IV. Элементы и узлы ЭВМ

Триггеры

Триггер – логический элемент, который может находиться в одном из двух устойчивых состояний.

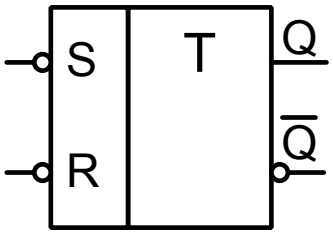
S, J – входы установки триггера в «1».
R, K – входы установки триггера в «0».
T – счетный вход триггера.
D – информационный вход триггера D
C – вход синхронизации
Q – прямой выход триггера
$\overline{Q}$ – инверсный выход триггера

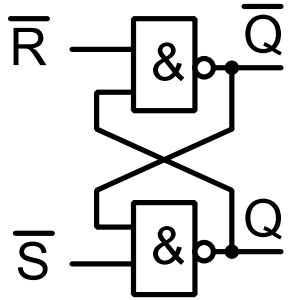
по логике:
- RS, D, T, JK

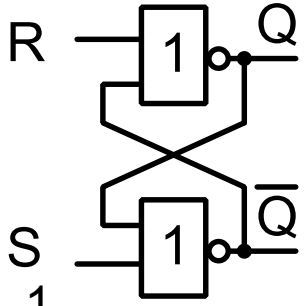
по способу приема:
- Асинхронные,
- Синхронные,
- Одноступенчатые,
- Двухступенчатые

по способу синхронизации
- управляемые фронтом/спадом (динамические).
- управляемые уровнем (защелки)

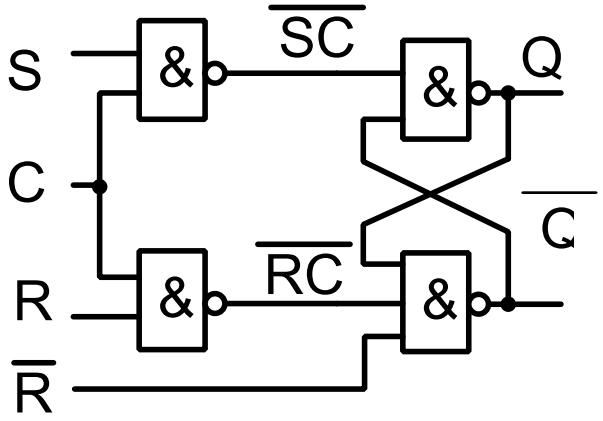
Одноступенчатый асинхронный RS-триггер

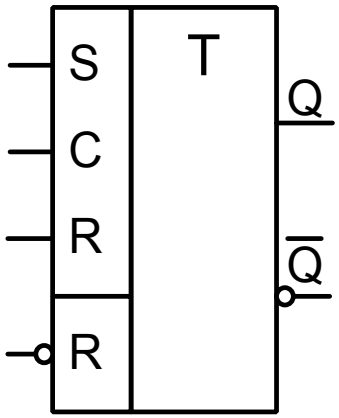




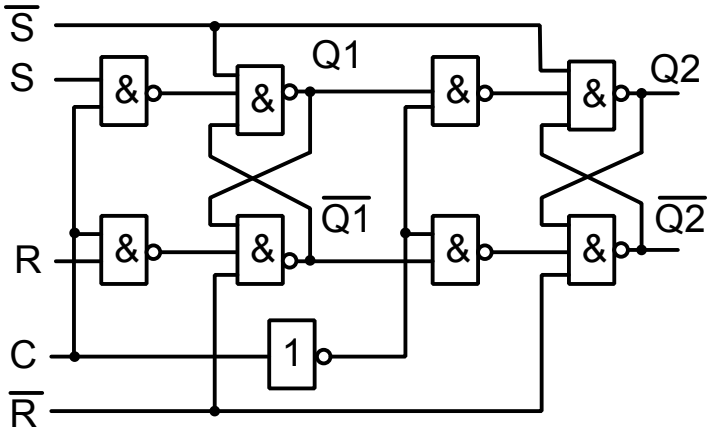


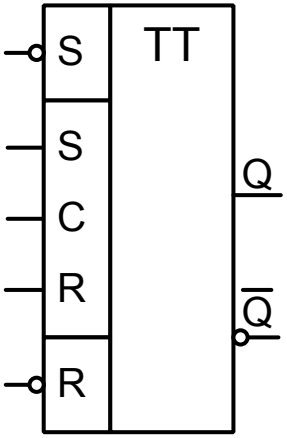
Одноступенчатый синхронный RS-триггер

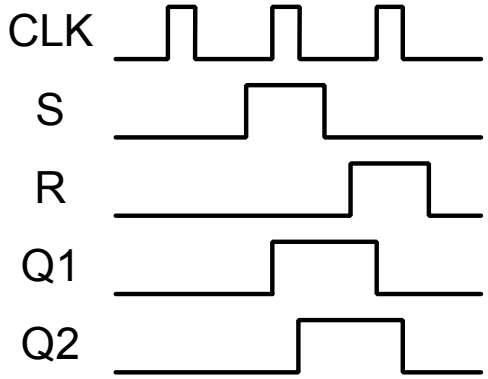




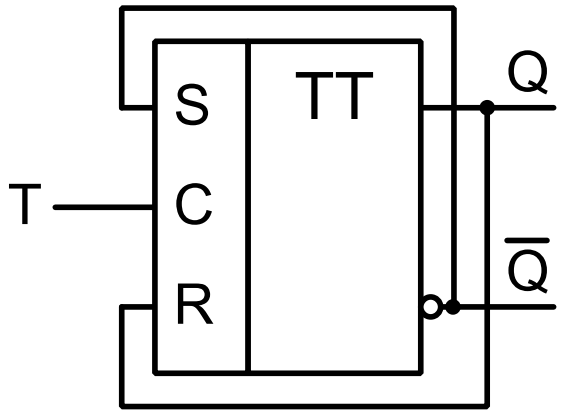
Двухступенчатый синхронный RS-триггер

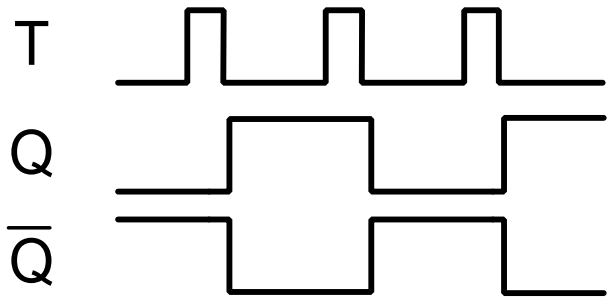




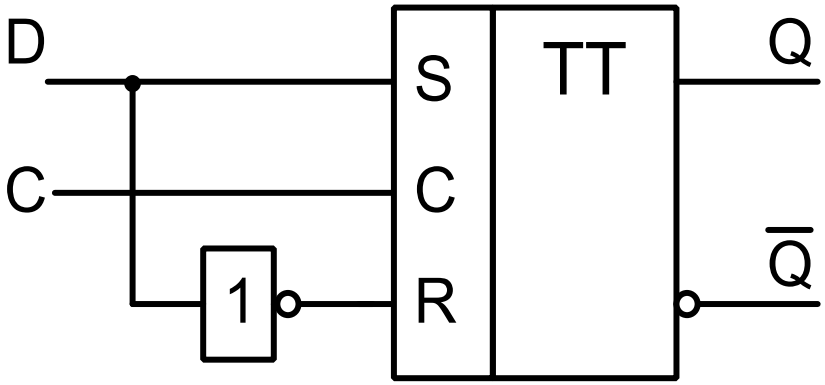


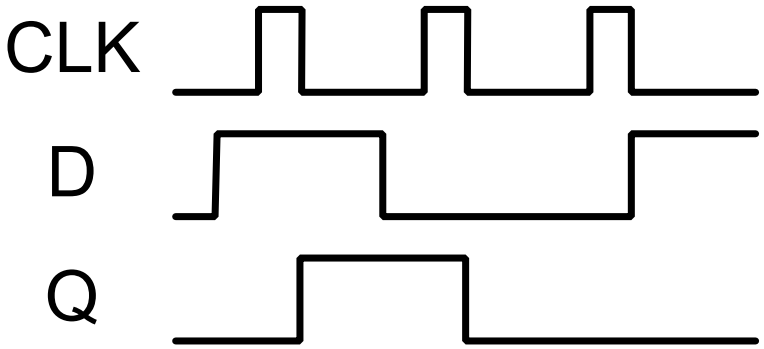
Т-триггер



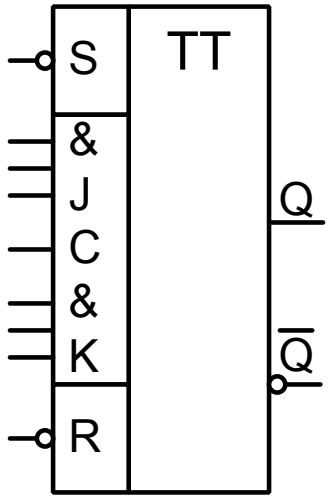


D-триггер

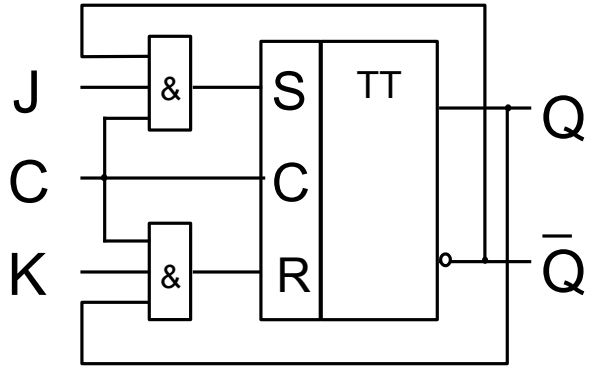




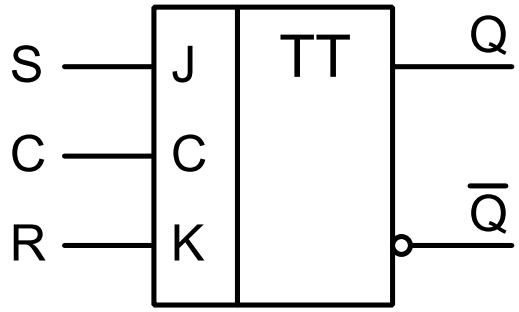
JK-триггер



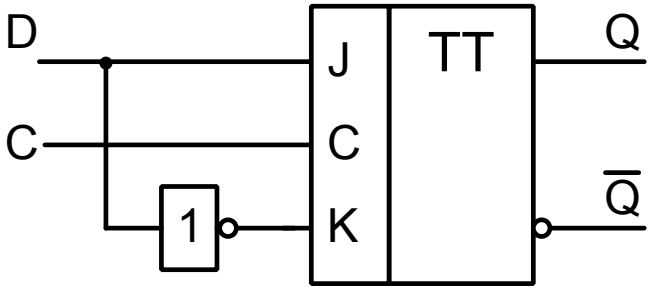
J(t)	K(t)	Q(t+1)	Режим
0	0	Q(t)	Хранение
0	1	0	Установка «0»
1	0	1	Установка «1»
1	1	Q(t)	Инверсия



