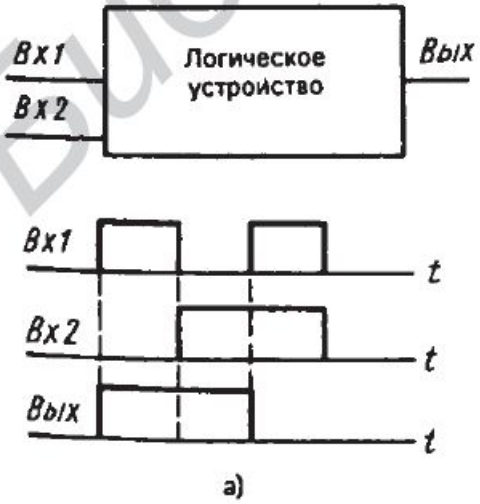
Проектирование микропроцессорных систем

к.т.н., доцент
Аминев Дмитрий Андреевич
aminev.d.a@ya.ru +79067406453

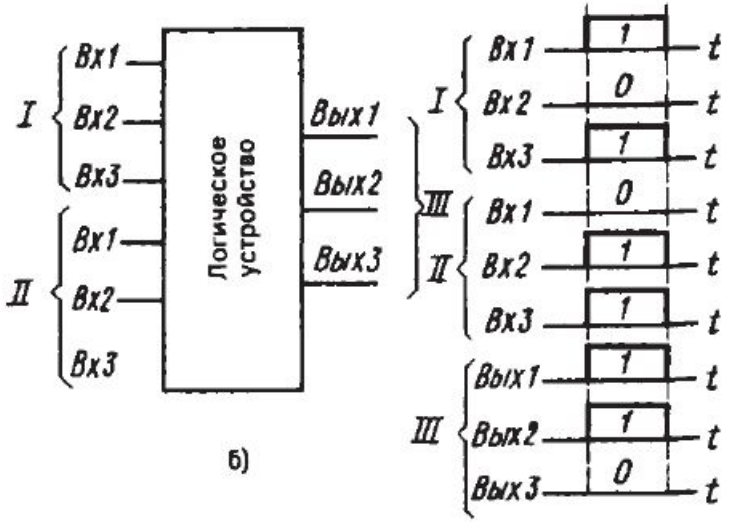
Основы цифровой схемотехники

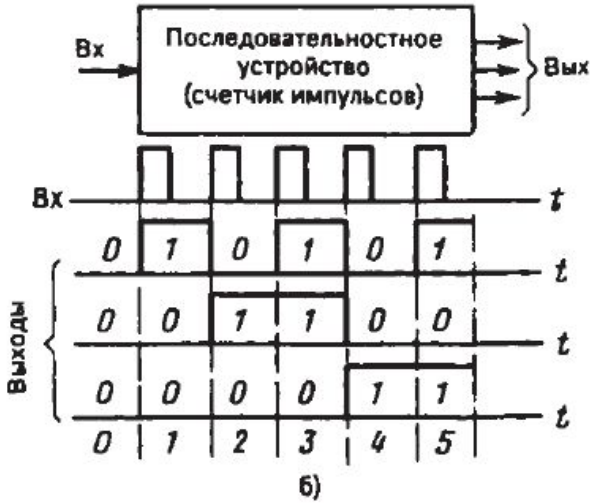
Устройства, предназначенные для формирования функций алгебры логики, называются *логическими устройствами* или *цифровыми устройствами*. Цифровые устройства (либо их узлы) можно делить на типы по различным признакам.

Последовательные



Параллельные

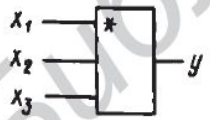




Логические элементы

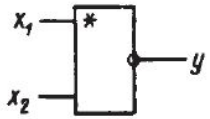
Логический элемент

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



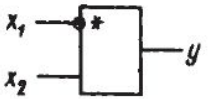
Элемент с инверсным выходом

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



Элемент с инверсным входом

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

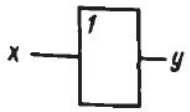


Здесь * — указатель функции, выполняемой логическим элементом.

Обозначение элементов, реализующих логические функции

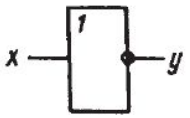
Повторитель

$y = x$



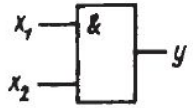
Инвертор (НЕ)

$y = \overline{x}$



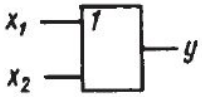
Конъюнктор (И)

$y = x_1 \cdot x_2$



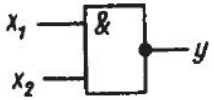
Дизъюнктор (ИЛИ)

$y = x_1 \vee x_2$



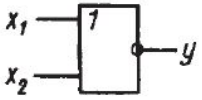
Элемент И-НЕ (элемент Шеффера)

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



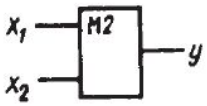
Элемент ИЛИ-НЕ (элемент Пирса)

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



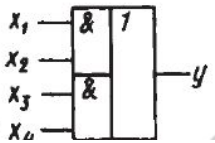
Сложение по модулю 2

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



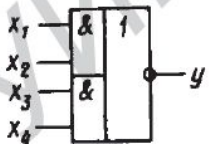
Элемент И-ИЛИ

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



Элемент И-ИЛИ-НЕ

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



Синтез КУ в базисе ИЛИ_НЕ и И-НЕ

КНФ - формула имеет вид конъюнкции дизъюнкций

ДНФ

$\neg A \wedge (B \vee C)$

СДНФ - это такая ДНФ, которая

$(A \wedge B) \vee \neg A$

удовлетворяет трём условиям:

в ней нет одинаковых элементарных конъюнкций; в каждой конъюнкции нет одинаковых пропозициональных букв

каждая элементарная конъюнкция с

$f(x_1, x_2, x_3) = (\overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3}) \vee (\overline{x_1} \wedge \overline{x_2} \wedge x_3)$

пропозициональную букву из входящих в данную ДНФ пропози

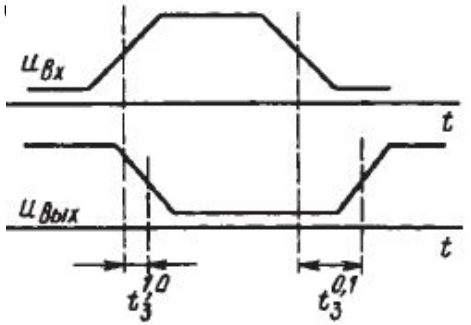
$x_1 \vee x_2 \vee \overline{x_3} \vee \overline{x_4}$

з, причём в одинаковом порядке.

СКНФ

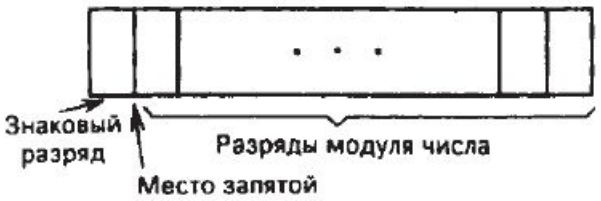
Таблица истинности

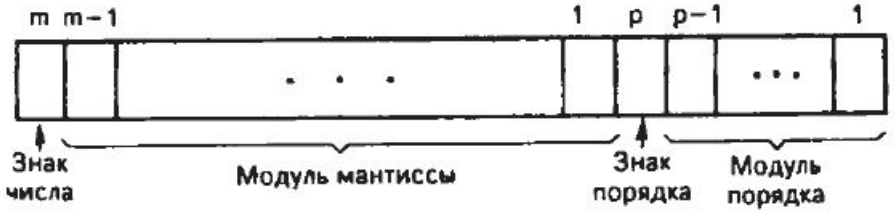
Минимизация методами Петрика, Карно, Вей



Представление чисел в цифровых устройствах

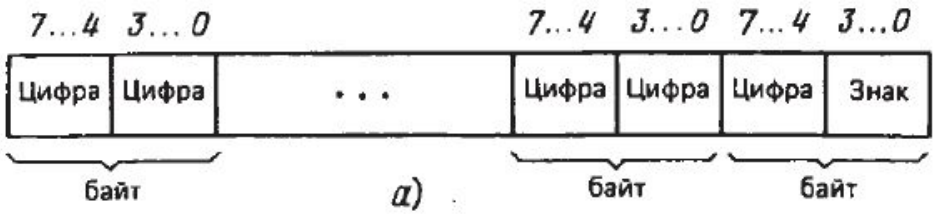
Числа в цифровых устройствах могут представляться в форме целых чисел, чисел с фиксированной запятой и чисел с плавающей запятой.



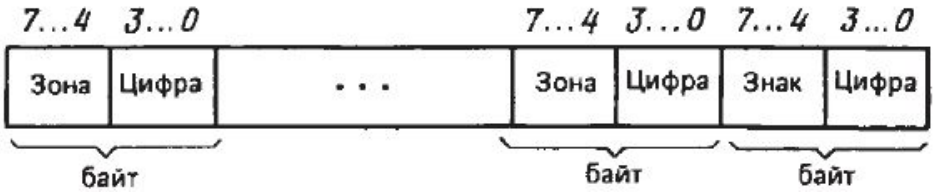


$N = M \cdot 2^{П}$, где M и $П$ — мантисса и порядок числа

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



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 $\overline{Q} = 1$, 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

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	$\overline{Q}_0$

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

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*;

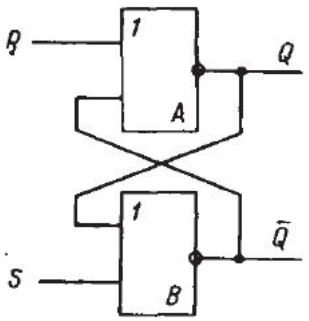
J — *вход установки универсального триггера в состояние 1*;

T — счетный вход;

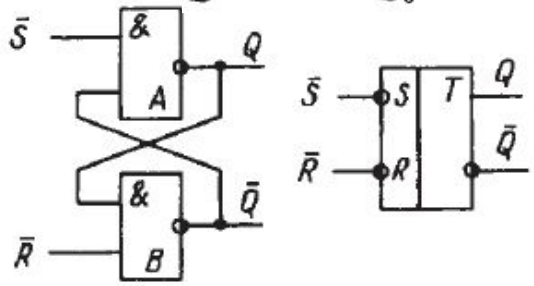
D (от англ. Delay) — *информационный вход установки триггера в состояние, соответствующее логическому уровню на этом входе*;

C — *управляющий (синхронизирующий) вход.*

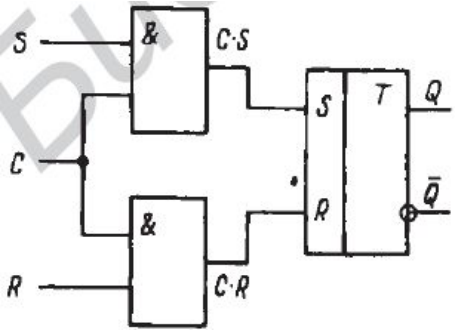
Наименование триггера определяется типами его входов. Например, RS-триггер — триггер, имеющий входы типов R и S.



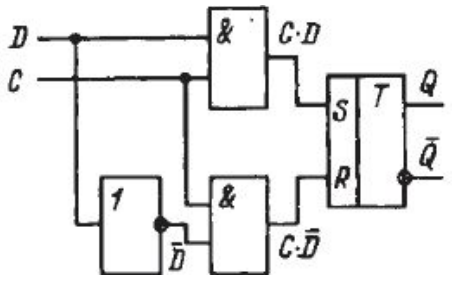
$Q = S \vee \overline{R} \cdot Q_0$

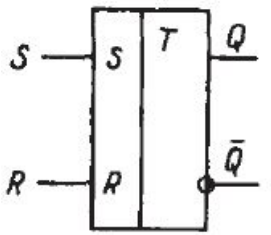


$Q = \overline{C} \cdot Q_0 \vee C \cdot (S \vee \overline{R} \cdot Q_0)$



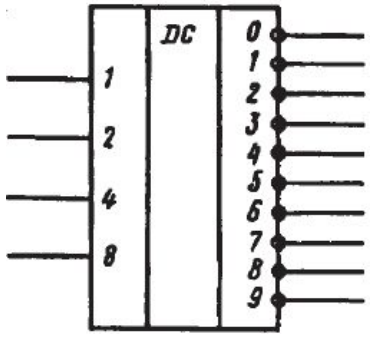
$Q = \overline{C} \cdot Q_0 \vee C \cdot D$

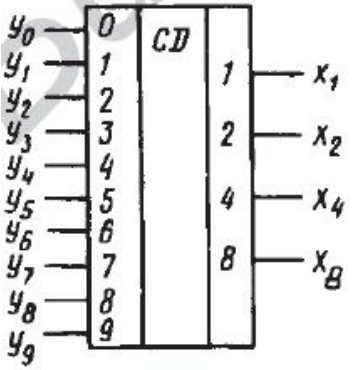




Шифраторы, дешифраторы, мультиплексоры, демультиплексоры

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





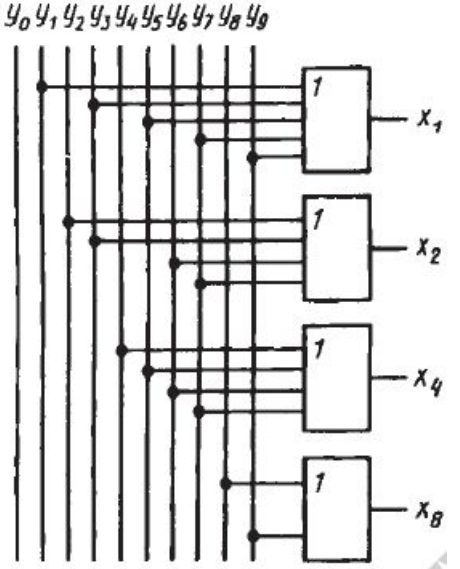
$$x_1 = y_1 \vee y_3 \vee y_7 \vee y_9.$$

Для остальных выходов

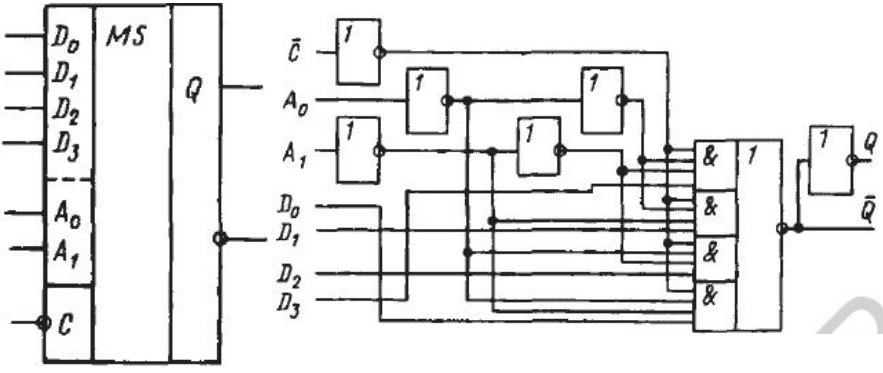
$$x_2 = y_2 \vee y_3 \vee y_6 \vee y_7,$$

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

$$x_8 = y_8 \vee y_9.$$



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.

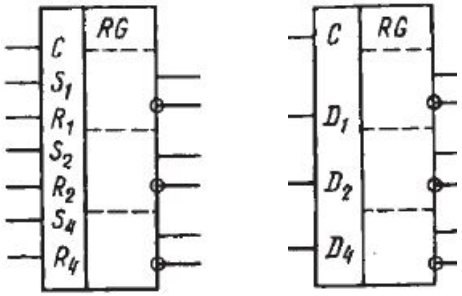


$$Q = (D_0 \cdot \overline{A}_1 \cdot \overline{A}_0 \vee D_1 \cdot \overline{A}_1 \cdot A_0 \vee D_2 \cdot A_1 \cdot \overline{A}_0 \vee D_3 \cdot A_1 \cdot A_0) \cdot C.$$

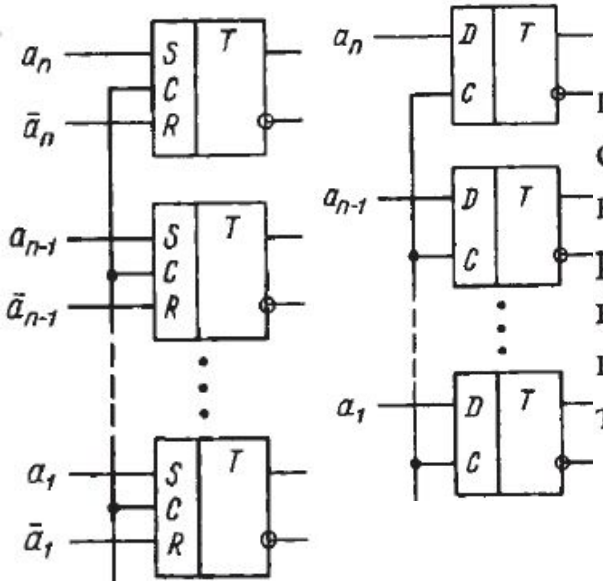
Адресные входы		Стробирующий сигнал	Выход
A_1	A_0	C	Q
×	×	0	0
0	0	1	D_0
0	1	1	D_1
1	0	1	D_2
1	1	1	D_3

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

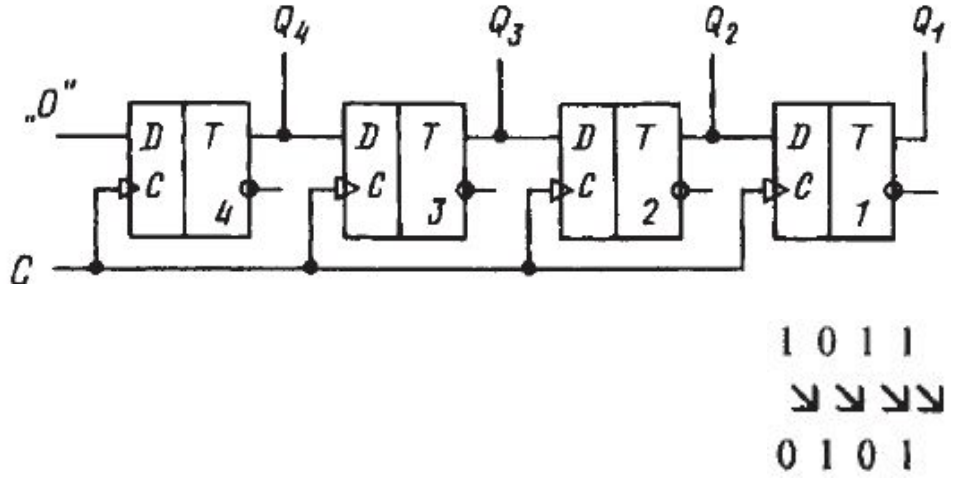


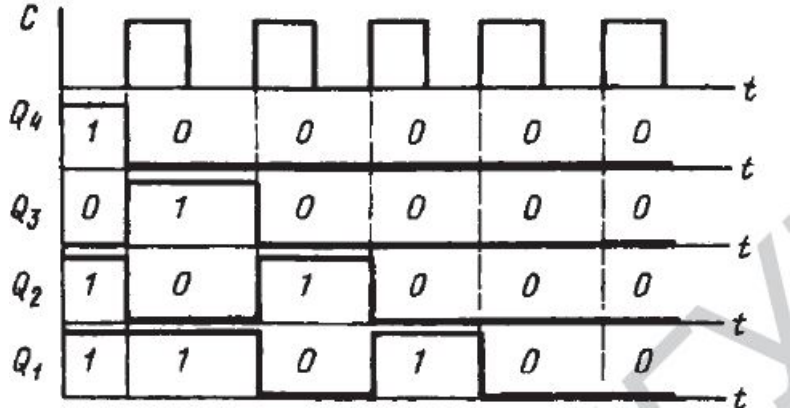
Основная функция регистров — хранение одного многоразрядного числа. При этом число должно быть представлено в двоичной системе счисления или в любой другой системе, но с двоичным представлением цифр разрядов (т.е. в любой двоично-кодированной системе счисления).



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

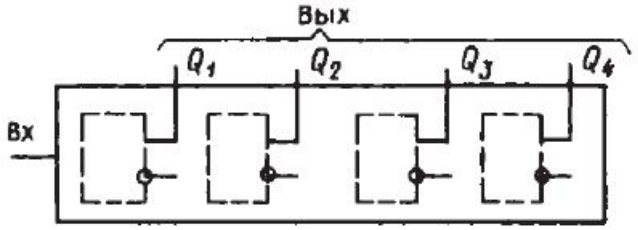




Счетчики, сумматоры

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

$N = 2^n - 1$



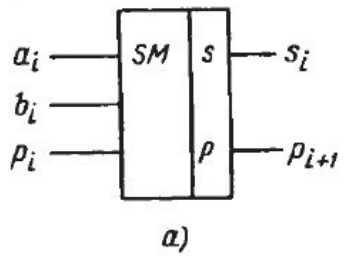
Делитель частоты — устройство, которое при подаче на его вход периодической последовательности импульсов формирует на выходе такую же последовательность, но имеющую частоту повторения импульсов, в некоторое число раз меньшую, чем частота импульсов входной последовательности.

Число поступивших импульсов	Состояние триггеров Q_4	Q_3	Q_2	Q_1	Число поступивших импульсов	Состояние триггеров 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

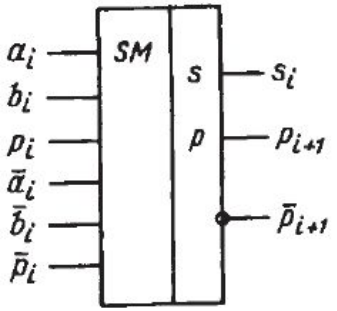
Одноразрядный двоичный сумматор. В каждом из разрядов определяется цифра суммы путем сложения по модулю 2 цифр слагаемых и поступающего в данный разряд переноса и формируется перенос, передаваемый в следующий разряд.

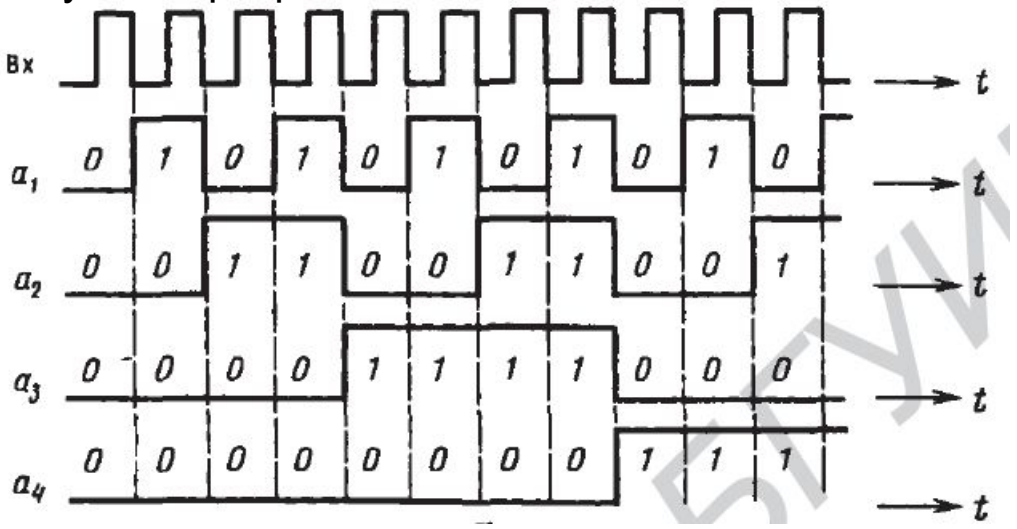
$$s_i = \overline{\bar{a}_i \cdot \bar{b}_i \cdot \bar{p}_i \vee a_i \cdot b_i \cdot \bar{p}_i \vee \bar{a}_i \cdot b_i \cdot p_i \vee a_i \cdot \bar{b}_i \cdot p_i}$$

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

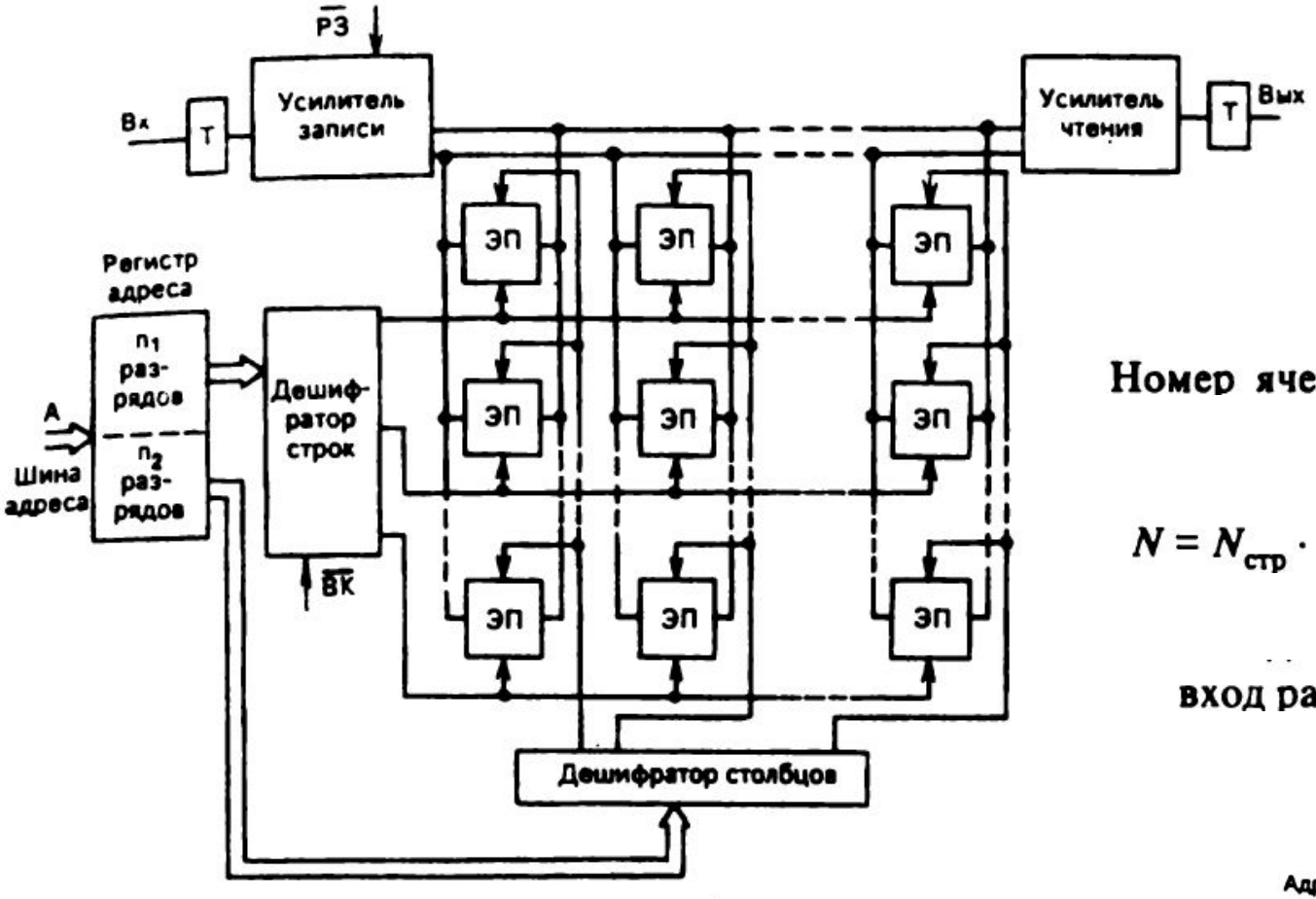


а)





Память



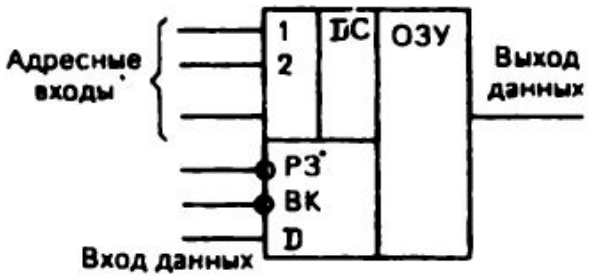
$$N = 2^m$$

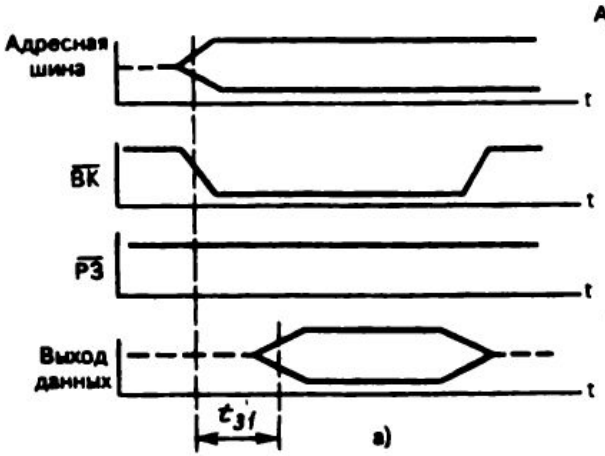
$$M = N \cdot n \text{ бит}$$

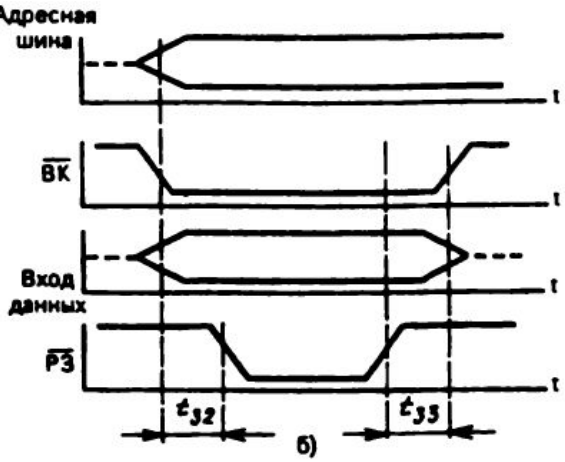
Номер ячейки называется *адресом*

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

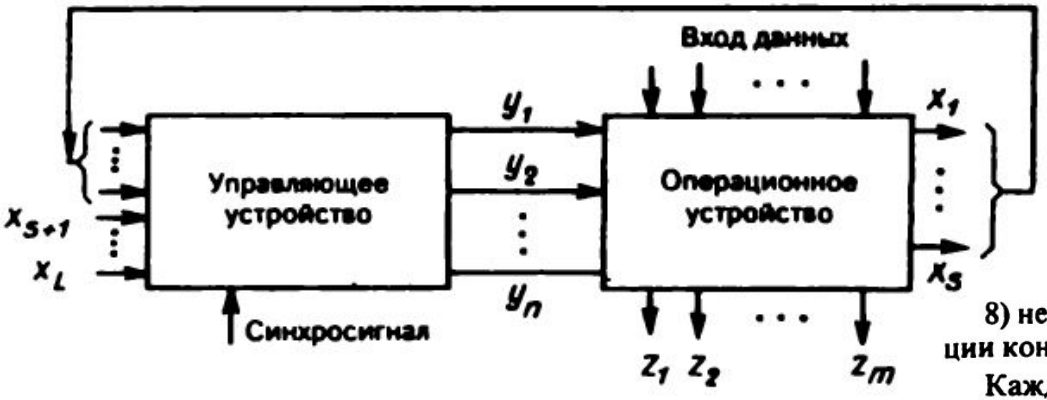
вход разрешения записи ($\overline{РЗ}$)







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.

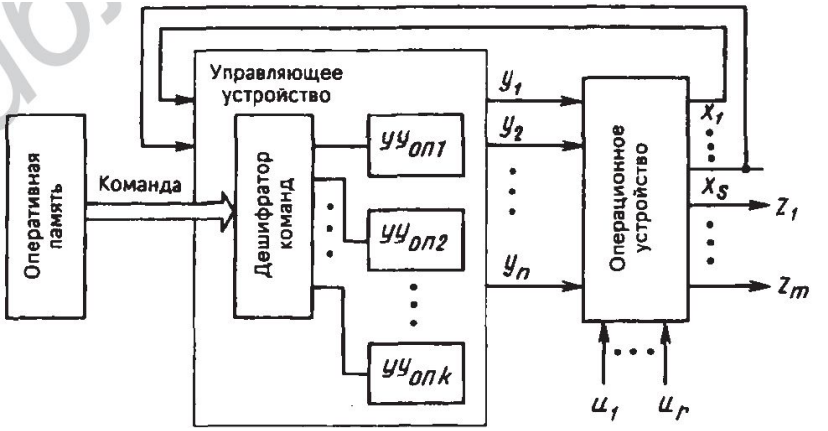
совокупность одновременно выполняемых микроопераций называется *микрокомандой*, а весь набор микрокоманд, предназначенный для решения определенной задачи, — *микропрограммой*.

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

Структура процессора в составе ЭВМ

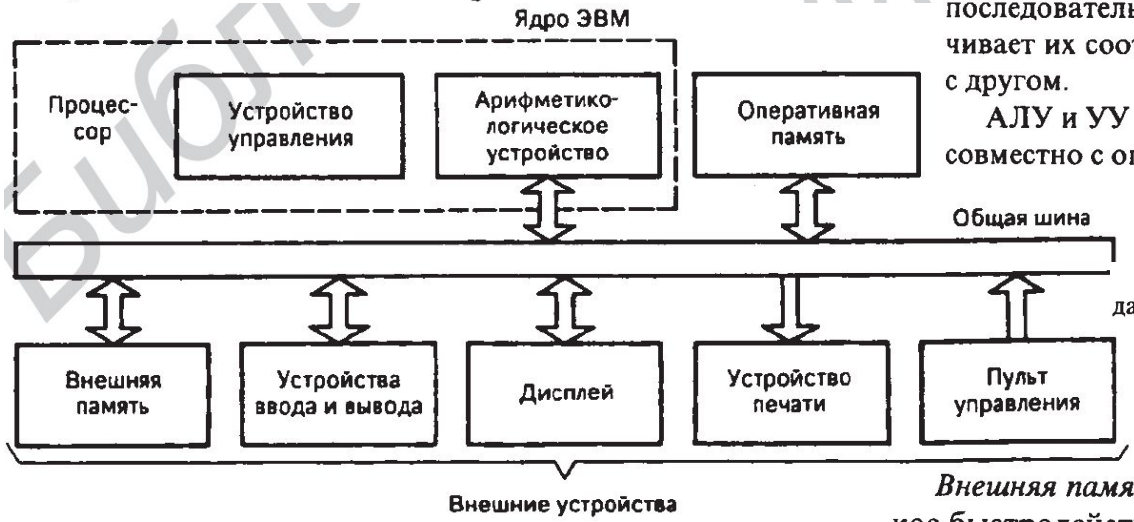
Оперативная память (ОП) служит для хранения программы, исходных данных задачи, промежуточных и конечных результатов решения задачи.

Арифметико-логическое устройство (АЛУ) предназначено для выполнения предусмотренных в ЭВМ арифметических и логических операций. Участвующие в операциях данные выбираются из ОП, результаты операций отсылаются в ОП. Для ускорения выборки операндов (данных, участвующих а операциях) АЛУ может снабжаться собственной местной памятью (сверхоперативным запоминающим устройством — СОЗУ) на небольшое число данных, но обладающей быстродействием, превышающим быстродействие ОП. При этом результаты операций, если они участвуют в последующих операциях, могут не отсылаться в ОП, а храниться в СОЗУ.



Устройство управления (УУ), посылая в определенной временной последовательности управляющие сигналы в устройства ЭВМ, обеспечивает их соответствующее функционирование и взаимодействие друг с другом.

АЛУ и УУ объединяют под общим названием *процессор*. Процессор совместно с оперативной памятью образует *ядро ЭВМ*.



С помощью *устройств ввода* и *вывода* ЭВМ может обмениваться данными, передаваемыми по линиям связи.

Команда – вид операции, подлежащей к исполнению в процессоре.

Внешняя память (ВП) — память, имеющая относительно невысокое быстродействие, но по сравненив с ОП существенно более высокую емкость. В силу того что быстродействие внешней памяти значительно ниже быстродействия АЛУ, последнее в процессе работы взаимодействует лишь с ОП, получая из нее команды и данные, отсылая в эту память результаты операций. Часто при решении сложных задач емкость ОП оказывается недостаточной. В этих случаях в процессе решения задач данные определенными порциями могут пересылаться из внешней памяти в ОП, откуда они затем выбираются для обработки в АЛУ.

Цифровые автоматы

Процессор является примером цифрового автомата — устройства, осуществляющего прием, хранение и преобразование дискретной информации по некоторому алгоритму. Теорию автоматов подразделяют на *абстрактную* и *структурную*. Абстрактная теория изучает поведение автомата, отвлекаясь от структуры (т.е. способа его построения, схемной реализации).

Автомат под действием входных сигналов принимает состояния в соответствии с набором значений входных сигналов и выдает сигнал, зависящий от внутреннего состояния либо от внутреннего состояния и входных сигналов. Для хранения внутреннего состояния автомат должен иметь память; таким образом, автомат является устройством с памятью, т.е. устройством последовательностного типа.

Множества:
Возможных входных сигналов $x_1, x_2, \ldots, x_n$
Внутренних состояний $a_0, a_1, \ldots, a_k$
Возможных выходных сигналов $y_1, y_2, \ldots, y_m$

Работа автомата определяется следущими функциями:
функцией переходов f**, которая определяет состояние автомата** $a(t+1)$ **в момент** $t+1$ **в зависимости от состояния автомата** $a(t)$ **и значения входного сигнала** $x(t)$ **в момент** t**:**

$$a(t+1) = f(a(t); x(t));$$

функцией выходов φ**, определяющей зависимость выходного сигнала автомата** $y(t)$ **от состояния автомата** $a(t)$ **и входного сигнала** $x(t)$**:**

$$y(t) = \varphi(a(t); x(t)).$$

Автомат с такой функцией выходов называется *автоматом Мили*. Другой тип автомата — *автомат Мура*. Особенность автомата Мура в том, что в нем выходной сигнал зависит лишь от внутреннего состояния $a(t)$ **и не зависит от входного сигнала. Функции переходов и выходов для него имеют вид**

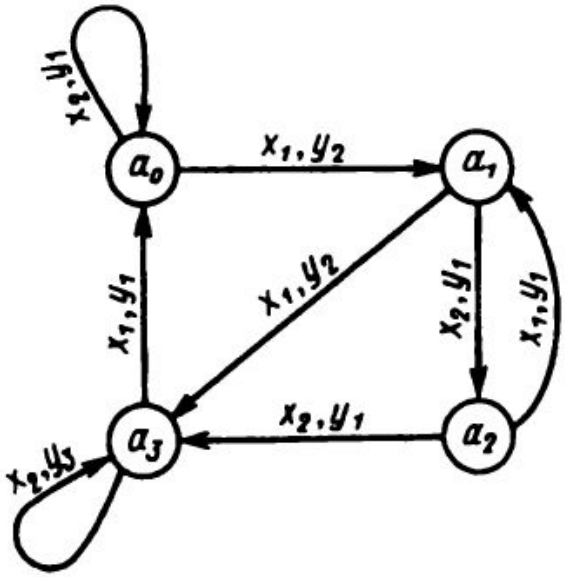
$$a(t+1) = f(a(t); x(t)), \quad y(t) = \varphi(a(t)).$$

Функционирование автомата может быть задано в форме *таблиц переходов* и *выходов* либо с помощью так называемого *графа*.

