» будет действовать высокий потенциал лог. 1. Открытый транзистор Ti, подключает второй вход элемента «И» к нулевому потенциалу, создавая на нем и на выходе элемента «И» сигнал лог. 0.

Схема программируемой матрицы соединений CPLD



Передача сигналов от ПМС в ФБ



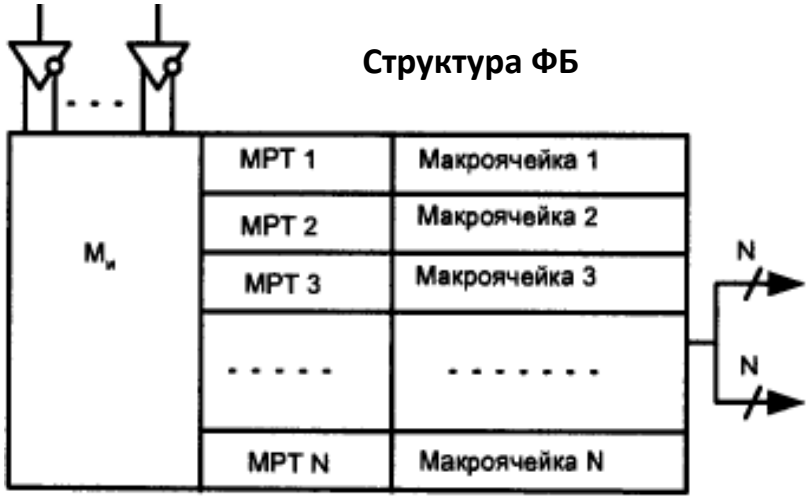
Задавая триггеру Tr_i лог.0, а остальным триггерам лог. 1, можно закрыть транзистор T_i, и открыть остальные транзисторы, что означает подключение вых. ФБ к i-ой вертикальной линии ПМС с образованием так называемого непрерывного соединения. Замкнутые транзисторные ключи имеют, схему замещения в виде инерционной RC-цепи и вносят основные задержки в процесс распространения сигнала. Такие ПМС эффективны в схемах с относительно небольшим числом коммутируемых блоков.

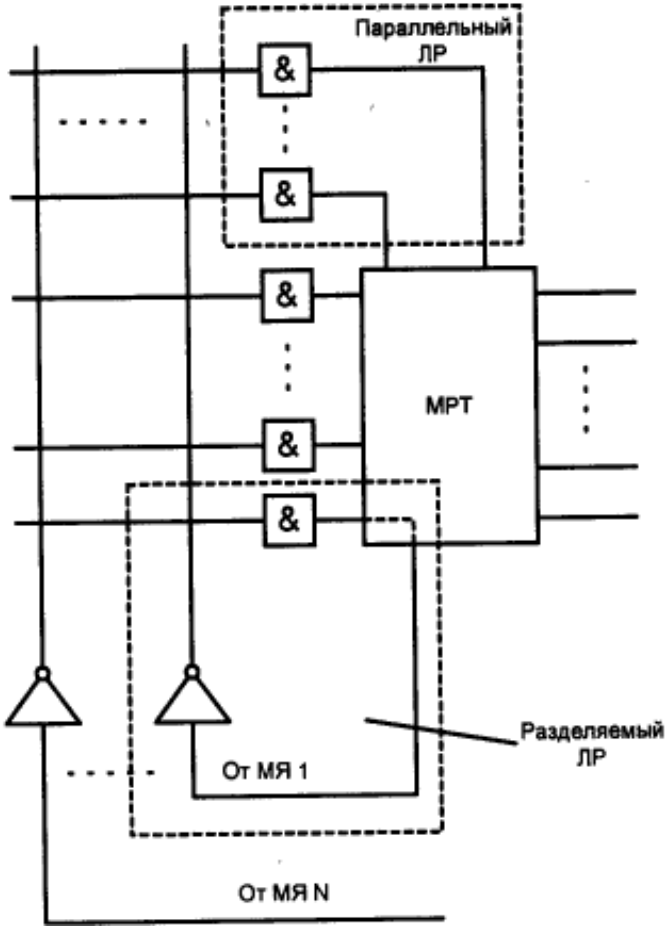
Функциональные блоки CPLD

Основными частями ФБ CPLD являются программируемая **матрица элементов И ($М_и$), матрица распределения термов (МРТ)** и группа из нескольких **N макроячеек**. Структура ФБ подобна ПЛМ. Многовходовая $М_и$ вырабатывает конъюнктивные термы для МРТ.

Простейшая схема выработки ДНФ для i-го канала. В функцию может входить до 5 термов.

Структура ФБ

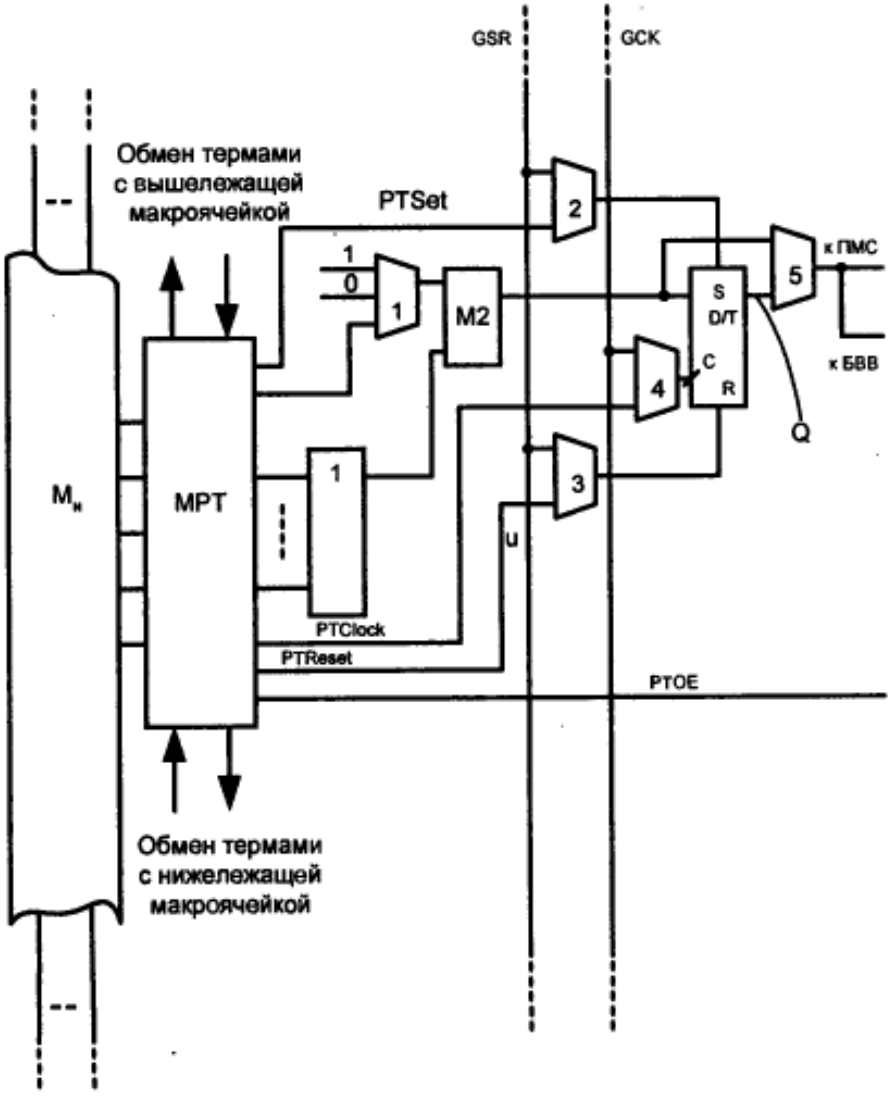




Последовательные (разделяемые) логические расширители (ЛР) создаются подачей инвертированного значения терма из МРТ данного канала на один из входов $М_и$.

Параллельный ЛР позволяет передавать термы от одного канала другому. Это даёт возможность организовывать цепочки для сбора термов от нескольких каналов в пределах одного ФБ. Термы от МРТ поступают в **макроячейку (МЯ),** содержащую программируемые мультиплексоры, триггеры и формирует группу выходных сигналов ФБ в нескольких вариантах.

Макроячейка ФБ CPLD



Пример схемы макроячейки функционального блока CPLD

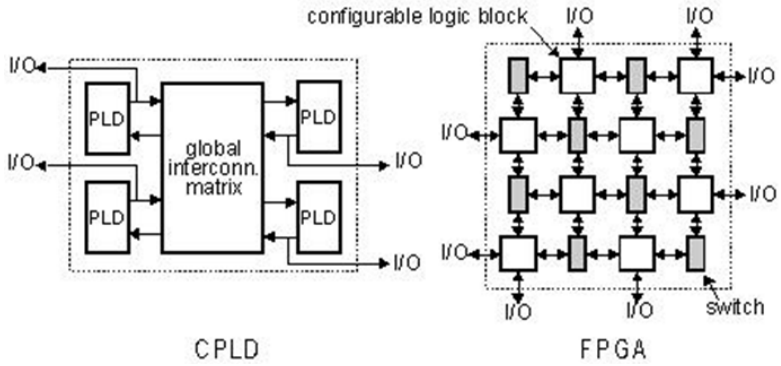
Каждый MUX передает на вых. сигнал с того или иного вх. Триггер может программироваться на D или T. Триггеры тактируются положительными фронтами синхросигналов и имеют вх. S и R. Вых. сигналы ФБ передаются в ПМС и в БВВ.

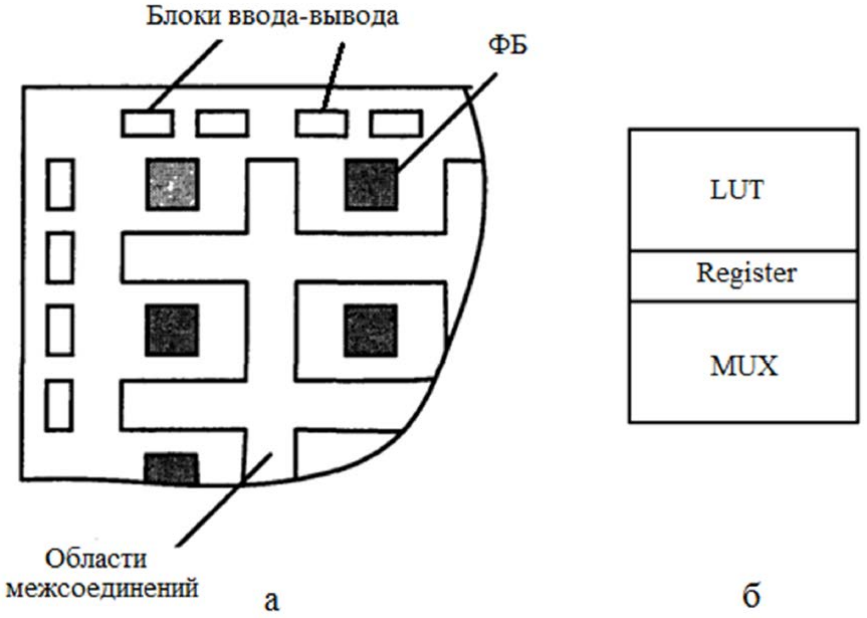
Аргументы x1, ..., xM реализуемой МЯ функции поступают на матрицу $M_и$ из ПМС. Аргументами для МЯ могут быть как вх. сигналы, от БВВ, так и сигналы обратных связей, подаваемые в матрицу И с вых. МЯ.

Вх. буферы преобразуют сигналы в парафазные так что в матрице имеется 2m вертикальных линий и образующие ее конъюнкторы имеют по m входов. 5 термов из матрицы И поступают на элемент ИЛИ для образования лог. функции. Для управления триггером и буферами БВВ вырабатываются также термы PTSet, PTClock, PTReset, которые могут быть использованы как сигналы установки, синхронизации и сброса триггера. Терм PTOE — программируемый терм управления третьим состоянием буфера БВВ (OE, Output Enable). Всего в матрице И программируются 9N термов. На вых. ИЛИ вырабатывается лог. функция в форме ДНФ ранга не более m. Ее значение передается дальше через элемент сложения по модулю 2, на второй вход которого, в зависимости от программирования MUX1, может быть подан лог.0, лог.1 или терм PT1. В первом случае функция передается без изменений (F= F*), во втором инвертируется (F= 6 !F*), в третьем передается в прямом виде во всех ситуациях за исключением такой, в которой PT1 = 1. MUX5 программируется для передачи на вых. МЯ либо функции F (комб. Вых.), либо состояния триг гера (регистровый вых.). Характер тактирования триггера определяется программированием MUX4, при этом возможно использование GCK или сигнала терма PTClock. Асинхронные установка и сброс триггера производятся либо глобальным сигналом GSR, либо термами PTSet и PTReset, что определяется программированием MUX2 и MUX3. Сам триггер программируется на задержку (D) или счетный (T). Основной вых. сигнал МЯ поступает как в ПМС, которая может направлять его по любому требуемому маршруту, так и в БВВ.

Программируемые пользователем вентильные матрицы FPGA

FPGA (Field Programmable Gate Array) - микросхема высокого уровня интеграции, содержащая во внутренней области матрицу идентичных ФБ и систему их межсоединений, размещенную между строками и столбцами матрицы, а в периферийной области —БВВ. Кроме этого в FPGA могут быть автоподстройки задержек в системе тактирования (PLL, Phase Locked Loop или DLL, Delay Locked Loop), JTAG и др.

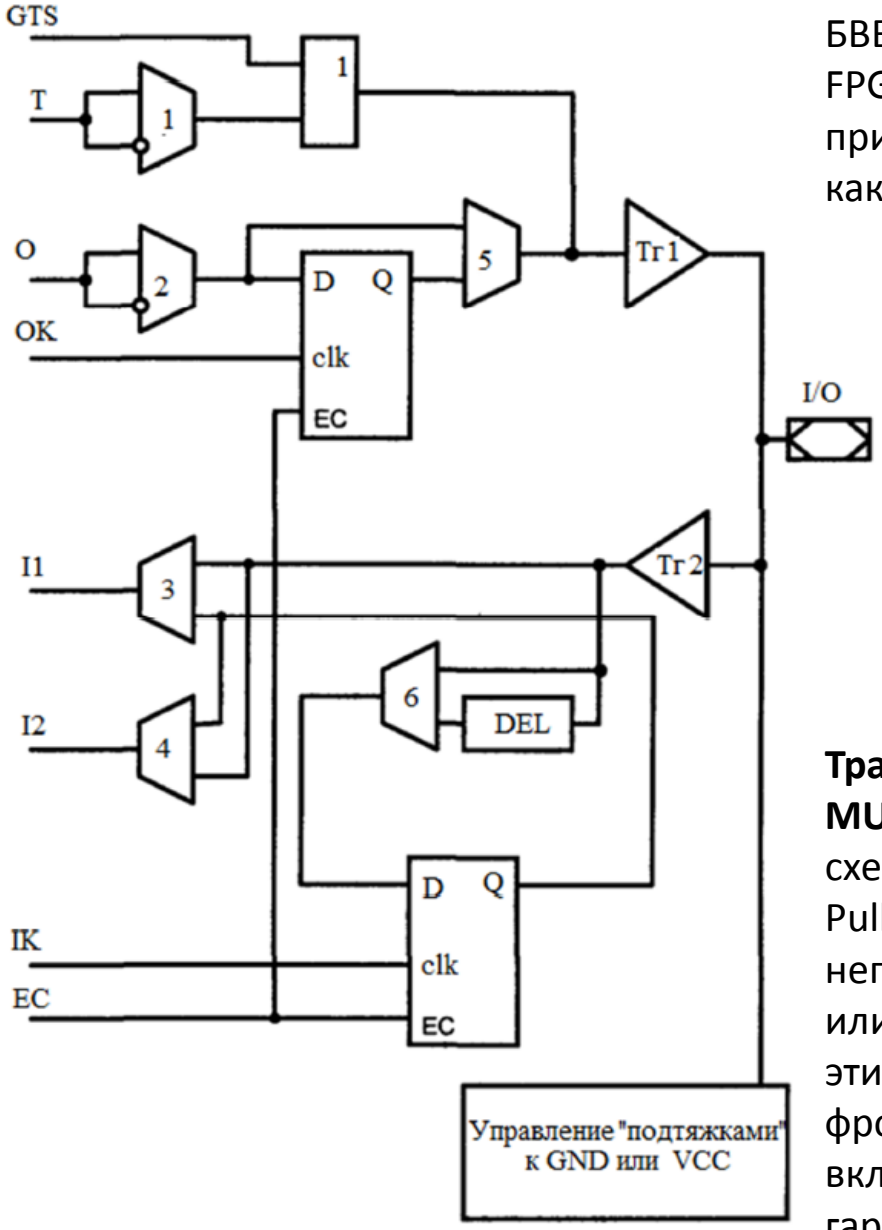




Все части FPGA конфигурируемые или реконфигурируемые средствами пользователей. На кристалле FPGA расположено множество КЛБ, каждый из которых меньше ПМЛ CPLD. КЛБ распределены по кристаллу среди программируемых соединений, а вся матрица окружена программируемыми БВВ. При конфигурировании FPGA ФБ настраиваются на выполнение необходимых операций преобразования данных, а система межсоединений – на требуемые связи между ФБ. В результате во внутренней области FPGA реализуется схема нужной конфигурации. Расположенные по краям БВВ можно программировать на выполнение требований множества стандартов (до 20).

Состав ФБ: функциональный преобразователь (ФП), реализованный в виде ПЗУ (LUT, Look-Up Table), триггер (регистр) и мультиплексоры для конфигурирования ФБ. LUT — наиболее распространенная разновидность ФП в FPGA со статической памятью конфигурации. В FPGA с однократным программированием перемычек находят применение ФП в виде простых логических вентилей (SLC, Simple Logic Cell) и логических модулей на основе MUX.

Блоки ввода/вывода FPGA



ББВ обеспечивают интерфейс между выводами корпуса FPGA и ее внутренними лог. схемами. Каждому выводу придается ББВ, который может быть конфигурирован как вход, выход или двунаправленный.

Тракт вывода: : вых. буф. 1, триггер 1, MUX 1, 2, 5 и «ИЛИ». Сигнал O можно получать в прямой или инв. форме в зависимости от MUX 2. **O** может передаваться на вых. буфер непосредственно или сниматься с триггера по MUX 5. T и GTS, согласно «ИЛИ», управляют переводом буфера в третье состояние, причем активный уровень T программируется MUX1. Буфер имеет программируемые крутизну фронта и уровни (КМОП/ТТЛ).

Тракт ввода: вх. буф. 2, триггер 2, программируемые **MUX** 3, 4, 6, элемент задержки DEL и программируемые схемы задания потенциалов выводу (Pull-Up (Vcc) или Pull-Down (GND)). **I** в зависимости от MUX 3 и 4 поступает непосредственно в систему коммутации FPGA по I1 и I2, или же фиксируется триггером и с его вых. передается в эти линии. Триггеры могут конфигурироваться как по фронту или как защелки D. На триггер 2 могут быть включены DEL (нижний вход MUX6). Включение DEL гарантирует необходимые временные соотношения между входными сигналами триггера D и GCK clk. Вх. буф. может конфигурироваться ТТЛ (1,2 В) или КМОП (0,5 Vcc).

Межсоединения FPGA

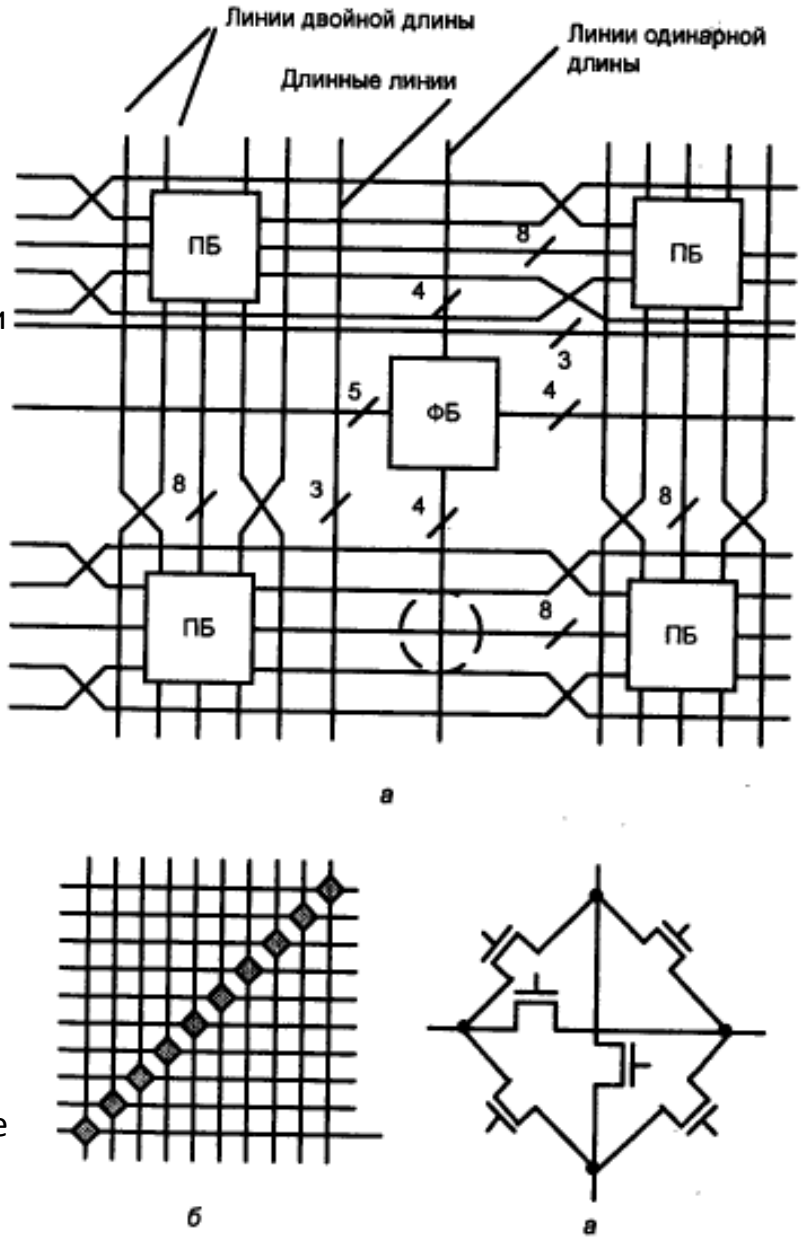
Система межсоединений имеет иерархический характер:

- основные связи;
- связи двойной длины огибают ПБ, соседние по отношению к данному;
- прямые связи для близлежащих ФБ;
- длинные линии, пересекающие кристалл по всей его длине или ширине.

Систему межсоединений FPGA образуют сегментированные линии и переключательные блоки ПБ (Programmable Switching Matrix) на пересечениях вертикальных и горизонтальных линий. ФБ имеют квадратные геометрические очертания, их выводы распределены по всем сторонам квадрата для облегчения коммутируемости. Для межсоединений ФБ во внутренней области кристалла имеются 3 типа связей: одинарной длины, двойной длины и длинные линии. В каждом пересечении ПБ цепь из 6 транзисторов. Сигнал, поступающий в ПБ по какой -либо линии (например, горизонтальной), может быть направлен вверх, вниз или прямо в зависимости от того, какой транзистор будет открыт при конфигурировании FPGA. Возможно разветвление.

Линии двойной длины сгруппированы в пары, имеется по 4 вертикальных и горизонтальных линии, обеспечивающих более быструю и эффективную передачу сигналов на средние расстояния. Длинные линии, рассчитанные на передачу сигналов на большие расстояния и при большой нагрузке, имеют в середине ключ, разделяющий линию на две части.

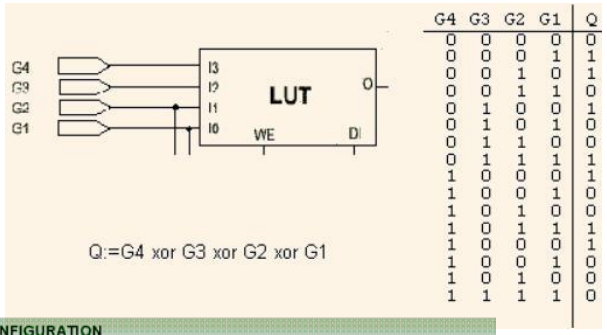
FPGA может иметь трассировочные ресурсы, расположенные в виде кольца вне пределов матрицы ФБ, позволяющие изменять назначение вводов/выводов микросхемы, без влияния на разводку печатных плат.



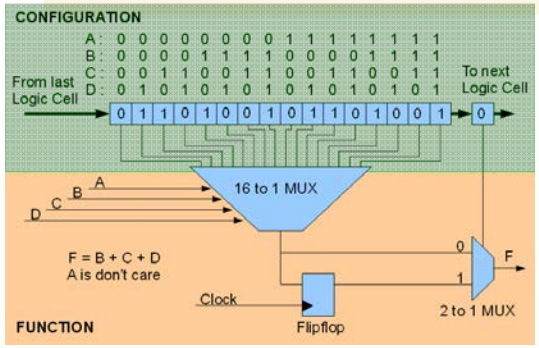
Пример системы коммутации FPGA (*а*), схема переключательного блока (*б*) и схема для установления соединений коммутируемых линий (*в*)

Функциональные блоки FPGA

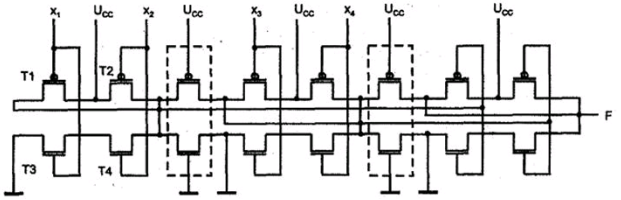
1.Табличный ФП типа LUT представляет собой ЗУ, хранящее значения искомых функций, считываемые по адресу-аргументу. ЗУ с организацией 2^m *n имеет *m* адресных входов и n выходных линий, может хранить таблицу для считывания я функций от *m* переменных. Время вычисления результата равно времени считывания слова из памяти.



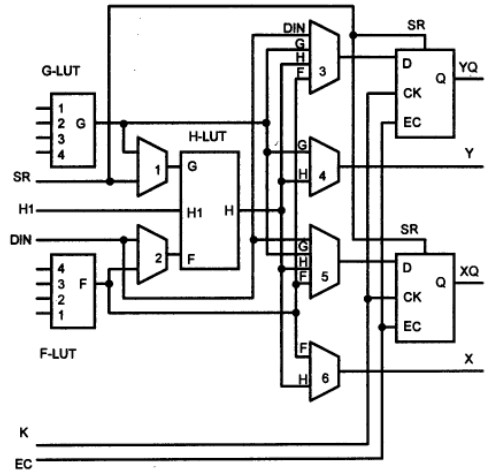
2.ФБ — *программируемые MUX* (фирма Actel) Вых. величина описывается некоторой так называемой порождающей функцией, соответствующей использованию всех входов схемы как информационных. При программировании на некоторые входы задаются константы 0 и 1, разные сочетания которых порождают целый.



3. *Мелкозернистые* ФБ, составленные из транзисторных пар, выделяемых из цепочек транзисторов с n - и p -каналами. Из таких пар собираются традиционные для КМОП - схем логические элементы.

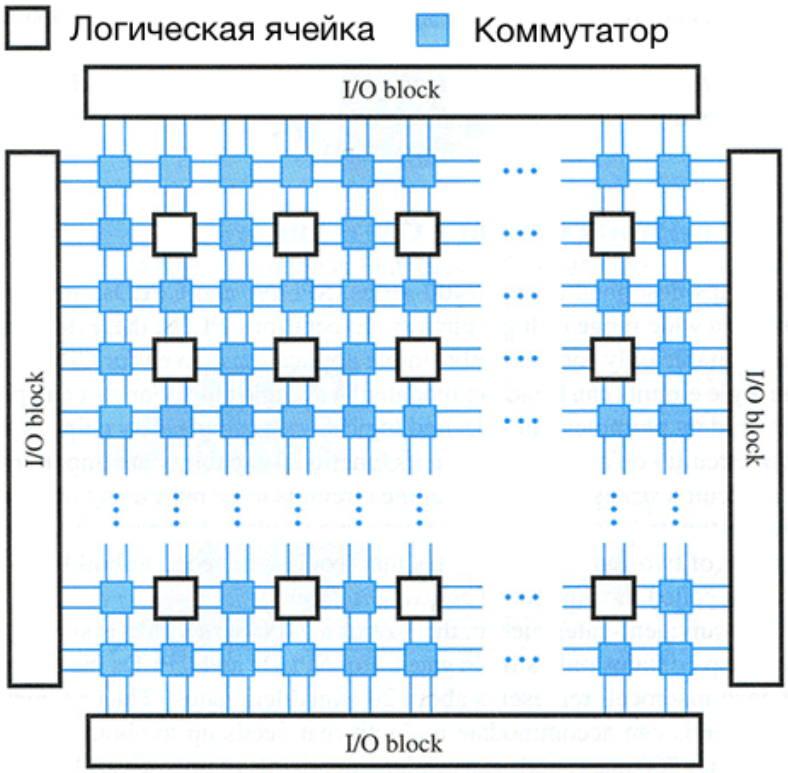


4. ФБ FPGA с триггерной памятью-конфигурации на примере микросхемы семейства Spartan фирмы Xilinx. В ФБ логические преобразования выполняются тремя LUT-блоками (функциональными преобразователями ФП) G, F и H. Преобразователи G и F — программируемые ЗУ 16x1, способные воспроизводить любые функции 4-х переменных, значения которых могут быть переданы на выходы Y и X через MUX 4 и 6 при соответствующем их программировании (через линии верхних вх. MUX).

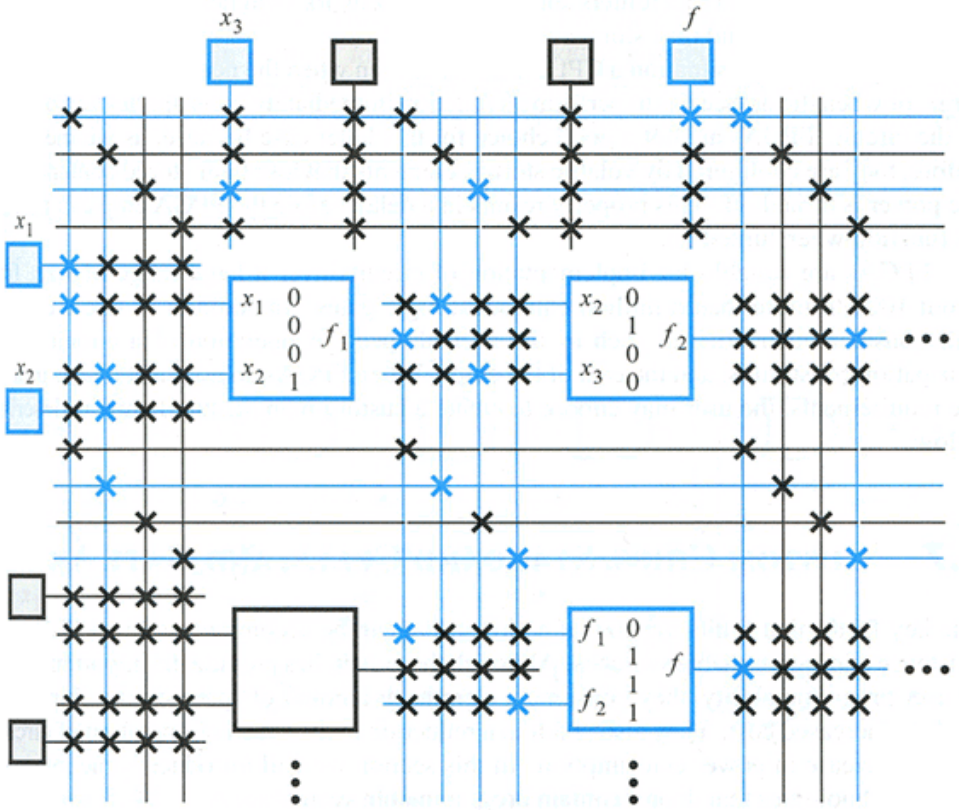


Функциональные блоки FPGA

FPGA (Field-Programmable Gate Array) — и программируемая вентильная матрица, и программируемая логическая интегральная схема (ПЛИС). ПЛИС не является процессором, которому нужно прописывать набор определенных инструкций. Это поле так называемых «логических вентилей», где можно строить свою электронную схему собственно «процессора».



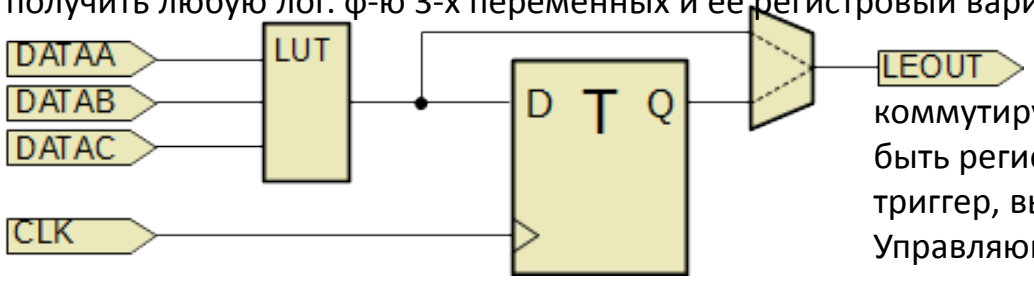
Общая схема FPGA



Секция программируемой FPGA

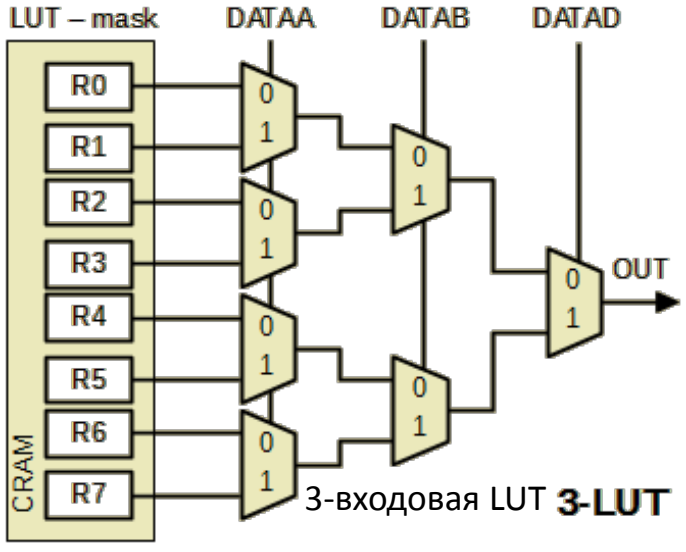
Функциональные блоки FPGA. Базовые

Простейший ЛЭ ПЛИС - соединение программируемого комбинационного устройства и D-триггера. Позволяет получить любую лог. ф-ю 3-х переменных и ее регистровый вариант.

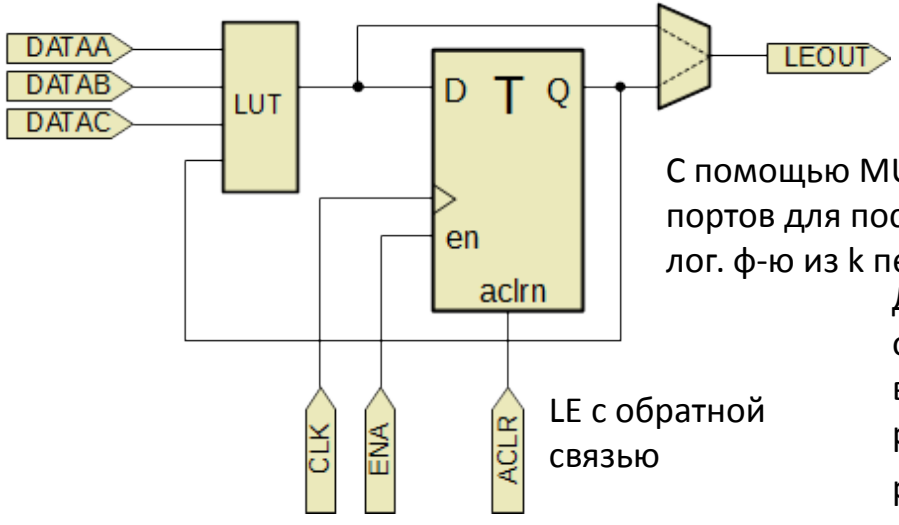


Если от ЛЭ только комб. устройство, то вых. MUX коммутирует вых. элемента LUT на LEOUT, если вых. должен быть регистровым, то сигнал с LUT защелкивается по CLK в D-триггер, вых. которого через MUX соединяется с LEOUT. Управляющий вход мультиплексора (на рисунке не показан) подключен к соответствующему биту конфигурационной памяти CRAM.

LUT (Look-Up Table "справочная таблица") – это метод реализации функции, в котором непосредственное вычисление заменяется поиском по таблице готовых решений. Позволяет реализовать любую лог. ф-ю в виде памяти SRAM, где адрес – это аргумент, а содержимое ячейки – значение. Чтобы описать лог. ф-ю 3-х переменных достаточно 8 ячеек. Требуемая таблица истинности хранится в виде маски (LUT-mask) в соответств. ячейке CRAM.



3-входовая LUT **3-LUT**

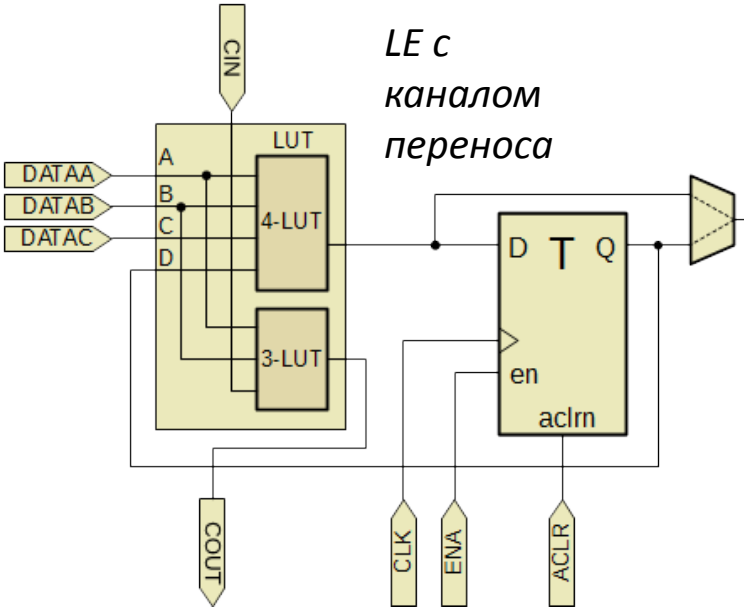


LE с обратной связью

С помощью MUX выбирается нужное значение. MUX управляют сигналы вх. портов для построения k-входовой LUT (k-LUT), которая реализует любую лог. ф-ю из k переменных, требуется 2^k бит SRAM и 2^{k-1} MUX.

Для реализации различных триггеров необходимо ввести обратную связь: с вых. Q подадим сигнал на один из вх. 4-х входового LUT. Добавим доп. линии управления: вход разрешения ENA и вход асинхронного сброса ACLR. В результате схема пригодную для синтеза любых триггеров

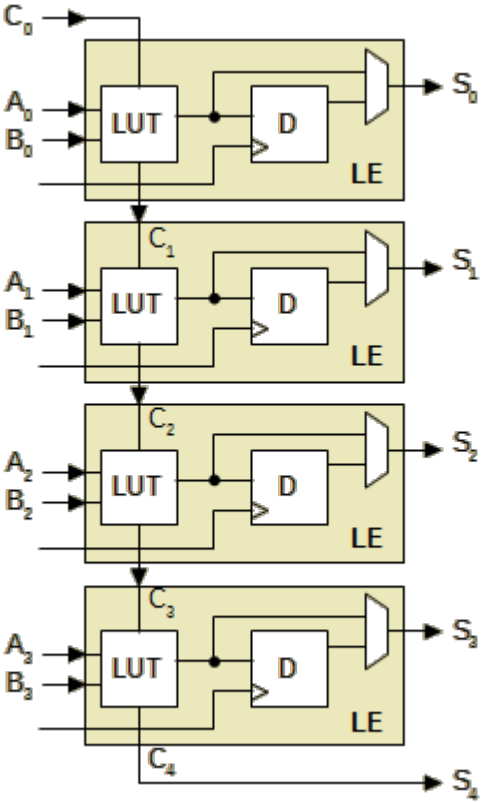
Функциональные блоки FPGA. Усовершенствованные



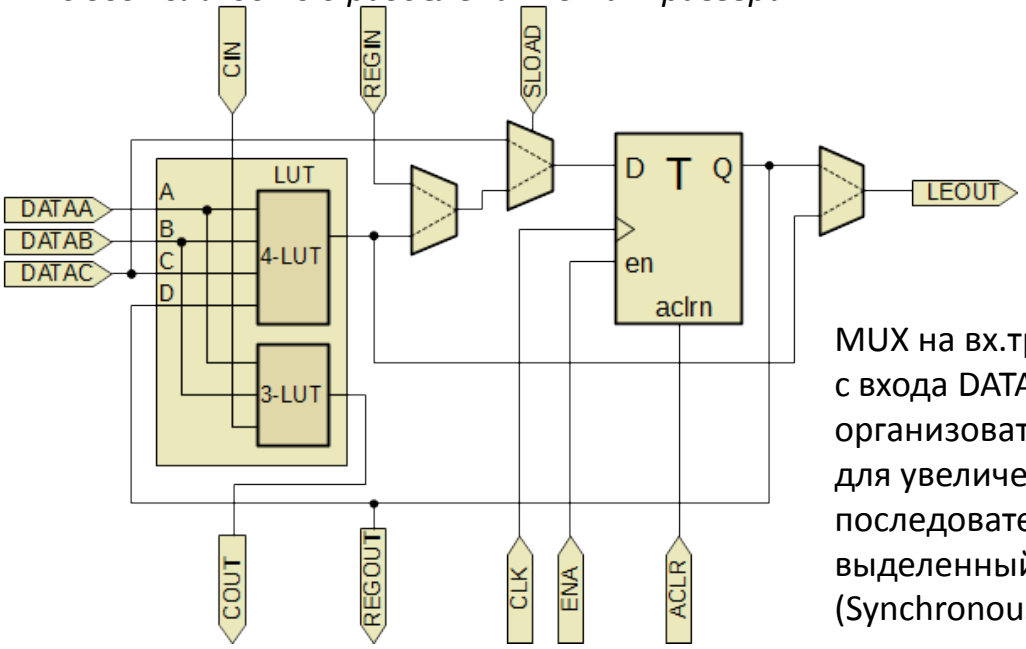
LE с каналом переноса

Для сумматоров требуется 2 вых. На вх. полного 2-ичного сумматора поступает два операнда и флаг переноса из младшего разряда, а на выходе сумма и перенос в следующий двоичный разряд. Т.к. арифметические задачи типичны для ПЛИС, для работы с переносом в базовом ЛЭ предусмотрен специальный канал. Используя канал переноса можно легко объединить ячейки для получения многоразрядного сумматора.

4-хразрядный сумматор с последовательным переносом, построенный на 4 базовых LE



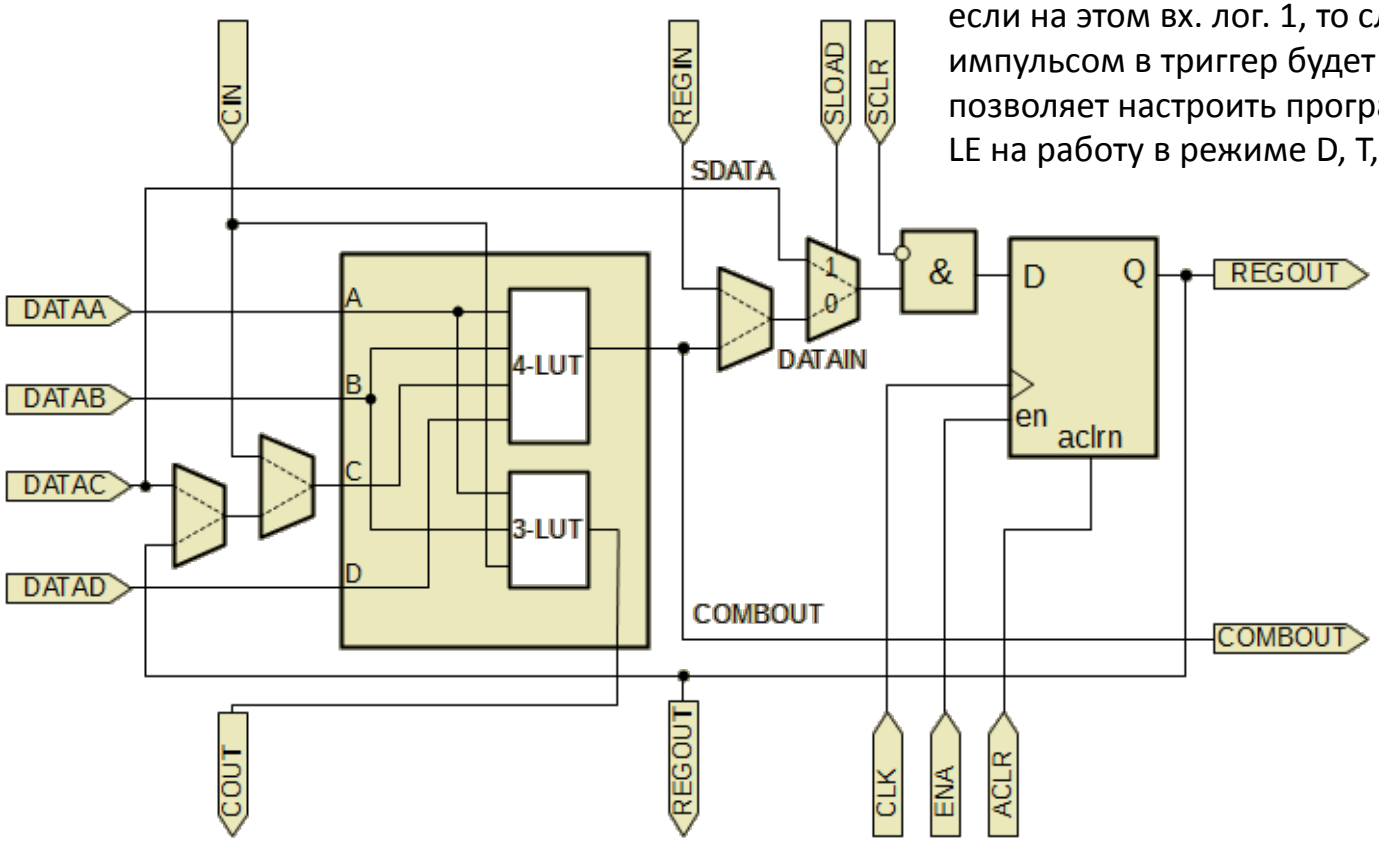
LE с возможностью разделения LUT и триггера



MUX на вх.триггера позволит выбирать источник сигнала: либо с входа DATAC, либо с выхода LUT. Появляется возможность организовать доп. канал соединения триггеров соседних LE для увеличения быстродействия при построении последовательных регистров. REGIN и REGOUT образуют выделенный канал для соединения триггеров, вход SLOAD (Synchronous Loading)

Функциональные блоки FPGA. Altera Cyclone IV

ЛЭ Altera Cyclone IV



Как в базовом каждый триггер имеет вх. данных, CLK, ENA и ACLR. Добавился сигнал синхронной очистки SCLR: если на этом вх. лог. 1, то следующим тактовым импульсом в триггер будет записал лог. 0. Все это позволяет настроить программируемый триггер каждой LE на работу в режиме D, T, JK или RS триггера.

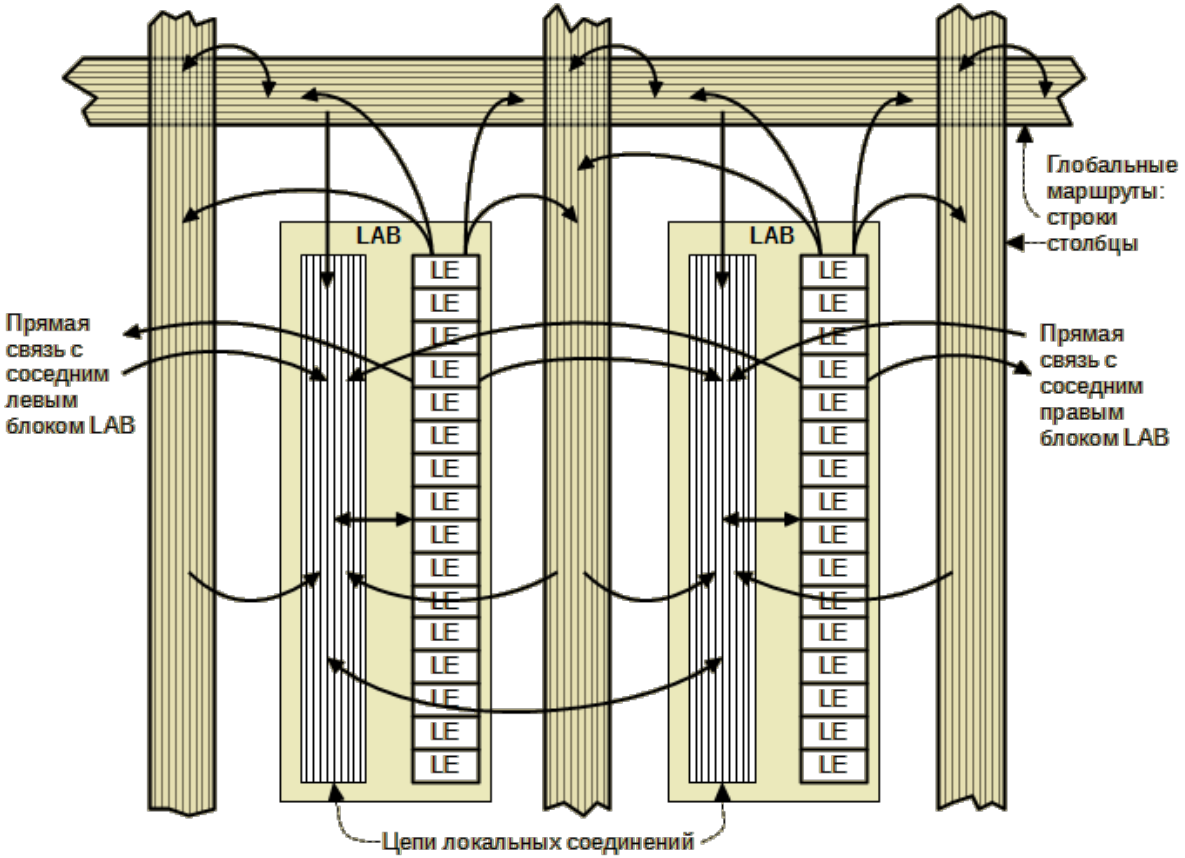
Комбинационное устройство усложнено. На входе С LUT мультиплексор выбирает источник, благодаря чему LUT может реализовывать лог. ф-ю 4-х переменных, в качестве переменной может быть использован флаг переноса или вых. собственного триггера.

2 режима Cyclone IV LUT: нормальный и арифметический. Quartus II при компиляции автоматически выберет оптимальный режим для реализации требуемой ф-ции. **Нормальный режим** для реализации основной логики и различных комб. ф-й. DATAA, DATAB, DATAC, DATAD поступают на 4 входа LUT. Компилятор автоматически выбирает вход переноса CIN, вход DATAC или вых.триггера (цепь обратной связи) в качестве одного из входов LUT. **Арифметический режим** для синтеза сумматоров, счетчиков, аккумуляторов и компараторов (цепей сравнения). LUT представляет собой полный одноразрядный сумматор, включающий обработчик логики флага переполнения. Компилятор сам создает цепи переноса во время синтеза многоразрядных арифметических устройств.

Функциональные блоки FPGA. LAB

ЛЭ LE в Cyclone IV объединяются в ЛБ **LAB** (Logic array blocks). LAB содержит: 16 LE; сигналы управления LAB; цепи флага переноса LE; цепи каскадного объединения регистров (вых. регистра одного LE с входами регистров прилегающих ячеек LE); цепи локальных соединений между LE в одном LAB.

Структура соединений LAB в коммутационном поле ПЛИС

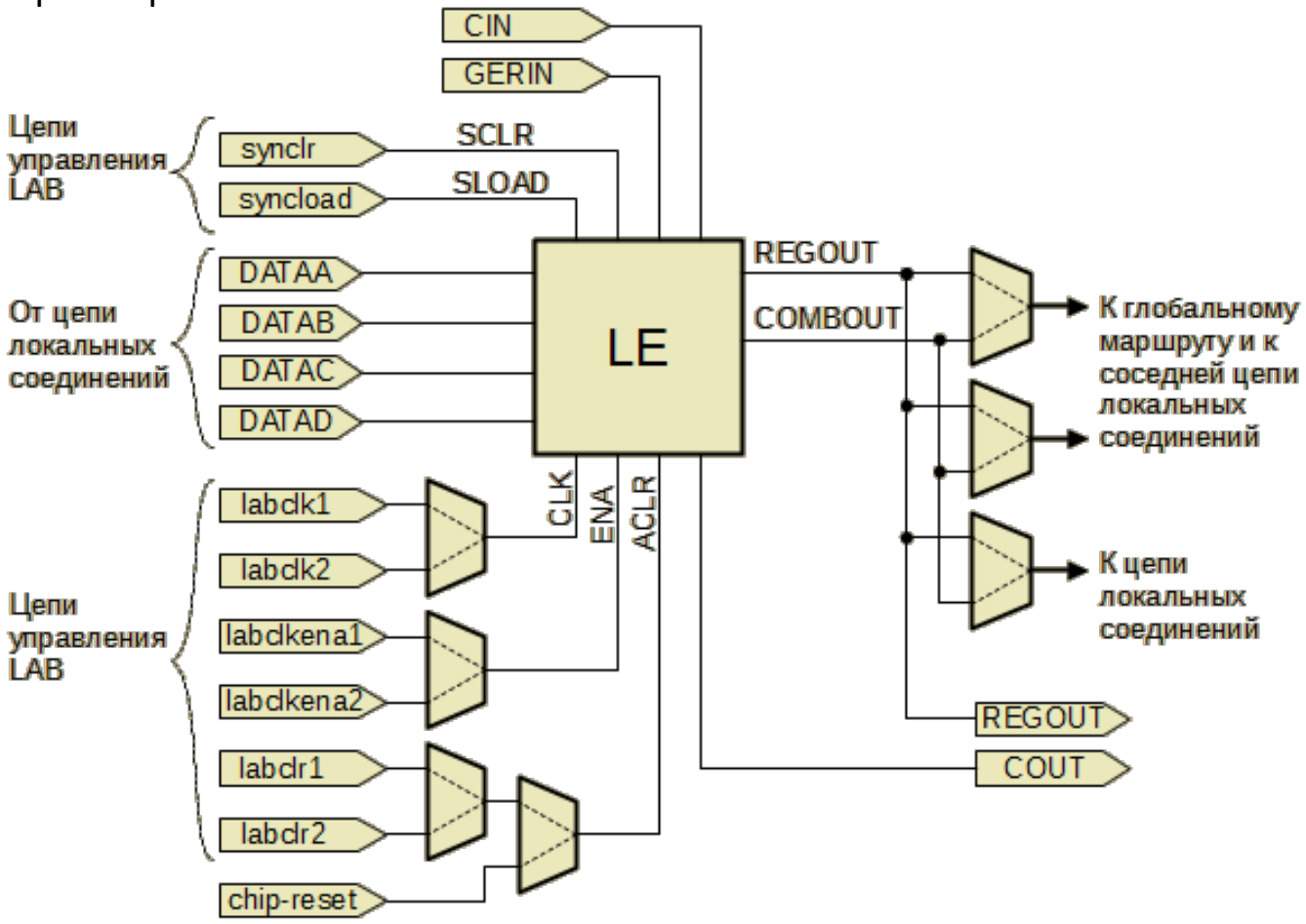


Компилятор размещает связанную логику в одном или соседних LAB, используя локальные цепи связи и связи регистров. На цепи локальных соединений поступают сигналы со строк и столбцов глобального коммуникационного поля и с выходов ячеек LE, принадлежащих этому же блоку LAB. Соседние LE, генератор с фазовой автоподстройкой частоты (PLL), ячейки памяти M9K RAM, встроенные умножители, расположенные с правой или левой стороны через специальные соединители, могут быть напрямую подключены к цепям локальных соединений LAB.