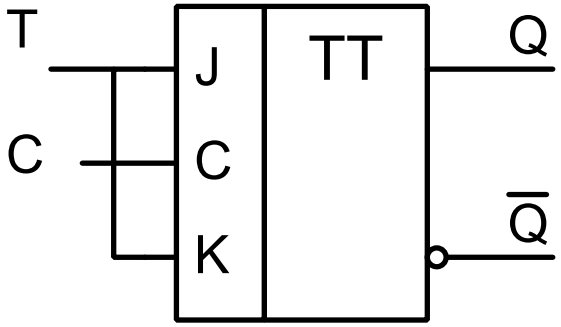
RS-триггер на основе JK-триггера



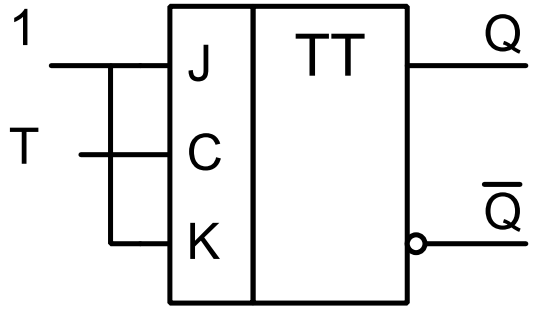
D-триггер на основе JK-триггера



Синхронный T-триггер на основе JK-триггера



Асинхронный T-триггер на основе JK-триггера



Динамические триггеры

DV-триггер

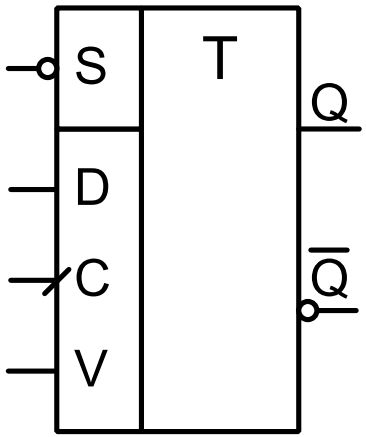


Схема DV-триггера

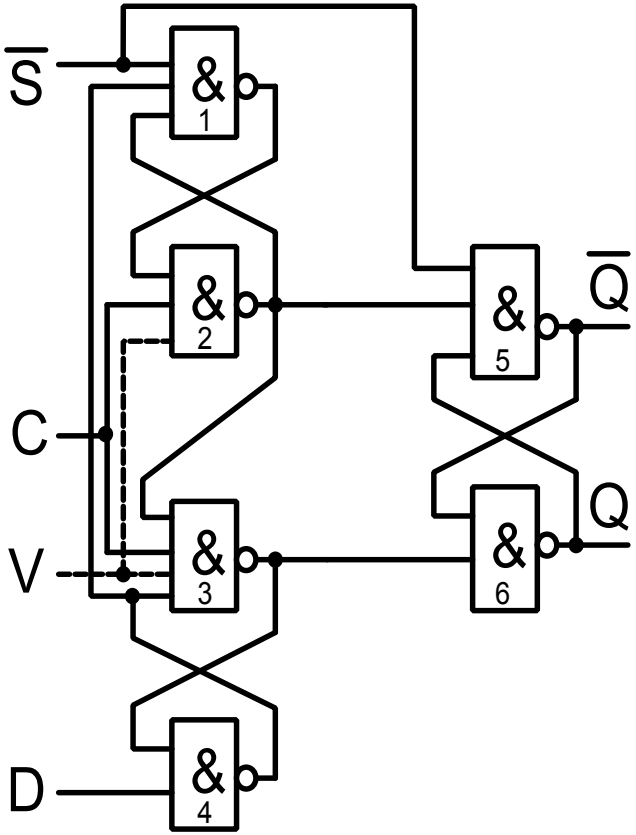


Диаграмма работы DV-триггера

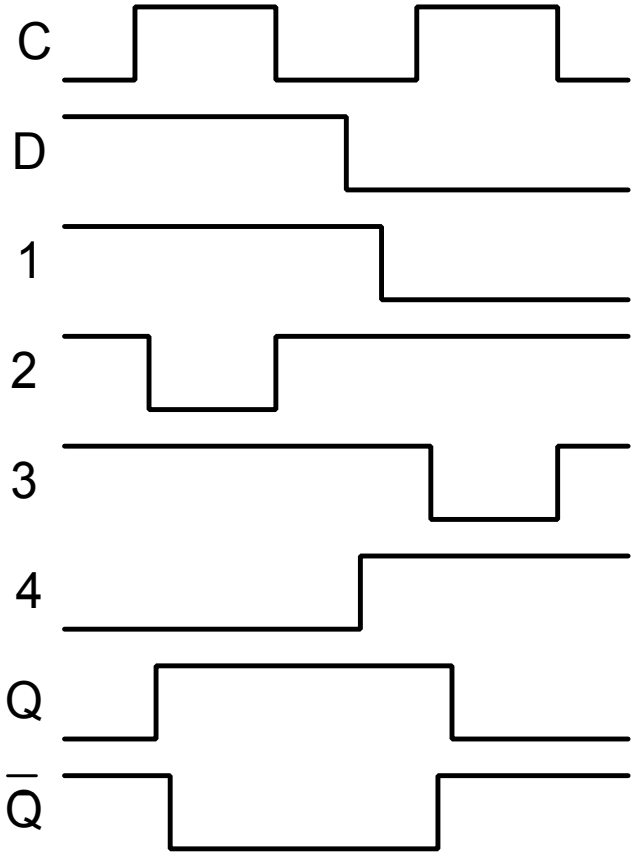
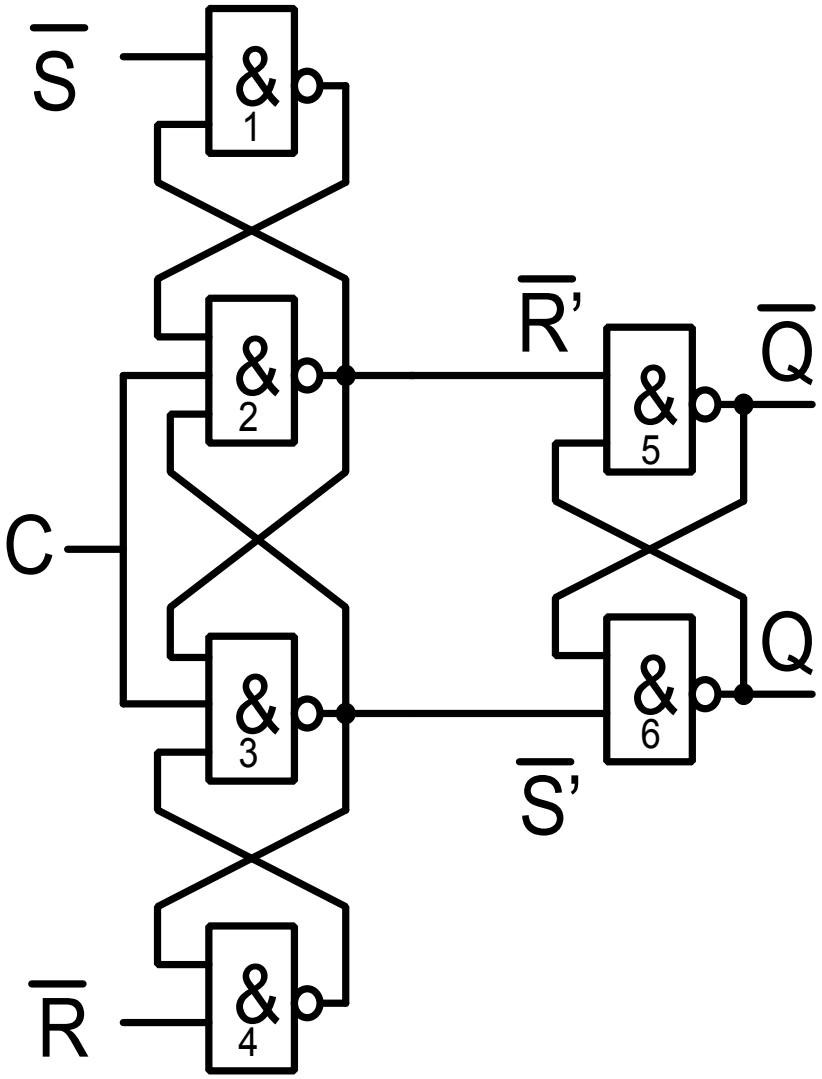
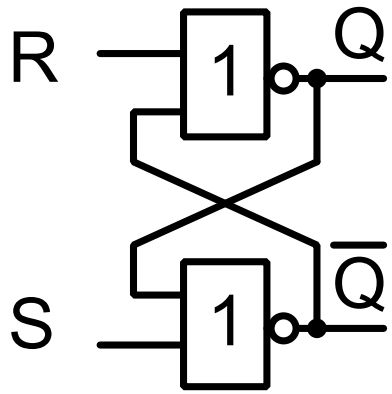
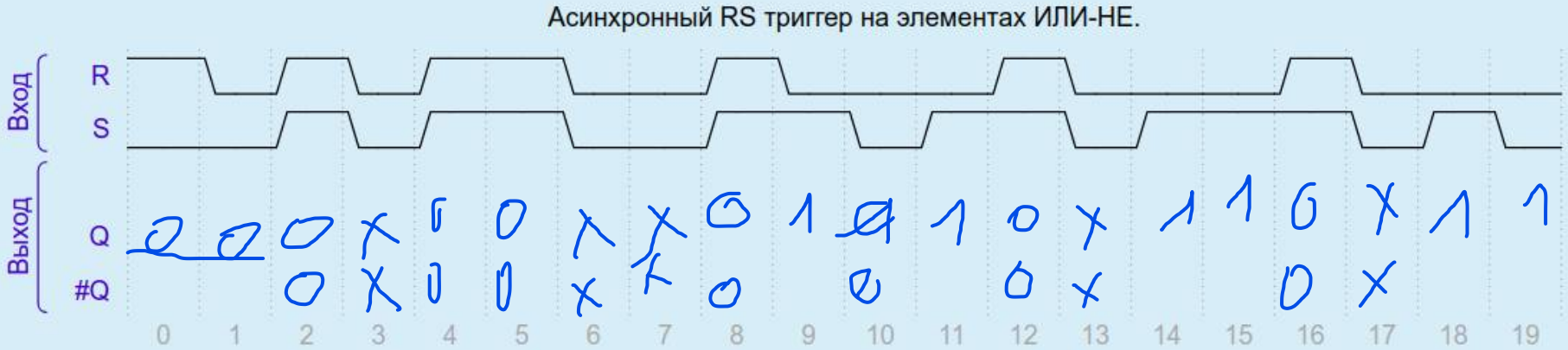