Цифровые автоматы (пример)

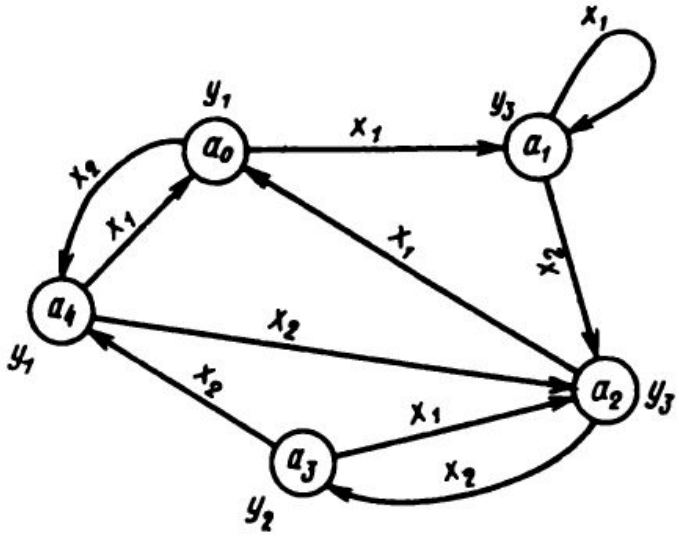
Входной сигнал	a_0	a_1	a_2	a_3
x_1	a_1/y_2	a_3/y_2	a_1/y_1	a_0/y_1
x_2	a_0/y_1	a_2/y_1	a_3/y_1	a_3/y_3

входное слово	x_2	x_1	x_2	x_2	x_1	x_2	x_1	x_1	x_2	...
состояние автомата	a_0	a_0	a_1	a_2	a_3	a_0	a_0	a_1	a_3	a_3
выходное слово	y_1	y_2	y_1	y_1	y_1	y_1	y_2	y_2	y_3	...



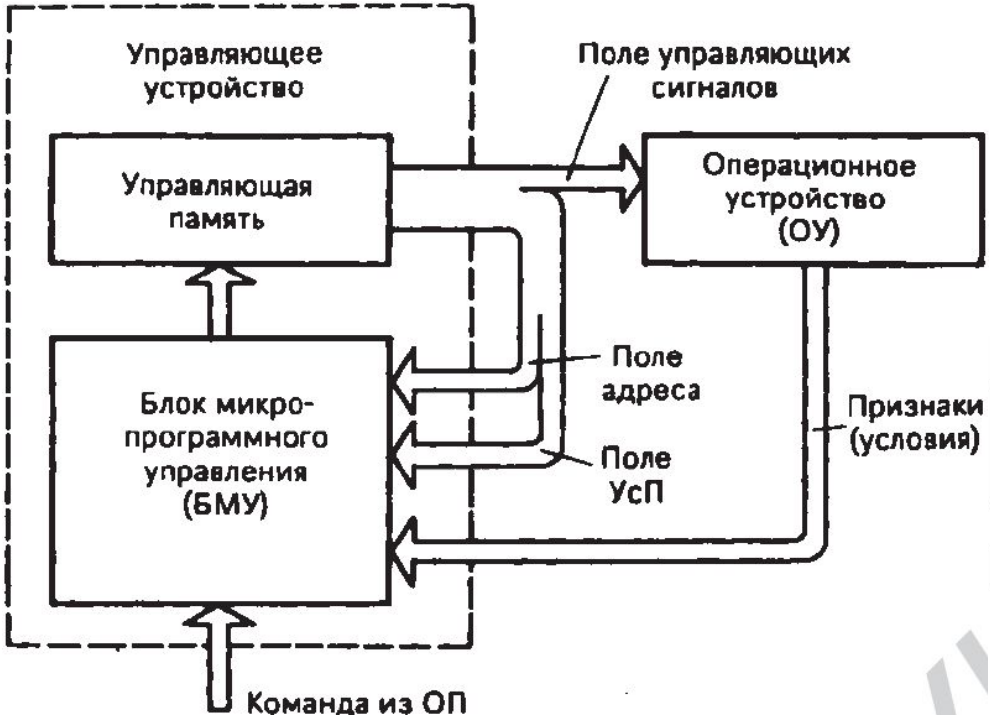
Входной сигнал	y_1	y_3	y_3	y_2	y_1
	a_0	a_1	a_2	a_3	a_4
x_1	a_1	a_1	a_0	a_2	a_0
x_2	a_4	a_2	a_3	a_4	a_2

входное слово	x_2	x_2	x_2	x_1	x_1	x_2	x_1	x_1	x_2	...
состояние автомата	a_0	a_4	a_2	a_3	a_2	a_0	a_4	a_0	a_1	a_2
выходное слово	y_1	y_1	y_3	y_2	y_3	y_1	y_1	y_1	y_3	...



Принцип микропрограммного управления

В управляющей памяти можно хранить много микропрограмм, предназначенных для выполнения различных операций. По выбранной из оперативной памяти команде в управляющей памяти находится соответствующая команде микропрограмма. Далее путем последовательного считывания микрокоманд найденной микропрограммы и их выполнения в операционном устройстве реализуется предусматриваемая командой операция. Такой способ реализации операций называется *микропрограммным*, а построенное на этом способе управляющее устройство — управляющим устройством с *программируемой логикой*.



Управляющая память хранит кодовые комбинации микрокоманд и выдает их в ОУ

ПА содержит адрес очередной МК. Считав из УП микрокоманду, по содержимому ее поля адреса определим адрес следующей МК. Но так можно получить адреса МК при отсутствии в алгоритме разветвлений, т.е. условных переходов (УсП). Для реализации условных переходов в МК надо предусмотреть поле условных переходов, в котором указывается, имеет ли место условный или безусловный переход и при условном переходе — на значения каких условий следует ориентироваться при определении адреса очередной МК.

Пример построения микропрограммы

Четырехразрядное поле адреса позволяет обращаться в любую ячейку УП с 16 ячейками (т.е. в УП с возможностью хранения до 16 микрокоманд).

Поле условных переходов содержит три разряда: разряд П, наличие единицы в котором указывает на то, что имеет место условный переход; разряды x_1 и x_2, наличие единицы в которых определяет условие, по которому происходит условный переход. Поле управляющих сигналов содержит семь разрядов и обеспечивает выдачу сигналов семи различных микроопераций.

Для хранения составляемой микропрограммы используем ячейки УП с последовательно нарастающими адресами: 0000, 0001, 0010,...

Микропроцессор

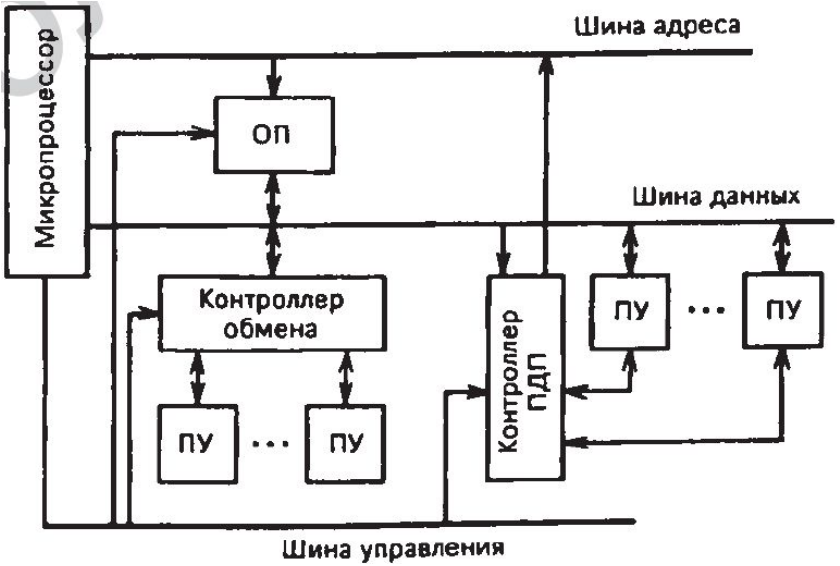
Микропроцессор — процессор (устройство, отвечающее за выполнение арифметических, логических операций и операций управления, записанных в машинном коде), реализованный в виде одной микросхемы или комплекта из нескольких специализированных микросхем (в отличие от реализации процессора в виде электрической схемы на элементной базе общего назначения или в виде программной модели). **Микроконтрооллер** (Micro Controller Unit, MCU) -микросхема, предназначенная для управления электронными устройствами. Сочетает на одном кристалле функции процессора и периферийных устройств, содержит ОЗУ, ПЗУ. Однокристальный компьютер, выполняющий относительно простые задачи. Отличается от микропроцессора интегрированными в микросхему устройствами ввода-вывода, таймерами и другими периферийными устройствами.

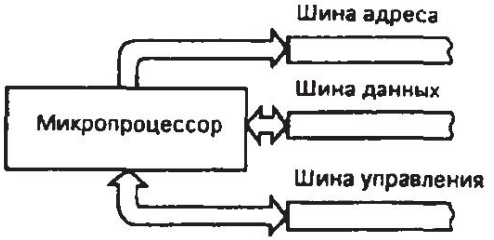
Микропроцессорная система (МПС) представляет собой функционально законченное изделие, состоящее из одного или нескольких устройств, главным образом из микропроцессора и/или микроконтроллера. **Микропроцессорное устройство (МПУ)** представляет собой функционально и конструктивно законченное изделие, состоящее из нескольких микросхем, в состав которых входит микропроцессор; предназначено для выполнения определённого набора функций: получение, обработка, передача, преобразование информации и управление.

Генератор определяет продолжительность выполнения команды. Чем выше частота, тем при прочих равных условиях более быстродействующей является МПС.

МПС содержит МП, ОЗУ и ПЗУ, интерфейсы ввода (пульт, АЦП, датчики …) и вывода.. (дисплей, управления, внешние устройства). Все блоки МПС связаны шинами передачи цифровой информации (магистральный принцип связи). Количество линий в шине данных соответствует разрядности МПС (количеству бит в слове данных). *Шина адреса* применяется для указания направления передачи данных — по ней передаётся адрес ячейки памяти или блока ввода-вывода, которые получают или передают информацию в

Обобщенная структура МПС

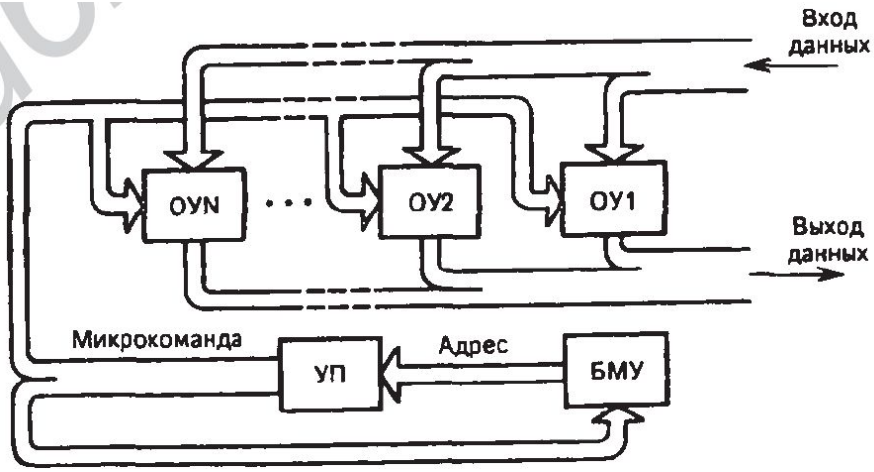


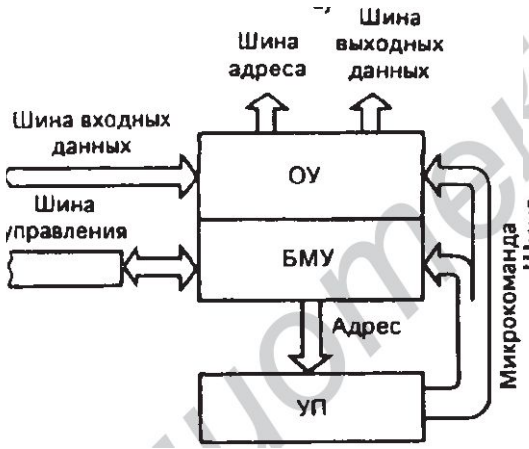


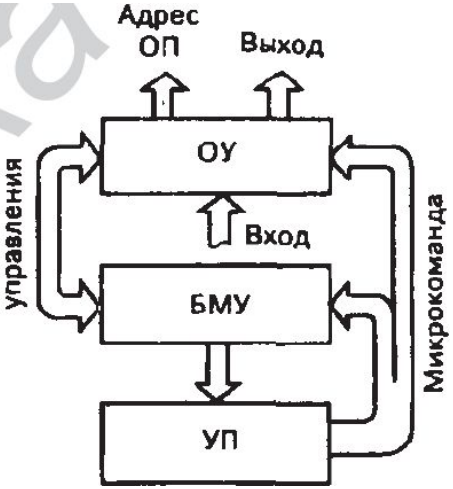
оперативная память (ОП), предназначенная для хранения и выдачи по запросам команд программ, определяющих работу микропроцессора, различных данных (исходных данных, промежуточных и конечных результатов обработки данных в микропроцессоре);

контроллеры — устройства, обеспечивающие обмен данными различных ПУ с микропроцессором и ОП.

При работе с высокоскоростными ПУ используется режим прямого доступа к памяти (ПДП)







Микропроцессор. Машинный код.

Машинный код (платформенно-ориентированный код), машинный язык — система команд (набор кодов операций) конкретной вычислительной машины, которая интерпретируется непосредственно процессором или микропрограммами этой вычислительной машины.
Компьютерная программа, записанная на машинном языке, состоит из машинных инструкций, каждая из которых представлена в машинном коде в виде **опкода** — двоичного кода отдельной операции из системы команд машины. Вместо числовых опкодов используют их условные буквенные **мнемоники**. Набор таких мнемоник, вместе с некоторыми дополнительными возможностями, называется языком ассемблера.
Совместимость Каждая модель процессора имеет свой собственный набор команд, во многих моделях наборы перекрываются. Процессор A совместим с процессором B, если процессор A полностью «понимает» машинный код процессора B. Если процессоры A и B имеют некоторое подмножество инструкций, по которым они взаимно совместимы, то говорят, что они одной «архитектуры» (имеют одинаковую архитектуру набора команд). Напр. IBM System/360 с разными шинами от 8 до 64 бит имеют общую архитектуру на уровне машинного языка.
Машинный код - самый низкий уровень представления скомпилированных (ассемблированных) компьютерных программ. *Свойства - громоздкость кода и трудоёмкость ручного управления ресурсами процессора, экстремальная оптимизация.* ПО пишется на языках высокого уровня и транслируется в машинный код компиляторами.
Абсолютный код — программный код, пригодный для прямого выполнения процессором, то есть код, не требующий дополнительной обработки (например, разрешения ссылок между различными частями кода или привязки к адресам в памяти, обычно выполняемой загрузчиком программ). Напр. исполнимые файлы в формате .COM и загрузчик ОС, располагаемый в MBR. **MBR** (master boot record, главная загрузочная запись) — код и данные, необходимые для последующей загрузки операционной системы и расположенные в первых физических секторах на устройстве хранения информации. MBR содержит небольшой фрагмент исполняемого кода, таблицу разделов и специальную сигнатуру. Функция MBR - «переход» в тот раздел жёсткого диска, с которого следует исполнять «дальнейший код» (обычно — загружать ОС). **Позиционно-независимый код** — программа, которая может быть размещена в любой области памяти, так как все ссылки на ячейки памяти в ней относительные (например, относительно счётчика команд). Программу

Прерывания процессора

Прерывание (interrupt) — сигнал, сообщающий процессору о наступлении какого-либо события, выполнение текущей последовательности команд приостанавливается, и управление передаётся обработчику прерывания, который реагирует на событие и обслуживает его, после чего возвращает управление в прерванный код.

В зависимости от источника возникновения сигнала, прерывания делятся на: **асинхронные**, или внешние (аппаратные) — события, которые исходят от внешних источников (например, периферийных устройств) и могут произойти в любой произвольный момент: сигнал от таймера, сетевой карты или дискового накопителя, нажатие клавиш клавиатуры, движение мыши. Запрос на прерывание (Interrupt request, IRQ); **синхронные**, или внутренние — события в самом процессоре как результат нарушения каких-то условий при исполнении машинного кода: деление на ноль или переполнение стека, обращение к недопустимым адресам памяти или недопустимый код операции; **программные** (частный случай внутреннего прерывания) — инициируются исполнением специальной инструкции в коде программы. Программные прерывания, как правило, используются для обращения к функциям встроенного программного обеспечения (firmware), драйверов и операционной системы.

Внешние прерывания в зависимости от возможности запрета делятся на: **маскируемые** — прерывания, которые можно запрещать установкой соответствующих битов в регистре маскирования прерываний; **немаскируемые** (Non-maskable interrupt, NMI) — обрабатываются всегда, независимо от запретов на другие прерывания (например сбой в микросхеме памяти).

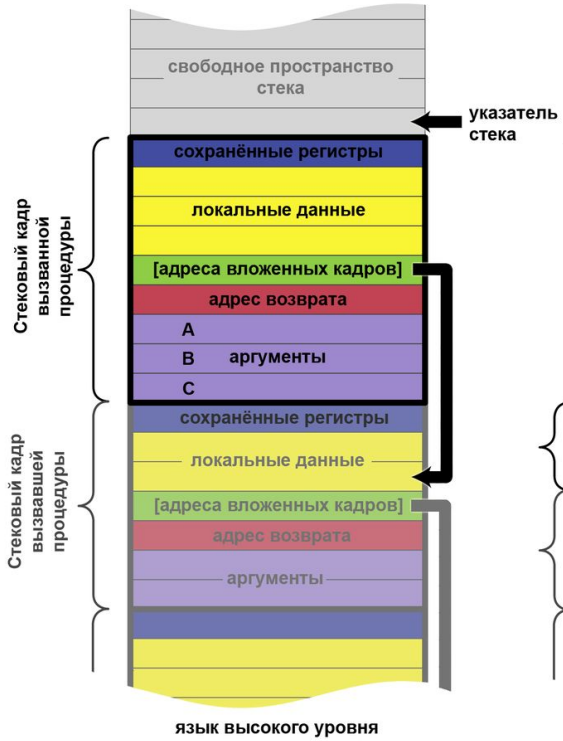
Приоритизация разделяет все источники прерываний на классы и каждому классу назначается свой уровень приоритета запроса на прерывание. **Относительное обслуживание прерываний** -если во время обработки прерывания поступает более приоритетное прерывание, то это прерывание будет обработано только после завершения текущей процедуры обработки прерывания. **Абсолютное** - текущая процедура обработки прерывания вытесняется, и процессор начинает выполнять обработку вновь поступившего

Стек процессора

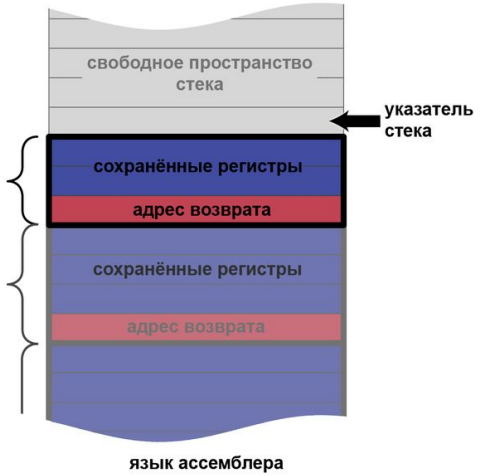
Стек - регистр хранящий информацию для возврата управления из подпрограмм (процедур) в программу и/или для возврата в программу из обработчика прерывания (в том числе при переключении задач в многозадачной среде).

При вызове подпрограммы или возникновении прерывания, в стек заносится адрес возврата — адрес в памяти следующей инструкции приостановленной программы и управление передается подпрограмме или подпрограмме-обработчику. При последующем вложенном или рекурсивном вызове, прерывании подпрограммы или обработчика прерывания, в стек заносится очередной адрес возврата и т. д. При возврате из подпрограммы или обработчика прерывания, адрес возврата снимается со стека и управление передается на следующую инструкцию приостановленной (под-) программы.

Назначение — отслеживать место, куда каждая из процедур должна вернуть управление после завершения. В стек заносится адрес команды, следующей за командой вызова («адрес возврата»). По завершении вызванная процедура выполнит команду возврата для перехода по адресу из стека. В стеке могут сохраняться: значения регистров с их последующим восстановлением данные стекового кадра языков высокого уровня; аргументы, переданные в функцию; локальные переменные — временные данные функции; другие произвольные данные



В **многозадачных** системах, каждая задача имеет свой собственный стек, и при переключении задачи указатель стека процессора переставляется на него.

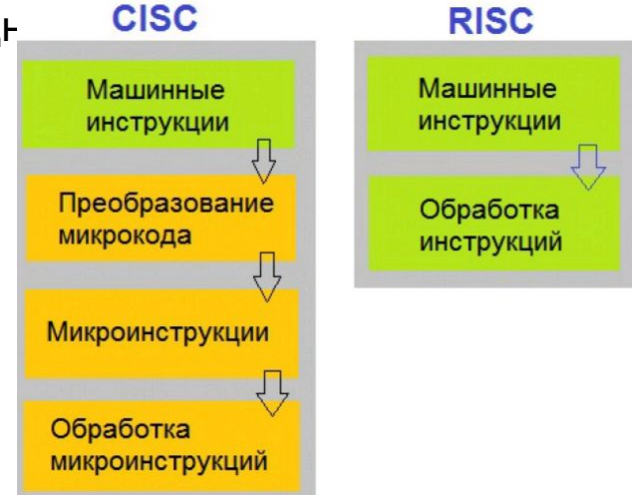


Архитектура процессора

Архитектура процессора — количественная составляющая компонентов микроархитектуры (регистр флагов или регистры процессора), рассматриваемая IT-специалистами в аспекте прикладной деятельности.

С точки зрения: программиста — совместимость с определённым набором команд (например, процессоры, совместимые с командамиIntel x86), их структуры (например, систем адресации или организации регистровой памяти) и способа исполнения (например, счётчик команд);

аппаратной составляющей вычислительной системы — это некий набор свойств и качеств, присущий целому семейству процессоров («внутренняя конструкция», «организация» этих процессоров). Имеются различные классификации архитектур процессоров как по организации (по количеству и скорости выполнения команд: RISC, CISC), так и по назначению (специализированные графические, математические или предн обработки сигналов).



RISC (restricted (reduced) instruction set computer - «компьютер с сокращённым набором команд») - архитектура процессора, в котором быстродействие увеличивается за счёт упрощения инструкций, чтобы их декодирование было более простым, а время выполнения - меньшим. Первые RISC-процессоры не имели инструкций умножения и деления. Эффективная суперскалярность (распараллеливание инструкций между несколькими исполнительными блоками).

CISC (Complex instruction set computing - компьютер с полным набором команд) - концепция проектирования процессоров, которая характеризуется следующим набором свойств:

-нефиксированное значение длины команды; -арифметические действия кодируются в одной команде;

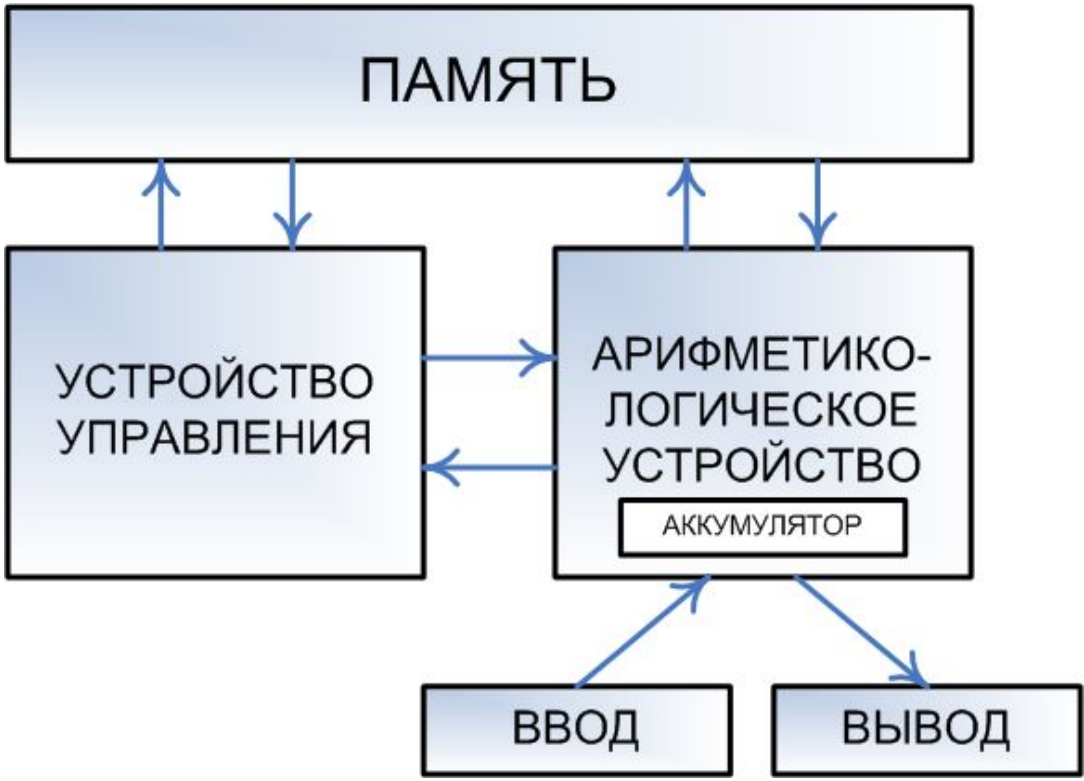
-небольшое число регистров, каждый из которых выполняет строго определённую функцию.

Недостатки CISC: высокая стоимость аппаратной части; сложности с распараллеливанием

Архитектуры

Архитектура Фон Неймана

Однородная память микропроцессора. В память могут записываться различные программы. При этом специальная программа-загрузчик работает с ними как с данными. Затем управление может быть передано этим программам и они уже начинают выполнять свой алгоритм.
+ Достигается максимальная гибкость микропроцессорной системы.
- Возможность непреднамеренного нарушения работоспособности системы (программные ошибки) и преднамеренное уничтожение ее работы (вирусные атаки).



Принцип однородности памяти. Совместное хранение команд и данных в памяти. Над командами можно выполнять такие же действия, как и над данными.

Принцип адресуемости памяти. Структурно основная память состоит из пронумерованных ячеек; процессору в произвольный момент времени доступна любая.

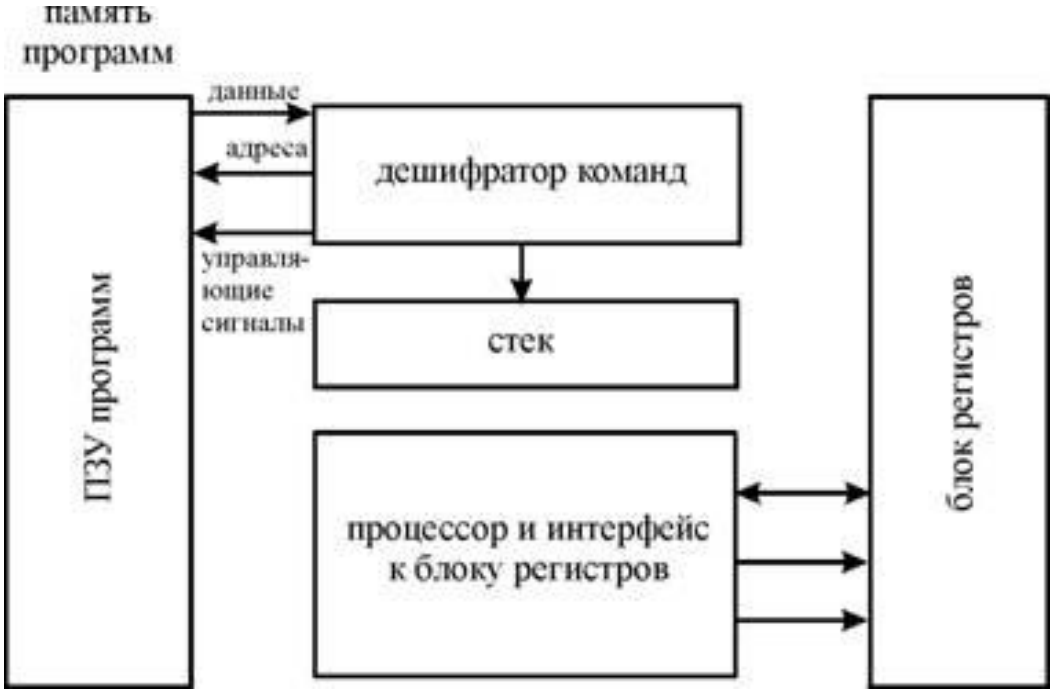
Принцип последовательного программного управления. Программа состоит из набора команд, которые выполняются процессором автоматически друг за другом в определенной последовательности.

Принцип жесткости архитектуры. Неизменяемость в процессе работы топологии, архитектуры, списка команд.

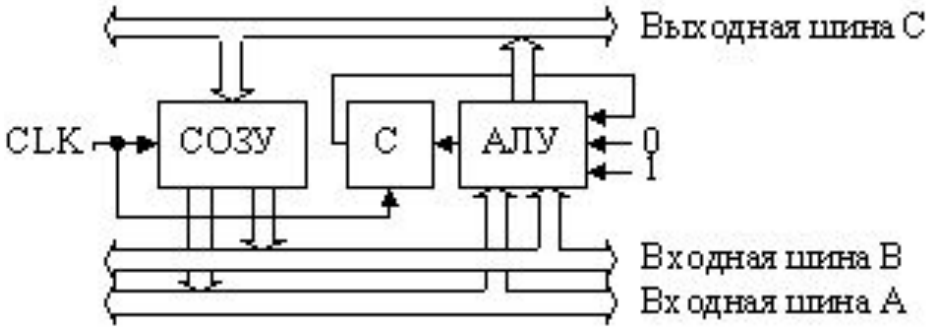
Принцип двоичного кодирования.

Гарвардская архитектура

Два вида памяти микропроцессора: Память программ (для хранения инструкций микропроцессора); память данных (для временного хранения и обработки переменных). Невозможно осуществить операцию записи в память программ, что исключает возможность случайного разрушения управляющей программы в случае ошибки программы при работе с данными или атаки третьих лиц. Для работы с памятью программ и с памятью данных организуются отдельные системные шины.



Применяется в микроконтролерах и в сигнальных процессорах , где требуется обеспечить высокую надёжность работы аппаратуры. В сигнальных процессорах Гарвардская архитектура дополняется применением трехшинного операционного блока микропроцессора.

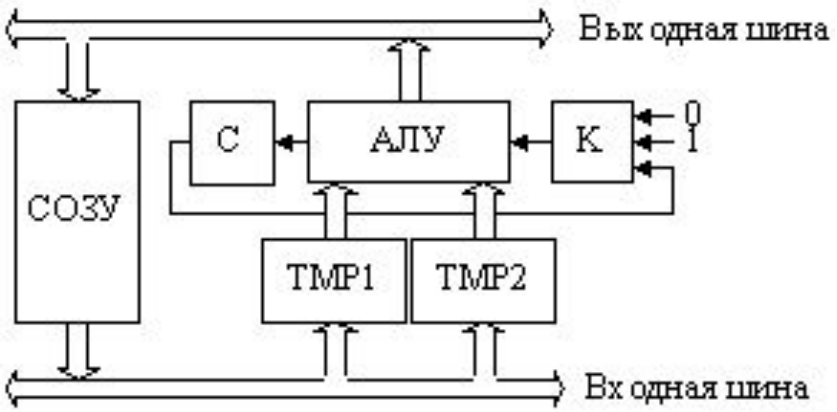


Трехшинная архитектура операционного блока позволяет совместить операции считывания двух операндов с записью результата выполнения команды в оперативную память микропроцессора. Это значительно увеличивает производительность сигнального микропроцессора без увеличения его тактовой частоты. *Шины передачи данных занимают огромную площадь на кристалле микросхемы.*

Типовые структуры операционного блока

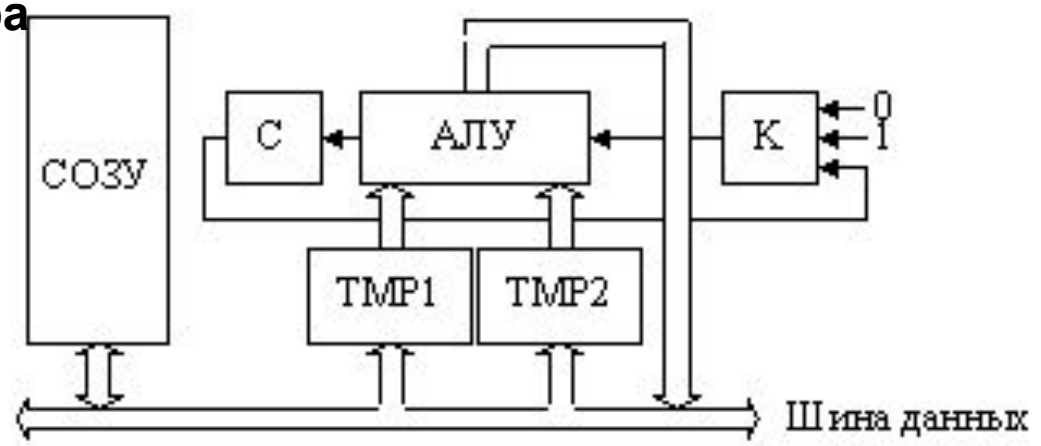
Двухшинная структура

Используется только две шины передачи данных. Для формирования двух источников данных для входов АЛУ в двухшинной схеме операционного блока микропроцессора используются два регистра временного хранения TMP1 и TMP2.



Одношинная структура

В результате того, что входные данные к арифметико-логическому устройству передаются по одной шине данных, получается, что для выполнения одной операции требуется, как минимум, два такта сигнала синхронизации CLK.



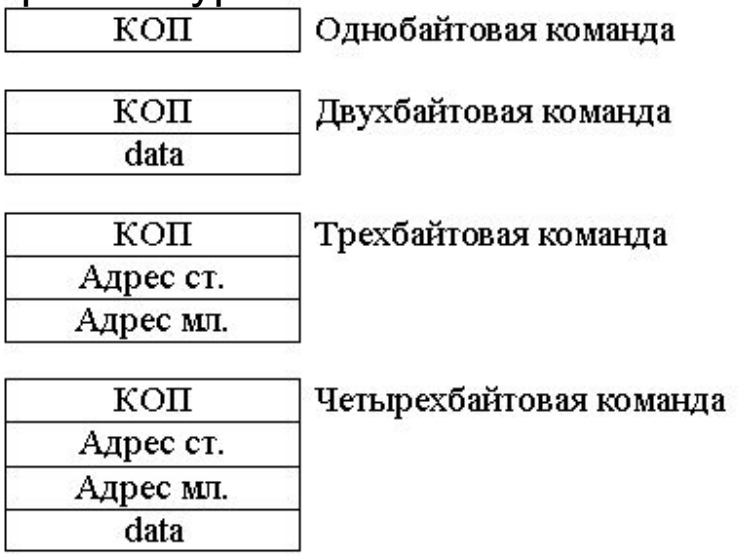
Это приводит к тому, что быстродействие микропроцессора, устроенного подобным образом, при той же частоте тактовой синхронизации будет ниже быстродействия микропроцессора, построенного на базе трехшинной структуры операционного блока. Занимает наименьшую площадь на кристалле.

Команды микропроцессора

Разрядность команд совпадает с разрядностью микропроцессора. Команда микропроцессора состоит из инструкции и обозначается код операции КОП. Команда может состоять только из КОП, когда не требуется указывать адрес операнда (это данные, над которыми команда производит какое либо действие), или может состоять из кода операции и адресов операндов или данных. Форматы команд очень сильно зависит от структуры процессора.

При помощи 1 байт слова можно закодировать 256 операций. Именно системой команд и определяется конкретное семейство процессоров. Однобайтовые команды позволяют работать с внутренними программно доступными регистрами процессора. Для выполнения одной и той же операции над разными регистрами процессора назначаются разные коды.

Форматы различных команд для восьмиразрядного процессора по архитектуре Фон-Неймана



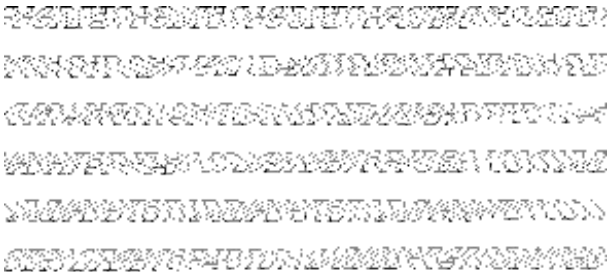
Язык программирования в котором для обозначения машинных команд используются мнемонические обозначения называется ассемблером.

Компилятор осуществляет трансляцию (преобразование) исходного текста программы (исходный модуль) в машинные коды (загрузочный модуль).

Мнемоническое обозначение операции и используемые ею операнды, которые перечисляются через запятую. При этом в большинстве процессоров операнд приёмник информации записывается первым, а операнд источник информации вторым.

Операция копирования - мнемоническое обозначение MOV; суммирования используется ADD; вычитания используется мнемоническое обозначение SUB; умножения используется мнемоническое обозначение MUL.

Фрагмент исполняемого 16-ти ричного машинного кода



```
MOV R0, A ;Скопировать содержимое регистра A в регистр R0
ADD A, R5 ;Просуммировать содержимое регистров R5 и A, результат поместить в регистр A
MOV A, 1025 ;Скопировать содержимое 1025 ячейки памяти в регистр A
ADD A, #110 ;Просуммировать содержимое регистра A с числом 110
```

32-разрядная архитектура ARM

На RISC-архитектуре, отклонения от принципов RISC:

- Переменное количество циклов выполнения для простых инструкций. Простые инструкции ARM могут потребовать на выполнение более одного цикла.
- Возможность соединять команды сдвига и вращения с командами обработки информации.
- Условное выполнение – инструкция выполняется только в том случае, если выполняется конкретное условие. Это увеличивает производительность и позволяет избавиться от операторов ветвления.
- Улучшенные инструкции – процессоры ARM поддерживают улучшенные DSP-инструкции для операций с цифровыми сигналами.

Программист может рассматривать ядро ARM как набор функциональных блоков – ALU, MMU и др., – соединенных шиной данных. Данные поступают в процессор через шину данных. Декодер инструкций обрабатывает инструкции перед их выполнением. ARM могут работать только с данными, которые записаны в регистрах, поэтому перед выполнением инструкций в регистры записываются данные для их выполнения. ALU считывает данные из регистров, выполняет необходимые операции и записывает результат обратно в регистр, откуда его можно записать во внешнюю память.