LE имеет соединение с 16-ю LE своего блока (включая саму себя) и 32-мя LE из LAB слева или справа. Всего до 48-ми соединений. Такие соединения минимизируют использование глобальных маршрутов, обеспечивают большую гибкость при синтезе схемы и увеличивают общее быстродействие.

Функциональные блоки FPGA. LE в структуре LAB

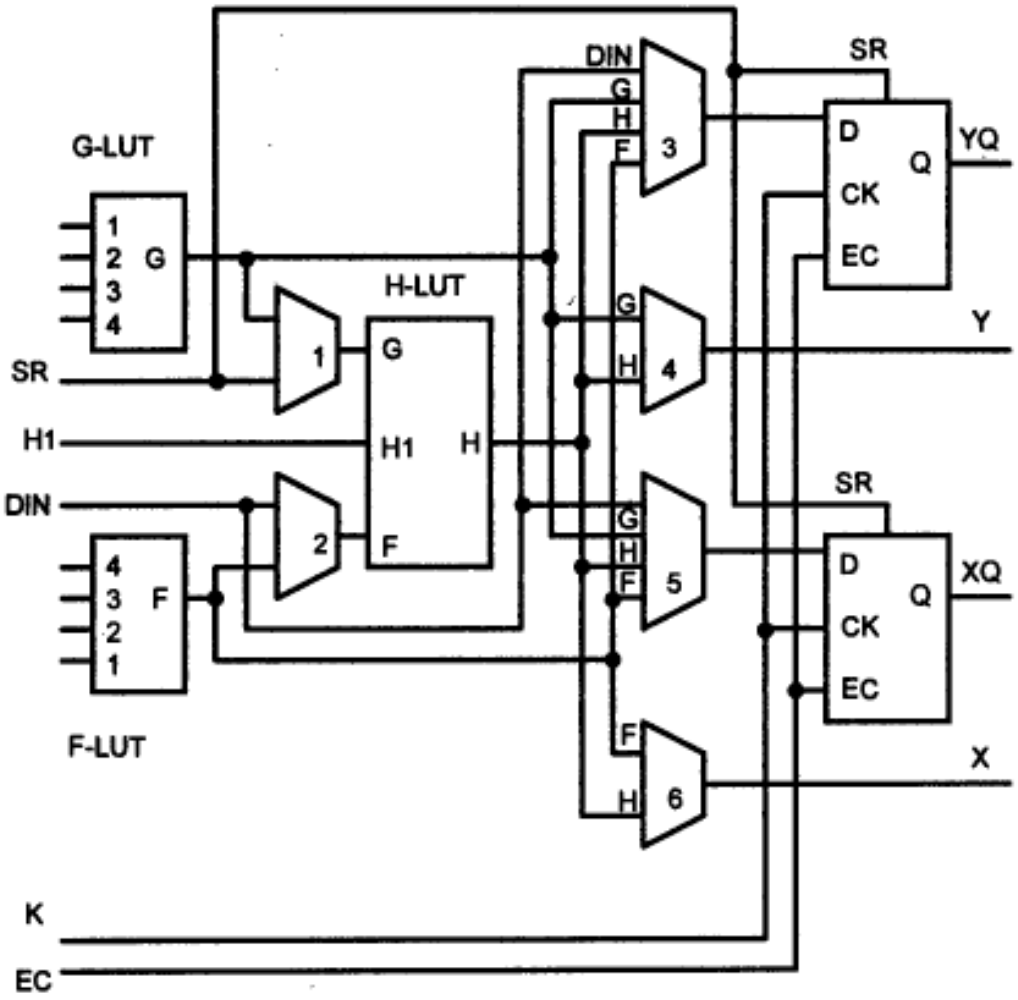
LE имеет 3 выхода: строки и столбцы глобальных соединительных трасс и на маршруты локальных соединений. Чтобы управлять LE в пределах одного LAB одновременно, в него встроена специальная логика и выделены особые линии – каналы управления. По таким каналам распространяются широковещательные (в пределах одного LAB) сигналы управления. Архитектура позволяет одновременно использовать до восьми управляющих сигналов: 2 тактовых (labclk1 и labclk2); 2 разрешения (labclkena1 и labclkena2); 2 асинхронного сброса (labclr1 и labclr2); синхронного сброса (synclr); синхронной загрузки (syncload). Synclr и syncload удобно использовать для синтеза счетчиков и регистров.



На уровне LE выбирается какой сигнал (labclk1+labckena1 или labclk2+labckena2) будет подан на триггер.
LAB может использовать оба фронта тактового сигнала, при этом логика выбора усложнится.

ФБ FPGA с триггерной памятью конфигурации

В ФБ микросхемы семейства Spartan фирмы Xilinx логические преобразования выполняются тремя LUT-блоками G, F и H. Преобразователи G и F — программируемые ЗУ 16x1, способные воспроизводить любые ф-ции 4-х переменных, значения которых могут быть переданы на вых. Y и X через MUX 4 и 6 при соответствующем их программировании (через линии верхних вх. MUX).



Через верх. вх. MUX1 и нижн. вх. MUX2 ф-ции G и F могут быть поданы на H-LUT (ЗУ 8x1) для образования H =f(G , F, H1) от более чем 4-х аргументов.

Аргументами для ФП-H, поступающими от MUX 1 и 2, в зависимости от их программирования может быть также G, HI , DIN; SR, HI , DIN; SR, HI , F. Линии DIN и SR используются либо для передачи в триггер входных данных и сигнала Set/Reset, либо как входы ФП-H.

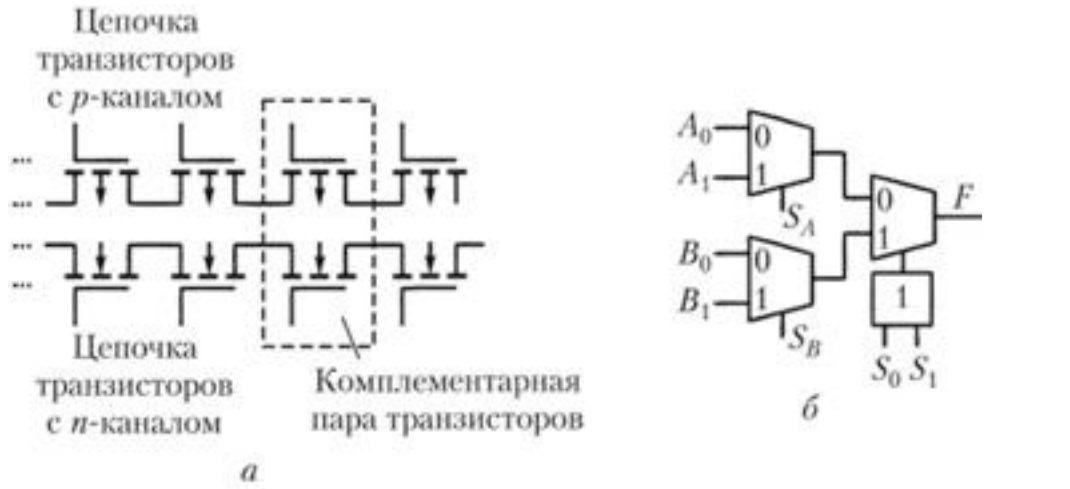
Такие ресурсы позволяют воспроизводить:

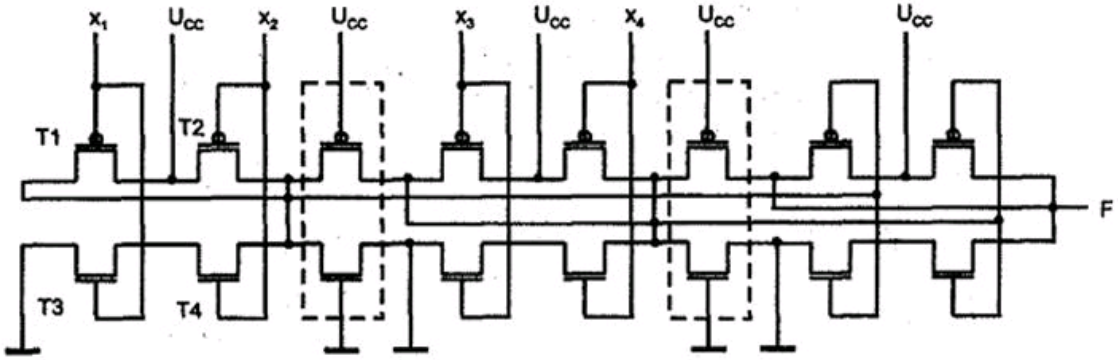
- любую функцию с числом аргументов до 4+ вторую такую же функцию + любую функцию с числом аргументов до трех;
- любую функцию 5 аргументов (одну);
- любую функцию 4 аргументов и одновременно некоторые функции 6 аргументов, некоторые функции с числом аргументов до 9.

MUX 3...6 направляют сигналы данных и управления на триггеры для фиксации и хранения вых. сигналов функциональных преобразователей или работать транзитно. Вх. сигналы DIN и H1 можно передавать любому триггеру.

Мелкозернистые ФБ FPGA

Двумя основными характеристиками ЛБ являются их "***зернистость***" и ***функциональность***"**.** "Зернистость" связана с тем, насколько "мелкими" будут части, из которых можно "собирать" нужные схемы, а "функциональность" – с тем, насколько велики логические возможности ЛБ.





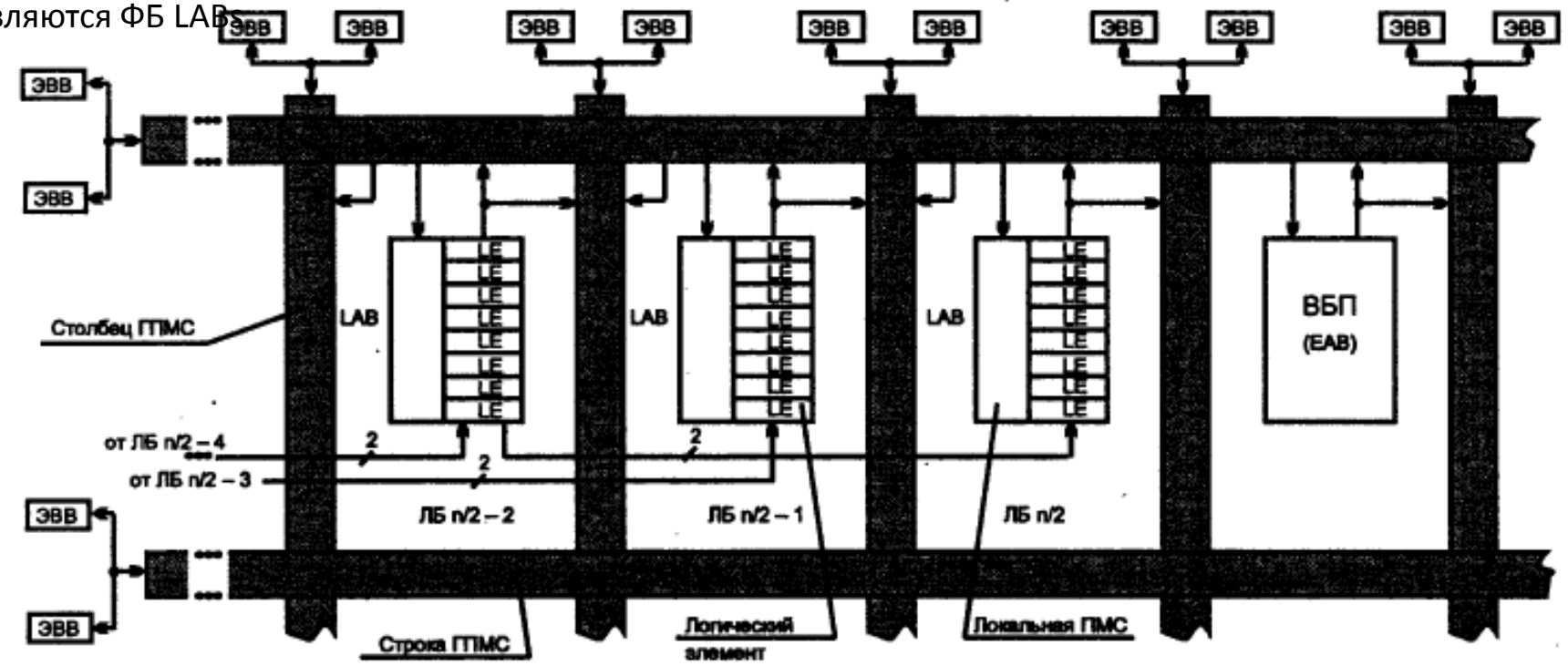
Мелкозернистый ЛБ содержит цепочки транзисторов с p- и n- каналами (а), между которыми имеются трассировочные каналы, в которых могут быть реализованы необходимые межсоединения элементов. На основе комплементарных пар можно строить "И-НЕ", "ИЛИ-НЕ" в КМОП-логике. Чтобы "разорвать" цепочки транзисторов, на затворы одной из пар подаются запирающие напряжения: напряжение питания – на верхний и корпус – на нижний. При этом транзисторы, находящиеся слева и справа от них, становятся изолированными друг от друга. Затем из полученных элементарных логических схем путем выполнения необходимых межсоединений осуществляется построение сложных логических устройств.

Мелкозернистость ЛБ ведет к большой гибкости, но усложняет систему межсоединений FPGA из-за большого числа программируемых точек связи.

Более крупный ЛБ состоит из трех MUX "2–1" и элемента "ИЛИ" (***6).*** Подключая к входам такого ЛБ переменные и константы, можно получить комбинационные ф-ции 2-х переменных, многие ф-ции 4-х переменных и некоторые ф-ции большего числа переменных, вплоть до 8.

ПЛИС с комбинированной архитектурой
Семейство FLEX

Комбинированные архитектуры сочетают черты CPLD и FPGA, имеют ФБ (LABs, Logic Array Blocks) с ЛЭ (LEs), содер-жащими ФП табличного типа (LUTs). ФБ расположены в виде матрицы, между их строками и столбцами проходят горизонтальные и вертикальные трассировочные кана-лы как в FPGA. Трассы в каналах не сегменти-рованы как CPLD. Поскольку в схемах с большим числом ФБ применение еди-ной коммутационной матрицы затруднено, система коммутации имеет два уровня межсоединений — глобальный и локальный. Локальная ПМС (ЛПМС) обеспечивает межсоединения ЛЭ, из которых составляются ФБ LABs.

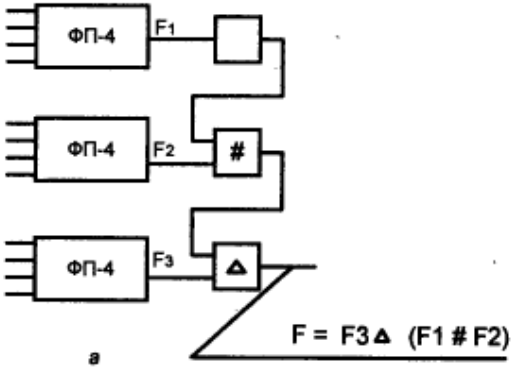


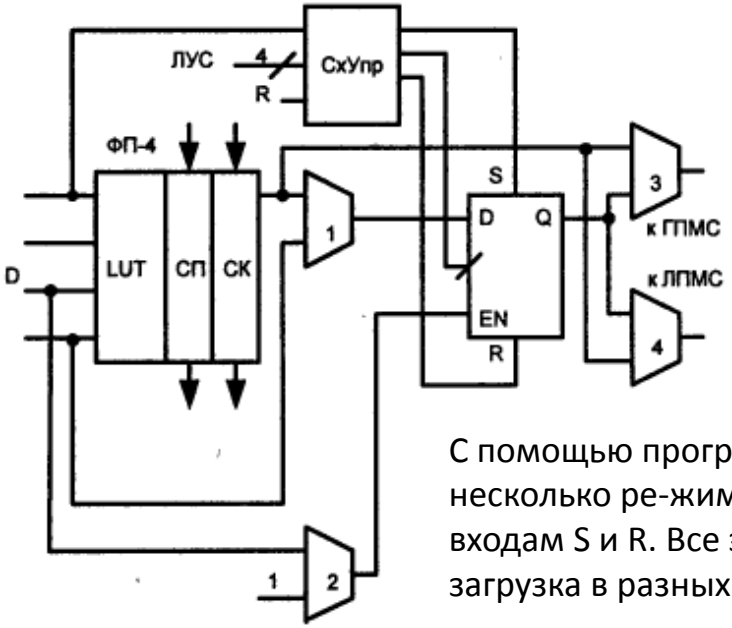
В состав LAB входят 8 ЛЭ. Со-единения между LAB обеспечиваются глобальной ПМС (ГПМС), к концам строк и столбцов которой под-ключаются БВВ (IOBs, Input/Output Blocks). Начиная с FLEX10K встроенные блоки памяти ВБП (EABs, Embedded Array Blocks), конфигурируется как ЗУ с организаци-ей 256x8, или 512x4, или 1024x2, или 2048x1 и использоваться для хранения данных и как табличный ФП для реализации сложных функ-ций с числом аргументов 8—10 (в частности, на блоках EAB строятся быстродействующие АЛУ, перемножители 4x4).

Логические элементы ПЛИС семейства FLEX

ЛЭ имеют 4-входовые ФП типа LUT, тракты переноса, образуемые цепями схем переноса СП, тракты каскадирования, образуемые схемами каскадирования (СК) с непо-средственными и быстродействующими связями между ЛЭ.

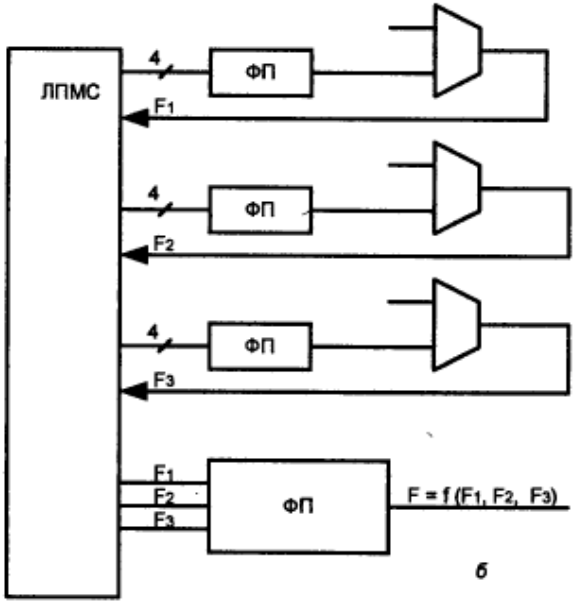
Ф-ции многих переменных можно получить исполь-зуя обратные связи. Сначала вырабатывается ф-ция 4-х переменных, затем она вводится в качестве одного из входов в другой ЛЭ и т. д. В результате вычисляется "функция от функций" с числом аргументов, превышающим 4.





Синхронный триггер получает сигналы от схемы управ-ления с R (Reset), D (одна из вх.линий ЛЭ) и ЛУС (4 линии локальных управляющих сигналов). Один из двух сигналов ЛУС по выбору используется для тактирования триггера, два дру-гих вместе с R и одним из входов D управляют режимами сбро-са/установки.

С помощью программирования можно задавать несколько ре-жимов воздействия на триггер по входам S и R. Все эти режимы (сброс, ус-тановка, загрузка в разных вариантах) асинхронны.

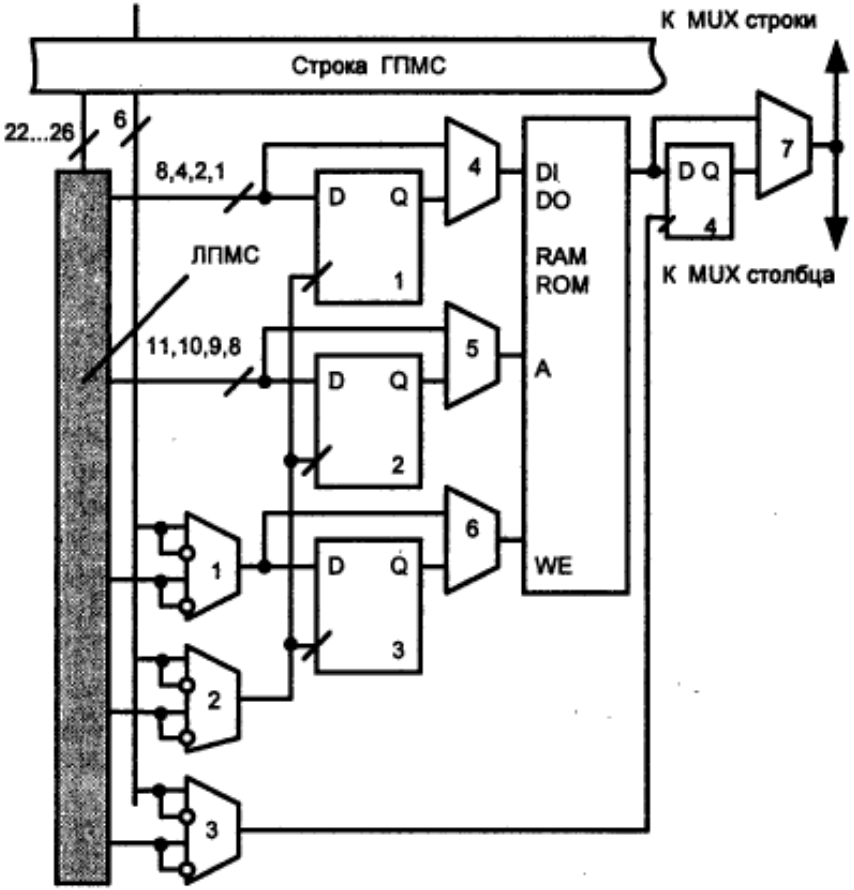


ФП 4 вх. 16 бит памяти воспроизводит ф-ции 4 ар-гументов при 16x1, или 3 аргументов в виде двух табличных ФП с организацией 8x1 (для разрядных схем сумматоров с последовательным переносом, разрядных схем некоторых счетчиков и т. д.) Длинные цепочки переносов формируются в пределах нескольких LAB. Задержка цепи переноса мала (1 нс), пригодна для быстродействующих устройств. Цепь каскадирования используется для получения ф-ций с числом аргументов более 4: 3 соседних ЛЭ для воспроизведения частичных функций, а затем с помощью СК сформировать из них окончательный результат. Частичные функ-ции смежных ЛЭ объединяются любой логической операцией над двумя пе-ременными, кроме сложения по модулю 2 и функции равнозначности.

Триггер может быть использован для фиксации значений ФП или как отдельный элемент с вхо. от одной из линий D в зави-симости от программирования MUX. Вых. сигнал ЛЭ через пр. MUX может подаваться в ГПМС или ЛПМС в программируемых вариантах комбинационного или регистрового выходов.

Блоки памяти ПЛИС семейства FLEX

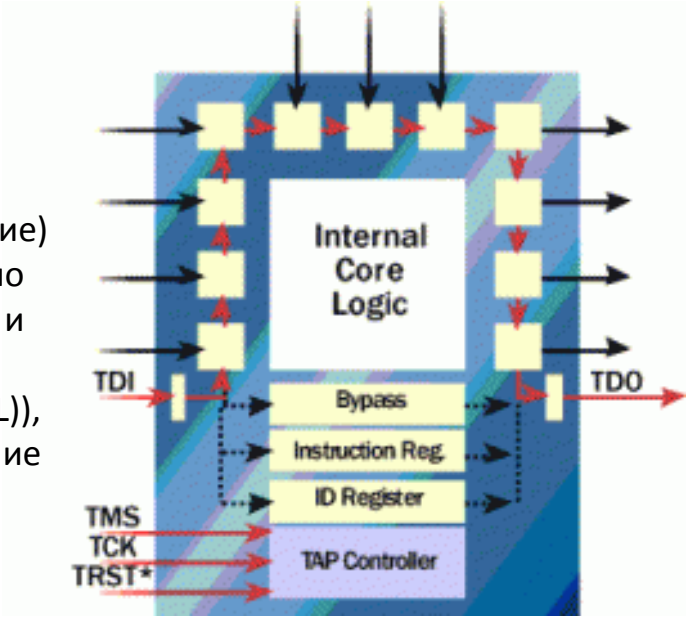
Встроенные блоки памяти ВВП общей емкостью приблизительно от 6 до 20 Кбит для разных представителей семейства. Отдельные блоки 2 Кбит размещены в середине каждой строки матрицы ЛБ. Блоки встроенной памяти можно использовать как по прямому назначению, т. е. как статическое ЗУ, так и для реализации ПЗУ и логических схем (табличных ФП повышенной размерности путем эмуляции ПЗУ с помощью загрузки таблицы в ОЗУ). Такие ФП дают более эффективные решения в сравнении с реализациями сложных функций средствами типовых ЛБ. Например, один встроенный блок памяти при организации 256x8 реализует перемножитель 4x4, способный работать на частотах до 50 МГц. Построение такого же перемножителя на типовых ЛБ потребовало бы занять 8 ЛБ, а частота работы перемножителя не превысила бы 20 МГц.
Блоки встроенной памяти ориентированы также на организацию буферов FIFO, а в микросхемах FLEX10KE и на построение двухпортовой памяти. Несколько блоков можно объединять для создания более емкой памяти. Так как блоки памяти расположены на том же кристалле, что и логическая часть схемы, работа с памятью отличается высоким быстродействием.



Сигналы управления регистрами 1 – 3 поступают от глобальной шины управ-ляющих сигналов с возможностью выбора их полярности (мультиплексоры 1 – 3). Выходные сигналы блока памяти с помощью мультиплексоров 7 – 9 могут передаваться как на линии строки, так и на линии столбца ГПМС в тактируемом или асинхронном вариантах.

Интерфейс JTAG

JTAG (*Joint Test Action Group*) — рабочая группа по разработке стандарта IEEE 1149 (**Standard Test Access Port and Boundary-Scan Architecture)** для подключения сложных цифровых микросхем или устройств уровня печатной платы к стандартной аппаратуре тестирования и отладки.

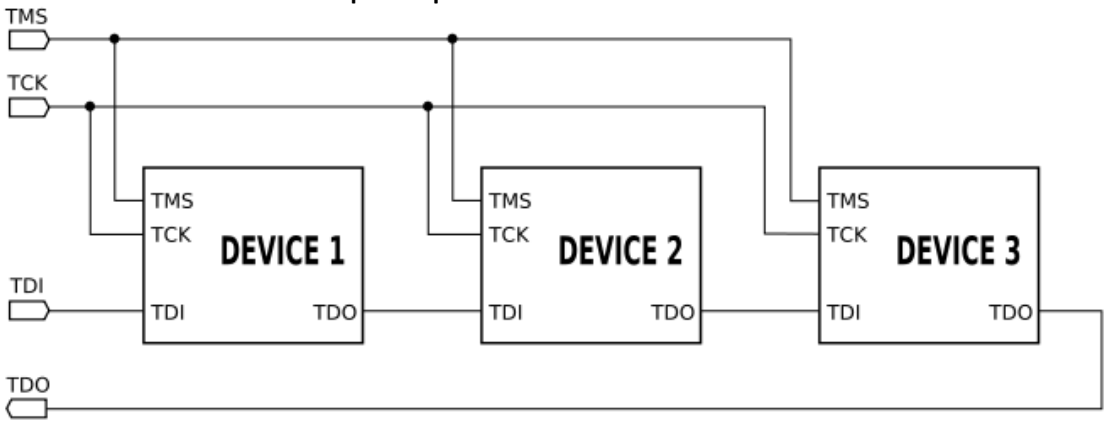


Предназначен для: выходного контроля микросхем при производстве; тестирования собранных печатных плат; прошивки микросхем с памятью; отладочных работ при проектировании аппаратуры и программного обеспечения. **Метод тестирования Boundary Scan** (граничное сканирование) - в микросхеме выделяются функциональные блоки, входы которых можно отсоединить от остальной схемы, подать заданные комбинации сигналов и оценить состояние выходов каждого блока. Весь процесс производится специальными JTAG командами (Boundary Scan Description Language (BSDL)), никакого физического вмешательства не требуется. Возможно подключение большого количества устройств (микросхем) через один физический порт (разъем).

Порт тестирования (**TAP** — Test Access Port) имеет 4 или 5 выводов
TDI (*test data input* — «вход тестовых данных») — вход последовательных данных периферийного сканирования. Команды и данные вводятся в микросхему с этого вывода по переднему фронту сигнала TCK;
TDO (*test data output* — «выход тестовых данных») — выход последовательных данных. Команды и данные выводятся из микросхемы с этого вывода по заднему фронту сигнала TCK;
TCK (*test clock* — «тестовое тактирование») — тактирует работу встроенного автомата управления периферийным сканированием. Максимальная частота сканирования периферийных ячеек зависит от используемой аппаратной части и на данный момент ограничена 25…40 МГц;
TMS (*test mode select* — «выбор режима тестирования») — обеспечивает переход схемы в/из режима тестирования и переключение между разными режимами тестирования.
В некоторых случаях к перечисленным сигналам добавляется сигнал **TRST** для инициализации порта тестирования, что необязательно, так как инициализация возможна путём подачи определённой последовательности сигналов на вход TMS. **TRST** (опционально) - сброс.

Интерфейс JTAG

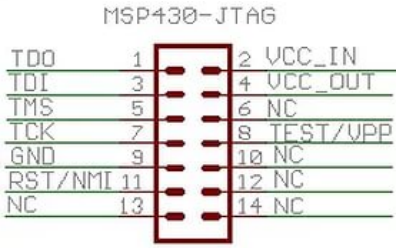
Общая цепочка JTAG

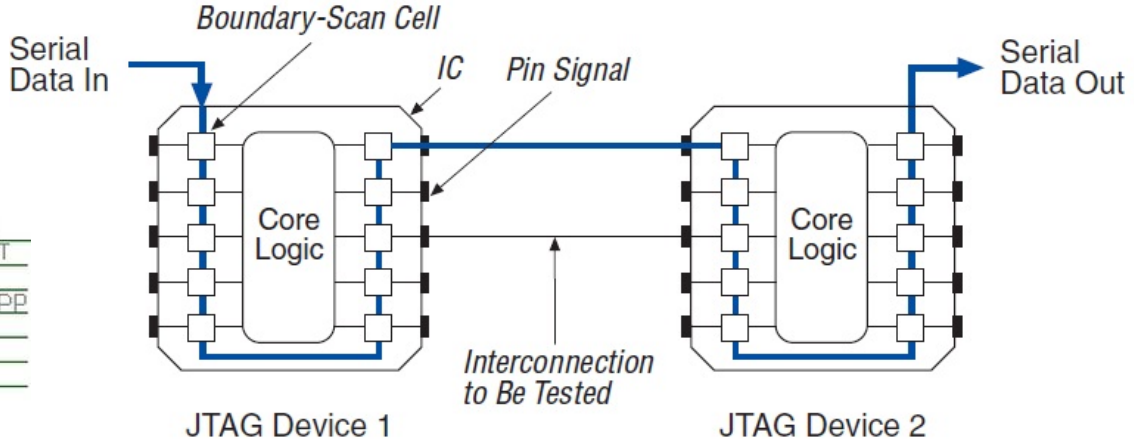


Возможность программирования микроконтроллера (или ПЛИС) и подключённой к его выводам микросхемы флэш-памяти. Два способа программирования флэш-памяти с использованием JTAG: через загрузчик с последующим обменом данными через память процессора, либо через прямое управление выводами микросхемы.

JTAG-USB переходник

Разъем

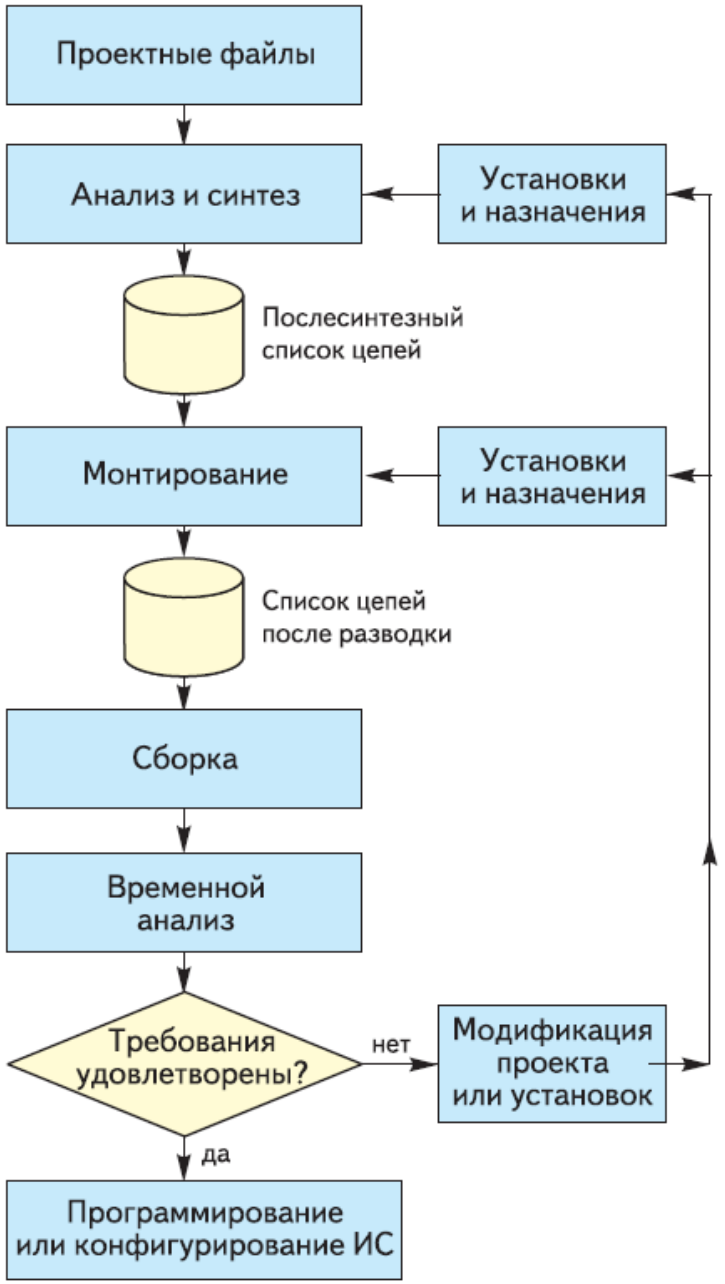




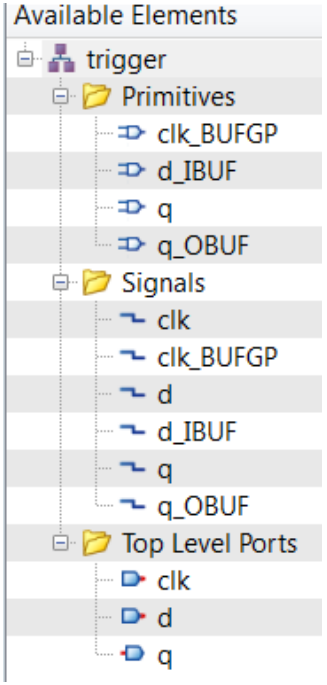
Компиляция проекта

Этапы компиляции

1. Анализ и синтез (Analysis&Synthesis). На этом этапе выполняется логический синтез файлов проекта с целью минимизации логических схем и реализации проекта с учетом имеющихся ресурсов (логических элементов, трасс, блоков ввода/вывода и т. д.). Здесь же создается база данных, интегрирующая все сведения о проекте.
2. Монтирование (Fitting). Размещение и трассировка проекта в выбранном устройстве с учетом установок (settings), заданных пользователем, а также различного рода назначений (assignments).
3. Сборка (Assembling). Создание файла конфигурации устройства на основе результатов размещения и трассировки.
4. Временной анализ (Time Analysis). Анализ временных характеристик проекта.



Компиляция проекта. Синтез

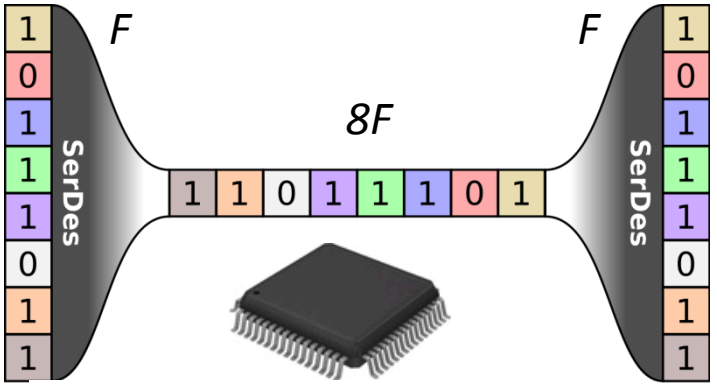


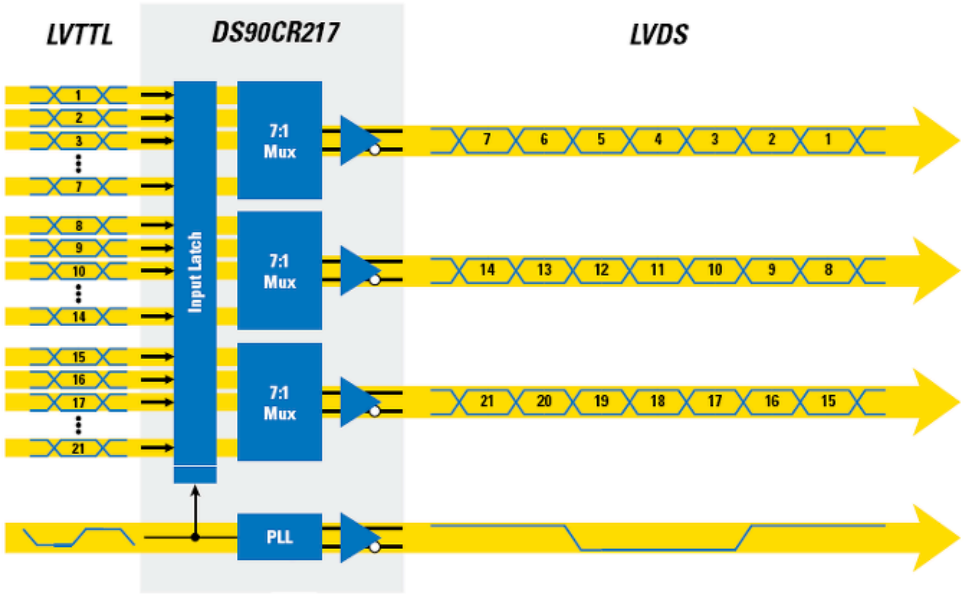
- Этап синтеза представляет собой процесс трансформации исходного HDL-описания проектируемого устройства в список цепей, выполненный на низком логическом уровне.
- Элементы низкоуровневого описания, формируемого в процессе синтеза, должны соответствовать архитектуре семейства ПЛИС, выбранного для реализации проекта.
- Синтезированный список цепей максимально адаптирован к ресурсам используемого ПЛИС.
- Результаты синтеза одного и того же проекта отличаются при использовании ПЛИС архитектур CPLD и FPGA.

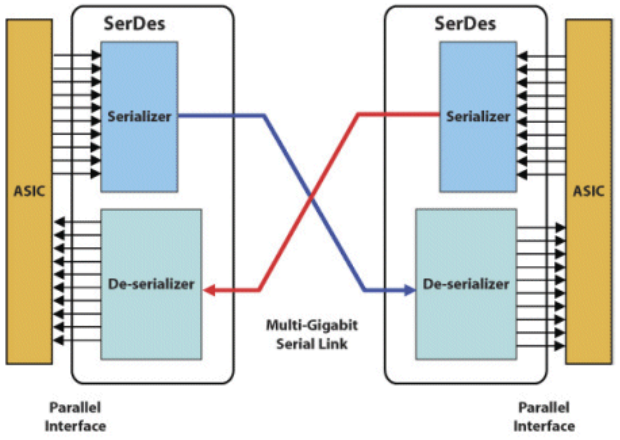
- Процесс синтеза начинается с анализа исходного HDL-описания проектируемого устройства, в ходе которого делаются попытки выделения блоков кода, представимых в виде соответствующих макросов (готовые оптимизированные синтезные структуры).
- Использование макросов позволяет повысить производительность разрабатываемого устройства, поэтому средства синтеза пытаются идентифицировать как можно большее их количество.
- Выделенные макросы в процессе последующей оптимизации, выполняемой на этапе синтеза, могут сохраняться в виде отдельных блоков или оптимизироваться совместно с окружающей логикой.

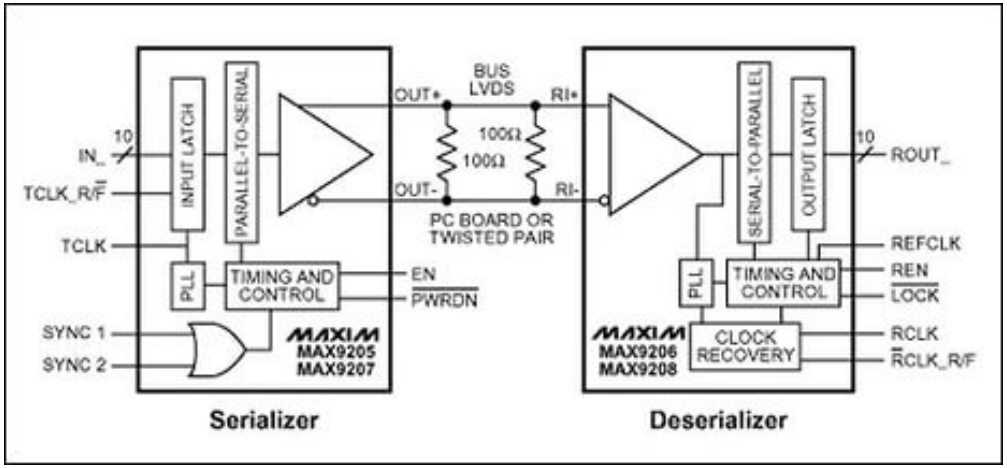
Сериалайзеры/десериалайзеры

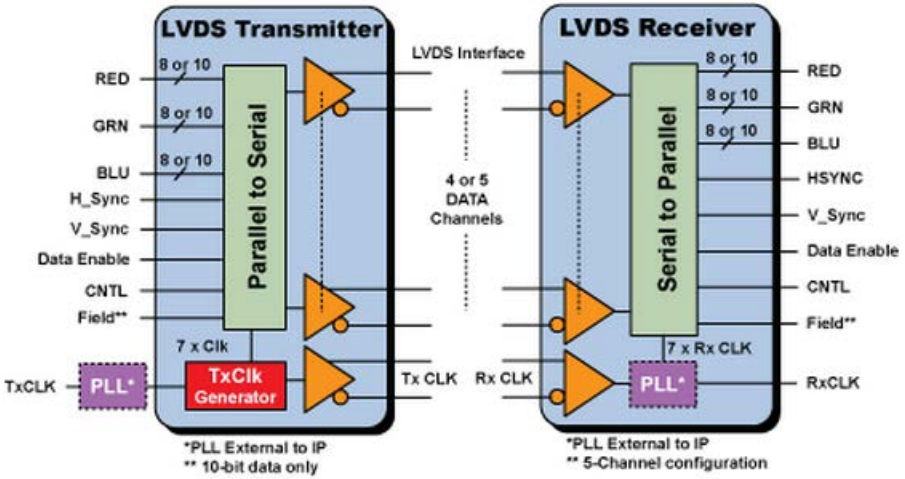
Сериализация — процесс перевода какой-либо структуры данных в последовательность битов. **Десериализация** — восстановление начального состояния структуры данных из битовой последовательности. **Типы**: с передачей синхросигнала, с жесткой частотой, с автоподстройкой *F.*



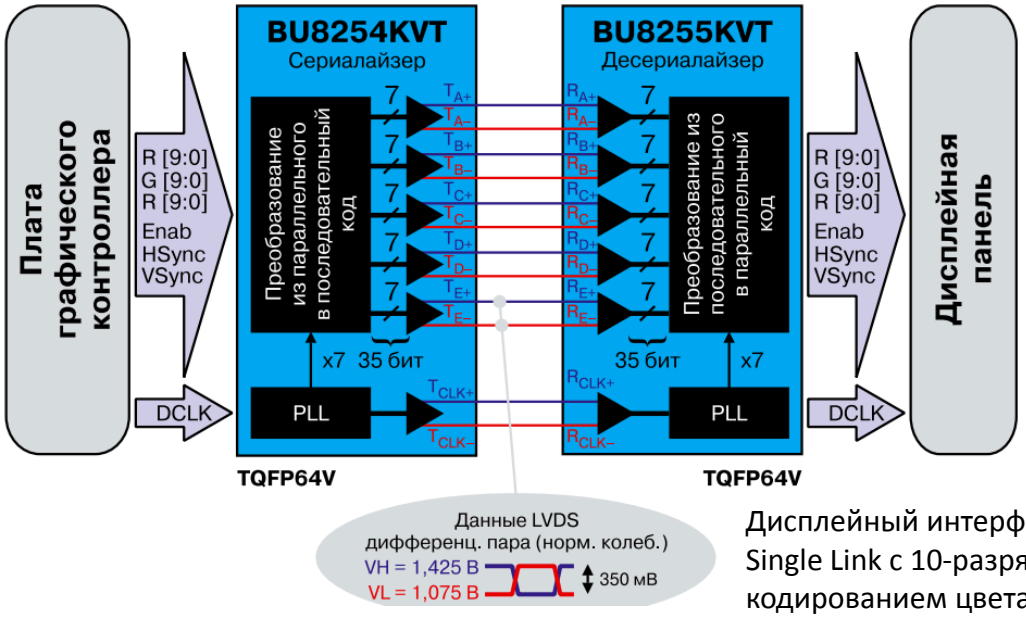


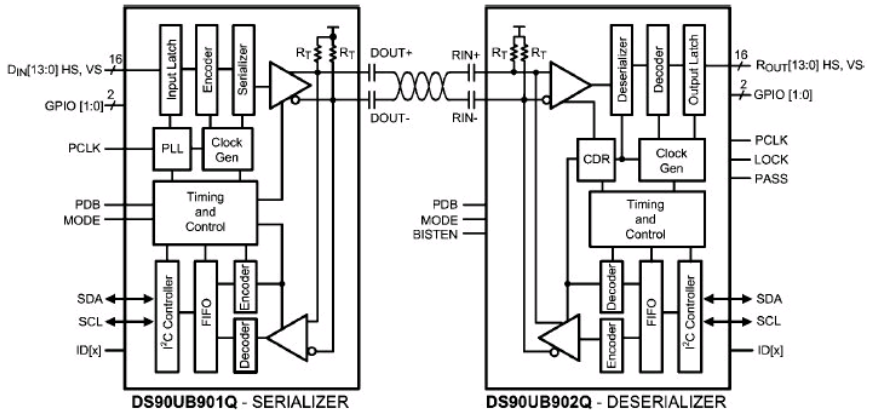




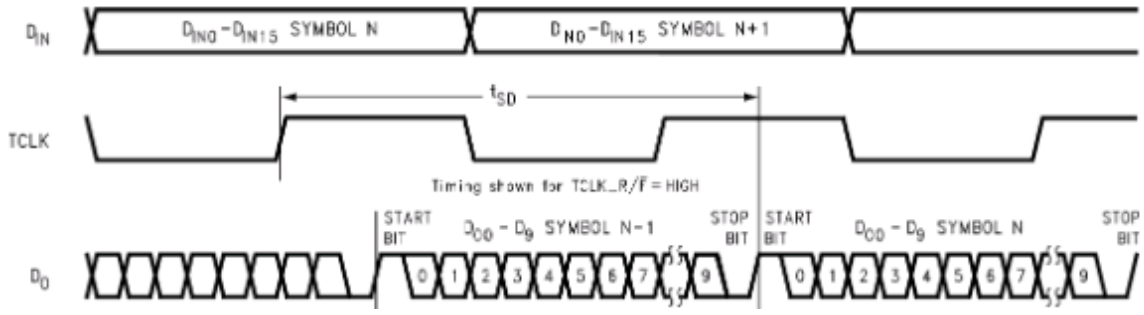


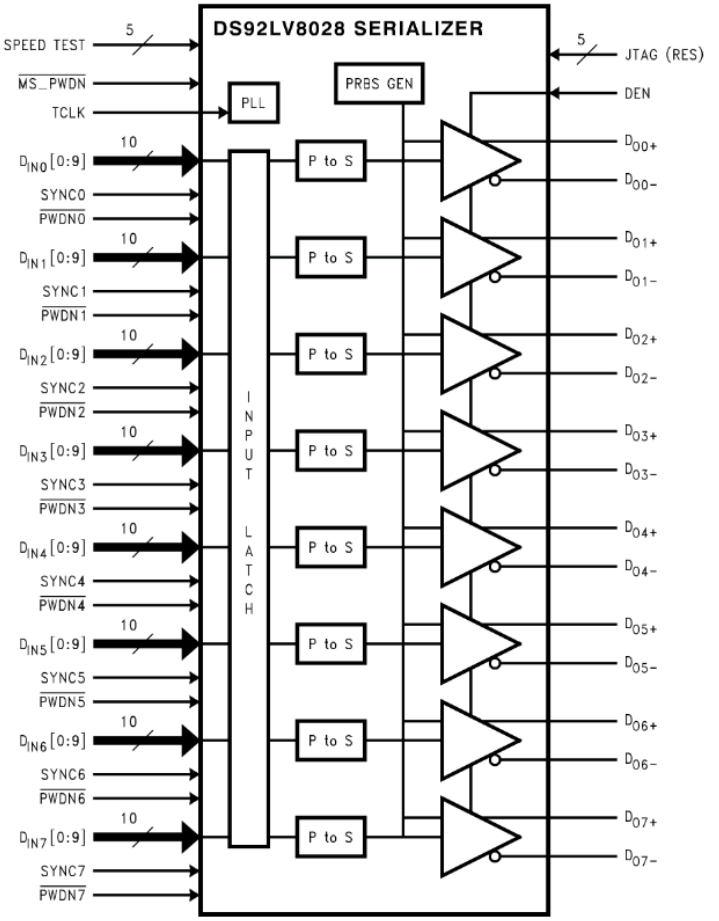
Сериалайзеры/десериалайзеры для видео приложений

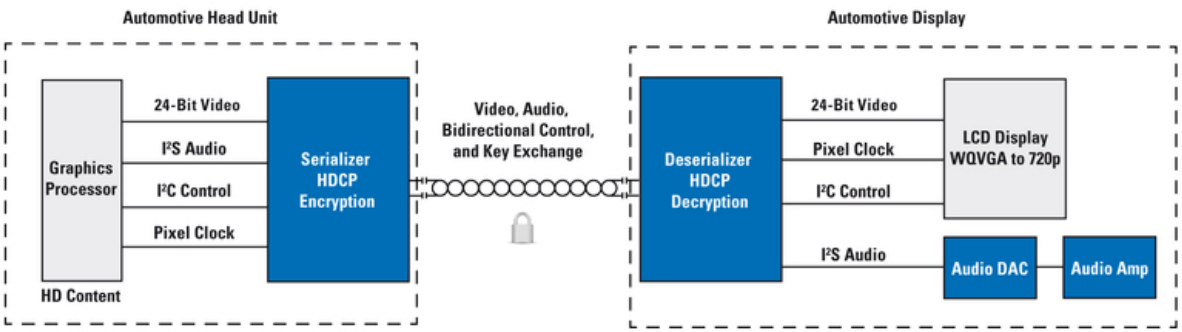




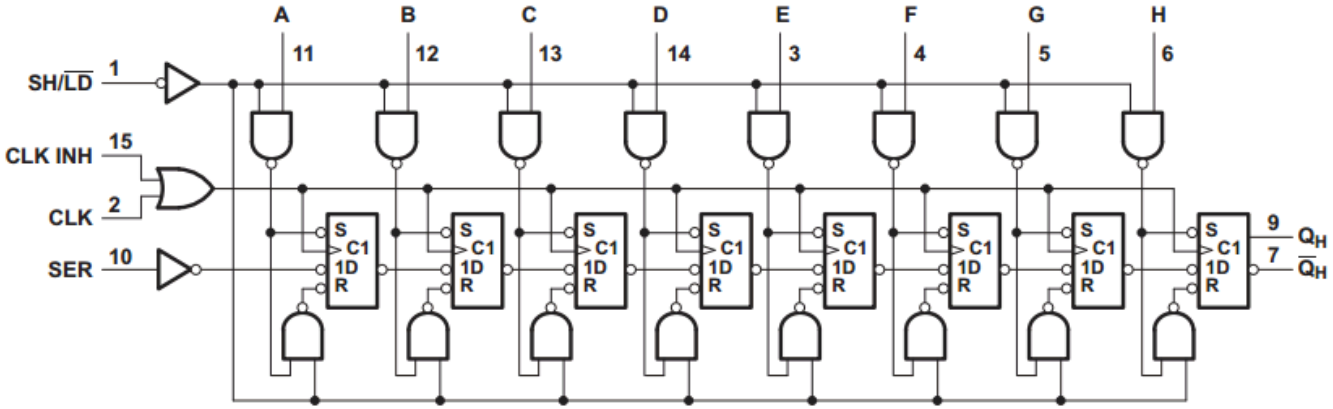
Дисплейный интерфейс Single Link с 10-разрядным кодированием цвета на чипсете BU8254/8255