Схема трех триггеров



Одноступенчатый асинхронный RS-триггер на элементах ИЛИ-НЕ





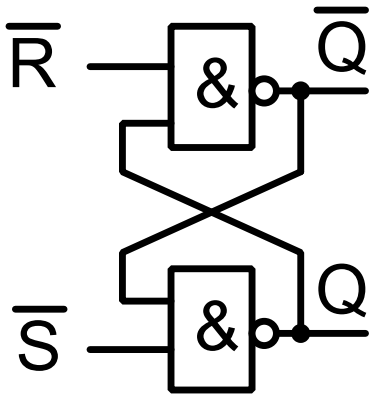
Нарисовать диаграмму выходных сигналов."#" - инверсный сигнал.

В ответе необходимо указать значения Q и #Q (т.е. не-Q) в виде последовательности 20 значений 0/1/X (X - неопределенное состояние).
Например: 0X11100X100000111101

Q: 000X00XX01110X110X11 ✔

#Q: 110X00XX00000X000X00 ✔

Одноступенчатый асинхронный RS-триггер на элементах И-НЕ



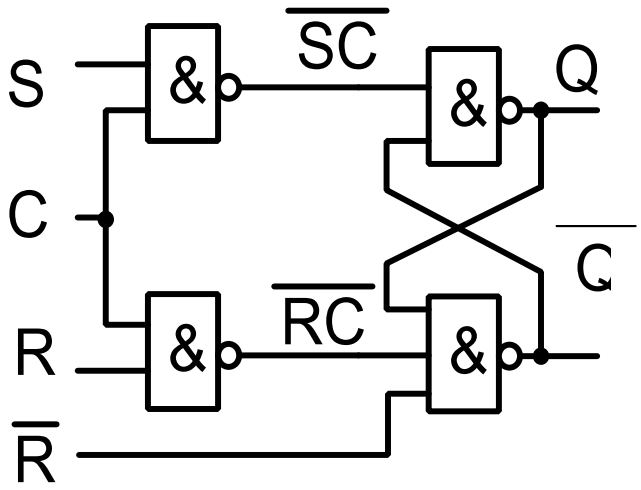
Нарисовать диаграмму выходных сигналов."#" - инверсный сигнал.

В ответе необходимо указать значения Q и #Q (т.е. не-Q) в виде последовательности 20 значений 0/1/X (X - неопределенное состояние).
Например: 0X11100X100000111101

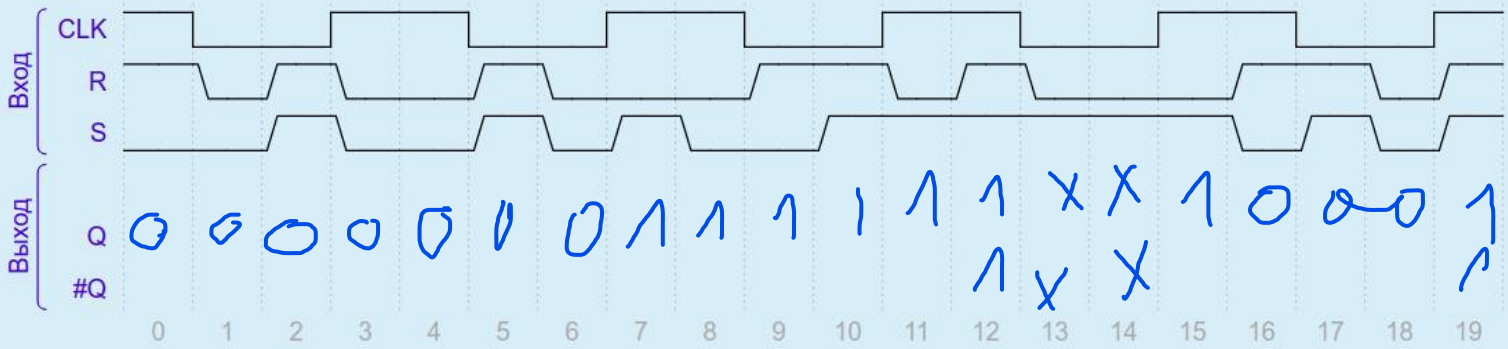
Q: 00100011011X11011100 ✔

#Q: 11011110111X10101011 ✔

Одноступенчатый синхронный RS-триггер на элементах И-НЕ



Синхронный одноступенчатый RS триггер на элементах И-НЕ.



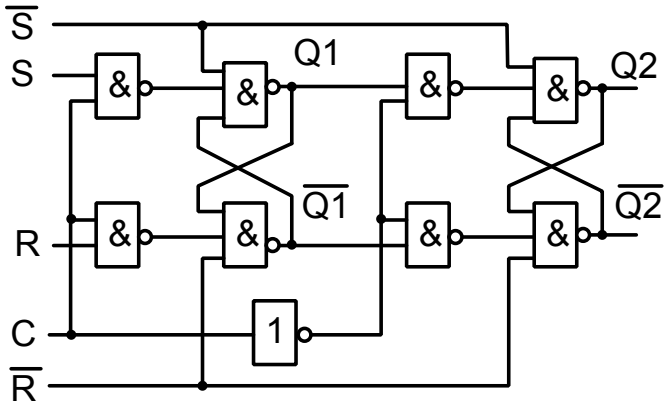
Нарисовать диаграмму выходных сигналов."#" - инверсный сигнал.

В ответе необходимо указать значения Q и #Q (т.е. не-Q) в виде последовательности 20 значений 0/1/X (X - неопределенное состояние).
Например: 0X11100X100000111101

Q: 0000000111111XX10001 ✔

#Q: 1111111000001XX01111 ✔

Двухступенчатый синхронный RS-триггер на элементах И-НЕ



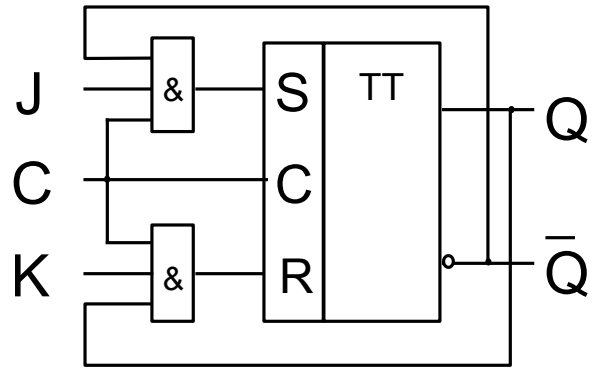
Нарисовать диаграмму выходных сигналов."#" - инверсный сигнал.

В ответе необходимо указать значения Q и #Q (т.е. не-Q) в виде последовательности 20 значений 0/1/X (X - неопределенное состояние).
Например: 0X11100X100000111101

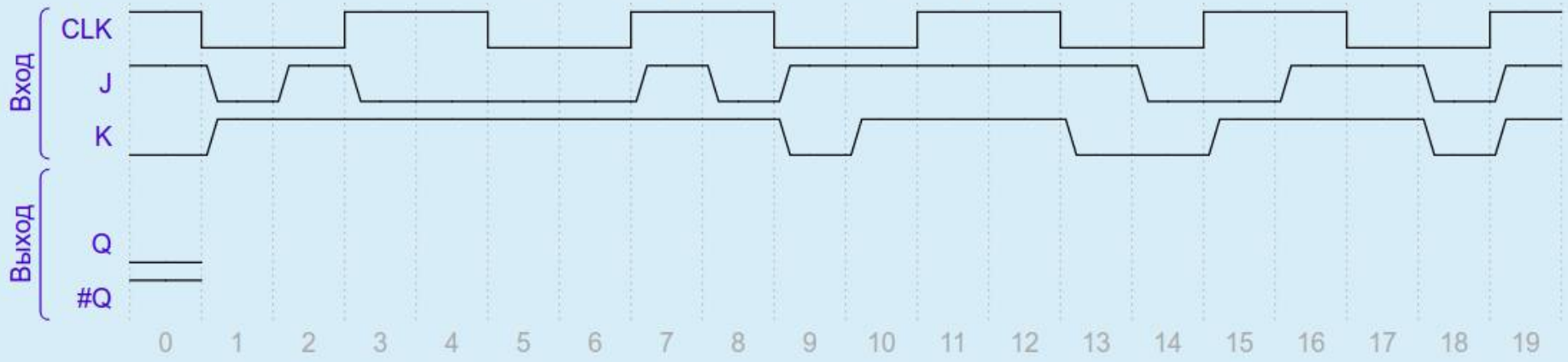
Q: 01111111100001111111

#Q: 10000000011110000000

Двухступенчатый синхронный JK-триггер на элементах И-НЕ



Синхронный двухступенчатый JK триггер на элементах И-НЕ.