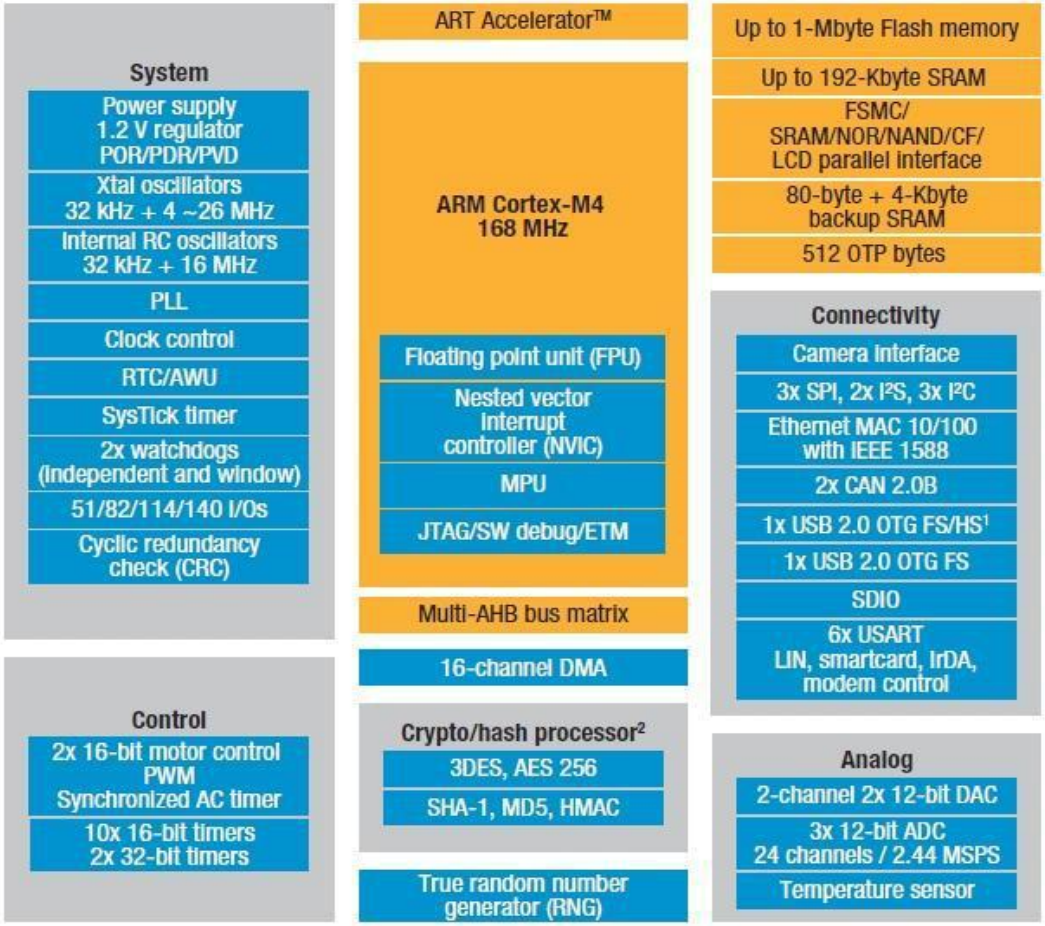
Процессоры ARM содержат до 18 регистров: 16 регистров данных и 2 регистра процессов. Все регистры содержат 32 бита и именуются от R0 до R15. Регистры R13, R14, R15 используются для выполнения определенных специфических задач: R13 используется в качестве указателя стека;

R14 используется как связывающий регистр;

R15 играет роль счетчика.

В зависимости от контекста эти регистры могут использоваться как регистры общего назначения. Также имеется два программных регистра, которые называются CPSR (Current Program Status Register) и SPSR (Saved Program Status Register), которые используются для сохранения состояния процессора и программы.

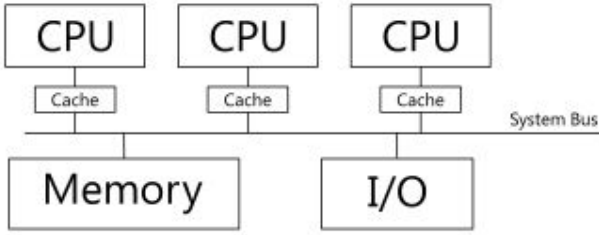
Cortex-M4 предназначены для использования в цифровой обработке сигналов (Digital Signal Processing, DSP). В общем виде микроконтроллеры, основанные на базе ARM Cortex-M4 имеют следующие внутренние: Микроконтроллер, установленный на рассматриваемой плате, STM32F407VG, в качестве основы использует именно решение ARM Cortex-M4.

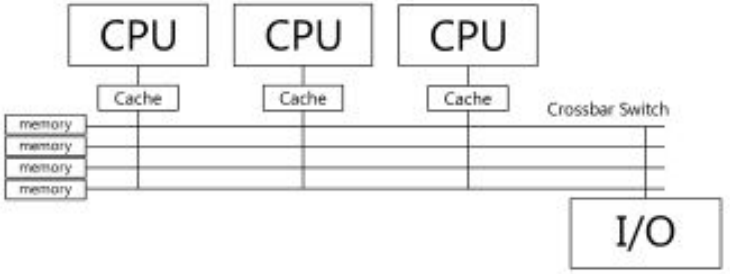


Характеристика	Значение
Ширина слов для данных, разряд	32
Архитектура	Гарвард
Конвейер	3-ступенчатый
Набор инструкций	RISC
Организация памяти программ, разряд	32
Буфер предвыборки, разряд	2x64
Средний размер инструкции, байт	2
Тип прерываний	Векторизированные
Задержка реагирования на прерывания	12 циклов
Режимы управления энергопотреблением	Сон, сон по выходу, глубокий сон, режим ожидания

Архитектура многоядерных процессоров

Архитектура многоядерных процессоров во многом повторяет архитектуру симметричных мультипроцессоров (SMP-машин) только в меньших масштабах и со своими особенностями. Тактовая частота снижена для уменьшения энергопотребления процессора без потери производительности и частота каждого ядра может меняться в зависимости от его индивидуальной нагрузки.
Симметричное мультипроцессирование (Symmetric Multiprocessing) - архитектура многопроцессорных компьютеров, в которой два или более одинаковых процессора сравнимой производительности подключаются единообразно к общей памяти (и периферийным устройствам) и выполняют одни и те же функции (почему, собственно, система и называется симметричной).
Ядро является полноценным микропроцессором, использующим: конвейеры, внеочередное исполнение кода, многоуровневый кэш, поддержка векторных команд. Суперскалярность в ядре не используется. Ядро использует технологию SMT для поочередного исполнения нескольких потоков, создавая иллюзию нескольких «логических процессоров» на основе каждого ядра (в Intel технология Hyper-threading, Sun UltraSPARC - 8 потоков на ядро).

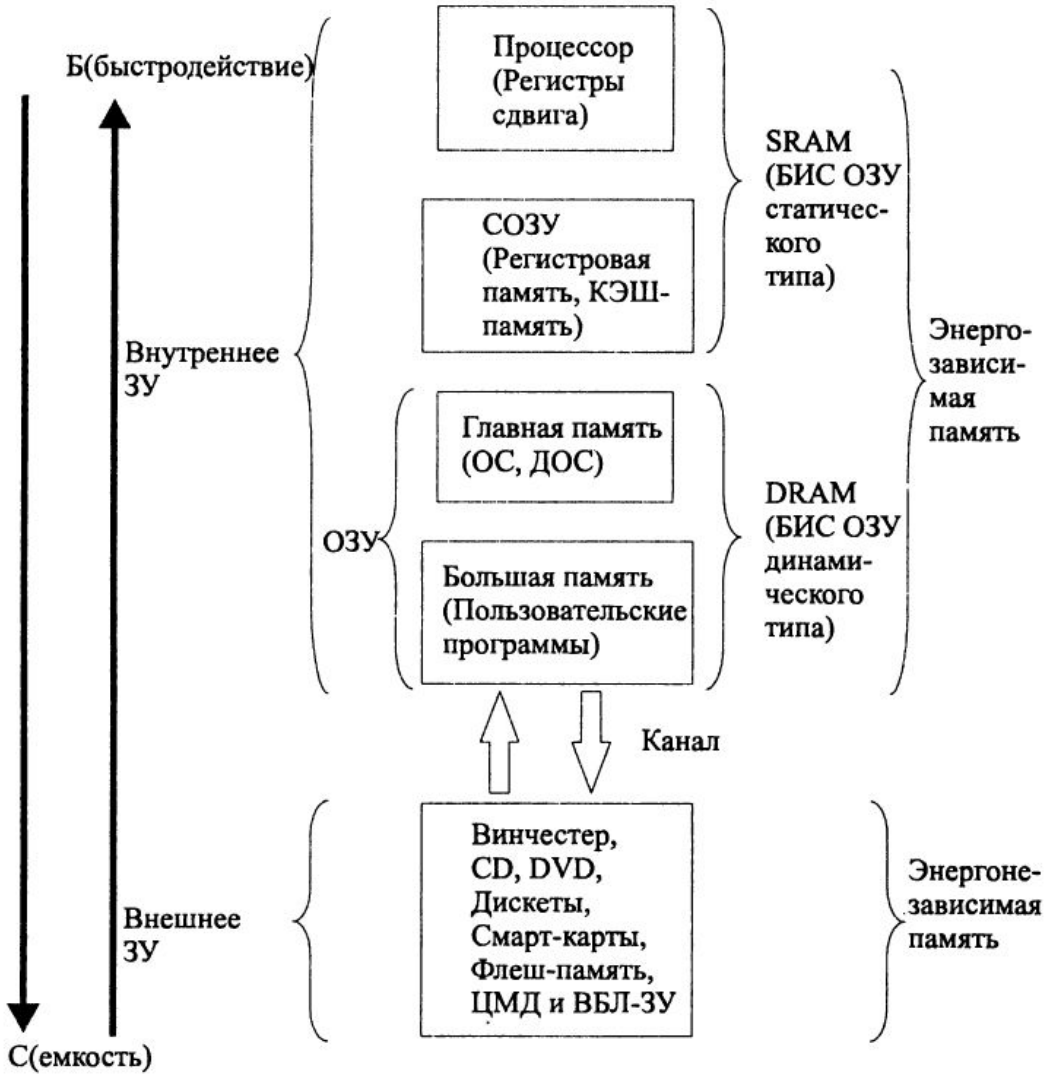




Способ связи между ядрами
разделяемая шина
сеть на каналах точка-точка
сеть с коммутатором
общая кэш-память.

Кэш-память. 1-го уровня обладает каждое ядро в отдельности. 2-го уровня: разделяемая — расположена на одном кристалле с ядрами и доступна каждому из них в полном объёме(Intel Core); индивидуальная — отдельные кеши равного объёма, интегрированные в каждое из ядер. Обмен данными из кешей 2-го уровня между ядрами осуществляется через контроллер памяти — интегрированный (Athlon 64 X2, Turion X2, Phenom) или внешний (использовался в Pentium D, в дальнейшем Intel отказалась от такого подхода).
Гомогенная архитектура - все ядра процессора одинаковы и выполняют одни и те же задачиIntel Core Duo, Sun SPARC T3, AMD Opteron; **гетерогенная архитектура** - ядра процессора выполняют разные задачи (Cell альянса IBM, Sony и Toshiba, у которого из девяти ядер одно является ядром процессора общего

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.

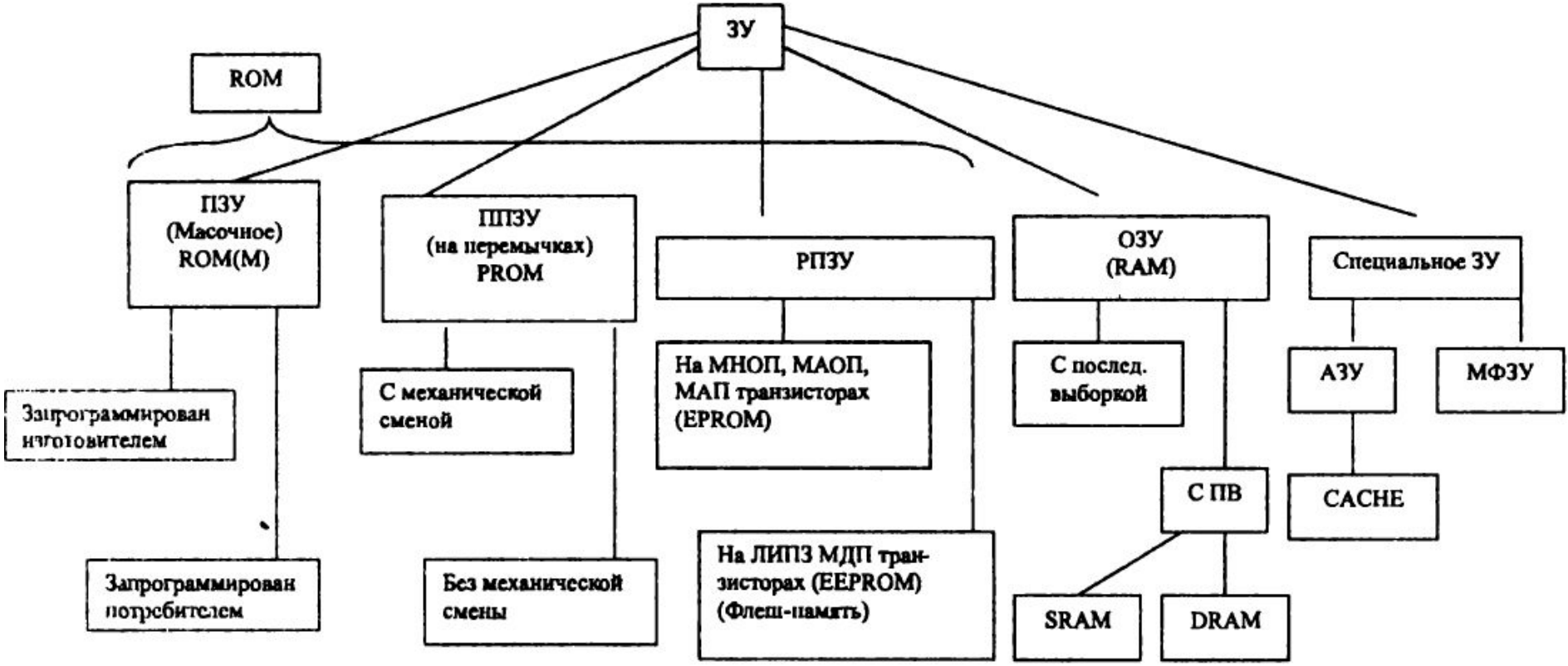
ROM — Read-Only Memory (ЗУ только для чтения или постоянное запоминающее устройство, ПЗУ).

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.

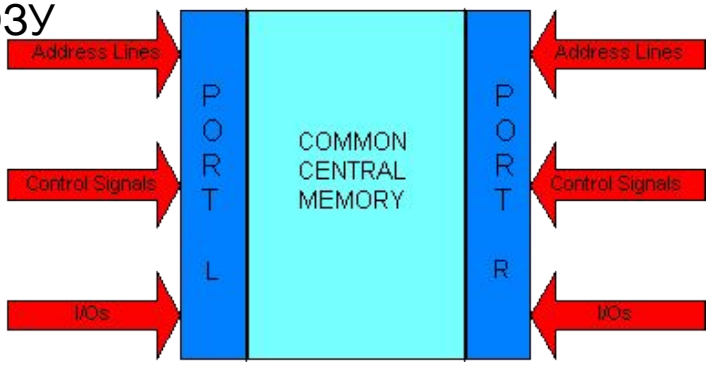
АЗУ — ассоциативные ЗУ. В АЗУ поиск ячейки ЗУ ведется не по адресам, а по специальному признаку, для которого в начале слова отводится несколько разрядов.

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

Многопортовая память

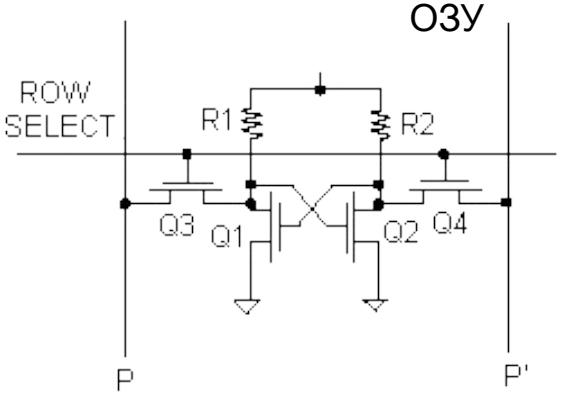
Многопортовая память - это статическое ОЗУ с двумя или более независимыми интерфейсами, обеспечивающими доступ к пространству памяти через разделенные шины адреса, данных и управления.
Единый массив памяти (COMMON CENTRAL MEMORY) и два независимых порта (PORT_L и PORT_R) для обращения к этому массиву. 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.

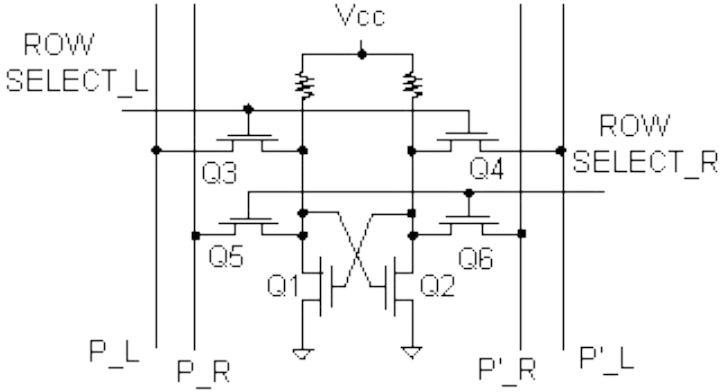
Структура двухпортового статического ОЗУ



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.



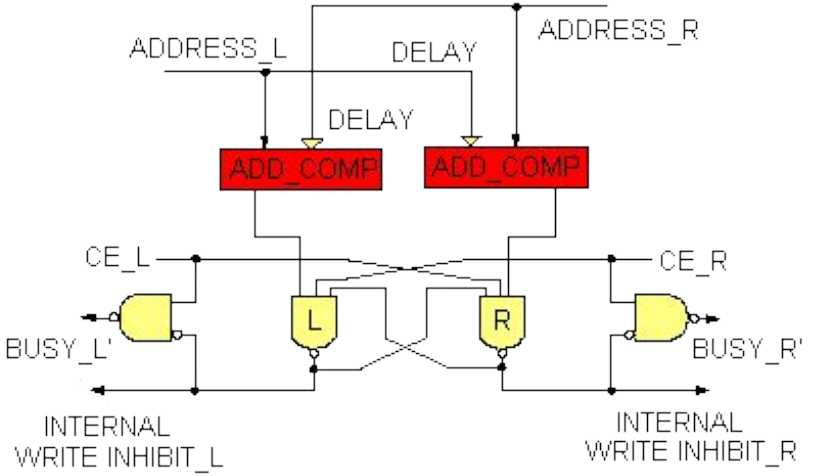


Во всех схемах с асинхронным доступом к общим ресурсам возникают конфликтные ситуации. Конфликты появляются при одновременном обращении двух независимых активных устройств к одной и той же ячейке памяти в процессе выполнения следующих операций: запись через порт L - запись через порт R; запись через 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

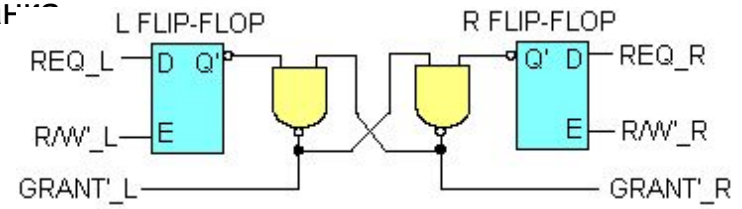
Арбитражная логика. Арбитр двухпортового ОЗУ устраняет конфликты. Сигналы адресных линий портов ADDRESS_L и ADDRESS_R поступают с двух направлений и, если их значения совпадают, то арбитр посылает одному из активных устройств сигнал BUSY' ("запрет доступа"). BUSY' поступает в опоздавшее к моменту арбитража 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 3И-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.

Схема формирования сигналов занятости бан...



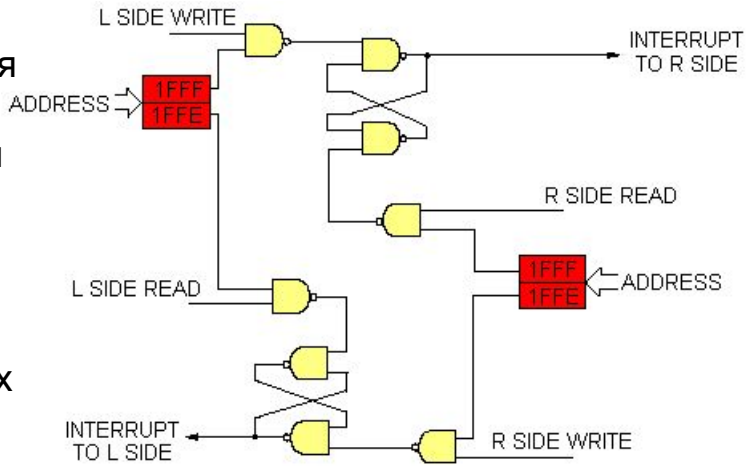
Семафоры - это программные арбитры, регулирующие очередность обращения двух или более независимых активных устройств к общему ресурсу. Несколько ячеек памяти, не входящих в рабочее пространство, используются как указатели занятости определенных сегментов (банков) памяти. "0" код в семафоре соответствует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

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'.

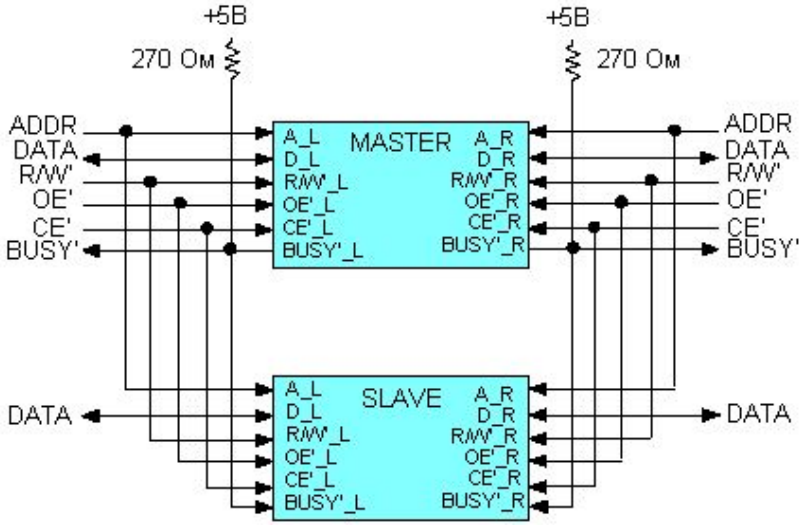
Принцип работы асинхронного двухпортового ОЗУ

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).

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.

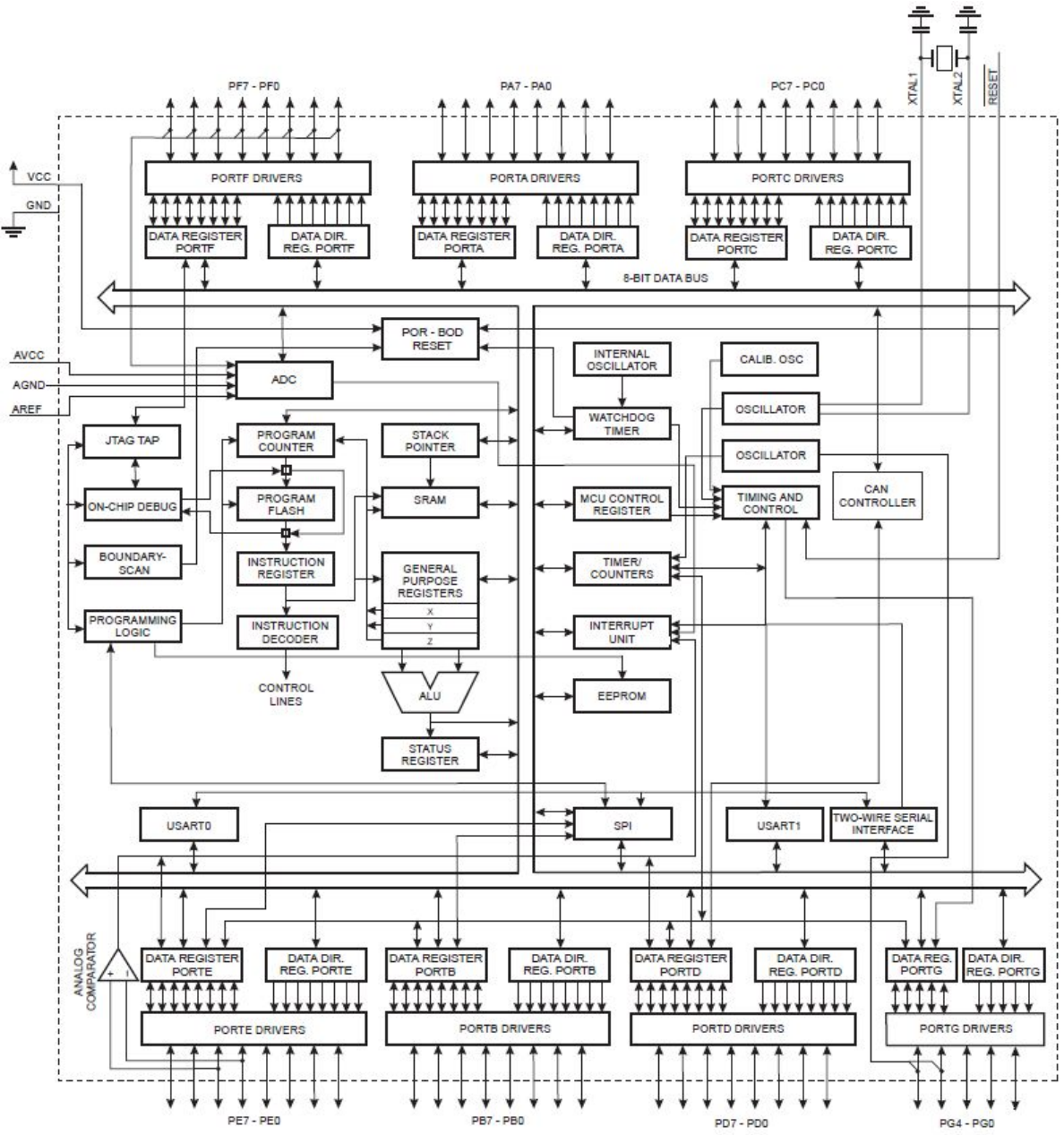
Пропускная способность — метрическая характеристика, показывающая соотношение предельного количества проходящих единиц (информации, предметов, объёма) в единицу времени через канал, систему, узел.

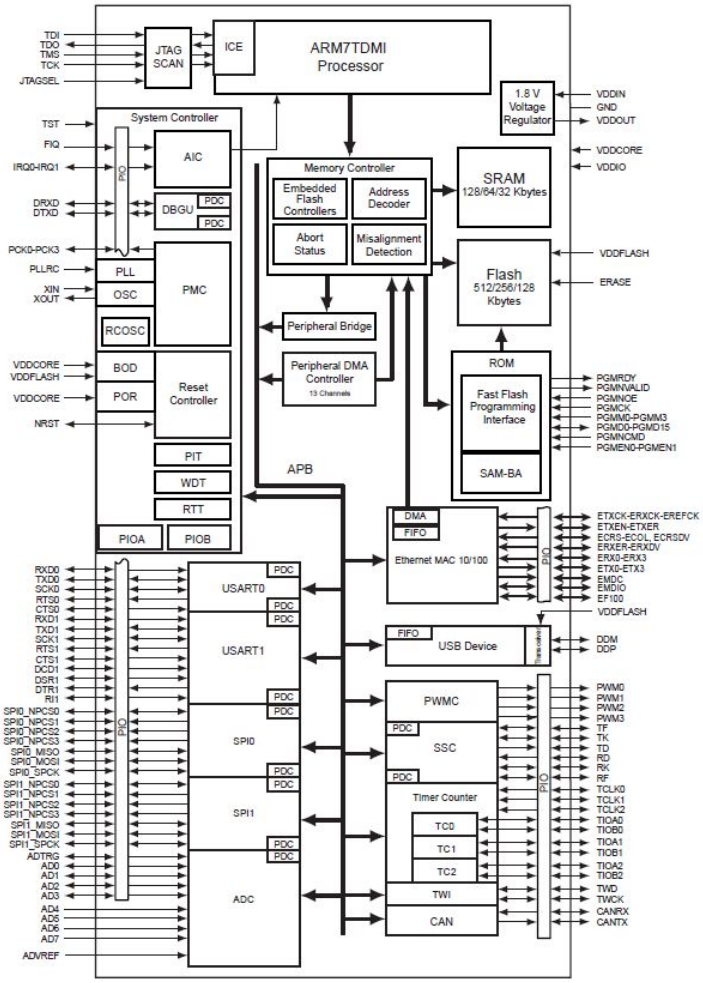
Пиковая пропускная способность — теоретическая максимальная пропускная способность; в реальных условиях производительность интерфейса, как правило, окажется значительно ниже, нежели та, что приведена в таблице.

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





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

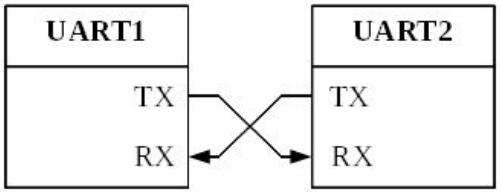
Память	Проп.сп
FPM RAM	1,408 Гбит/с
EDO RAM	2,112 Гбит/с
SPARC MBus	2,550 Гбит/с
PC1600 (DDR-200)	12,50 Гбит/с
PC3-19200 (DDR3-2400)	150,00 Гбит/с

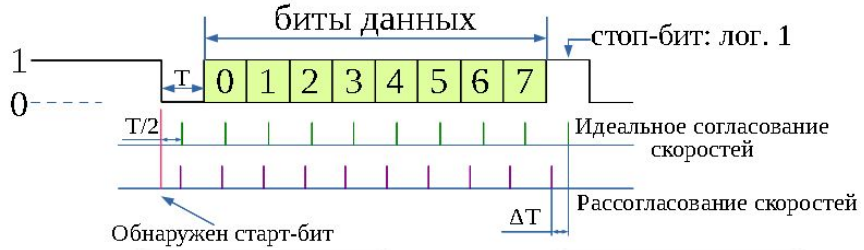
Для внутр. накопителей	Проп.сп
ATA-1 (DMA-0)	33,6 Мбит/с
ATA-2 (DMA-2)	133 Мбит/с
ATA-4 (UDMA-0)	133 Мбит/с
ATA-4 (UDMA-1)	200 Мбит/с
ATA-7 (UDMA-6)	1066 Мбит/с
SATA 1.x 1.5Gb/s	**1,2** Гбит/с
SATA 3.x 6Gb/s	**4,8** Гбит/с

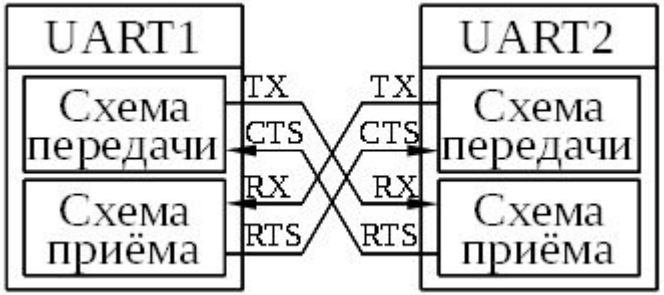
Внешн. устройства	Проп.сп
RS-232	**230,4** Кбит/с
USB 1.0 Low Speed	**1,5** Мбит/с
USB 1.0 Full Speed	**12** Мбит/с
USB 2.0 Hi-Speed	**480** Мбит/с
SATA 2.0	**2,4** Гбит/с
FireWire 3200	**3,2** Гбит/с
USB 3.0	**4,8** Гбит/с
SATA 3.0	**6** Гбит/с
HDMI 1.3-1.4a	**10,2** Гбит/с
HDMI 2.0a	**18** Гбит/с
Thunderbolt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

Интерфейс UART

UART (Universal Asynchronous Receiver/Transmitter) - универсальный асинхронный приёмопередатчик, интерфейс для связи цифровых устройств, предназначенный для передачи данных в последовательной форме. 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. **USART** (Universal Synchronous-Asynchronous Receiver/Transmitter) - универсальный синхронно-асинхронный приёмопередатчик - аналогичный UART интерфейс, но дополнительно к возможностям UART, поддерживает режим синхронной передачи данных - с использованием дополнительной линии тактового сигнала. UART может использоваться как для взаимодействия компонентов внутри одного устройства, так и для подключения устройств между собой. **RS-232** (Recommended Standard 232) — физический уровень асинхронного (UART) интерфейса, обеспечивает передачу данных и специальных сигналов между терминалом (Data Terminal Equipment, DTE) и коммуникационным





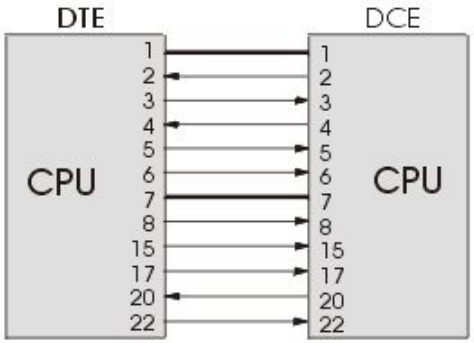


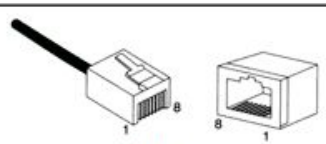
Управление потоком данных

Контроль состояния вх CTS - передатчик перед отправкой очередного фрейма проверяет: Если CTS=0, передача происходит, иначе - нет. Если сигнал CTS=1 во время передачи фрейма, текущая передача будет завершена. Приёмник, устанавливает на вых RTS значение лог. 0, если он готов принимать данные и лог. 1, требуя от

Интерфейс RS-232-C

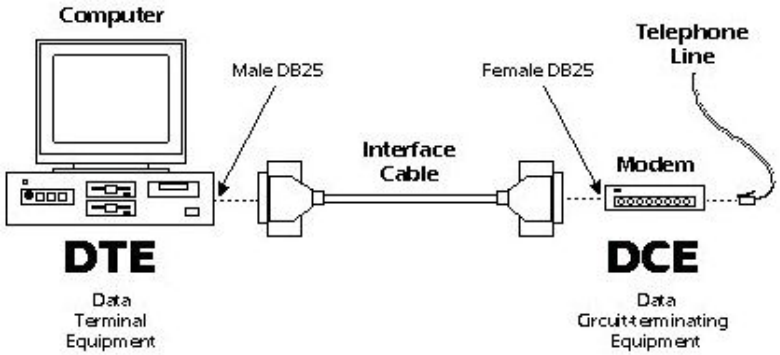
RS-232-C соединяет два устройства. Линия передачи первого соединяется с линией приема второго и наоборот (полный дуплекс). Данные в RS-232C передаются в последовательном коде побайтно. Каждый байт обрамляется стартовым и стоповыми битами. Данные могут передаваться как в одну, так и в другую сторону (дуплексный ре



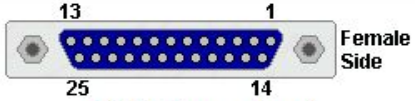


RJ-45

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

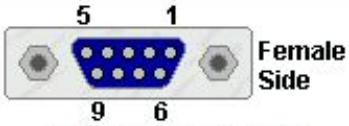


Стандарт	EIA RS-232-C, CCITT V.24
Скорость передачи	115 Кбит/с (максимум)
Расстояние передачи	15 м (максимум)
Характер сигнала	несимметричный по напряжению
Количество драйверов	1
Количество приемников	1
Схема соединения	полный дуплекс, от точки к точке



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	-	-

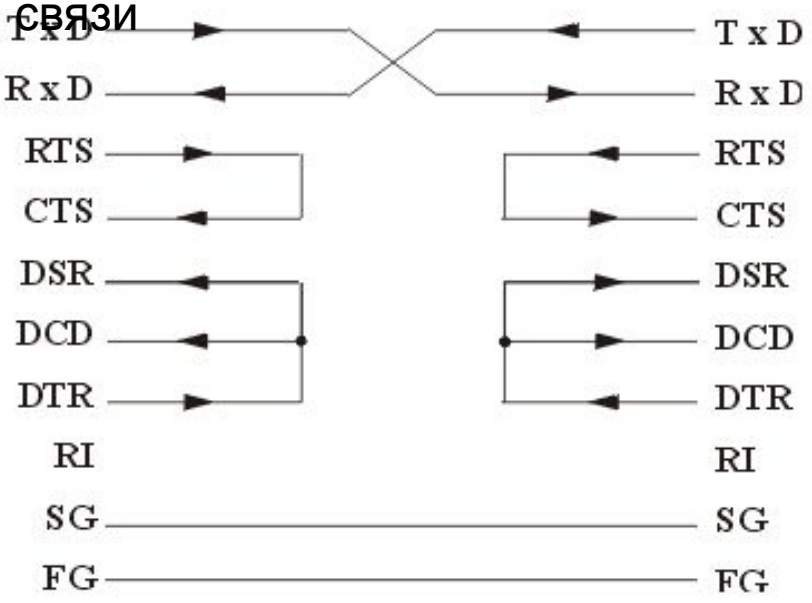


DB9 Розетка (мама)

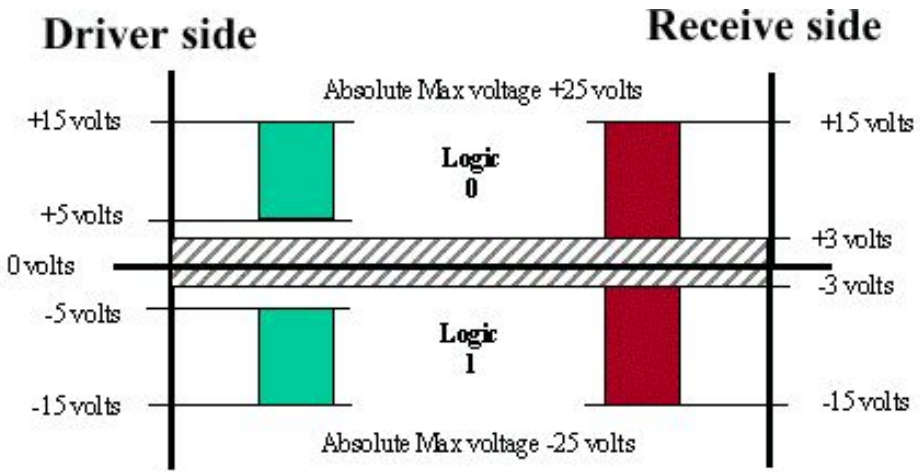
Контакт	Обозн.	Направление	Описание
1	CD	<--	Carrier Detect
2	RXD	<--	Receive Data
3	TXD	-->	Transmit Data
4	DTR	-->	Data Terminal Ready
5	GND	---	System Ground
6	DSR	<--	Data Set Ready
7	RTS	-->	Request to Send
8	CTS	<--	Clear to Send
9	RI	<--	Ring Indicator

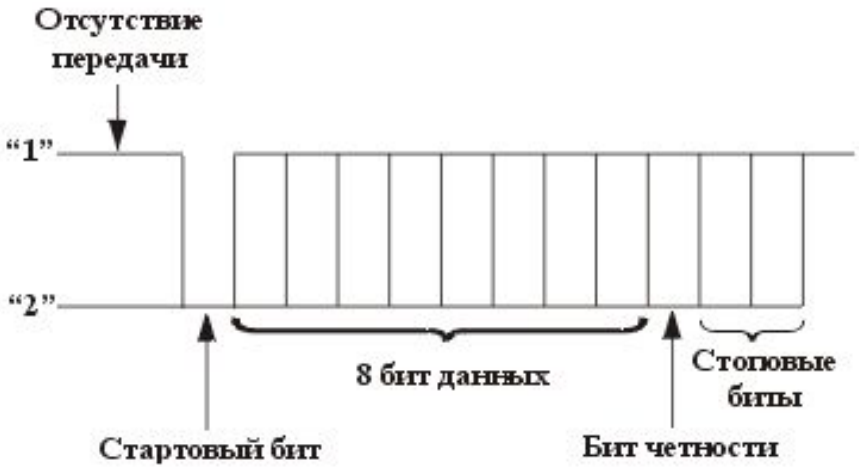
Интерфейс RS-232-C

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.