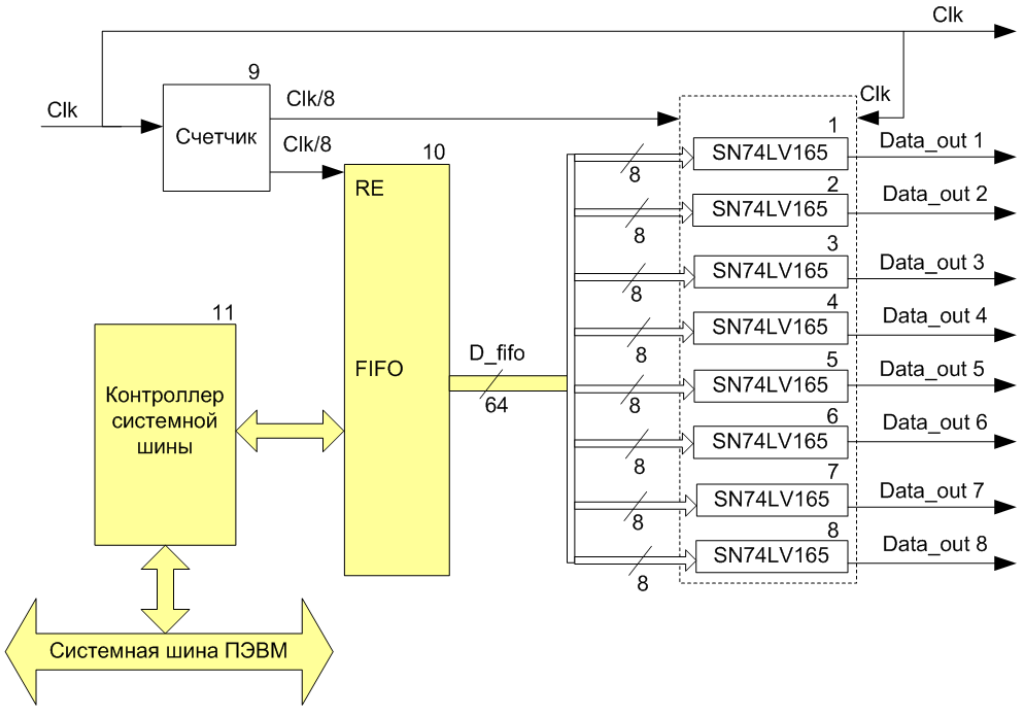


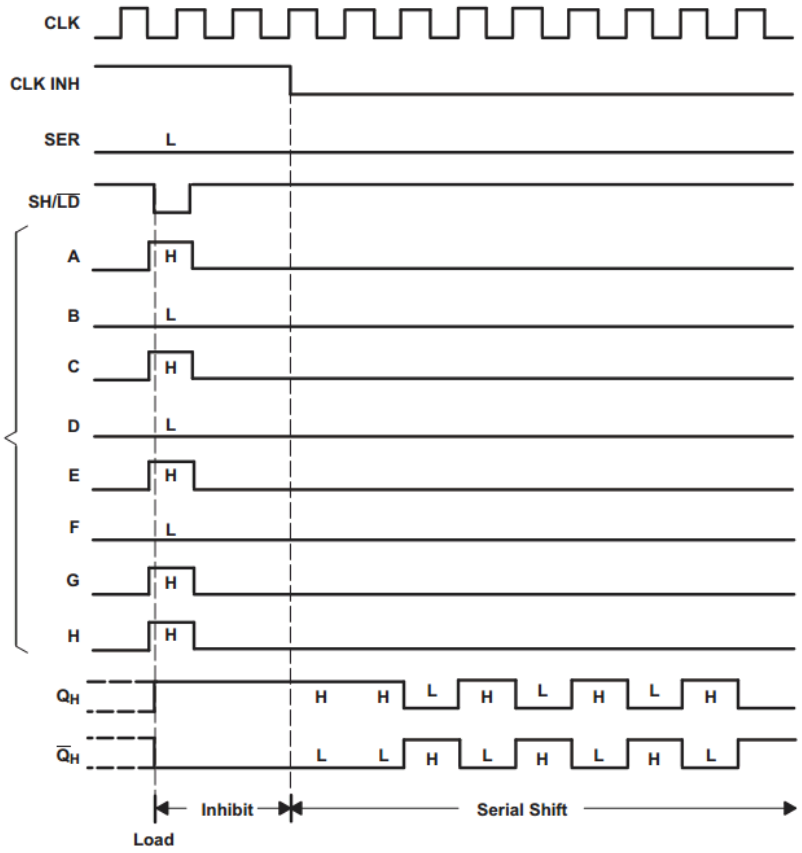


Сериалайзеры 8-1

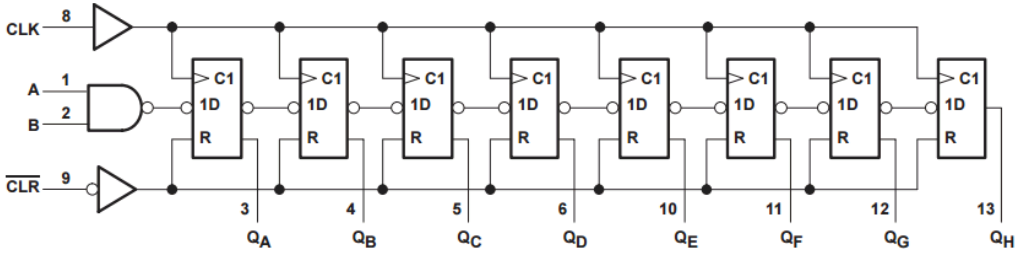


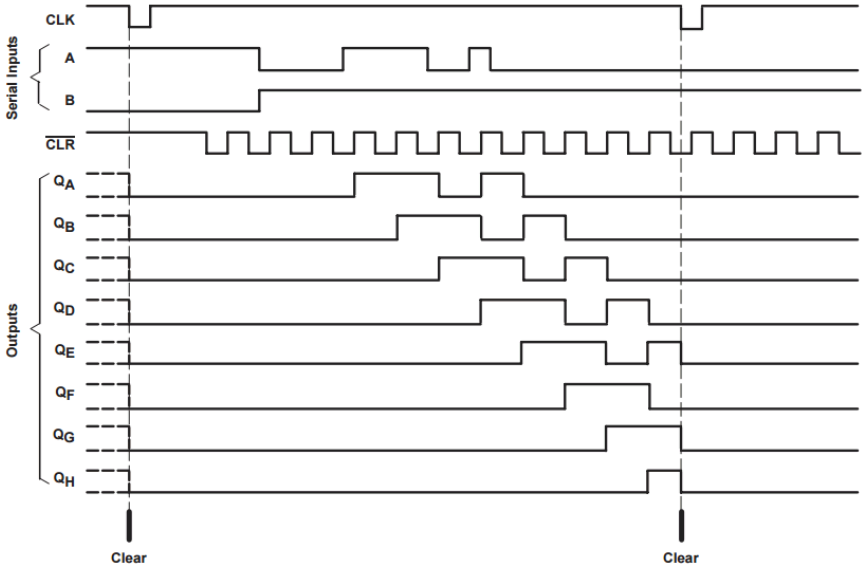
INPUTS			OPERATION
SH/$\overline{LD}$	CLK	CLK INH	
L	X	X	Parallel load
H	H	X	Q_0
H	X	H	Q_0
H	L	↑	Shift
H	↑	L	Shift



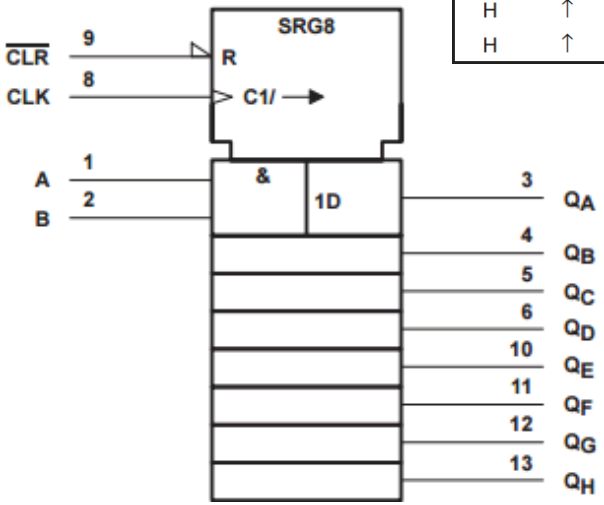


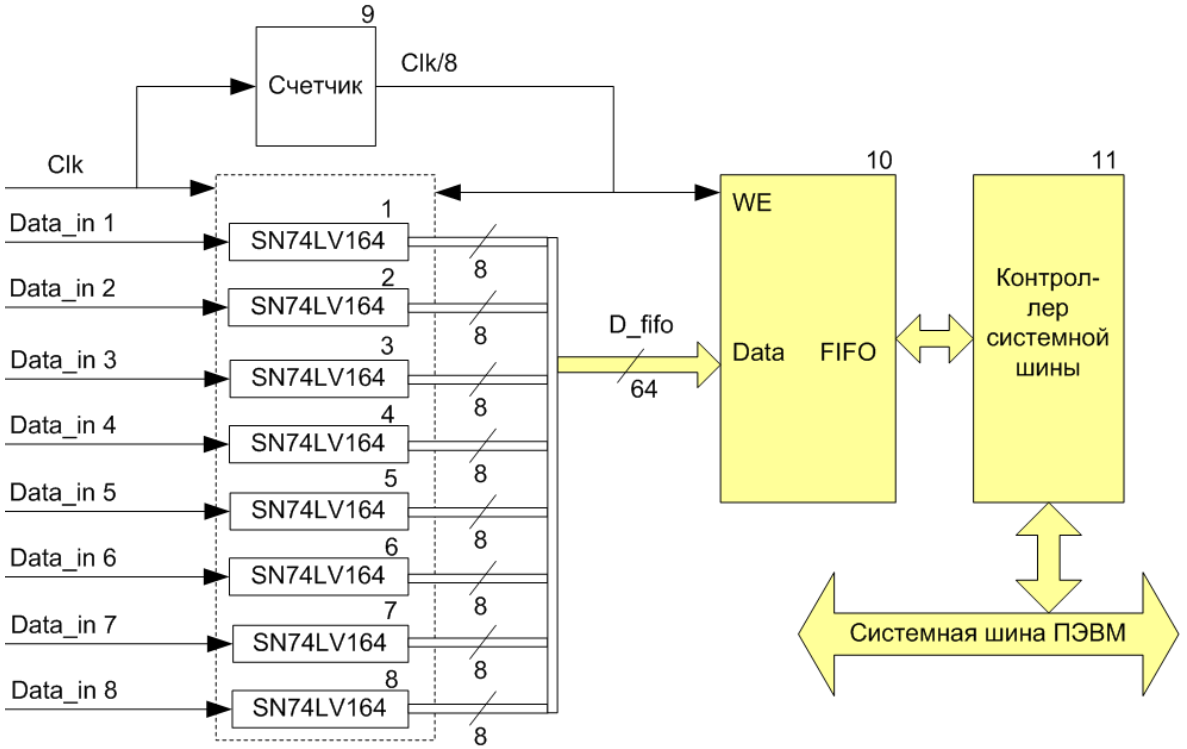
Десериалайзеры 1-8





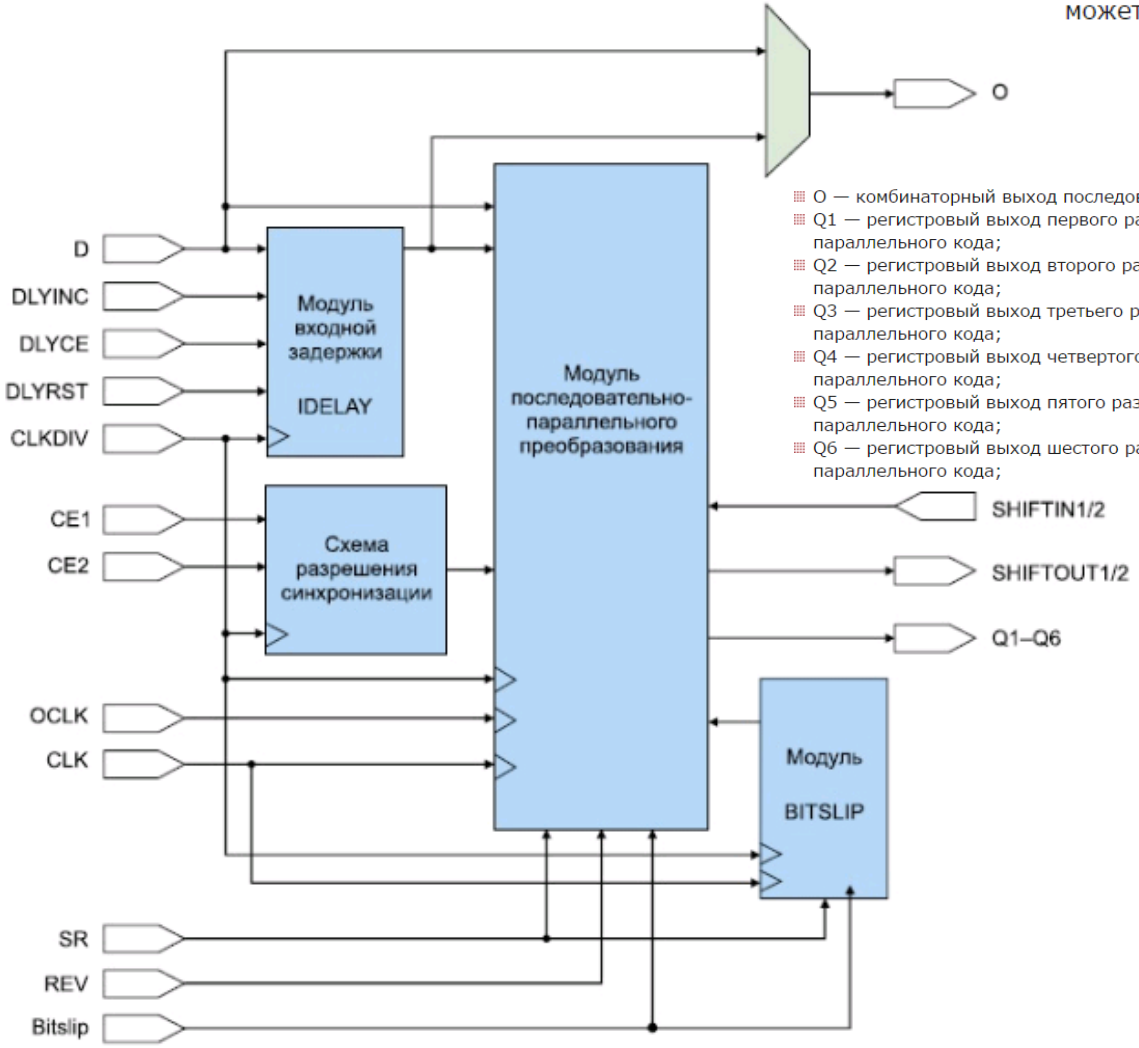
INPUTS				OUTPUTS		
$\overline{CLR}$	CLK	A	B	Q_A	$Q_B \ldots Q_H$	
L	X	X	X	L	L	L
H	L	X	X	Q_{A0}	Q_{B0}	Q_{H0}
H	↑	H	H	H	Q_{An}	Q_{Gn}
H	↑	L	X	L	Q_{An}	Q_{Gn}
H	↑	X	L	L	Q_{An}	Q_{Gn}





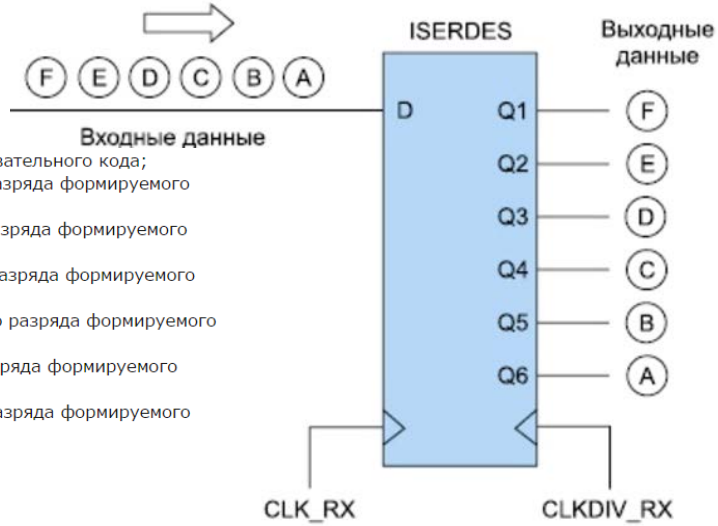
Сериалайзеры/десериалайзеры в ПЛИС

Структура входного последовательно-параллельного преобразователя ПЛИС серии Virtex-4



- O — комбинаторный выход последовательного кода;
- Q1 — регистровый выход первого разряда формируемого параллельного кода;
- Q2 — регистровый выход второго разряда формируемого параллельного кода;
- Q3 — регистровый выход третьего разряда формируемого параллельного кода;
- Q4 — регистровый выход четвертого разряда формируемого параллельного кода;
- Q5 — регистровый выход пятого разряда формируемого параллельного кода;
- Q6 — регистровый выход шестого разряда формируемого параллельного кода;

Модуль преобразования последовательного кода в параллельный формирует на выходе код, максимальное число разрядов которого может достигать шести.



- SHIFTOUT1, SHIFTOUT2 — выходы данных, предназначенные для организации каскадного соединения последовательно-параллельных преобразователей;
- BITSLIP — вход сигнала активизации функции ***Bitslip***;
- CE1, CE2 — входы сигналов разрешения синхронизации;
- CLK — вход сигнала синхронизации;
- CLKDIV — вход тактового сигнала, формируемого путем деления частоты сигнала синхронизации CLK, для регистровых выходов преобразователя, схемы входной задержки и модуля, осуществляющего функцию ***Bitslip***;
- D — вход данных, поступающих в виде последовательного кода;
- DLYCE — вход сигнала разрешения коррекции значения входной задержки;
- DLYINC — вход сигнала выбора режима изменения входной задержки (инкрементный или декрементный);
- DLYRST — вход сигнала сброса модуля входной задержки (установки заданного начального значения входной задержки);
- OCLK — вход сигнала синхронизации, предназначенного для организации передачи выходных данных, соответствующей высокоскоростным интерфейсам памяти;
- REV — зарезервированный вход, подключаемый к общему проводу;
- SHIFTIN1, SHIFTIN2 — входы данных, предназначенные для организации каскадного соединения последовательно-параллельных преобразователей;
- SR — вход сигнала сброса.

Сериалайзеры/десериалайзеры в ПЛИС

Схема формирования выходного кода в последовательно-параллельном преобразователе ПЛИС серии Virtex-4 в режиме "MEMORY"

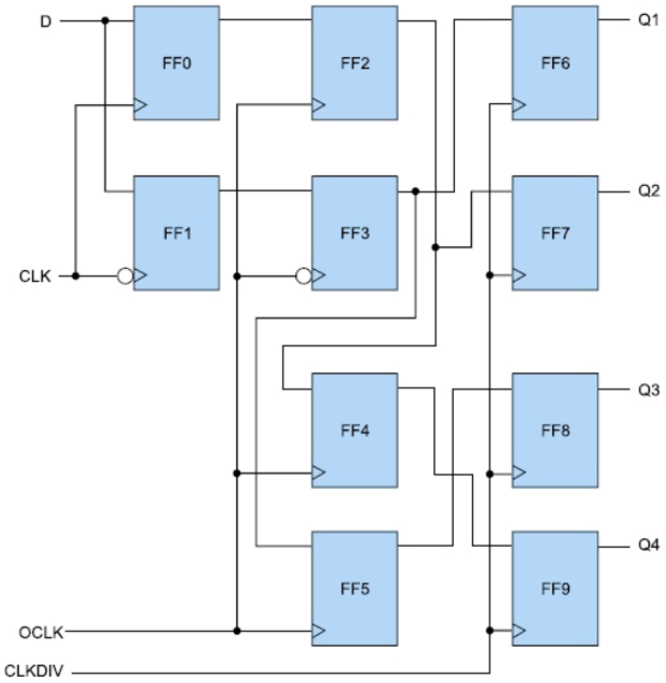
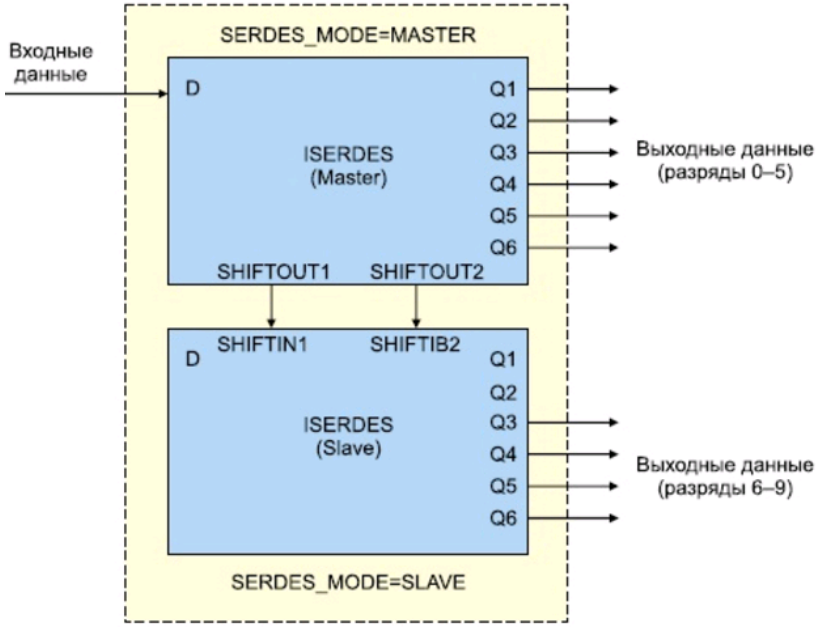
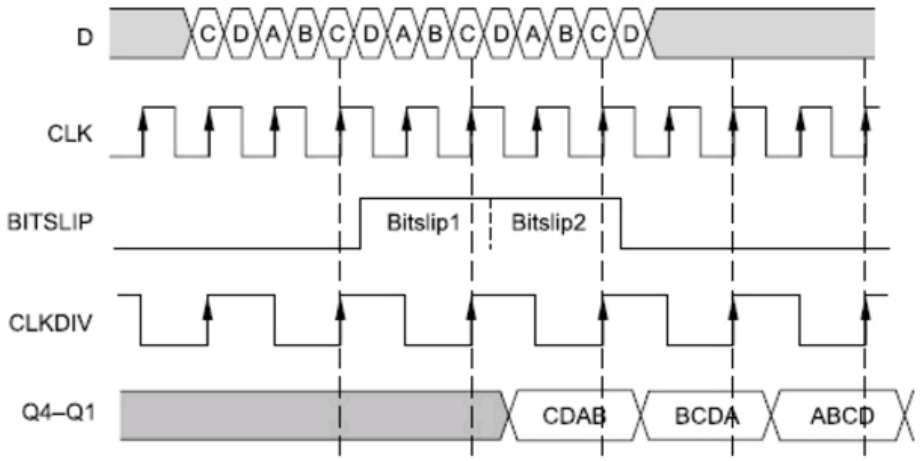


схема каскадного соединения двух входных последовательно-параллельных преобразователей ISERDES ПЛИС серии Virtex-4

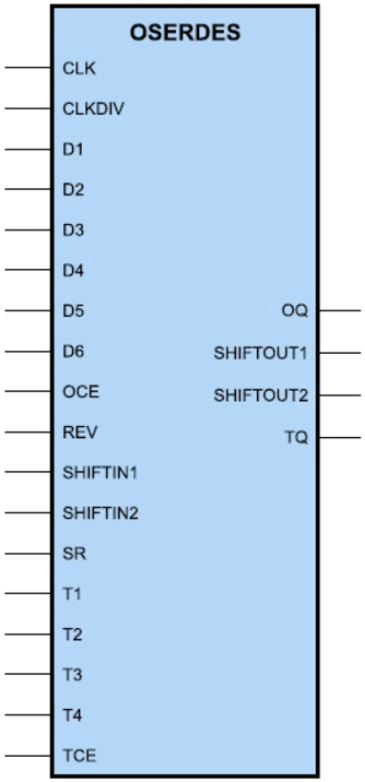


применение функции Bitslip в последовательно-параллельных преобразователях ISERDES ПЛИС серии Virtex-4



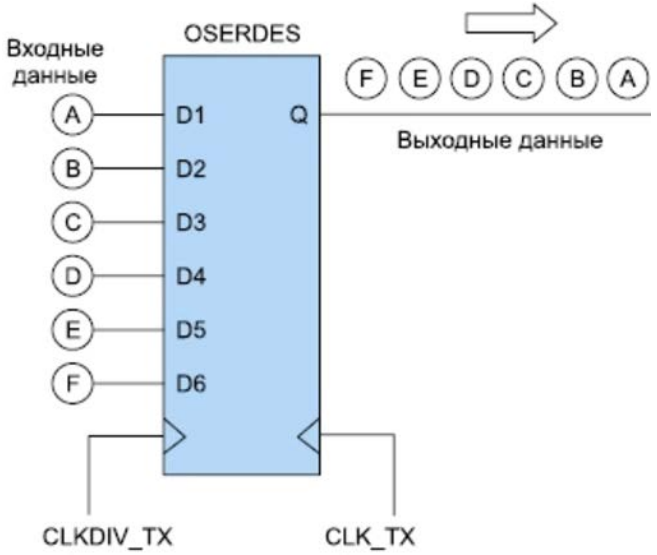
- ***BITSLIP_ENABLE*** — предоставляет возможность использования модуля, реализующего функцию ***Bitslip***, при конвертировании входного последовательного кода данных в параллельный.
- ***DATA_RATE*** — используется для выбора режима (скорости) передачи данных.
- ***DATA_WIDTH*** — определяет разрядность выходного параллельного кода данных.
- ***INTERFACE_TYPE*** — позволяет выбрать интерфейс передачи выходных данных.
- ***IOBDELAY*** — указывает выходы, в цепи сигналов которых задействуется модуль входной задержки.
- ***IOBDELAY_TYPE*** — применяется для выбора типа используемого модуля входной задержки.
- ***IOBDELAY_VALUE*** — определяет значение начальной или фиксированной задержки используемого модуля входной задержки.
- ***NUM_CE*** — указывает количество входов сигналов разрешения синхронизации.
- ***SERDES_MODE*** — устанавливает режим функционирования формируемого входного последовательно-параллельного преобразователя.

Сериалайзеры/десериалайзеры в ПЛИС



- OQ — выход данных, представленных в виде последовательного кода.
- SHIFTOUT1 — выход, предназначенный для организации каскадного соединения формируемых преобразователей (подключается к входу SHIFTIN1 ведущего модуля преобразования данных).
- SHIFTOUT2 — выход, используемый для каскадного соединения формируемых преобразователей (подключается к входу SHIFTIN2 ведущего модуля преобразования данных).
- TQ — выход сигнала управления тристабильным буферным элементом.
- CLK — вход сигнала синхронизации.
- CLKDIV — вход тактового сигнала, формируемого путем деления частоты сигнала синхронизации CLK.
- D1–D6 — входы данных, представленных в виде параллельного двоичного кода.
- OCE — вход сигнала разрешения синхронизации для выходного регистра.
- REV — зарезервированный вход, подключаемый к общему проводу.
- SHIFTIN1 — вход, предназначенный для организации каскадного соединения формируемых преобразователей (подключается к выходу SHIFTOUT1 подчиненного модуля преобразования данных).
- SHIFTIN2 — вход, используемый для каскадного соединения формируемых преобразователей (подключается к выходу SHIFTOUT2 подчиненного модуля преобразования данных).
- SR — вход сигнала асинхронного сброса.
- T1–T4 — входы параллельного двоичного кода управления выходом тристабильного буферного элемента.
- TCE — вход сигнала разрешения синхронизации для регистра сигнала управления тристабильным выходом.

Соответствие разрядов данных на входах и выходе



Структура выходного параллельно-последовательного преобразователя ПЛИС серии Virtex-4

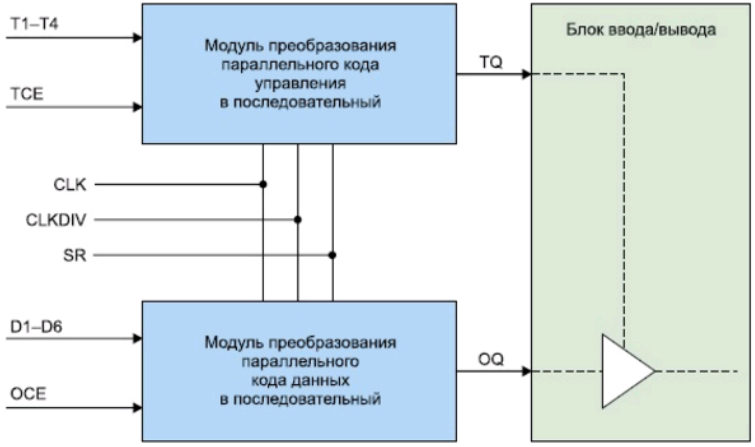
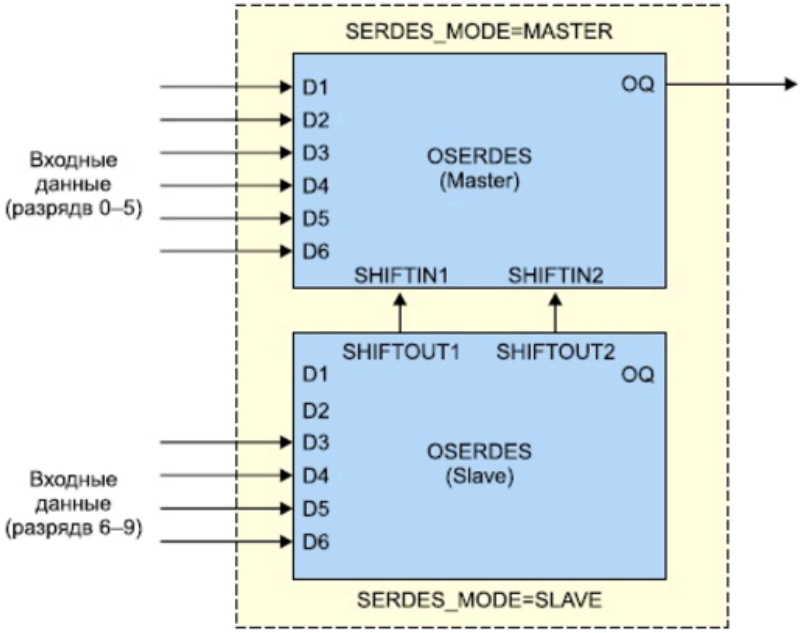
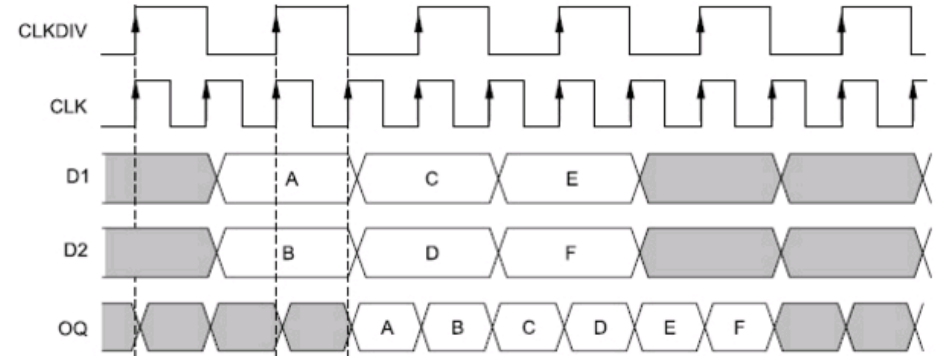


Схема каскадного соединения



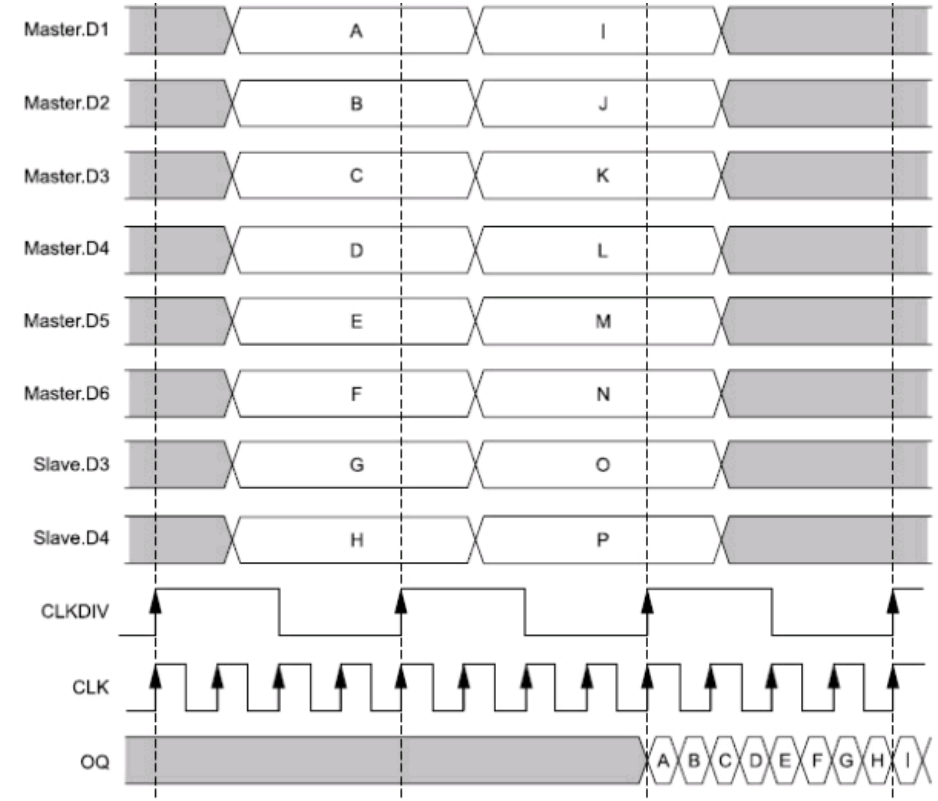
Сериалайзеры/десериалайзеры в ПЛИС

Временные диаграммы сигналов, соответствующие стандартному режиму функционирования параллельно-последовательного преобразователя ПЛИС серии Virtex-4

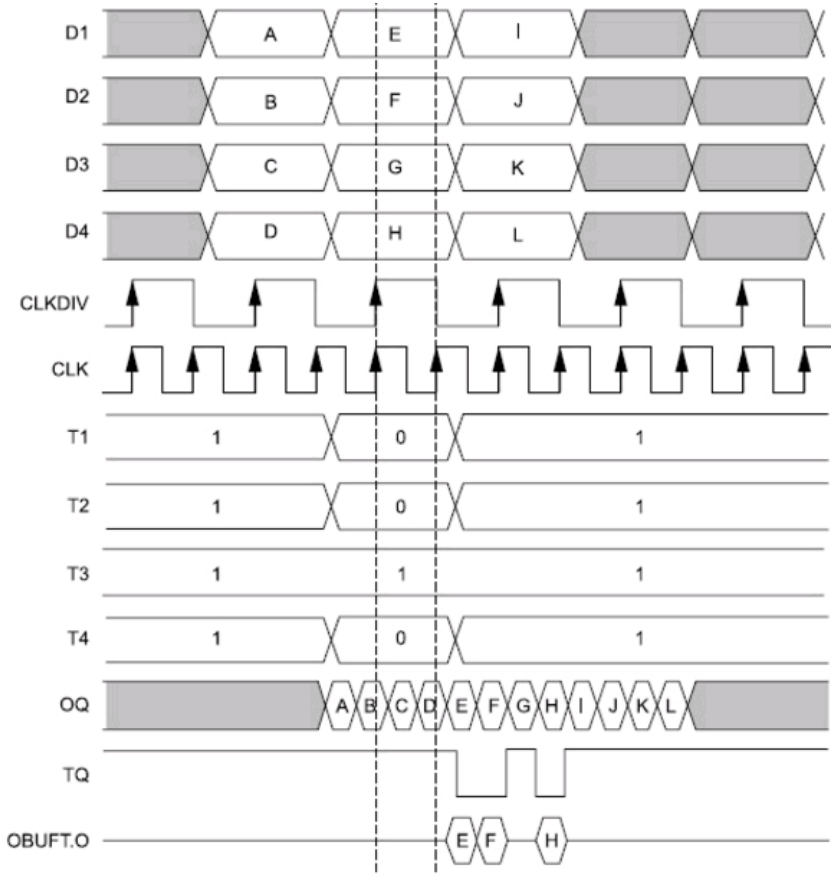


- ***DATA_RATE_OQ*** — определяет режим (скорость) передачи данных на выходе OQ.
- ***DATA_RATE_TQ*** — устанавливает режим (скорость) передачи данных на выходе сигнала управления тристабильным буферным элементом.
- ***DATA_WIDTH*** — указывает разрядность входного параллельного кода данных.
- ***INIT_OQ*** — определяет начальное состояние выхода данных OQ.
- ***INIT_TQ*** — устанавливает начальное состояние выхода сигнала управления TQ.
- ***SERDES_MODE*** — задает режим функционирования создаваемого выходного параллельно-последовательного преобразователя.
- ***SRVAL_OQ*** — указывает состояние выхода данных OQ при активном уровне сигнала на входе сброса.
- ***SRVAL_TQ*** — определяет состояние выхода сигнала управления TQ при наличии активного уровня сигнала на входе сброса.
- ***TRISTATE_WIDTH*** — задает разрядность параллельного кода управления тристабильным выходом.

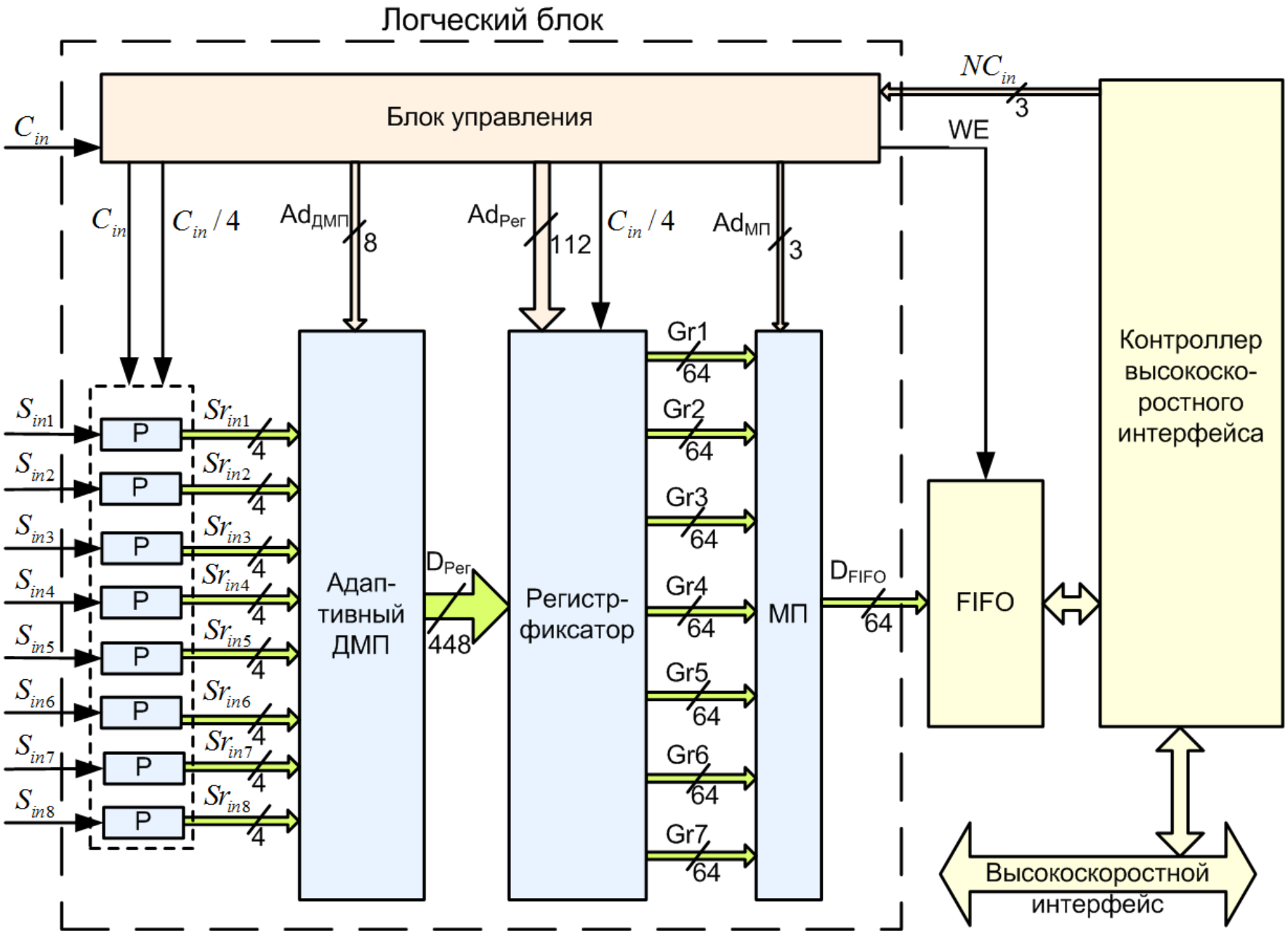
параллельного кода в последовательный, функционирующего с удвоенной скоростью передачи данных



формирование последовательного кода данных на тристабильном выходе параллельно-последовательного преобразователя

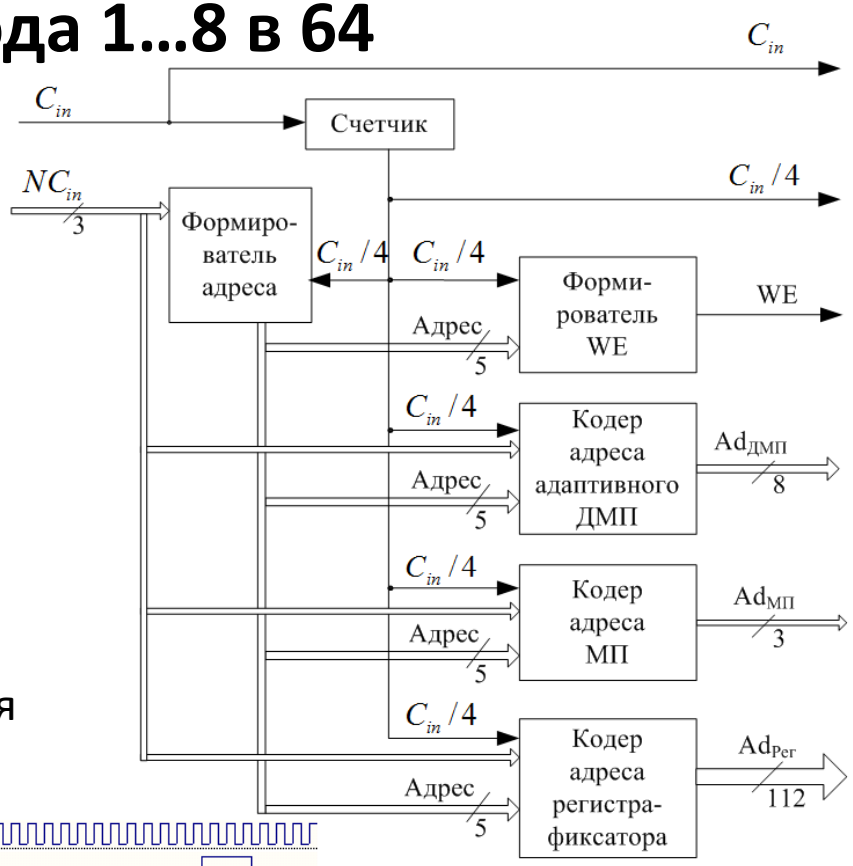


Преобразователь 1…8 в 64. Схема

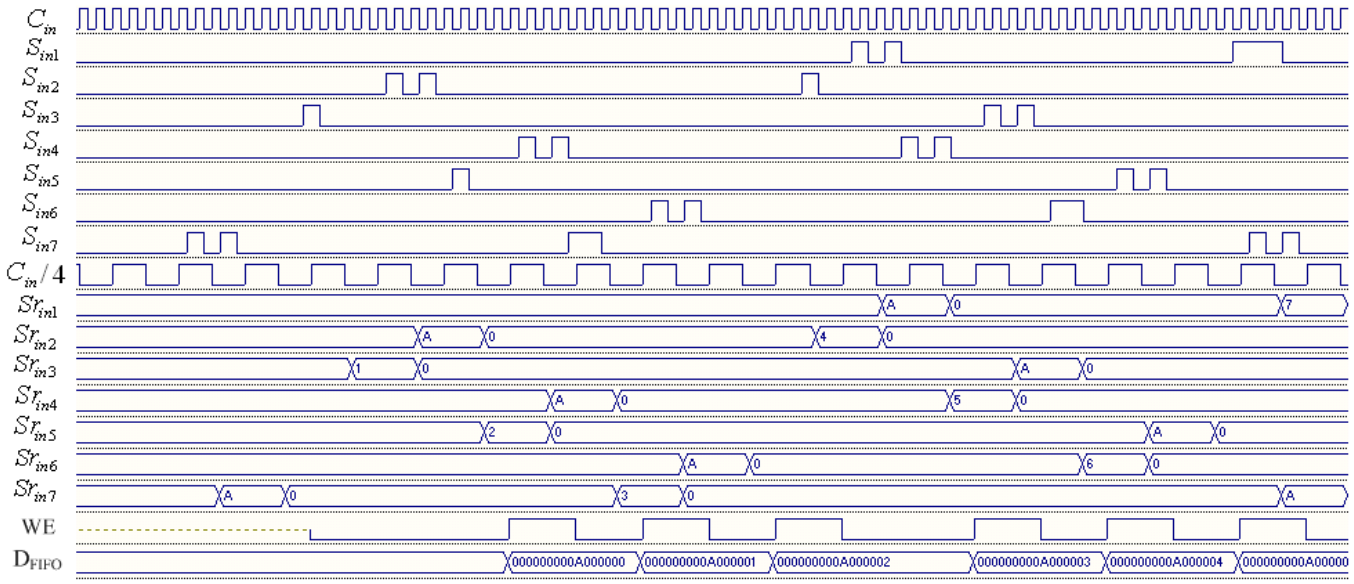


Преобразователь ввода 1…8 в 64

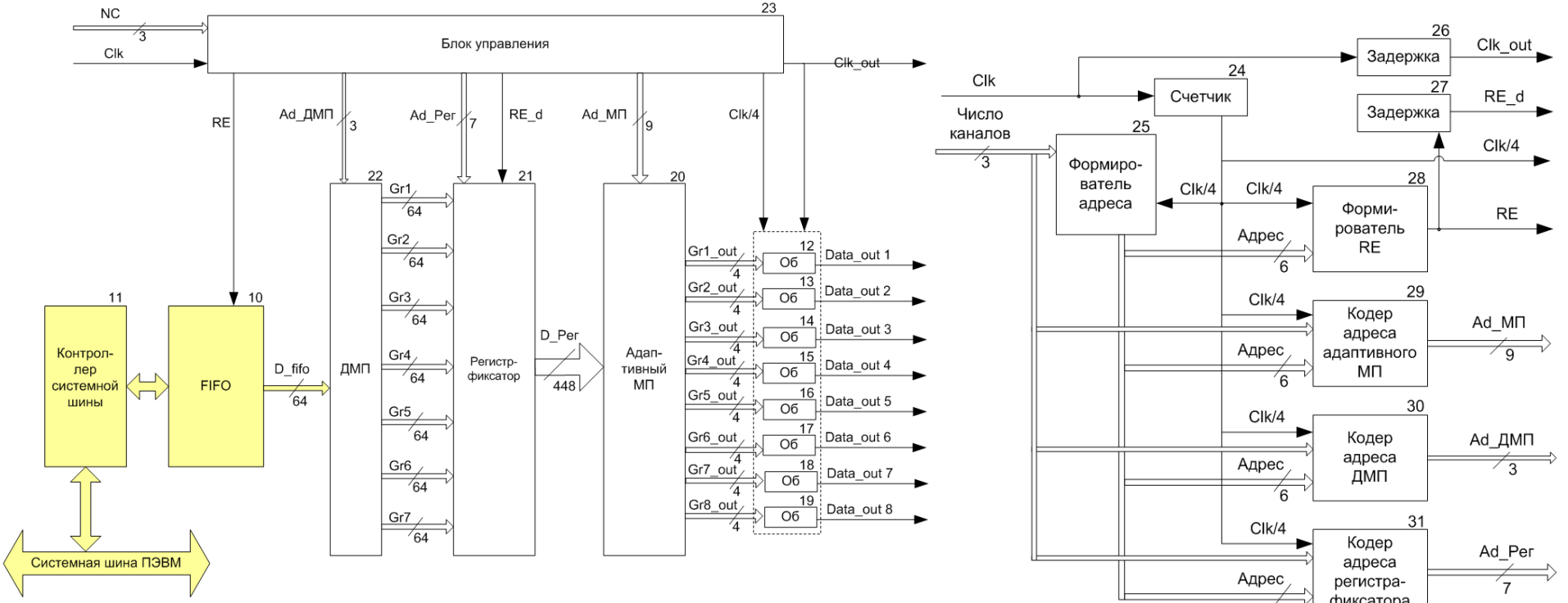
Схема блока управления



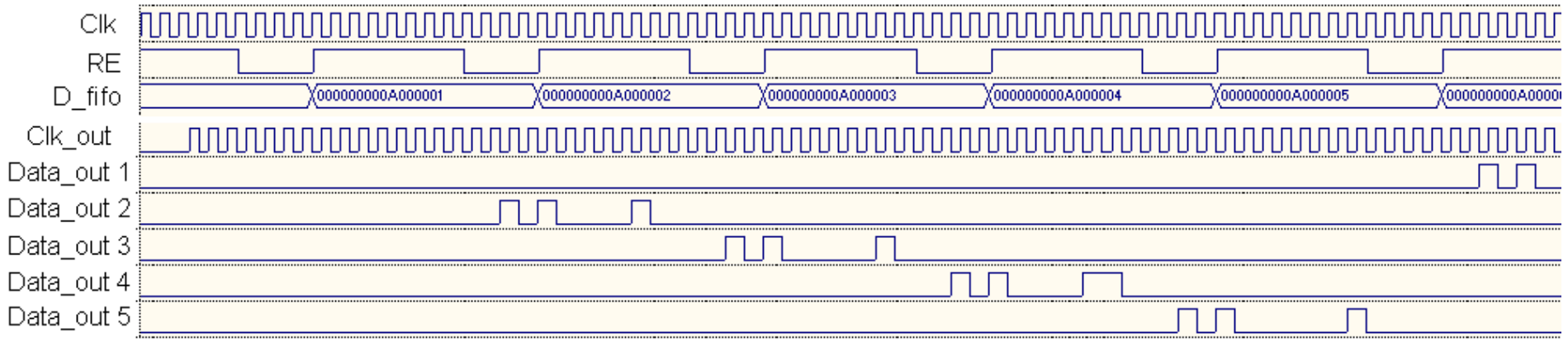
Временные диаграммы работы устройства для случая ввода данных по 7 каналам



Преобразователь вывода 64 в 1…8



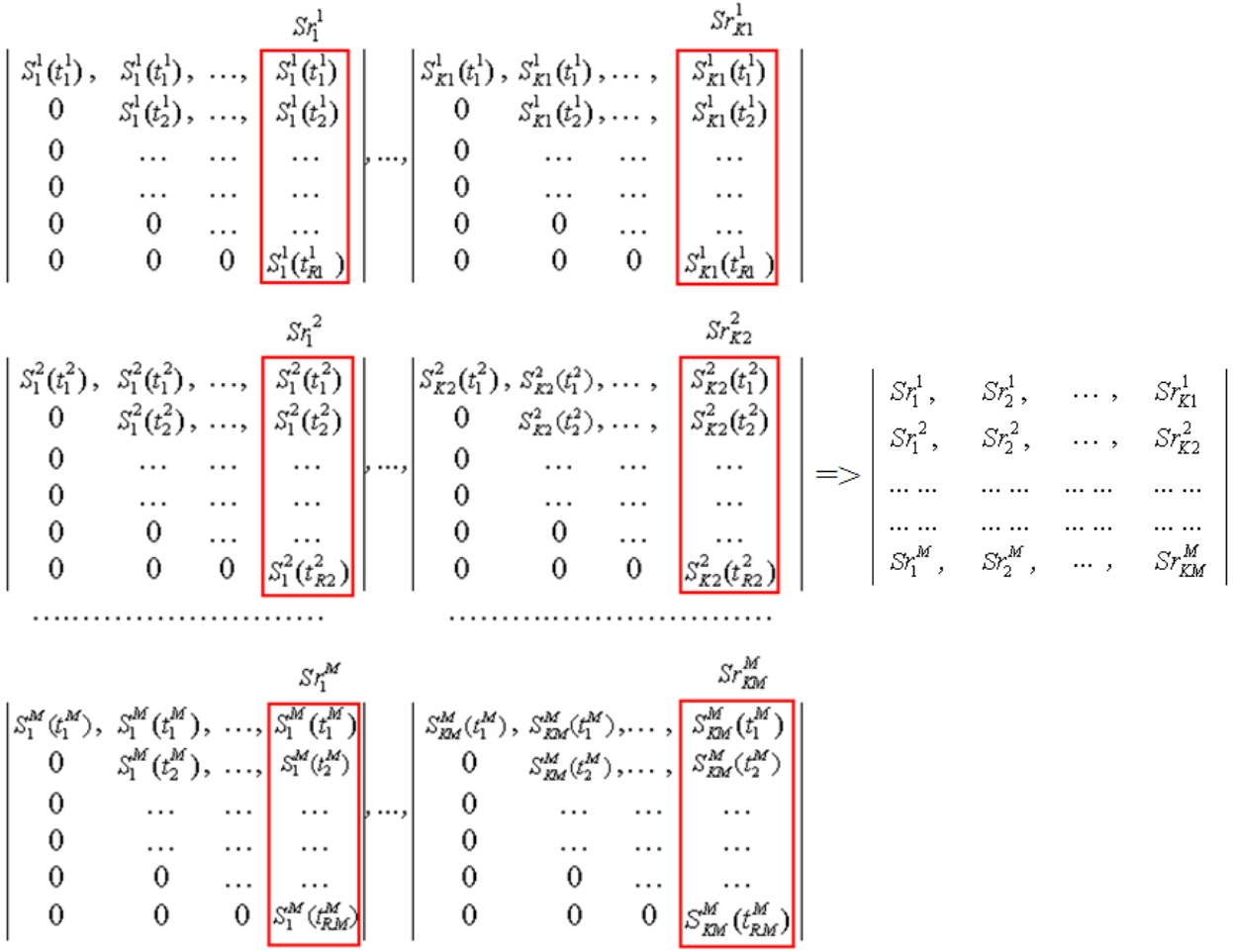
Временные диаграммы работы устройства для случая вывода данных по 5 каналам



Метод преобразования потоков данных: ввод и распараллеливание

В общем случае вводится *M* групп входных потоков, в каждую из которых входит *K1, K2,…,KM* информационных сигналов *S* и один общий сигнал синхронизации *C*, которые можно представить в виде матрицы *A* (верхний индекс сигналов *S* соответствует номеру группы, а нижний индекс – номеру информационного сигнала в группе).

$$A=\begin{vmatrix} S_2^1, & \dots, & S_{K1}^1 & C_1 \\ S_2^2, & \dots, & S_{K2}^2 & C_2 \\ \dots\dots & \dots\dots & \dots\dots & \dots\dots \\ \dots\dots & \dots\dots & \dots\dots & \dots\dots \\ \dots\dots & \dots\dots & \dots\dots & \dots\dots \\ S_2^M, & \dots, & S_{KM}^M, & C_M \end{vmatrix}$$



Здесь $R1, R2, \dots, RM$ являются параметрами распараллеливания каждого входного потока – на сколько потоков распараллеливается входной поток. Времена

$$t_1^1, t_2^1, \dots, t_R^M$$

соответствуют моментам появления фронта тактового сигнала каждой входной группы.