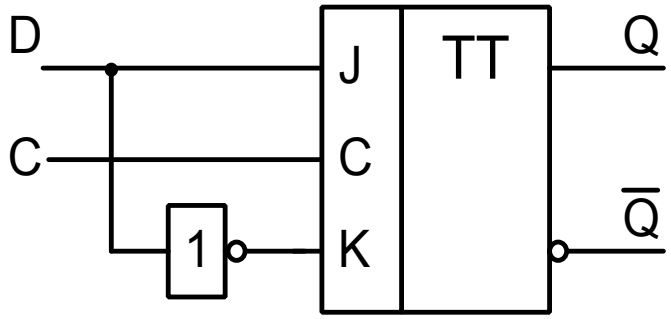
Нарисовать диаграмму выходных сигналов."#" - инверсный сигнал.

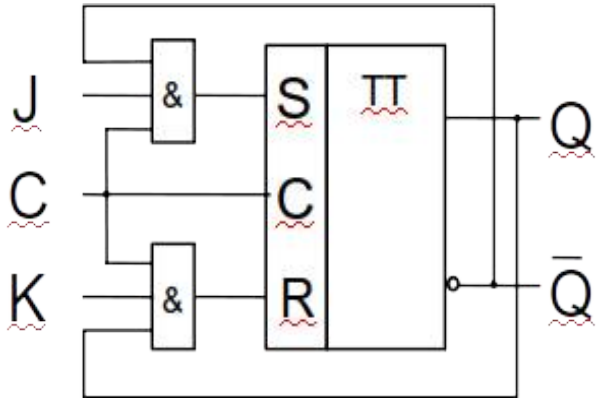
В ответе необходимо указать значения Q и #Q (т.е. не-Q) в виде последовательности 20 значений 0/1/X (X - неопределенное состояние).
Например: 0X11100X100000111101

Q: 01111000000001111000 ✔

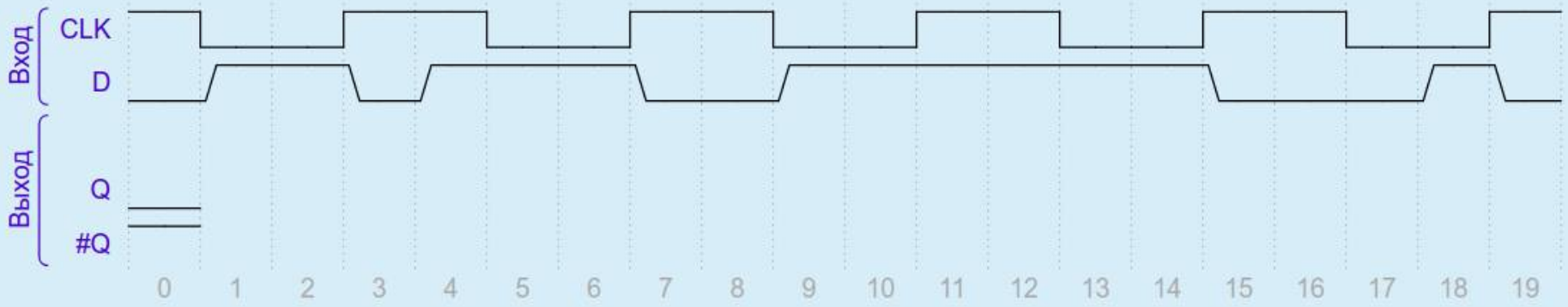
#Q: 10000111111110000111 ✔

Двухступенчатый синхронный D-триггер на элементах И-НЕ





Синхронный двухступенчатый D триггер на элементах И-НЕ.



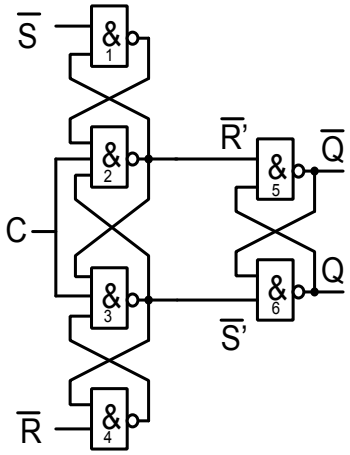
Нарисовать диаграмму выходных сигналов."#" - инверсный сигнал.

В ответе необходимо указать значения Q и #Q (т.е. не-Q) в виде последовательности 20 значений 0/1/X (X - неопределенное состояние).
Например: 0X11100X100000111101

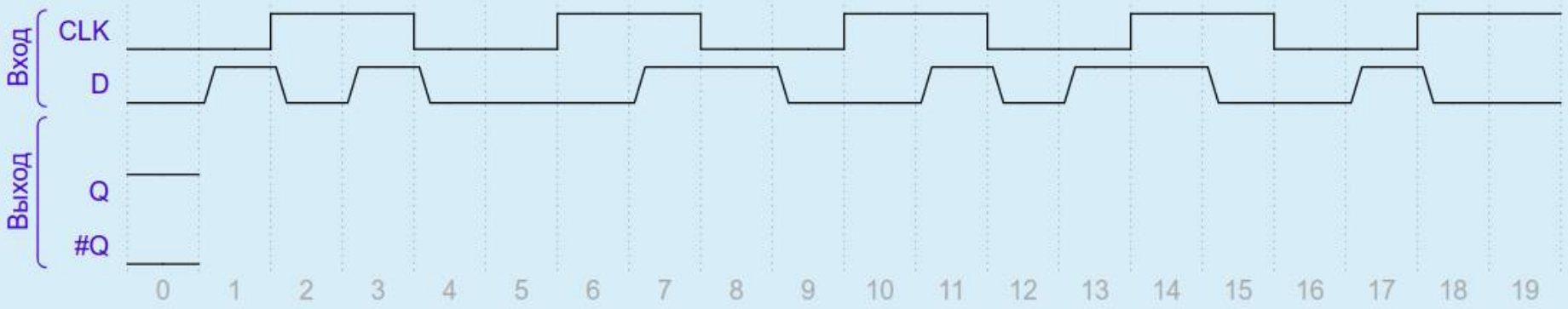
Q: 00000111100001111000 ✔

#Q: 11111000011110000111 ✔

Динамический D-триггер на элементах И-НЕ



Динамический D триггер.



Нарисовать диаграмму выходных сигналов."#" - инверсный сигнал.

В ответе необходимо указать значения Q и #Q (т.е. не-Q) в виде последовательности 20 значений 0/1/X (X - неопределенное состояние).
Например: 0X11100X100000111101

Q: 11111100000000111111

#Q: 00000011111111000000

Регистры

Регистром называется устройство, предназначенное для запоминания слова, а также для выполнения над словом некоторых логических преобразований.

Операции, выполняемые регистром:
- Сброс (установка в 0);
- Прием слова (запись);
- Выдача слова (чтение);
- Сдвиг слова (сдвиг вправо, влево, циклический сдвиг);
- Преобразование параллельного кода в последовательный;
- Поразрядные логические операции.

Условное обозначение универсального регистра

Сигнал	Тип	Описание
CLK	Вход	Синхросигнал
RESET	Вход	Сброс
LOAD	Вход	Разр. загрузки
HOLD	Вход	Запр. сдвига
UP	Вход	Направление
DL	Вход	Младший бит
DU	Вход	Старший бит
D#	Вход	Входное слово
OE	Вход	Разрешение выдачи
Q#	Выход	Выходное слово
QL	Выход	Младший бит
QU	Выход	Старший бит

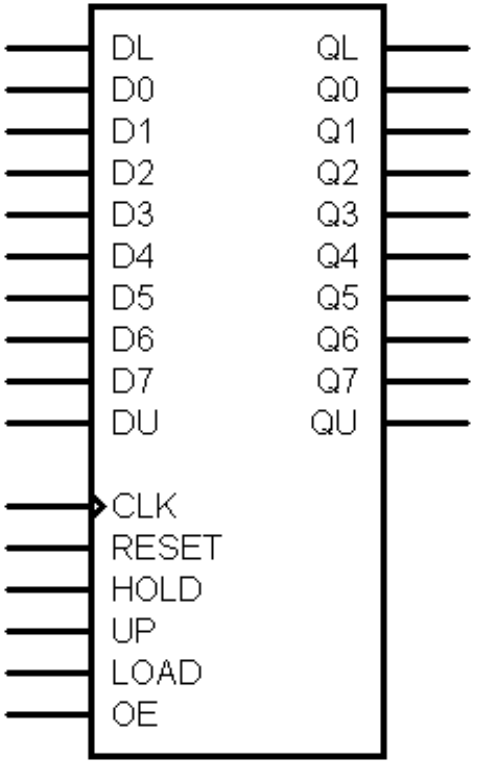
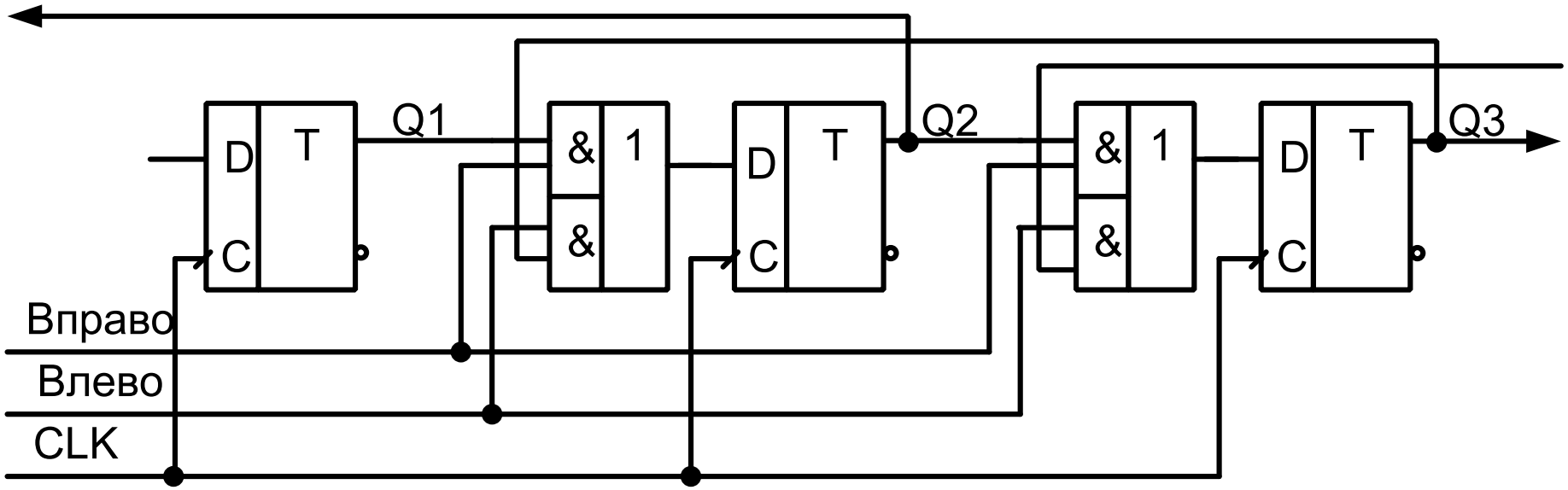


Схема сдвигового регистра



Счетчики

Счетчиком называется узел ЭВМ, предназначенный для подсчета входных сигналов.
Модуль счета: число возможных состояний счетчика.