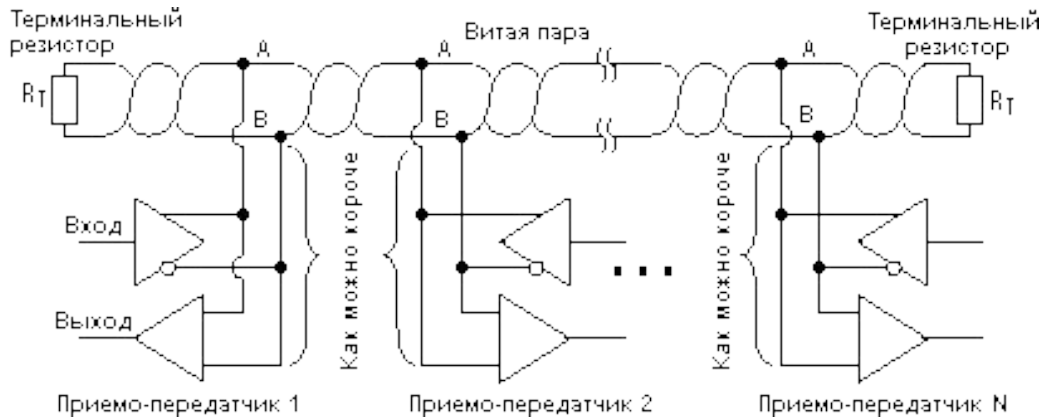
Интерфейс RS-485

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.

Интерфейс RS-485 обеспечивает 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.

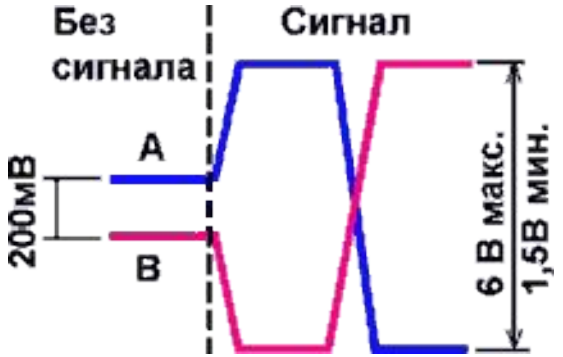
Один передатчик рассчитан на управление 32 стандартными приемниками. Стандарт не нормирует формат информационных кадров и протокол обмена. Для передачи байтов данных используются фреймы RS-232: стартовый бит, биты данных, бит паритета (если нужно), стоповый бит.

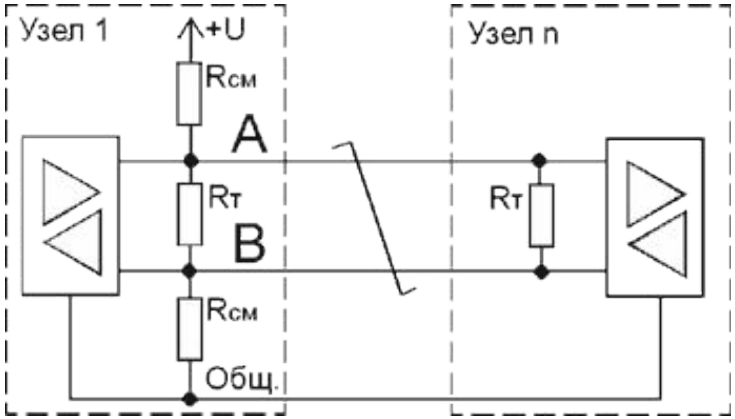
Протоколы обмена в большинстве систем работают по принципу "ведущий-ведомый". Одно устройство на магистрали является ведущим (master) и инициирует обмен посылкой запросов подчиненным устройствам (slave), которые различаются логическими адресами (протокол Modbus RTU). Тип соединителей и распайка не оговариваются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



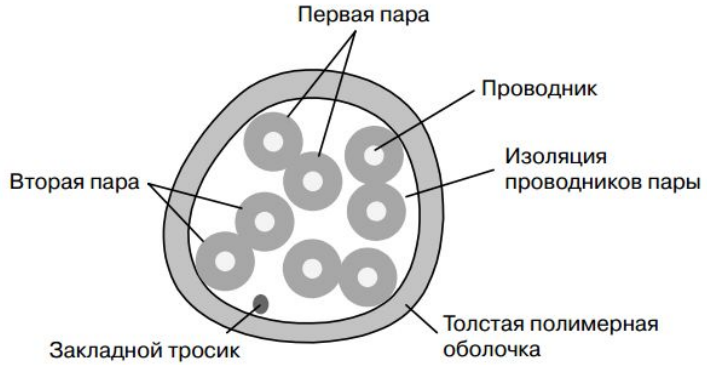
Уровни сигналов

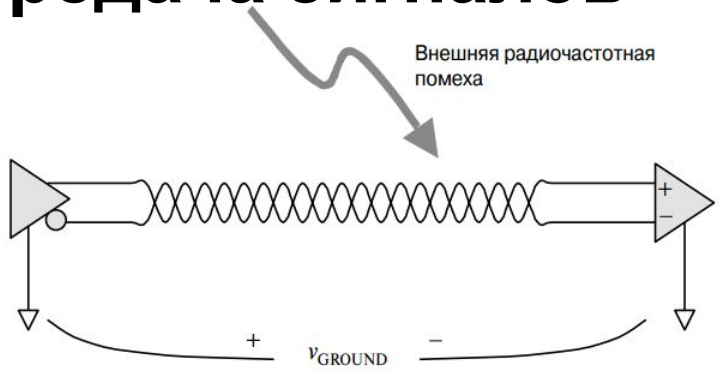
Интерфейс RS-485 использует балансную (дифференциальную) схему передачи сигнала. Это означает, что уровни напряжений на сигнальных цепях A и B меняются в противофазе, как показано на приведенном ниже рисунке:





Дифференциальная передача сигналов





В двухпроводной линии передачи дифференциальное напряжение d определяется как разница между мгновенными напряжениями a и b на обоих проводниках:

$$d = a - b, \tag{6.2}$$

где a и b — напряжения на проводниках, измеренные по отношению к произвольно выбранному общему опорному уровню.

В двухпроводной линии передачи синфазное напряжение c определяется как среднее арифметическое мгновенных напряжений a и b на обоих проводниках:

$$c = \frac{a+b}{2}, \tag{6.3}$$

где a и b — напряжения на проводниках, измеренные по отношению к произвольно выбранному общему опорному уровню (обычно по отношению к локальной земле, но в ряде случаев — по отношению к локальному опорному слою или другой локальной опорной точке).

ис. 6.8. Хороший дифференциальный приемник нейтрализует обой шум, поражающий в равной степени оба проводника диф-

нечетный сигнал: $$o = \frac{a-b}{2},$$

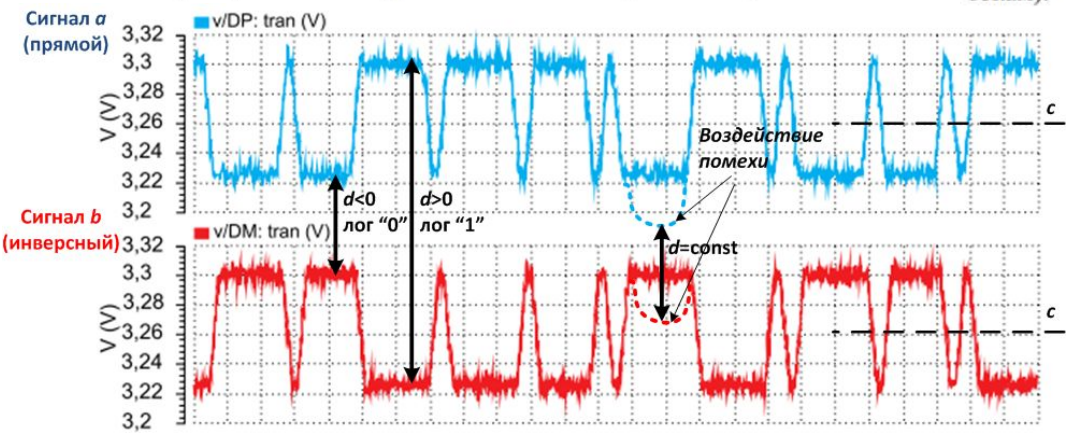
четный сигнал: $$e = \frac{a+b}{2},$$

сигнал в первом проводнике: $$a = e + o,$$

сигнал во втором проводнике: $$b = e - o,$$

Дифференциальное и синфазное напряжения совместно представляют собой вариант описания исходного сигнала, часто называемый *декомпозицией* исходного сигнала. По известным дифференциальному и синфазному напряжениям сигнала можно восстановить напряжения a и b (такое же разложение применимо и к токам).

$$a = c + d/2$$
$$b = c - d/2$$



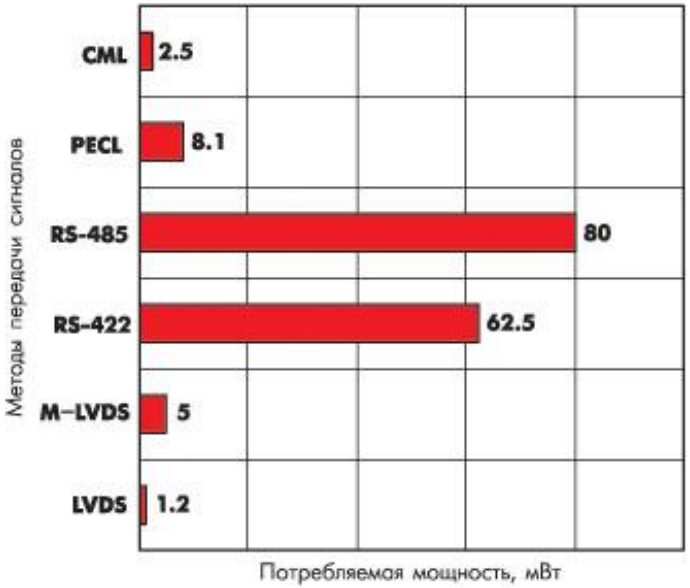
Низковольтная дифференциальная передача сигналов

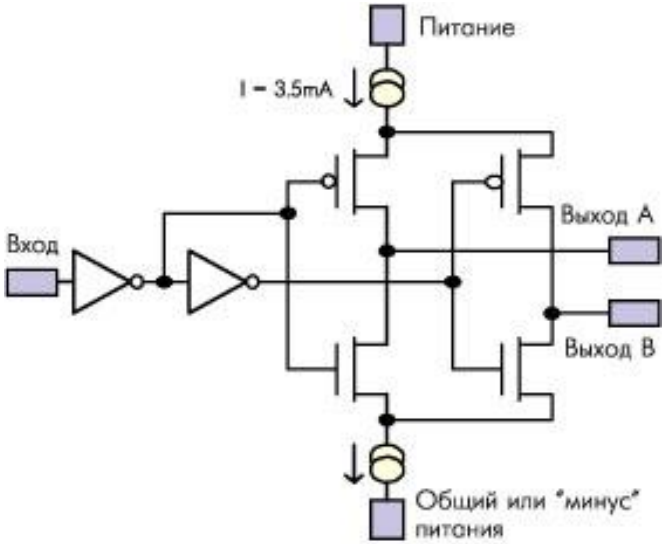
LVDS (Low Voltage Differential Signaling) - передача информации дифференциальными сигналами малых напряжений (до 350 мВ) на двух линиях печатной платы или сбалансированного кабеля со скоростью до сотен и даже нескольких тысячь мегабит в секунду (Mbps). Выходной ток передатчика составляет от 2,47 до 4,54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

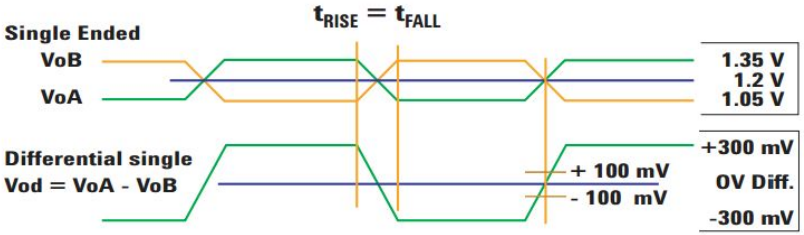
Стандарты:

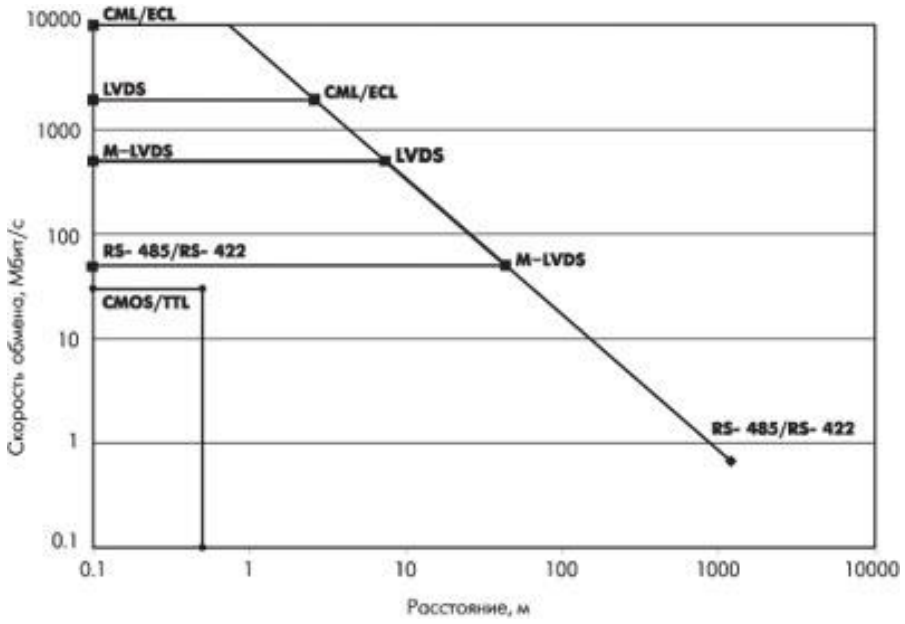
TIA/EIA (Telecommunications Industry Association/Electronic Industries Association) - ANSI/TIA/EIA-644 (LVDS)

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



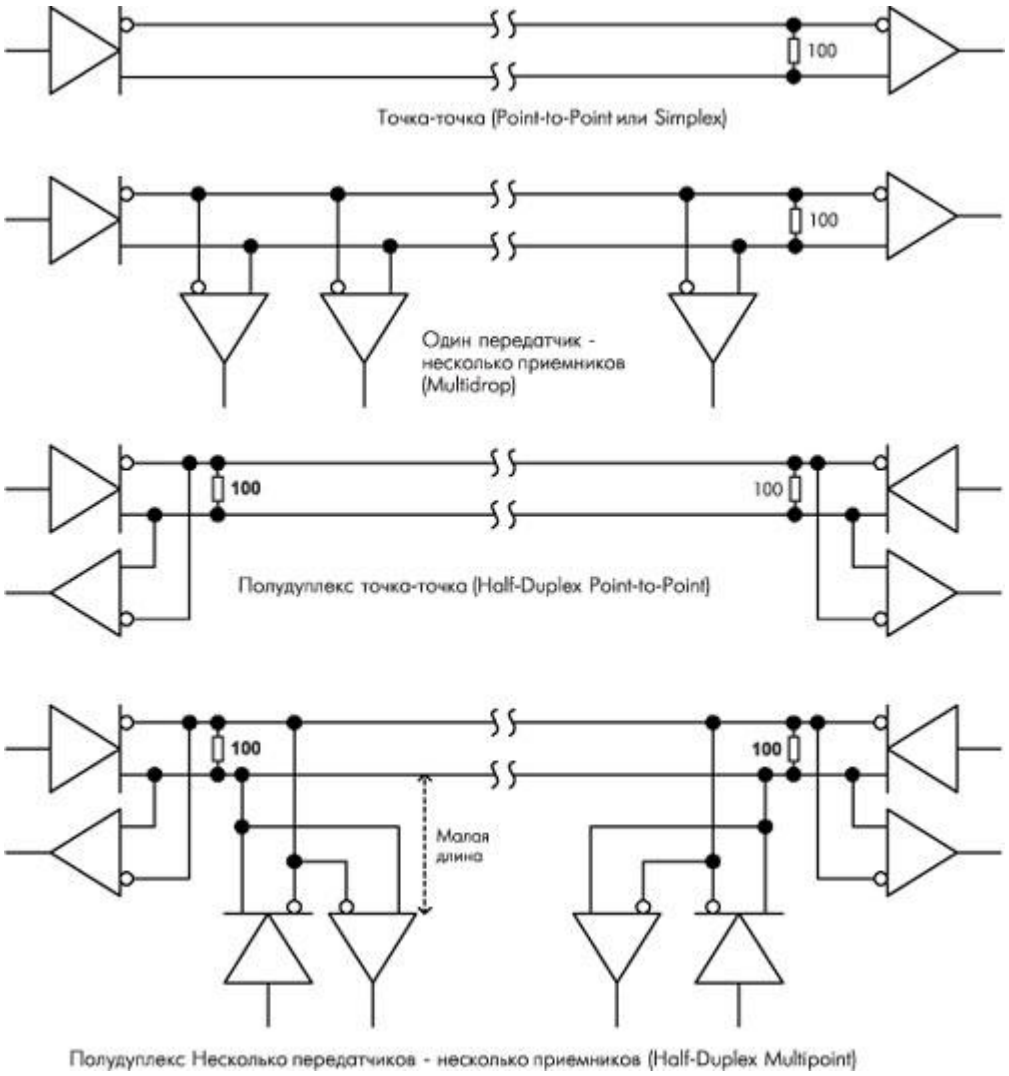






Низковольтная дифференциальная передача сигналов

Подключения



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)

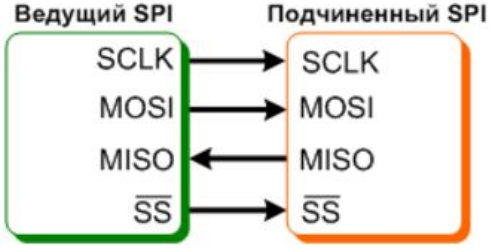
Параметр	Наименование	Мин.	Макс.	Ед. изм.
Vod	Дифференциальное выходное напряжение	247	454	мВ
Vos	Опорное напряжение	1.125	1.375	В
DVod	Изменение VoD		50	мВ
DVos	Изменение VoS		50	мВ
I_{SB}	Ток короткого замыкания		24	мА
Параметр	Наименование	Мин.	Макс.	Ед. изм.
tr, tf	Длительность выходного фронта/ 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

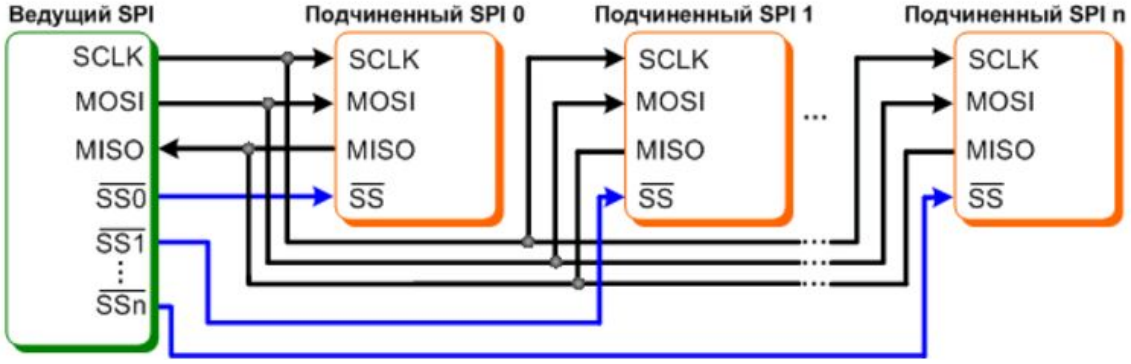
Интерфейс SPI

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

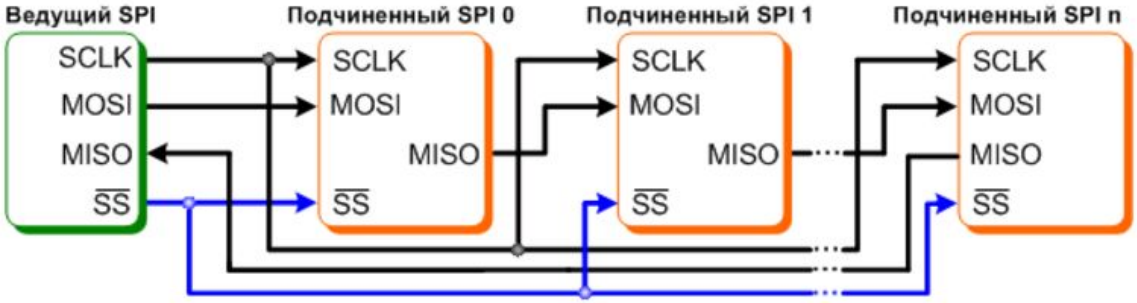
MOSI — выход ведущего, вход ведомого (*Master Out Slave In*) для передачи данных от ведущего устройства ведомому;
MISO — вход ведущего, выход ведомого (*Master In Slave Out)* для передачи данных от ведомого устройства ведущему;
SCLK — последовательный тактовый сигнал (*Serial Clock*) для передачи тактового сигнала для ведомых устройс
CS или **SS** — выбор микросхемы, выбор ведомого (*Chip Select, Slave Select*)

Простейшее подключение





Независимое

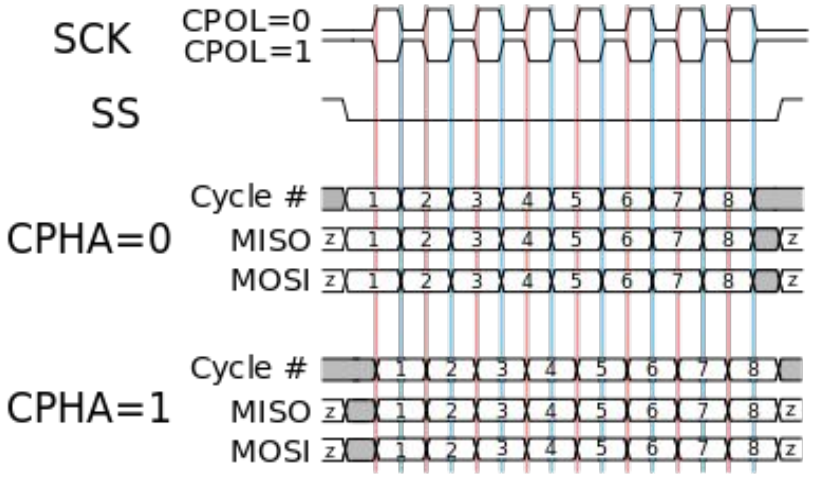


Каскадное

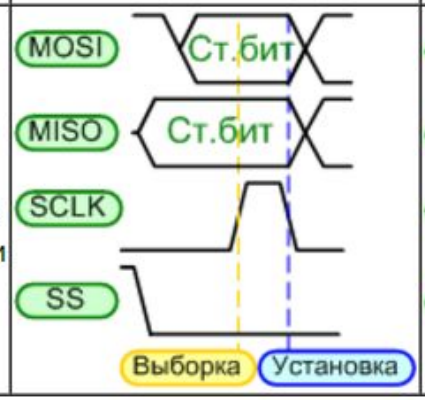
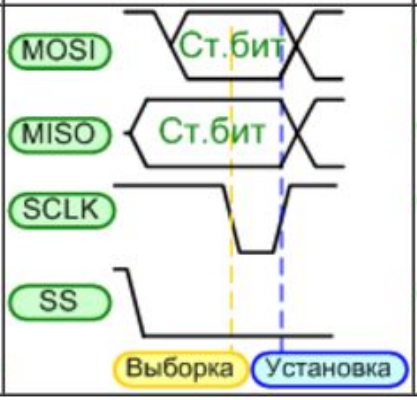
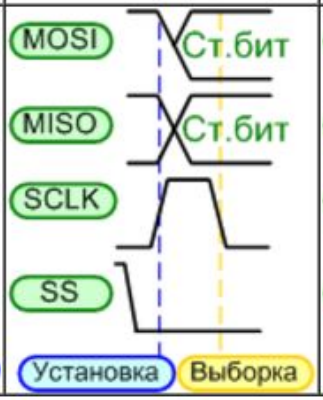
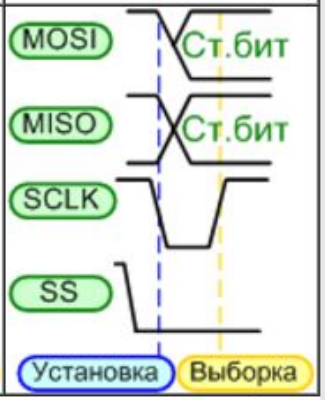
Интерфейс SPI

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 (если CPOL=0, то линия синхронизации до начала цикла передачи и после его окончания имеет низкий уровень (т.е. первый фронт нарастающий, а последний - падающий), иначе, если CPOL=1, - высокий (т.е. первый фронт падающий, а последний - нарастающий));
CPHA - фаза синхронизации; от этого параметра зависит, в какой последовательности выполняется установка и выборка данных (если CPHA=0, то по переднему фронту в цикле синхронизации будет выполняться выборка данных, а затем, по заднему фронту, - установка данных; если же CPHA=1, то установка данных будет выполняться по переднему фронту в цикле синхронизации, а выборка - по заднему).

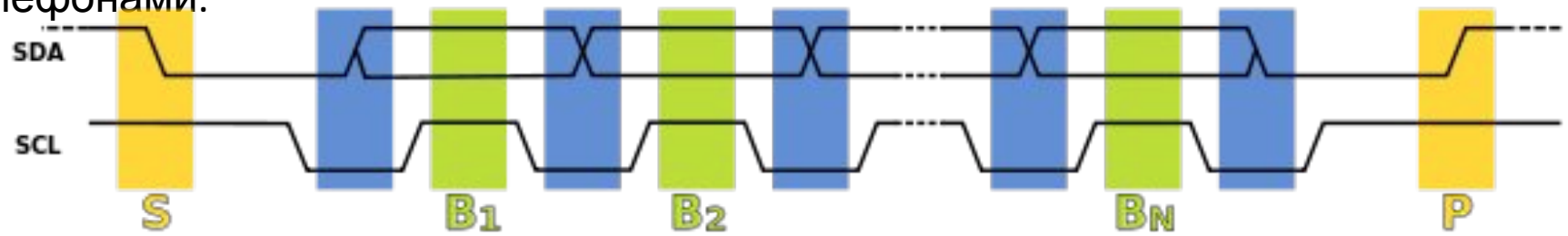


Ведущая и подчиненная микросхемы, работающие в различных режимах SPI, являются несовместимыми.

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.



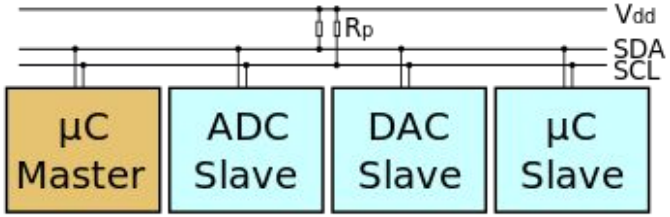
Разработана фирмой Philips в начале 1980-х как простая шина внутренней связи для создания управляющей электроники. Версия 1998 г. стандарта 2.0 - 3,4 Мбит/с, до 127 устройств, напряжения +5 В или +3,3 В.

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.

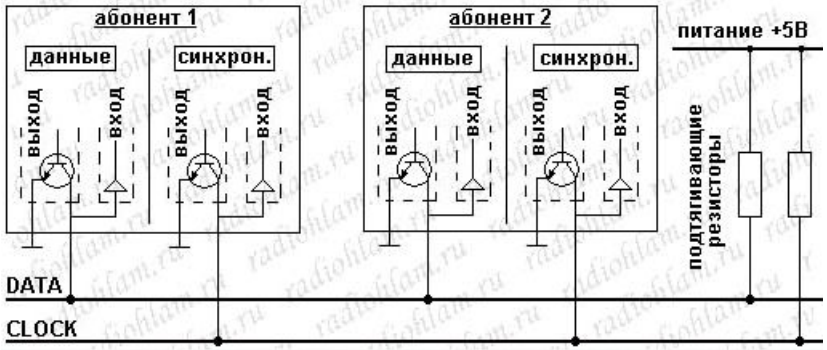
Интерфейс I2C

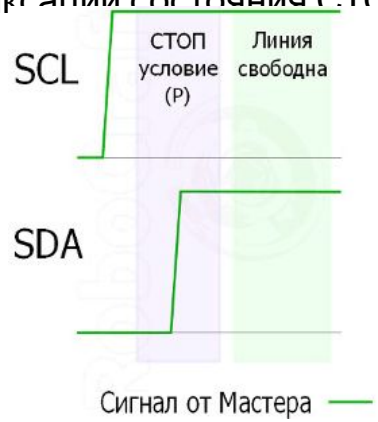
Две двунаправленные линии, подтянутые к напряжению питания и управляемые через открытый коллектор или открытый сток — последовательная линия данных (SDA, *Serial DAta*) и последовательная линия тактирования (SCL, *Serial CLock*).



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

1) Сеанс передачи от "Master" к "Slave"

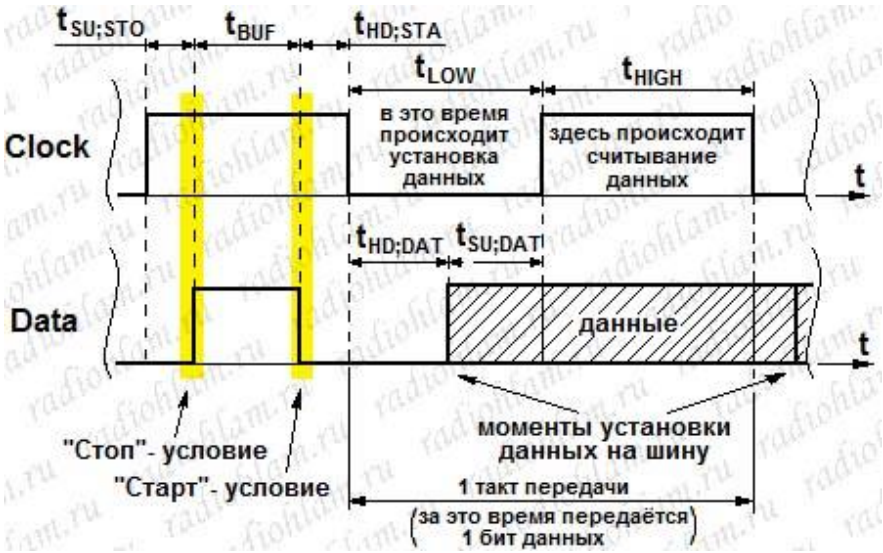
2) Сеанс передачи от "Slave" к "Master" (чтение из "Slave")

- **S** — "Старт" - условие
- **P** — "Стоп" - условие
- **A** — бит подтверждения ($\overline{ACK}$)
- **Ā** — отсутствие подтверждения

Цветом ячейки обозначено - какое именно устройство выставляет даный бит на шину DATA:

- бит выставляется "Master" - устройством
- бит выставляется "Slave" - устройством

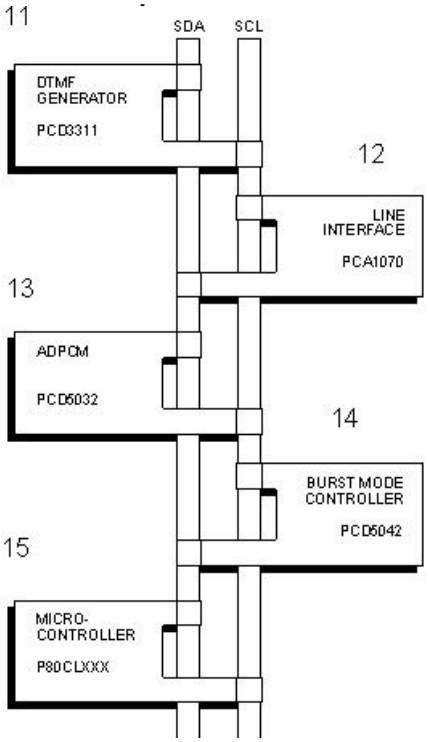
Интерфейс I2C временная диаграмма



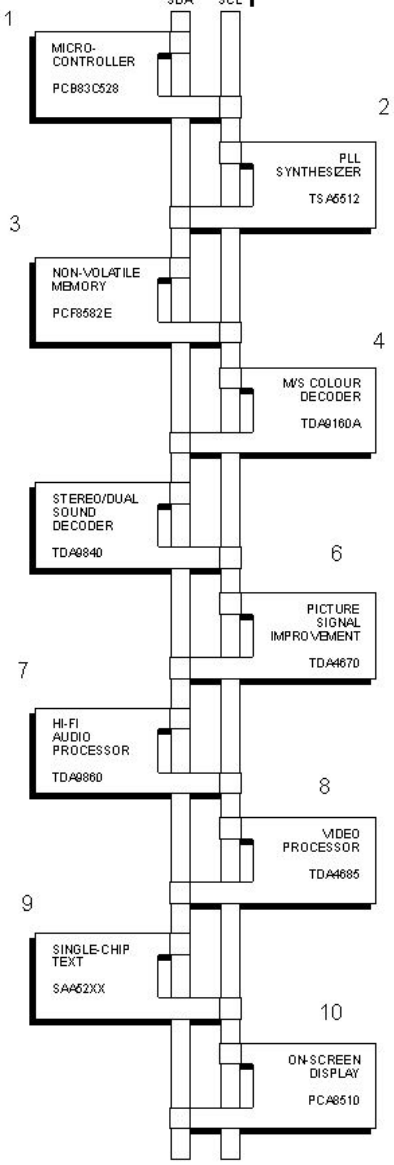
Минимальные значения времени в таблице указаны для максимальной скорости передачи 100 кбит/с.