Метод преобразования потоков данных: демультиплексирование и буферизация

Демультиплексирование математически описывается как группировка матрицу распараллеленных потоков *Sr* в единый вектор с учетом распределения *R(X)*

$Sd^1(t), Sd^2(t), \ldots, Sd^M(t)$ - Демультиплексированные входные потоки даных.

$$\begin{vmatrix} Sd^1(t) \\ Sd^2(t) \\ \ldots\ \ldots \\ \ldots\ \ldots \\ Sd^M(t) \end{vmatrix} = \begin{vmatrix} Sr_1^1, & Sr_2^1, & \ldots, & Sr_{K1}^1 \\ Sr_1^2, & Sr_2^2, & \ldots, & Sr_{K2}^2 \\ \ldots\ \ldots & \ldots\ \ldots & \ldots\ \ldots & \ldots\ \ldots \\ \ldots\ \ldots & \ldots\ \ldots & \ldots\ \ldots & \ldots\ \ldots \\ Sr_1^M, & Sr_2^M, & \ldots, & Sr_{KM}^M \end{vmatrix} \times R(X)$$

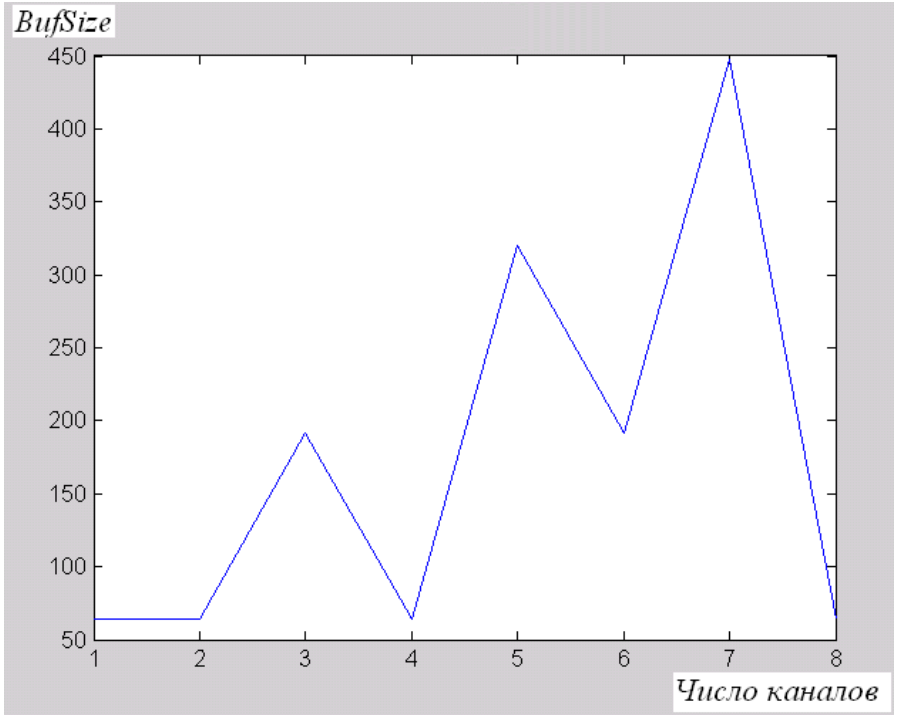
Необходимый размер *BufSize* (бит) буферной памяти

$$BufSize = \sum_1^M HOK[(R1, R1, \ldots, RM) \,\&\, (X_M - X_{M-1})]$$

где *НОК* – функция вычисления наименьшего общего кратного.

Так как входные потоки $S_1^1, S_2^1, \ldots, S_{KM}^M$ распараллеливаются в соответствии с выражением, то размер буфера становится зависимым от разрядностей распараллеленных потоков $R1, R1, \ldots, RM$ и распределения R(X) пропускной способности интерфейса ввода.

Зависимость *BufSize* от числа каналов ввода *NC* для 64-разрядного интерфейса



Универсальный преобразователь входных потоков. Входные сигналы, множество потоков на разных скоростях

Fc_M – частота тактового сигнала C_M

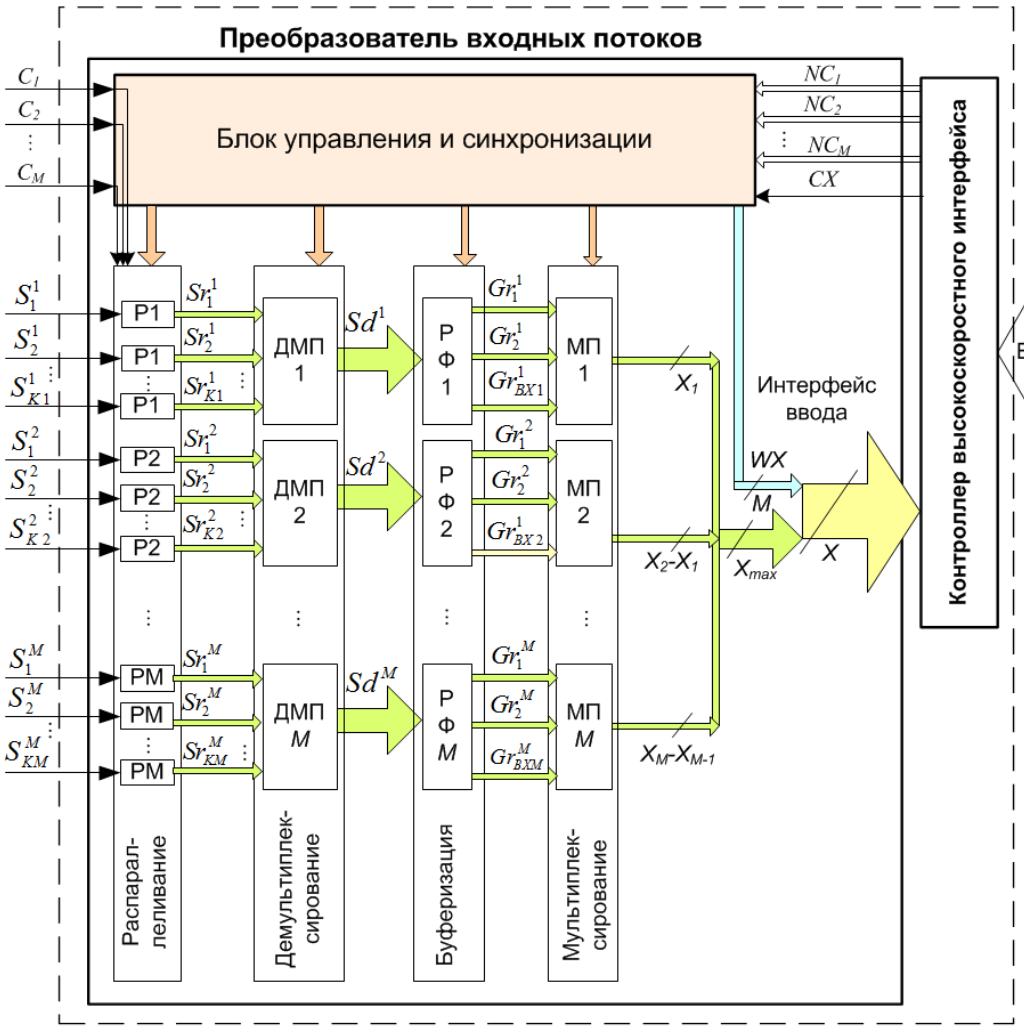
Значение скорости V_M (бит/с) потока от М-й группы связано с частотой тактового сигнала C_M и вычисляется по формуле

$$V_M = NC_M \times F_{CM}$$

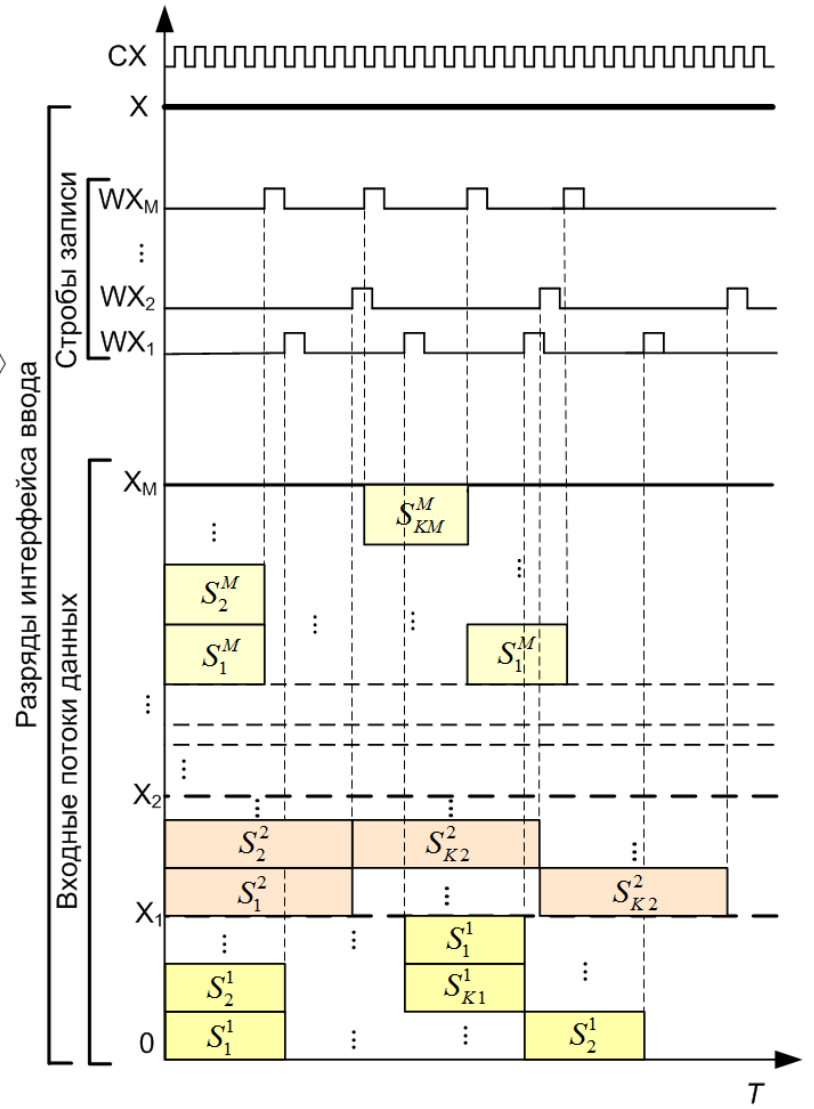
NC_M – выбранное число каналов ввода для М-й группы входных сигналов

Универсальный преобразователь входных потоков

Схема



Временные диаграммы



Разрядности $B_1, B_2, ..., B_M$ потоков $Sd^1, Sd^2, ..., Sd^M$ Разрядности

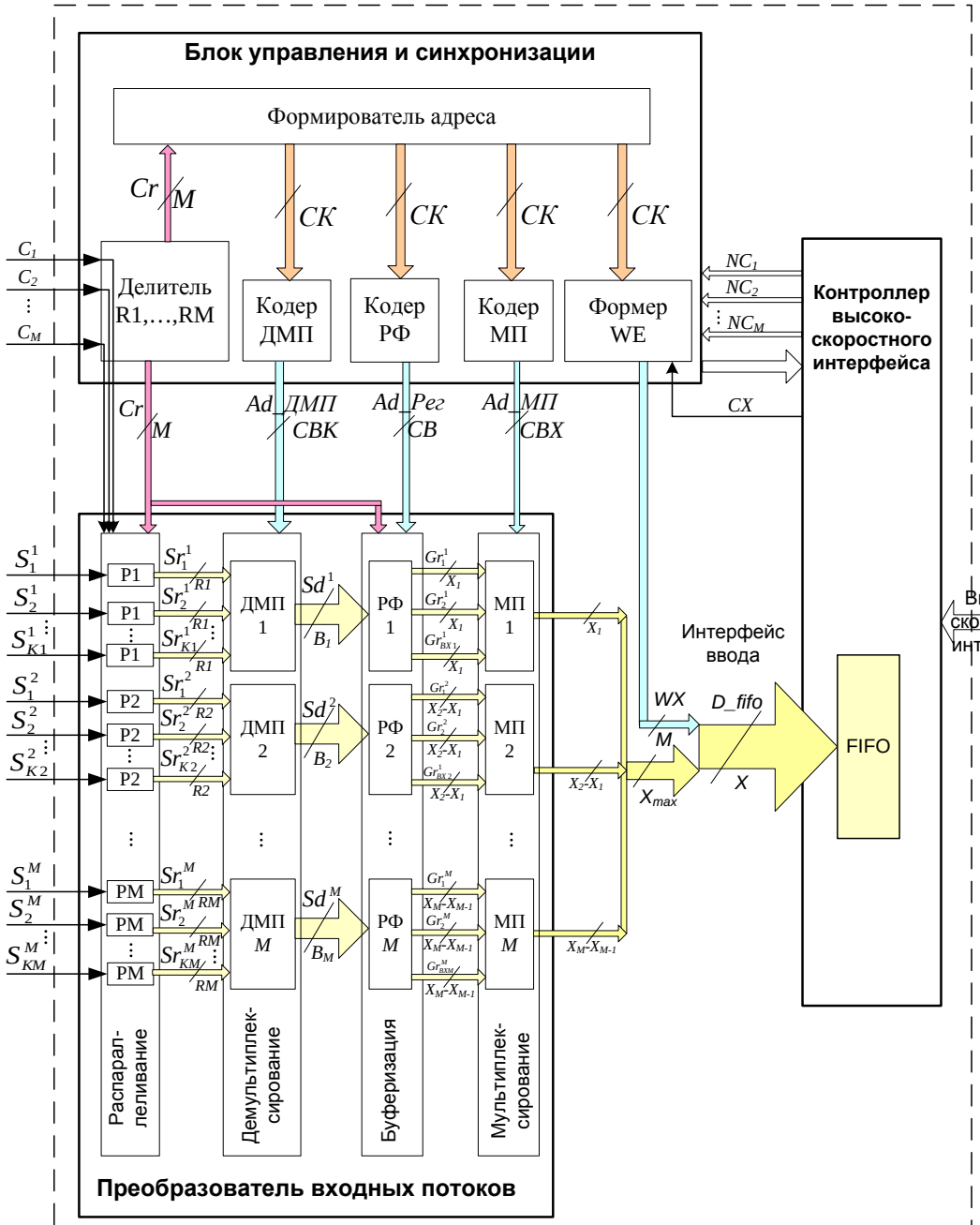
$$B_M = \text{НОК}[RM \,\&\, (X_M - X_{M-1})]$$

$$BX_M = \frac{B_M}{X_M - X_{M-1}}$$

$WX_1, WX_2, ..., WX_M$ - стробирующие сигналы

В высокоскоростном интерфейсе выделяются определенные разряды для ввода каждой группы $(1; X_1)$ $(X_1 + 1; X_2)$ $(X_{M-1} + 1; X_M)$

Универсальный преобразователь входных потоков



Задержка преобразования каждого входного потока измеряется тактами $C_1, C_2, ..., C_M$ синхросигналов и вычисляется по формуле:

$$CXK_M = ceil(\frac{X_M - X_{M-1}}{RM}) + 1$$

где +1 задержка на один такт при записи в РФ.

Блок управления и синхронизации обеспечивает согласование работы остальных элементов подсистемы ввода данных. На его входы подаются входные тактовые сигналы $C_1, C_2, ..., C_M$ и тактовый сигнал контроллера системной шины СХ.

Блок управления содержит делитель $R1, R2, ..., RM$ частот тактовых сигналов $C_1, C_2, ..., C_M$ в соответствие с кратностью распараллеливания, формирователь адреса, кодер ДМП, кодера РФ, кодер МП и формер WE. Формирователь адреса задает универсальный адрес для кодера ДМП, кодера РФ, кодера МП и формера WE. Разрядность СК универсального адреса рассчитывается по формуле:

$$CK = \sum_1^M ceil(\log_2 RM)$$

ceil – функция округления числа до большего целого значения

Кодеры адреса ДМП, МП и регистров-фиксаторов представляют собой комбинационные логические схемы.

Универсальный преобразователь входных потоков

Кодер ДМП задает адресацию Ad_ДМП для демультиплексоров ДМП1, ДМП2,…, ДМПМ . Его разрядность СВК рассчитывается по формуле:

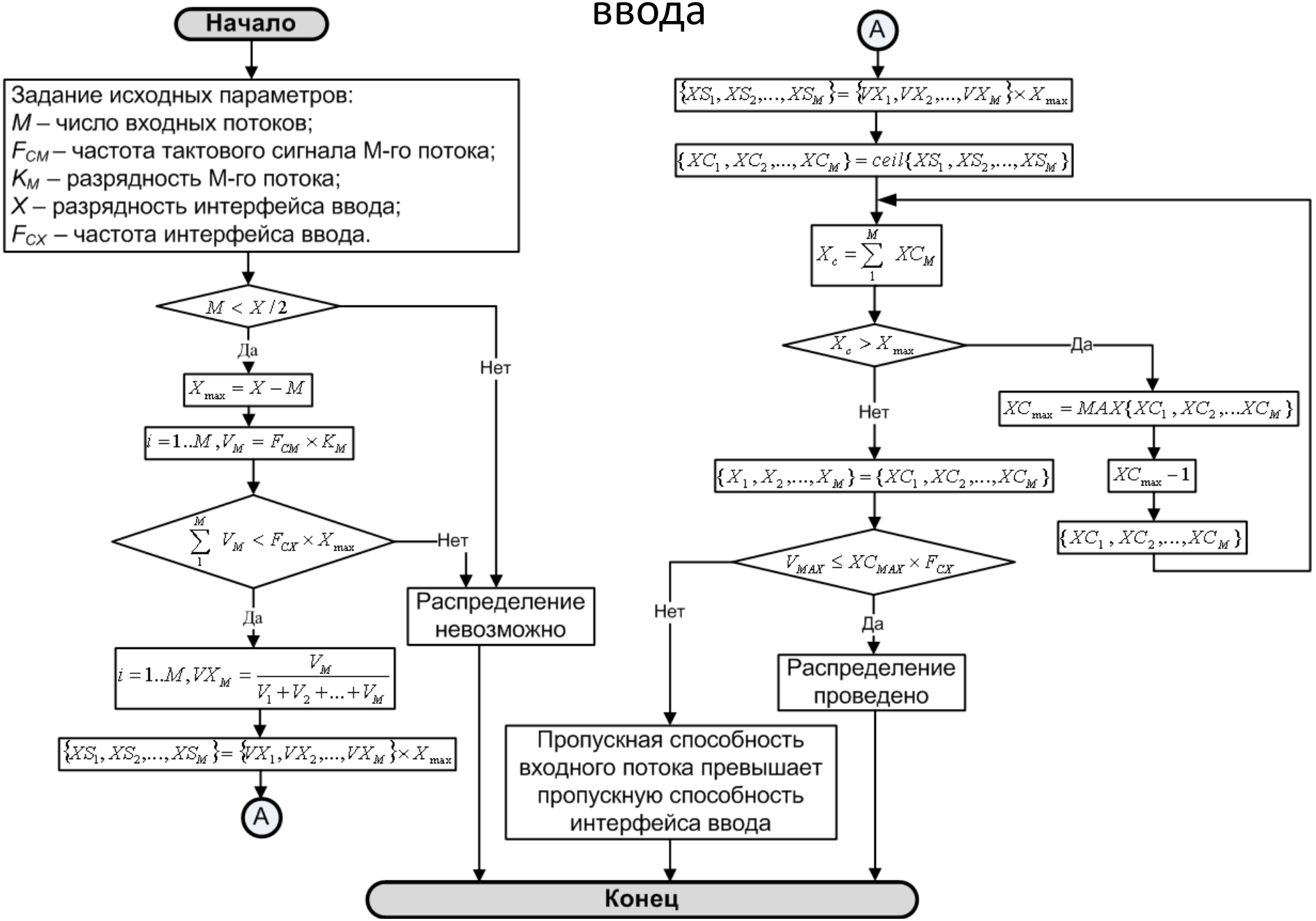
$$CBK = \sum_{1}^{M} ceil[\log_2(\frac{B_M}{RM})]$$

Кодер РФ задает адресацию Ad_Рег для регистров-фиксаторов РФ1,РФ2,…,РФМ. Его разрядность СВ рассчитывается по формуле: $CB = \sum_{1}^{M} ceil(\log_2 B_M)$

Кодер МП задает адресацию Ad_МП для мультиплексоров МП1, МП2,…,МПМ. Его разрядность СВХ рассчитывается по формуле: $CBX = \sum_{1}^{M} ceil(\log_2 BX_M)$

Формер WE предназначен для формирования стробирующих сигналов $WX_1, WX_2, \ldots, WX_M$ записи каждой группы входного потока данных в соответствие с выходным пакетом данных.

Алгоритм распределения пропускной способности интерфейса ввода

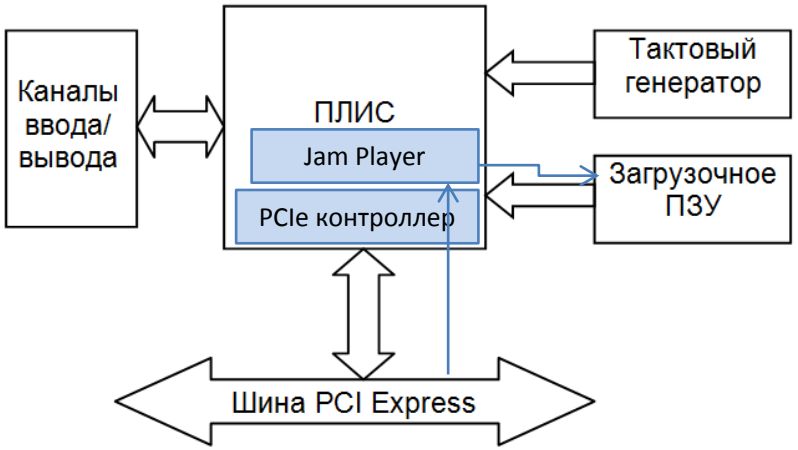


Пример распределения пропускной способности интерфейса ввода

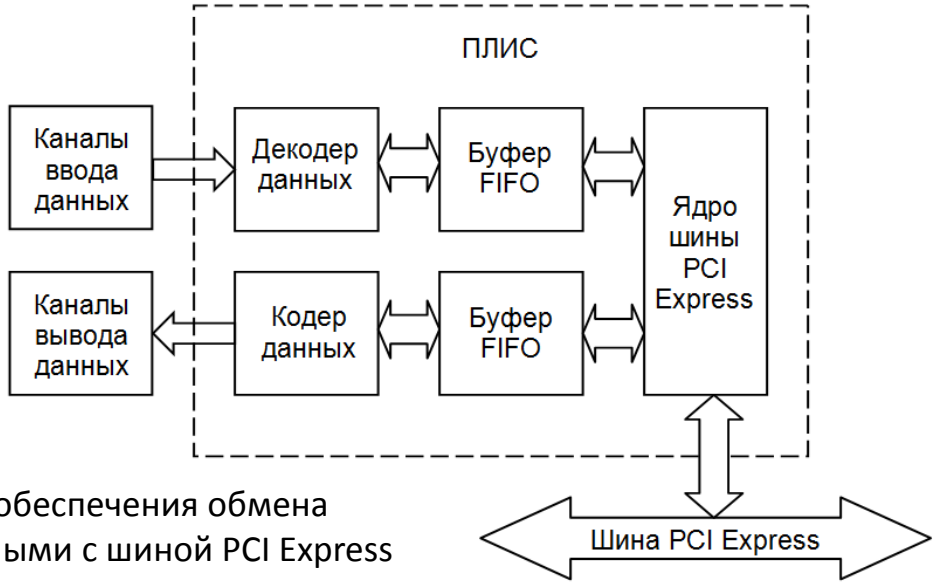
Разрядность входных потоков	*Частота тактового сигнала Fc,* [МГц]	*Разрядность интерфейса ввода X*	*Скорость потока V, Мбит/с*	*Распределенная пропускная способность для потока, $Fx*(X_M – X_{M-1})$, Мбит/с*
12	100	192	1200	191*Fx
1 20 3	300 5 250	36	30 100 750	157*Fx 8*Fx 86*Fx
32 12 3 1 1	76 46 67 60 20	57	2432 552 201 60 20	37*Fx 9*Fx 4*Fx 1*Fx 1*Fx
10 3 26 34 32 4 3 8 32	456 65 234 12 3 87 53 21 13	20	4560 195 6084 408 96 348 159 168 416	1*Fx 1*Fx 1*Fx 1*Fx 1*Fx 1*Fx 1*Fx 1*Fx 1*Fx
43 7 8	9 54 23	5	387 378 184	*Распределение невозможно т.к.* M>(X/2)

Примечание: *Fx* – частота тактового сигнала интерфейса ввода.

Рекомендации по адаптации ресурсов ПЛИС XC4VFX20



Модуль предназначен для обеспечения ввода/вывода цифровых сигналов по нескольким каналам.
Для загрузки конфигурации ПЛИС можно использовать разные методы. При первом методе для хранения структуры ПЛИС используется загрузочное ПЗУ, которое программируется заранее (до установки в модуль) или через специально предусмотренный интерфейс JTAG (после установки в модуль).
При другом методе загрузка конфигурации ПЛИС производится с помощью интерфейса Jam Player через шину PCI Ex-press непосредственно в ПЛИС, либо в загрузочное ПЗУ.
Тактовый генератор формирует тактовый сигнал, который обеспечивает синхронизацию работы логических блоков ПЛИС.



Для обеспечения обмена данными с шиной PCI Express в схеме содержится логическое ядро шины.
Оно реализует как физический интерфейс шины, так и протоколы обмена данными с шиной.
Буферы FIFO необходимы для согласования скоростей обмена данными между ядром шины и кодером и декодером данных. Эти буферы имеют емкость по 64 Кбайт и по две 64-разрядные шины для ввода и вывода данных.
Декодер данных осуществляет преобразование (распараллеливание) данных, поступающих от каналов ввода в виде 4, 8, 12, 16, 20, 24, 28 или 32 потоков, в 64 потока для буфера FIFO.
Кодер данных осуществляет преобразование (объединение) 64 потоков данных, поступающих от буфера FIFO в 4, 8, 12, 16, 20, 24, 28 или 32 потоков для каналов вывода.

Рекомендации по адаптации ресурсов ПЛИС XC4VFX20
Исходный вариант

```
architecture rpi of rpi is
signal mem              : std_logic_vector (128 downto
0):=(others=>'0');
begin
p1 : process(clk)
variable adr            : integer range 0 to 128 := 0 ;
begin
if(reset = '0') then
            adr:= 0;
            mem <=(others=>'0');
elsif(rising_edge(clk4) and start='1')then
case (adr) is
 when 64 =>             data_out<=mem(63 downto  0);
 when 128 =>data_out<=mem(127 downto  64);  adr:=0;
end case;
mem(adr+31 downto  adr)<= di8 & di7& di6& di5& di4& di3& di2&
di1;
adr:=adr + 32;

when others =>mem <= (others=>'0');
end if;
end process;
end rpi;
```

Этот фрагмент описывает процесс преобразования восьми входных четырехразрядных потоков di1, di2 … di8 в единый 64-разрядный поток data_out (с соответствующим уменьшением скорости передачи по каждому каналу). Следует заметить, что такое “прямое” решение приводит к созданию ресурсоемких сумматоров для вычисления значений переменных и регистров для хранения этих значений, например, при выполнении операций mem(adr+31 downto adr)<= di8 & di7& di6& di5& di4& di3& di2& di1 и adr:=adr + 32.

Рекомендации по адаптации ресурсов ПЛИС XC4VFX20
Оптимизированный вариант

```
architecture rpi of rpi is
signal mem0 : std_logic_vector(3 downto
0):=(others=>'0');
signal mem1                    :
std_logic_vector(3 downto
0):=(others=>'0');
…
signal mem31: std_logic_vector(3
downto 0):=(others=>'0');
begin
p1: process(clk)
variable adr  : integer range 0 to 33 := 0 ;
begin
if(reset = '0') then
                adr:= 0;
mem0 <=(others=>'0');
mem1 <=(others=>'0');
…
mem31 <=(others=>'0');
elsif(rising_edge(clk) and start='1') then
```

Задания прямых соединений между регистрами с использованием мультиплексоров.

```
case (adr) is
 when 0  =>   mem0 <=di1;mem1 <=di2; mem2 <=di3; mem3 <=di4; mem4
<=di5;mem5 <=di6; mem6 <=di7;mem7 <=di8;
when 1  =>   mem8 <=di1 ; mem9 <=di2;mem10 <=di3 ; mem11 <=di4;
mem12 <=di5;mem13 <=di6;mem14 <=di7;mem15 <=di8;
when 2 =>    mem16 <=di1 ;mem17 <=di2 ;mem18 <=di3; mem19
<=di4;mem20 <=di5;mem21 <=di6; mem22 <=di7; mem23 <=di8;
data_out (63 downto  0)<=
mem15&mem14&mem13&mem12&mem11&mem10&mem9&mem8&m
em7&mem6&mem5&mem4&mem3&mem2&mem1&mem0;
when 3 =>    mem24 <=di1 ;mem25 <=di2 ;mem26 <=di3 ;mem27
<=di4;mem28 <=di5;mem29 <=di6;mem30 <=di7; mem31 <=di8;
when 4 => data_out (63 downto  0)<=
mem31&mem30&mem29&mem28&mem27&mem26&mem25&mem24
&mem23&mem22&mem21&mem20&mem19&mem18&mem17&mem1
6;
mem0 <=di1;mem1 <=di2; mem2 <=di3; mem3 <=di4; mem4 <=di5;mem5
<=di6; mem6 <=di7;mem7 <=di8;       adr:=0;
end case;
adr:=adr + 1;
when others => mem0 <= (others=>'0') ;  mem1<= (others=>'0') ;
…
mem31<= (others=>'0') ;
end if;
end process;
end rpi;
```

Рекомендации по адаптации ресурсов ПЛИС XC4VFX20

Наименование ресурса ПЛИС	Всего имеется в ПЛИС	Вариант VHDL-описания			
		Исходный		Адаптированный	
		Затрата (шт.)	Затрата (%)	Затрата (шт.)	Затрата (%)
Количество секций	8544	19935	233	5453	63
Количество триггеров в секциях	17088	4047	23	3760	22
Количество 4-входовых таблиц преобразования	17088	37697	220	10013	58
Количество блоков ввода/вывода	320	136	42	181	56
Количество буферов FIFO16/RAM16	68	1	1	1	1
Количество глобальных тактовых линий GCLK	32	9	28	16	50
Количество буферов BUFR	16	2	12	2	12
Количество блоков управления синхрониза-цией DCM_ADV	4	1	25	1	25

В целом кодер и декодер данных с описанной структурой создают суммарную емкость прошивки, которая может не уложиться в объем одной ПЛИС, что во многих случаях недопустимо.

Семинар 2. Мультиплексоры

Процессная форма описания n-разрядного мультиплексора 2-1

VHDL

```
entity mux_2_1 is
generic (n:positive:=1);
 port (a,b:in std_logic_vector
   (n-1 downto 0);
    c:in std_logic;
   y:out std_logic_vector (0 to 2)
    );
end;
architecture behavior of mux_2_1 is
constant one : std_logic:='1';
begin
   process (a,b,c)
    begin
     if c=one then
        y<=a;
     else
        y<=b;
     end if;
  end process;
end behavior;
```

VERILOG

```
module mux_2_1(a, b, c, y );
    parameter n=1;

    input [n-1:0] a,b;
    input c;
    output [n-1:0]  y;
    reg [n-1:0]  y;

    always (a or b or c )
      begin
        if (c == 1'b1)
            y<=a;
        else
            y<=b;
      end // always

endmodule  // MUX_2_1
```

Пример 2. Потоковая форма описания n-разрядного мультиплексора 2-1 (входы a, b, выход y). В описании в «потоковом» (dataflow) стиле используют краткую форму записи процесса.

VHDL

```
architecture beh_short of mux2_1 is
 constant  one : std_logic:='1';
      begin
       y<= a when c=one else b;
end beh_short;
```

VERILOG

```
module mux_2_1_S(a,b,c,y);
 input c;parameter n=1;
 input[n-1:0]  a,b; output[n-1:0] y;
 assign  y =( c==1'b1)? a: b;
endmodule  // mux_2_1_S
```

Пример 3. Одноразрядный мультиплексор 4_1.

```
entity mux4 is
  port (c, d, e, f : in std_logic;
        s : in std_logic_vector
                  (1 downto 0);
       muxout : out std_logic );
end mux4;
architecture beh of mux4 is
begin
mux: process (s, c, d, e, f)
begin
  case s is
     when "00" => muxout <= c;
     when "01" => muxout <= d;
     when "10" => muxout <= e;
     when "11" => muxout <= f;
     when others => muxout <= 'X';
  end case;
end process mux;
end beh;
```

```
module mux4 (c,d,e,f,s,muxout);
    input c,d,e,f;

    input [1 :0]s; output muxout;
     reg muxout;


 always (s or c or  d or  e or  f)
     begin
     case (s)
         2'b00 : muxout <= c;
         2'b01 : muxout <= d;
         2'b10 : muxout <= e;
         2'b11: muxout <= f;
        default: muxout <= 1'bx;
       endcase
   end // always
endmodule //mux4
```

Пример 4. Описание n-разрядного мультиплексора 3-1 с поразрядным кодированием селектора (one hot). Несинтезабельная часть описания используется при

VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;
entity mux3oh is

generic(size :integer:=1;
check:std_logic:='1');
port( o:out std_logic_vector
        (size-1 downto 0);
   d0,d1,d2:in std_logic_vector
        (size-1 downto 0);
  s0, s1, s2: in std_logic);
end;
architecture beh of mux3oh is
constant one :std_logic:='1';
signal onehotcheck:std_ulogic;
signal tmp0,tmp1,tmp2:
std_logic_vector(size-1 downto 0);
begin
tmp0,tmp1,tmp2;
process (d0,d1,d2,s0,s1,s2)
 begin
  if s0=one then tmp0 <= d0;
   else tmp0 <= (others=>'X');
  end if;
  if s1=one then tmp1 <= d1;
       else tmp1 <= (others=>'X');
        end if;
  if s2=one then  tmp2 <= d2;
   else tmp2 <= (others=>'X');
   end if;

-- one hot контроль ниже  --

-- synopsys translate_off
 onehotcheck<= (s0 and s1) or
    (s0 and s2) or (s1 and s2);
 -- synopsys translate_on
 end process;
process(tmp0, tmp1, tmp2,
  onehotcheck)

 begin
 -- synopsys translate_off
  if (check =one and
     onehotcheck='1'
      ) then
        o <= (others=>'X');
  else
  -- synopsys translate_on
     o <= tmp0 or tmp1 or tmp2;
    end if;
end process;
end;
```

VERILOG

```
    module mux3oh(o, d0, d1, d2,  s0,
                                  s1,s2);

     parameter size = 1, check = 1;

    output [size-1:0] o; reg [size-1:0] o;
    input [size-1:0] d0, d1, d2;

    input s0, s1, s2;

                                  reg [size-1:0]

    always @(d0 or d1 or d2 or s0 or s1 or s2).
        begin


    if(s0) tmp0 = d0; else tmp0 = 0;


    if(s1) tmp1 = d1; else tmp1 = 0;


     if(s2) tmp2 = d2; else tmp2 = 0;
       end
   // one hot check
      wire  onehotcheck;
  // synopsys translate_off
     assign onehotcheck=(s0 & s1) |
     (s0 &  s1) |(s1 & s2);
   // synopsys translate_on


    always @(tmp0 or tmp1 or
          tmp2 or onehotcheck)
      begin
   // synopsys translate_off
      if (check == 1 &&
        onehotcheck == 1)

        o <= {size{1'bx}};
        else
   // synopsys translate_on
         o <= tmp0 | tmp1 | tmp2;
       end
endmodule
```

Семинар 2-3. Мультиплексоры

Пример 5. Структурное VERILOG-описание 8-входового n-разрядного мультиплексора, построенного из 4- и 2-разрядных.

```
module mux8n(yo, d0, d1, d2, d3, d4, d5, d6, d7, s);
parameter size = 1;
output [size-1:0] yo;
input [size-1:0] d0, d1, d2, d3, d4, d5, d6, d7;
input [2:0] s;
   wire [size-1:0] t0;
   mux4n #(size) i0(t0, d0, d1, d2, d3, {s[1], s[0]});   wire [size-1:0] t1;
   mux4n #(size) i1(t1, d4, d5, d6, d7, {s[1], s[0]});   wire [size-1:0] t2;
   mux2n #(size) i2(t2, t0, t1, s[2]);
   assign yo = t2;
endmodule
```

Пример 6. Описание 2^n-входового 1-разрядного мультиплексора.

VHDL

```
entity mux8_1 is
port (a,b,c,d,e,f: in std_logic;
        sel: in std_logic_vector(2 downto 0);
        mux_out: out std_logic);

end;
architecture mux_infer of mux8_1 is
 begin
   process (a,b,c,d,e,f,sel)
    begin
       case (sel) is--synopsys infer_mux
         when "000" =>mux_out<= a  and b,
         when "001" =>mux_out<= a  or c;
         when "010" =>mux_out<= d  or e;
         when others =>mux_out<= d  xor f;
      end case
    end process;
 end;
```

VERILOG

```
module mux8_1
   (a,b,c,d,e,f,sel,mux_out);
  input a,b,c,d,e,f;
  input [2:0]sel;
  output mux_out;
  reg mux_out;

 //VERILOG-2000
 always @( a,b,c,d,e,f,sel)

  case (sel) //synopsys infer_mux
     3'b000: mux_out=a & b;
     3'b001: mux_out=a | b;
     3'b010: mux_out=d | e;
     default: mux_out=d ^ f;
  endcase

endmodule
```

VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity mux_n is
 generic(n:integer:=5);
  port (I_bus : in std_logic_vector
                    (2** n-1 downto 0);
          s : in std_logic_vector
                     (n-1 downto 0);
       muxout : out std_logic );
end;
architecture beh of mux_n is
begin
mux: muxout<=I_bus((conv_integer( s)));
end;
```

VERILOG

```
module mux_n (I_bus,s,muxout);
   parameter n=5;

  input[1 <<n -1:0] I_bus;

  input [n-1 :0]s;
  output muxout;


   assign muxout=I_bus[s] ;
endmodule
```

Семинар 3. Дешифраторы

Пример 1. Простой дешифратор 2-4.

VHDL

```
entity decod2_4   is
 port(a:in std_logic_vector
                (1 downto 0);
      sel:out std_logic_vector
                (3 downto 0);
end decod2_4   ;
architecture  synt of decod2_4 is
begin
process (a)
  begin
   case (a) is
    when "00" => sel<="0001";
    when "01" => sel<="0010";
    when "10" => sel<="0100";
    when others => sel<="1000'
   end case;
end process; end;
```

VERILOG

```
module vdecod2_4(a,sel);

 input [1:0]a;

output [3:0]sel;
reg[3:0] sel


always @(a)

case(a)
    2'b00: sel=4'b0001;
    2'b01: sel=4'b0010;
    2'b10: sel=4'b0100;
    default: sel=4'b1000;
 endcase
endmodule
```

Дешифратор 3-8.

VHDL

```
library ieee;
use ieee.numeric_std.all;
entity decoder3_8 is
  port (
    a : in std_logic_vector(2 downto 0)
    q : out std_logic_vector(7 downto (
end decoder3_8;

architecture behavior of decoder3_8 is
   signal n_a : unsigned(2 downto 0);
   signal n_q : unsigned(7 downto 0);
begin
  n_a <= unsigned(a);
  process (n_a)
   variable i: integer;
  begin

    for i in 7 downto 0 loop
      if (n_a=i) then
        n_q(i) <= '1';
      else
        n_q(i) <= '0';
      end if;
    end loop;
  end process;
  q <= std_logic_vector(n_q);
end behavior;
```

Пример 2. Энергосберегающий дешифратор 2-4 с сигналом разрешения.

Такой дешифратор активизируется только в те моменты, когда его выход задействован, т. е. подан сигнал разрешения en, что полезно в смысле энергосбережения.

VHDL

```
entity decod2_4en   is

    port(en: std_logic;
         a:in std_logic_vector
                (1 downto 0);
        sel:out std_logic_vector
          (3 downto 0));
 end decod2_4en   ;
architecture  synt of decod2_4en is
 begin
  process (en,a)
  variable xx: std_logic_vector
              (2 downto 0);
 begin
   xx:= (en & a);
   case xx is
    when "100" => sel<="0001" ;
    when "101" => sel<="0010" ;
    when "110" => sel<="0100";
    when "111" => sel<="1000" ;
   when others => sel<= "0000";
  end case;
 end process;
end;
```

VERILOG

```
module  decod2_4en
            (en,a,sel);
    input en;

     input [1:0]a;

     output [3:0]sel;
     reg[3:0] sel;


     always @(en or a)


 begin

  case({en,a})
        3'b100: sel=4'b0001;
        3'b101: sel=4'b0010;
        3'b110: sel=4'b0100;
        3'b111: sel=4'b1000;
        default :sel=4'b0000;
   endcase
end
endmodule
```

Дешифратор 3-8.

VERILOG

```
module decoder3_8 (a,q);

      input [2:0]a;
      output [7:0]q;
            reg[7:0]q;
      integer i;

always @(a)

for( i=7;i>=0;i=i-1)begin
if(a==i)

  q[i] <= 1;
else
  q[i] <= 0;

end

endmodule
```

Семинар 3. Регистры, счетчики

Пример 3. n-разрядный регистр с асинхронным сбросом и разрешением/приема

VHDL

```
entity regenn is

  generic (size:positive:=32);
  port ( rout: out std_logic_vector
              (size-1 downto 0);
         rin : in std-logic_vector
              (size-1 downto 0);
        en,clr_n,clk: in std_logic);
  end;
  architecture beh of regenn is

 begin
   process(clk,clr_n) begin
     if (clr_n ='0' ) then
         rout <= (others=>'0');
    elsif clk'event and clk='1' then
         if  en='1' then
     rout <= rin after 1 ns;
        end if;
   end if;
 end process;
 end;
```

VERILOG

```
module regenn(rout, rin,
                    en, clr_n, clk);
  parameter size = 32;

  output [size-1:0] rout;
  reg [size-1:0] rout;
  input [size-1:0]  rin;
  input en,clr_n,clk;

  always @(posedge clk or
          negedge clr_n)

  begin
     if (!clr_n)
           rout <= 0;
     else
           if (en)
            rout <= #(1) rin;

  end

endmodule
```

Пример 6. n-разрядный счетчик с асинхронным сбросом

VHDL

```
 library IEEE; use IEEE.std_logic_1164.all;
 entity count_up is
generic(n: integer:=7);
port (reset,clk: in std_logic;
     count:out  std_logic_vector(n-1 downto 0));
end;
library IEEE; use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all; use IEEE.std_logic_unsigned.all;
architecture beh of count_up is
 begin
 process (reset,clk)
variable count_tmp:  integer;
begin
    if reset ='1' then     count_tmp :=0 ;
    elsif clk'event and clk='1' then
              count_tmp:= count_tmp+1;
    end if;
    count<=conv_std_logic_vector( count_tmp,n) after 1 ns;
 end process;
 end;
```

VERILOG

```
module count_up(reset,clk,count);
parameter n=7;
 input reset,clk;  output [n-1:0] count;
reg [n-1:0] count;
always @(posedge reset or  posedge clk )
 begin
          if (reset==1)begin
            count=0;
          end
          else begin
            count=count+1;
          end
     end
endmodule
```

Пример 4. Триггер-защелка (регистр)

VHDL

```
entity  rlatchn     is
generic(size: integer:=32);
port( rout:out std_logic_vector
          (size-1 downto 0);
       rin:in std_logic_vector
          (size-1 downto 0);
       en: in std_logic);
end;
architecture beh of rlatchn is
begin
  process (en,rin)
   begin
   if en ='1' then
       rout <=  rin after 1 ns;

  end if;
end process;
end;
```

VERILOG

```
module rlatchn(rout,rin,en);
 parameter size = 32;

 output [size-1:0] rout;

 input [size-1:0] rin;
 input en;

 reg [size-1:0] rout;

always @(en or rin)
 begin
    if (en)

      rout <= #1 rin;
 end
endmodule
```

Пример 5. Образ 4-разрядного сдвигового регистра с последовательным вхо-
ом и выходом (в комментариях предполагаемые описания входов-выходов).

VHDL

```
--        CLK,din: in STD_LOGIC;
--        DOUT: out STD_LOGIC;

process (clk)
   variable r_s: std_logic_vector
              (3 downto 0);
begin
   if clk'event and clk='1' then
      r_s := din & r_s(3 downto 1);
   end if;
   dout <= r_s(0) after 1 ns;
end process;
```

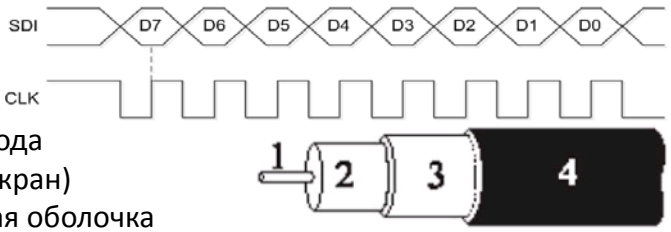
VERILOG

```
     //input clk,din;
     //output dout;reg dout;
     //reg [3:0] r_s;
always @(posedge clk)
begin


 if (clk==1'b1)
   r_s= {din,r_s[3:0]};

   dout<= #1 r_s[0];
end //always
```

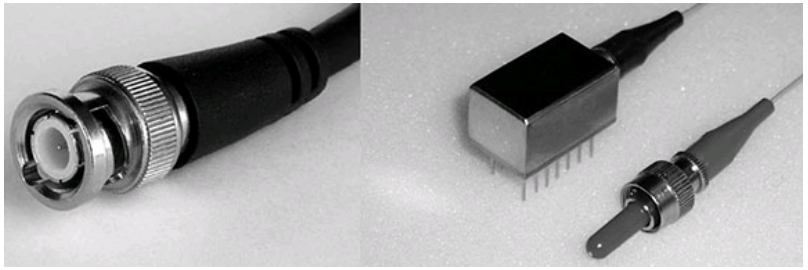
Селекция кадрового синхроимпульса. Интерфейсы

Передача видео (SDI), звука (opt 5.1), приём от спутниковой антенны, ethernet.

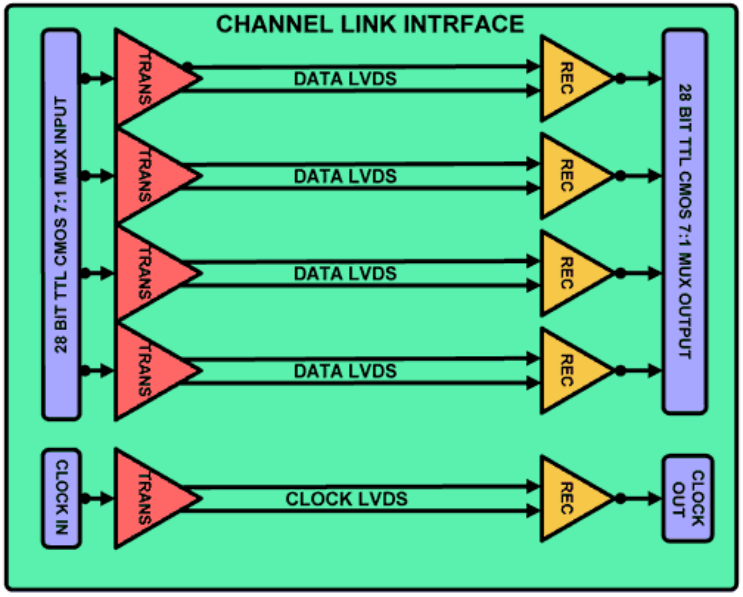
Интерфейс SDI используется для передачи некомпрессированных и некодированных цифровых видеосигналов в профессиональном телевизионном оборудовании. Передача потока данных 270 Мбит/с возможна на расстояние до 300 м по коаксиальному кабелю. Интерфейс является самосинхронизируемым. Кадровая синхронизация осуществляется специальным синхронизирующим пакетом данных. Видеопоток передается по однопроводной цифровой линии.

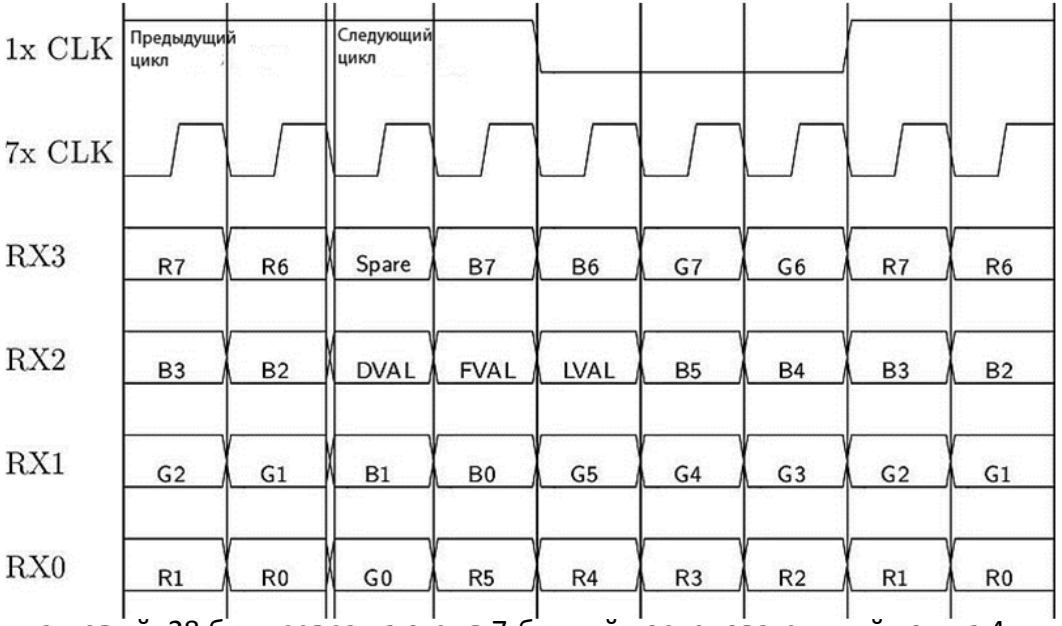


центральный провод (жила)
изолятор центрального провода
экранирующий проводник (экран)
внешний изолятор и защитная оболочка



Интерфейс CameraLink разработан для высокоскоростных видеосистем, таких как камеры и фрэймграбберы. Каждый информационный сигнал сопровождается двумя тактовыми сигналами и обеспечивает достоверность селекции кадрового синхроимпульса.





Сериалайзер получает 28 одиночных сигналов данных и один тактовый. 28 бит превращаются в 7-битный последовательный код на 4 вых. шинах. Затем 4 потока данных и один тактовый поток отправляются по 5 парам LVDS. Десериалайзер принимает 4 потока данных и один тактовый LVDS, а затем "разворачивает" его в 28 бит и тактовый сигнал и передаёт по параллельной выходной шине на плату.

Селектор кадрового синхроимпульса на ПЛИС

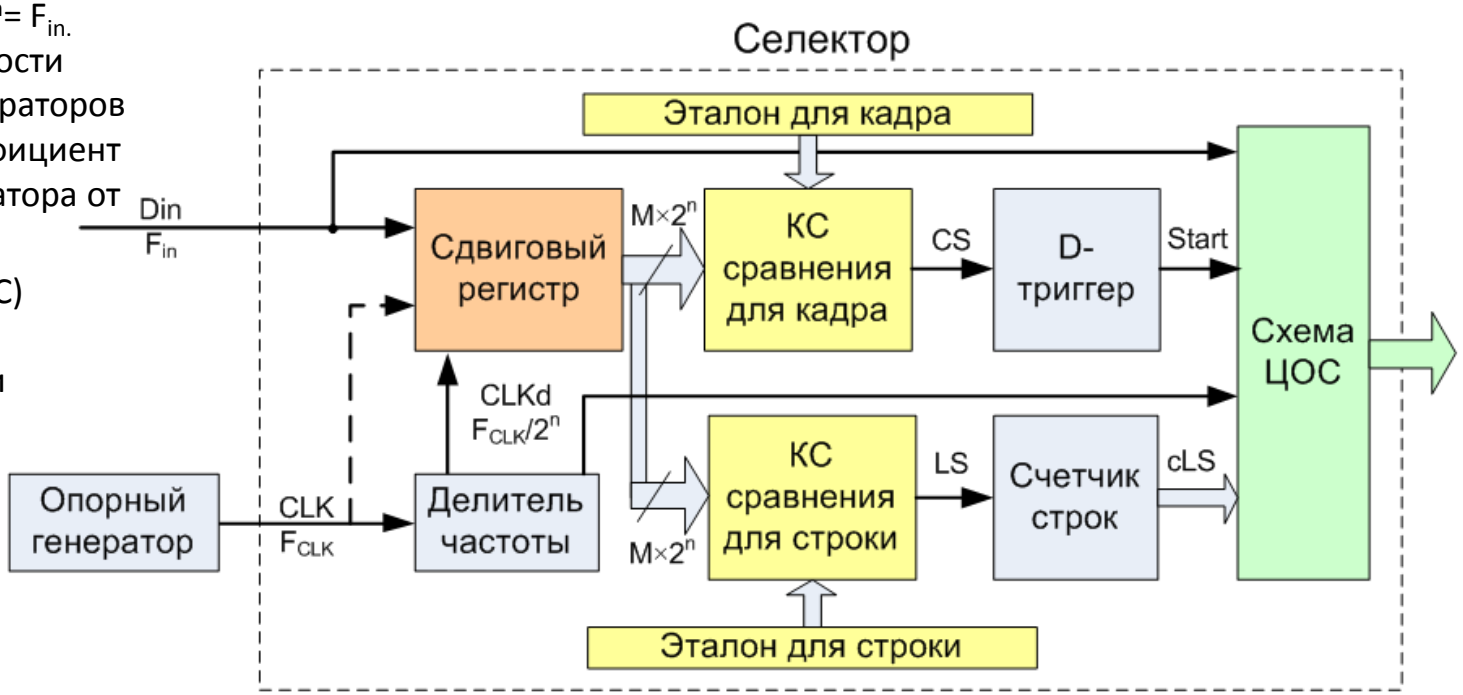
Для регистрации и дальнейшей обработки видео, передаваемых по однонаправленным линиям, необходима **кадровая и строчная селекции**. Устройство, определяющее метку начала кадра (и строки) и формирующее сигнал старта для последующих процессов обработки, называется **кадровым селектором**.

Поток видео состоит из кадров. **Кадр содержит**: кадровый CS синхроимпульс из *M* (×2^n) бит, биты пикселей, строчные LS синхроимпульсы.

Входной цифровой поток Din частотой F_{in} поступает на вход сдвигового регистра разрядностью $M \times 2^n$ с тактовой частотой CLKd. Опорный генератор формирует тактовый сигнал CLK частотой F_{CLK} . Делитель частоты выдает сигнал CLK с частотой $F_{CLK}/2^n$, где 2^n– требуемая степень деления.

В идеальном случае $F_{CLK}/2^n = F_{in}$. Однако учитывая погрешности частоты выпускаемых генераторов $F_{CLK}/2^n = k \times F_{in}$, где *k* – коэффициент отклонения частоты генератора от номинального значения.

Комбинационная схема (КС) сравнения формирует импульс совпадения метки начала кадра или строки с эталоном, который преобразуется в сигнал запуска (CSt и LSt) последующих процессов обработки в схеме ЦОС с помощью D-триггера.



Обязательное условие работы селектора, то есть частота $\mathbf{F_{CLK}}$ генератора должна быть в 2^n больше входной частоты F_{in}:

$F_{CLK}/2^n = k \times Fin$, где n = 0, 1, 2 ….

Селекция от однонаправленной одноразрядной цифровой линии

Обязательное условие работы селектора - частота F_{clk} должна быть в 2^n больше входной частоты F_{in}: $F_{CLK}/2^n = k \times F_{in}$, где n = 0, 1, 2 ….