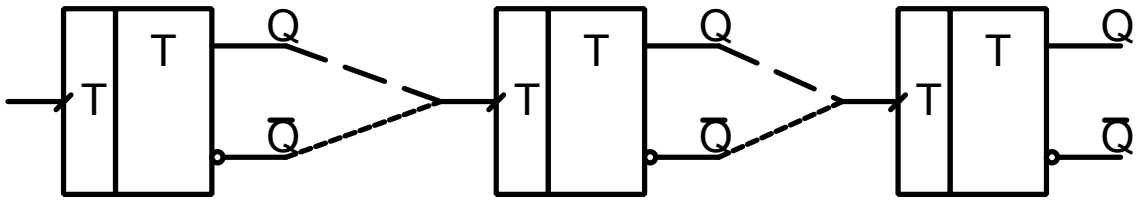
Классификация счетчиков.
По способу счета: суммирующие, вычитающие, реверсивные.
По модулю счета: двоичные, десятичные, ….
По способу распространения переноса: с параллельным переносом, с последовательным переносом, с групповой структурой.
По способу синхронизации: асинхронные, синхронные.
По режиму работы: для подсчета входных сигналов, для деления частоты.

Таблица состояний

$X_{сч}$	Q4	Q3	Q2	Q1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
		…		
15	1	1	1	1

перенос

— — Обратный счет



------- Прямой счет

Счетчик с последовательным переносом

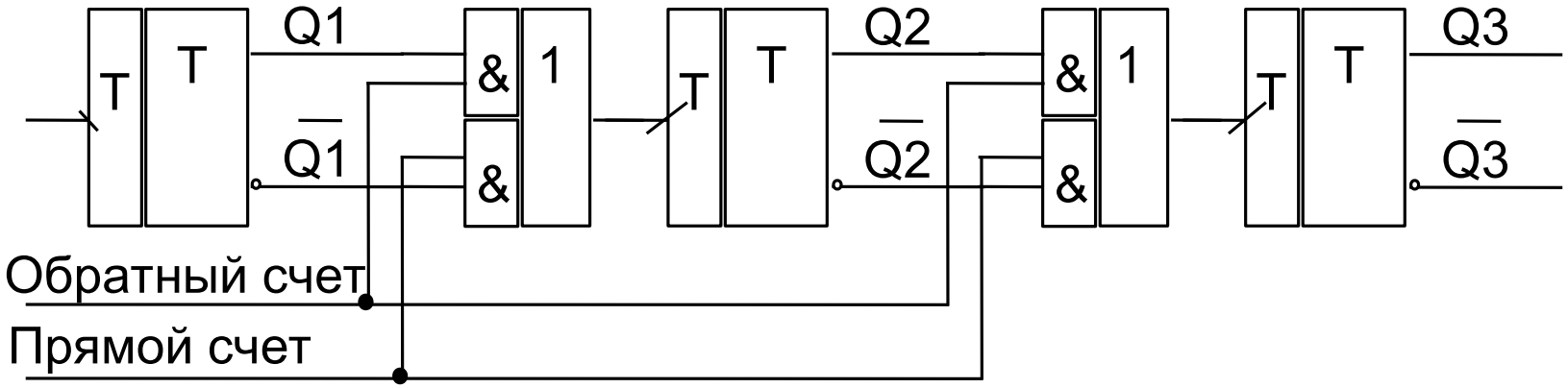
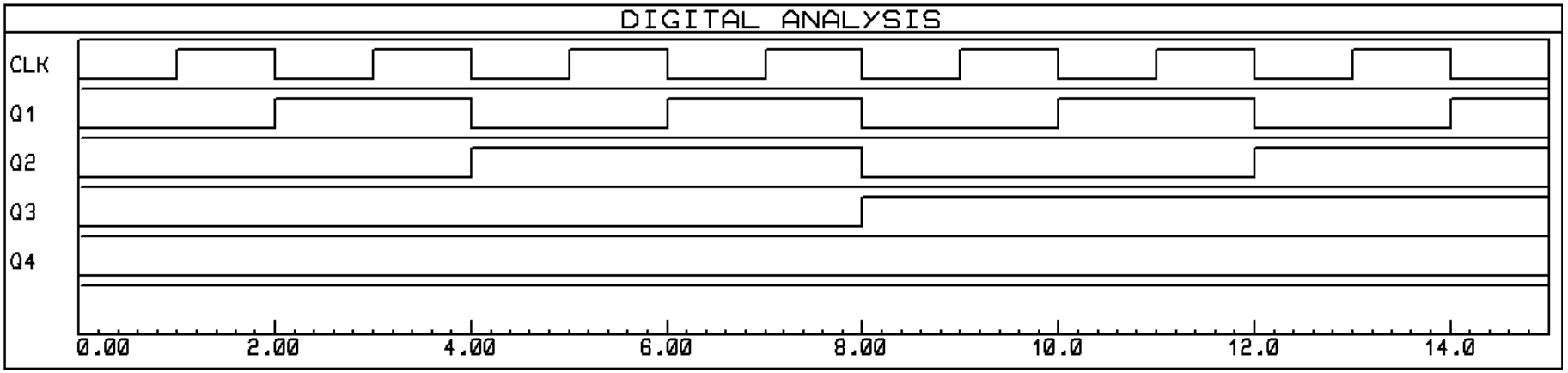


Диаграмма работы (прямой счет)



Счетчик с параллельным переносом

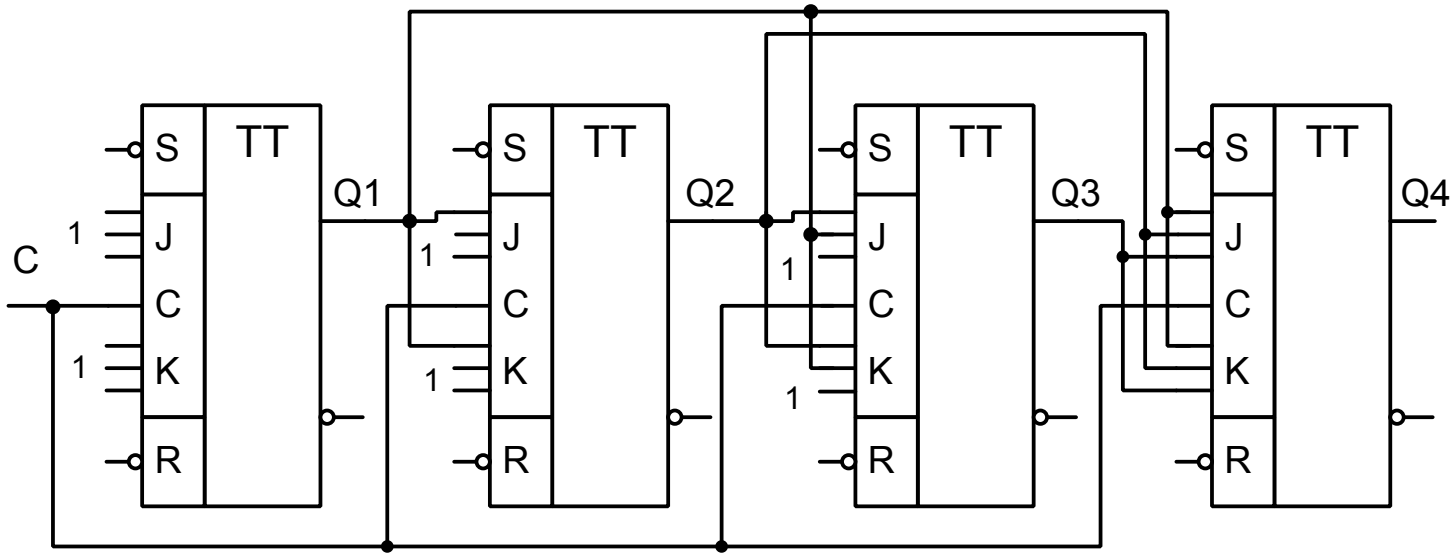
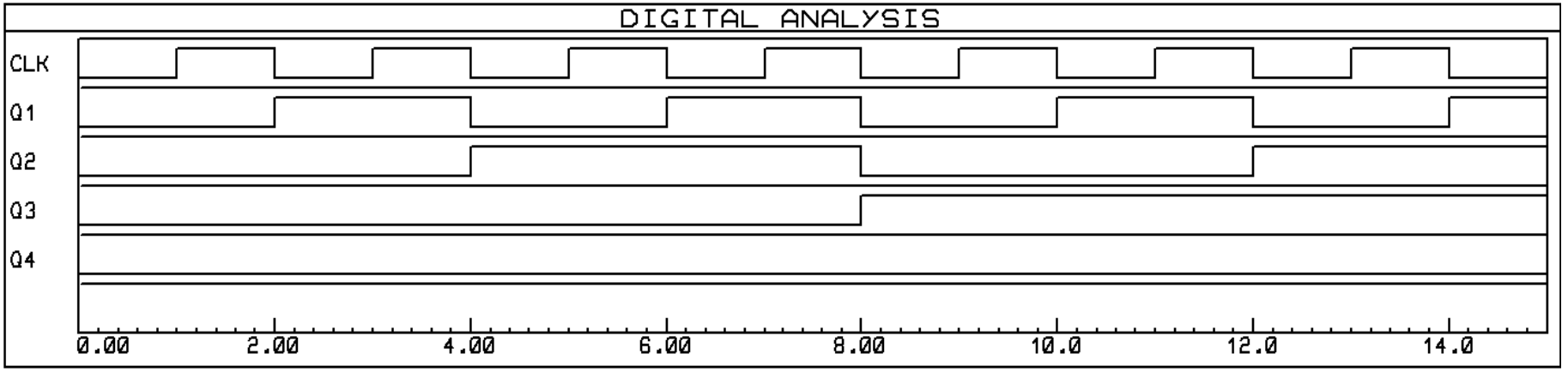