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



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

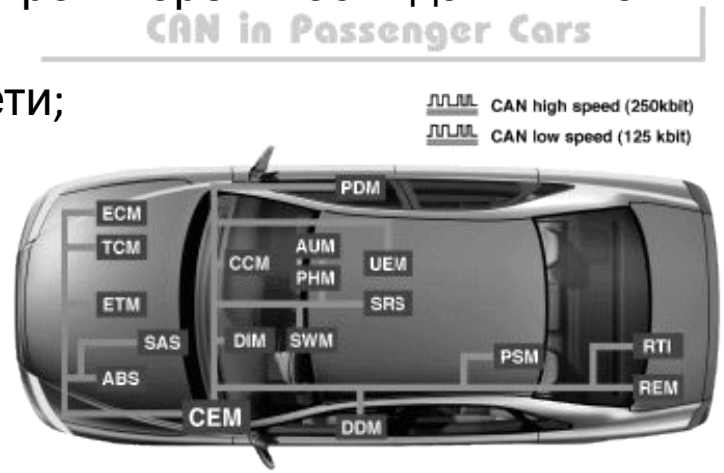
Свойства:

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

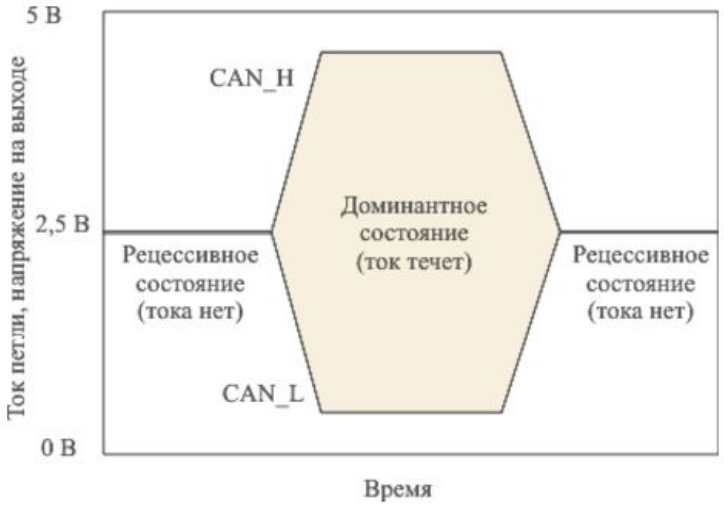


Расстояние, м	25	50	100	250	500	1000	2500	5000
Скорость, Кбит/с	1000	800	500	250	125	50	20	10

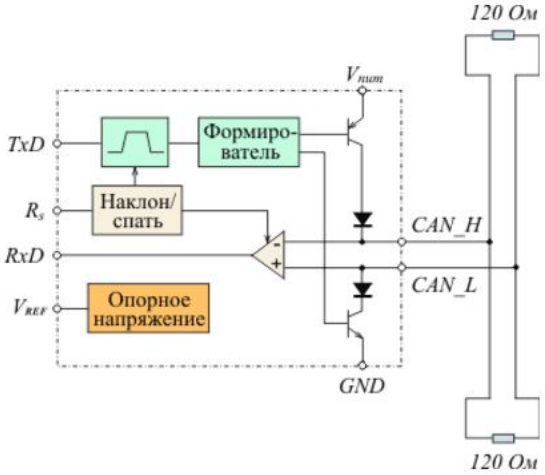
Если один из передатчиков устанавливает в сети логический ноль, а второй - логическую единицу, то это состояние *не является аварийным* - линия остается в состоянии логической единицы

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



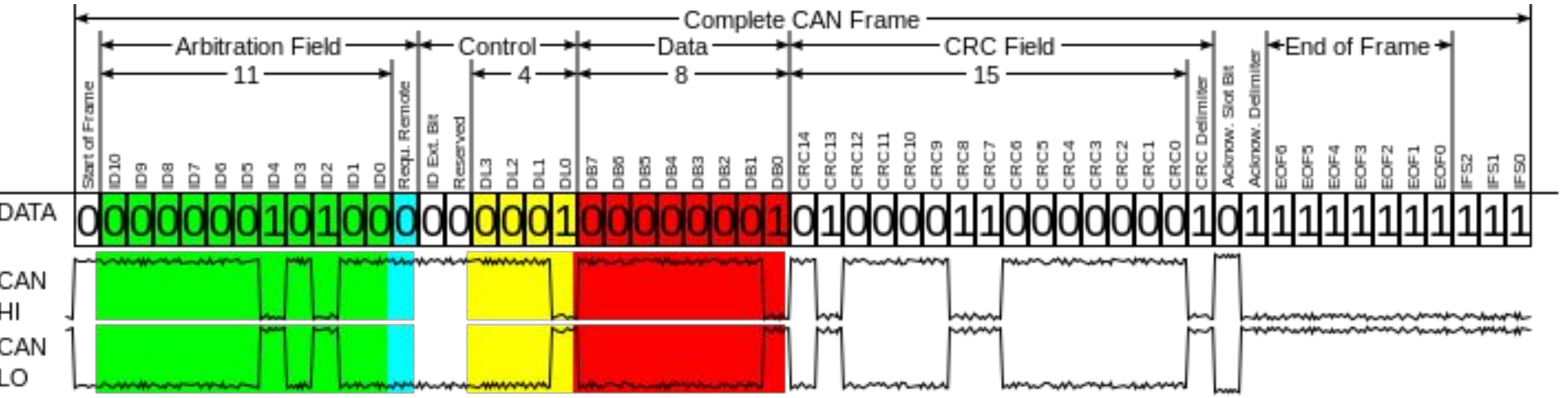
Параметр	Обозн.	Ед. измерения	Мин.	Ном.	Макс	Условие
Для рецессивного состояния шины						
Потенциалы на вых. передатчика	CAN_H	В	2,0	2,5	3	Без нагрузки
	CAN_L	В	2,0	2,5	3	
Диф. напряжение на выходе передатчика	Vdiff	мВ	-500	0	50	Без нагрузки
Диф. напряжение на вх. приемника	Vdiff	В	-1	-	0,5	Без нагрузки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

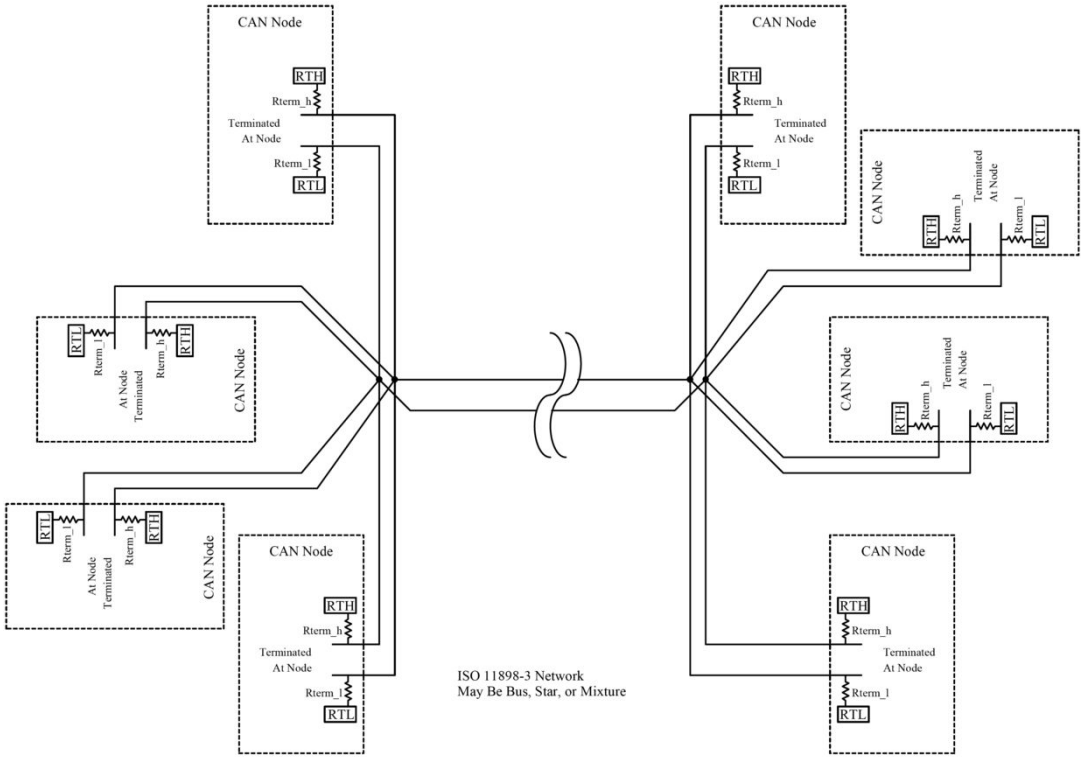
Интерфейс CAN. Протокол

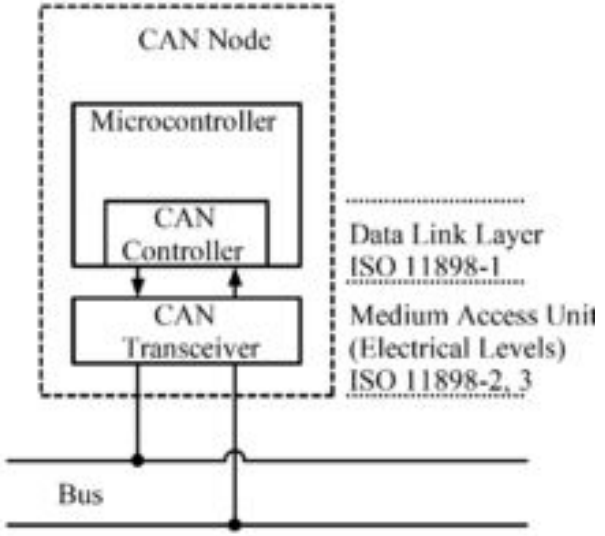
Виды кадров
Кадр данных (data frame) — передаёт данные;
Кадр удаленного запроса (remote frame) — служит для запроса на передачу кадра данных с тем же идентификатором;
Кадр перегрузки (overload frame) — обеспечивает промежуток между кадрами данных или запроса;
Кадр ошибки (error frame) — передаётся узлом, обнаружившим в сети ошибку.
Кадры данных и запроса отделяются от предыдущих кадров *межкадровы*
и 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		

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 корпорацией Dallas Semiconductor в конце 90-х.

Топология сети — общая шина. Сеть устройств 1-Wire со связанным основным устройством названа «MicroLan».

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.

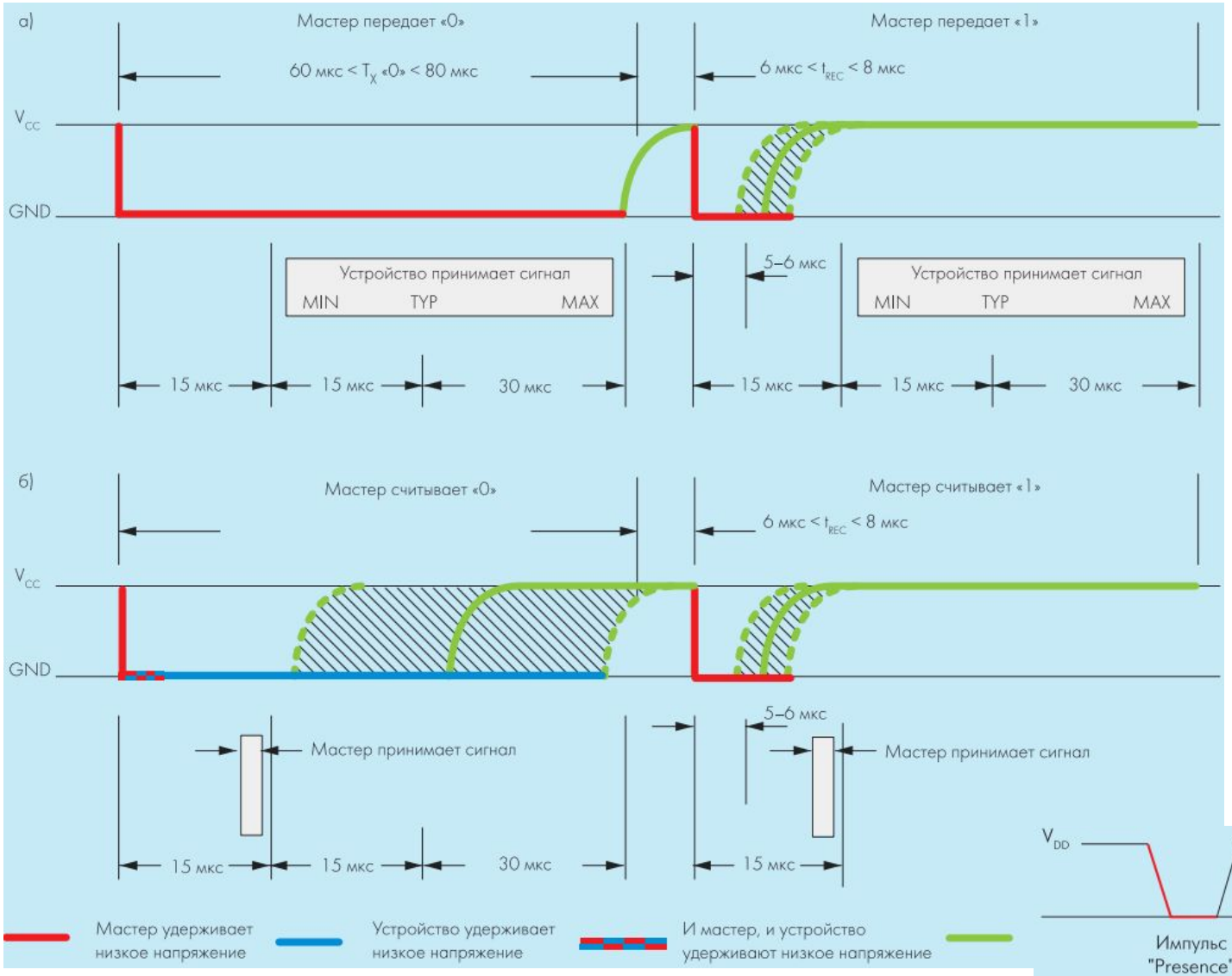
Расстояние до 300 м при условиях:

- применение специального кабеля IEEE1394 (Firewire);
- использование специального драйвера сети (активное подтяжка с мощным током);
- топология "общая шина" с единым стволом.

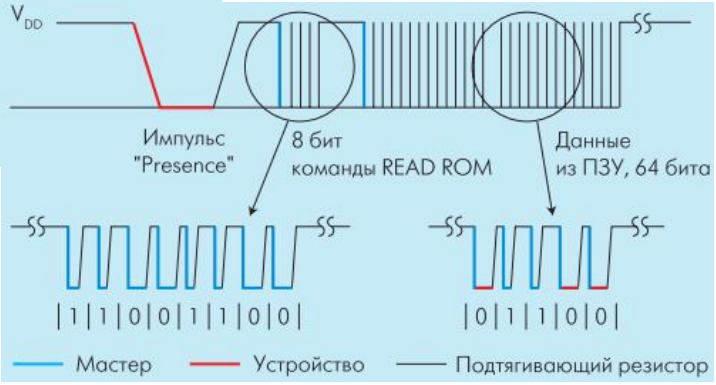
Кабель категории 5 (Cat. 5) — тип кабеля для передачи сигналов, состоящий из 4-х витых пар. CAT-5, Разъем

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

Интерфейс 1-wire протокол

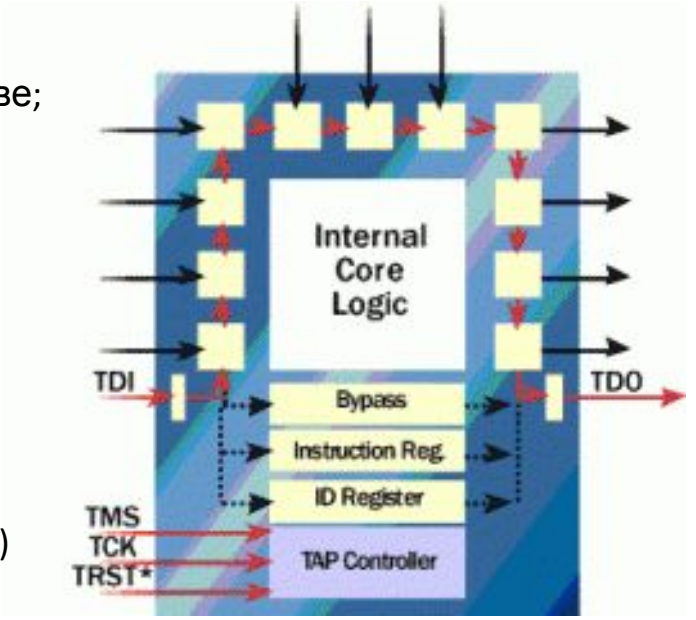


Передача информационных битов по шине 1-Wire: а – мастер передает сигналы, б – мастер считывает сигналы



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;

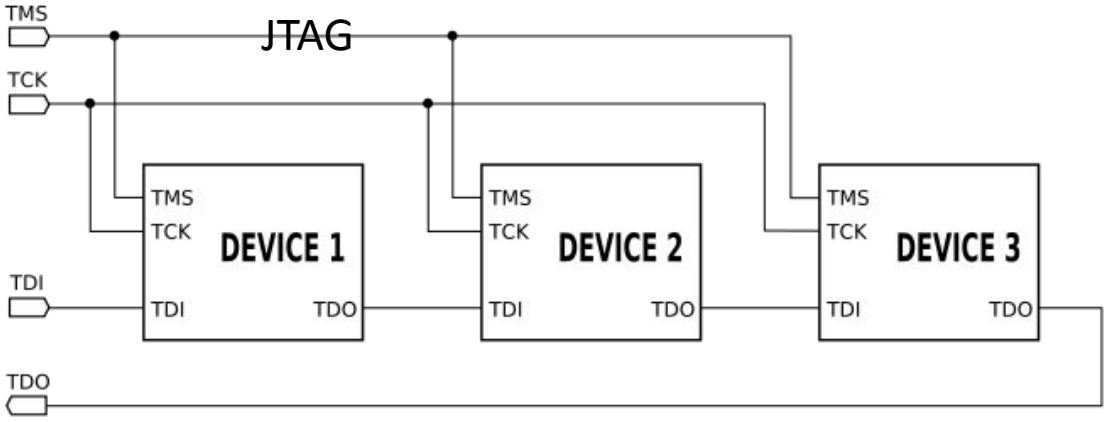
TCK (*test clock* — «тестовое тактирование») — тактирует работу встроенного автомата управления периферийным сканированием. Максимальная частота сканирования периферийных ячеек зависит от используемой аппаратной части и на данный момент ограничена 25…40 МГц;

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

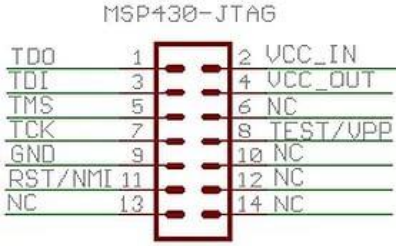
Общая цепочка JTAG

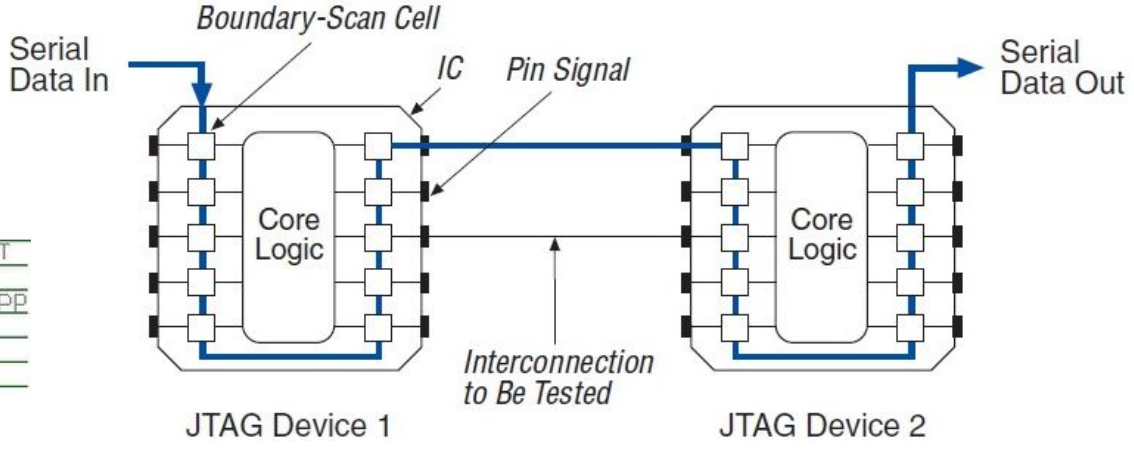


Возможность программирования микроконтроллера (или ПЛИС) и подключённой к его выводам микросхемы флэш-памяти. Два способа программирования флэш-памяти с использованием JTAG: через загрузчик с последующим обменом данными через память процессора, либо через прямое управление выводами микросхемы.

JTAG-USB переходник

Разъем





Проектирование МПС. Уровни представления МПС

В начальной стадии проектирования МПС на концептуальном уровне. В процессе разработки происходит переход от одного уровня ее представления к другому, более детальному. Каждая абстракция несет в себе только информацию, которая соответствует данному уровню.

Уровни абстрактного представления МПС:

1) **"черный ящик"** (внешние спецификации; внешние характеристики);

2) **Структурный** – компоненты МПС: микропроцессорами, ЗУ, УВВ, внешние ЗУ, каналы связи; создается МПС, описывается функциями отдельных устройств и их взаимосвязью, информационными потоками.

3) **Программный** разделяется на два подуровня: команд процессора и языковой. МПС интерпретируется как последовательность операторов или команд, вызывающих то или иное действие над некоторой структурой данных;

4) **Логический,** присущ дискретным системам. **Подуровень переключательных схем** образуется вентилями и построенными на их основе операторами обработки данных. Переключательные схемы подразделяются на комбинационные и последовательностные (с памятью). Поведение системы описывается алгеброй логики, моделью конечного автомата, входными/выходными последовательностями 1 и 0. Комбинационные схемы представляются таблицей истинности. Последовательностные схемы могут описываться диаграммами или таблицами входов/выходов, в которых определены взаимно однозначные соответствия между входами схемы, внутренними состояниями (комбинациями значений элементов памяти) и выходами.

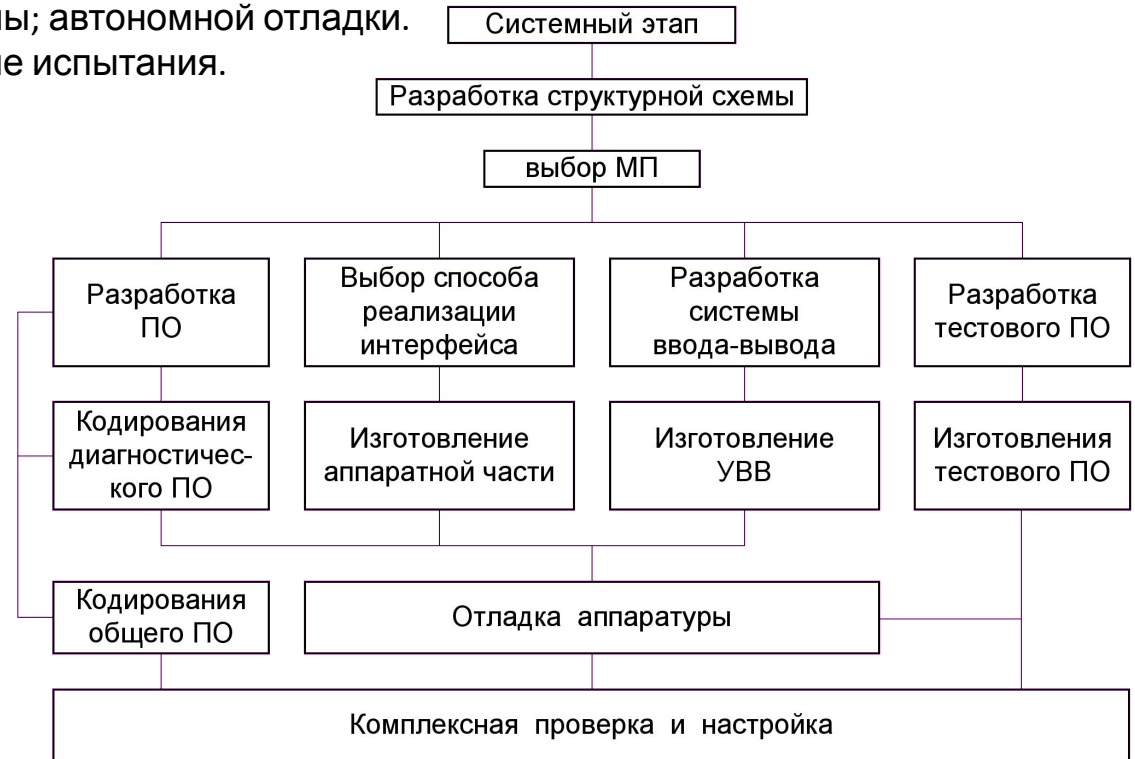
Подуровень регистровых пересылок характеризуется более высокой степенью абстрагирования и представляет собой описание регистров и передачу данных между ними. *Информационная часть* образуется регистрами, операторами и путями передачи данных. *Управляющая* определяет зависящие от времени сигналы, инициирующие пересылку данных между регистрами.

5) **Схемный** - резисторы и конденсаторы. Показателями поведения системы на этом уровне служат напряжение и ток, представляемые в функции времени или частоты. Этот уровень

Этапы проектирование МПС

1. Формализация требований к системе (составляются внешние спецификации, перечисляются функции системы, формализуется техническое задание (ТЗ) на систему, формально излагаются замыслы разработчика в официальной документации).
2. Разработка структуры и архитектуры системы (определяются функции отдельных устройств и программных средств, выбираются микропроцессорные наборы, на базе которых будет реализована система, определяются взаимодействие между аппаратными и программными средствами, временные характеристики отдельных устройств и программ).
3. Разработка и изготовление аппаратных средств и программного обеспечения системы (после определения функций, реализуемых аппаратурой, и функций, реализуемых программами, схемотехники и программисты одновременно приступают к разработке и изготовлению соответственно опытного образца и программных средств. Разработка и изготовление аппаратуры состоят из разработки структурных и принципиальных схем, изготовления прототипа, автономной отладки. Разработка программ состоит из разработки алгоритмов; написания текста исходных программ; трансляции исходных программ в объектные программы; автономной отладки.
4. Комплексная отладка и приемосдаточные испытания.

Основные приемы:
1) останов функционирования системы при возникновении определенного события;
2) чтение (изменение) содержимого памяти или регистров системы;
3) пошаговое отслеживание поведения системы;
4) отслеживание поведения системы в реальном времени;
5) временное согласование программ.
Комплексная отладка завершается приемосдаточными испытаниями, показывающими соответствие спроектированной системы техническому заданию. Для проведения комплексной отладки МПС используют логические анализаторы и комплексы: оценочные, отладочные, развития



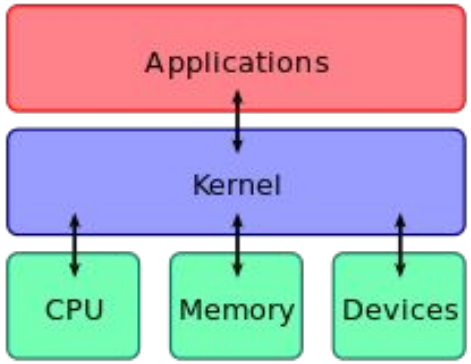
Операционная система МПС. Общие сведения.

Операционная система - комплекс взаимосвязанных программ, предназначенных для управления ресурсами МПС и организации взаимодействия с пользователем. ОС между устройствами с их микроархитектурой, машинным языком и драйверами — с одной стороны — и прикладными программами с другой.

Драйвер - компьютерное ПО, с помощью которого ОС получает доступ к аппаратному обеспечению некоторого устройства.

Разработчикам ПО ОС позволяет абстрагироваться от деталей реализации и функционирования устройств, предоставляя минимально необходимый набор функций. В сложных вычислительных МПС ОС является основной, наиболее важной (а иногда и единственной) частью системного программного обеспечения.

Ядро (kernel) — центральная часть ОС, обеспечивающая приложениям координированный доступ к ресурсам МПС, таким как процессорное время, память, внешнее аппаратное обеспечение, внешнее устройство ввода и вывода информации. Также обычно ядро предоставляет сервисы файловой системы и сетевых протоколов.



API (интерфейс программирования приложений, интерфейс прикладного программирования)— набор готовых классов, процедур, функций, структури констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах. Используется программистами при написании всевозможных приложений.

Операционная система МПС. Функции.

Основные функции:

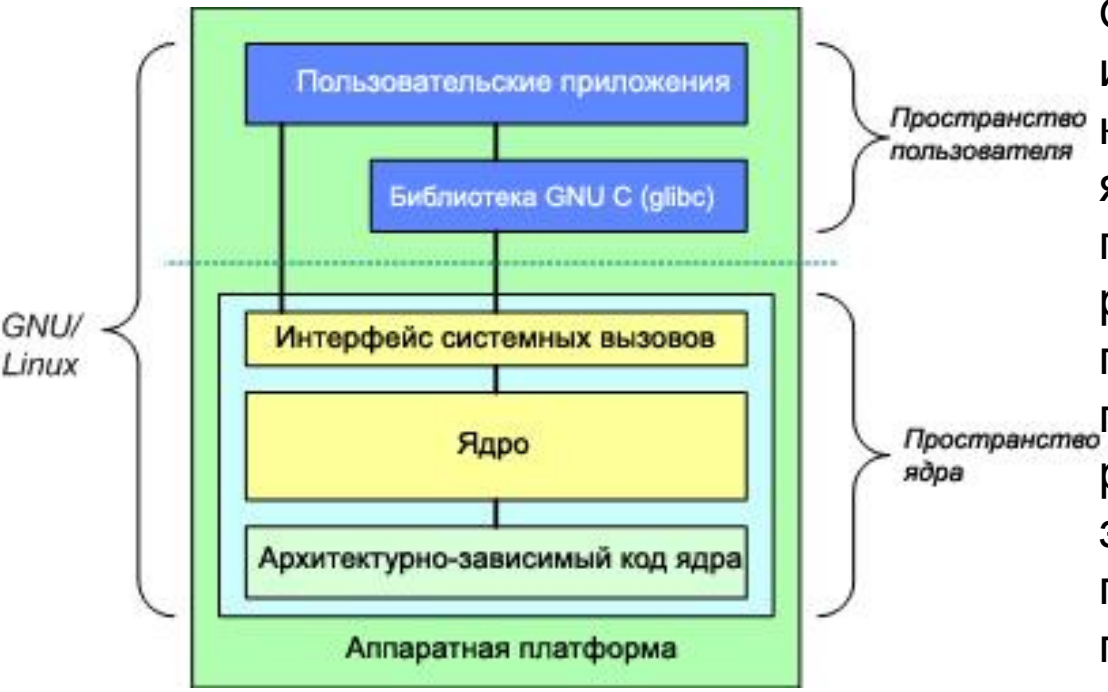
- Исполнение запросов программ (ввод и вывод данных, запуск и остановка других программ, выделение и освобождение дополнительной памяти и др.).
- Загрузка программ в оперативную память и их выполнение.
- Стандартизованный доступ к периферийным устройствам (устройства ввода-вывода).
- Управление оперативной памятью (распределение между процессами, организация виртуальной памяти).
- Управление доступом к данным на энергонезависимых носителях (таких как НЖМД, SSD, оптические диски и др.), организованным в той или иной файловой системе.
- Обеспечение пользовательского интерфейса.
- Сохранение информации об ошибках системы.

Дополнительные функции:

- Параллельное или псевдопараллельное выполнение задач (многозадачность). Эффективное распределение ресурсов вычислительной системы между процессами.
- Разграничение доступа различных процессов к ресурсам.
- Организация надёжных вычислений (невозможности одного вычислительного процесса намеренно или по ошибке повлиять на вычисления в другом процессе), основана на разграничении доступа к ресурсам.
- Взаимодействие между процессами: обмен данными, взаимная синхронизация.
- Защита самой системы, а также пользовательских данных и программ от действий пользователей (злонамеренных или по незнанию) или приложений.
- Многопользовательский режим работы

ОС Linux

Linux — общее название Unix-подобных ОС, основанных на одноимённом ядре. **UNIX** — семейство переносимых, многозадачных и многопользовательских операционных систем. Ядро Linux создаётся и распространяется в соответствии с моделью разработки свободного и открытого программного обеспечения, соответствующее стандартам POSIX.



Архитектура GNU/Linux

GNU C (glibc)предоставляет интерфейс системных вызовов, который обеспечивает связь с ядром и дает механизм для перехода от приложения, работающего в пространстве пользователя, к ядру. Ядро и пользовательское приложение располагаются в разных защищенных адресных пространствах. Каждый процесс в пространстве пользователя имеет свое собственное виртуальное адресное пространство.

Интерфейс системных вызовов реализует базовые функции, например, чтение и запись. **Архитектурно-независимый код ядра** является общим для всех процессорных архитектур, поддерживаемых Linux. **Архитектурно-зависимый код**, образующий BSP (Board Support Package - пакет поддержки аппаратной платформы) зависит от процессора и платформы для конкретной

ОС Linux. Ядро.

SCI - уровень, предоставляющий средства для вызова функций ядра из пространства пользователя. Этот интерфейс может быть архитектурно зависимым, даже в пределах одного процессорного семейства. SCI представляет собой службу мультиплексирования и демультиплексирования вызова функций. Реализация SCI находится в ./linux/kernel, а архитектурно-зависимая часть - в ./linux/arch.

PM – исполнение процессов (*потоков)*, соответствующих отдельным виртуализованным объектам процессора (код потока, данные, стек, процессорные регистры). Ядро предоставляет интерфейс программирования приложений (API) через SCI для создания нового процесса (порождения копии, запуска на исполнение, вызова функций POSIX, остановки процесса (kill, exit), взаимодействия и синхронизации между процессами (сигналы или механизмы POSIX).

MM - средства управления памятью, аппаратные механизмы для установления соответствия между физической и виртуальной памятью

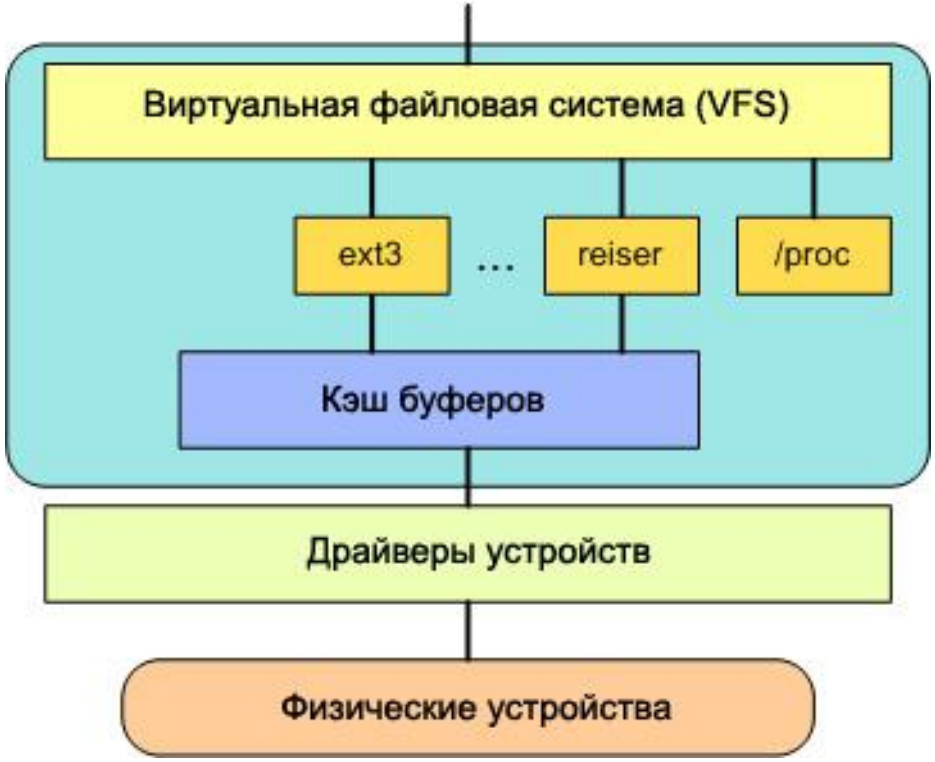
VFS предоставляет общую абстракцию интерфейса к файловым системам, уровень коммутации между SCI и файловыми системами

Сетевой стек имеет многоуровневую архитектуру, повторяющую структуру протоколов IP, TCP.

DD - возможность работы с конкретными аппаратными устройствами (I2C, USB, BlueTooth)

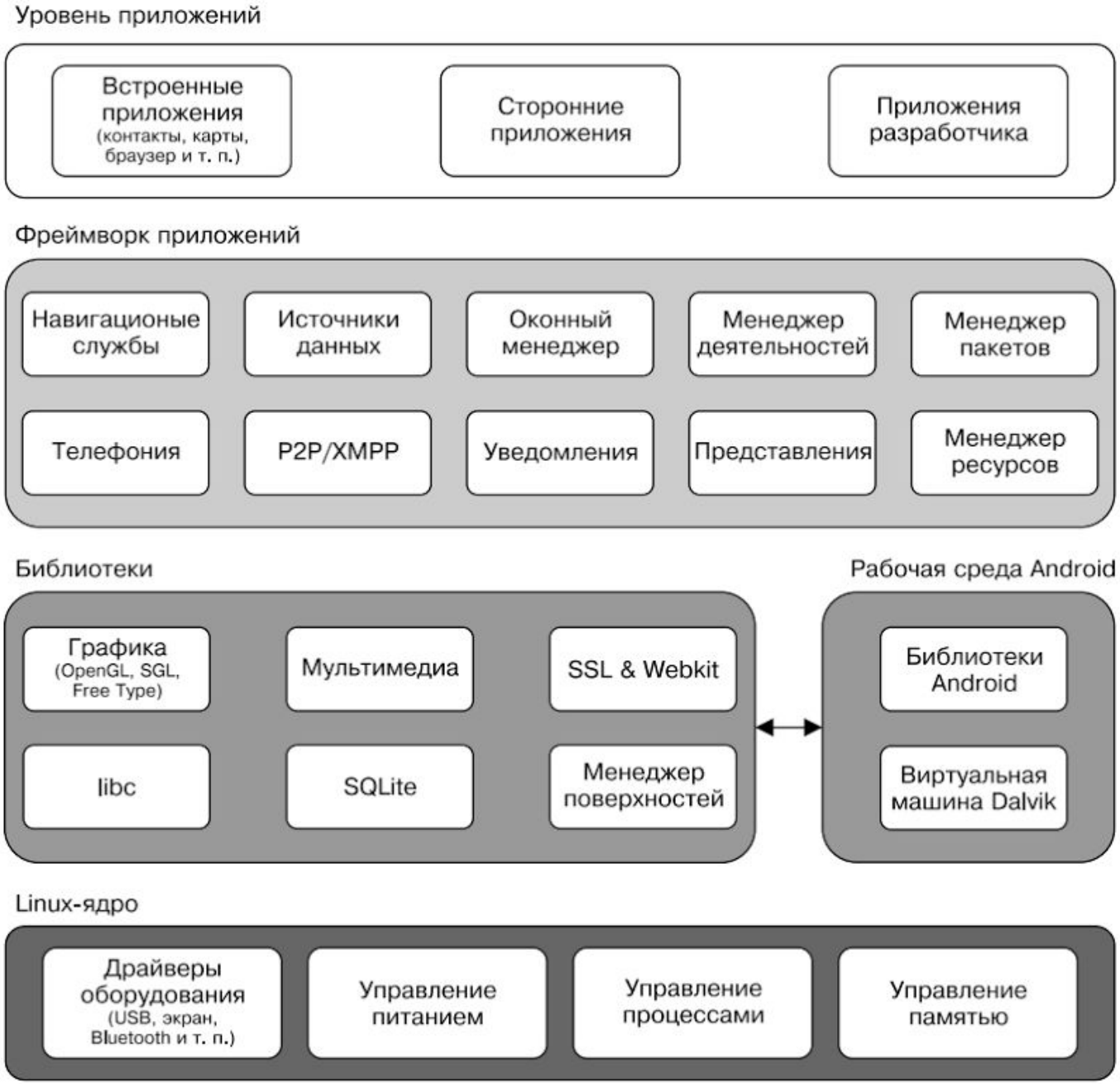
ОС Linux. Виртуальная файловая система.

VFS предоставляет коммутационную матрицу между пользователями и файловыми системами



На верхнем уровне VFS располагается единая API-абстракция таких функций, как открытие, закрытие, чтение и запись файлов. На нижнем уровне VFS находятся абстракции файловых систем, которые определяют, как реализуются функции верхнего уровня. Они представляют собой подключаемые модули для конкретных файловых систем (которых существует более 50). Исходные коды файловых систем находятся в ./linux/fs. Ниже уровня файловой системы находится кэш буферов, предоставляющий общий набор функций к уровню файловой системы (независимый от конкретной файловой системы). Этот уровень кэширования оптимизирует доступ к физическим устройствам за счет краткосрочного хранения данных (или упреждающего чтения, обеспечивающего готовность данных к тому моменту, когда они понадобятся). Ниже кэша буферов находятся драйверы устройств, реализующие интерфейсы для

Структура и состав ОС Android для смартфона



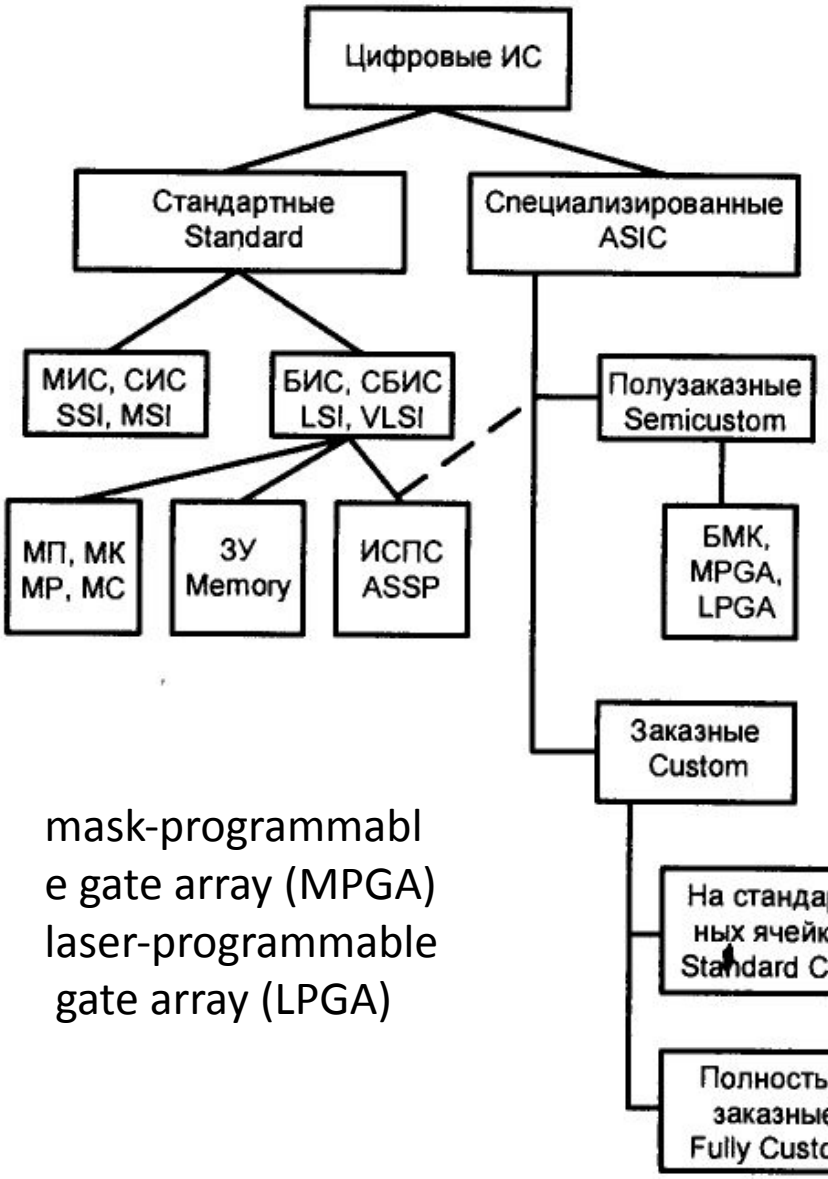
Приложению предоставляются уже реализованные возможности других приложений, к которым разрешено получать доступ.