Каждый бит потока Din может кодироваться 2^n битами в $M \times 2^n$ разрядном сдвиговом регистре, коэффициент отклонения частоты *k* определяется выражением:

$$\frac{M-1}{M} < k < \frac{M+1}{M}$$

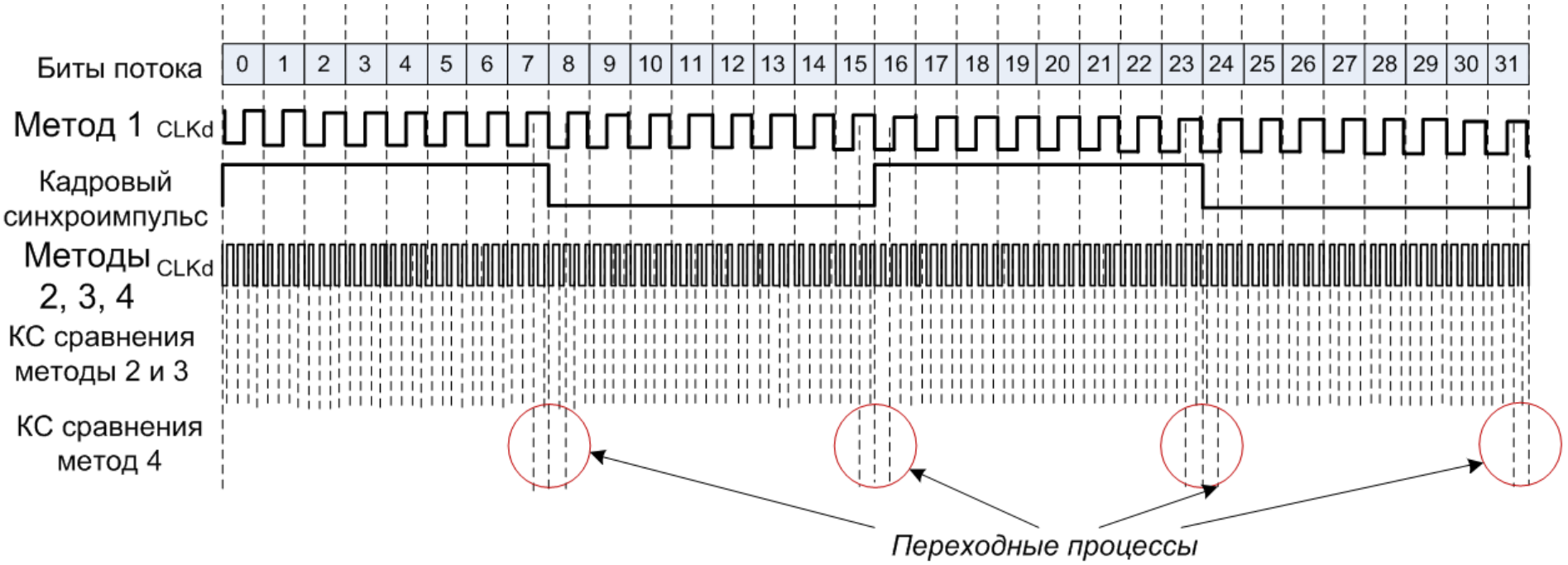
При невыполнении этого условия число бит в метке кадра или строки не будет соответствовать и селекция не сработает или сработает на иную кодовую комбинацию.

Предлагается 4 метода селекции:

1) селекция на частоте потока k×Fin;
2) селекция на частоте k× Fin×2^n в 2^n раз большей частоты потока с жесткой привязкой к эталону;
3) селекция на частоте k× Fin×2^n в2^n раз большей частоты потока с гибкой привязкой к эталону;
4) селекция на частоте k× Fin×2^n в 2^n раз большей частоты потока с учетом формы и переходных процессов кадрового синхроимпульса.

Селекция от одноразрядной цифровой линии. Методы

Здесь кадровый синхроимпульс состоит из 32 битов и сопровождается тактовым сигналом CLKd. Кадровый синхроимпульс представляет собой последовательность из 8“1”,8“0”,8“1”,8“0”.



В методе 1 частота селекции F_{CLKd} = Fin частоте битового потока. Разрядность сдвигового регистра =M.
В методе 2 частота селекции F_{CLKd} = Fin×2^n (n=2) - больше частоты Fin потока Din и сравнение с эталоном проводится на каждом такте. Разрядность сдвигового регистра равна M×2^n.
В методе 3 частота селекции F_{CLKd} = Fin×2^n (n=2) - больше частоты Fin потока Din, но сравнение каждого бита потока Din с эталоном проводится по *соответствующим* 2-му и 3-му битам сдвигового регистра разрядностью M×2^n=M×4, 1-й и 4-й разряды не учитываются.
В методе 4 частота селекции F_{CLKd} = Fin×2^n (n=2) - больше частоты Fin потока Din, однако сравнение с эталоном проводится с учетом переходных процессов кадрового синхроимпульса (биты 7, 8, 15, 16, 23, 24 и 31).

Селекция от однонаправленной одноразрядной цифровой линии

Сравнительный анализ методов по числу переходных процессов и коэффициента отклонения частоты селекции

Метод	Число переходных процессов	*k*
1	0	33/32=1,03125
2	1	129/128=1,0078125
3	32	130/128=1,015625
4	4	130/128=1,015625
Генератор FTX16	–	16МГц±30Гц=1,000001875

Метод 1 прост в реализации, но имеет большой *k* и большую вероятность сбоя при несовпадении, как частоты, так и фазы сигнала. **Метод 2.** Селекция кадра на частоте 2^n большей частоты потока с жесткой привязкой к эталону уже при n=2 и более дает положительные результаты, однако метод ограничен коэффициентом отклонения частоты *k*.

Метод 3 и Метод 4. Селекция кадра на частоте 2^n большей частоты потока с гибкой привязкой к эталону и селекция с учетом переходных процессов позволяет достичь хорошего значения коэффициента *k*.

Опорные генераторы частоты F_{CLK} не идеальны и имеют свой собственный коэффициент отклонения. Например, генератор FTX16 компании Fronter имеет отклонение частоты 30 Гц (*k* = 1,000001875). Такое отклонение значительно меньше методов 2, 3, 4 ,что не влияет на процесс селекции.

все методы кроме а) способны обеспечить стабильную кадровую селекцию в реальных условиях с применением генератора на 16МГц±30Гц.

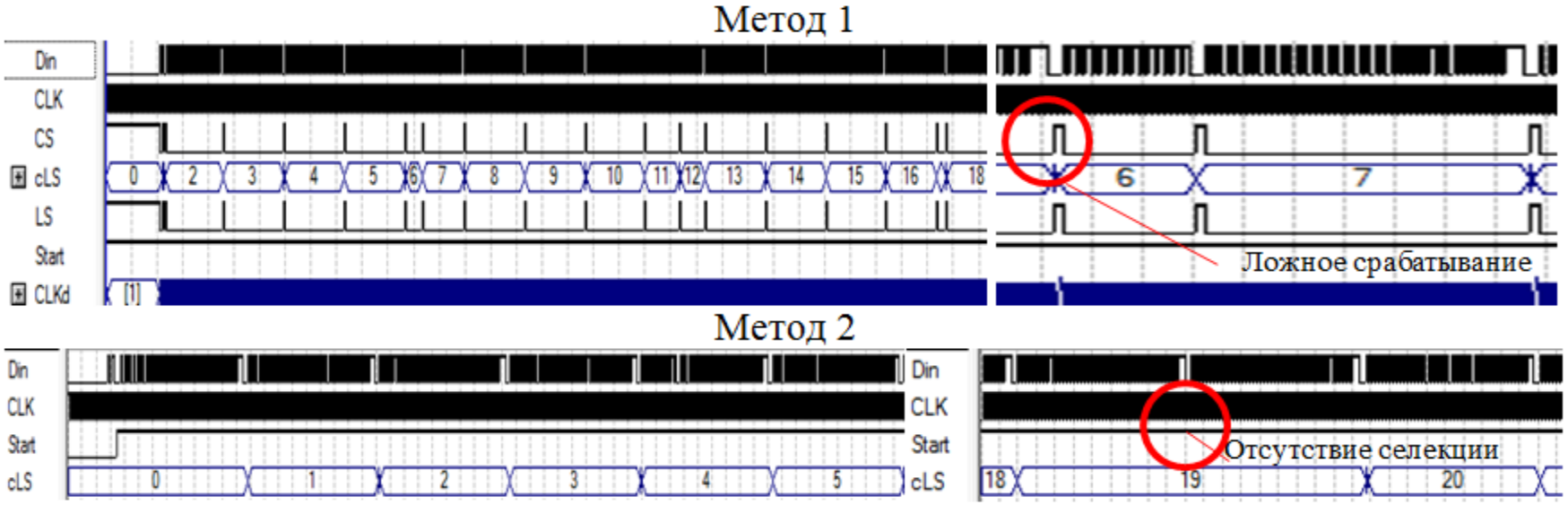
Методы селекции кадрового синхроимпульса на ПЛИС

Метод1

```
reg    [31:0]S; //сдвиговый регистр
always @(posedge CLK) begin S <= {S[30:0],Din}; end
```

Метод 2

```
reg [127:0]S; //сдвиговый регистр
always @(posedge CLK) begin S <= {S [126:0],Din}; end
reg [4:0]CLKd; reg Start; reg [5:0]cLS;
initial  Start =0; //initial LS=0; always @(negedge CLK)
begin if (S==128'b11111111111111111111111111111111000000000000000000000000000001111111111111111111111111111111000000000000000000000000000000) begin Start<=1;end
if(S==128'b11111111111111111111111111111111111111111111111111111111111111110000000000000000000000000000000000000000000000000000000000000000 & Start <=1) begin cLS<=cLS+1; end end
```



Метод 1 обеспечивает селекцию кадровых и строчных синхроимпульсов с частыми ложными срабатываниями. Такое решение является недопустимым для передачи видеопотока даже низкого качества.

Метод 2 в принципе позволяет осуществлять селекцию синхроимпульсов, но имеется высокая вероятность отсутствия селекции при несовпадении фаз тактового сигнала CLK и входных данных Din.

Методы селекции кадрового синхроимпульса на ПЛИС

Метод3

wireCS; // *КС сравнения для кадра*

assignCS=(~S[0]|~S[1]|~S[2]|~S[3])&(~S[4]|~S[5]|~S[6]|~S[7])&…&(~S[28]|~S[29]|~S[30]|~S[31])&(S[32]|S[33]|S[34]|S[35])&(S[36]|S[37]|S[38]|S[39])&…&(S[60]|S[61]|S[62]|S[63])&(~S[64]|~S[65]|~S[66]|~S[67])&(~S[68]|~S[69]|~S[70]|~S[71])&…&(~S[92]|~S[93]|~S[94]|~S[95])&(S[96]|S[97]|S[98]|S[99])&(S[100]|S[101]|S[102]|S[103])&…&(S[124]|S[125]|S[126]|S[127]);

Недостатки – высока вероятность захвата данных от пикселей в потоке

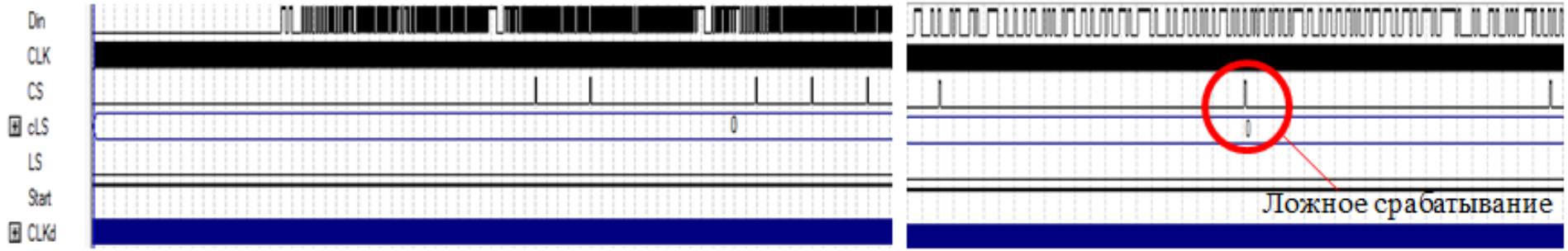
Метод 4 (жирным шрифтом отмечены переходные процессы в кадровом и строчном синхросигнале)

wireCS; assign // *КС сравнения для кадра*

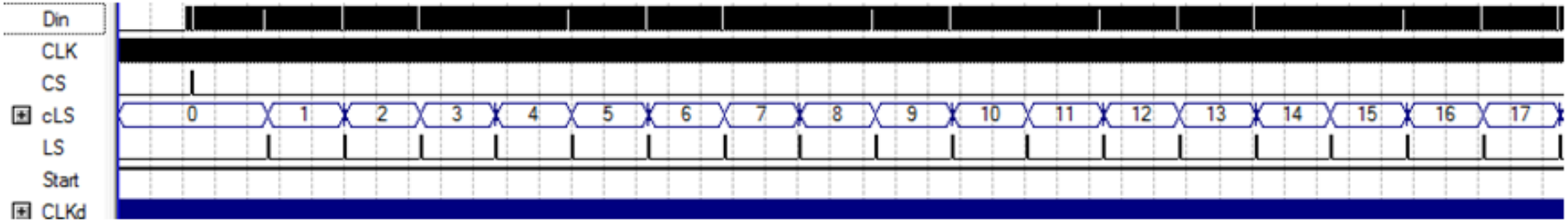
CS=**(~S[0]|~S[1])**&~S[2]&~S[3]&…&~S[29]&**(~S[30]|~S[31])&(S[32]|S[33])**&S[34]&S[35]&…&S[61]&**(S[62]|S[63])&(~S[64]|~S[65])**&~S[66]&~S[67]&~S[68]&~S[69]&…&~S[93]&**(~S[94]|~S[95])&(S[96]|S[97])**&S[98]&S[99]&…&S[127];

wire LS; assign // *КС сравнения для строки* LS=**(~S[0]|~S[1])**&~S[2]&~S[3]&…&~S[61]&**(~S[62]|~S[63])&(S[64]|S[65])**&S[66]& …&S[127];

Метод 3



Метод 4



Метод 3 имеет большое количество ложных срабатываний из-за высокой вероятности корреляции данных с эталоном синхроимпульсов. **Метод 4** исключает недостатки методов 1, 2, 3 по средствам отслеживания переходных процессов в синхроимпульсах, и обеспечивает наибольшую вероятность селекции.

Таким образом, селекция кадра на частоте 2^n меньшей частоты потока с гибкой привязкой к эталону имеет наибольшее отклонение частоты.

Синтез частоты

Синтезатор частоты - электронное устройство, способное из опорной частоты получать на выходе требуемую частоту или набор частот, согласно управляющим сигналам.

Методы синтеза частот*:*

• прямой аналоговый синтез (Direct Analog Synthesis, DAS) на основе структуры смеситель/фильтр/делитель, когда выходная частота получается непосредственно из опорной частоты посредством операций смешения, фильтрации, умножения и деления;

• косвенный (indirect) синтез на основе фазовой подстройки частоты (Phase Locked Loop, PLL), когда выходная частота получается с помощью дополнительного генератора (Voltage Controlled Oscillator, VCO), который охвачен петлей фазовой автоподстройки;

• прямой цифровой синтез (Direct Digital Synthesis, DDS), когда выходной сигнал синтезируется цифровыми методами;

• гибридный синтез, комбинация нескольких методов.

Параметры, характеризующие качество синтезатора частоты:

• чистота спектра выходного сигнала (уровень побочных компонентов и уровень шума);

• диапазон перестройки (полоса частот выходного сигнала);

• скорость перестройки;

• частотное разрешение;

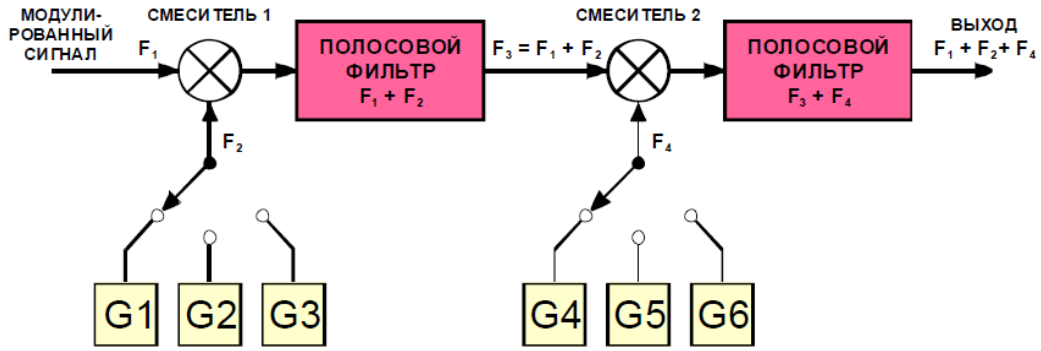
• количество разных генерируемых частот;

• гибкость (возможность осуществления различных видов модуляции);

• неразрывность фазы выходного сигнала при перестройке.

Прямой аналоговый синтез (DAS)

Качество выходного сигнала напрямую связано с качеством опорного. Низкий фазовый шум вследствие прямого синтеза. Быстрая перестройка по частоте. Неразрывность фазы при перестройке.

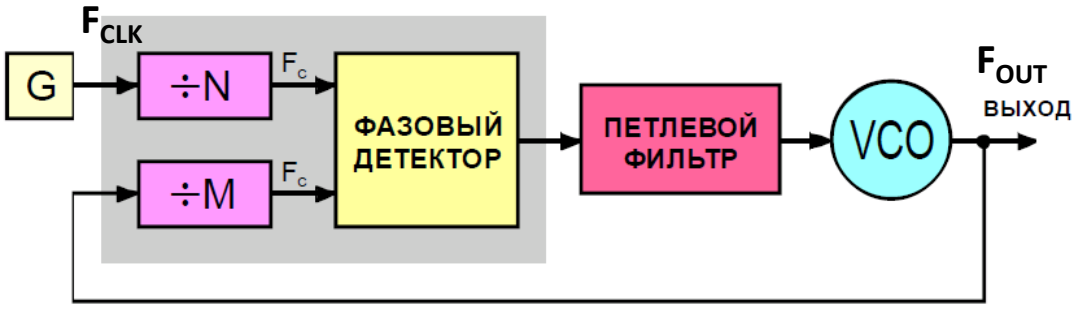


Для перестройки по F используется переключаемый банк опорных генераторов G1-G6. Подходит для радиостанций с небольшим количеством каналов. Но для обеспечения широких возможностей по

Используя делители частоты (структура смеситель/фильтр/делитель) можно уменьшить количество необходимых опорных генераторов.

Косвенный синтез частоты на основе фазовой автоподстройки (PLL)

Сравнение частоты и фазы выходного сигнала, источником которого служит генератор, управляемый напряжением (VCO), с сигналом опорного генератора. Обнаружение ошибки осуществляется с помощью фазового детектора, который работает на определенной частоте F_C, называемой частотой сравнения, которая получается путем деления на N частоты опорного генератора G. Частота выходного сигнала вначале делится на M, а потом сравнивается с частотой F_C. Если частота отклоняется, обнаруженная ошибка вызывает изменение управляющего напряжения VCO, что приводит к уменьшению отклонения.



Поскольку делители имеют целочисленные коэффициенты деления, шаг сетки такого синтезатора определяет частота сравнения.

Выходная частота определяется по формуле: $\mathbf{F_{OUT} = F_C \cdot M = (F_{CLK}/N) \cdot M = F_{CLK} \cdot (N/M),}$ где F_{OUT} – выходная частота, F_C – частота сравнения, N – коэффициент деления опорной частоты, M – коэффициент деления выходной частоты.

PLL синтезатор умножает опорную частоту в N/M раз. Коэффициенты N и M могут задаваться микроконтроллером.

Фазовый детектор является источником дополнительных фазовых шумов. Уменьшение шага сетки увеличивает фазовые шумы.

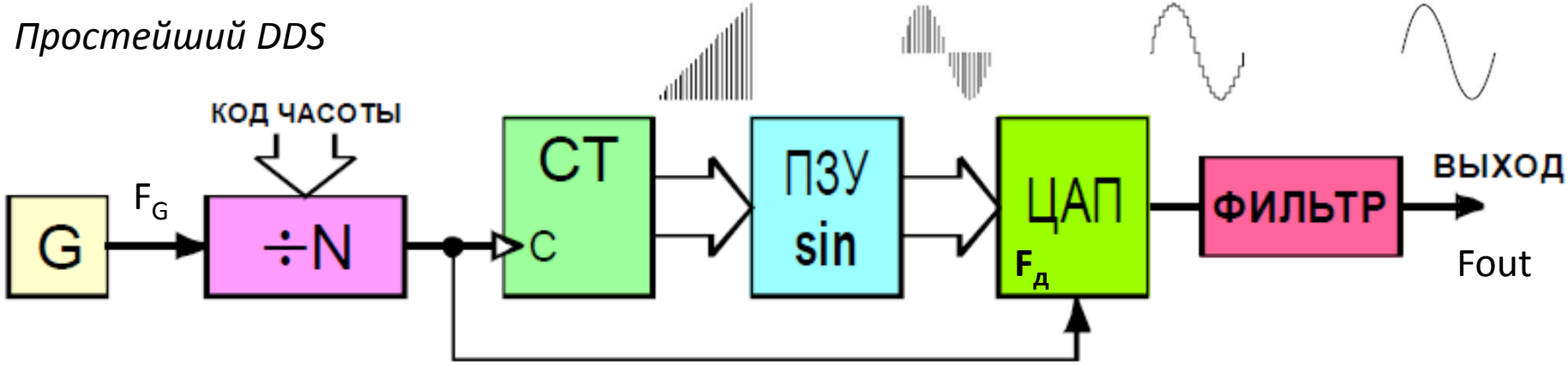
Медленное переключение F. Для получения малого шага перестройки по F объединяют в одном синтезаторе несколько петель PLL.

Многопетлевой PLL синтезатор является весьма дорогим и громоздким устройством.

Прямой цифровой синтез. Структура простейшего DDS

Задача DDS – получить на выходе сигнал синусоидальной формы с заданной **Fout**. На вход ЦАП необходимо подать последовательность отсчетов функции sin, следующих с частотой дискретизации $\mathbf{F_д}$. Закон изменения функции sin во времени реализуется полиномиально на АЛУ с низким быстродействием. Скоростной вариант - перекодировочная таблица (**Look Up Table)** в ПЗУ (таблица одного периода функции *sin*). Код, подаваемый на адресные входы ПЗУ является аргументом ф-ции sin, а выходной код ПЗУ равен значению функции для данного аргумента. Аргумент функции sin, или фаза, в отличие от значения функции, меняется во времени линейно двоичным счетчиком **СТ**. СТ формирует адрес для ПЗУ, отсчеты с выхода ПЗУ поступают на ЦАП, который формирует на выходе sin сигнал, подвергающийся фильтрации в ФНЧ и поступающий на выход. Для перестройки *Fвых* используется делитель частоты на N с переменным коэффициентом деления, на вход которого поступает тактовый сигнал с опорного генератора G.

Простейший DDS

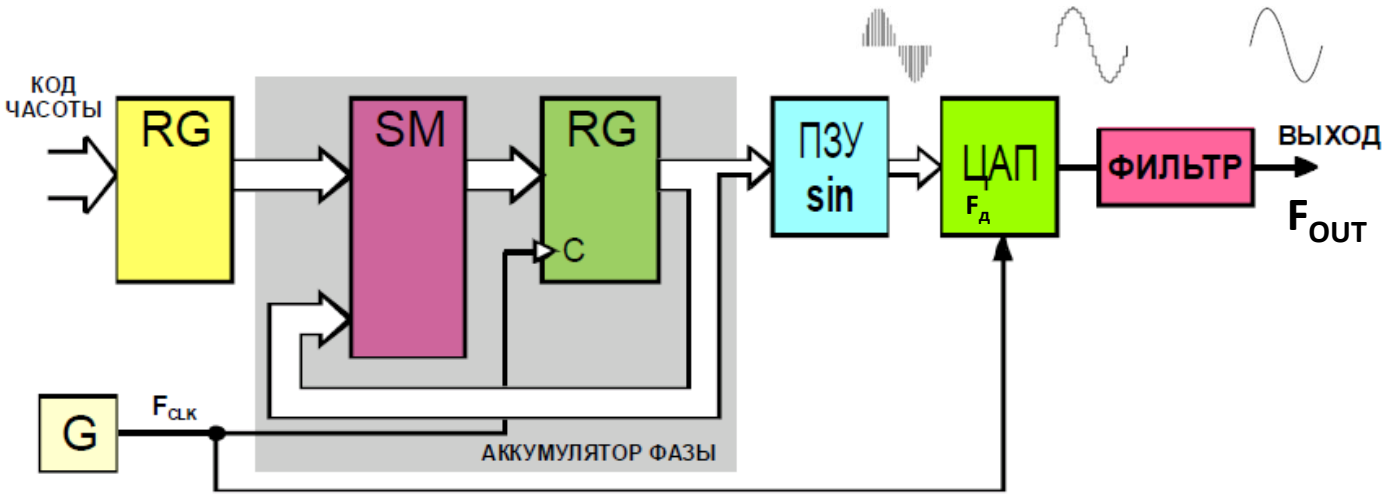


Недостатки структуры:

1. Неудовлетворительная способность к перестройке по частоте, поскольку F_G испытывает деление на целое число, шаг перестройки будет переменным, причем, чем меньше коэффициент деления N, тем больше относительная величина шага. Шаг будет недопустимо грубым при малых N.
2. При перестройке *Fвых* будет меняться $\mathbf{F_д}$. Это затрудняет фильтрацию выходного сигнала.
3. Неоптимальное использование скоростных характеристик ЦАП, – они будут в полной мере использованы лишь на максимальной выходной частоте. Гораздо логичнее всегда, независимо от выходной частоты, работать на постоянной $\mathbf{F_д}$, близкой к максимальной для используемого ЦАП.

Прямой цифровой синтез. DDS на основе накапливающего сумматора

Адресный счетчик СТ ПЗУ заменен накапливающим сумматором **SM** (регистр **RG**, который в каждом такте работы устройства перезагружается величиной, равной старому содержимому, плюс некоторая постоянная добавка). Как и для **СТ**, содержимое **RG** линейно увеличивается во времени, но приращение не всегда является единичным, а зависит от величины постоянной добавки. Когда накапливающий **SM** используется для формирования кода фазы - явл. аккумулятором фазы (**АФ**). Выходной код **АФ** - код мгновенной фазы выходного сигнала. Постоянная добавка, которая используется при работе **АФ** – явл. приращением фазы за один такт работы устройства. Чем быстрее изменяется фаза во времени, тем больше частота генерируемого сигнала. Значение приращения фазы фактически является кодом $\mathbf{F_{OUT}}$. Если приращение фазы равно 1, то поведение накапливающего сумматора ничем не отличается от поведения двоичного **СТ**. Если приращение фазы будет равно 2, то код фазы будет изменяться вдвое быстрее. При этом на ЦАП коды будут поступать с той же частотой, но они будут представлять собой не соседние отсчеты функции sin, а взятые через один. $\mathbf{F_{OUT}}$ при этом будет вдвое большей, а $\mathbf{F_д}$ останется прежней.



Аккумулятор фазы работает с периодическими переполнениями, обеспечивая арифметику по модулю 2N, соответствующее поведению функции sin с периодом 2π. Частота переполнений **АФ** равна $\mathbf{F_{OUT}}$ и определяется формулой:

$$\mathbf{F_{OUT} = M \cdot F_{CLK}/2N}$$

где $\mathbf{F_{OUT}}$ – выходная частота, $\mathbf{F_{CLK}}$ – тактовая частота, **M** – код частоты, **N** – разрядность аккумулятора фазы.

$\mathbf{F_{CLK}}$ делится на некоторое число, которое определяется кодом **M** частоты и разрядностью **N** аккумулятора фазы.

Шаг перестройки частоты не зависит от ее значения и равен

$$\mathbf{\Delta F_{OUT} = F_{CLK}/2N}$$

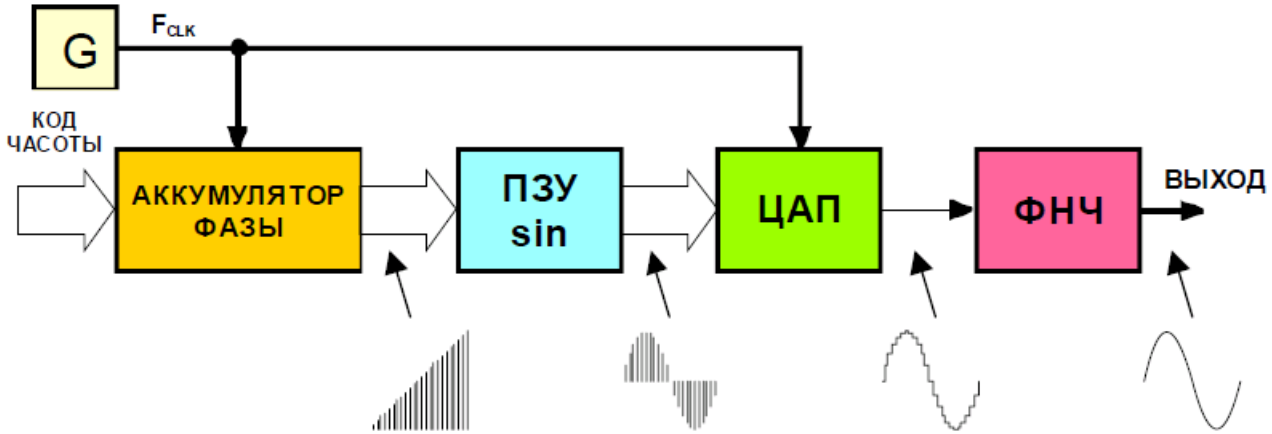
Если увеличить разрядность накапливающего сумматора N, то уменьшится шаг перестройки частоты. Например, если N =32 бита, а F_{CLK}=50 МГц, то частотное разрешение составит порядка 0.01 Гц.

Увеличение разрядности аккумулятора фазы не требует обязательного увеличения разрядности адреса ПЗУ. Для адресации можно использовать лишь необходимое количество старших разрядов кода фазы.

Для уменьшения объема ПЗУ можно использовать свойства симметрии функции sin. В большинстве DDS в ПЗУ содержится только 1/4 периода. Правда, при этом немного усложняется логика формирования адреса.

Прямой цифровой синтез. DDS на основе накапливающего сумматора

Аккумулятор фазы формирует последовательность кодов мгновенной фазы сигнала, которая изменяется линейно. Скорость изменения фазы задается кодом частоты. В ПЗУ линейно изменяющаяся фаза преобразуется в изменяющиеся по синусоидальному закону отсчеты выходного сигнала. Эти отсчеты поступают на ЦАП, на выходе которого формируется синусоидальный сигнал, состоящий из «ступенек». Эти «ступеньки» фильтруются с помощью аналогового ФНЧ, на выходе которого получается синусоидальный сигнал.



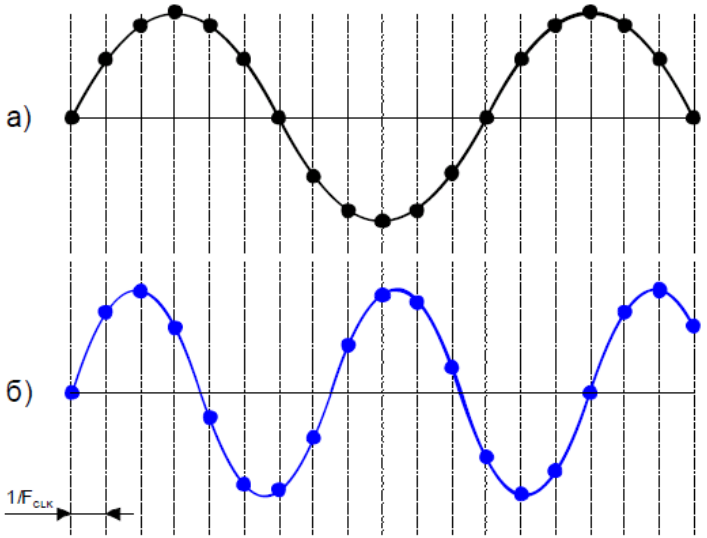
DDS могут также иметь:
- умножитель опорной частоты;
- доп. цифровой сумматор для программирования фазы;
- инверсный sinc фильтр для компенсации неравномерности АЧХ;
- доп. цифровой умножитель для амплитудной модуляции;
- доп. ЦАП для получения квадратурных сигналов I и Q;
- доп. компаратор с низким джиттером для получения цифрового тактового сигнала;
- доп. регистры частоты и фазы для высокоскоростной модуляции

Numerically Controlled Oscillator (NCO) - DDS, не имеющие ЦАП.

Положения выборок выходного сигнала для разных частот

Выходной синусоидальный сигнал восстанавливается из отдельных отсчетов. Целое число отсчетов на период укладывается лишь в частном случае.

В большинстве случаев на каждом новом периоде сигнала отсчеты лежат в новых местах .



Прямой цифровой синтез (DDS)

Основные преимущества DDS:

• цифровое управление частотой и фазой выходного сигнала;

• высокое разрешение по частоте и фазе;

• быстрый переход на другую частоту (или фазу), перестройка по частоте без разрыва фазы, без выбросов и других аномалий, связанных с временем установления;

• исключена необходимость применения точной подстройки опорной частоты;

• цифровой интерфейс легко позволяет реализовать микроконтроллерное управление;

• для квадратурных синтезаторов имеются DDS с I и Q выходами, которые работают согласованно.

Частотное разрешение DDS составляет сотые, и даже тысячные доли Гц при Fout порядка десятков МГц. Скорость перестройки ограничена практически только быстродействием цифрового управляющего интерфейса. Поскольку выходной сигнал синтезируется в цифровом виде, очень просто осуществить модуляцию различных видов.

DDS простой в управлении, высокоинтегрированный, малых габаритов. Современные DDS используют субмикронную CMOS-технологию, 3-х вольтовую логику, миниатюрные корпуса. Постоянно уменьшается цена.

Ограничения, связанные с процессом дискретизации и цифро-аналогового преобразования, который имеет место в DDS:

• максимальная выходная частота не может быть выше половины тактовой (на практике она еще меньше). Это ограничивает области применения DDS на HF (3—30 МГц) и часть VHF (30—300 МГц) диапазонах

• отдельные побочные компоненты на выходе DDS могут быть значительно большими, чем у других видов синтеза. Спектральная чистота выходного сигнала DDS сильно зависит от качества ЦАП

• потребляемая DDS мощность практически прямо пропорциональна тактовой частоте и может достигать сотен милливатт. При больших тактовых частотах DDS могут оказаться непригодными для устройств с батарейным питанием.

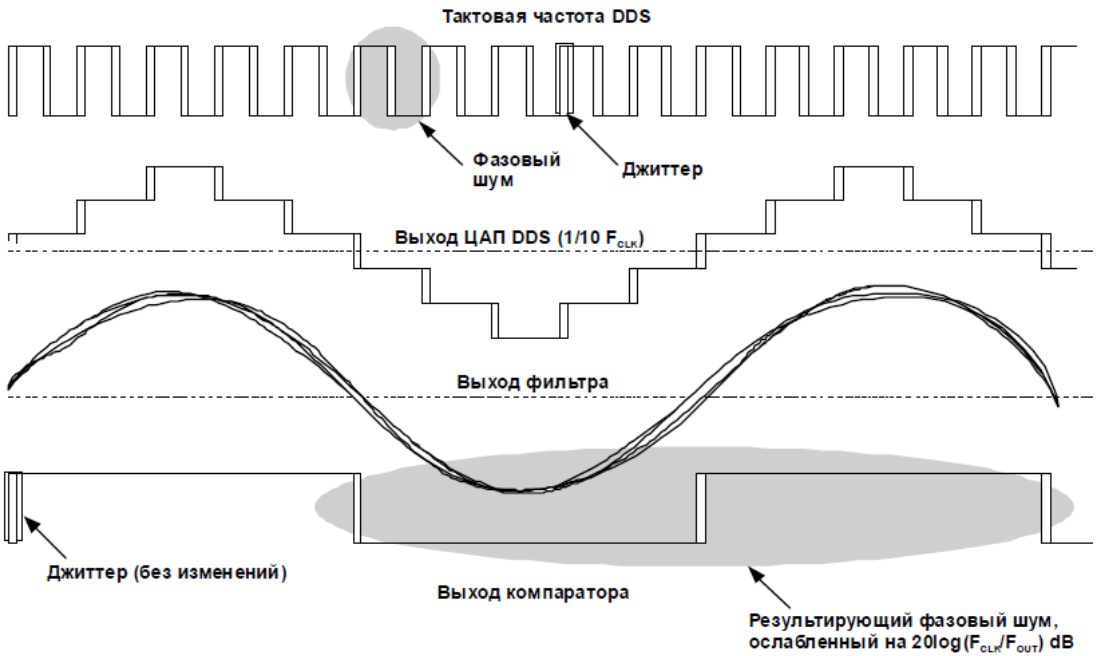
Прямой цифровой синтез. Источник тактового сигнала DDS

Характеристики источника тактового сигнала G:

1. нестабильность частоты (PPM), Parts Per Million - количество миллионных частей от какой-то средней величины;
2. джиттер (пс, нс) - фазовое дрожание цифрового сигнала, нежелательные фазовые или частотные случайные отклонения передаваемого сигнала;
3. фазовый шум (в dBc/Гц, относительно уровня несущей).

G является главным источником фазовых шумов, даже несмотря на их уменьшения при делении частоты в DDS. Фазовый шум выходного сигнала DDS *теоретически* меньше фазового шума тактового сигнала на **$20\log(F_{CLK}/F_{OUT})$** dB. Типичным для собственного фазового шума DDS является значение -130 dBc/Гц при расстройке на 1 КГц от выходной частоты. *Практически* если G имеет меньшие фазовые шумы, на выходе DDS все равно не может быть получено их меньшее значение. Поэтому эту величину называют «остаточный фазовый шум».

Фазовый шум и джиттер на выходе DDS



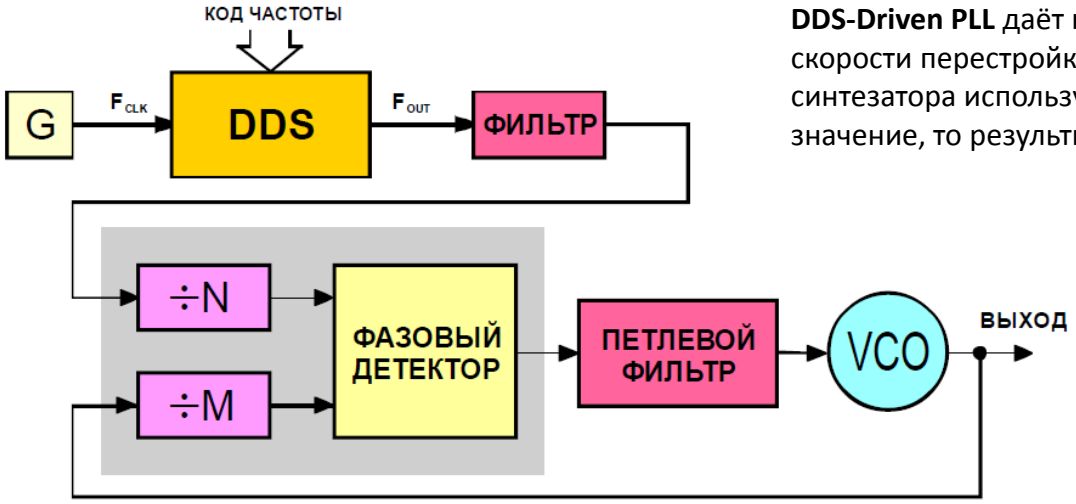
Относительное отклонение частоты на выходе DDS равно относительному отклонению частоты тактового сигнала. Относительный джиттер при делении частоты становится меньше, хотя абсолютное значение джиттера не улучшается.

Значение выходной частоты и частотное разрешение

По формуле **$F_{OUT} = M \cdot F_{CLK}/2N$** тактовая частота делится на величину 2N/M. Поскольку N и M – целые числа, требуемая выходная частота, например, 20 МГц, точно может быть получена не всегда. Может быть получена весьма близкая частота, отстоящая от требуемой не дальше шага перестройки, например, 19.9999999954 МГц или 20.000000009 МГц. Такая погрешность не имеет значение на практике.

Существует также гибридный синтезатор, где в качестве опорного генератора DDS используются кварцевый генератор VCXO , подстраиваемый с помощью PLL в зависимости от отклонения выходной частоты. Такая структура позволяет получить на выходе точные значения частот, правда шаг сетки будет такой же, как и у обычных PLL синтезаторов. Вследствие применения VCXO фазовый шум такого гибридного синтезатора будет намного меньше, чем у обычного PLL синтезатора.

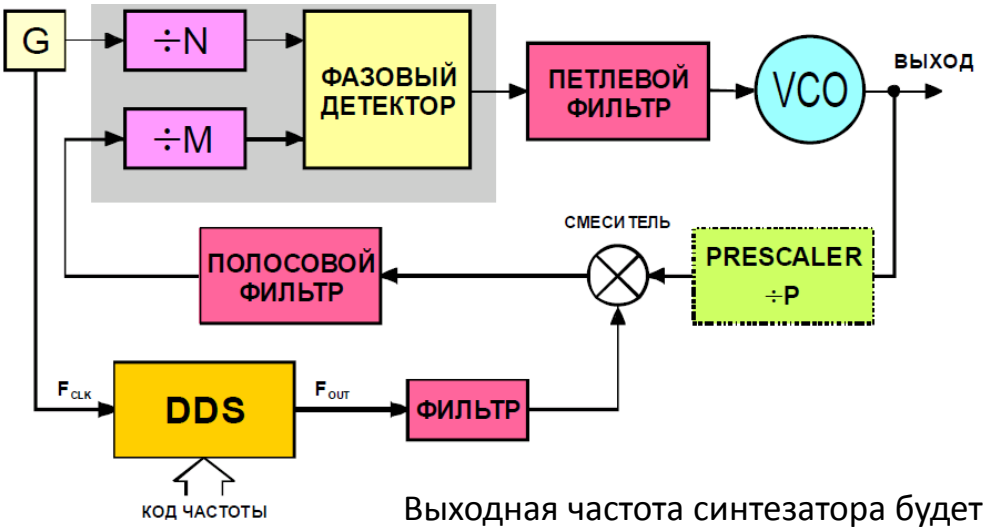
Прямой цифровой синтез. Гибридный PLL/DDS синтезатор (DDS-Driven PLL)



DDS-Driven PLL даёт наилучшие параметры в части полосы частот, разрешения, скорости перестройки, чистоты выходного спектра. Вместо опорной ***F*** для PLL синтезатора используется ***Fout*** DDS. Т.к. шаг перестройки у DDS имеет малое значение, то результирующий шаг также будет оставаться очень малым.

Диапазон вых. частот как в PLL (до нескольких ГГц.)
Комбинируя перестройку DDS и PLL, можно перекрыть очень широкий диапазон частот, в то время как ***Fout*** DDS будет меняться в малом диапазоне. Для фильтрации вых. DDS используются полосовые фильтры.
Шаг перестройки частоты с помощью PLL составляет FDDS_AV·M/N, где FDDS_AV – средняя частота на вых. DDS. Необходимый для непрерывного перекрытия частот диапазон перестройки ***Fout*** DDS равен FDDS_AV·(M/N)min, где (M/N)min – минимальное отношение M/N при перестройке PLL.

PLL-синтезатор со сдвигом частоты с помощью DDS



Чтобы получить высокое частотное разрешение для PLL-синтезатора, добавляется сдвиг вых. частоты, выполненный с помощью DDS. Структура как и у многопетлевого PLL-синтезатора, только вместо PLL высокого разрешения используется DDS. В этом случае частотное разрешение будет таким же, как и у DDS (или в P раз хуже, если применен дополнительный прескалер). Такой синтезатор имеет широкую полосу рабочих частот, свойственную PLL-синтезаторам.

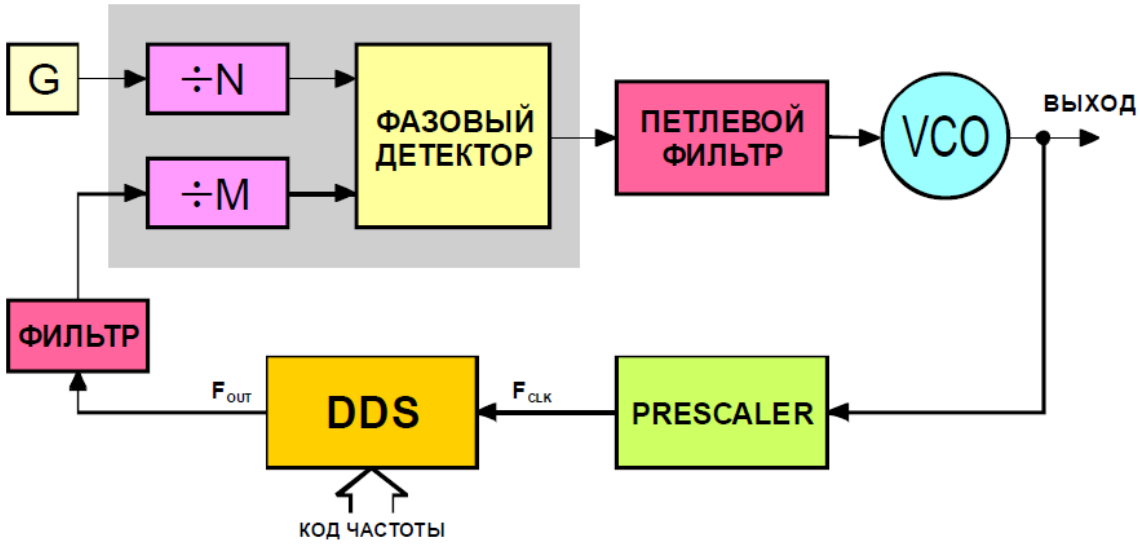
Выходная частота синтезатора будет определяться формулой: $\mathbf{F_{OUT} = (P \cdot M/N) \cdot F_{CLK} + P \cdot FDDS}$

Если дополнительный делитель частоты на P отсутствует, то следует принять P = 1.

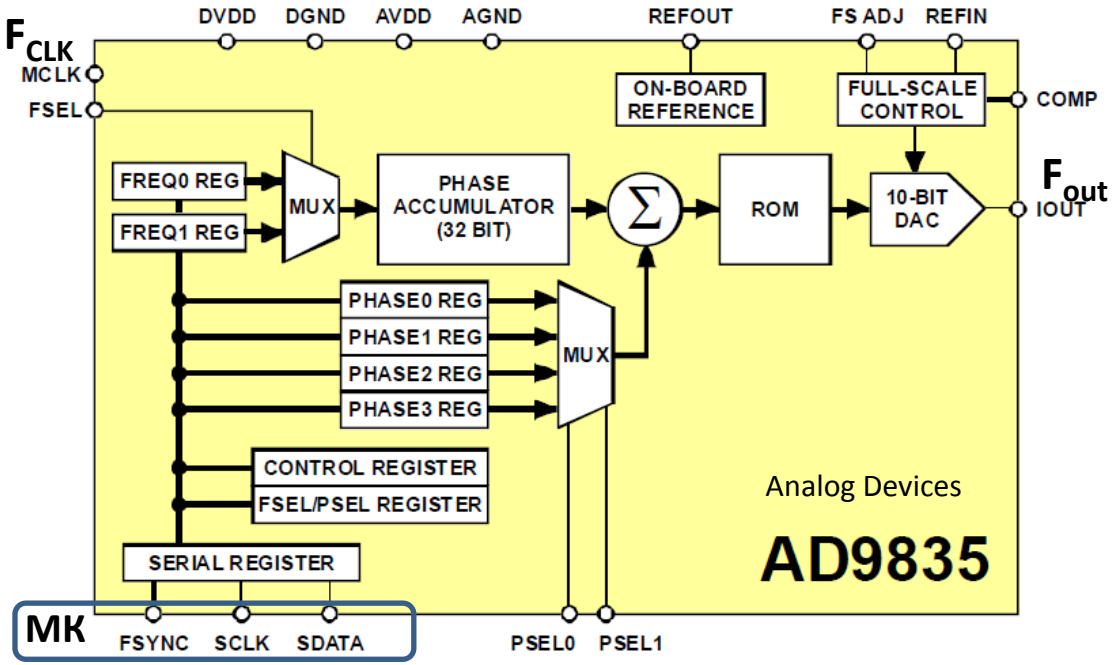
PLL обеспечивает грубый шаг FCLK/N, а внутри шага перестройку обеспечивает DDS. Соответственно рабочая полоса частот DDS должна иметь ширину не менее, чем один шаг PLL.

Прямой цифровой синтез. Fractional PLL синтезатор.

DDS в петле PLL даёт дробные коэффициенты умножения F. PLL производит умножение опорной частоты на величину K = M/N, где M – коэфф. деления Fвых. VCO), N – коэффициент деления опорной частоты.
Если последовательно с M-делителем включить DDS, то результирующий коэффициент умножения K = 2N·M/N·MDDS, где MDDS – код частоты DDS.
В качестве опорной частоты DDS используется Fвых частота PLL, поделенная прескалером.
Сохраняя все качества PLL синтезатора, такой синтезатор будет иметь более высокое частотное разрешение.



Пример DDS



Микросхема имеет 16 выводов, $F_{CLK\ max}$ =50 МГц, Vcc=+5 В, потребляемая мощность до 200 мВт. Управление осуществляется от 3-х проводного последовательного интерфейса частотой до 20 МГц. Ток 10-разрядного ЦАП Iвых=4 мА. 32-разрядный аккумулятор фазы при F_{CLK} =50 МГц обеспечивает частотное разрешение 0.01 Гц. Код фазы разрядностью 12 бит. Для осуществления фазовой модуляции между аккумулятором фазы и ПЗУ включен сумматор, на который поступает код фазы с одного из четырех регистров.
Переключение регистров может осуществляться как через последовательный интерфейс, так и с помощью внешних выводов PSEL0 и PSEL1.

Характеристики наиболее распространенных интегральных DDS

Тип	Fosc max, МГц	Разр. кода частоты, бит	Разр. кода sin, бит	Разр. кода cos, бит	Встроен-ный ЦАП	Модуляция	Шина управ-ления	Кол-во выводов	Особенности
Intersil									
HSP45102	40	32	12	-	-	QPSK,BFSK	S	28	NCO
HSP45106	33	32	16	16	-	FM,PM,PSK,FSK	S	85	NCO
HSP45116	52	32	16	16	-	AM,FM,PM,PSK,FSK,QAM	P	160	NCO+Mixer
ISL5314	125	48	14	-	+	QPSK,FSK	S, P	48	DDS
Qualcomm									
Q2240I-1	50	24	10	-	-		P	44	NCO
Q2240I-2	100	32	12	-	-		S	64	NCO
Q2240I-3	100	32	12	-	-		P	64	NCO
Q2368	130	32	12	-	-	BFSK,BPSK,QPSK,PSK	S, P	100	NCO
Q2334	50	32	12	-	-	PSK,FSK,BFSK	P	68	NCO
Gigabit Logic									
10G102	1000	32	12*	-	-		P	68	сумматор
10G103	1000	32	12*	12*	-		P	68	сумматор
Analog Devices									
AD7008	50	32	10	-	+	AM,QAM,PSK,FSK	S, P	44	DDS
AD9831	25	32	10	-	+	PSK,FSK	P	48	DDS
AD9830	50	32	10	-	+	PSK,FSK	P	48	DDS
AD9850	125	32	10	-	+	PM,FM	S, P	28	DDS
AD9851	180	32	10	-	+	PM,FM	S, P	28	DDS
AD9832	25	32	10	-	+	PSK,FSK	S	16	DDS
AD9835	50	32	10	-	+	PSK,FSK	S	16	DDS
AD9852	300	48	12	12**	+	AM,FM,PSK,FSK	S, P	80	DDS
AD9854	300	48	12	12**	+	AM,FM,PSK,FSK	S, P	80	DDS
AD9856	160	32	12	-	+	AM,QAM	P	48	Upconverter

* - разрядность адреса для внешнего ПЗУ. ** - может работать как программно-управляемый ЦАП.

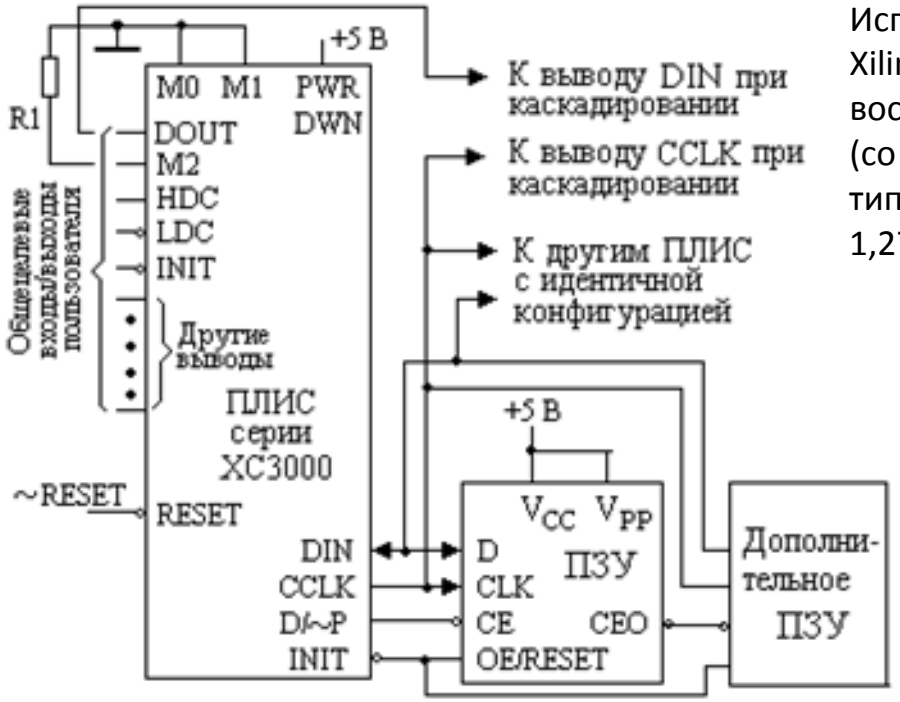
Полными DDS являются модели Analog Devices и ISL5314 Intersil. Остальные микросхемы Numerically Controlled Oscillators (NCOs) и требуют внешнего ЦАП. Самые быстродействующие Gigabit Logic содержат только аккумулятор фазы и требуют внешнего ПЗУ. Некоторые микросхемы имеют дополнительные узлы: например, HSP45116 имеет смеситель, а AD9856 представляет собой квадратурный up-converter. DDS имеют возможность регулировки амплитуды в цифровом виде.

Загрузка конфигурации в ПЛИС фирмы Xilinx

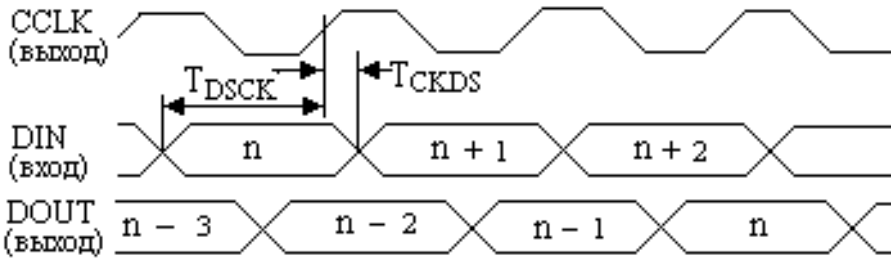
Режим конфигурирования (загрузки битового потока) ПЛИС определяется значениями логических сигналов на выводах M0, M1 и M2 микросхемы.