Диаграмма работы

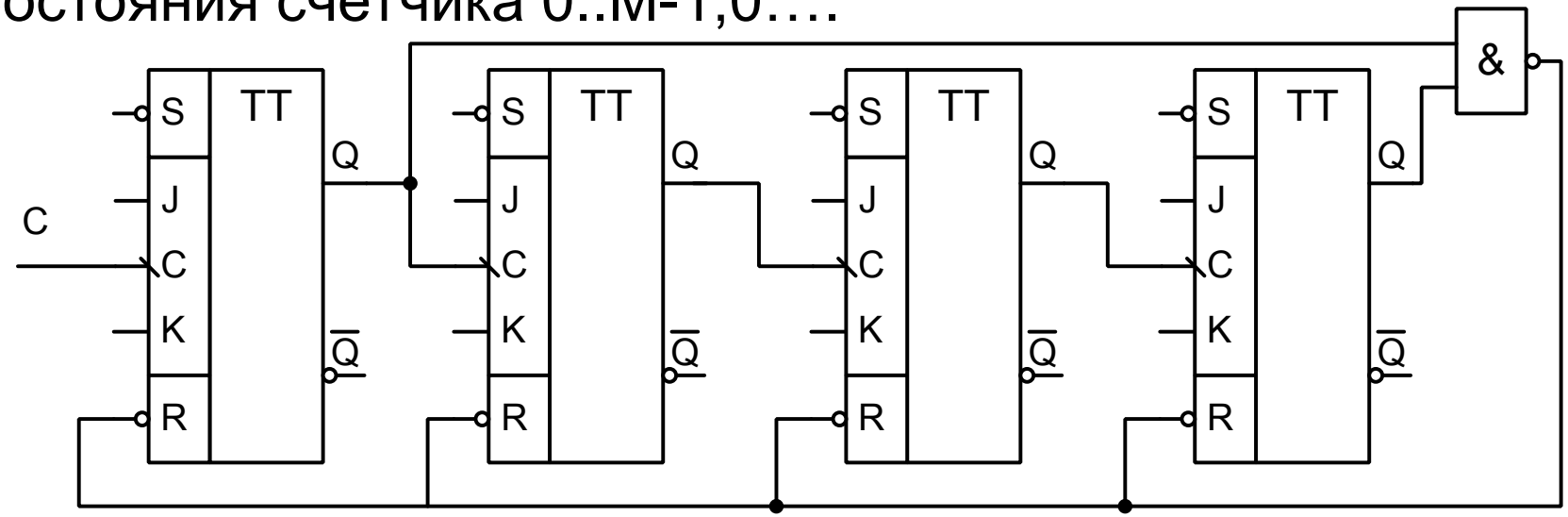


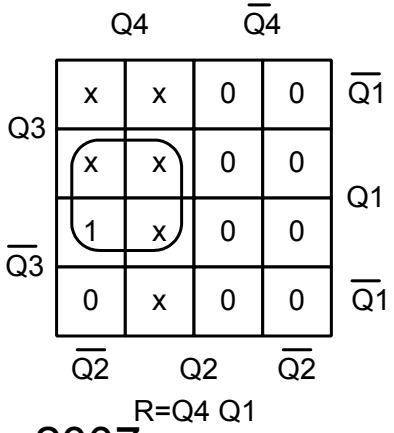
Построение счетчиков с произвольным модулем счета

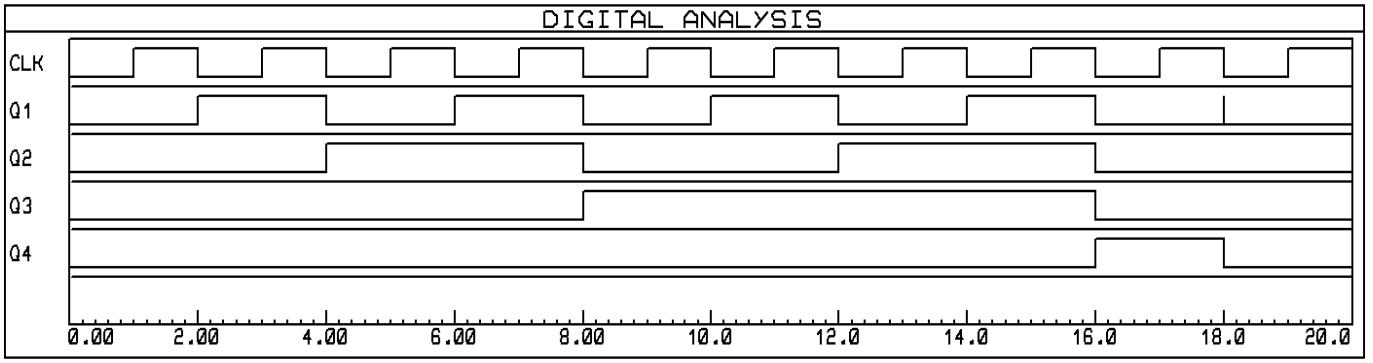
Метод управляемого сброса

Построить счетчик с модулем счета M=9.

Состояния счетчика 0..M-1,0….







Метод модификации связей

Построить счетчик с модулем счета M=7.
Состояния счетчика 0,2,3..7,0,2…(состояние «1» пропущено).

Таблица функционирования счетчика

Q(t)	Q(t+1)	Функции возбуждения					
		J2	K2	J1	K1	J0	K0
000	010	0	x	1	x	0	x
010	011	0	x	x	0	1	x
011	100	1	x	x	1	x	1
100	101	x	0	0	x	1	x
101	110	x	0	1	x	x	1
110	111	x	0	x	0	1	x
111	000	x	1	x	1	x	1

Минимизация функций возбуждения

J2

	Q0	$\overline{Q0}$		
Q1	1	x	x	0
$\overline{Q1}$	x	x	x	0
	$\overline{Q2}$	Q2		$\overline{Q2}$

J2=Q0

J1

	Q0	$\overline{Q0}$		
Q1	x	x	x	x
$\overline{Q1}$	x	1	0	1
	$\overline{Q2}$	Q2		$\overline{Q2}$

J1=Q0 + $\overline{Q2}$

J0

	Q0	$\overline{Q0}$		
Q1	x	x	1	1
$\overline{Q1}$	x	x	1	0
	$\overline{Q2}$	Q2		$\overline{Q2}$

J0=Q1+Q2

K2

	Q0	$\overline{Q0}$		
Q1	x	1	0	x
$\overline{Q1}$	x	0	0	x
	$\overline{Q2}$	Q2		$\overline{Q2}$

K2=Q0 Q1

K1

	Q0	$\overline{Q0}$		
Q1	1	1	0	0
$\overline{Q1}$	x	x	x	x
	$\overline{Q2}$	Q2		$\overline{Q2}$