Фреймворк лишь выполняет написанный для него код приложения, в отличие от библиотек, которые исполняются сами. Фреймворк содержит в себе большое количество библиотек с разной функциональностью и назначением, в то время как библиотеки объединяют в себе наборы функций, близких по логике.

Библиотеки предназначены для обеспечения базового функционала приложений: предоставление реализованных алгоритмов для вышележащих уровней, поддержку файловых форматов, осуществление кодирования и декодирования информации (мультимедийные кодеки), отрисовку графики и др. Библиотеки реализованы на C/C++ и скомпилированы под конкретное аппаратное обеспечение устройства.

LINUX-ядро обеспечивает функционирование системы и отвечает за безопасность, управление памятью, энергосистемой и процессами, а также предоставляет сетевой стек и модель драйверов. Ядро также действует как уровень абстракции

Классификация интегральных схем



mask-programmabl
e gate array (MPGA)
laser-programmable
gate array (LPGA)

Микросхемы малого и среднего уровней интеграции МИС и СИС

БИС/СБИС микропроцессоров и микроконтроллеров, МП и МК (Microprocessors, MP и Microcontrollers, MC). Многочисленные микросхемы МП и МК широко и успешно применяются при решении задач программной реализации алгоритмов.

БИС/СБИС запоминающих устройств

Интегральные схемы с программируемой пользователем структурой (ИСПС). Эти схемы ознаменовали появление нового перспективного направления в развитии элементной базы электронного приборостроения.

Специализированные ИС, как уже отмечалось, в той или иной мере проектируются для конкретного заказчика. При этом от заказчика, как правило, требуются затраты значительных средств при длительных сроках разработки

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

К *полузаказным схемам* относятся базовые матричные кристаллы БМК

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.

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"* либо *"цифровые интегральные схемы с программируемой структурой"*.

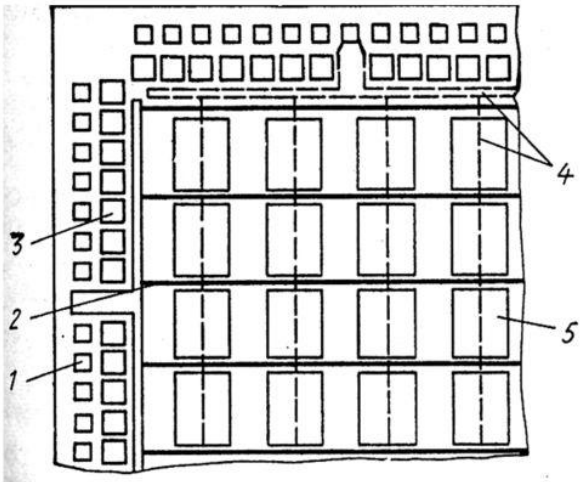
Классификация интегральных схем

БМК

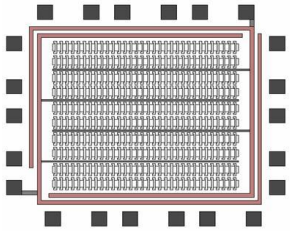
Кристалл МаБИС – БМК, представляет собой многослойную пластину, на которой реализованы несоединенные активные и пассивные компоненты, сгруппированные, как правило, в топологические ячейки (ТЯ).

Определена библиотека функциональных элементов (БФЭ).

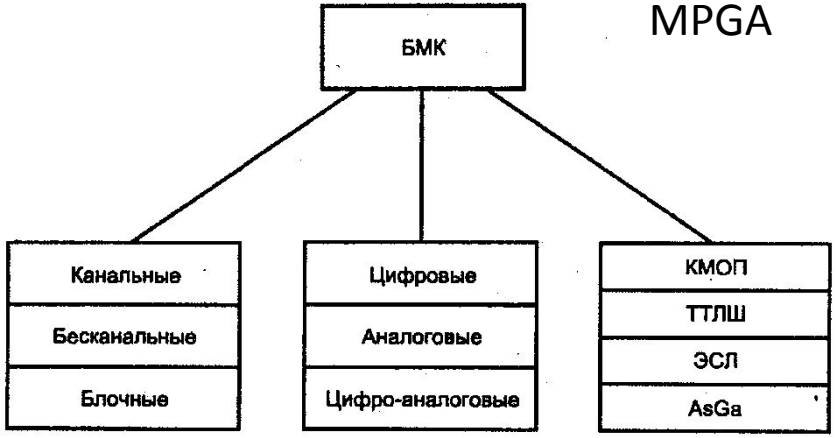
Для ее элементов существуют фотошаблоны – эталоны металлизации соединения компонентов.

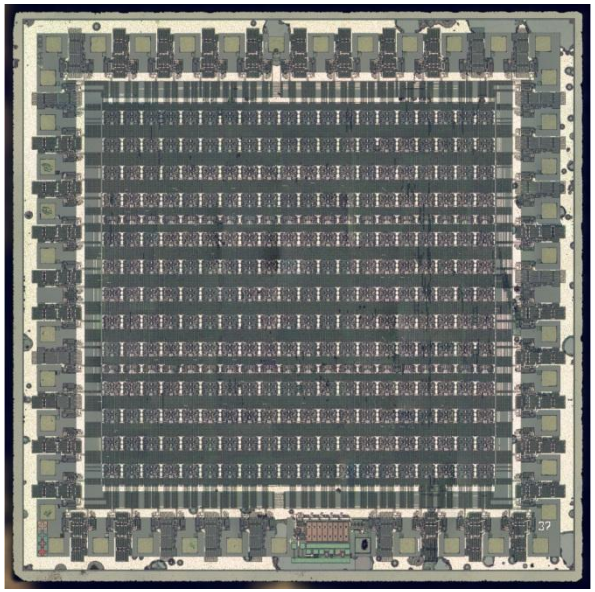


1 — внешний вывод (контактная площадка); *2* — шина питания (первый слой); *3* — периферийная ячейка; *4* — шина «земли» (второй слой); *5* — топологическая (макро) ячейка



MPGA





Программируемые логические матрицы (ПЛМ)

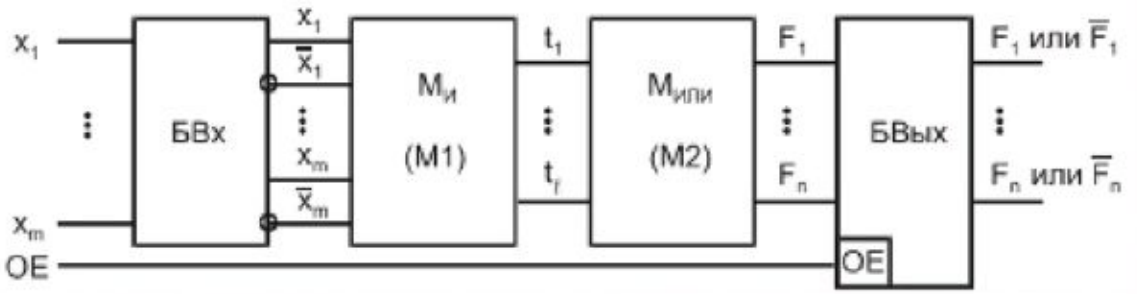


Рис. 2 Базовая структура ПЛМ

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

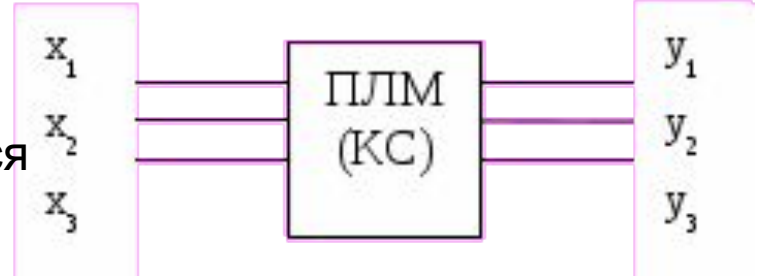
Ми – матрица элементов И (конъюнкторов), на выходе которой формируются λ термов.

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

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, эти «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 (конструктором), без участия изготовителя. При программировании имеющиеся в «полуфабрикатах» ЛЭ организуются в специальные логические структуры, выполняющие заданные логические функции (любой сложности). *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

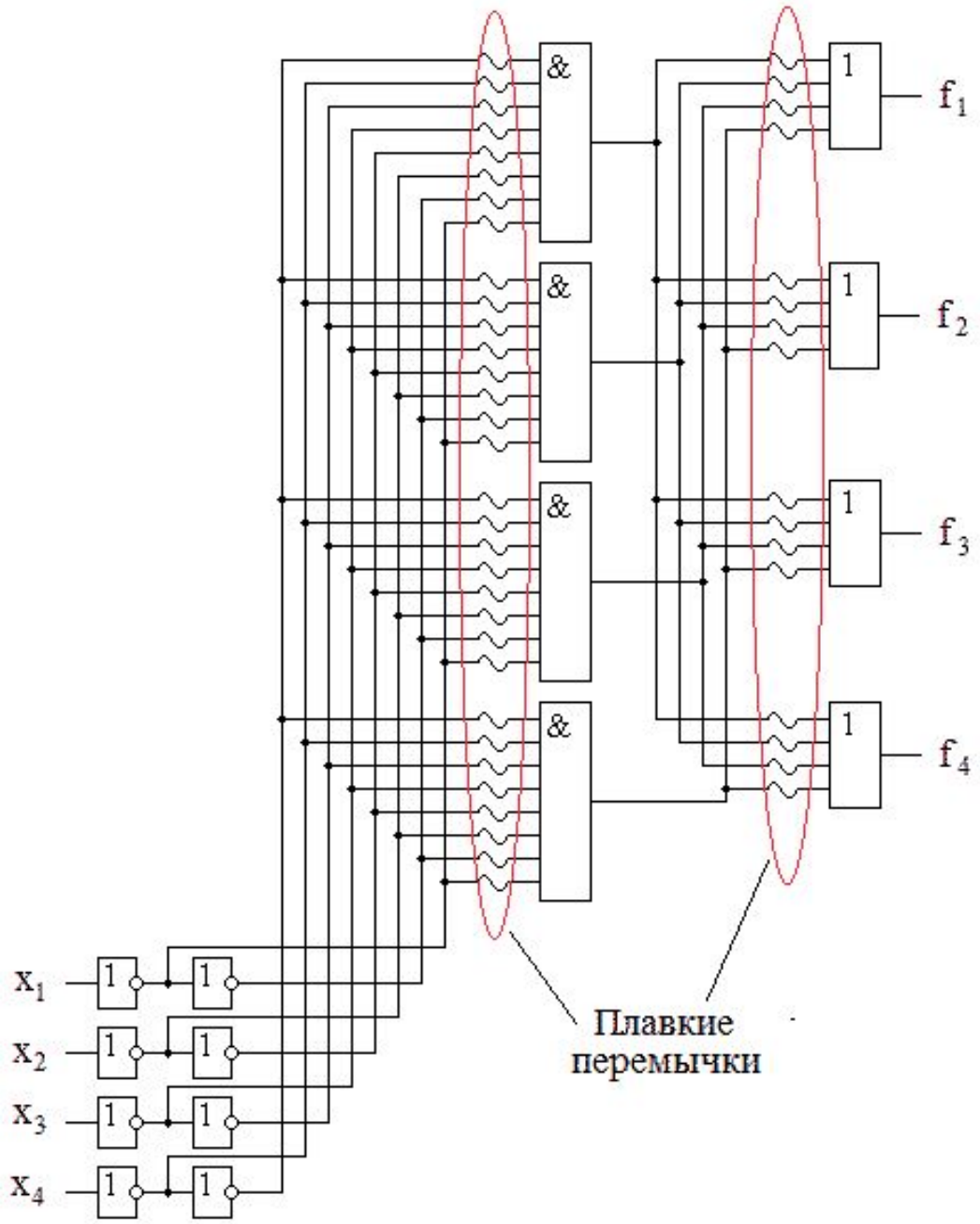
Структура ПЛМ

Основная идея работы ПЛМ (PLA — Programmable logic Array) заключается в реализации логической функции, представленной в СДНФ — дизъюнктивной нормальной форме.

Логические элементы "И" способны реализовать любой минтерм СДНФ, Логические элементы "ИЛИ" осуществляют суммирование термов, требующихся по логическому выражению СДНФ.

В схеме ПЛМ, приведенной на рисунке, ранг терма ограничен количеством входов и равен четырем, количество термов тоже равно четырем.

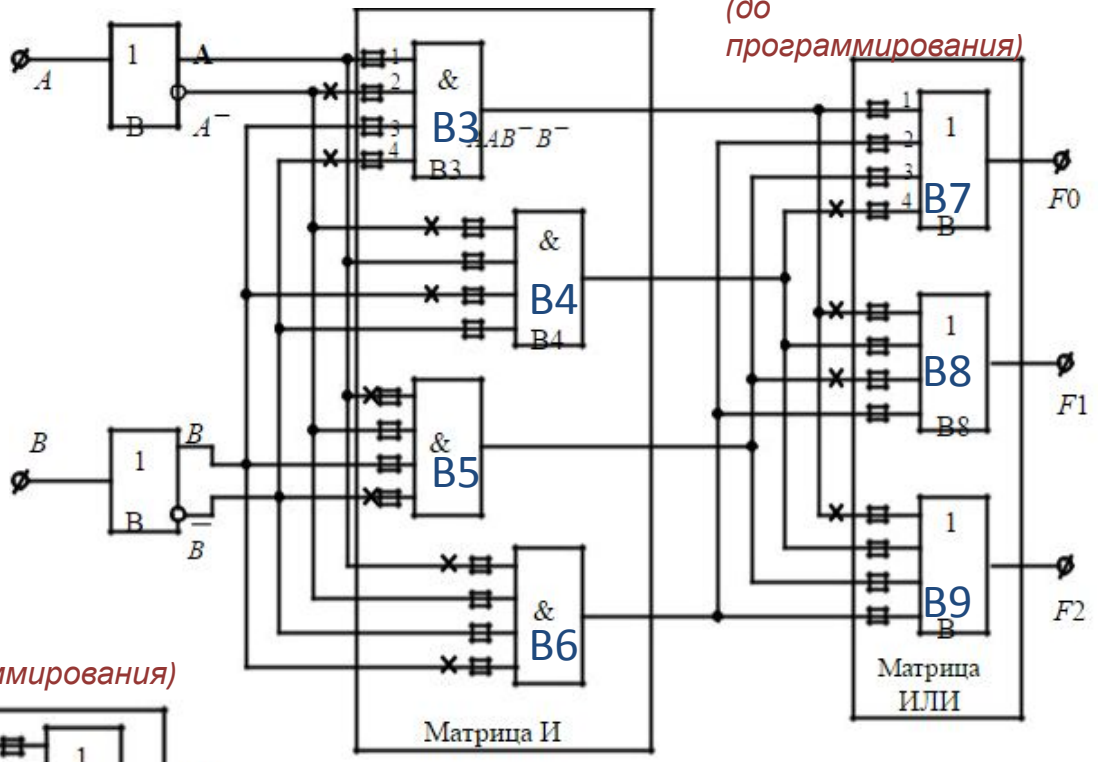
В выпускавшихся микросхемах ПЛМ количество входов было 16 (максимальный ранг минтерма 16), количество термов 32 и



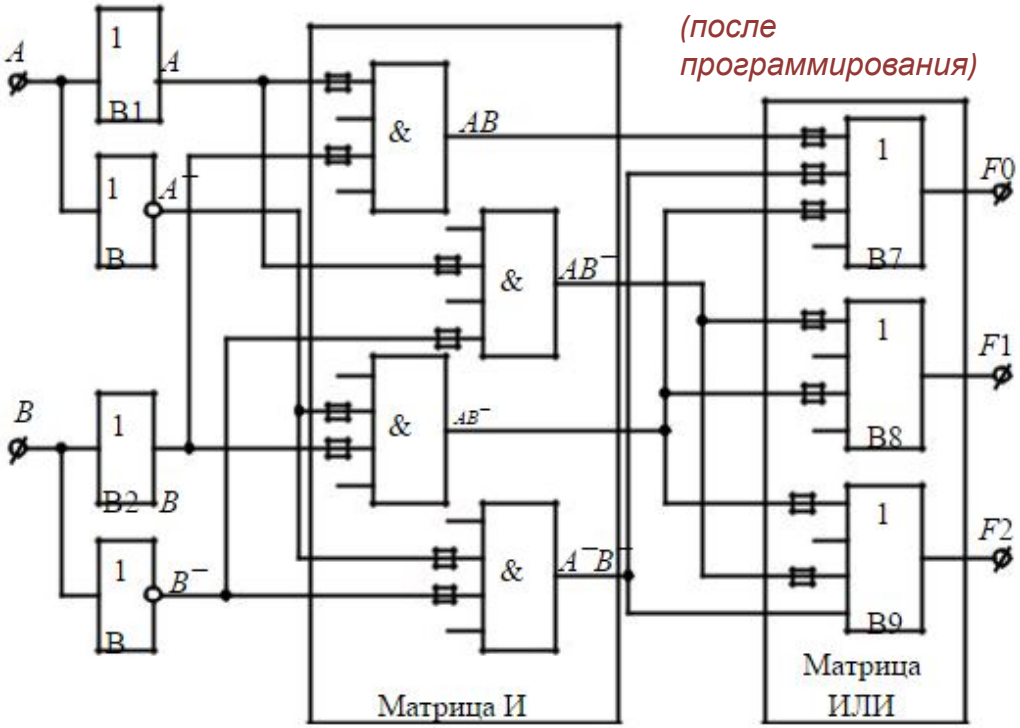
ПЛМ на плавких перемычках

Каждый из вх. сигналов (A,B) и их инверсий соединяется с одним из входов всех схем И через плавкие перемычки (ПП) . Выходы каждого из конъюнкторов (B3-B6) соединяются с входами всех схем ИЛИ (B7-B9) через ПП. Без пережигания ПП на всех выходах И образуются одинаковые логические произведения (термы) A ~A B ~B , а на всех выходах ИЛИ – одинаковые функции Fi из 4-х одинаковых термов . К тому же этот терм равен нулю (A ~A = 0, B ~B = 0). Программирование такой ПЛМ производится пережиганием тех перемычек, которые окажутся ненужными для выполнения заданных функций Fi. В результате программирования часть ЛЭ может быть исключена.

(до программирования)



(после программирования)



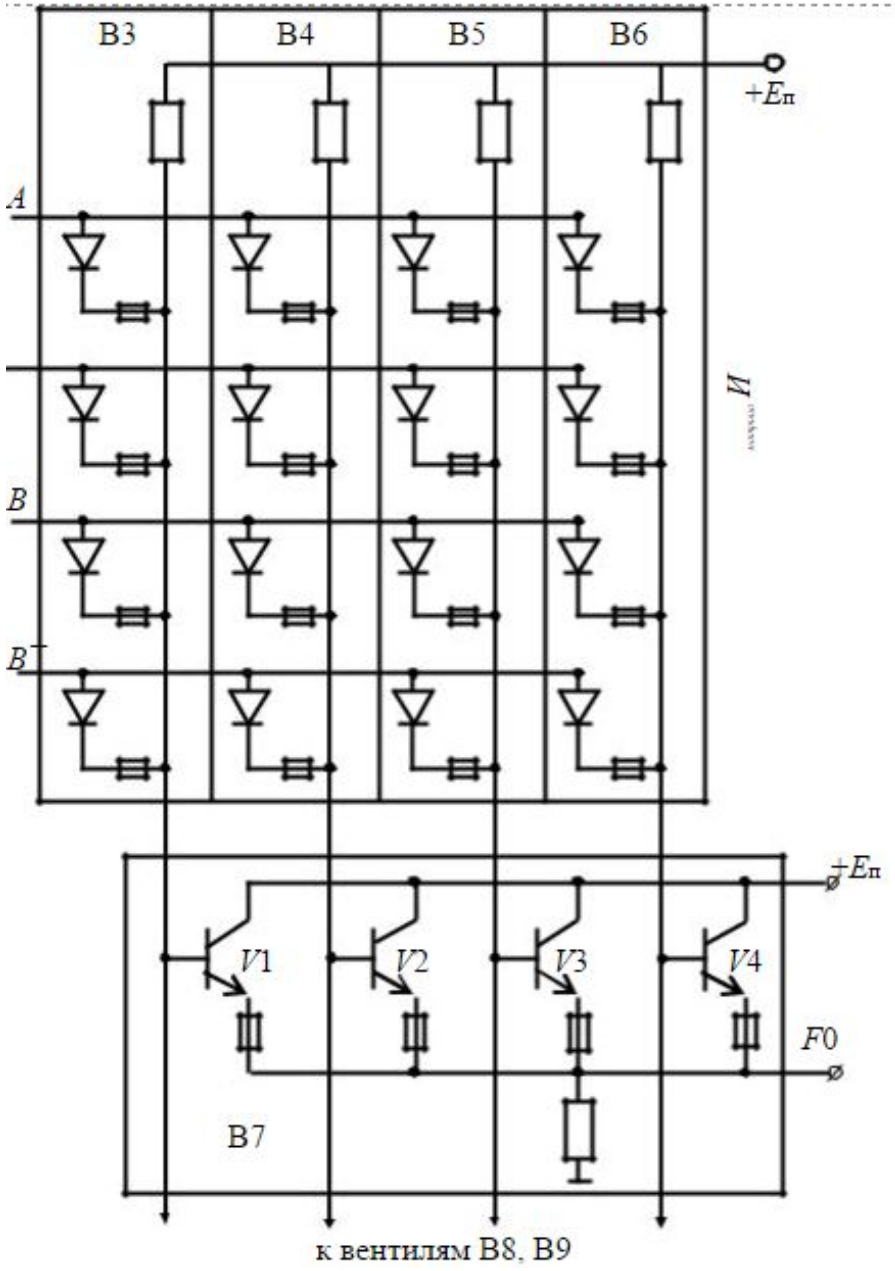
Например, при программировании пережжены перемычки, которые указаны значком **x** (вх. 2, 4 вентиля B3 и др.). Разорванные перегоревшими ПП соединения на схеме не показаны, от них остались свободные выводы. На выходах конъюнкторов приведены образуемые ими термы AB,A~B ,~AB, ~A~B. На выходах ПЛМ (выходы вентилей B7, B8, B9) образуются логические функции:

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 выходных функций (Fi) и входных сигналов (A,B,C, …) достигает десятков, а число термов (конъюнктивных членов) – сотен.

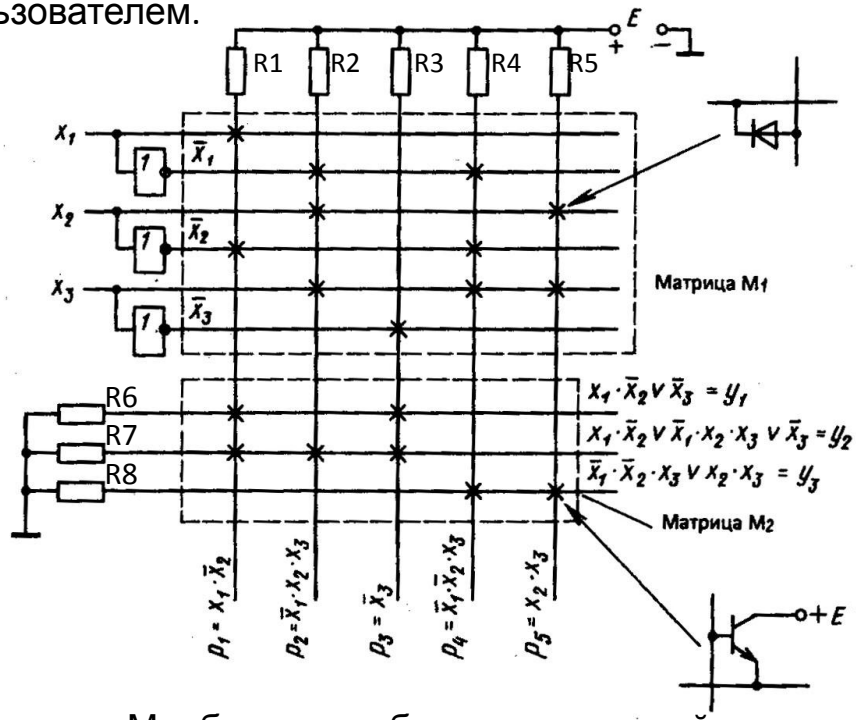
Разновидности ПЛМ:

1. Обе матрицы могут быть выполнены на однотипных ЛЭ, например на базовых ТТЛ элементах И-НЕ. Тогда вентили второй ступени (В7, …, В9) 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

Программируемые логические матрицы

Программируемая логическая матрица (ПЛМ) – это универсальная структура, позволяющая запрограммировать систему булевых функций путем организации связи между вертикальными и горизонтальными шинами. Набор этой связи программируется, в результате программирования логической матрицы и может реализовать заданную систему выражений. Такая матрица может быть настроена на выполнение любой логической функции определенной сложности. ПЛМ может осуществляться на заводе в процессе изготовления микросхемы на этапе формирования элементов в узлах матриц. Программирование может выполнять пользователем.

Матрица M_1 реализует необходимые конъюнкции, причем, если необходимо установить связь в матрице М1, на пересечении устанавливается диод, позволяющий осуществлять гальваническую связь между горизонтальной шиной, имеющей соответствующую переменную, и вертикальной шиной, имеющей необходимую конъюнкцию. 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 и появление потенциалов.



При программировании **X** указывается наличие связей в матрице M_2 ,т.е. «**X** » на схеме означают, что в этих местах матрицы имеются транзисторы. При срабатывании транзистора, соответственно на R6, R7 или R8 появляется *лог. 1* и становится истинной соответствующие функции y_1-y_3.

Учитывая, что любая булевая функция может быть представлена в СДНФ, которая затем может быть минимизирована, программирование логической матрицы позволяет осуществить построение матрицы любой комбинированной схемы, которая отсутствует в памяти. Схема работает только при наличии входных сигналов и не запоминает предыдущее состояние.

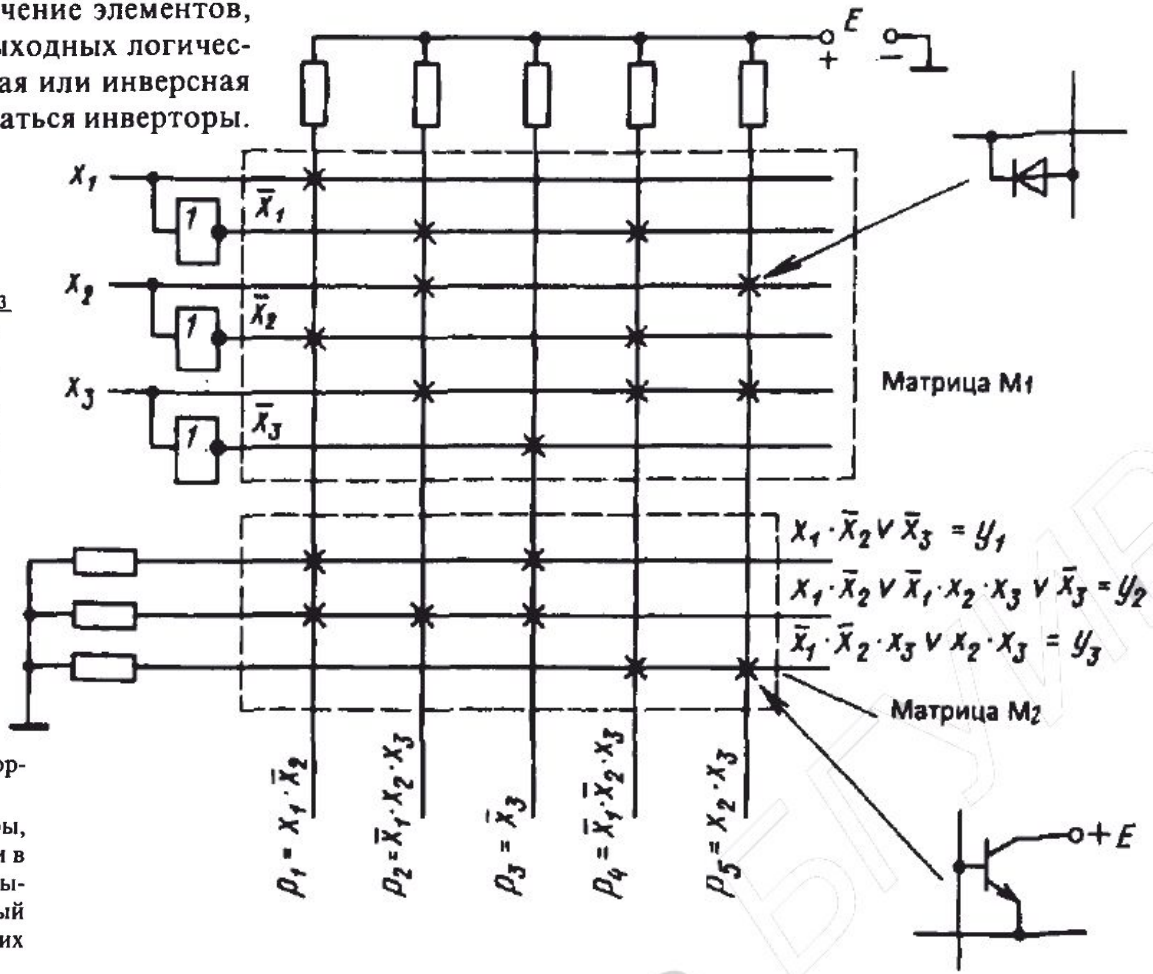
Матрица M_2 обладает собирательным свойством и программируется для организации необходимой ДНФ. На пересечении матрицы устанавливаются программируемые тсятранзисторы p-n типа. В цепи эмиттера этого транзистора имеется сопротивление, которое при программировании прожигается. В результате связь между горизонтальной и вертикальной

Программируемые логические матрицы

Цепи входных переменных x_1, x_2, ... и их инверсий $\bar{x}_1$, $\bar{x}_2$, ... составляют горизонтальные цепи матрицы M_1, вертикальными цепями которой служат так называемые *цепи конъюнкции*. Другую матрицу M_2 образуют цепи конъюнкции с горизонтальными цепями выходов y_1, y_2, ... В узлах матрицы M_1 включены элементы, с помощью которых на цепях конъюнкции могут формироваться любые требуемые конъюнкции входных переменных, имеющиеся в узлах матрицы M_2 элементы позволяют формировать на выходных цепях любые требуемые дизъюнкции функций, полученных на цепях конъюнкций. В процессе программирования ПЛМ в узлах матриц M_1 и M_2 производят подключение элементов, которые необходимы для реализации требуемых выходных логических функций y_1, y_2, ... В зависимости от того, прямая или инверсная функция реализуется, в выходные цепи могут включаться инверторы.

Матрица M_1 содержит горизонтальные цепи, на которых действуют входные переменные x_1, x_2, ... и их инверсии $\bar{x}_1$, $\bar{x}_2$, ..., и вертикальные цепи, на которых формируются конъюнкции p_1, p_2, ... В отдельных узлах матрицы между ее вертикальными и горизонтальными цепями включены диоды. На вертикальной цепи образуется высокий потенциал (уровень *лог.1*) в том случае, когда на всех входах, идущих к узлам, содержащим диоды, действует высокий потенциал (уровень *лог.1*), закрывающий диоды. Если хотя бы на одном из таких входов низкий потенциал (уровень *лог.0*), открывается диод и уровень *лог.0* с этого входа через открытый диод передается на вертикальную цепь матрицы.

	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



Включая в соответствующие узлы диоды, можно на выводах p_i сформировать любые конъюнкции входных переменных и их инверсий.

В узлах матрицы M_2 между цепями p_i и y_j включены транзисторы, базы которых подключены к цепям p_i, а эмиттеры — к цепям y_j. Если в цепи p_i действует высокий потенциал (уровень *лог.1*), транзистор оказывается в открытом состоянии и высокий потенциал через открытый транзистор передается в цепь y_j и $y_j = 1$ независимо от уровней на других выходах матрицы M_1.

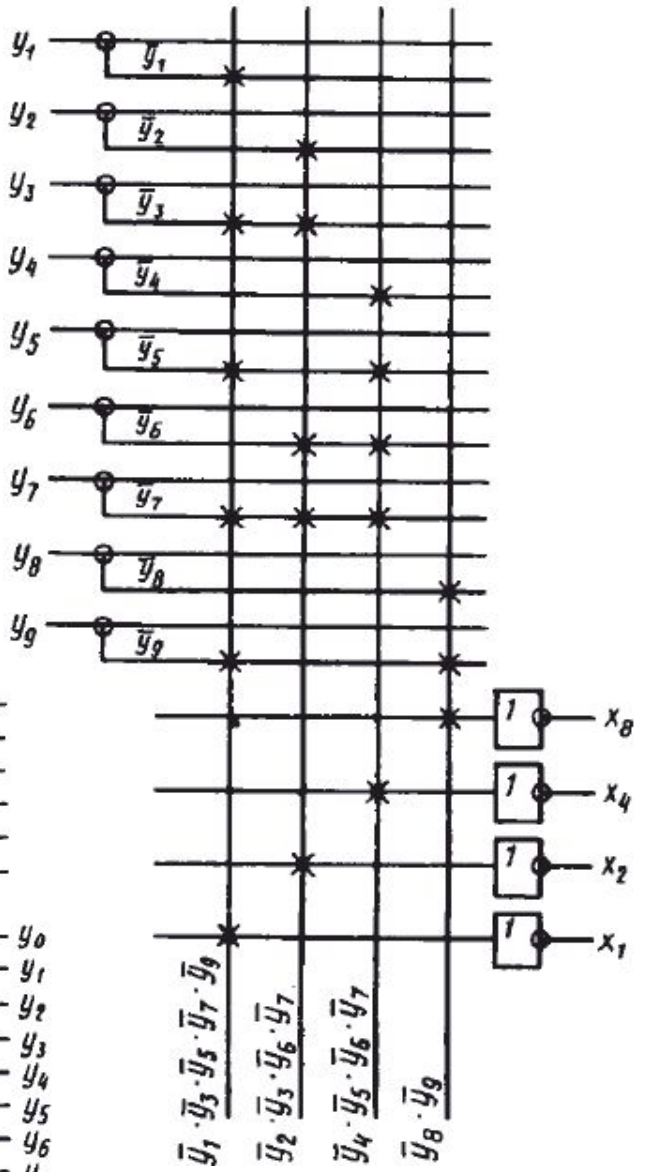
ПЛМ. Пример реализации шифратора/дешифратора

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).

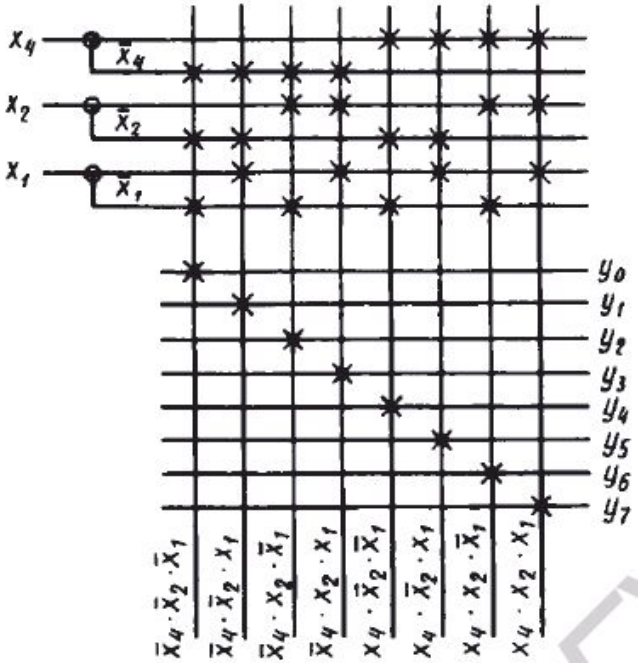
Функционирование такого дешифратора определяется следующими логическими выражениями:

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

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

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

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



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.