Наименование режима	M0	M1	M2
Ведущий последовательный (**Master Serial**)	0	0	0
Ведущий параллельный низкий (**Master Parallel Low**)	0	0	1
Ведущий параллельный высокий (**Master Parallel High**)	0	1	1
Периферийный (**Peripheral**)	1	0	1
Подчинённый (**Slave**)	1	1	1
Скоростной (**Express**)	0	1	0

Ведущий последовательный режим конфигурации



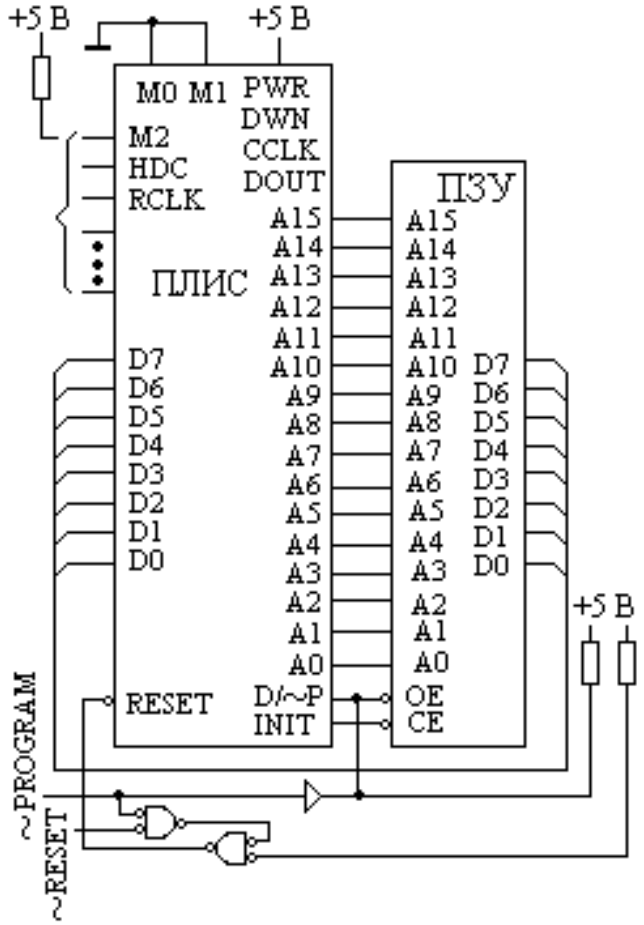
Использует последовательное одноразрядное ПЗУ. Фирма Xilinx разработала МС ПЗУ серии XC1700, выпускаемые в восьмивыводных керамических и пластмассовых DIP-корпусах (со штыревыми выводами, 2,54 мм), миниатюрных корпусах типа SOIC (с планарными выводами и шагом между ними 1,27 мм).

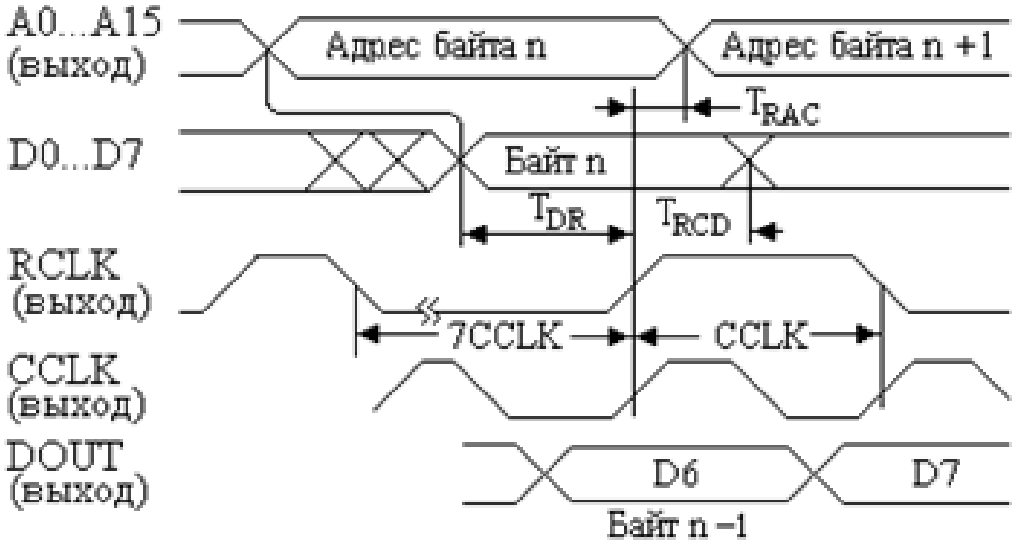


Параметр	Обозначение	Min	Размерность
Предустановка данных	T_{DSCK}	60	нс
Удержание данных	T_{CKDS}	0	нс

Загрузка конфигурации в ПЛИС фирмы Xilinx. Ведущий параллельный режим

Хранение программы конфигурации в восьмиразрядном ПЗУ с произвольной выборкой.



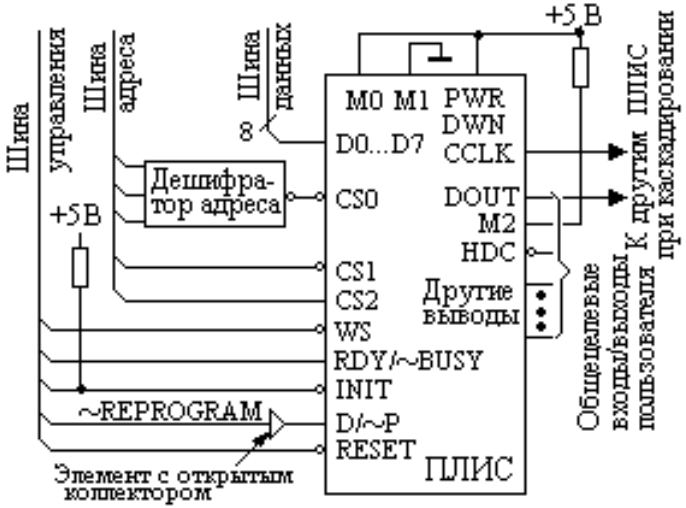


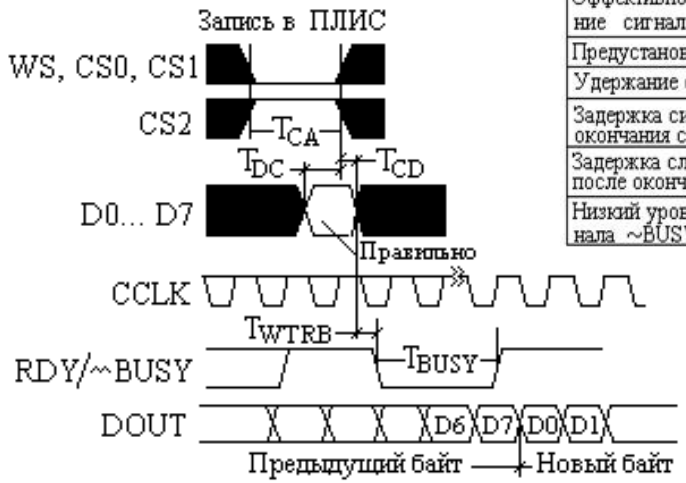
Параметры	Обозн.	Min	Max	Разм.
Установка адреса	T_{RAC}	0	200	нс
Предустановка данных	T_{DRC}	60		нс
Удержание данных	T_{RCD}	0		нс
Высокий уровень RCLK	T_{RCH}	600		нс
Низкий уровень RCLK	T_{RCL}	4,0		мкс

Битовой поток может храниться, начиная с нулевого адреса (для режима **Master Low**), или с верхнего адреса (для режима **Master High**). Хранение битового потока с верхнего адреса используется, когда в этой же микросхеме ПЗУ размещаются программы микропроцессоров, тем самым удаётся сократить количество микросхем ПЗУ на печатной плате.

Загрузка конфигурации в ПЛИС фирмы Xilinx. Периферийный режим

Предполагает наличие вспомогательного дешифратора адреса, подключаемого к системной шине. Когда на вых. CSO этого дешифратора устанавливается лог. 0, а на вх. CS1 и CS2 подаются 0 и 1 соответственно, то ПЛИС подготавливается к приёму одного байта битового потока конфигурации с шины данных, выставляя сигнал “Готов” (Ready) – лог. 1 на выходе RDY/~BUSY. Этот байт запоминается по заднему фронту сигнала WS и затем записывается в «теневое» ЗУ. На время записи ПЛИС вырабатывается сигнал «Занято» (Busy) – логический 0 на выводе RDY/~BUSY.

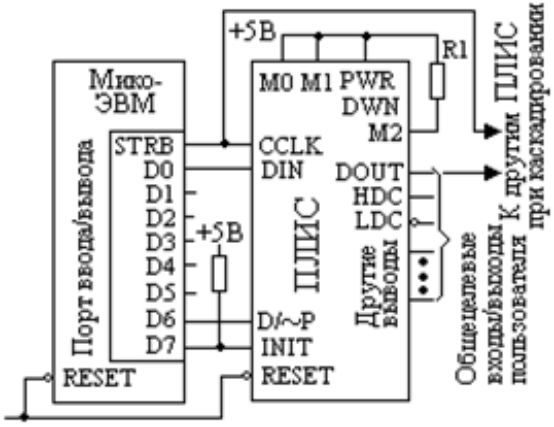




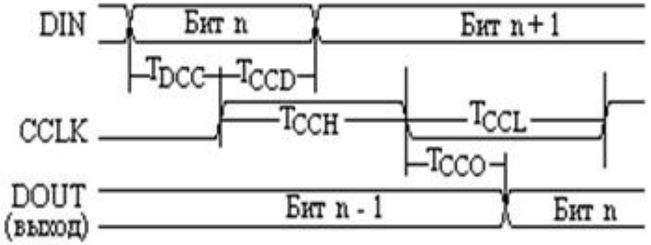
Параметр	Обозн.	Min	Max	Разм.
Эффективное время записи (совпадение сигналов CS0, CS1, CS2, WS)	T_{CA}	100		нс
Предустановка сигналов данных	T_{DC}	60		нс
Удержание сигналов данных	T_{CD}	0		нс
Задержка сигнала RDY/~BUSY после окончания сигнала ~WS	T_{WTRB}		60	нс
Задержка следующего сигнала ~WS после окончания сигнала ~BUSY		0		нс
Низкий уровень вырабатываемого сигнала ~BUSY	T_{BUSY}	2,5	9	периоды CCLK

XC3000

Подчинённый последовательный режим



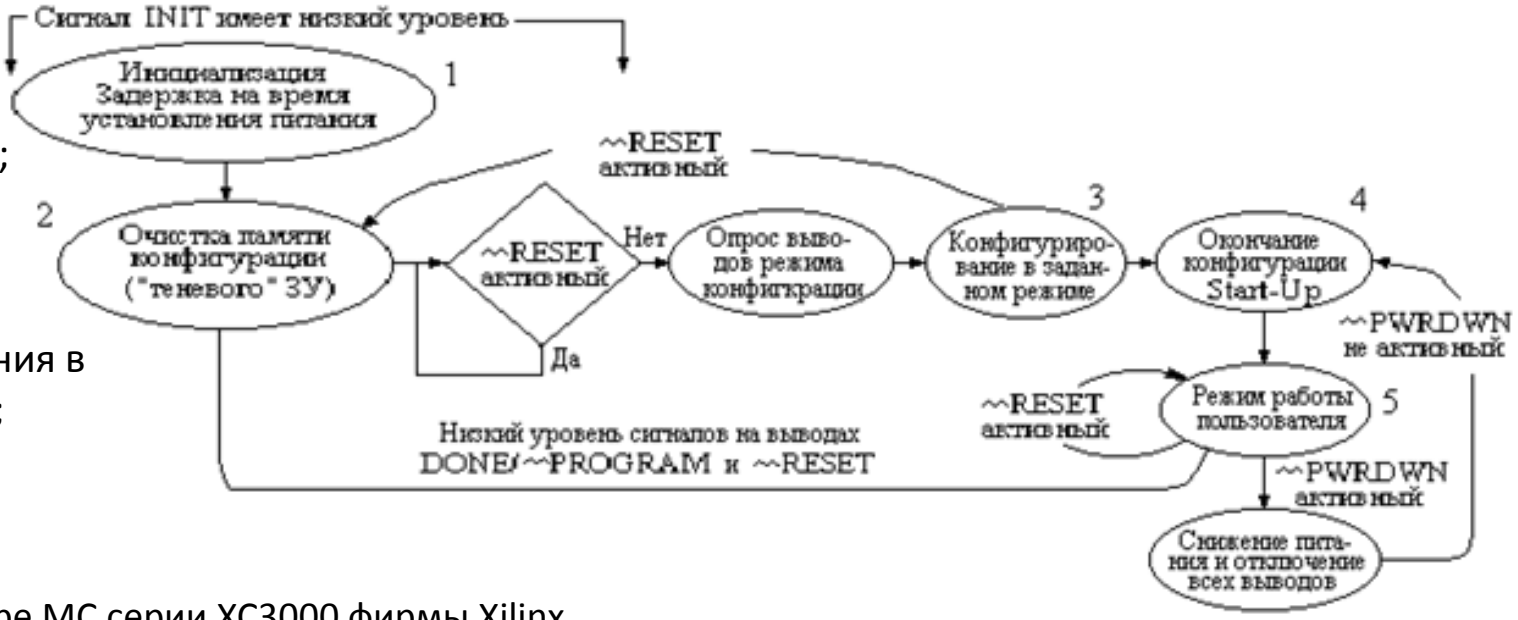
Предполагает побитную подачу потока конфигурации из внешнего источника. Приём производится по переднему фронту тактового сигнала, подаваемого на вход CCLK.



Для начала конфигурирования на вход D/~P следует подать лог. 0 (несколько мкс). Когда счётчик тактовых импульсов на кристалле ПЛИС насчитает заданное количество импульсов, активизируются входы/выходы ПЛИС и выход D/~P переходит в высокоомное состояние.

Загрузка конфигурации в ПЛИС фирмы Xilinx. Состояния

5 состояний:
1. инициализации;
2. очистки памяти конфигурации («теневого» ЗУ);
3. конфигурирования в заданном режиме;
4. запуска;
5. работы пользователя.



Состояния на примере МС серии XC3000 фирмы Xilinx.

После включения питания и нарастания напряжения питания до +2,5… 3 В на вых. INIT (с открытым коллектором) устанавливается лог. 0 - ПЛИС перешла в состояние **инициализации**, которое длится несколько тысяч внутренних тактов (номин. частоты 1 МГц). За это время напряжение питания успеет достичь +4 В. Для ведущих режимов конфигурирования состояние длится 2^{16} тактов (что составляет от 43 до 130 мс), для не ведущих режимов 2^{14} тактов (от 11 до 33 мс). После окончания инициализации все вых. буферы программируемых БВВ переходят в третье состояние и на выходах через высокоомный «подтягивающий» R устанавливается напряжение питания +5 В.

Далее наступает состояние **очистки внутренней памяти конфигурации** («теневого» ЗУ). Продолжительность зависит от ёмкости «теневого» ЗУ (до 375 тактов = до 750 мкс для XC3090). Когда очистка заканчивается, внутреннее устройство управления (**УУ**) ПЛИС проверяет входной лог. уровень на выводе **.** Если **RESET** = 1, выход с открытым коллектором **INIT** переходит в закрытое состояние, показывающее, что *инициализация и очистка памяти закончились*.

Затем ПЛИС считывает значения сигналов на входах M0, M1, M2 и переходит в состояние конфигурирования, во время которого в «теневое» ЗУ загружается программа конфигурации.

Загрузка конфигурации в ПЛИС фирмы Xilinx. Битовый поток

Программа конфигурации представляет собой последовательный поток битов и загружается в «теневое» ЗУ как в сдвиговый регистр. Программа содержит битовое поле, которое указывает её длину. Когда вводится верное число бит, соответствующее данному значению длины, вывод **Done/~Program** ПЛИС (выход с открытым коллектором) переходит в высокоомное состояние, позволяющее через внутренний «подтягивающий» резистор получить сигнал **Done** (СДЕЛАНО) – лог. 1, указывающую, что конфигурирование данной микросхемы завершено.

Структура битового потока конфигурации

11111111 0010 197 разряда - счётчик> 0111	- Начальные биты (4 бита минимум) - Начало заголовка (преамбула) - Длина программы конфигурации - Конец заголовка (4 бита минимум)	Заголовок
0<Фрейм данных #001>111 0<Фрейм данных #002>111 0<Фрейм данных #003>111 … 0<Фрейм данных #195>111 0<Фрейм данных #196>111 0<Фрейм данных #197>111	Для XC3020 197 фреймов данных конфигурации Каждый фрейм содержит: - стартовый бит (0); - 71 бит данных; - три стоповых бита.	Программа конфигурации (повторяется для каждой ПЛИС в гирляндной цепи)
1111	Конец программы (постамбула) (4 бита минимум)	

Процесс конфигурирования должен либо закончиться, либо его можно прервать и запустить снова. Частичное конфигурирование невозможно. После окончания конфигурирования ПЛИС переходит в состояние работы пользователя и начинает выполнять логические функции, определённые разработчиком при проектировании устройства. Во время работы ПЛИС можно вновь перевести в состояние инициализации и повторить процесс конфигурирования. Возможность по переконфигурированию может быть запрещена установкой соответствующих битов в потоке конфигурации. В этом случае переконфигурирование ПЛИС осуществляется только через отключение и последующее включение питания.

Во время инициализации, очистки памяти и конфигурирования на всех выводах ПЛИС (за исключением тех, которые используются в целях конфигурирования) внутреннее **УУ** ПЛИС поддерживает сигнал лог. 1 через внутренние «подтягивающие» (к +5 В) резисторы. После перехода в состояние работы все контакты ПЛИС одновременно начинают функционировать в соответствии с назначением, заданным разработчиком при проектировании устройства.

Загрузка конфигурации в ПЛИС фирмы Xilinx. Варианты

Битовый поток, кроме данных о конфигурации КЛБ и БВВ, содержит также ряд установок, выбираемых пользователем в программе **среде разработки Xilinx**, а именно:

- уровень входных сигналов;
- возможность обратного считывания;
- “подтягивающий” резистор на выводе **DONE/~PROGRAM**;
- время формирования сигнала завершения программирования **DONE**;
- время снятия внутреннего сигнала сброса **MR (Master Reset)**;
- деление на два частоты кварцевого генератора.

*Уровень входных сигналов***.** Во время конфигурирования все выводы имеют ТТЛ-совместимые уровни. После конфигурирования пользователь может выбрать одновременно для всех входов либо ТТЛ- либо КМОП-совместимость.
Входы **DWRDWN, TCLKIN** и **BCLKIN** имеют только КМОП-совместимые уровни.
Возможность обратного считывания. Можно задать однократное, многократное считывание или запретить эту возможность. Обратное считывание позволяет проверить конфигурацию и считать состояния входов и выходов всех триггеров ЛБ и БВВ.
«Подтягивающий» резистор на выходе DONE/~PROGRAM (с открытым коллектором). Позволяет обойтись без внешнего **R**.
Времена формирования сигналов DONE и MR задают временную диаграмму четвёртого состояния ПЛИС – состояния запуска, рассмотренную в следующей лекции.
Деление на два частоты кварцевого генератора позволяет получить симметричный тактовый сигнал, вырабатываемый встроенным кварцевым генератором.

Загрузка конфигурации в ПЛИС фирмы Xilinx. После конфигурирования

Начало работы ПЛИС – 5-е состояние, запуск (**Start-Up**), промежуточное между конфигурированием и работой. Происходит переключение от одного источника тактовых сигналов (тактовые сигналы конфигурации) к другому (системные тактовые сигналы).

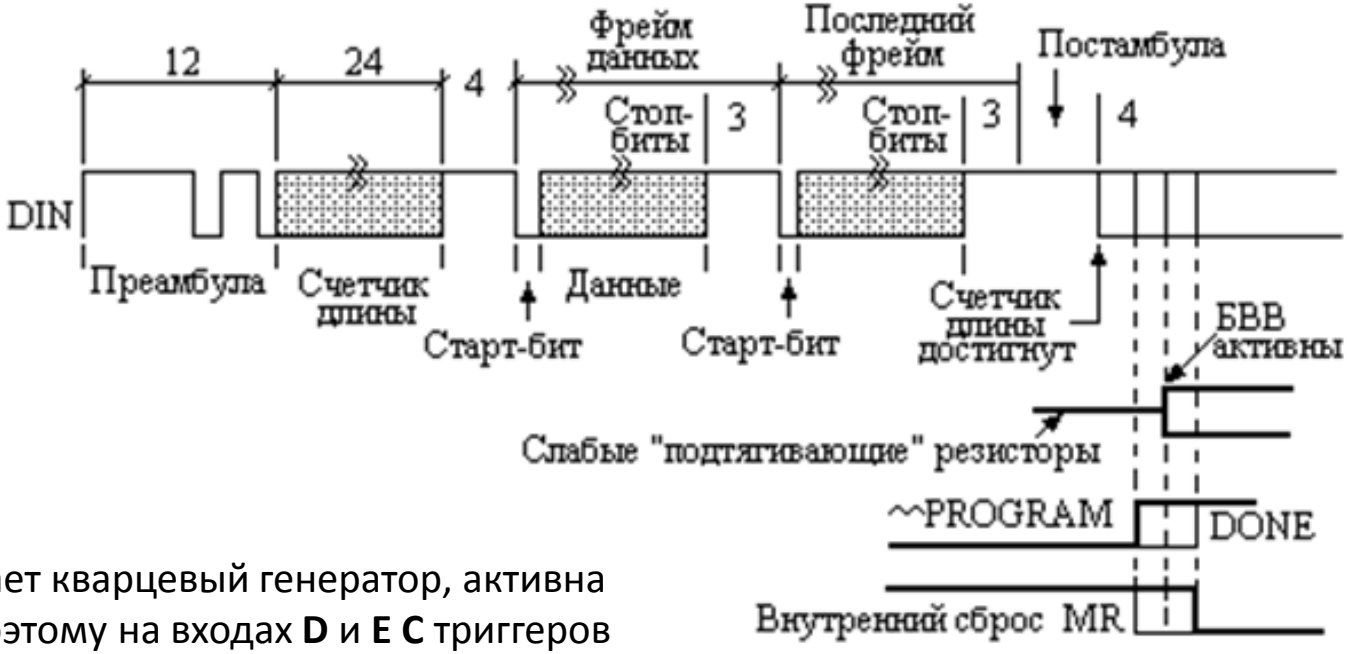
Можно выбирать моменты формирования сигнала **DONE**, сигнализирующего о завершении конфигурирования, и снятия внутреннего сигнала сброса **MR** (**Master Reset**) со всех триггеров ПЛИС.

Для каждого из сигналов можно установить момент на один такт раньше или на один такт позже активизации входов/выходов.

Процесс конфигурирования не синхронизирован с тактовой частотой системы, с которой должна работать ПЛИС после конфигурирования.

Временная диаграмма окончания процессов конфигурирования

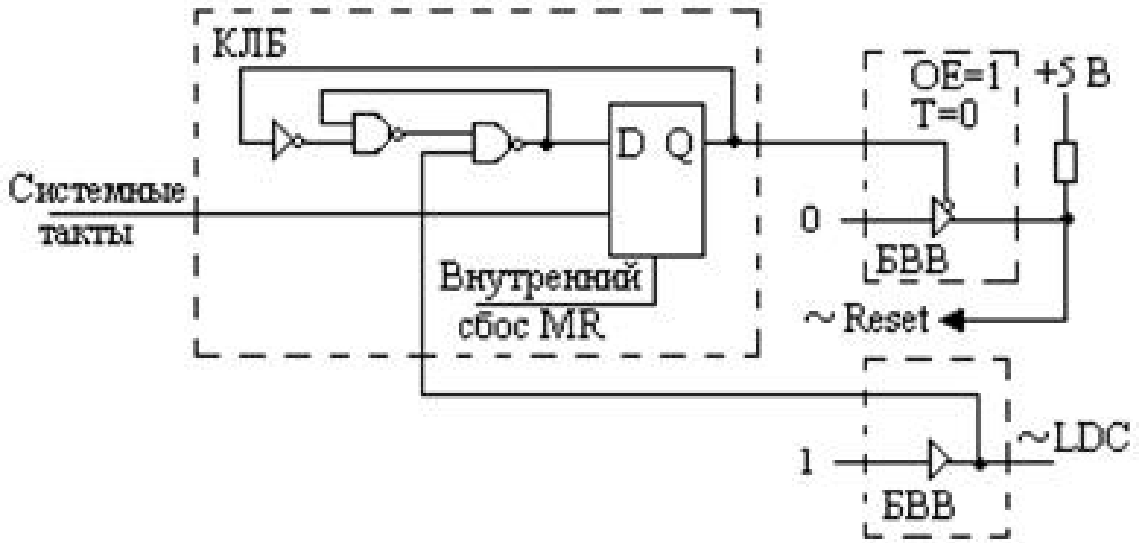
По переднему фронту последнего такта сигнала конфигурации **CCLK** сни мается сигнал внутреннего сброса **MR**, удерживающий триггеры КЛБ и БВВ в 0.



К этому моменту уже работает кварцевый генератор, активна внутренняя логика ПЛИС. Поэтому на входах **D** и **E C** триггеров установлены неправильные сигналы, которые по приходу первого системного тактового импульса на С-вход триггера переведут его в неверное состояние.

Загрузка конфигурации в ПЛИС фирмы Xilinx. После конфигурирования

Схема формирования синхронного сигнала ~RESET после окончания процесса конфигурирования ПЛИС.

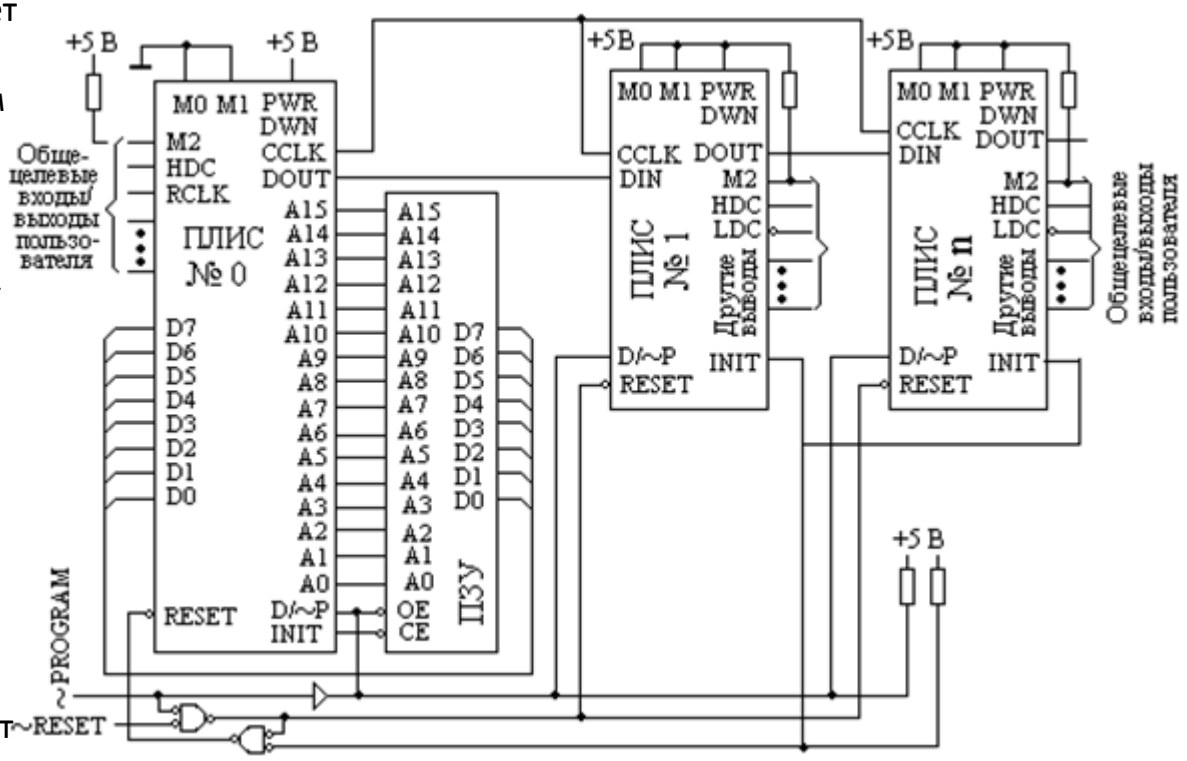


Один КЛБ и два БВВ, один из которых вырабатывает во время конфигурирования сигнал **~LDC** (**Low During Configuration** – низкий уровень сигнала во время конфигурации).
Во время конфигурирования сигнал **~Reset** схемы, соединённый с входом **RESET** ПЛИС, не активен (лог. 1), но все триггеры ПЛИС удерживаются в 0 внутренним сигналом сброса **MR**, действующим только во время конфигурирования.
После окончания конфигурирования сигнал **~LDC** снимается, на **D**-вход триггера подаётся 1, но **~Reset** продолжает удерживаться в лог. 1, так как выход Q триггера находится в 0.
По первому систем-ному тактовому импульсу триггер перебрасывается в 1, ~Reset переходит в 0 и снова сбрасывает все триггеры ПЛИС, в том числе и триггер рассматриваемой схемы, после чего сигнал ~Reset переходит в 1 (т.е. внутренний сброс MR снимается).

Загрузка конфигурации в ПЛИС фирмы Xilinx. Каскадирование

Каскадирование ПЛИС при конфигурировании. Способность ПЛИС вырабатывать 16-разрядный адрес ПЗУ обеспечивает возможность хранения множества программ конфигурации в едином ПЗУ, благодаря чему из одного источника могут конфигурироваться несколько ПЛИС, соединённых в гирляндную цепь. Причём первая ПЛИС №0 использует ведущий режим конфигурирования, остальные – подчинённый, управляемый от первой ПЛИС. Для гарантии инициализации всех подчинённых ПЛИС №1…№n, ПЛИС №0 имеет в 16 раз длиннее паузу инициализации после включения питания. можно увеличить задержку до начала конфигурирования с помощью сигнала **~RESET** (лог. 0), поступающего на входы **RESET** всех ПЛИС.

Битовый поток конфигурации представляет соединение битовых потоков всех ПЛИС и имеет один общий заголовок с указателем суммарной длины потока. После начала конфигурирования заголовок битового потока проходит через все ПЛИС от входа **Din** к выходу **Dout** (с задержкой на 1 такт в каждой ПЛИС) и поэтому все ПЛИС узнают, сколько тактов **CCLK** будет длиться конфигурирование. После загрузки заголовка на вых. **Dout** устанавливаются сигналы лог. 1. Это значение сохраняется до тех пор, пока не заполнится все «теневые» ЗУ. Таким образом, ведущая ПЛИС первую часть битового потока конфигурации записывает в своё «теневое» ЗУ, а оставшуюся часть передаёт через выход **Dout** на подчинённые ПЛИС.



Первая подчинённая ПЛИС «забирает» свою часть битового потока конфигурации и т.д. Когда же счётчики тактовых сигналов насчитают нужное количество тактов, указанное в заголовке битового потока, все ПЛИС завершат конфигурирование.

Интерфейсы

Интерфейс - совокупность средств и методов взаимодействия между элементами системы. Совокупность унифицированных технических и программных средств и правил (описаний, соглашений, протоколов), обеспечивающих одновременное взаимодействие устройств и/или программ в вычислительной системе или обеспечение соответствия систем. Если интерфейс стандартизирован, это даёт возможность модифицировать сам объект, не перестраивая принципы его взаимодействия с другими объектами.

Физический (аппаратный) интерфейс — способ взаимодействия физических устройств.

Для микропроцессоров и ПЛИС проводные интерфейсы.

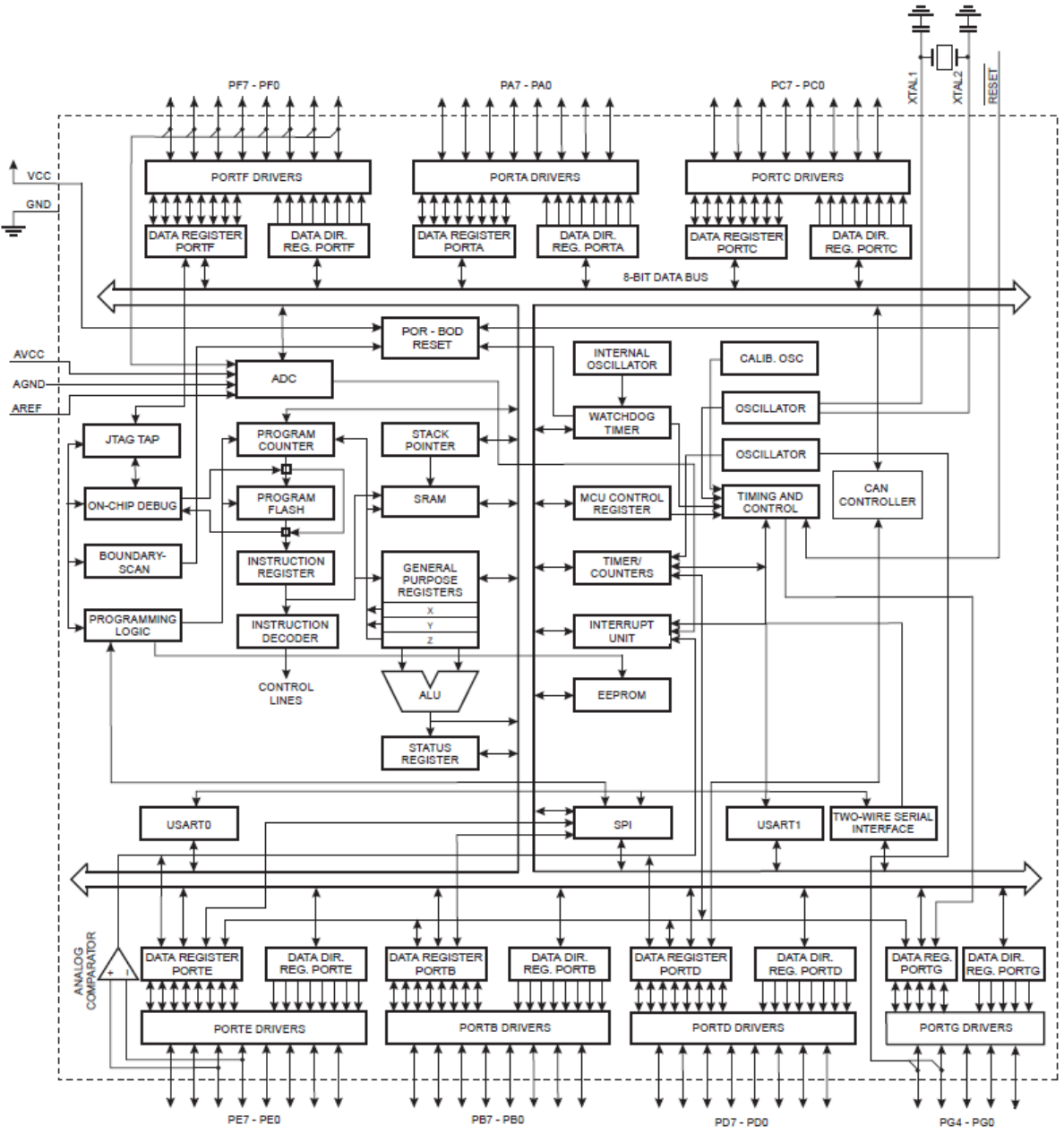
Пропускная способность — метрическая характеристика, показывающая соотношение предельного количества проходящих единиц (информации, предметов, объёма) в единицу времени через канал, систему, узел.

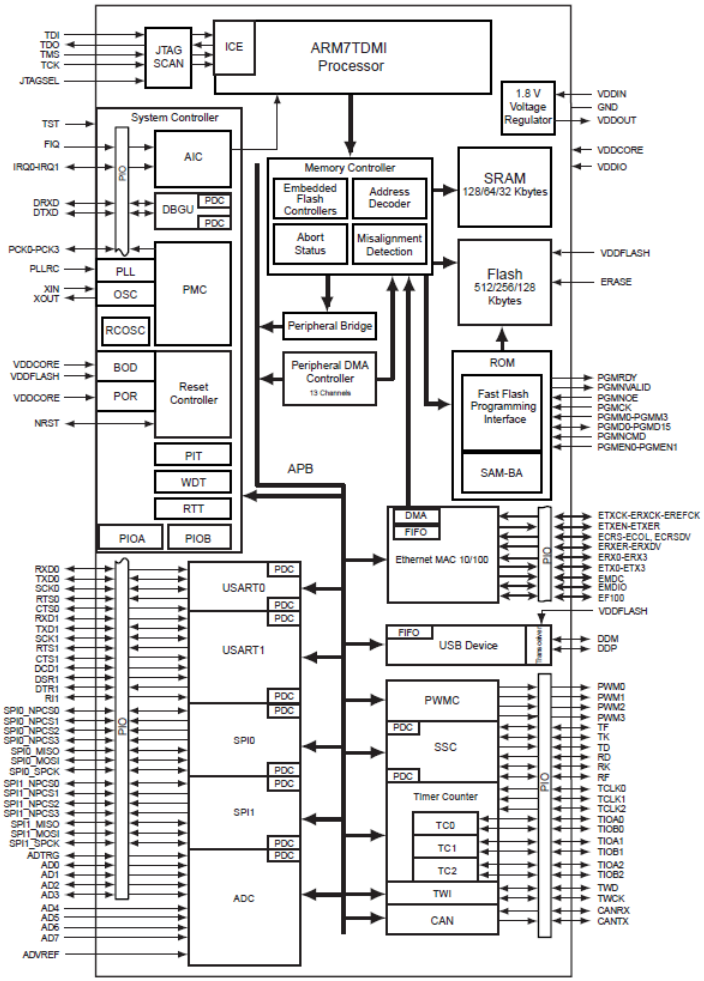
Пиковая пропускная способность — теоретическая максимальная пропускная способность; в реальных условиях производительность интерфейса, как правило, окажется значительно ниже, нежели та, что приведена в таблице.

Компьютерная шина - подсистема, служащая для передачи данных между функциональными блоками компьютера (процессорами). В устройстве шины можно различить механический, электрический (физический) и логический (управляющий) уровни. В отличие от соединения точка-точка, к шине обычно можно подключить несколько устройств по одному набору проводников. Каждая шина определяет свой набор коннекторов (разъемов, соединений) для физического подключения устройств, карт и кабелей. Параллельные шины (данные переносятся по словам, распределенные между несколькими проводниками), последовательные (данные переносятся побитово).

Управление передачей по шине реализуется на уровне прохождения сигнала (мультиплексоры, демультиплексоры, буферы, регистры, шинные формирователи) и со стороны операционной системы (драйвер).

Интерфейсы микропроцессора





Протоколы обмена в большинстве систем работают по принципу "ведущий-ведомый". Одно устройство на магистрали является ведущим (master) и инициирует обмен посылкой запросов подчиненным устройствам (slave), которые различаются логическими адресами (протокол Modbus RTU, Тип соединителей и распайка не оговариваются стандартом).

Пропускные способности проводных интерфейсов

Для локальн. сетей	Проп.сп.
Ethernet (10BASE-X)	**10** Мбит/с
Fast Ethernet (100BASE-X)	**100** Мбит/с
Gigabit Ethernet (1000BASE-X)	**1** Гбит/с
InfiniBand SDR 1X	**2** Гбит/с
InfiniBand DDR 1X	**4** Гбит/с
InfiniBand QDR 1X	**8** Гбит/с
InfiniBand SDR 4X	**8** Гбит/с
10-гигабитный Ethernet (10Gbase-X)	**10** Гбит/с
InfiniBand DDR 4X	**16** Гбит/с
InfiniBand SDR 12X	**24** Гбит/с
InfiniBand QDR 4X	**32** Гбит/с
InfiniBand DDR 12X	**48** Гбит/с
InfiniBand QDR 12X	**96** Гбит/с
100-гигабитный Ethernet (100GBASE-X)	**100** Гбит/с

Для расширения портативных устройств	Проп.сп.
PC Card, 32 разряда, байтами	**267** Мбит/с
PC Card, 32 разряда, двойными словами	**1067** Мбит/с
ExpressCard при PCI Express	**2000** Мбит/с

Шина	разрядн / частота	Проп.сп.
I²C	— / —	**3,4** Мбит/с
ISA	8 / 4,77 МГц	**9,6** Мбит/с
PCI 2.0	32 / 33 МГц	1 Гбит/с
PCI 2.1-3.0	64 / 33 МГц	2 Гбит/с
AGP 1.0 (1x)	32 / 66 МГц	2 Гбит/с
PCI 2.1-3.0	64 / 66 МГц	4 Гбит/с
AGP 1.0 (2x)	32 / 66 МГц	4,2 Гбит/с
RapidIO LP-LVDS	8 / 250 МГц	8528 Мбит/с
AGP 2.0 (4x)	32 / 66 МГц	8528 Мбит/с
PCI-X DDR (266)	64 / 266 МГц	17 066 Мбит/с
AGP 3.0 (8x)	64 / 66 МГц	34 133 Мбит/с
PCI-X QDR (533)	64 / 266 МГц	34 133 Мбит/с
PCI Express 2.0	— / —	**128,00** Гбит/с
PCI Express 3.0	— / —	**204,80** Гбит/с
HyperTransport 3.1	32 / 3,20 ГГц	**409,60** Гбит/с

Память	Проп.сп
FPM RAM	1,408 Гбит/с
EDO RAM	2,112 Гбит/с
SPARC MBus	2,550 Гбит/с
PC1600 (DDR-200)	12,50 Гбит/с
PC3-19200 (DDR3-2400)	150,00 Гбит/с

Для внутр. накопителей	Проп.сп
ATA-1 (DMA-0)	33,6 Мбит/с
ATA-2 (DMA-2)	133 Мбит/с
ATA-4 (UDMA-0)	133 Мбит/с
ATA-4 (UDMA-1)	200 Мбит/с
ATA-7 (UDMA-6)	1066 Мбит/с
SATA 1.x 1.5Gb/s	**1,2** Гбит/с
SATA 3.x 6Gb/s	**4,8** Гбит/с

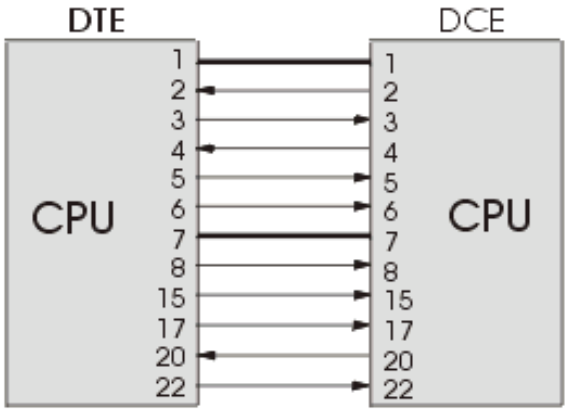
Внешн. устройства	Проп.сп
RS-232	**230,4** Кбит/с
USB 1.0 Low Speed	**1,5** Мбит/с
USB 1.0 Full Speed	**12** Мбит/с
USB 2.0 Hi-Speed	**480** Мбит/с
SATA 2.0	**2,4** Гбит/с
FireWire 3200	**3,2** Гбит/с
USB 3.0	**4,8** Гбит/с
SATA 3.0	**6** Гбит/с
HDMI 1.3-1.4a	**10,2** Гбит/с
HDMI 2.0a	**18** Гбит/с
Thunderbolt 3	**40** Гбит/с

Интерфейс RS-232, UART

RS-232 (Recommended Standard 232) — физический уровень асинхронного (UART) интерфейса. RS-232 обеспечивает передачу данных и некоторых специальных сигналов между терминалом (Data Terminal Equipment, DTE) и коммуникационным устройством (Data Communications Equipment, DCE).

Универсальный асинхронный приёмопередатчик (Universal Asynchronous Receiver-Transmitter) — узел вычислительных устройств, предназначенный для организации связи с другими цифровыми устройствами. Преобразует передаваемые данные в последовательный вид так, чтобы было возможно передать их по цифровой линии другому аналогичному устройству. Представляет собой логическую схему, с одной стороны подключённую к шине вычислительного устройства, а с другой имеющую два или более выводов для внешнего соединения.

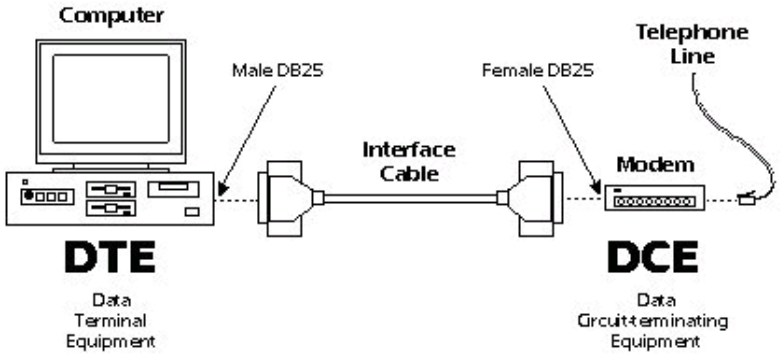
RS-232-C соединяет два устройства. Линия передачи первого соединяется с линией приема второго и наоборот (полный дуплекс).

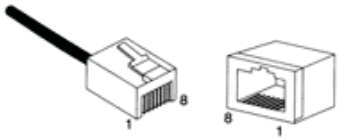


Стандарт	EIA RS-232-C, CCITT V.24
Скорость передачи	115 Кбит/с (максимум)
Расстояние передачи	15 м (максимум)
Характер сигнала	несимметричный по напряжению
Количество драйверов	1
Количество приемников	1
Схема соединения	полный дуплекс, от точки к точке

Интерфейс RS-232-C

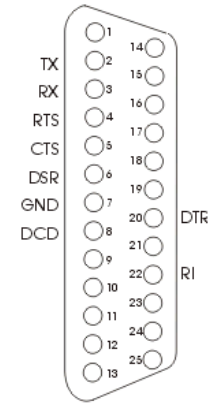
Данные в RS-232C передаются в последовательном коде побайтно. Каждый байт обрамляется стартовым и стоповыми битами. Данные могут передаваться как в одну, так и в другую сторону (дуплексный режим).

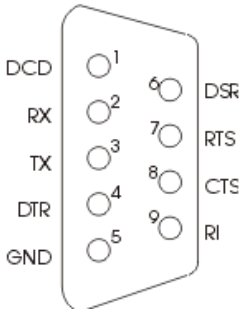


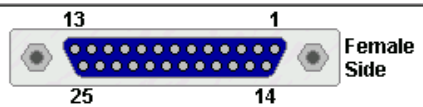


RJ-45

Контакт	Обозн.	Направление	Описание
1	RI	<--	Ring Indicator
2	CD	<--	Carrier Detect
3	DTR	-->	Data Terminal Ready
4	GND	---	System Ground
5	RxD	<--	Receive Data
6	TxD	-->	Transmit Data
7	CTS	<--	Clear to Send
8	RTS	-->	Request to Send

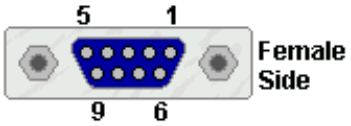






DB25 Розетка (мама)

Контакт	Обозн.	Направление	Описание
1	SHIELD	---	Shield Ground - защитная земля, соединяется с корпусом устройства и экраном кабеля
2	TXD	-->	Transmit Data - Выход передатчика
3	RXD	<--	Receive Data - Вход приемника
4	RTS	-->	Request to Send - выход запроса передачи данных
5	CTS	<--	Clear to Send - вход разрешения терминалу передавать данные
6	DSR	<--	Data Set Ready - вход сигнала готовности от аппаратуры передачи данных
7	GND	---	System Ground - сигнальная (схемная) земля
8	CD	<--	Carrier Detect - вход сигнала обнаружения несущей удаленного модема
9-19	N/C	-	-
20	DTR	-->	Data Terminal Ready - выход сигнала готовности терминала к обмену данными
21	N/C	-	-
22	RI	<--	Ring Indicator - вход индикатора вызова (звонка)
23-25	N/C	-	-

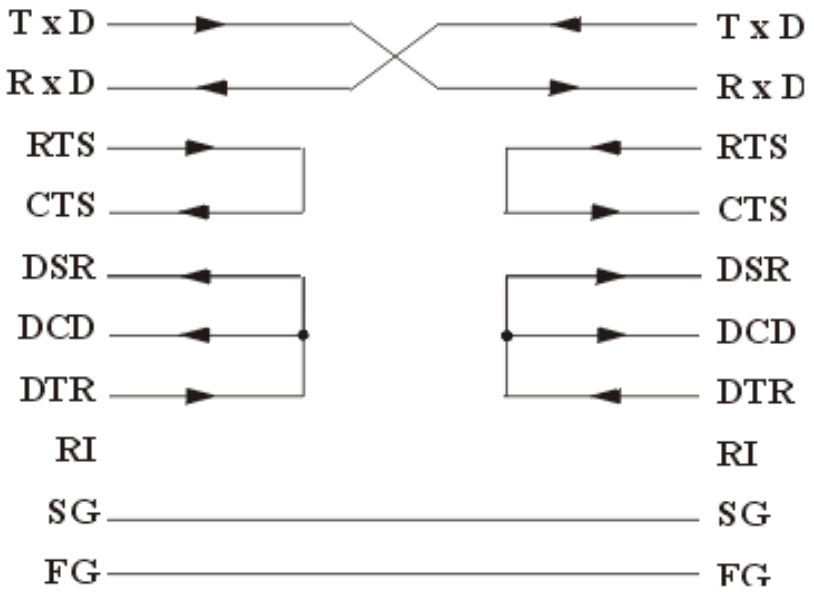


DB9 Розетка (мама)

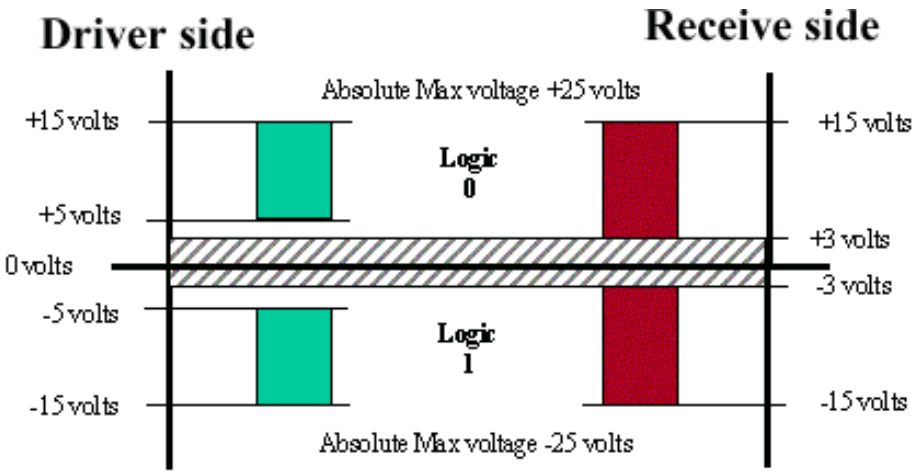
Контакт	Обозн.	Направление	Описание
1	CD	<--	Carrier Detect
2	RXD	<--	Receive Data
3	TXD	-->	Transmit Data
4	DTR	-->	Data Terminal Ready
5	GND	---	System Ground
6	DSR	<--	Data Set Ready
7	RTS	-->	Request to Send
8	CTS	<--	Clear to Send
9	RI	<--	Ring Indicator

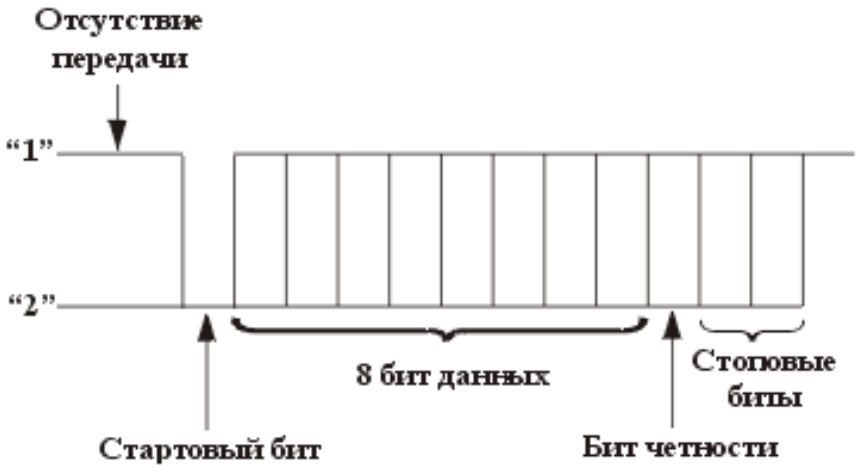
Интерфейс RS-232-C

Схема 4-проводной линии связи



FG - заземление **-TxD** - данные, передаваемые компьютером в последовательном коде (логика отрицательная).**-RxD** - данные, принимаемые компьютером в последовательном коде (логика отрицательная).**RTS** - сигнал запроса передачи. Активен во все время передачи.**CTS** - сигнал сброса передачи. Активен во все время передачи. Готовность приемника. **DSR** - готовность данных. **SG** - сигнальное заземление. **DCD** - обнаружение несущей данных (детектирование принимаемого сигнала). **DTR** - готовность выходных данных. **RI** - индикатор вызова.





Интерфейс RS-485

RS-485 — TIA/EIA-485 Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems (Электрические характеристики передатчиков и приемников, используемых в балансных цифровых многоточечных системах). Соединения контроллеров и другого оборудования и возможность объединения нескольких устройств.

Обмен данными между несколькими устройствами по одной двухпроводной линии связи в полудуплексном режиме.

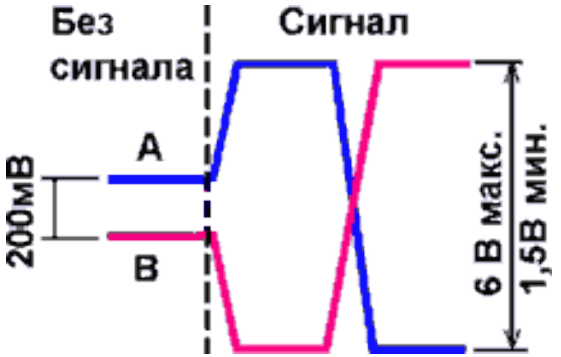
Скорость до 10 Мбит/с. Дальность зависит от скорости: при скорости 10 Мбит/с максимальная длина линии — 120 м, 100 кбит/с — 1200 м.