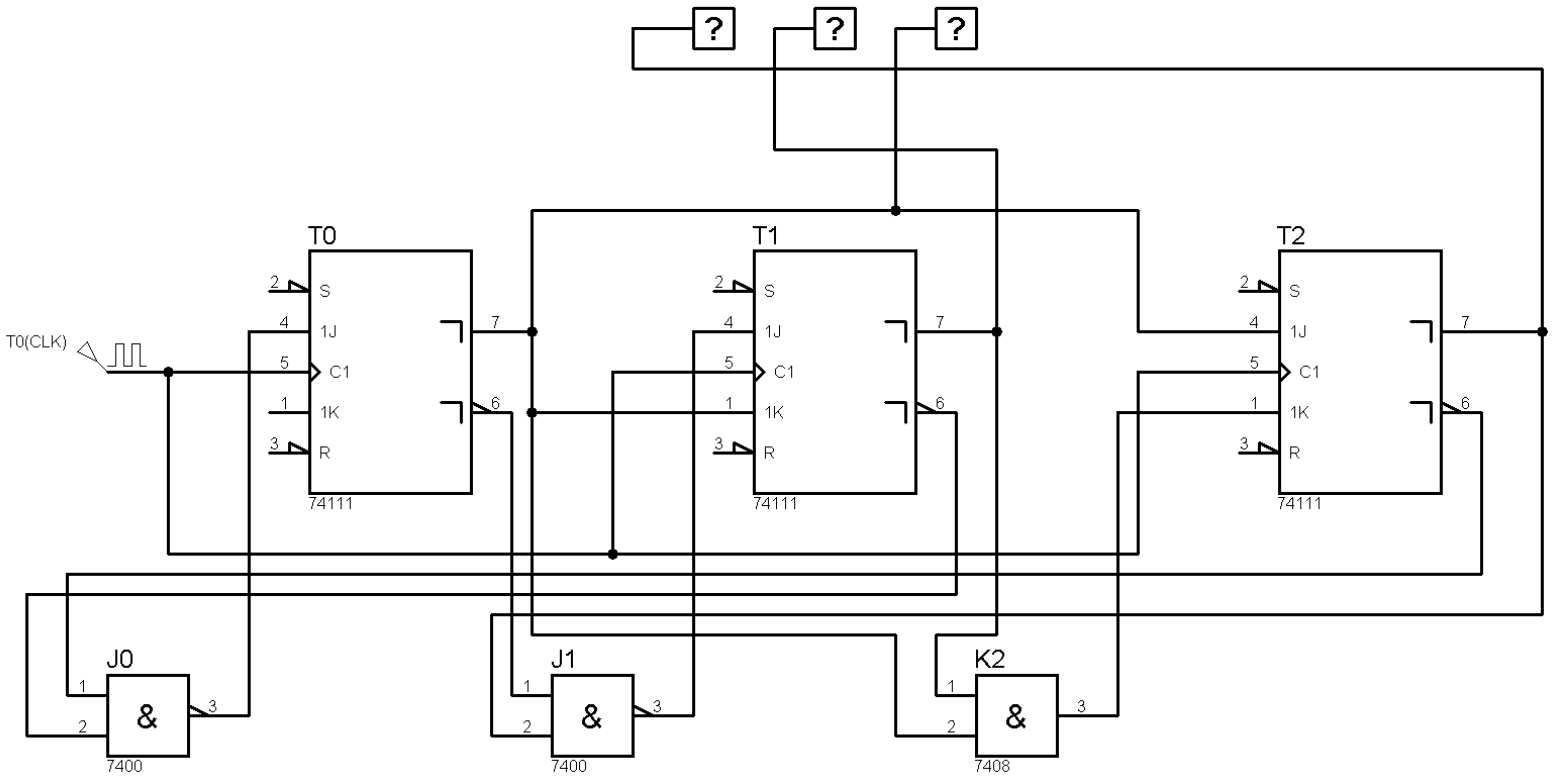
K1=Q0

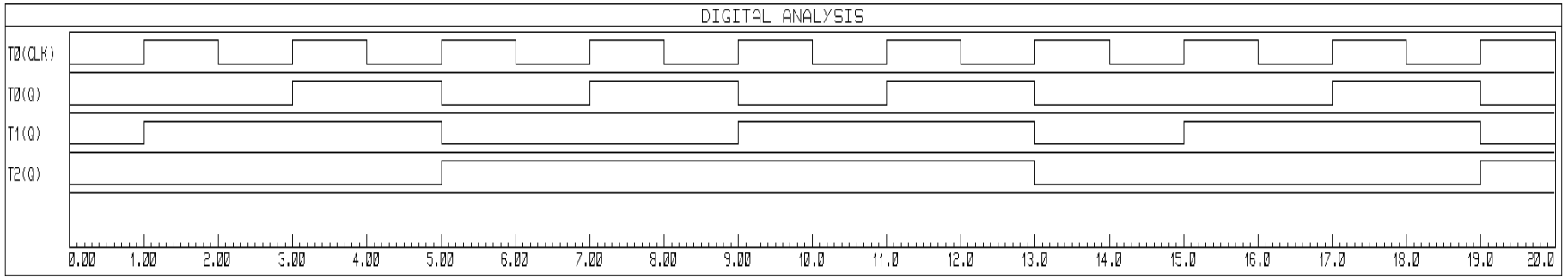
K0

	Q0	$\overline{Q0}$		
Q1	1	1	x	x
$\overline{Q1}$	x	1	x	x
	$\overline{Q2}$	Q2		$\overline{Q2}$

K0=1

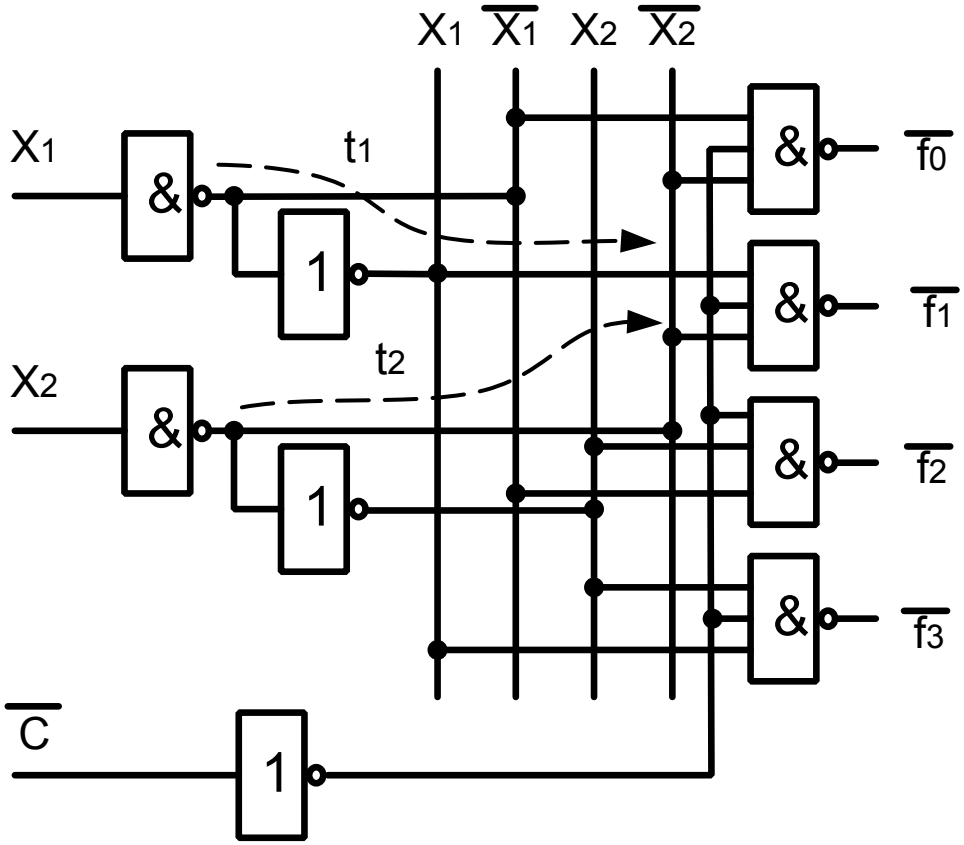
Схема счетчика

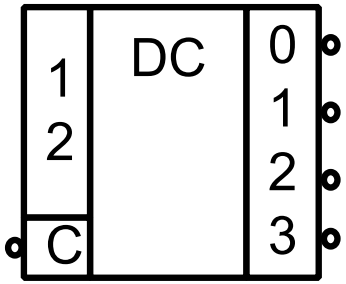




Дешифраторы

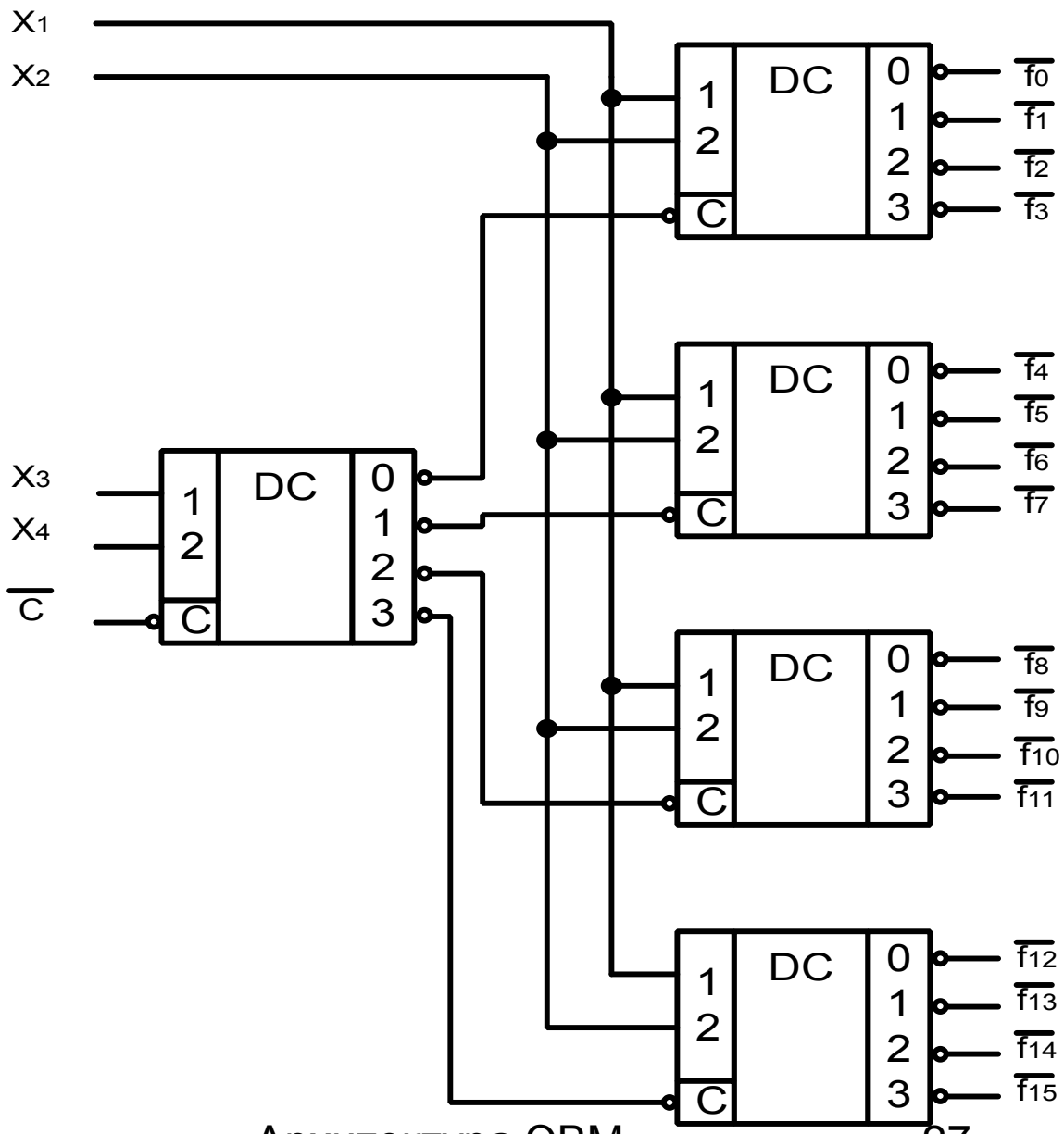
Дешифратором называется комбинационная схема, преобразующая код, подаваемый на входы, в сигнал на одном из выходов.