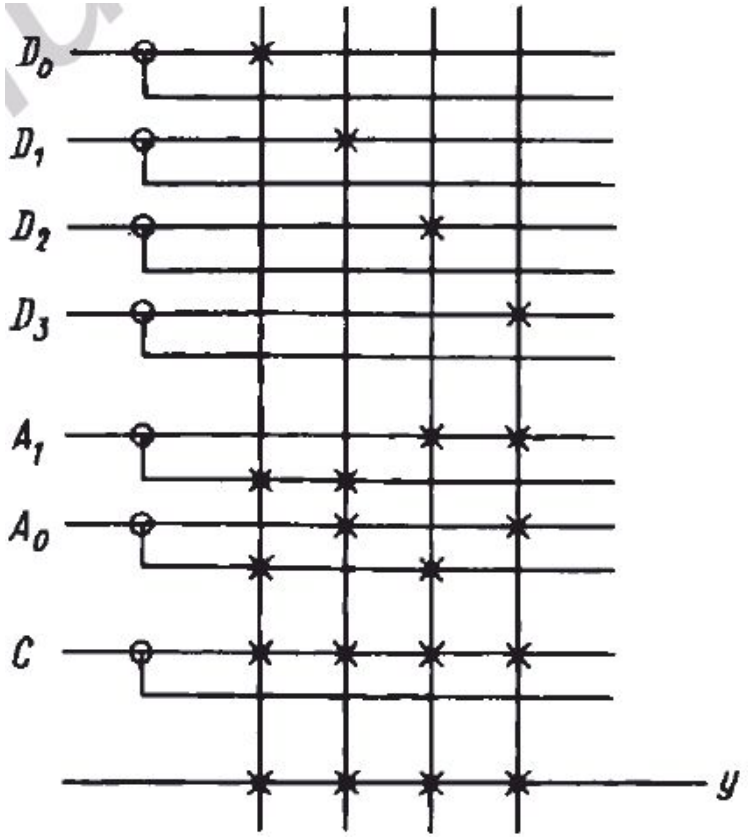
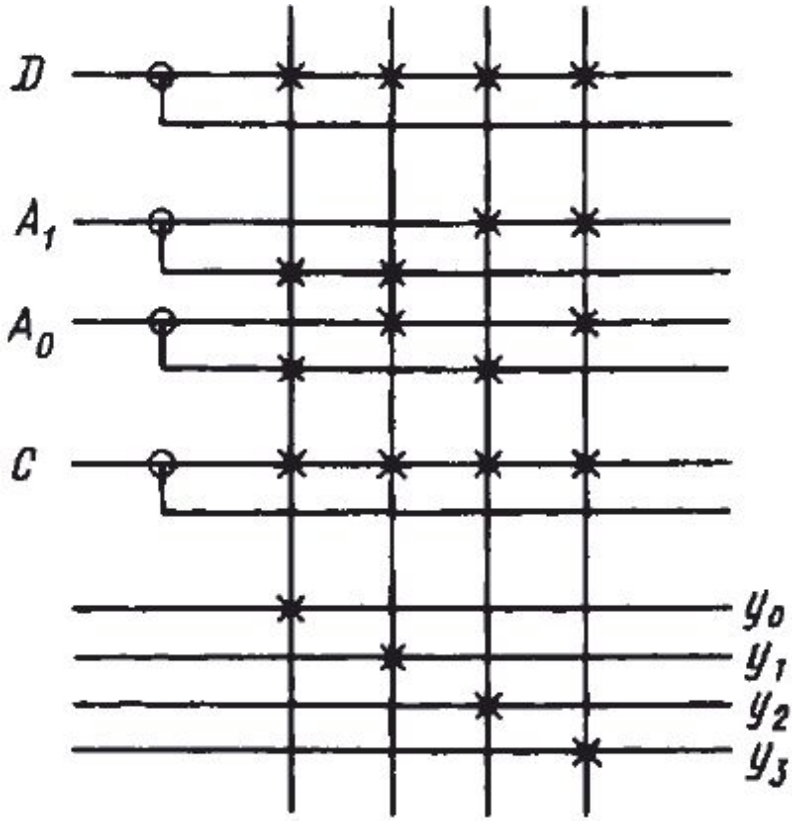


Схема де

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;

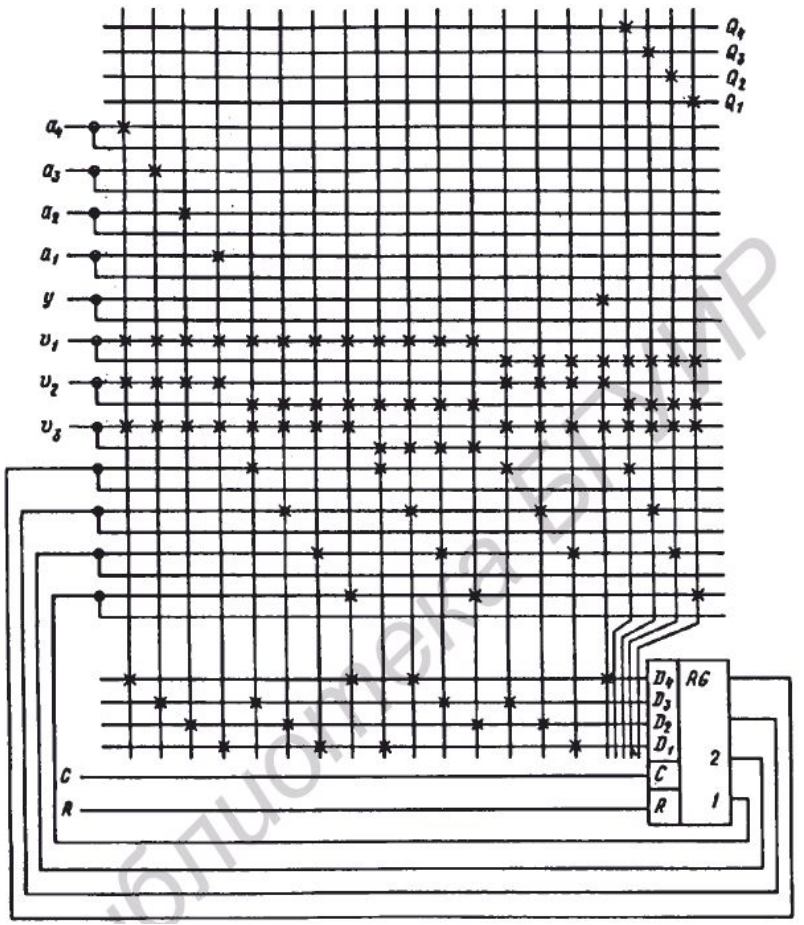
циклический сдвиг содержимого регистра влево или вправо (сдвиг с передачей выдвигаемой из регистра цифры в освобождающий при сдвиге разряд регистра);

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

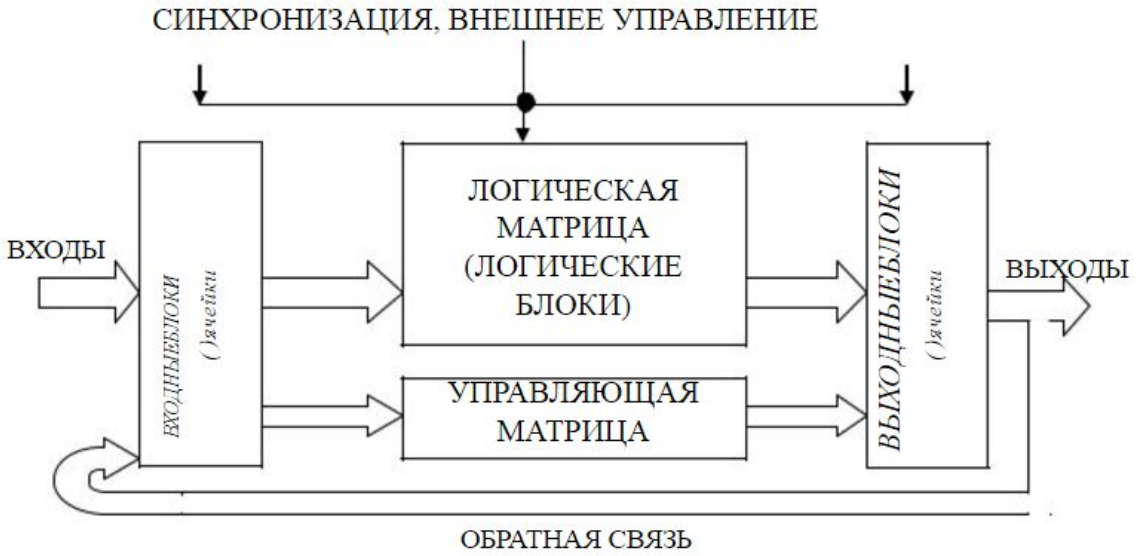


Пусть вид выполняемой в регистре функции задается комбинацией v_1, v_2, v_3 в соответствии с табл. Число выполняемых в регистре функций ограничим пятью

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

Основу всех ПЛИС составляет логическая матрица. Она может быть полной (из элементов И и ИЛИ), матрицей из однотипных элементов (И-НЕ,ИЛИ-НЕ и др.) или с фиксированными элементами ИЛИ. Часто логическая матрица разбивается на логические блоки с элементами запоминания.



«Программируемые вентильные матрицы» фирмы Xilinx (XC2018, XC2064, …, XC3090) имеют особенности: 1.Логическая матрица построена на базе КМОП-элементов, состоит из до сотни и более логических блоков (ЛБ) с динамически изменяемой конфигурацией, построенных на основе ячеек статических ЗУ с произвольной выборкой. 3.Каждый ЛБ имеет 4 – 5 логических входов, два выхода и тактовый вход (синхровход). Такой ЛБ может реализовать любую логическую функцию 4-5 логических переменных.

4.Каждый ЛБ может воспринимать на входе и формировать на выходе сигналы как положительной, так и отрицательной логики. 5.Вокруг логической матрицы расположено несколько десятков двунаправленных

БВВ тоже с изменяемой конфигурацией. 6. Каждый БВВ содержит входной регистр, схему настройки входного порогового напряжения и выходную схему с тремя состояниями (0,1, z). ЛБ и БВВ соединяются между собой. Для обеспечения дополнительных возможностей в изменении конфигурации ЛБ и БВВ используются разнообразные соединительные элементы, программируемые потребителем: а) горизонтальные и вертикальные металлизированные линии, проложенные между ЛБ и БВВ и разделённые на сегменты; б) элементы обмена на координатных переключателях, соединяющие между собой отдельные сегменты металлизированных линий;

в) программируемые соединительные точки, связывающие металлизированные линии с логическими блоками и блоками ввода-вывода.

Во вх. блоках имеются формирователи парафазных сигналов (A,~A , …) и элементы запоминания (триггеры, регистры). Вых. ячейки могут быть многофункциональными и выполнять функции от простого линейного до управляемого выхода с управлением полярностью выхода (при возможности выбора активно-низкого (отрицательная логика) или активно-высокого(положительная логика) выходного сигнала), с запоминанием сумм произведений, с разрешением (запретом) выхода, с внутренней синхронизацией и т.д. Выбор функции многофункциональных выходных ячеек (блоков) осуществляет управляющая матрица. Под действием обратной связи (ОС) выходные блоки могут быть преобразованы в двунаправленные БВВ. Синхронные узлы имеют входы синхронизации. Сигналы внешнего управления могут выполнять такие функции, как предварительная установка, сброс или загрузка регистров, разрешение (запрет) выходов и др. В составе некоторых высокоинтегрированных ПЛИС имеются макроячейки высокого уровня сложности – счётчики, мультиплексоры, дешифраторы, АЛУ и ЗУ.

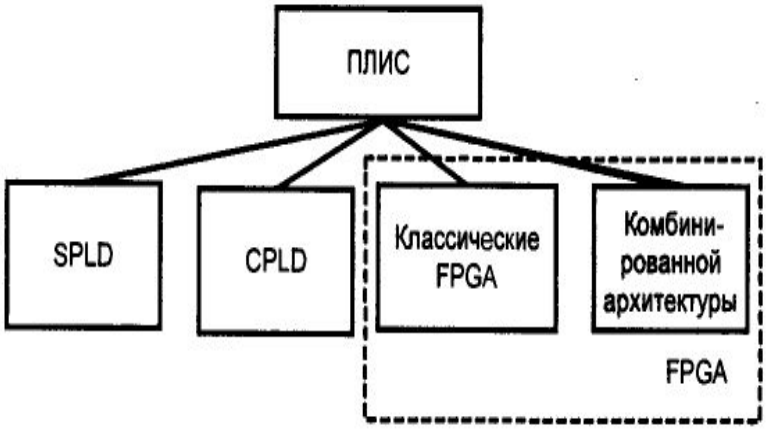
Классификация ПЛИС

- 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



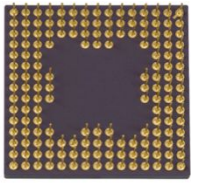
В сложных программируемых логических схемах CPLD (Complex Programmable Logic Devices) несколько блоков, подобных ПМЛ, объединяются средствами программируемой коммутационной матрицы. В CPLD могут входить сотни блоков и десятки тысяч эквивалентных вентилей. Архитектуры CPLD разрабатываются фирмами Altera, 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).

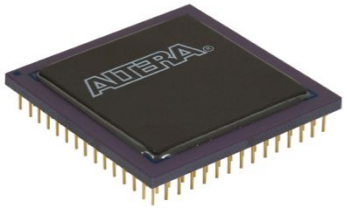
Оба эти подкласса микросхем реализуют дизъюнктивные нормальные формы (ДНФ) переключательных функций, а их основными блоками являются две матрицы: матрица элементов И и матрица элементов ИЛИ, включенные последовательно. Такова структурная модель ПЛМ и ПМЛ. Технически они могут быть выполнены и как последовательность 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.

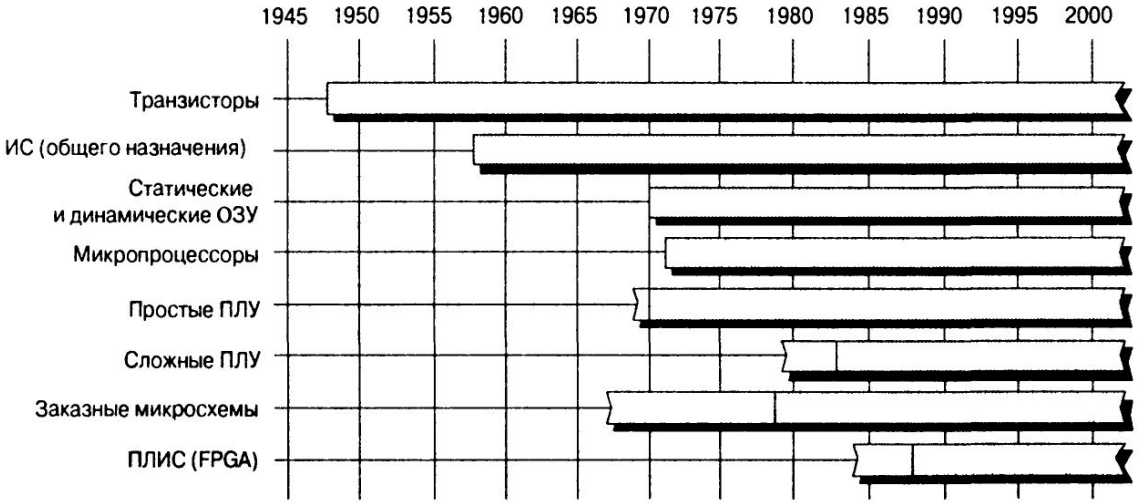
Свойства и преимущества ПЛИС



- 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



XILINX
UltraSCALE

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.

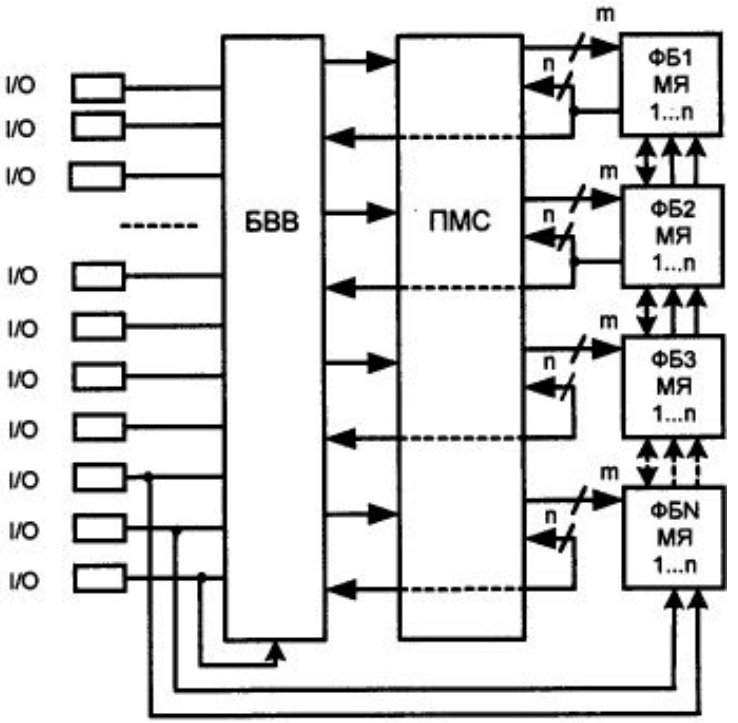
Возможности HDL в области отображения основных характеристик (структурный, временной, функциональный аспект) аппаратуры

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».

CPLD – сложные программируемые логические устройства

CPLD — микросхемы высокого уровня интеграции, основными частями которых являются:

- PAL (GAL) — подобные функциональные блоки;
- система коммутации, позволяющая объединять функциональные блоки в единое устройство, выполненная в виде матрицы соединений.
- блоки ввода/вывода.



В структурной схеме приняты следующие обозначения. Через ФБ (FB) обозначены функциональные блоки, число которых N зависит от уровня интеграции микросхемы и изменяется в довольно широких пределах. В каждом ФБ имеется *n макроячеек* МЯ (MC, Macrocells). Функциональные блоки получают входные сигналы от программируемой *матрицы соединений* ПМС (PIA, Programmable Inerconnect Array). Число таких сигналов *m*. Выходные сигналы ФБ поступают как в ПМС, так и в *блоки ввода/вывода* CPLD (IOBs, Input/Output Blocks, БВВ). ПМС обеспечивает полную коммутируемость функциональных блоков, т. е. возможность подавать сигналы с любого их выхода на любой вход.

Блоки ввода/вывода связаны с внешними двунаправленными выводами I/O, которые, в зависимости от программирования, могут быть использованы как входы или как выходы. Три нижних вывода либо специализируются для подачи на матрицу функциональных блоков сигналов GCK (Global Clocks) глобального тактирования, сигналов GSR (Global Set/Reset) глобальной установки/сброса и сигналов GTS (Global 3-state Control) глобального управления третьим состоянием выходных буферов, либо эти же выводы могут быть использованы для операций ввода/вывода. Здесь и далее термин *"глобальный"* применяется для сигналов, общих для всей микросхемы.

Блоки ввода/вывода CPLD

Блоки ввода/вывода соединяют внешние контакты микросхемы с ее внутренними цепями. Характерным примером такого блока может служить БВВ

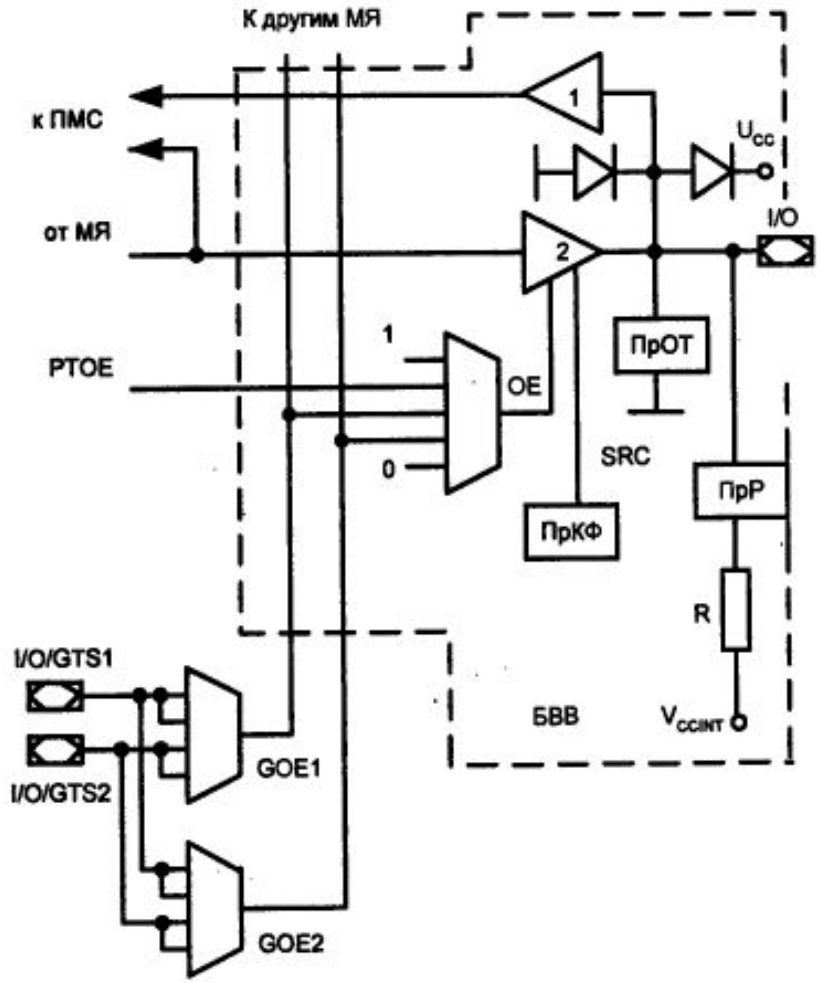


Схема программируемой общей точки ПрОТ позволяет пользователю при необходимости получать дополнительный "заземленный" вывод. Дополнительные выводы для системы "заземления" повышают ее качество и тем самым снижают уровень помех в микросхеме.

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 V_{CCINT} — R. Резистор R используется и в некоторых других режимах, а в рабочих режимах отключается.

Выходной буфер 2 получает сигналы разрешения работы OE и управления крутизной фронта выходного напряжения SRC (Slew Rate Control). Сигнал OE с помощью программируемого мультиплексора MUX3 вырабатывается в нескольких вариантах: от терма PTOE, получаемого от макроячейки, от любого из глобальных сигналов управления третьим состоянием (GOE1, GOE2), от константы 1 и от константы 0. Глобальные сигналы управления третьим состоянием образуются с возможностью выбора любой полярности исходных сигналов GTS1 и GTS2.

Выходные буферы конфигурируются для работы с напряжениями питания 5 или 3,3 В при подключении внешнего источника питания с тем или иным уровнем напряжения (эти цифры относятся к рассматриваемому блоку типа XC9500, сейчас у блоков ввода/вывода нередко уровни выходных сигналов могут выбираться из многих возможностей, в том числе из таких низких напряжений, как 2,5 и 1,8 В).

Согласно принятой нами классификации в класс CPLD попадают ПЛИС с уровнем интеграции 600—20 000 эквивалентных вентилей, числом макроячеек 32—512, числом функциональных блоков 2—16 и временем распространения сигнала от любого входа до любого выхода 5—20 нс. Эти CPLD

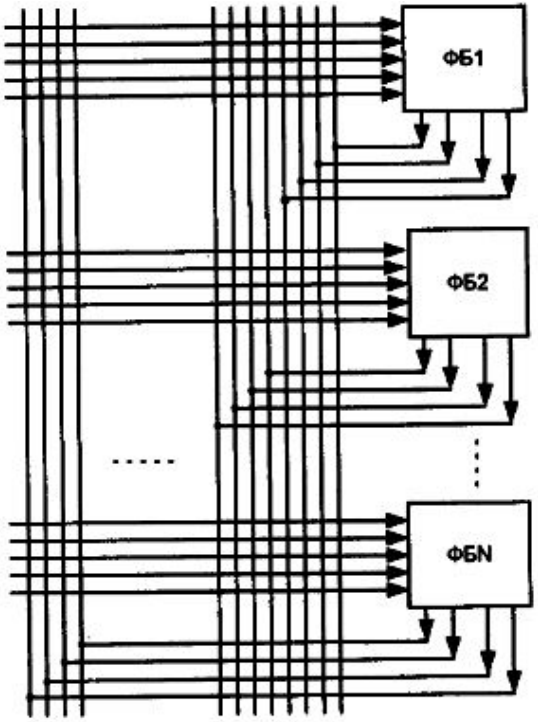
Основой БВВ служат два буфера — входной (1) и выходной (2). Чтобы обеспечить постоянство уровней напряжения, поступающих на входной буфер, и их независимость от амплитуды входных сигналов, в схеме вырабатывается внутреннее напряжение питания V_{CCINT} и вводится цепь из двух фиксирующих диодов.

Программируемая матрица соединений

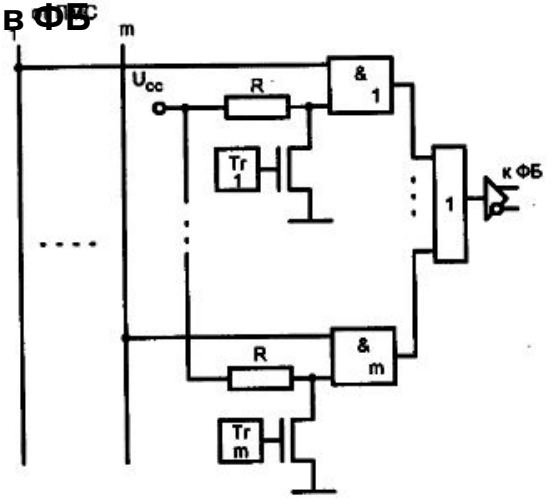
В программируемой матрице соединений ПМС (рис. 1.5, *а*) выходы функциональных блоков ФБ подключаются к вертикальным непрерывным (не сегментированным) линиям, причем каждому выходу соответствует своя линия. Входы ФБ связаны с горизонтальными линиями, пересекающими все вертикальные линии. На пересечениях горизонтальных и вертикальных линий имеются программируемые точки связи, так что любой вход ФБ может быть подключен к любому выходу, чем обеспечивается так называемая полная коммутируемость блоков.

Достоинством ПМС рассмотренного типа является *малая и предсказуемая задержка* коммутируемых сигналов, т. к. для каждого соединения образуется идентичный всем другим канал связи с малым числом программируемых ключей или даже их отсутствием, если передача сигналов из ПМС в ФБ организована так, как показано на рис. 1.5, *б*. В этом случае программируемых ключей в цепи передачи сигнала нет, программируются только напряжения на нижних входах конъюнкторов, и ФБ получит сигнал от *i*-й вертикальной линии ПМС ($i = 1, 2, ..., m$), если транзистор T_i будет заперт, и на нижнем входе *i*-го конъюнктора будет действовать высокий потенциал логической единицы. Открытый транзистор T_i подключает нижний вход конъюнктора к нулевому потенциалу, создавая на нем и на выходе конъюнктора сигнал логического нуля. Таким образом, задавая триггеру T_i состояние логического нуля, а остальным триггерам состояние логической единицы, можно обеспечить закрытое состояние транзистора T_i и открытое состояние всех других транзисторов, что означает подключение выхода ФБ к *i*-й вертикальной линии ПМС с образованием так называемого непрерывного соединения.

Схема программируемой матрицы соединений CPLD

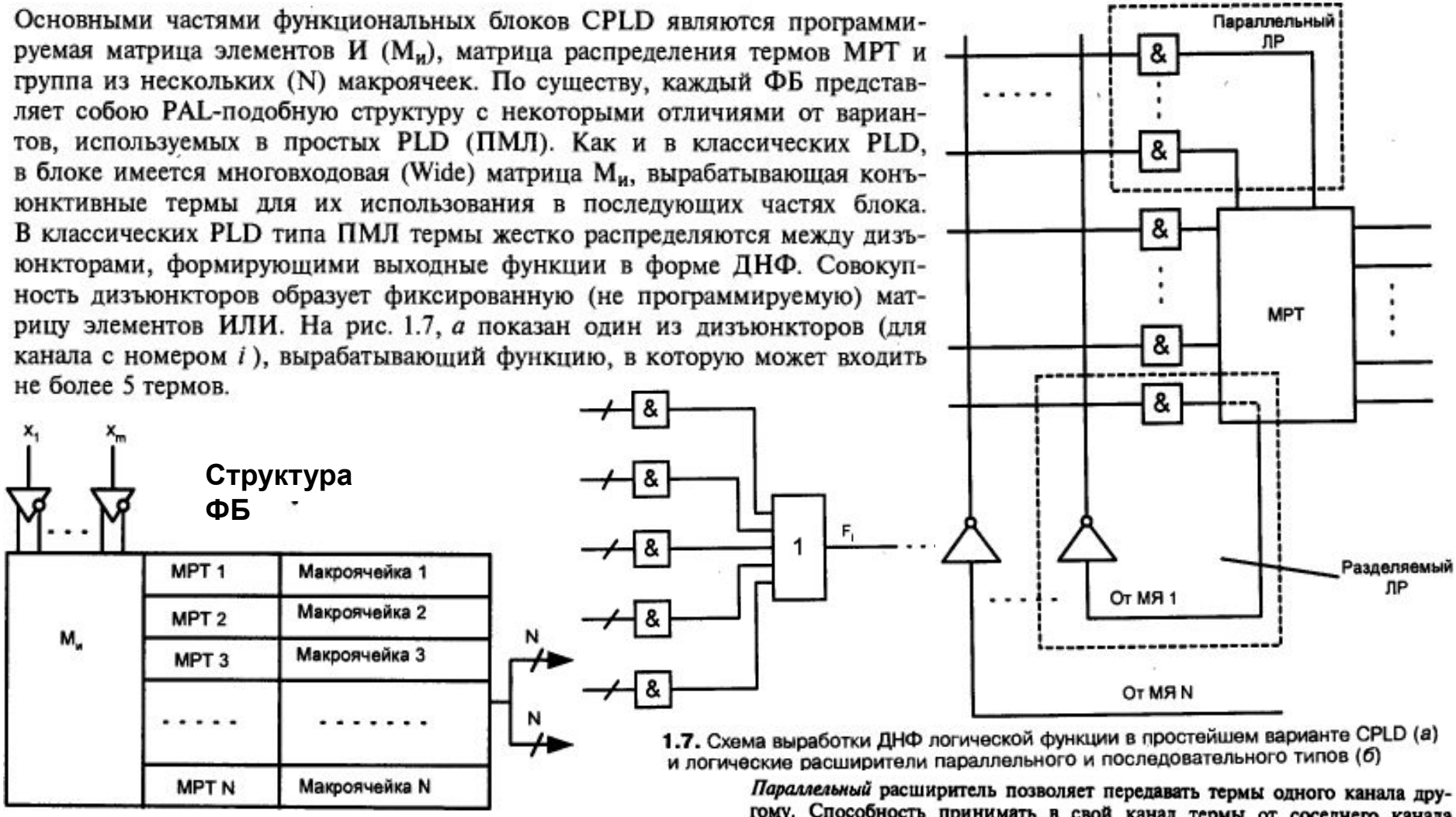


Передача сигналов от ПМС в ФБ



Функциональные блоки CPLD

Основными частями функциональных блоков CPLD являются программируемая матрица элементов И ($M_и$), матрица распределения термов МРТ и группа из нескольких (N) макроячеек. По существу, каждый ФБ представляет собою PAL-подобную структуру с некоторыми отличиями от вариантов, используемых в простых PLD (ПМЛ). Как и в классических PLD, в блоке имеется многовходовая (Wide) матрица $M_и$, вырабатывающая конъюнктивные термы для их использования в последующих частях блока. В классических PLD типа ПМЛ термы жестко распределяются между дизъюнкторами, формирующими выходные функции в форме ДНФ. Совокупность дизъюнкторов образует фиксированную (не программируемую) матрицу элементов ИЛИ. На рис. 1.7, *а* показан один из дизъюнкторов (для канала с номером *i*), вырабатывающий функцию, в которую может входить не более 5 термов.



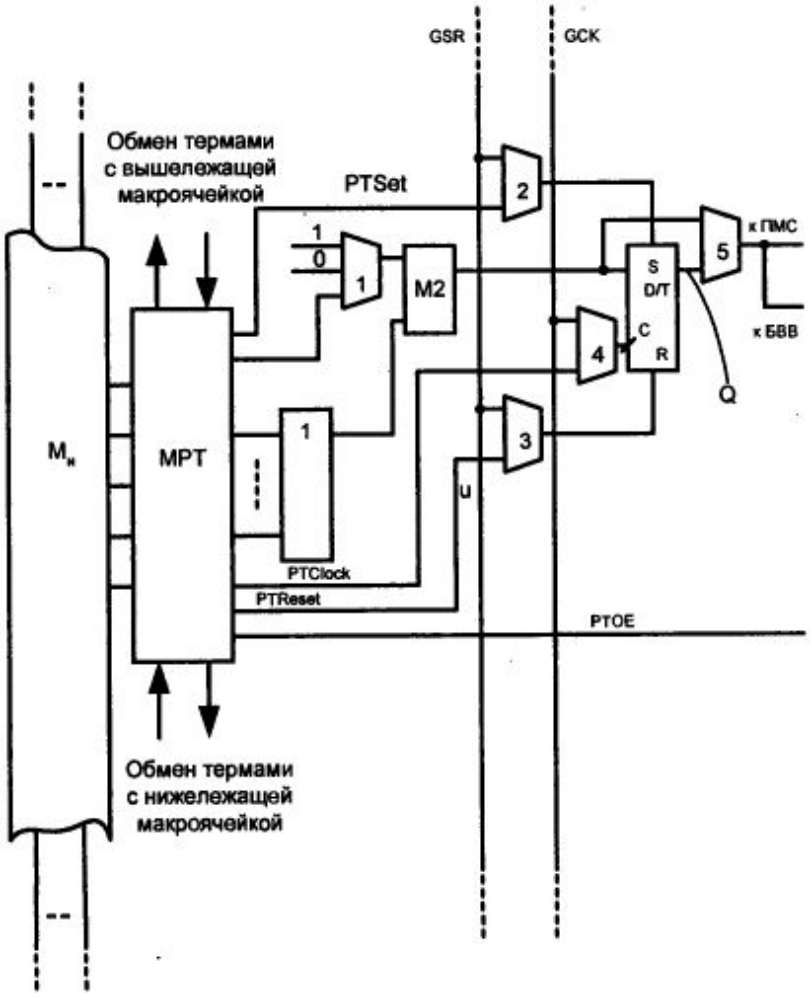
1.7. Схема выработки ДНФ логической функции в простейшем варианте CPLD (*а*) и логические расширители параллельного и последовательного типов (*б*)

Схемотехнически в операциях распределения термов по каналам ФБ задействованы как непосредственно цепи коммутации между входами и выходами МРТ, так и логические расширители последовательного и параллельного типов. *Последовательные (разделяемые, общие)* логические расширители создаются подачей инвертированного значения терма из МРТ данного канала обратно на один из входов матрицы $M_и$ (рис. 1.7, *б*). Переданный в матрицу $M_и$ терм становится доступным для использования во всех каналах данного

Параллельный расширитель позволяет передавать термы одного канала другому. Способность принимать в свой канал термы от соседнего канала обычно означает и возможность приема через него термов и более далеких каналов с образованием цепочки для сбора термов от нескольких каналов (например, в пределах целого функционального блока). Можно, естественно, и отдавать собственные термы или их часть другим каналам (в частности, соседним, а через них и более далеким).

Термы от МРТ поступают далее на часть ФБ, называемую *макроячейкой* (МЯ). Макроячейка содержит в качестве основы программируемые мультиплексоры, триггер (или триггеры) и формирует группу выходных сигналов ФБ в нескольких их вариантах.

Макроячейка ФБ CPLD



Пример схемы макроячейки функционального блока CPLD

Мультиплексор MUX5 программируется для передачи на выход МЯ либо непосредственно значения функции *F* (комбинационный выход), либо состояния триггера (регистровый выход). Характер тактирования триггера определяется программированием мультиплексора MUX4, при этом возможно использование глобального синхросигнала (GCK, Global Clock) или сигнала, порождаемого термом PTClock. Асинхронные установка и сброс триггера производятся либо глобальным сигналом (GSR, Global Set/Reset), либо термами PTSet и PTReset, что определяется программированием мультиплексоров MUX2 и MUX3. Сам триггер программируется на режимы триггера задержки (типа D) или счетного (типа T).

Термин "макроячейка" трактуется в разных источниках по-разному. Наряду с указанным определением встречаются и такие, согласно которым макроячейкой называют часть ФБ, содержащую все его последовательно включенные части (группу элементов И, МРТ и все последующие элементы, относящиеся к данному каналу выработки выходных функций).

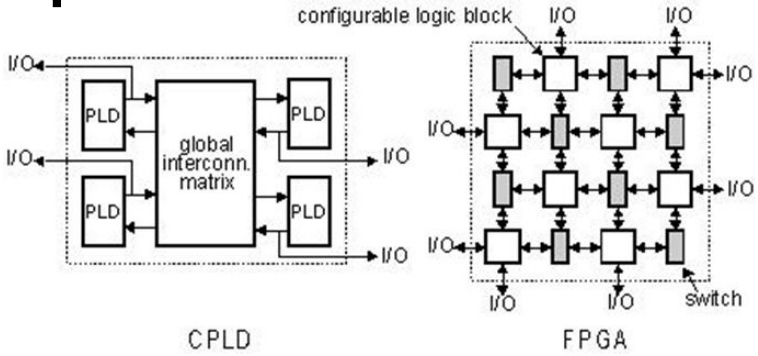
Программируемость мультиплексоров в этой схеме и всех дальнейших не отображается, т. к. она присуща всем имеющимся в схеме мультиплексорам, если не оговорено противоположное. В зависимости от программирования каждый мультиплексор передает на выход сигнал с того или иного входа. Триггер может программироваться на режимы работы триггера типа D или T. Заметим, кстати, что при описании микросхем программируемой логики триггеры в иностранной литературе чаще всего называют *регистрами*. Триггеры тактируются положительными фронтами синхросигналов и имеют входы установки S и сброса R. Выходные сигналы ФБ передаются в ПМС и в блоки ввода/вывода ББВ.

Аргументы *x*1, ..., *x*M реализуемой макроячейкой функции поступают на матрицу $M_н$ из ПМС. Аргументами для МЯ могут быть как входные сигналы, поступающие извне через ББВ, так и сигналы обратных связей, подаваемые в матрицу И с выходов макроячеек. На рис. 1.8 входные сигналы матрицы независимо от их характера обозначены через *x*1, ..., *x*M. Входные буферы преобразуют сигналы в парафазные, представляя каждый сигнал его прямым и инверсным значениями, так что в матрице имеется 2*m* вертикальных линий и образующие ее конъюнкторы имеют по 2*m* входов. Пять термов из матрицы И поступают на элемент ИЛИ для образования логической функции. Для управления триггером и буферами блока ввода/вывода 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 ББВ (OE, Output Enable). Всего в матрице И на рис. 1.8 программируются 9N термов.

На выходе элемента ИЛИ вырабатывается логическая 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 мультиплексора 1, может быть подан логический нуль, логическая единица или терм PT1. В первом случае функция передается без изменений ($F = F^*$), во втором инвертируется ($F = !F^*$), в третьем передается в прямом виде во всех ситуациях за исключением такой, в которой PT1 = 1.

Программируемые пользователем вентильные матрицы FPGA

В наиболее типичном варианте FPGA представляет собою микросхему высокого уровня интеграции, содержащую во внутренней области матрицу идентичных функциональных блоков и систему их межсоединений, размещенную между строками и столбцами матрицы, а в периферийной области — блоки ввода/вывода (рис. 1.10, *а*). Кроме этого варианта существуют FPGA, в которых функциональные блоки расположены по строкам (строковые FPGA), однако рассматривать эти варианты отдельно нет оснований, поскольку существенные черты FPGA остаются одинаковыми для обоих вариантов.