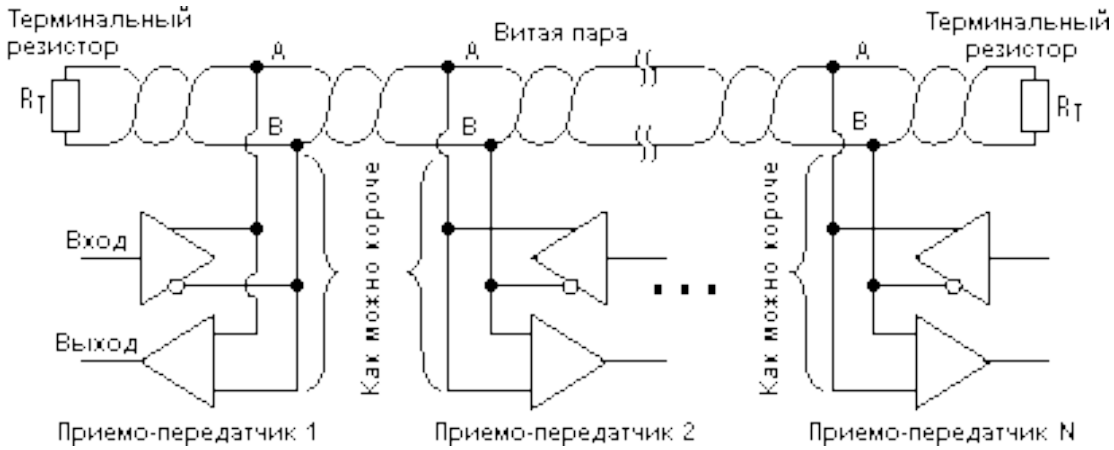
1 передатчик рассчитан на управление 32 приемниками.

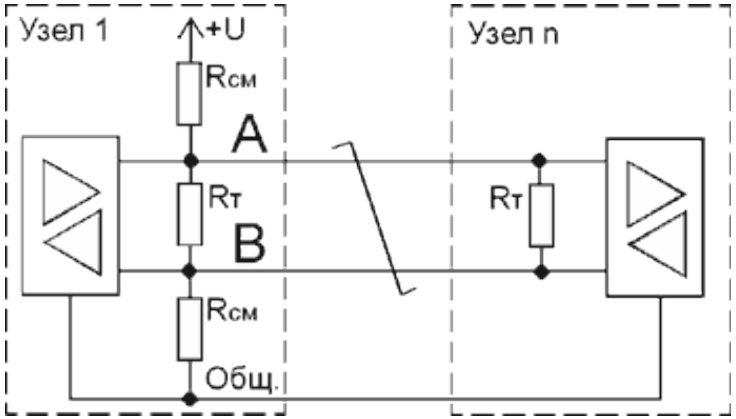
Стандарт не нормирует формат информационных кадров и протокол обмена. Для передачи байтов данных используются фреймы RS-232: стартовый бит, биты данных, бит паритета, стоповый бит.

Уровни сигналов

Интерфейс RS-485 использует балансную (дифференциальную) схему передачи сигнала. Это означает, что уровни напряжений на сигнальных цепях A и B меняются в противофазе, как показано на приведенном ниже рисунке:







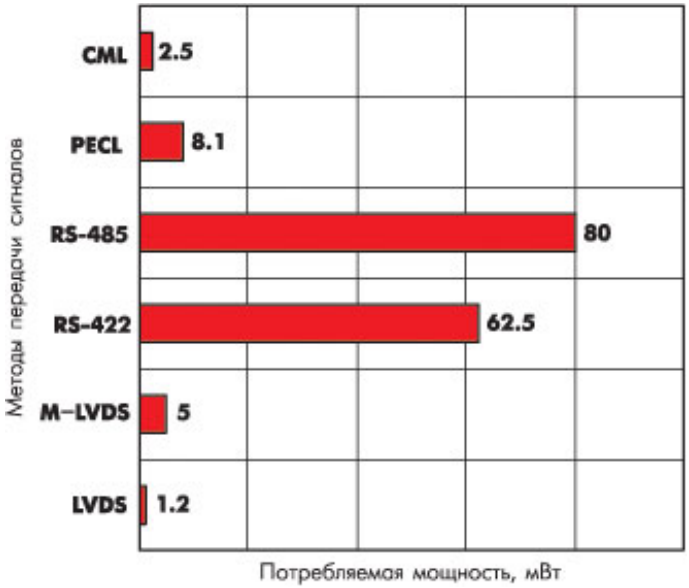
Низковольтная дифференциальная передача сигналов

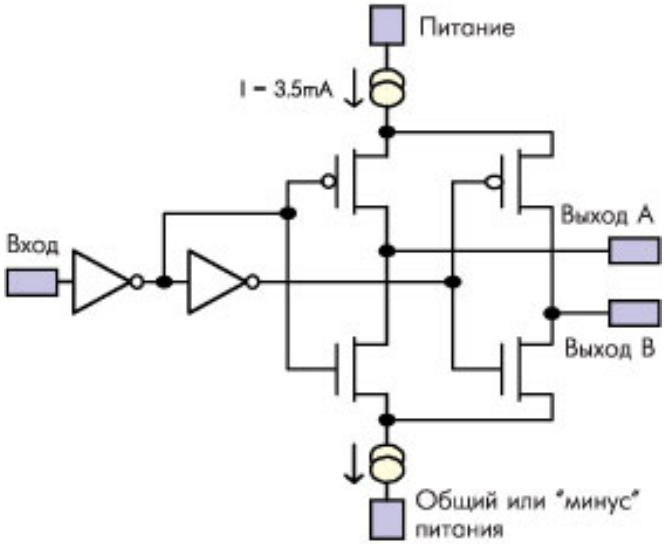
LVDS (Low Voltage Differential Signaling) - передача информации дифференциальными сигналами малых напряжений (до 350 мВ) на двух линиях печатной платы или сбалансированного кабеля со скоростью до сотен и даже нескольких тысячь мегабит в секунду (Mbps). Выходной ток передатчика составляет от 2,47 до 4,54 мА.

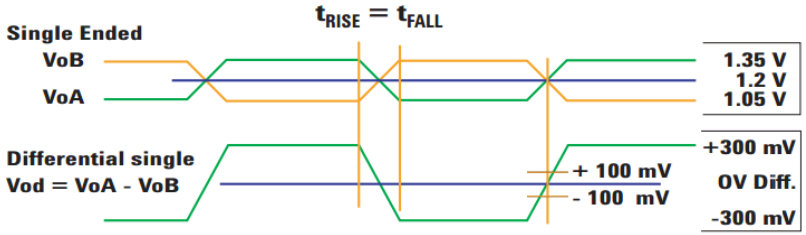
Стандарты:

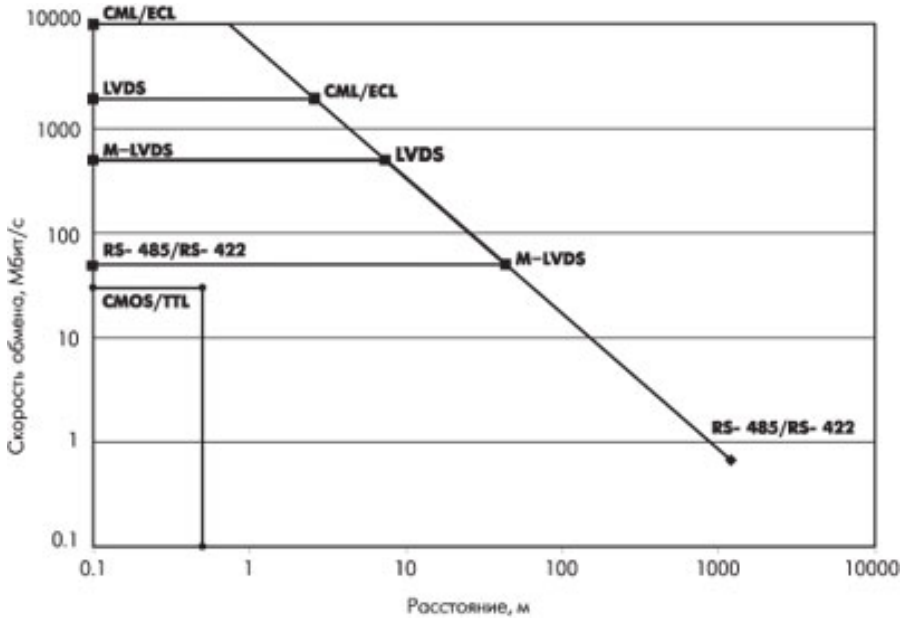
TIA/EIA (Telecommunications Industry Association/Electronic Industries Association) - ANSI/TIA/EIA-644 (LVDS)

IEEE (Institute for Electrical and Electronics Engineering) - IEEE 1596.3



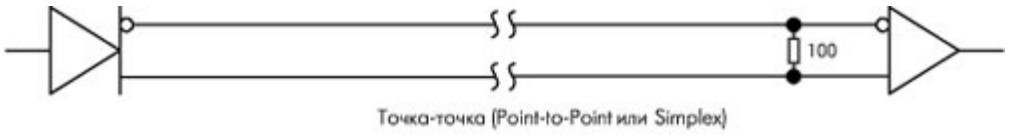




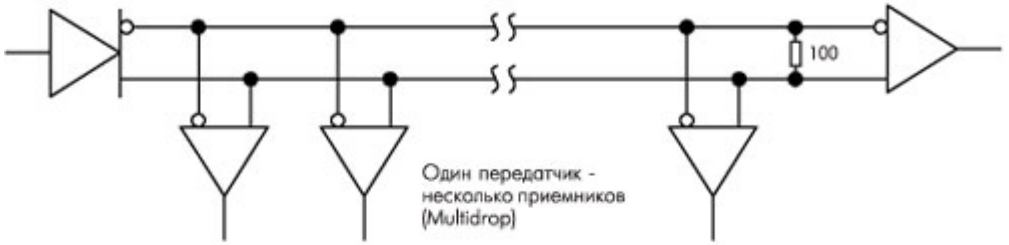


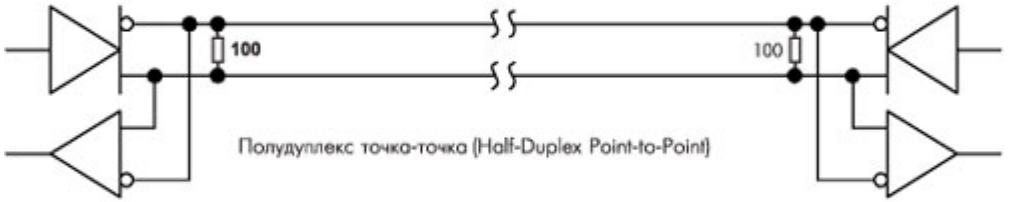
Низковольтная дифференциальная передача сигналов

Подключения

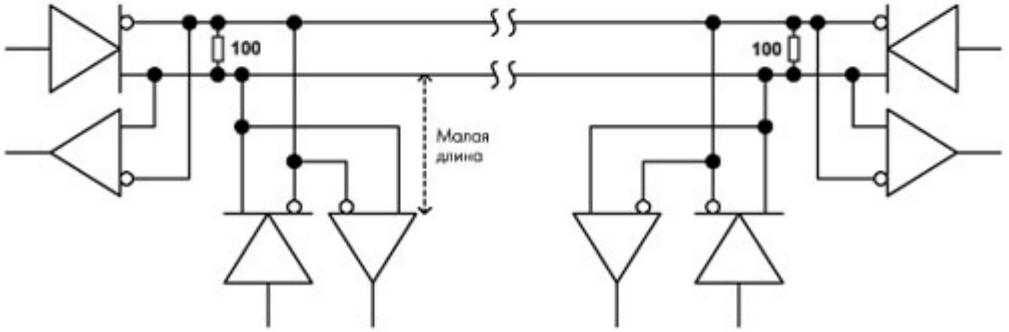


Точка-точка (Point-to-Point или Simplex)





Полудуплекс точка-точка (Half-Duplex Point-to-Point)



Полудуплекс Несколько передатчиков - несколько приемников (Half-Duplex Multipoint)

Применение

•PC/Computing Telecom/Datacom Consumer/Commercial
•Персональные компьютеры: Flat панели , шины мониторов, соединения SCI процессоров, шины принтеров, цифровые копиры, системмные кластеры, шины мультимедиа периферии.
•Передача данных: трансляция, адресная мультиплексия, хабы
•Потребительские системы: видео шины, телевизоры, игровые дисплеи и т.д.
LVDS используется в таких компьютерных шинах как HyperTransport, FireWire, USB 3.0, PCI Express, DVI, Serial ATA, SAS и RapidIO. Современные ПЛИС (например, от Altera или Xilinx) имеют LVDS-порты, что позволяет разрабатывать любые устройства, работающие с шиной на основе LVDS-технологии.

Параметры трансивера (пример)

Параметр	Наименование	Мин.	Макс.	Ед. изм.
Vod	Дифференциальное выходное напряжение	247	454	мВ
Vos	Опорное напряжение	1.125	1.375	В
DVod	Изменение VoD		50	мВ
DVos	Изменение VoS		50	мВ
I_{SB}	Ток короткого замыкания		24	мА
Параметр	Наименование	Мин.	Макс.	Ед. изм.
tr, tf	Длительность выходного фронта/ спада для скорости 200 Мбит/с	0.26	1.5	нс
tr, tf	Длительность выходного фронта/ спада для скорости < 200 Мбит/с	0.26	30 % от ширины бита	нс
IIN	Входной ток приемника		20	мкА
^{v}th	Изменение напряжения		±100	мВ
Vin	Диапазон входного напряжения	0	2.4	В

Интерфейс SPI

SPI (*Serial Peripheral Interface,* последовательный периферийный интерфейс) — последовательный синхронный стандарт передачи данных в режиме полного дуплекса, предназначенный для обеспечения простого и недорогого сопряжения микроконтроллеров и периферии.

Любая передача синхронизирована с общим тактовым сигналом, генерируемым ведущим устройством (процессором). Принимающая (ведомая) периферия синхронизирует получение битовой последовательности с тактовым сигналом. К одному последовательному периферийному интерфейсу ведущего устройства-микросхемы может присоединяться несколько микросхем. Ведущее устройство выбирает ведомое для передачи, активируя сигнал «выбор кристалла» (*chip select*) на ведомой микросхеме. Периферия, не выбранная процессором, не принимает участия в передаче по SPI.

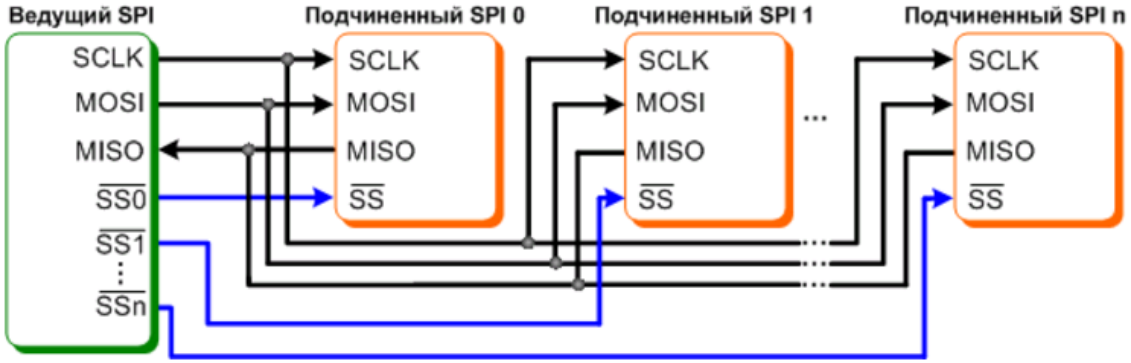
MOSI — выход ведущего, вход ведомого (*Master Out Slave In*) для передачи данных от ведущего устройства ведомому;

MISO — вход ведущего, выход ведомого (*Master In Slave Out)* для передачи данных от ведомого устройства ведущему;

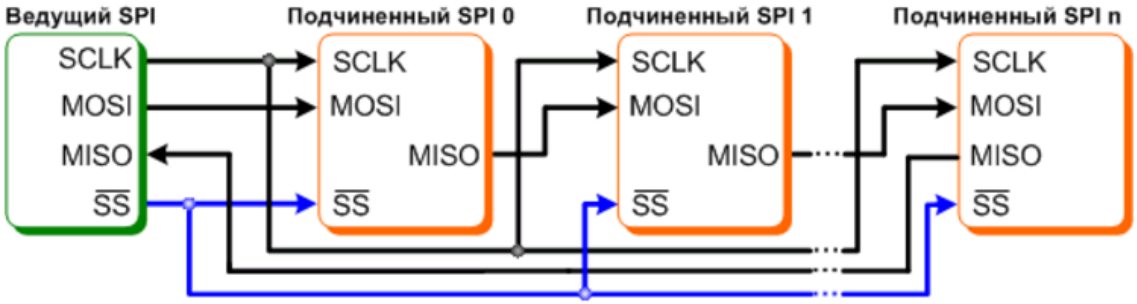
SCLK — последовательный тактовый сигнал (*Serial Clock*) для передачи тактового сигнала для ведомых устройств.

CS или **SS** — выбор микросхемы, выбор ведомого (*Chip Select, Slave Select*)

Простейшее подключение



Независимое подключение



Каскадное подключение

Интерфейс SPI

Протокол идентичен логике сдвигового регистра, побитного ввода и вывода данных по определенным фронтам сигнала синхронизации. Установка данных при передаче и выборка при приеме выполняются по противоположным фронтам синхронизации. В качестве первого фронта в цикле передачи может выступать нарастающий или падающий фронт.

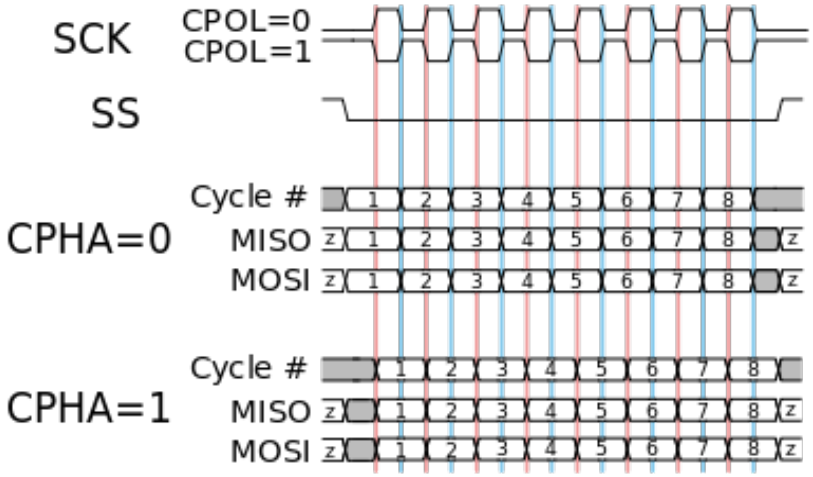
Возможно четыре режима работы интерфейса SPI, характеризующиеся двумя параметрами: **CPOL** - исходный уровень сигнала синхронизации; **CPHA** - фаза синхронизации, в какой последовательности выполняется установка и выборка данных.

CPOL=0: линия синхронизации до начала цикла передачи и после его окончания имеет низкий уровень (т.е. первый фронт нарастающий, а последний - падающий).

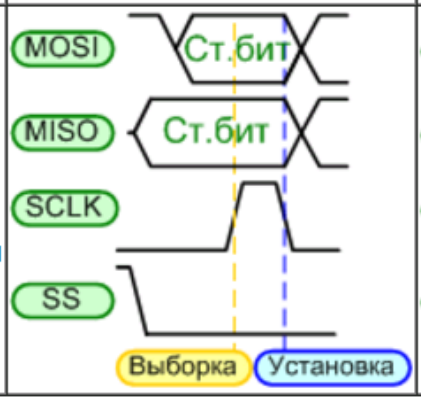
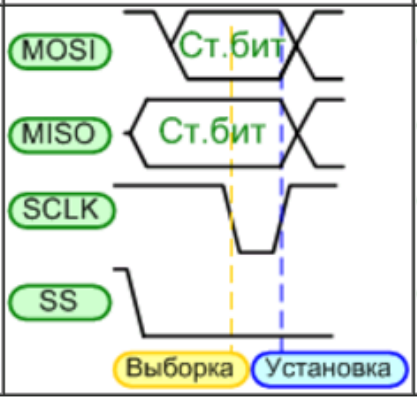
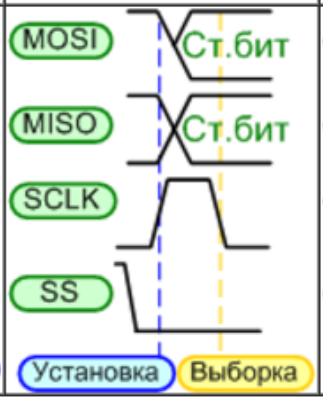
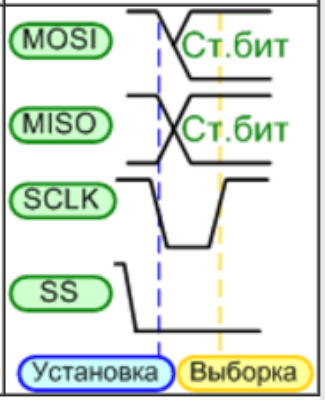
CPOL=1: высокий, т.е. первый фронт падающий, а последний - нарастающий;

CPHA=0: по переднему фронту в цикле синхронизации будет выполняться выборка данных, а затем, по заднему фронту, - установка данных;

CPHA=1: установка данных будет выполняться по переднему фронту в цикле синхронизации, а выборка - по заднему.

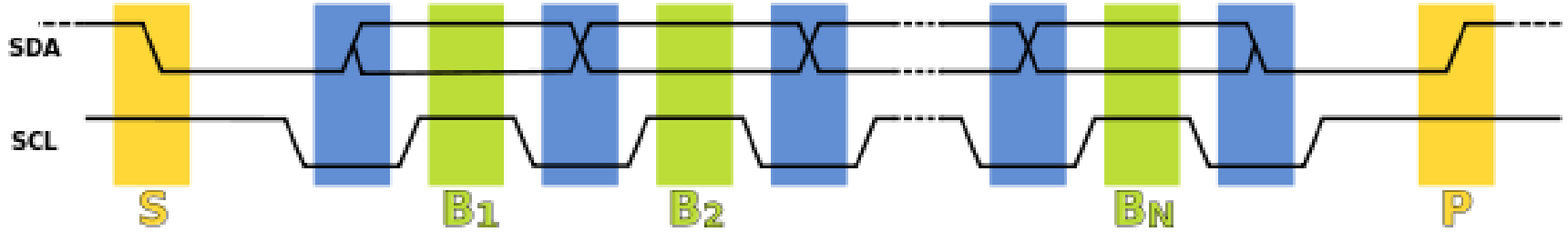


Ведущая и подчиненная микросхемы, работающие в различных режимах SPI, являются несовместимыми.

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

Интерфейс I2C

I²C (Inter-Integrated Circuit) — последовательная шина данных для связи интегральных схем, использующая две двунаправленные линии связи (SDA и SCL). Используется для соединения низкоскоростных периферийных компонентов с материнской платой, встраиваемыми системами и мобильными телефонами.

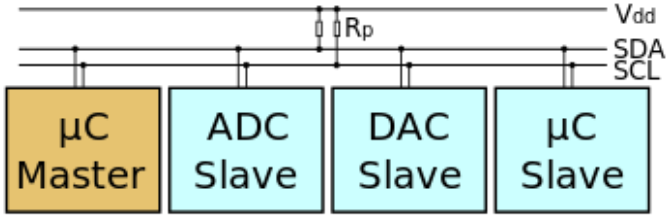


Разработана фирмой Philips в начале 1980-х как простая шина внутренней связи для создания управляющей электроники.
Версия 1998 г. стандарта 2.0 - 3,4 Мбит/с, до 127 устройств, напряжения +5 В или +3,3 В.
Адресация включает 7-битное адресное пространство с 16 зарезервированными адресами (до 112 свободных адресов для подключения периферии на одну шину).

Применение:
доступ к модулям памяти NVRAM;
доступ к низкоскоростным ЦАП/АЦП;
регулировка звука в динамиках;
управление светодиодами;
чтение информации с датчиков мониторинга и диагностики оборудования (термостат центрального процессора или скорость вращения вентилятора охлаждения);
чтение информации с часов реального времени (кварцевых генераторов);
управление включением/выключением питания системных компонент;
информационный обмен между микроконтроллерами.

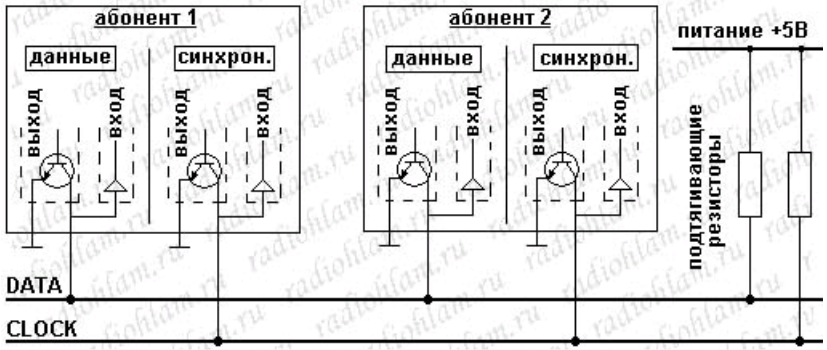
Интерфейс I2C

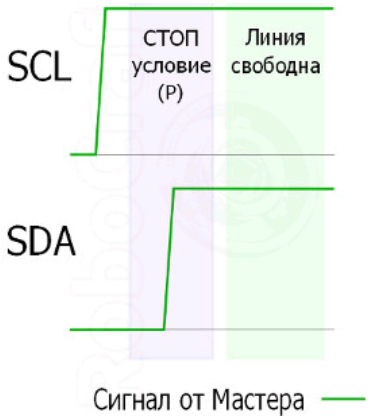
Две двунаправленные линии, подтянутые к напряжению питания и управляемые через открытый коллектор или открытый сток — последовательная линия данных (SDA, *Serial DAta*) и последовательная линия тактирования (SCL, *Serial CLock*).



Протокол

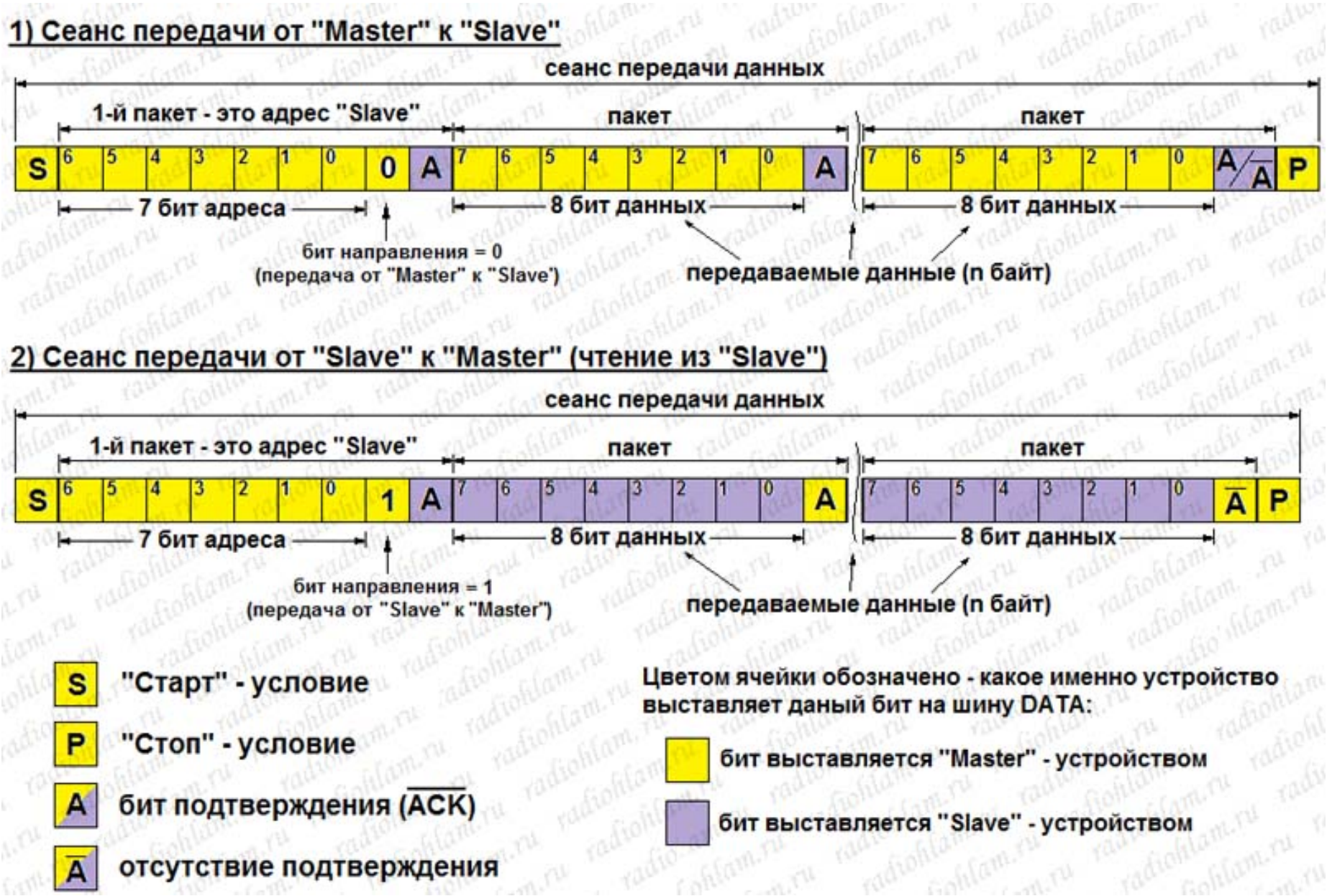
Ведущий формирует состояние СТАРТ: генерирует переход сигнала SDA из ВЫСОКОГО состояния в НИЗКОЕ при ВЫСОКОМ уровне на SCL. Этот переход воспринимается всеми устройствами, подключенными к шине, как признак начала процедуры обмена. Каждый ведущий генерирует свой собственный сигнал синхронизации при пересылке данных по шине. Процедура обмена завершается тем, что ведущий формирует состояние СТОП — переход состояния SDA из низкого состояния в ВЫСОКОЕ при ВЫСОКОМ состоянии SCL. Шина считается освободившейся через некоторое время после фиксации состояния СТОП.



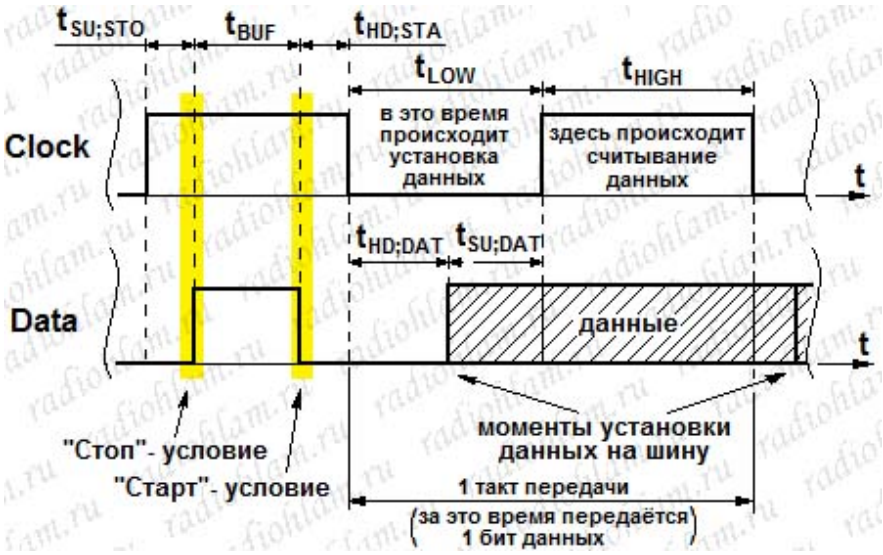


После формирования состояния СТАРТ ведущий опускает состояние SCL в НИЗКОЕ состояние и выставляет на SDA старший бит первого байта сообщения. Количество байт в сообщении не ограничено. Для подтверждения приёма байта от ведущего-передатчика ведомым-приёмником вводится специальный бит подтверждения, выставляемый на шину SDA после приёма 8 бита данных.

Интерфейс I2C протокол



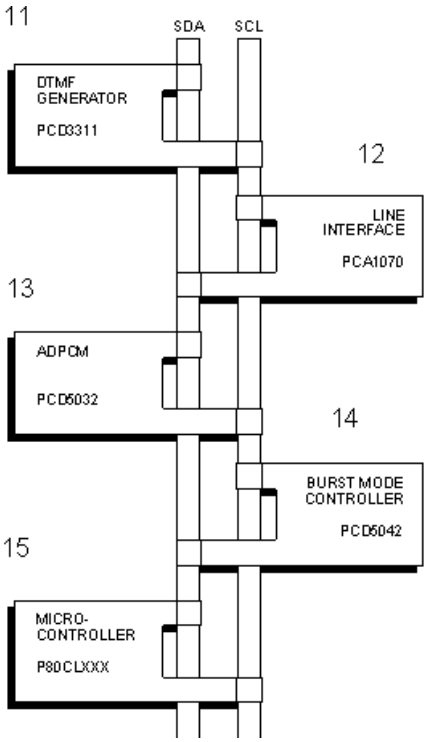
Интерфейс I2C временная диаграмма



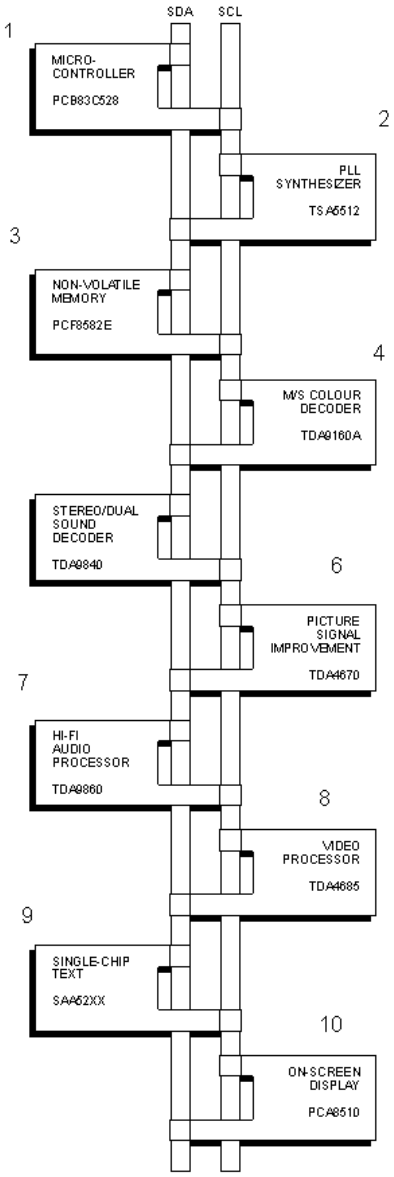
Минимальные значения времени в таблице указаны для максимальной скорости передачи 100 кбит/с.

Параметр	Обозн.	Мин.знач.	Комментарий
Свободная шина	t_{BUF}	4,7 мкс	это минимальное время, в течении которого обе линии должны находиться в свободном состоянии перед подачей "Старт"-условия
Фиксация "Старт"-условия	$t_{HD;STA}$	4,0 мкс	минимальное время от подачи "Старт"- условия до начала первого такта передачи
Готовность "Стоп"-условия	$t_{SU;STO}$	4,0 мкс	минимальное время, через которое можно подавать "Стоп"- условие после освобождения шины Clock
Длительность LOW полупер. шины Clock	t_{LOW}	4,7 мкс	минимальная длительность полупериода установки данных (когда на шине Clock низкий уровень)
Длительность HIGH полупер. шины Clock	t_{HIGH}	4,0 мкс	минимальная длительность полупериода считывания данных (когда на шине Clock высокий уровень)
Удержание данных	$t_{HD;DAT}$	0	то есть данные на шину Data можно выставлять сразу после спада на линии Clock
Готовность данных	$t_{SU;DAT}$	250 нс	то есть поднимать уровень на шине Clock можно не ранее 250 нс после установки данных на шине Data

I2C в телефоне



I2C в телевизоре



Интерфейс CAN

CAN (*Controller Area Network* — сеть контроллеров) — стандарт промышленной сети, ориентированный прежде всего на объединение в единую сеть различных исполнительных устройств и датчиков.

Режим передачи — последовательный, широковещательный, пакетный.
CAN разработан компанией Robert Bosch GmbH в 1980-х и в настоящее время широко распространён в промышленной автоматизации, автомобильной промышленности и др. Стандарт для автомобильной автоматики.

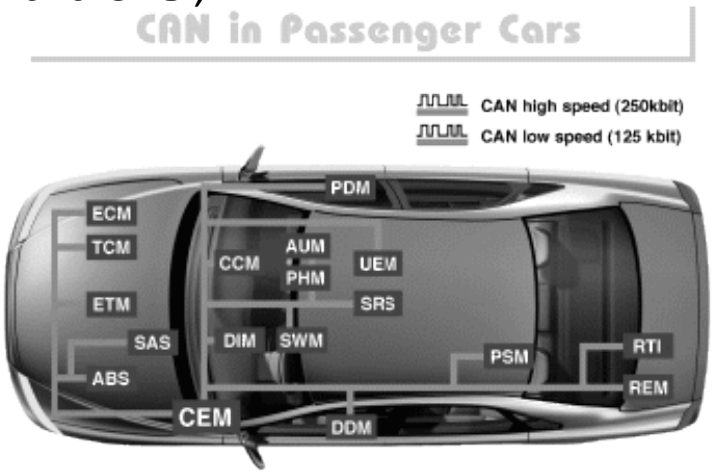
Модель OSI

№	Название уровня	Подуровни CAN	Примечание
7	Прикладной		Стандартом CAN не установлен. Определен стандартами , CANopen, DeviceNet, SDS, CAN, Kingdom и др.
6	Представления	Нет	Нет
5	Сеансовый	Нет	Нет
4	Транспортный	Нет	Нет
3	Сетевой	Нет	Нет
2	Канальный (передачи данных)	LLC	Подтверждение фильтрации, уведомление о перегрузке, управление восстановлением данных
		MAC	Формирование пакетов данных, кодирование, управление доступом, обнаружение ошибок, сигнализация об ошибках, подтверждение приема, преобразование из последовательной формы в параллельную и обратно
1	Физический	Физический	Обеспечение надежной передачи на уровне байтов (кодирование, контрольная сумма, временные диаграммы, синхронизация). Требования к линии передачи

Интерфейс CAN

Свойства:

каждому *сообщению* (не устройству) *устанавливается свой приоритет*; гарантированная величина паузы между двумя актами обмена; гибкость конфигурирования и возможность модернизации системы; широковещательный прием сообщений с синхронизацией времени; непротиворечивость данных на уровне всей системы; допустимость нескольких ведущих устройств в сети; обнаружение ошибок и их сигнализация; автоматический повтор передачи сообщений с ошибкой; автоматическое различение сбоев и отказов с отключением отказавших модулей.

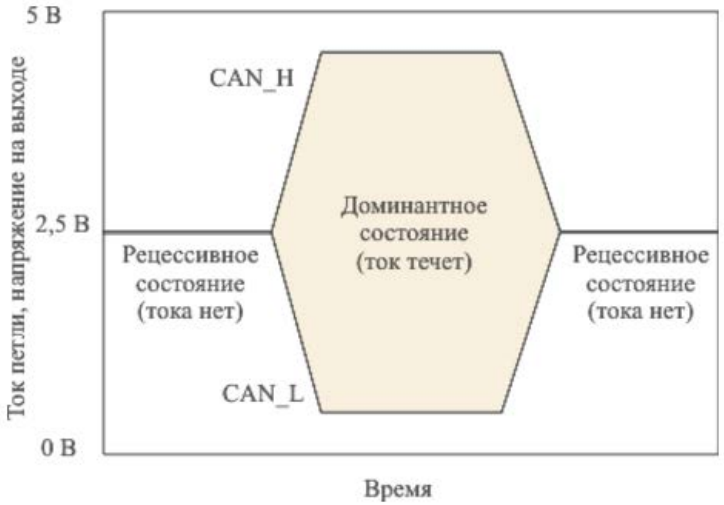


Расстояние, м	25	50	100	250	500	1000	2500	5000
Скорость, Кбит/с	1000	800	500	250	125	50	20	10

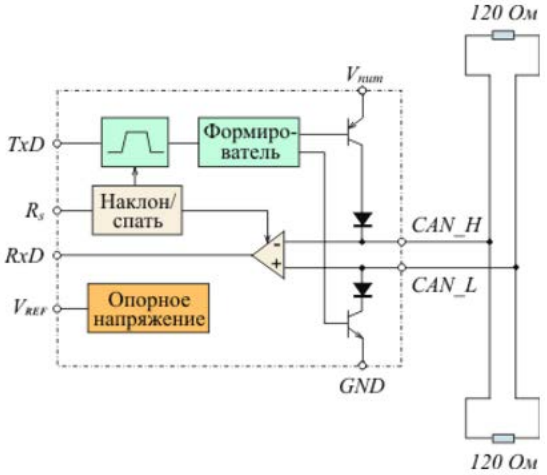
Если один из передатчиков устанавливает в сети логический ноль, а второй - логическую единицу, то это состояние *не является аварийным* - линия остается в состоянии логической единицы

Контакт	Сигнал	Примечание
1	-	Зарезервирован
2	CAN_ L	Сигнал линии
3	CAN_ GND	"Земля"
4	-	Зарезервирован
5	(CAN_ SHLD)	Экран кабеля (не обязательно)
6	(GND)	"Земля" (не обязательно)
7	CAN_ H	Сигнал линии
8	-	Зарезервирован
9	(CAN_ V+)	Внешнее питание (не обязательно, для питания передатчиков с гальванической изоляцией)

Интерфейс CAN. Трансивер



" *доминантное состояние*" состояние линии для обозначения состояния линии с током, "*рецессивное* состояние" как противоположное доминантному



Параметр	Обозн.	Ед. измерения	Мин.	Ном.	Макс	Условие
Для рецессивного состояния шины						
Потенциалы на вых. передатчика	CAN_H	В	2,0	2,5	3	Без нагрузки
	CAN_L	В	2,0	2,5	3	
Диф. напряжение на выходе передатчика	Vdiff	мВ	-500	0	50	Без нагрузки
Диф. напряжение на вх. приемника	Vdiff	В	-1	-	0,5	Без нагрузки
Для доминантного состояния шины						
Потенциалы на вых. передатчика	CAN_H	В	2,75	3,5	4,5	С нагрузкой
	CAN_L	В	0,5	1,5	2,25	
Диф. напряжение на выходе передатчика	Vdiff	В	1,5	2	3	С нагрузкой
Диф. напряжение на вх. приемника	Vdiff	В	-0,9	-	5	С нагрузкой

Интерфейс CAN. Протокол

Виды кадров

Кадр данных (data frame) — передаёт данные;

Кадр удаленного запроса (remote frame) — служит для запроса на передачу кадра данных с тем же идентификатором;

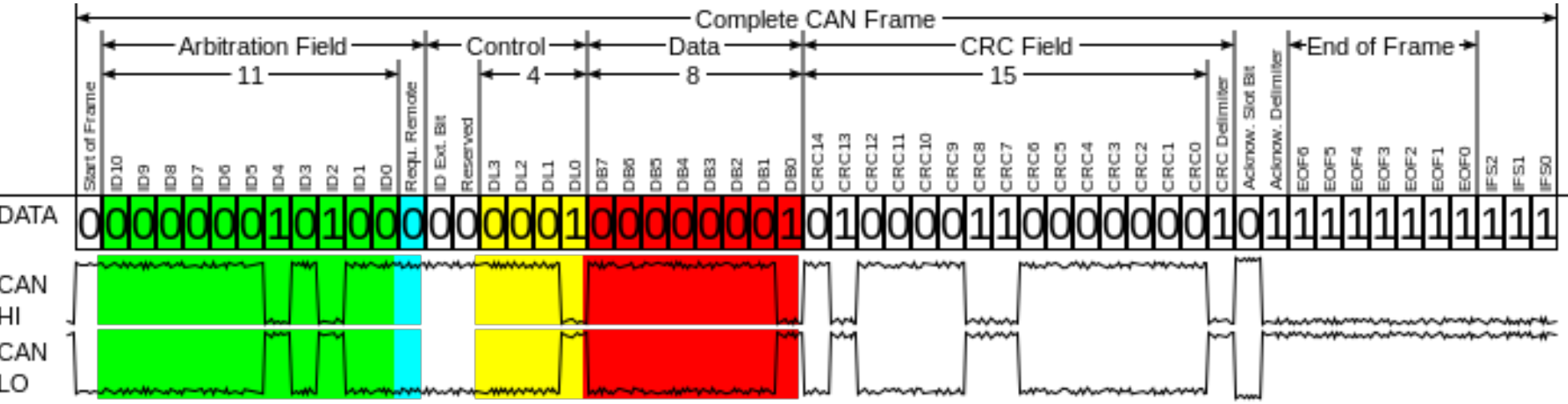
Кадр перегрузки (overload frame) — обеспечивает промежуток между кадрами данных или запроса;

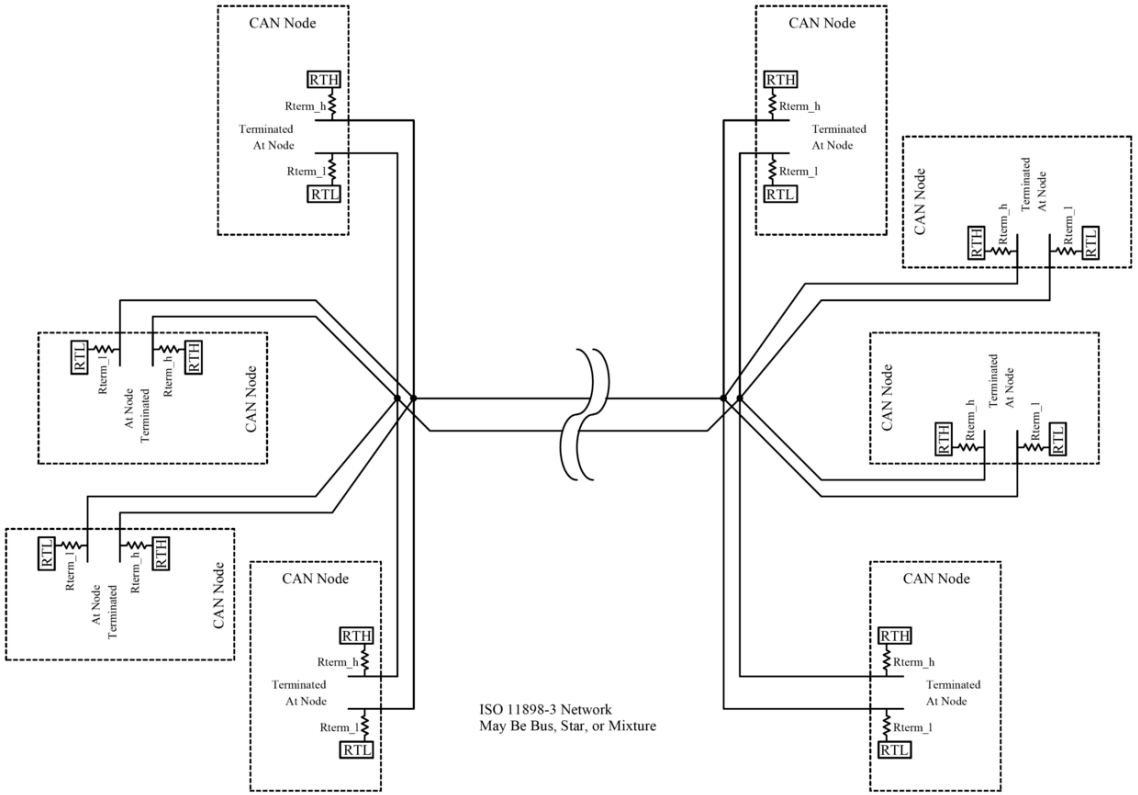
Кадр ошибки (error frame) — передаётся узлом, обнаружившим в сети ошибку.

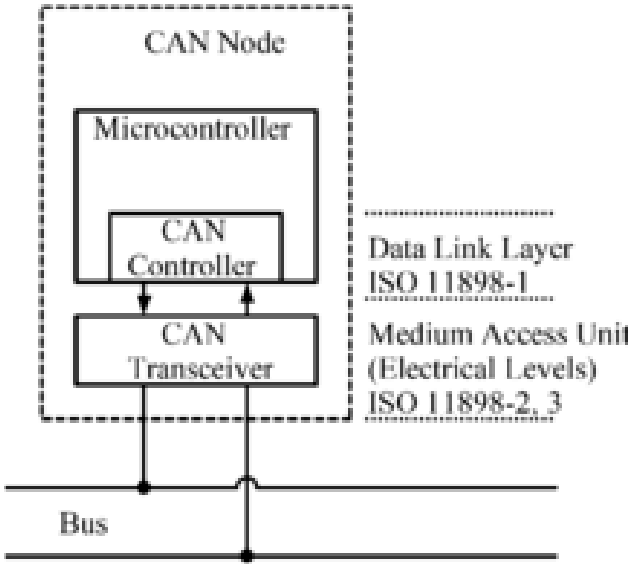
Кадры данных и запроса отделяются от предыдущих кадров *межкадровым промежутком*.

Поле	Длина (бит)	Описание
Начало кадра	1	Сигнализирует начало передачи кадра
Идентификатор	11	Уникальный идентификатор
Запрос на передачу (RTR)	1	Должен быть доминантным
Бит расширения идентификатора (IDE)	1	Должен быть доминантным (определяет длину идентификатора)
Зарезервированный бит (r0)	1	Резерв
Длина данных (DLC)	4	Длина поля данных в байтах (0-8)
Поле данных	0-8 байт	Передаваемые данные (длина в поле DLC)
Контрольная сумма (CRC)	15	Контрольная сумма всего кадра
Разграничитель контрольной суммы	1	Должен быть рецессивным
Промежуток подтверждения (ACK)	1	Передатчик шлёт рецессивный, приёмник вставляет доминанту
Разграничитель подтверждения	1	Должен быть рецессивным
Конец кадра (EOF)	7	Должен быть рецессивным

Интерфейс CAN. Протокол







Интерфейс 1-wire

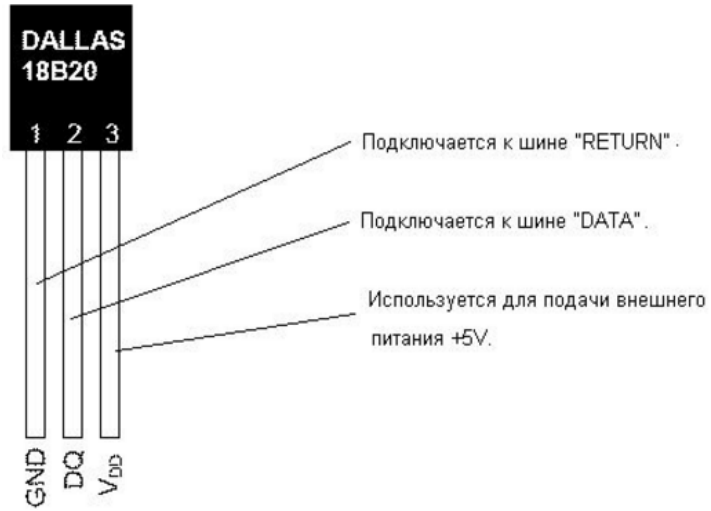
1-Wire (один провод) — двунаправленная шина связи для устройств с низкоскоростной передачей данных (до 125 Кбит/с), в которой данные передаются по цепи питания (то есть всего используются два провода — один для заземления, а второй для питания и данных; в некоторых случаях используют и отдельный провод питания).
Разработан Dallas Semiconductor в конце 90-х. Топология сети — общая шина. Сеть типа MicroLAN - архитектура с одним ведущим шины и многочисленными ведомыми.
Применение: недорогие простые устройства, цифровые термометры и измерители параметров внешней среды; аккумуляторные батареи ноутбуков и сотовых телефонов.

Интегральная схема включает конденсатор ёмкостью 800 пФ для питания от линии данных (так называемое паразитное питание); большое расстояние передачи.
Расстояние до 300 м при условиях:

- применение специального кабеля IEEE1394 (Firewire);
- использование специального драйвера сети (активная подтяжка с учётом тока);
- топология "общая шина" с единым стволом.

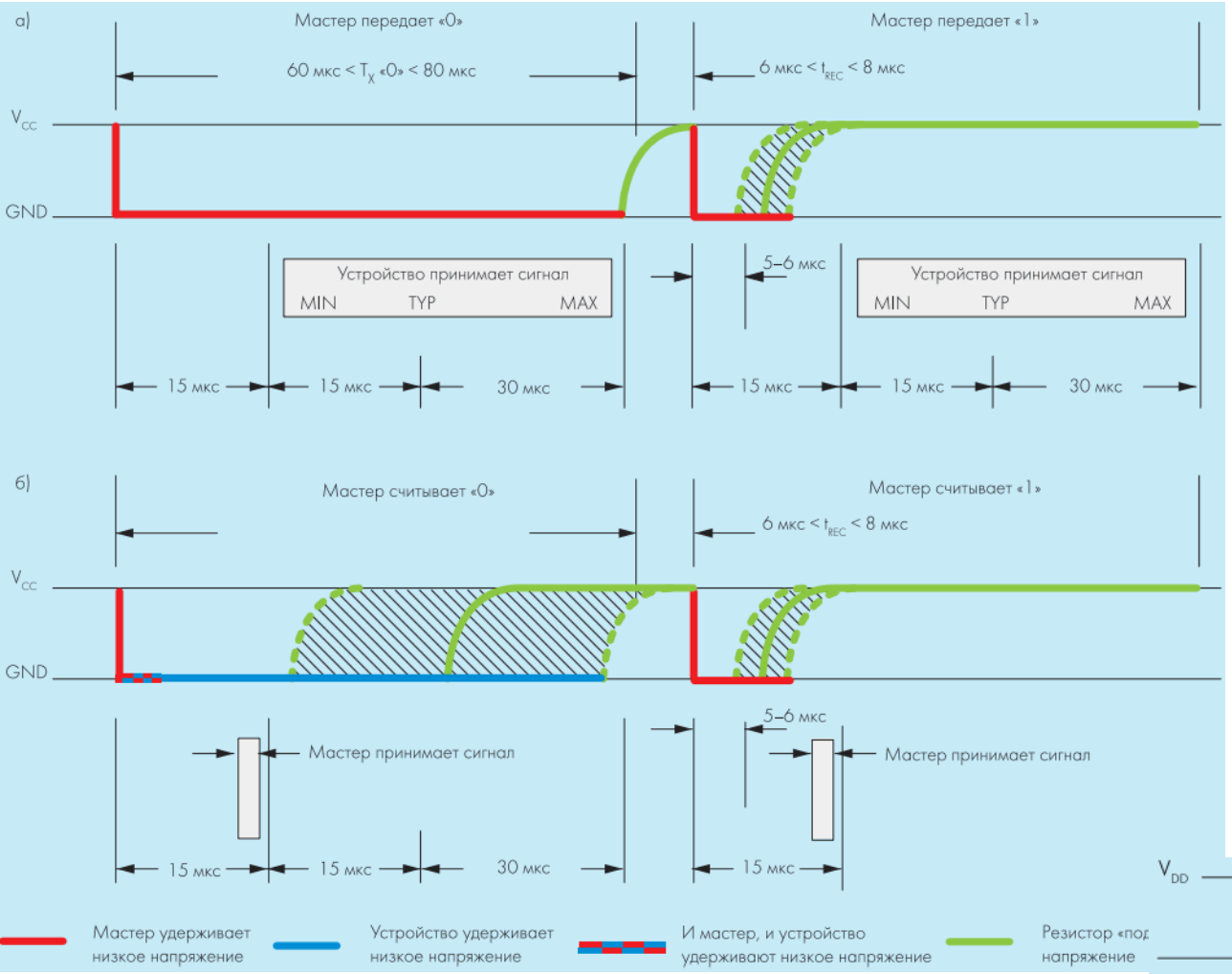
Кабель категории 5 (Cat. 5) — тип кабеля для передачи сигналов, состоящий из 4-х витых пар. CAT-5, Разъем RJ11 двухпроводный.