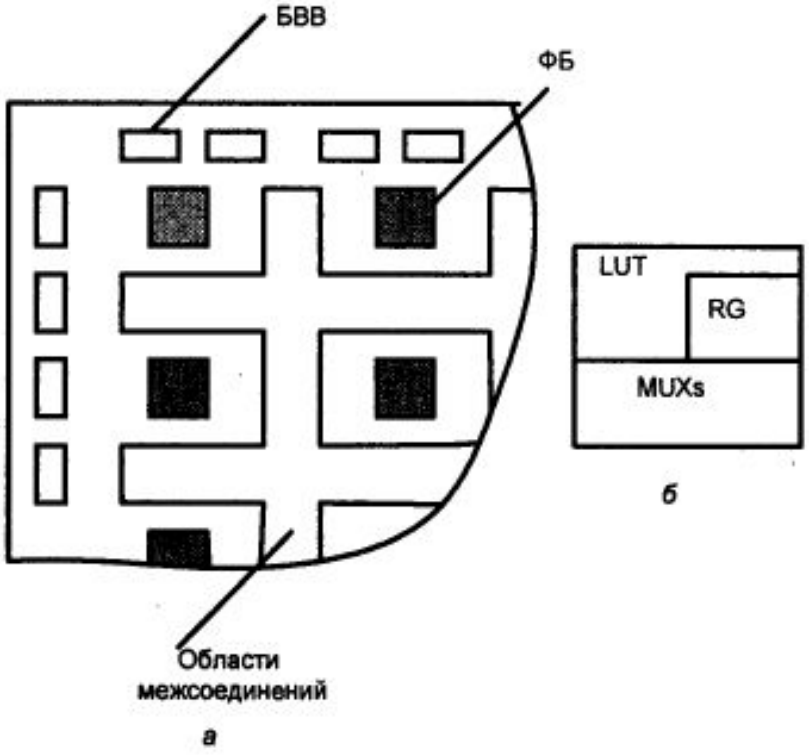


Рис. 1.10. Обобщенная структура FPGA (*а*) и основные части их функциональных блоков (*б*)

Все части FPGA (функциональные блоки ФБ, система межсоединений и блоки ввода/вывода БВВ) являются конфигурируемыми или реконфигурируемыми, причем (в отличие от БМК) средствами самих пользователей.

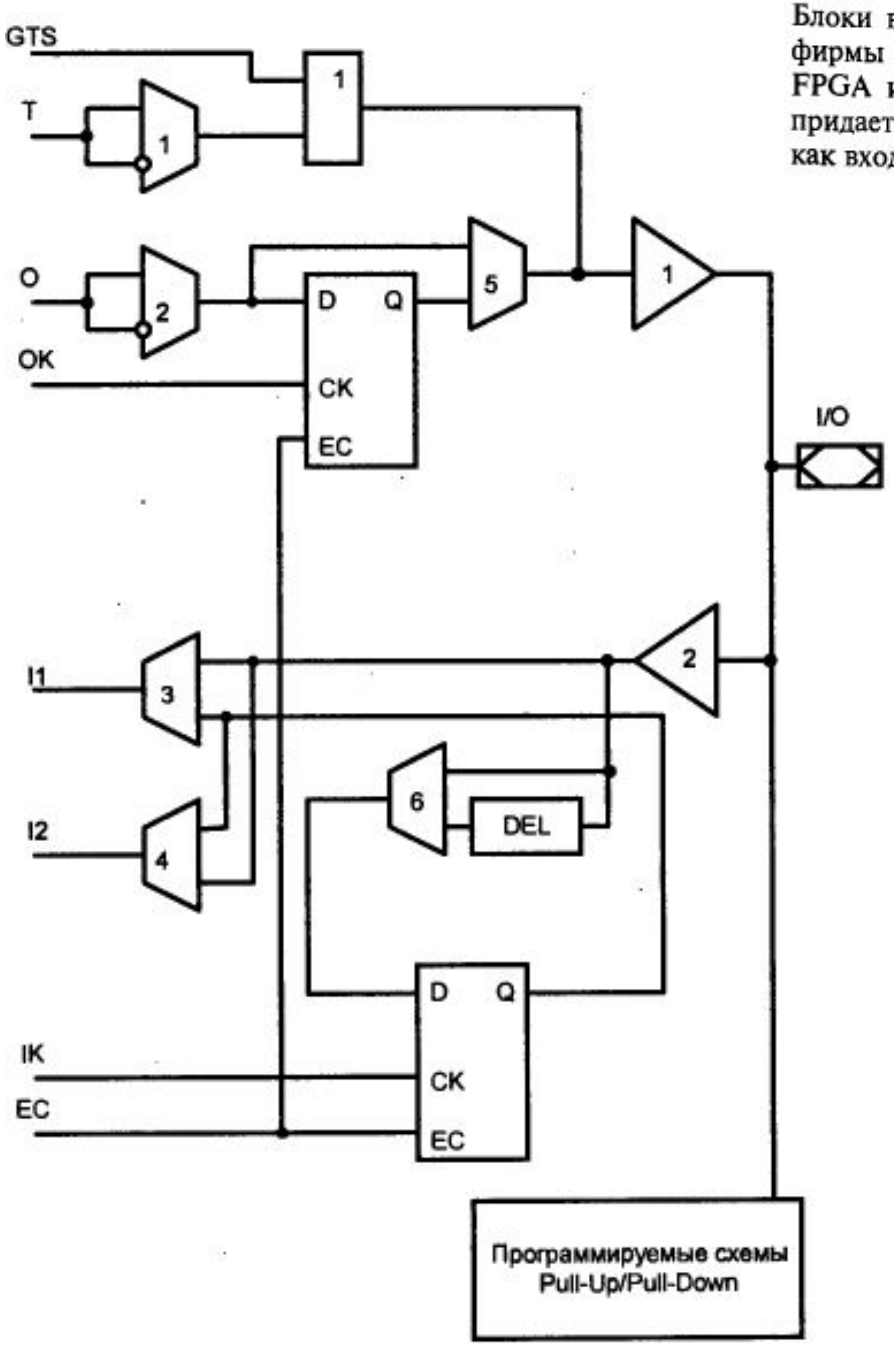
Перечисленные части — основа FPGA. Кроме них современные варианты FPGA, как правило, оснащены дополнительными средствами для автоподстройки задержек в системе тактирования (PLL, Phase Locked Loop или DLL, Delay Locked Loop), средствами поддержки интерфейса JTAG и др.

При конфигурировании FPGA функциональные блоки настраиваются на выполнение необходимых операций преобразования данных, а система межсоединений — на требуемые связи между функциональными блоками. В результате во внутренней области FPGA реализуется схема нужной конфигурации. Расположенные по краям кристалла блоки ввода/вывода обеспечивают интерфейс FPGA с внешней средой. Блоки ввода/вывода современных FPGA можно программировать на выполнение требований множества стандартов передачи данных (число таких стандартов может доходить до 20).

На рис. 1.10, *б* укрупненно показан состав типичного функционального блока ФБ, в который входят функциональный преобразователь ФП, реализованный в виде программируемого запоминающего устройства (LUT, Look-Up Table), триггер (регистр) и мультиплексоры, играющие роль средств конфигурирования ФБ.

LUT — наиболее распространенная разновидность ФП в FPGA со статической памятью конфигурации. В схемах FPGA с однократным программированием перемычек находят применение ФП в виде простых логических вентилей (SLC, Simple Logic Cell) и логических модулей на основе мультиплексоров [27].

Блоки ввода/вывода FPGA



Блоки ввода/вывода, показанные на примере микросхем семейства Spartan фирмы Xilinx (рис. 1.12), обеспечивают интерфейс между выводами корпуса FPGA и ее внутренними логическими схемами. Каждому выводу корпуса придается блок ввода/вывода БВВ, который может быть конфигурирован как вход, выход или двунаправленный вывод.

Работа БВВ как выходного блока обслуживается следующими элементами: выходным буфером 1, триггером 1, мультиплексорами 1, 2, 5 и логической схемой ИЛИ. Выводимый сигнал O можно получать в прямой или инверсной форме в зависимости от программирования мультиплексора 2. Этот сигнал может передаваться на выходной буфер непосредственно или сниматься с триггера при соответствующем программировании мультиплексора 5. Сигналы T и GTS (Global Tri-State), согласно логике ИЛИ, управляют переводом буфера в третье состояние, причем активный уровень сигнала T программируется с помощью мультиплексора 1. Внутренние программируемые цепи триггера (на рисунке не показаны) позволяют изменять полярность тактирующего фронта. Сам буфер имеет программируемые крутизну фронта выходного сигнала и его уровни (КМОП/ТТЛ). Крутизна фронтов в некритичных к скорости передачи цепях снижается для уменьшения уровня помех на шинах питания и земли. Используется так называемый мягкий старт (Soft Start-Up), снижающий помехи при конфигурировании схемы и переходе ее к рабочему режиму, когда одновременно активизируются многие буферы. Первая активизация автоматически происходит с пологими фронтами перепадов напряжения. Затем вступает в силу заданный выбор той или иной крутизны фронтов в зависимости от принятой конфигурации БВВ.

Тракт ввода сигналов содержит входной буфер 2, триггер 2, программируемые мультиплексоры 3, 4, 6, элемент задержки ЭЗ и программируемые схемы задания определенных потенциалов выводу, к которому не подключен вводимый или выводимый сигнал (схемы Pull-Up/Pull-Down). Вводимый сигнал в зависимости от программирования мультиплексоров 3 и 4 или поступает непосредственно в систему коммутации FPGA по входным линиям I1 и I2, или же фиксируется триггером и с его выхода передается в эти линии. Триггеры могут конфигурироваться как тактируемые фронтом или как защелки (D-триггеры, управляемые уровнем). Выбор осуществляется присвоением триггеру соответствующего библиотечного символа. В цепи передачи сигнала на триггер 2 могут быть включены элементы задержки (при передаче сигнала через нижний вход мультиплексора 6). Включение задержки гарантирует необходимые временные соотношения между входными сигналами триггера D и глобальным сигналом тактирования.

Входной буфер может конфигурироваться для восприятия входных сигналов с пороговым значением ТТЛ (1,2 В) или КМОП (0,5 U_{cc}). Выходные уровни тоже конфигурируются, две глобальные регулировки входных порогов и выходных уровней независимы.

Межсоединения FPGA

система межсоединений имеет, как правило, *иерархический характер*, и в ней сочетаются различные типы сегментов (основные связи, связи двойной длины, прямые связи для близлежащих функциональных блоков, длинные линии, пересекающие кристалл по всей его длине или ширине).

Систему межсоединений FPGA образуют сегментированные линии и переключательные блоки ПБ (PSM, Programmable Switching Matrix). Функциональные блоки имеют квадратные геометрические очертания, их выводы распределены по всем сторонам квадрата для облегчения коммутируемости. Для межсоединений функциональных блоков во внутренней области кристалла имеются три типа связей: одинарной длины, двойной длины и длинные линии. Упрощенная система коммутации FPGA показана на рис. 1.13, *а* на примере FPGA семейства XC4000 фирмы Xilinx.

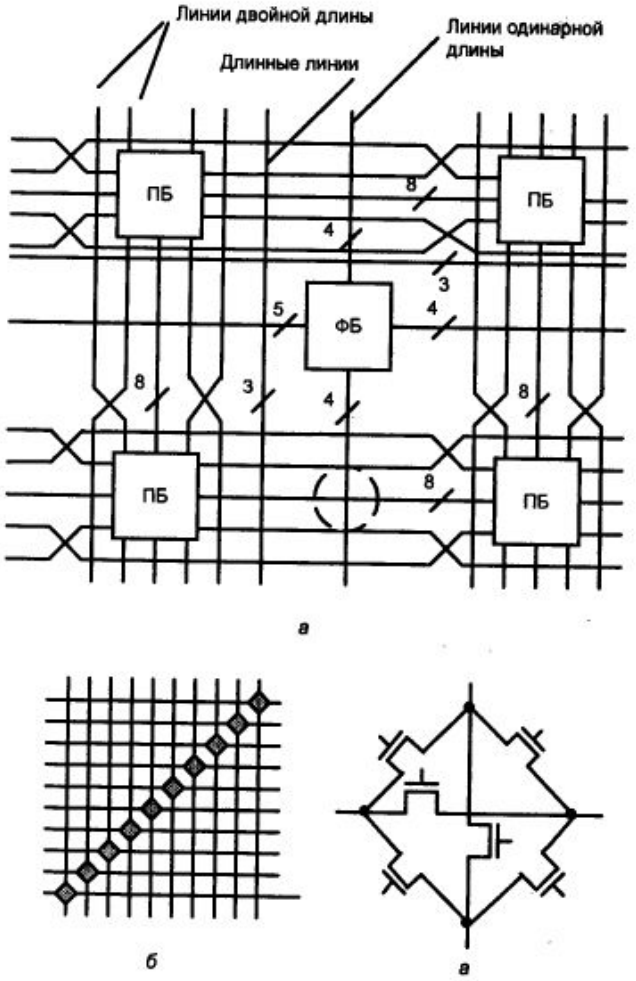
На пересечениях вертикальных и горизонтальных каналов расположены переключательные блоки (рис. 1.13, *б*). В пределах ПБ пересекаются вертикальные и горизонтальные линии связей, и в каждом пересечении имеется цепь из 6 транзисторов для установления того или иного соединения. Сигнал, поступающий в ПБ по какой-либо линии (например, горизонтальной), может быть направлен вверх, вниз или прямо в зависимости от того, какой транзистор будет открыт при конфигурировании FPGA. Возможна и одновременная передача сигнала по нескольким направлениям, если требуется его разветвление.

Линии одинарной длины осуществляют, преимущественно, межсоединения соседних или близлежащих ФБ, линии двойной длины огибают переключательные блоки, соседние по отношению к данному, и проходят к следующим, чем облегчается установление более длинных связей. По три длинных линии, пересекающих весь кристалл по длине (ширине), реализуются сверху, снизу и по обоим бокам ФБ.

Выводы функциональных блоков пересекают горизонтальные и вертикальные каналы трассировки, проходящие непосредственно около них, и могут программируемыми точками связи подключаться к линиям каналов. Дальнейшее направление сигналов в нужные цепи осуществляется переключательными блоками.

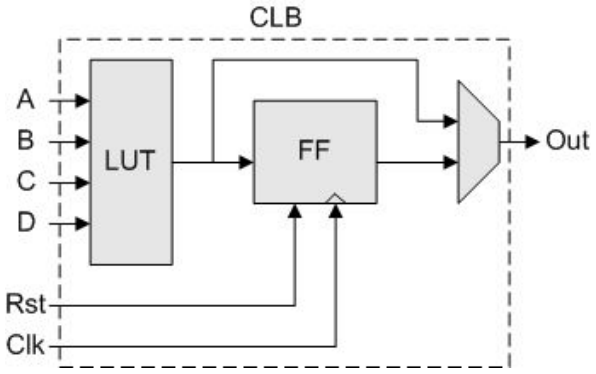
Линии двойной длины сгруппированы в пары, имеется по 4 вертикальных и горизонтальных линии, обеспечивающих более быструю и эффективную передачу сигналов на средние расстояния. Длинные линии, рассчитанные на передачу сигналов на большие расстояния и при большой нагрузке, имеют в середине ключ, разделяющий линию на две части ((рис. 1.13, *в*).

Кроме системы коммутации для функциональных блоков FPGA (в частности, семейство Spartan) может иметь дополнительные трассировочные ресурсы, расположенные в виде кольца вне пределов матрицы ФБ. Эти ресурсы позволяют изменять назначение вводов/выводов микросхемы и облегчают тем самым модификацию проекта, реализованного на FPGA, без влияния на разводку печатных плат, на которых монтируются микросхемы.



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

Функциональные блоки FPGA

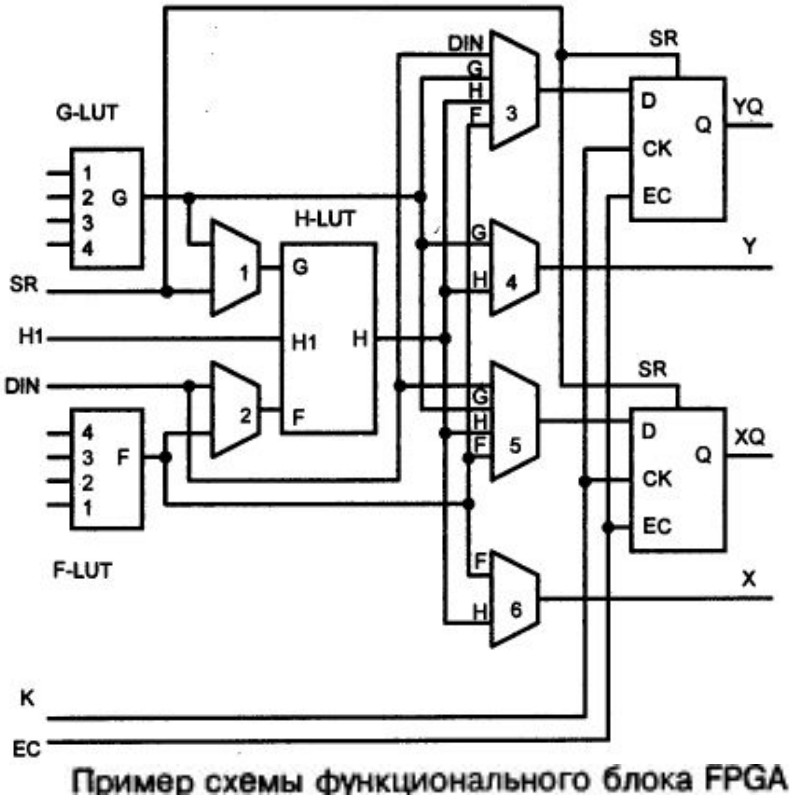


1.Табличный ФП типа LUT представляет собой ЗУ, хранящее значения искомых функций, считываемые по адресу-аргументу. ЗУ с организацией $2^m * n$ имеет *m* адресных входов и n выходных линий, может хранить таблицу для считывания я функций от *m* переменных. Время вычисления результата равно времени считывания 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 цепоче транзисторов с п - и р -каналами. Из таких пар собираются традиционные для КМОП - схем логические элементы.

ФБ FPGA с триггерной памятью-конфигурации на примере микросхемы семейства Spartan фирмы Xilinx. В функциональных блоках этих микросхем 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).



Пример схемы функционального блока FPGA

Через верх. вх. MUX1 и нижн. вх. MUX2 ф-ции G и F могут быть поданы на ФП-Н (ЗУ 8x1) для образования "ф-ции от ф-ций" с целью получения результирующей ф-ции, зависящей от более чем 4-х аргументов. К третьему входу ФП-Н подключен входной сигнал H1, так что H =f(G , F, H1). Аргументами для ФП-Н, поступающими от MUX 1 и 2, в зависимости от их программирования может быть не только набор G, F, H1 , но также G, HI , DIN; SR, HI , DIN; SR, HI , F. Линии DIN и SR используются либо для передачи в триггер непосредственно входных данных и сигнала установки/сброса (Set/Reset), либо как входы ФП-Н.

Перечисленные ресурсы логической части ФБ позволяют воспроизводить:

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

Сигналы HI , DIN, SR, EC являются для ФБ входными. MUX 3...6 направляют сигналы данных управления на триггеры 1 и 2. Триггеры могут использоваться для фиксации и хранения выходных сигналов функциональных преобразователей или же работать независимо. Входной сигнал ФБ DIN и Сигнал H1 можно передавать любому триггеру.

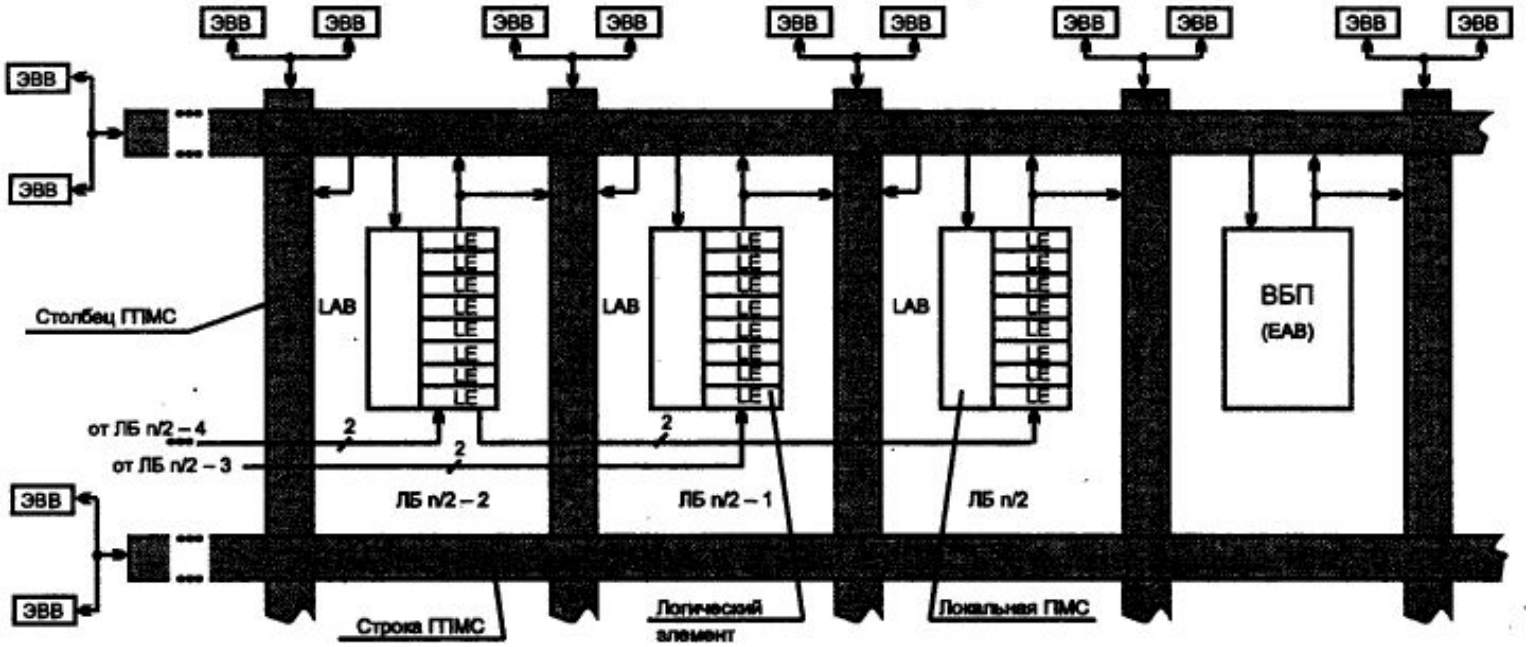
ПЛИС с комбинированной архитектурой
Семейство FLEX

По мере роста уровня интеграции ПЛИС их архитектуры усложнялись, появились *архитектуры в той или иной мере сочетающие достоинства CPLD и FPGA*. К числу появившихся микросхем с такой архитектурой в первую очередь можно отнести семейство FLEX (Flexible Logic Element matriX) фирмы Altera. Комбинированные архитектуры затем нашли применение в многочисленных БИС/СБИС, в том числе типа "система на кристалле". Микросхемы высшей сложности строят по все более оригинальным архитектурам, не только сочетающим черты CPLD и FPGA, но и обладающим новыми особенностями. На рис. 1.14 приведена структура микросхем семейства FLEX.

В схеме на рис. 1.14 показан включенный в середине строк встроенный блок памяти (у большинства микросхем семейства FLEX10K его емкость равна 2048 бит). Такой блок может конфигурироваться как ЗУ с организацией 256×8, или 512×4, или 1024×2, или 2048×1 и использоваться не только для хранения данных, но и как табличный ФП для реализации сложных функций с числом аргументов 8—10 (в частности, на блоках EAB строятся быстродействующие арифметико-логические устройства АЛУ, перемножители 4×4 и т. д.).

Микросхемы семейства FLEX имеют функциональные блоки (LABs, Logic Array Blocks) с логическими элементами ЛЭ (LEs, Logic Elements), содержащими функциональные преобразователи ФП табличного типа (LUTs). Функциональные блоки расположены в виде матрицы, между их строками и столбцами проходят горизонтальные и вертикальные трассировочные каналы, что характерно для FPGA. В то же время, трассы в каналах не сегментированы, а непрерывны, что типично для CPLD. Поскольку, как уже отмечалось, в схемах с большим числом функциональных блоков применение единой коммутационной матрицы затруднено, система коммутации имеет два уровня межсоединений — глобальный и локальный. Локальная программируемая матрица соединений (локальная ПМС или ЛПМС) обеспечивает межсоединения логических элементов ЛЭ, из которых составляются функциональные блоки LABs. В состав LAB входят 8 логических элементов. Соединения между блоками LAB обеспечиваются глобальной программируемой матрицей соединений ГПМС, к концам строк и столбцов которой подключаются блоки ввода/вывода (IOBs, Input/Output Blocks).

Начиная с семейства FLEX10K, в составе многих микросхем ПЛИС появились встроенные блоки памяти ВБП (EABs, Embedded Array Blocks). Ранее имелась возможность использования в качестве субмодулей памяти лишь тех ресурсов, которые имеются в ФП типа LUT.



Логические элементы ПЛИС семейства FLEX

Логические элементы микросхем семейства FLEX (рис. 1.15) имеют в своей основе 4-входовые ФП табличного типа (LUT). Особенностью схем, которые могут быть построены из этих логических элементов, является наличие специальных трактов переноса, образуемых цепочками схем переноса СП, и трактов каскадирования, образуемых схемами каскадирования СК с непосредственными и быстродействующими связями между логическими элементами по указанным трактам.

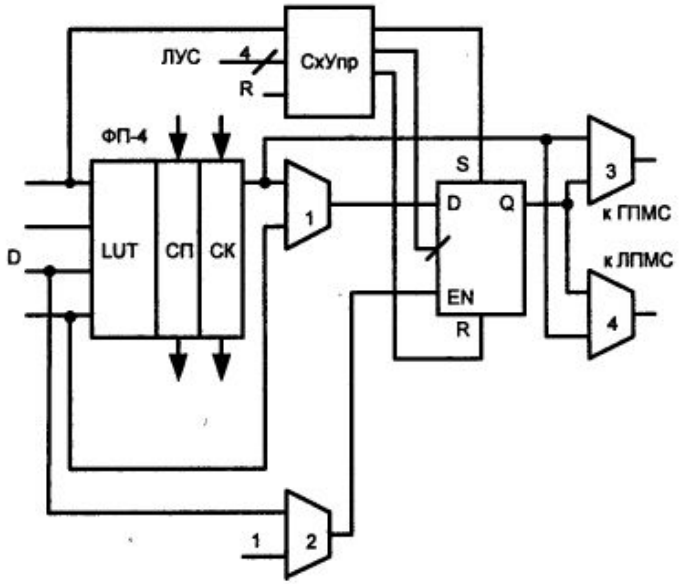


Рис. 1.15. Схема логического элемента микросхем семейства FLEX

ФП с 4 входами имеет 16 бит памяти и для воспроизведения функций 4 аргументов организуется в варианте 16×1. Те же самые 16 бит можно использовать в виде двух табличных ФП с организацией 8×1, реализующих две функции 3 переменных, что отвечает, например, потребностям построения разрядных схем сумматоров с последовательным переносом, разрядных схем некоторых счетчиков и т. д. Длинные цепочки переносов формируются в пределах нескольких LAB. В микросхемах семейства FLEX задержка цепи переноса мала (приблизительно 1 нс), что делает целесообразным применение в проектах многих простых схем с последовательными переносами даже для быстродействующих устройств.

Цепочка каскадирования используется для получения функций с числом аргументов более 4. Три соседних ЛЭ можно применить для воспроизведения частичных функций, а затем с помощью каскадирования сформировать из этих функций окончательный результат (рис. 1.16, *а*). При получении из частичных функций единой функции многих переменных частичные функции смежных ЛЭ объединяются любой логической операцией над двумя переменными, кроме сложения по модулю 2 и функции равнозначности (на рис. 1.16, *а* функции обозначены условными значками).

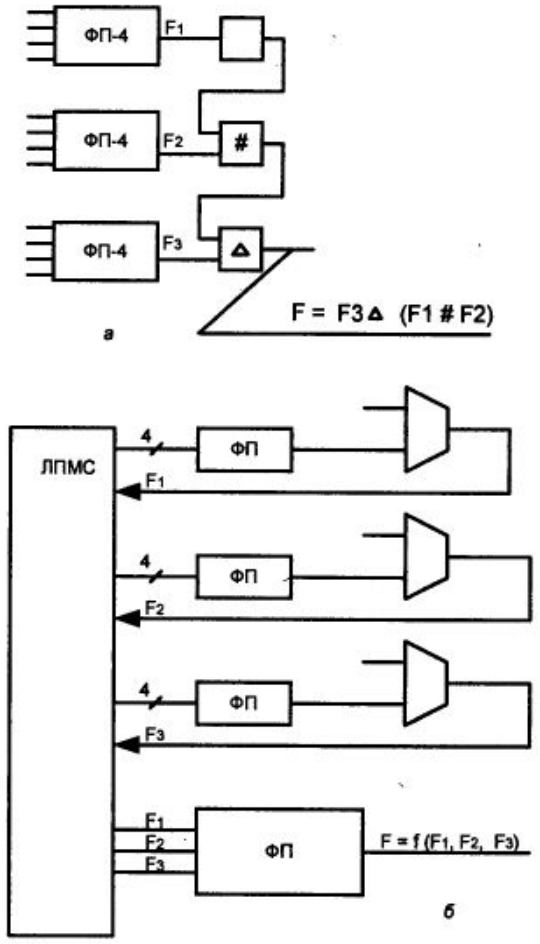


Рис. 1.16. Способы воспроизведения функций многих переменных методами каскадирования (*а*) и декомпозиции *(б)*

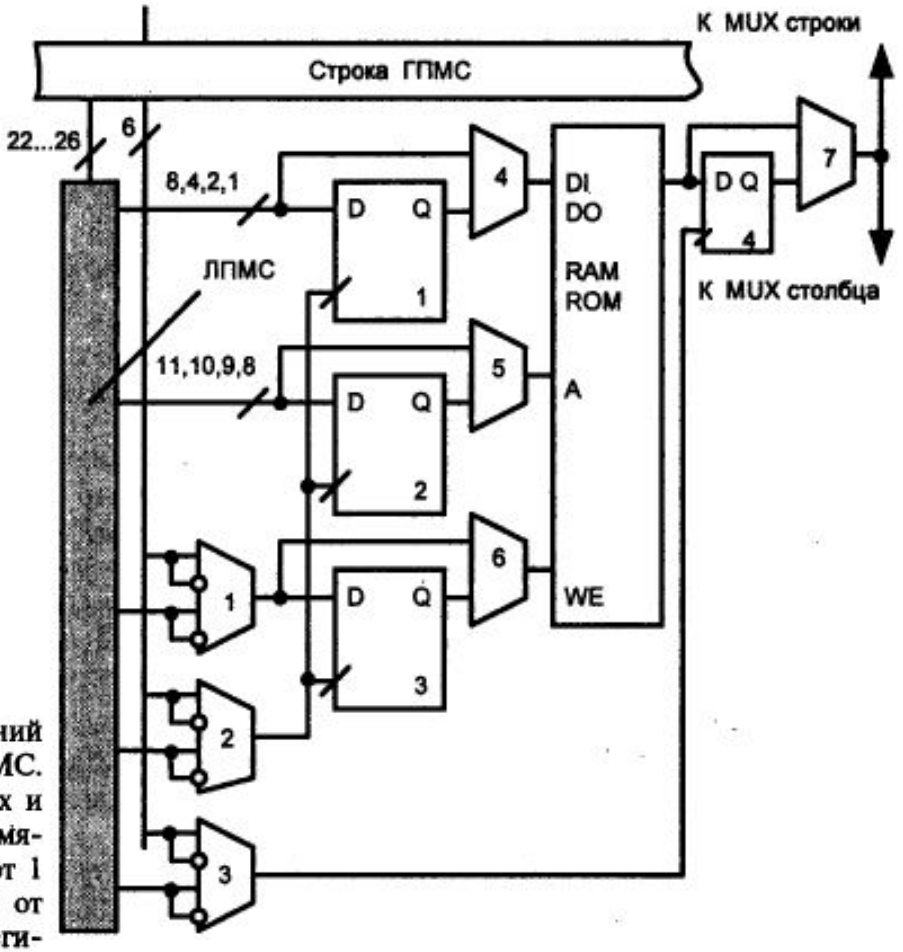
Функции многих переменных можно получить и другим способом, используя обратные связи. При этом сначала вырабатывается некоторая функция

Блоки памяти ПЛИС семейства FLEX

В состав СБИС семейства FLEX10K были впервые включены встроенные блоки памяти ВБП 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 логических блоков. Например, один встроенный блок памяти при организации 256×8 реализует перемножитель 4×4, 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.

В структуре встроенных блоков памяти (рис. 1.17) кроме модуля памяти RAM/ROM имеется несколько синхронных D-триггеров и программируемых мультиплексоров. Локальная программируемая матрица соединений ЛПМС получает 22—26 сигналов от строки глобальной матрицы ГПМС. Регистры 1 и 2 программируются для передачи в модуль памяти данных и адресов разной разрядности в зависимости от заданной конфигурации памяти. В блоке с емкостью 2 Кбит разрядность данных может изменяться от 1 до 8, а разрядность адреса от 11 до 8. Запись в память в зависимости от программирования мультиплексоров 4—6 может быть синхронной (от регистров по сигналам тактирования) или асинхронной (непосредственно от ЛПМС).



.17. Структура встроенных блоков памяти в микросхемах семейства FLEX10K

Сигналы управления регистрами 1—3 поступают от глобальной шины управляющих 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.

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;
```

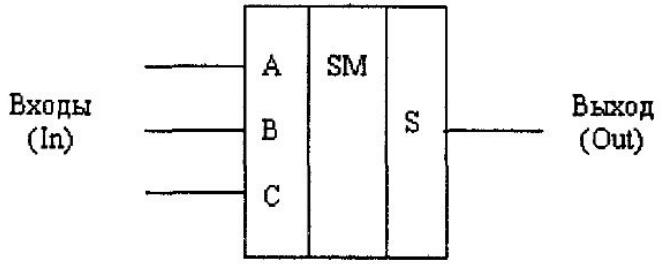


Рис. 1.2. Интерфейс объекта проекта SM

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).

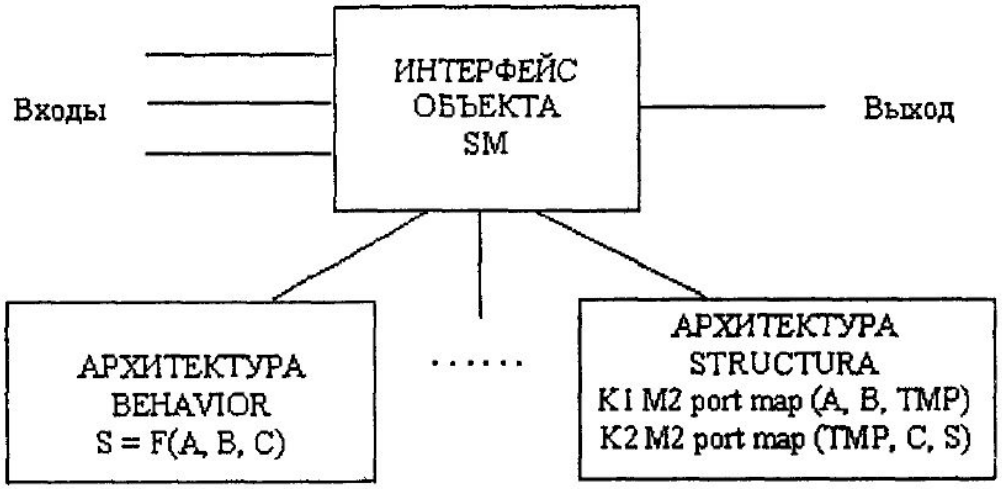


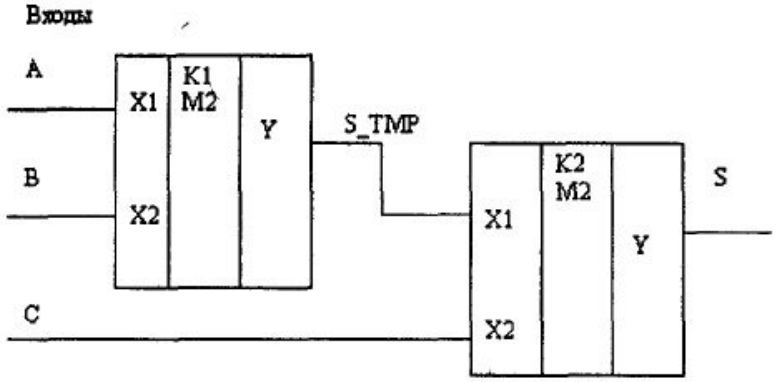
Рис. 1.3. Пример соответствия множества архитектур одному интерфейсу в VHDL

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

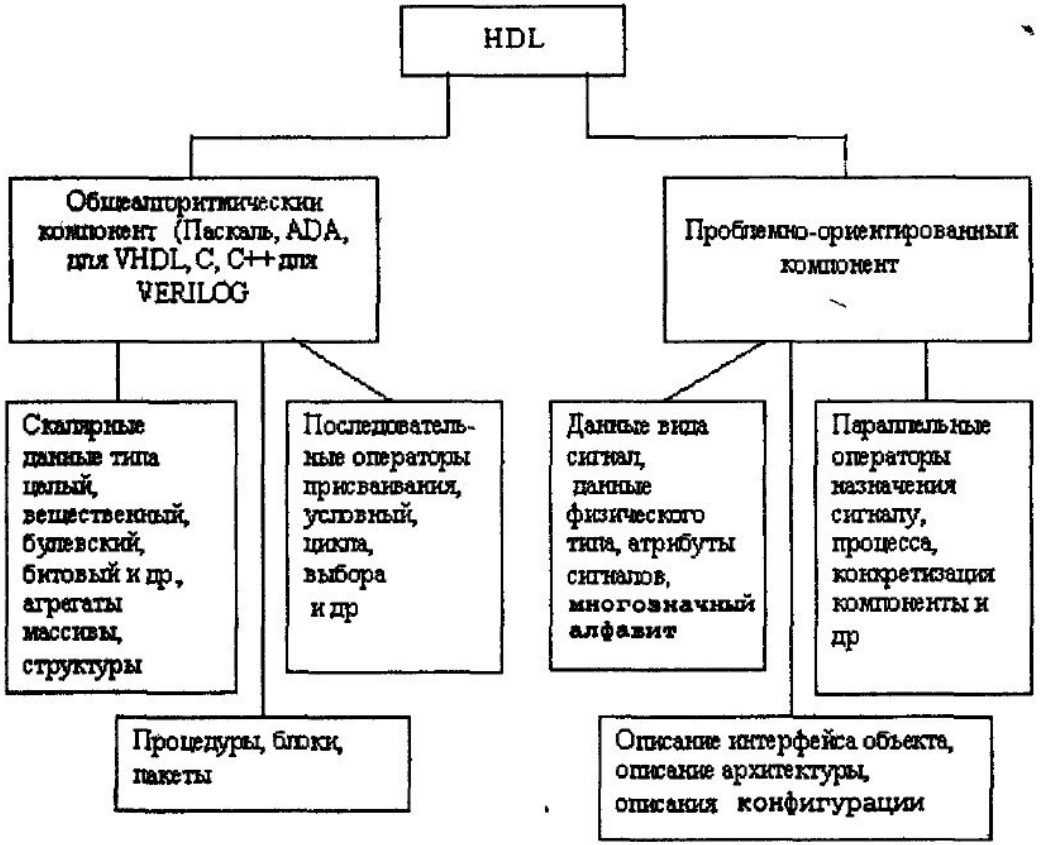
```
    `timescale 1 ns/100 ps
  module adder_tb;

  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
integer	
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

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 сбр
сом RST_N и тактовым сигналом CLK. Триггер принимает информацию
фронту CLK. VERILOG-описание выполнено в стиле VERILOG-2000, VHDL
пользует 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
```