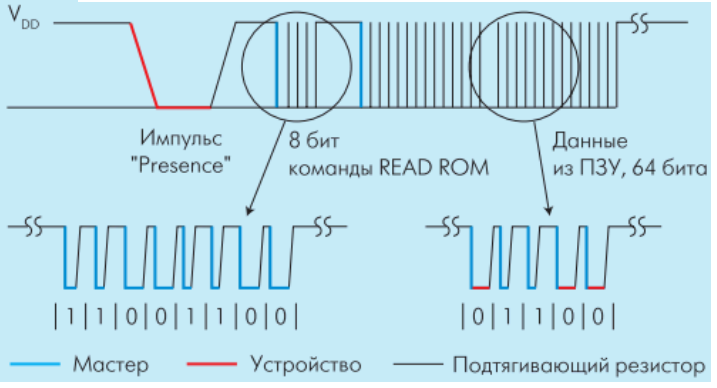
Высокоточный цифровой термометр MicroLAN. от -55° С до +125° С. Считывается код температуры



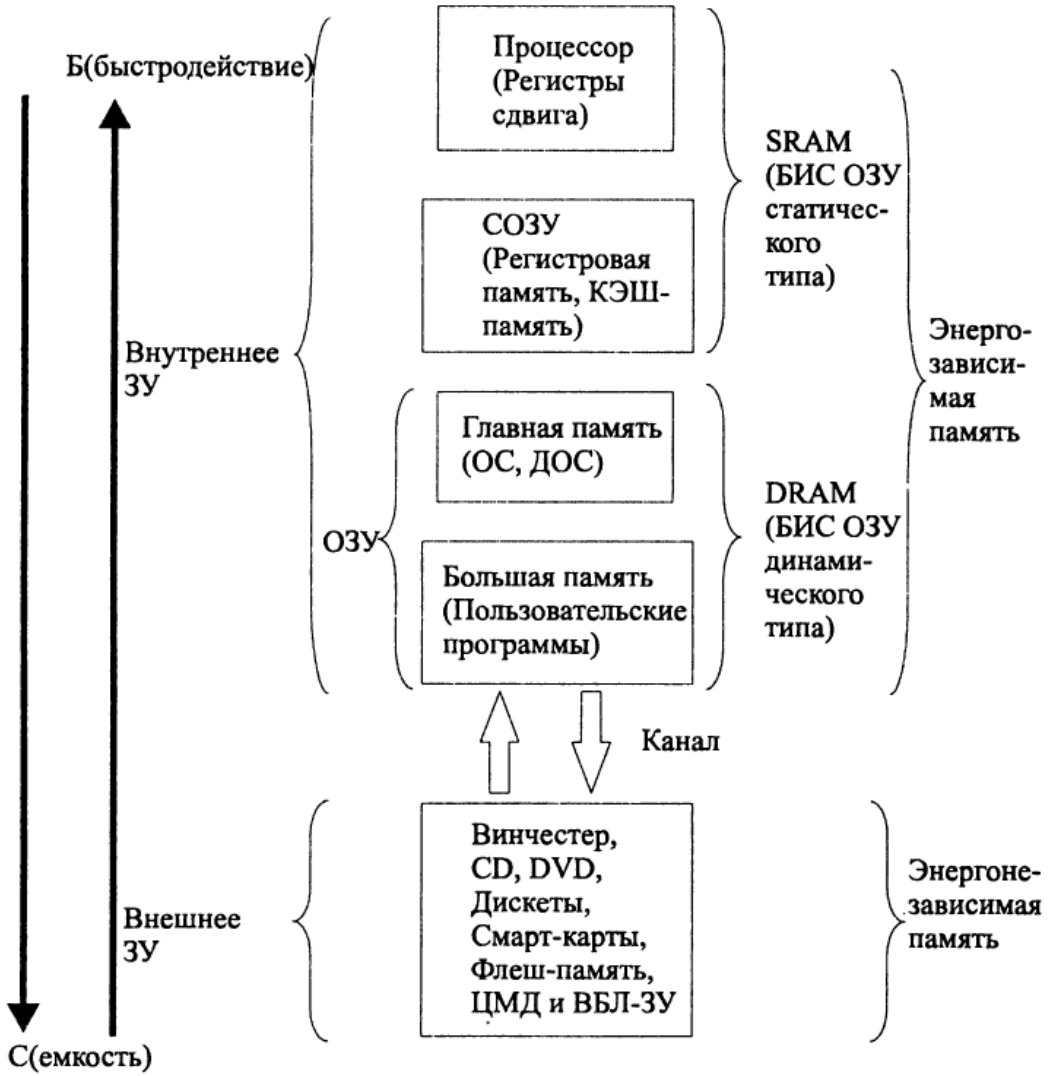
Интерфейс 1-wire протокол

Передача информационных битов по шине 1-Wire: а – мастер передает сигналы, б – мастер считывает сигналы





Иерархия ЗУ



ROM — Read-Only Memory (ЗУ только для чтения или постоянное запоминающее устройство, ПЗУ).

PROM — Programmable ROM (программируемое ПЗУ, ППЗУ).

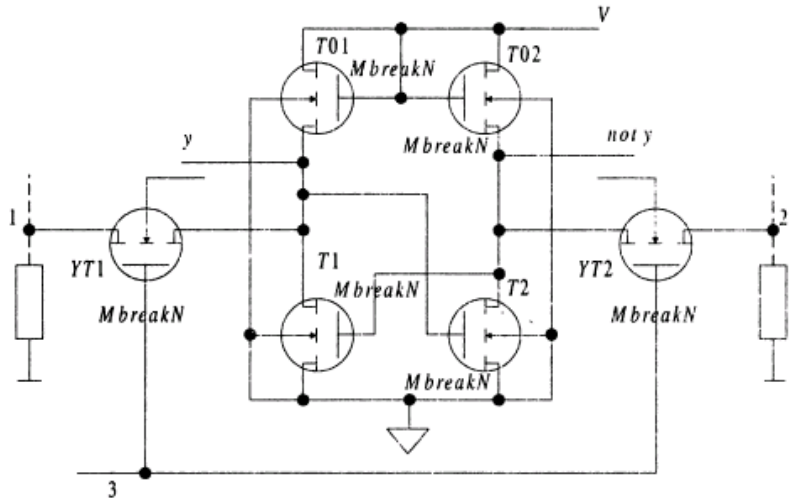
EPROM — Electrically PROM (стираемое программируемое ПЗУ, СППЗУ).

EEPROM — Electrically Erasable PROM (электрически стираемое программируемое ПЗУ).

ЗУ статической памяти

ЗЭ SRAM на n-МОП-транзисторах

На рис. 18.4 изображен простейший вариант ЗЭ на n-МОП-транзисторах или RS-триггер на конъюнкторах с отрицанием.



По входам 1 и 2 управляющих транзисторов (УТ) триггеры или ЗЭ объединены в столбцы. По входу 3 в строки.

О режиме хранения: при $y = 0$ транзисторы $T01$ и $T1$ открыты, $T02$ и $T2$ закрыты, но напряжение $U_{зи}T1$ уменьшается (токи утечки), поэтому в состоянии хранения $T02$ приоткрывается и дозаряжает $U_{зи}T1$.

Для записи надо подать воздействие по всем трем входам:

- для записи "1" подать напряжение U (единичный уровень) на входы 1 и 3 и "0" (нулевой уровень) на вход 2.

  Рассмотрим процесс во времени: был "0", записываем "1", то есть $T01$ и $T1$ были открыты и по ним протекал ток, а $T02$ и $T2$ были закрыты. После подачи импульса сначала $C_{зи}T1 = C_{вх}T1$ разряжается через $YT2$, затем $C_{вх}T2$ заряжается через $YT1$;

- для записи "0" подать напряжение U на входы 2 и 3 и "0" на вход 1.

Для считывания воздействуем U (единичным уровнем) только на 3 вход, а к выходам 1 и 2 подаем "0" через нагрузку.

Транзисторы $YT1$ и $YT2$ используются в режиме двусторонней проводимости, в зависимости от того, записываем мы или считываем, "Сток" и "Исток" меняются местами. Эти транзисторы называются *двунаправленными ключами ввода/вывода*.

ЗЭ SRAM на К-МОП-схемах

Рассмотрим простейший вариант ЗЭ на К-МОП-схемах или RS-триггер на конъюнкторах с отрицанием (рис. 18.5).

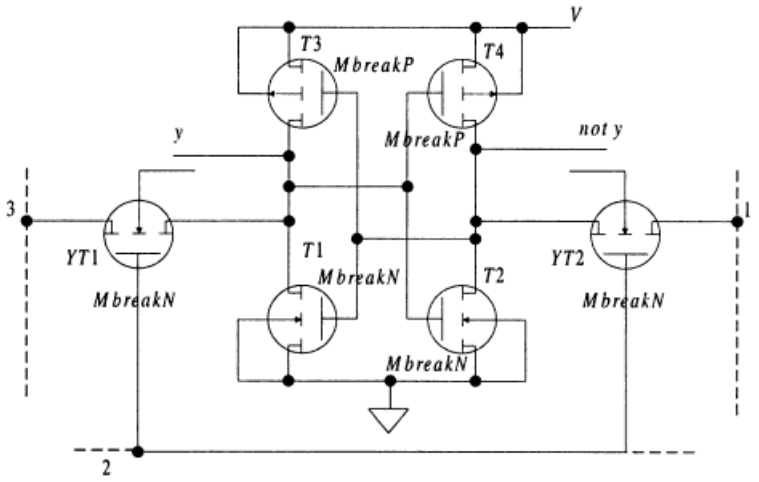


Рис. 18.5. ЗЭ SRAM на К-МОП-схемах

По входам 1 и 3 ЗЭ объединены в столбцы, а по входам 2 — в строки.

Транзисторы УТ по аналогии используются в режиме двусторонней проводимости.

О режиме хранения: здесь транзисторы у конъюнкторов разной проводимости, поэтому когда один транзистор конъюнктора открыт, другой заперт, и емкость $C_{зи}$ подзаряжается от источника через открытый транзистор соседнего конъюнктора.

Для записи "1" на вход 3 подается "0", а на вход 1 и 2 подается "1".

Для записи "0" на вход 1 подается "0", а на вход 3 и 2 подается "1".

Для считывания воздействуем "1" только на 2 вход, а к 1 и 3 входам подключаем "0" через нагрузку.

ЗУ динамической памяти

Эти ЗЭ относятся к элементам БИС ОЗУ динамического типа на МДП-транзисторах. Схема приведена на рис. 18.6.

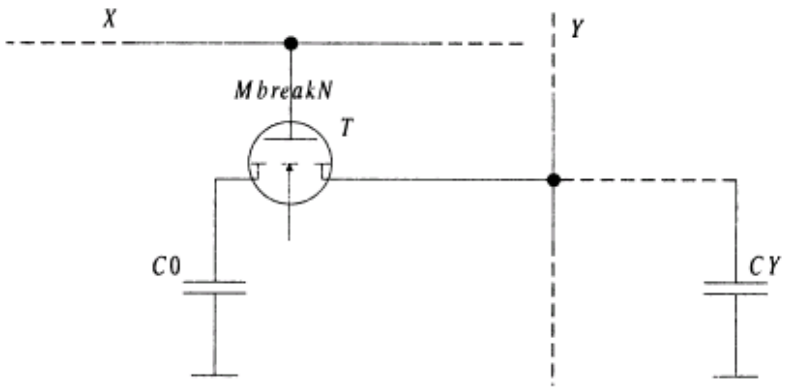


Рис. 18.6. ЗЭ DRAM на МДП-транзисторах

Принцип действия основан на хранении информации на конденсаторах.

Режим хранения: напряжение U на шине строки $X = 0$, поэтому транзистор T закрыт. При закрытом транзисторе T конденсатор $C0$ будет отключен от шины Y, поэтому на $C0$ будет храниться либо напряжение, соответствующее единице $U1$, либо напряжение, соответствующее нулю $U0$. В режиме хранения из-за токов утечки напряжение $UC0$ будет относительно медленно (по сравнению с временами записи и чтения) меняться, поэтому необходима регенерация (периодическое восстановление) напряжений $U1$ или $U0$ на конденсаторе $C0$.

Режим записи: на соответствующую адресу шину Y подается напряжение $U1$, либо $U0$, затем подается положительный импульс выборки на шину X. При этом в остальных элементах выбранной строки X идет регенерация.

Режим считывания: на шину столбца Y подаем напряжение $U_{on}(U0 < U_{on} < U1)$, оно поддерживается емкостью C_y, затем подается импульс выборки на шину X, поэтому транзистор отпирается, и $C0$ и C_y оказываются параллельно включенными и взаимно перезаряжаются, поэтому на шине Y либо $(U_{on} + dU1)$, если считываем "1", либо $(U_{on} + dU0)$, если считываем "0". Затем напряжение на шине Y подаем на усилитель-считыватель (например, четное число инверторов), который формирует либо напряжение $U1$, либо напряжение $U0$.

Регенерация — это считывание, преобразование в напряжение $U1$ или $U0$ усилителем-считывателем и последующая перезапись. Регенерация проходит одновременно для всех элементов строки.

Таким образом, динамические элементы памяти маломощны (потребляют энергию во время регенерации, записи и считывания). Динамическая память (DRAM) имеет высокую информационную емкость БИС в 4—16 раз больше, чем у БИС памяти статического типа (SRAM), но быстродействие у динамической памяти ниже (большое время выборки).

Двух портовая память FPGA

В состав СБИС семейства FLEX10K были впервые включены встроенные блоки памяти ВВП общей емкостью приблизительно от 6 до 20 Кбит для разных представителей семейства. Отдельные блоки емкостью 2 Кбит были размещены в середине каждой строки матрицы логических блоков. 'Блоки встроенной памяти можно использовать как по прямому назначению, т. е. как статическое ЗУ, так и для реализации ПЗУ и логических схем (табличных ФП повышенной размерности путем эмуляции ПЗУ с помощью загрузки таблицы в ОЗУ). Такие ФП дают более эффективные решения в сравнении с реализациями сложных функций средствами типовых логиче- ских блоков. Например, один встроенный блок памяти при организации
256x8 реализует перемножитель 4x4, способный работать на частотах до 50 МГц. Построение такого же перемножителя на типовых ЛБ потребовало бы занять 8 логических блоков, а частота работы перемножителя не превысила бы 20 МГц.
Блоки встроенной памяти ориентированы также на организацию буферов FIFO, а в микросхемах FLEX10KE и на построение двухпортовой памяти. Несколько блоков можно объединять для создания" более емкой памяти. Так как блоки памяти расположены на том же кристалле, что и логическая часть схемы, работа с памятью отличается высоким быстродействием.

Распределенная память FPGA и встроенные блоки памяти

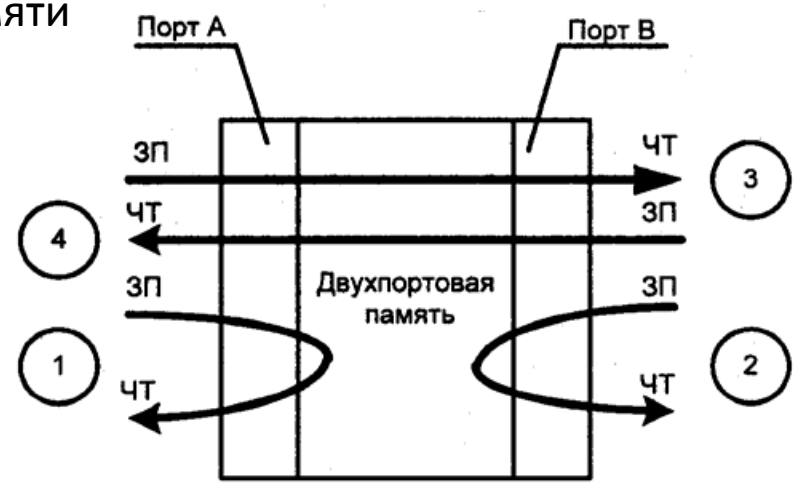


Рис. 25. Операции и потоки данных в двухпортовом ВБП

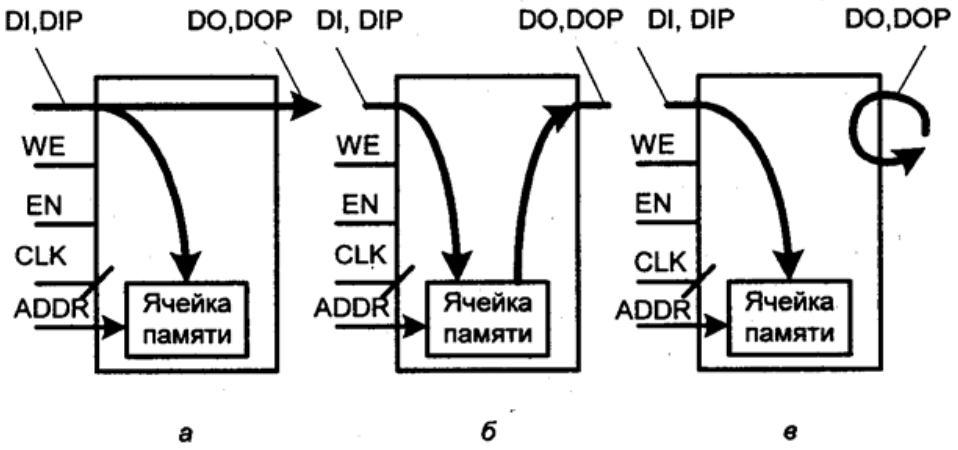
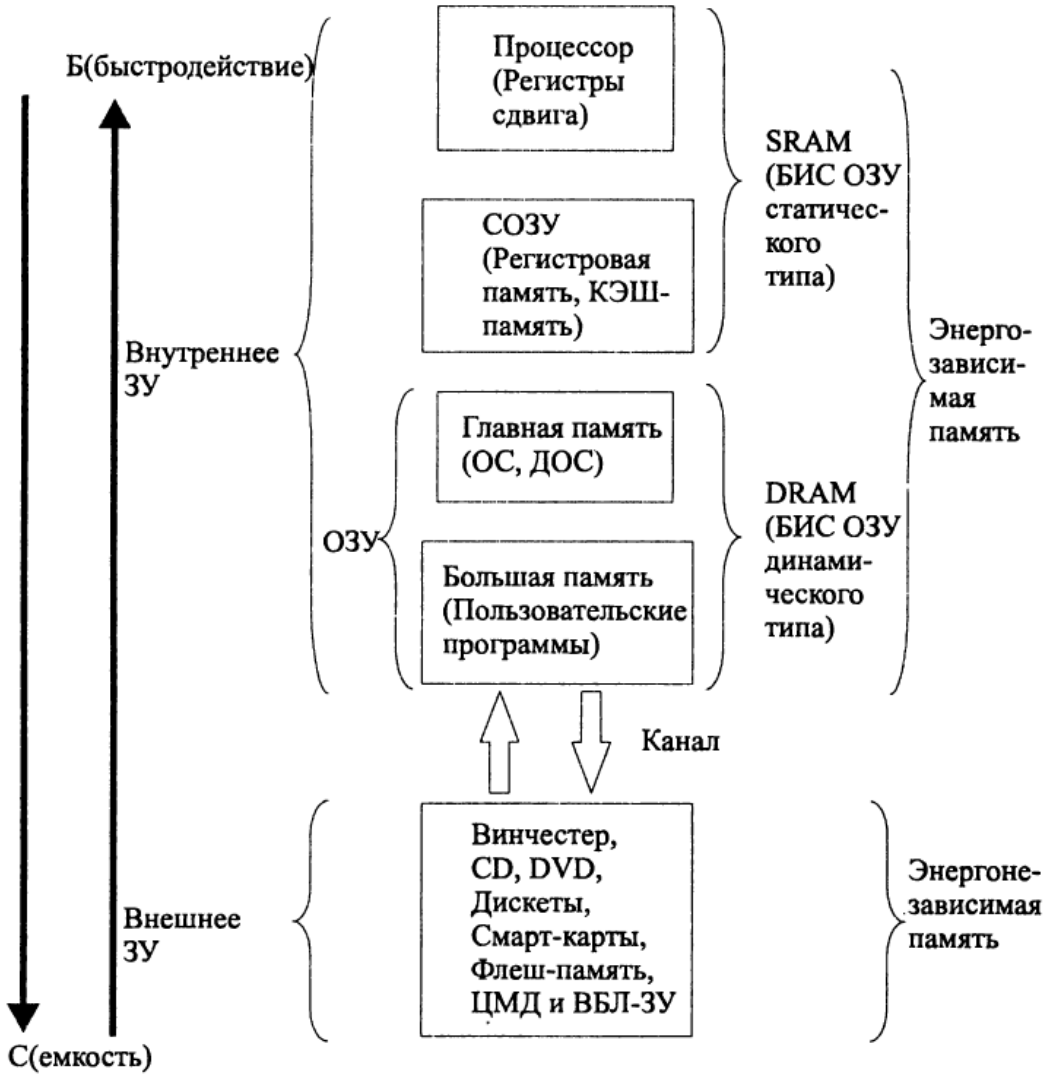


Рис. 26. Варианты поведения двухпортового ВБП при записи данных. Режимы WRITE_FIRST (*а*), READ_FIRST(*б*) и NO_CHANGE (*в*)

Иерархия ЗУ



Применение многопортовых ЗУ
Сетевые устройства с разделяемыми ресурсами и многопроцессорные устройства обработки данных. В качестве примеров можно привести ATM и Ethernet коммутаторы и маршрутизаторы, базовые станции, устройства промышленной автоматики на базе DSP.

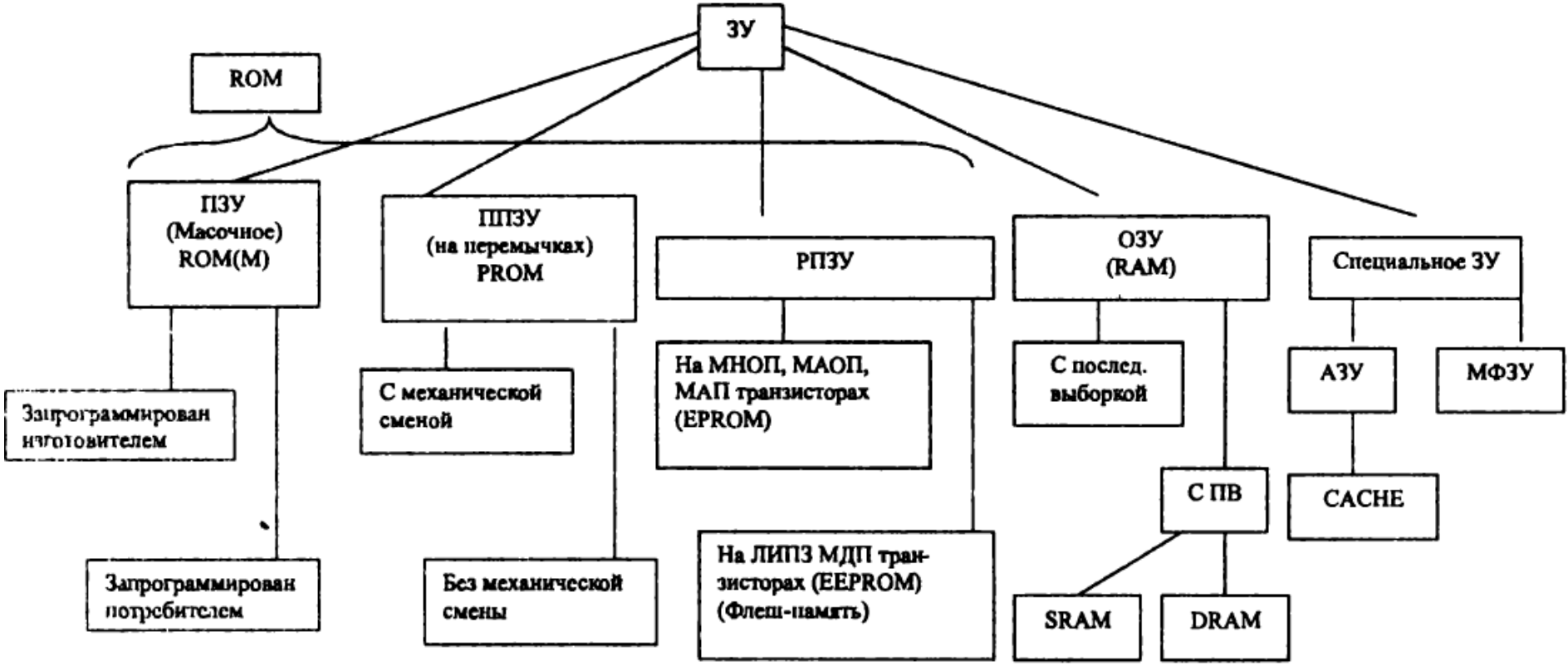
ROM — Read-Only Memory (ЗУ только для чтения или постоянное запоминающее устройство, ПЗУ).

PROM — Programmable ROM (программируемое ПЗУ, ППЗУ).

EPROM — Electrically PROM (стираемое программируемое ПЗУ, СППЗУ).

EEPROM — Electrically Erasable PROM (электрически стираемое программируемое ПЗУ).

Функциональная классификация ЗУ



RAM — Random Access Memory (оперативное запоминающее устройство, ОЗУ).

SRAM — Static RAM (статическое ОЗУ).

DRAM — Dynamic RAM (динамическое ОЗУ).

CACHE Memory (кэш, быстродействующая буферная память небольшой емкости).

ПЗУ — постоянное ЗУ (информация записывается один раз в заводских условиях).

ППЗУ — полупостоянное или программируемое постоянное ЗУ (информация записывается один раз в домашних условиях).

РПЗУ — репрограммируемое ЗУ (здесь запись требует гораздо больше времени и энергии, чем считывание).

ОЗУ — оперативное ЗУ (здесь информацию одинаково легко записывать и считывать).

ПВ — произвольная выборка. Если иначе, то последовательные запись и считывание. ЗУ с ПВ разбивают на участки, в каждом из которых может быть записано определенное число бит (машинное слово или ячейка памяти). Эти участки пронумерованы, а номера называют адресами. Таким образом, записываемую или считываемую информацию направляют по адресам. При ПВ каждый бит может иметь свой адрес.

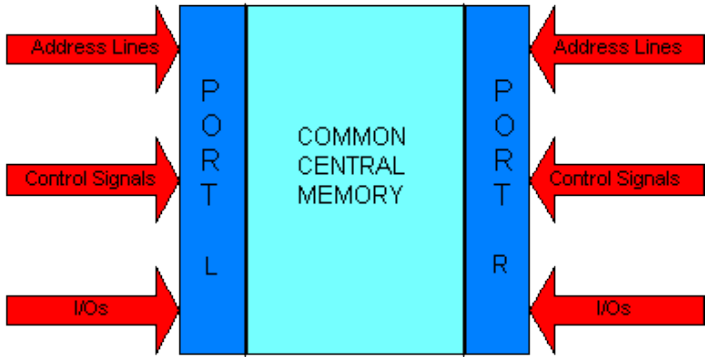
АЗУ — ассоциативные ЗУ. В АЗУ поиск ячейки ЗУ ведется не по адресам, а по специальному признаку, для которого в начале слова отводится несколько разрядов.

МФЗУ — многофункциональное ЗУ. Под многофункциональностью понимают логическую обработку информации перед запоминанием.

Многопортовая память

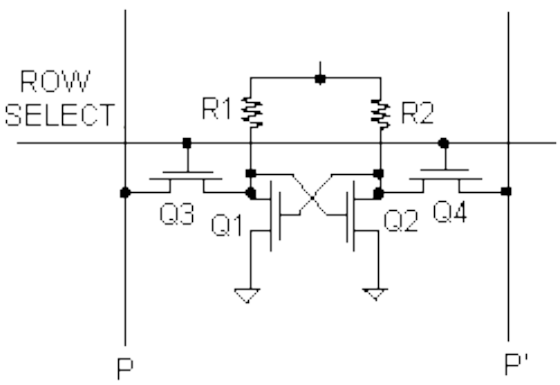
Многопортовая память - это статическое ОЗУ с двумя или более независимыми интерфейсами, обеспечивающими доступ к пространству памяти через разделенные шины адреса, данных и управления.
Единый массив памяти (COMMON CENTRAL MEMORY) и два независимых порта (PORT_L и PORT_R) для обращения к этому массиву. Элементарная ячейка двухпортовой памяти реализована на 6 транзисторах. Основу ячейки составляет статический триггер, выполненный на транзисторах Q1, Q2. Ключевыми транзисторами Q3, Q4 триггер соединен с разрядными шинами P_L, P'_L, а ключевыми транзисторами Q5, Q6 - с разрядными шинами P_R, P'_R. По этим шинам к триггеру подводится при записи и отводится при считывании информация.

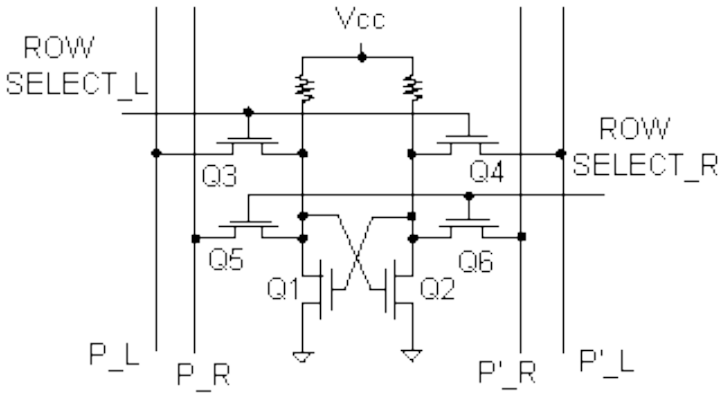
Структура двухпортового статического ОЗУ



Статический элемент обычного и двухпортового ОЗУ

Ключевые транзисторы затворами соединены с шинами выбора строки ROW SELECT_L и ROW SELECT_R соответственно. При возбуждении строки одним из сигналов выборки ключевые транзисторы открываются и подключают входы-выходы триггера к разрядным шинам.



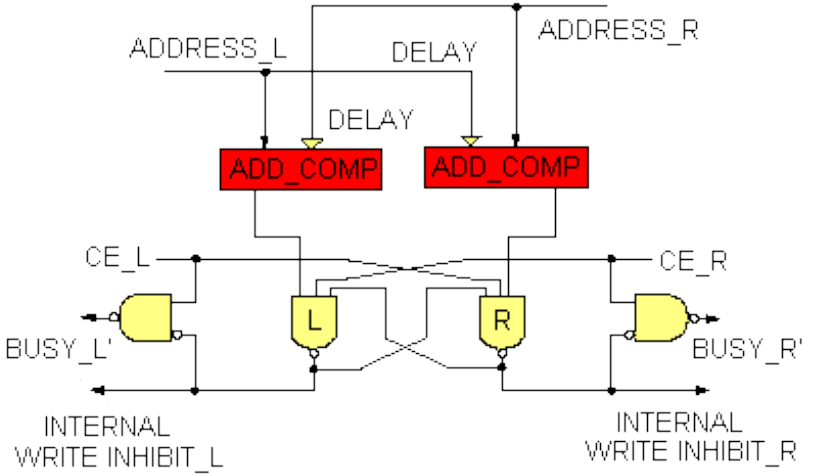


Во всех схемах с асинхронным доступом к общим ресурсам возникают конфликтные ситуации. Конфликты появляются при одновременном обращении двух независимых активных устройств к одной и той же ячейке памяти в процессе выполнения следующих операций: запись через порт L - запись через порт R; запись через порт L - чтение через порт R. При выполнении операции "запись через порт L - запись через порт R" состояние ячейки памяти будет оставаться неопределенным до тех пор, пока одно из активных устройств не завершит обращение к ней и не закончатся переходные процессы. Триггер примет устойчивое состояние, определенное "опоздавшим" устройством. При строго одновременном обращении триггер может принять любое состояние. При выполнении операции " запись через порт L - чтение через порт R" неопределенность существует только в отношении считываемых данных. С одинаковой вероятностью может быть считано как предыдущее значение ячейки памяти, так и вновь записанное в процессе текущего цикла обращения к памяти.
Архитектура двухпортовой памяти предусматривает несколько способов разрешения таких конфликтных ситуаций: с помощью арбитражной логики, семафоров или запросов на прерывания.

Принцип работы асинхронного двухпортового ОЗУ

Арбитражная логика. Арбитр двухпортового ОЗУ устраняет конфликты. Сигналы адресных линий портов ADDRESS_L и ADDRESS_R поступают с двух направлений и, если их значения совпадают, то арбитр посылает одному из активных устройств сигнал BUSY' ("запрет доступа"). BUSY' поступает в опоздавшее к моменту арбитража активное устройство, а при строго одновременных обращениях - в устройство, выбранное случайным образом.

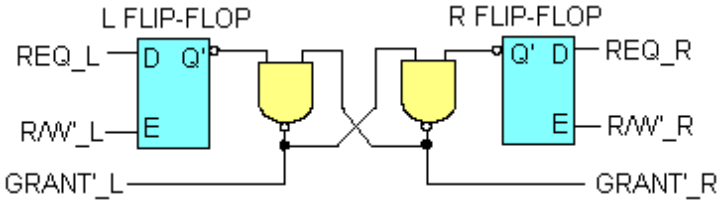
BUSY' удерживается все время, пока не закончится операция обращения к памяти. Дополнительно с BUSY' внутри кристалла формируется сигнал INTERNAL WRITE INHIBIT ("блокировка записи"). При выполнении операции типа "чтение через порт R - чтение через порт L" арбитр также формирует сигналы занятости, но блокирование сигналов чтения не производится и информация считывается одновременно через оба порта. Если адреса запрашиваемых ячеек разные, то доступ к содержимому ячейки памяти также производится одновременно через оба порта, т.к. в этом случае конфликтов не возникает.



Арбитр содержит элементы задержки DELAY, схему сравнения адресных линий ADD_COMP, логические элементы ЗИ-НЕ, соединенные по схеме триггера, логические элементы для формирования сигналов занятости (рис.3). Сигналы CE_L=0 и CE_R=0 вызывают формирование сигналов BUSY_L'=1 и BUSY_R'=1, что соответствует отсутствию запрета доступа к ОЗУ со стороны обоих активных устройств.

Семафоры - это программные арбитры, регулирующие очередность обращения двух или более независимых активных устройств к общему ресурсу. Несколько ячеек памяти, не входящих в рабочее пространство, используются как указатели занятости определенных сегментов (банков) памяти. "0" код в семафоре соответствует занятому банку, а не "0" - свободному.

Схема формирования сигналов занятости банка

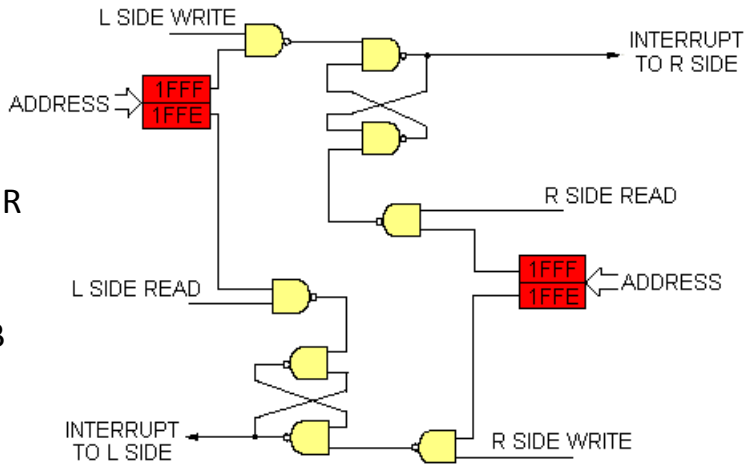


Алгоритм программного арбитража: активное устройство формирует запрос на обращение к банку памяти путем записи "0" в соответствующую ячейку, используемую как семафор; активное устройство считывает состояние семафора, сравнивает полученный код с "0" кодом и, если банк занят (код не "0") переходит в состояние ожидания; если банк свободен, активное устройство получает доступ к его содержимому; активное устройство заканчивает обмен и освобождает занимаемый банк памяти путем записи "1" в соответствующий семафор. Семафорная логика содержит два триггера-защелки и логические элементы 2И-НЕ, соединенные по схеме триггера для формирования сигналов занятости банка GRANT'.

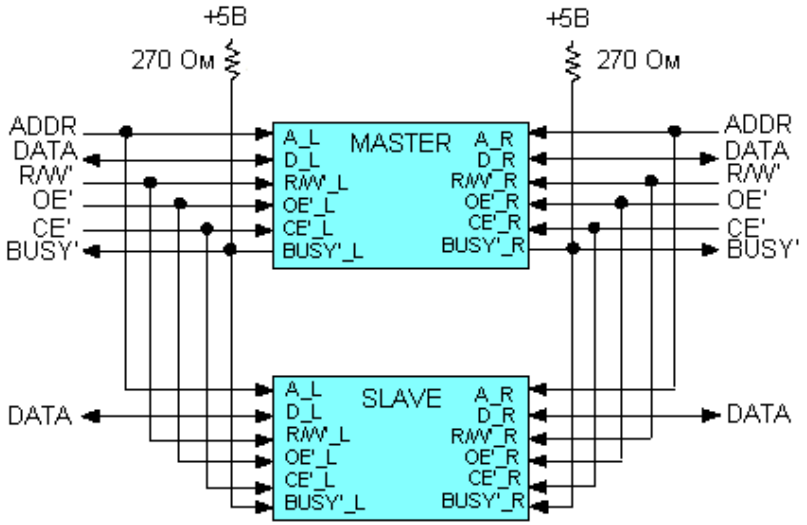
Принцип работы асинхронного двухпортового ОЗУ

Прерывания. Интерфейс системы прерываний асинхронных двухпортовых ОЗУ содержит буфер сообщений и логику формирования запросов на прерывания INTERRUPT TO L(R) SIDE. Например, запрос на прерывание INTERRUPT TO R SIDE формируется в случае записи данных через порт L в ячейку памяти с адресом 1FFFh ("буфер сообщений"). Считывание содержимого этой ячейки памяти через порт R приведет к автоматическому снятию этого запроса. При записи данных через порт R в ячейку памяти с адресом 1FFEh внутрисхемной логикой формируется запрос на прерывания INTERRUPT TO L SIDE. Ячейки, используемые в качестве буферов сообщений, входят в рабочее пространство памяти. В тех случаях, когда обслуживание по прерываниям не требуется, они используются как ячейки общего назначения.

Схема формирования сигналов запросов на прерывания



Наращивание разрядности двухпортовых ОЗУ



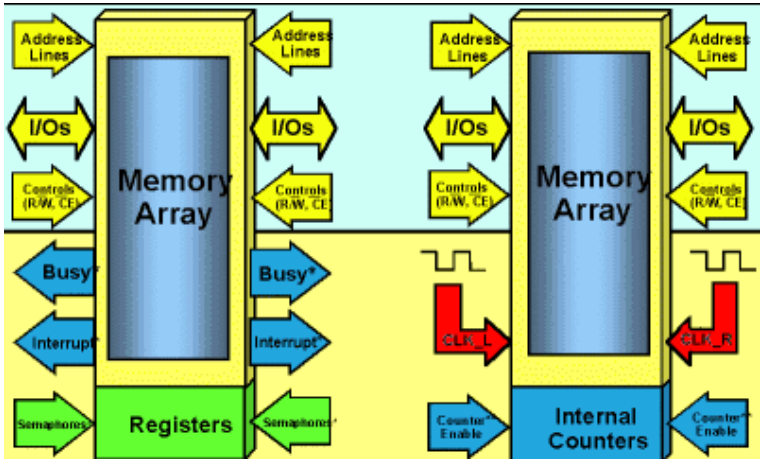
Система ведущий/ведомый. Наращивание емкости двухпортовых ОЗУ достигается путем соединения всех одноименных выводов микросхем, кроме CE' ("выбор кристалла"). Выводы сигналов занятости BUSY в этом случае соединяются по схеме "монтажное ИЛИ". Наращивание разрядности шин данных осуществляется путем соединения всех одноименных входов микросхем, кроме информационных, и характеризуется одной особенностью: с целью предотвращения тупиковых ситуаций (одновременная выдача сигналов занятости для обоих портов) используется система "ведущий/ведомый", предусматривающая применение микросхем двухпортовых статических ОЗУ с различной реализацией арбитражной логики.

Первый тип арбитражной логики носит название "MASTER" и обеспечивает возможность работы микросхем памяти в режимах "обычный" или "ведущий" (формирует сигналы BUSY'_L, BUSY'_R). Второй тип носит название "SLAVE" и обеспечивает возможность работы только в режиме "ведомый" (принимает сигналы занятости, сформированные ведущим устройством).

Синхронное 2-хпортовое ОЗУ (Synchronous Dual-Port RAMs)

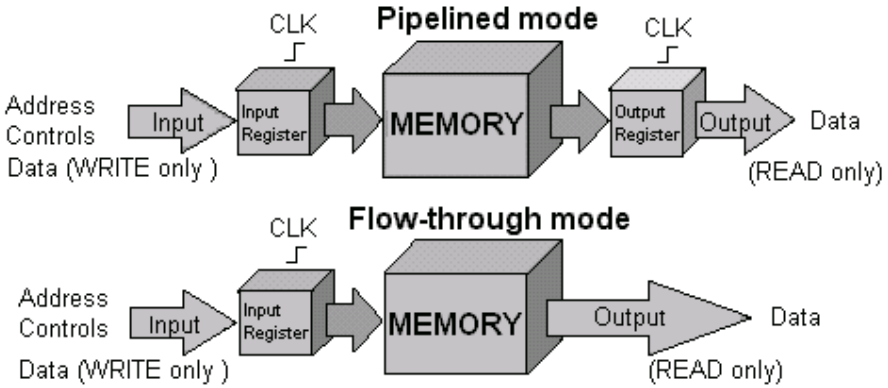
Особенности семейства Synchronous Dual-Port RAM : синхронный интерфейс с раздельными сигналами синхронизации CLK_R и CLK_L и внутренние счетчики (Internal counters) для организации пакетного режима передачи данных . Поскольку обязательным условием доступа активных устройств к пространству такой памяти является их взаимная синхронизация от одного системного таймера, никакой дополнительной логики (арбитраж, семафоры или прерывания) для разрешения конфликтных ситуаций не требуется. Операции обращения к ячейкам асинхронной памяти могут выполняться в произвольные моменты времени при условии соблюдения необходимых временных соотношений между сигналами установки адреса, управления, чтения/записи данных. Операции обращения к ячейкам синхронного двух-портового ОЗУ осуществляются строго под управлением внешнего сигнала синхронизации (CLK_R для порта R и CLK_L для порта L).

Отличие синхронного двухпортового ОЗУ от асинхронного



Архитектура синхронного двухпортового ОЗУ оптимизирована для применения в вычислительных сетях (ATM и Ethernet коммутаторы/маршрутизаторы) и системах беспроводной телефонии (базовые станции), обеспечивая следующие синхронные режимы работы памяти: Pipelined (конвейерный), Flow-through (сквозной) и Burst (пакетный).

Режимы Pipelined и Flow-through отличаются структурой выходного устройства. В Pipelined дополнительный буферный регистр-защелка по выходу (Output register) позволяет организовать конвейерный доступ к данным (одновременно с чтением по предыдущему адресу осуществляется запрос по следующему). Это позволяет сократить общее время обращения к памяти. Недостатком этого режима является задержка на один период сигнала синхронизации при считывании первого слова. Во втором случае (режим Flow-through) считываемые данные непосредственно поступают на выходную шину микросхемы памяти (Output). Это позволяет обеспечить минимальную задержку при считывании первого слова. Однако все последующие обращения к памяти в этом режиме будут проходить за более длительное время, чем в режиме Pipelined. Режимы Flow-through и Pipelined задаются пользователем аппаратно.

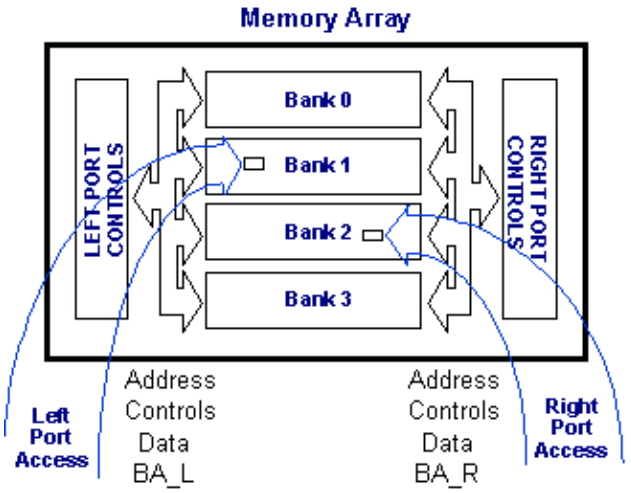


Режим Burst предназначен для выполнения операций над последовательными потоками параллельных данных (например, потоки речевых сообщений) и имеет некоторое сходство с работой памяти FIFO. Работа в этом режиме начинается с параллельной загрузки начального значения внутреннего счетчика через внешнюю шину адреса. В дальнейшем при каждом обращении к памяти состояние внутреннего счетчика циклически инкрементируется. Наличие в синхронном двухпортовом ОЗУ счетчиков адреса позволяет освободить ресурсы управляющего процессора для других операций.

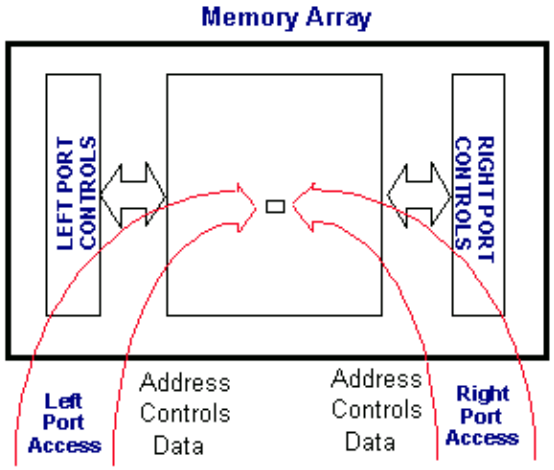
Двухпортовое ОЗУ с переключаемыми банками памяти (Bank-Switchable Dual-Port RAMs)

Bank-Switchable - это статическое ОЗУ с двумя интерфейсами, обеспечивающими независимый доступ к банкам в пространстве памяти. По структуре это промежуточный вариант между обычным ОЗУ и двухпортовой памятью. Как и в обычном ОЗУ элементарная ячейка выполнена на 4-х транзисторах. А наличие 2-х интерфейсов с разделенными шинами адреса, данных и управления обеспечивает возможность независимого обращения к банкам в пространстве памяти по аналогии с тем, как это происходит в стандартной 2-хпортовой памяти. Выбор банка осуществляется посредством установки адреса банка на дополнительных линиях BA_R или BA_L.

Микросхемы памяти Bank-Switchable подразделяются на асинхронные и синхронные. Конфликтные ситуации в асинхронной памяти с переключаемыми банками разрешаются с помощью семафорной логики и системы прерываний. Синхронные устройства Bank-Switchable являются могут иметь доп. функции: поддержку организации обмена данными между шинами с разным форматом слова, наличие внутренних счетчиков адреса, работа в режимах Pipelined и Flow-through.



а)

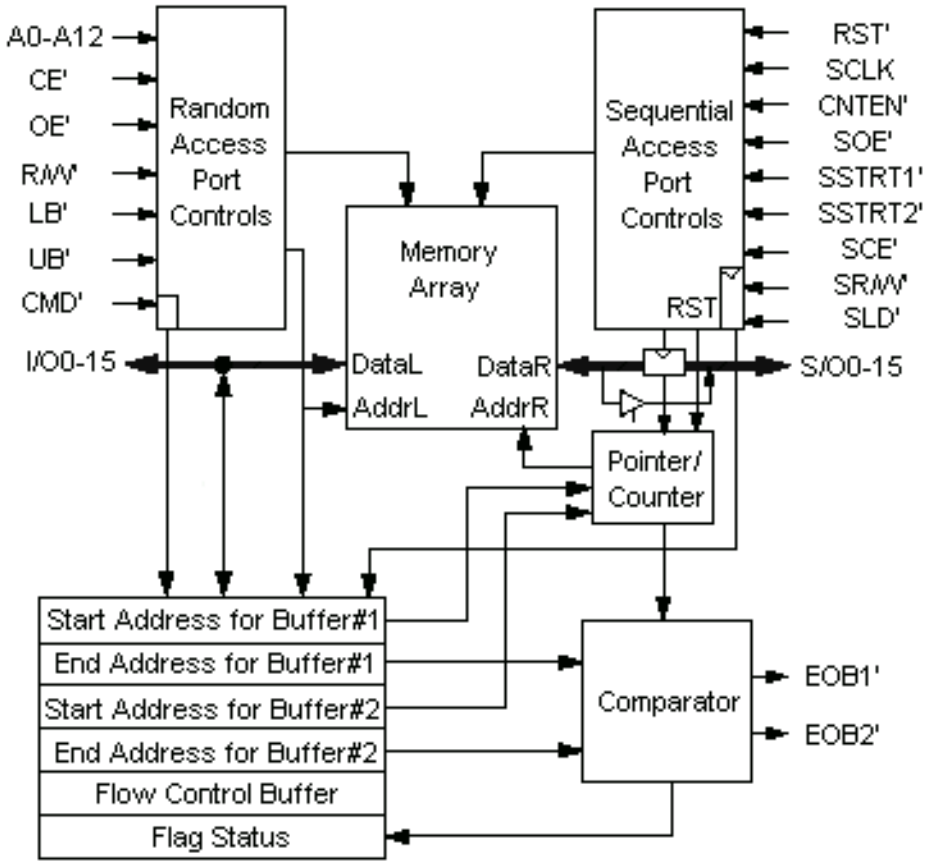


б)

Память с переключаемыми банками (а) и стандартная двухпортовая память (б)

Двухпортовое ОЗУ с последовательно-произвольным доступом (SARAM)

SARAM первая среди микросхем многопортовой памяти, в состав которой был введен синхронный интерфейс. Представляет собой специализированное двухпортовое ОЗУ, обеспечивающее возможность работы с двумя различными типами интерфейсов: асинхронный интерфейс с произвольным доступом, с одной стороны, и синхронный интерфейс с последовательным доступом к данным, с другой стороны. Асинхронный включает стандартные сигналы интерфейса SRAM (шину адреса, шину данных, сигналы управления CE', OE' и R/W'), за исключением сигнала CMD' (разрешение доступа к содержимому внутренних регистров конфигурации синхронного интерфейса). При 0 на вх CMD' под управлением A0-A2 и R/W' осуществляется чтение/запись содержимого одного из шести внутренних регистров: указатель начального адреса буф#1, указатель конечного адреса буф#1, указатель начального адреса буфера#2, указатель конечного адреса буфера#2, регистр режима, регистр состояния. Указатели позволяют разбить общее пространство памяти на два произвольных подпространства с символическими именами "буф#1" и "буф#2".



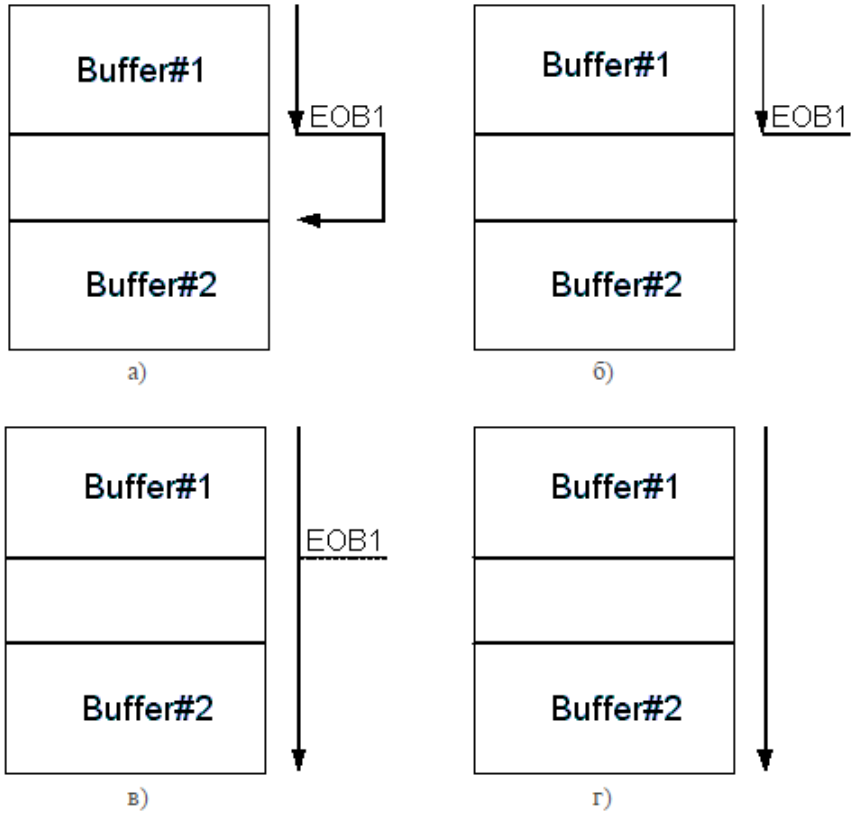
Организация синхронного интерфейса предусматривает возможность повторного считывания данных из памяти. SSTRT1'(SSTRT2') синхронного интерфейса обеспечивают загрузку начального значения адресного указателя буфера#1(буфера#2) во внутренний счетчик адреса под управлением SCLK при 0 на вх. SSTRT1' (SSTRT2'). Содержимое счетчика может быть изменено и путем внешней загрузки данных через шину S/O. При 0 на вх SLD' под управлением SCLK запись данных во входной регистр синхронного интерфейса. По следующему такту SCLK эти данные переносятся во внутренний счетчик адреса. *Недостатком* двухпортовой памяти SARAM является отсутствие арбитражной логики. Поэтому в алгоритме работы управляющего процессора обязательно должна быть предусмотрена процедура выявления и разрешения конфликтных ситуаций в процессе обмена данными

Двухпортовое ОЗУ с последовательно-произвольным доступом (SARAM)

Регистр режима позволяет запрограммировать последовательность операций, выполняемых в случае достижения внутренним счетчиком конечного адреса текущего буфера.

Режим	Краткие характеристики
BUFFER CHAINING	Установка флага "EOB1(EOB2)" и переход к начальному адресу буфера#2(буфера#1)) (рис.5,а);
STOP MODE	Установка флага "EOB1(EOB2)" и запрет инкрементирования содержимого внутреннего счетчика (рис.5,б);
LINEAR MODE	Установка флага "EOB1(EOB2)" и продолжение циклического инкрементирования содержимого внутреннего счетчика (рис.5,в);
MASK MODE	Циклическое инкрементирование содержимого внутреннего счетчика без изменения состояния регистра статуса (рис.5,г).

Режимы работы синхронного интерфейса



Области применения синхронной 2-хпортовой памяти: вычислительные сети (ATM и Ethernet коммутаторы/маршрутизаторы) и беспроводная телефония (базовые станции). Микросхемы двухпортовой памяти с переключаемыми банками предназначены для применения в системах цифровой обработки изображений, системах промышленной автоматики и периферийных контроллерах, оборудование для сетей передачи данных (мосты, маршрутизаторы, фреймеры и др.)

Основой большинства вышеперечисленных устройств являются высокопроизводительные RISC-контроллеры и ЦСП. Главными требованиями, предъявляемым к характеристикам SARAM, являются высокие быстродействие и скорость передачи данных.