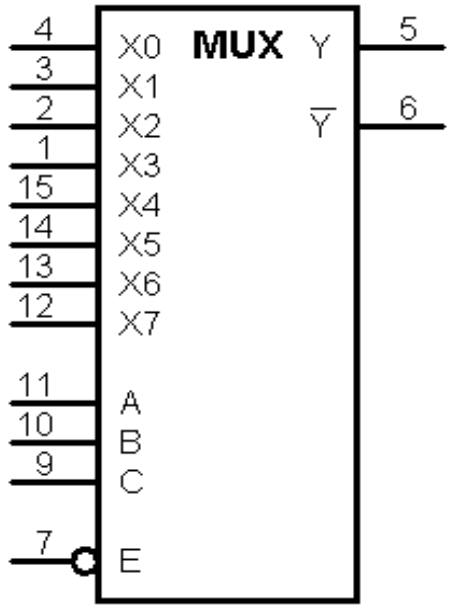
Статический риск: кратковременное изменение сигнала, который должен остаться неизменным.
Динамический риск: многократное переключение элемента вместо ожидаемого однократно.

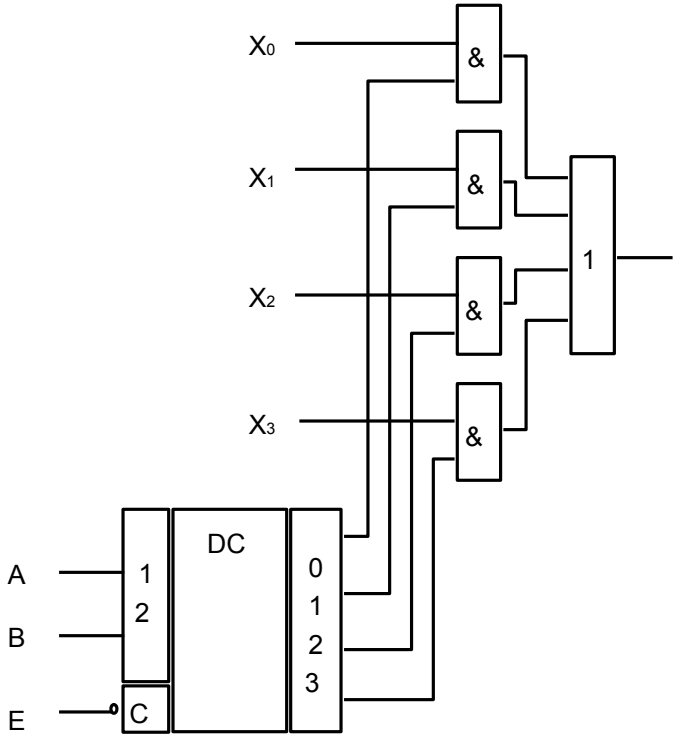
Наращивание размерности дешифраторов



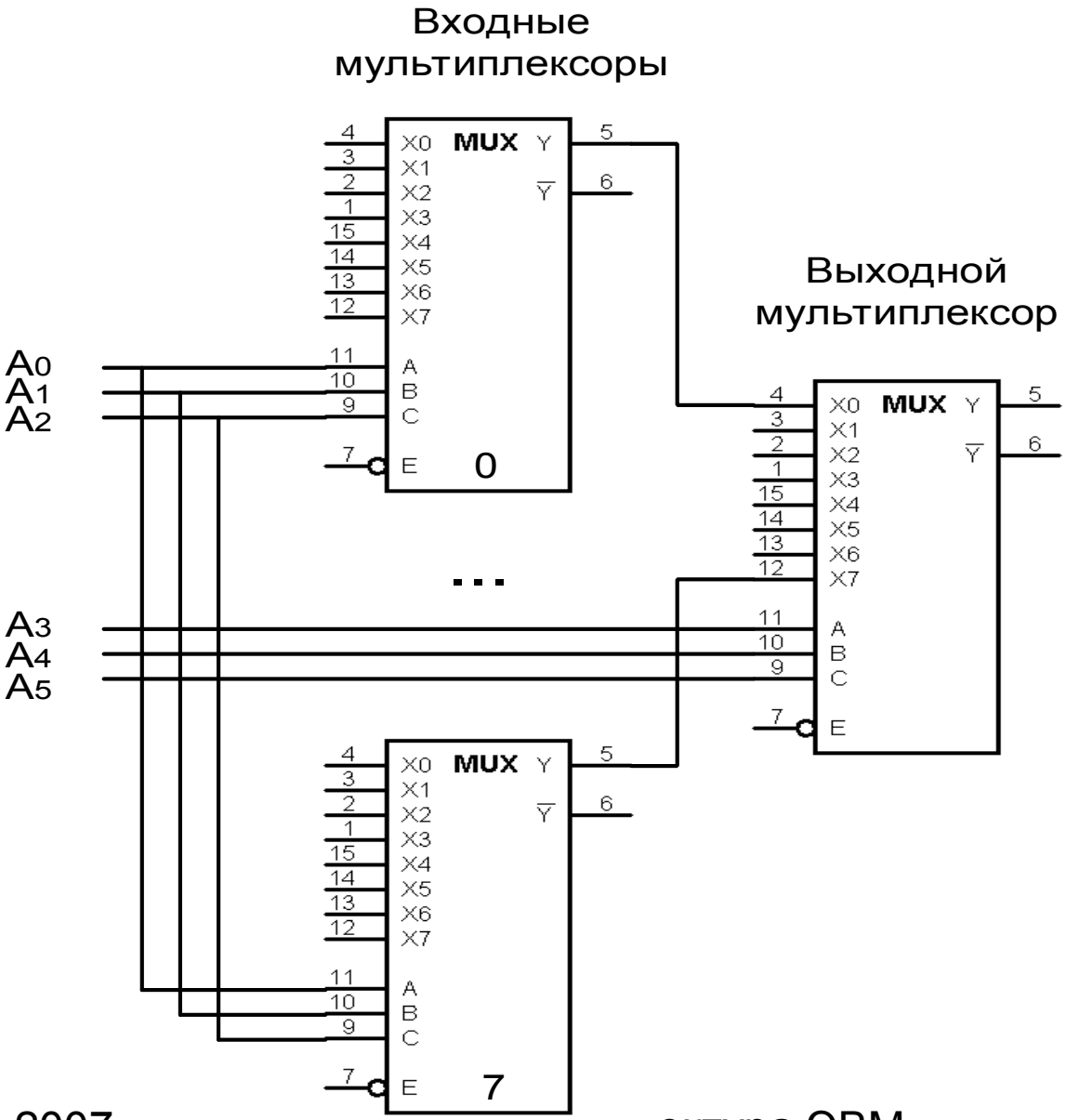
Мультиплексоры

Мультиплексором называется комбинационная схема, осуществляющая передачу сигнала с одной из входных информационных линий на выход.

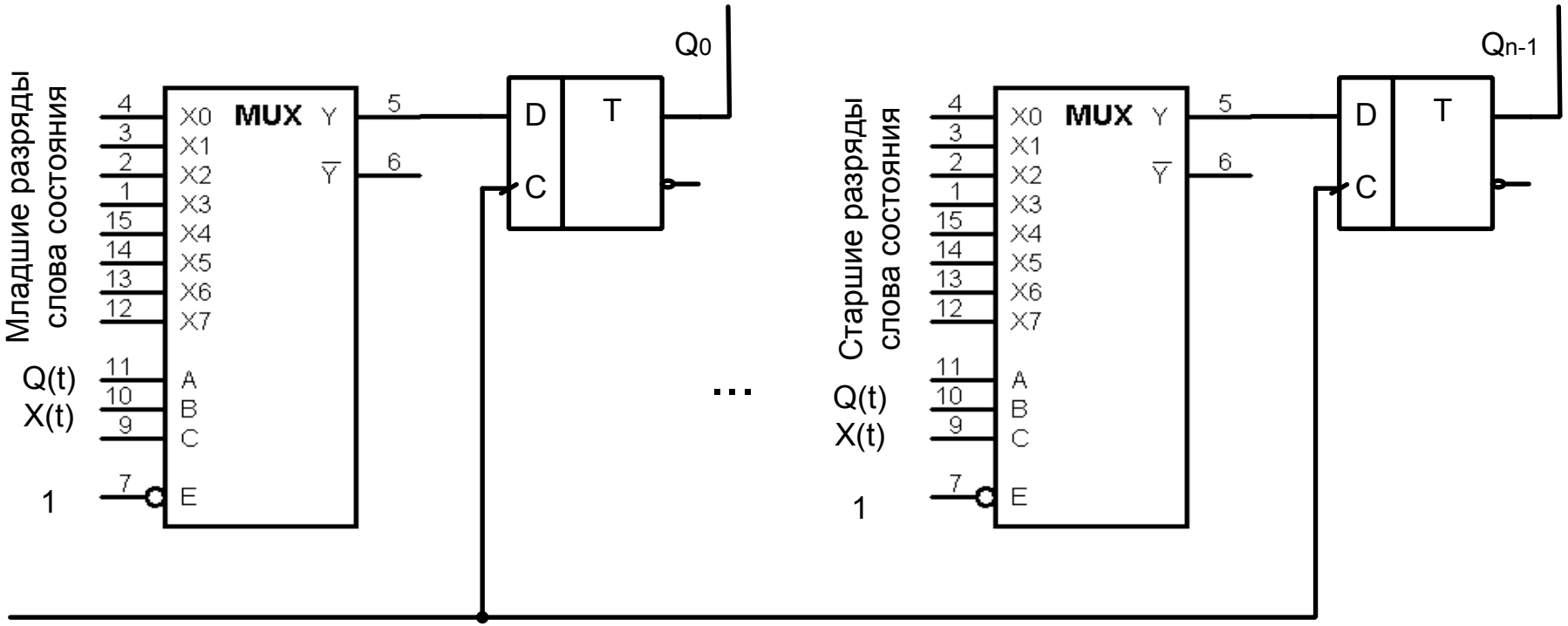




Наращивание размерности мультиплексоров

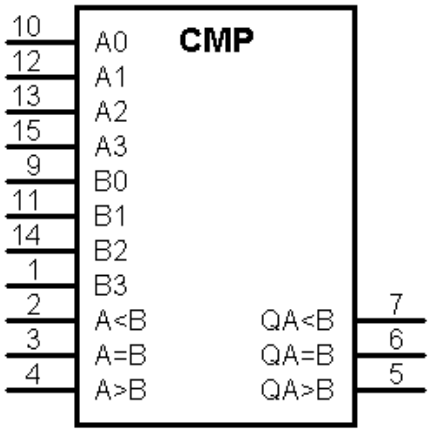


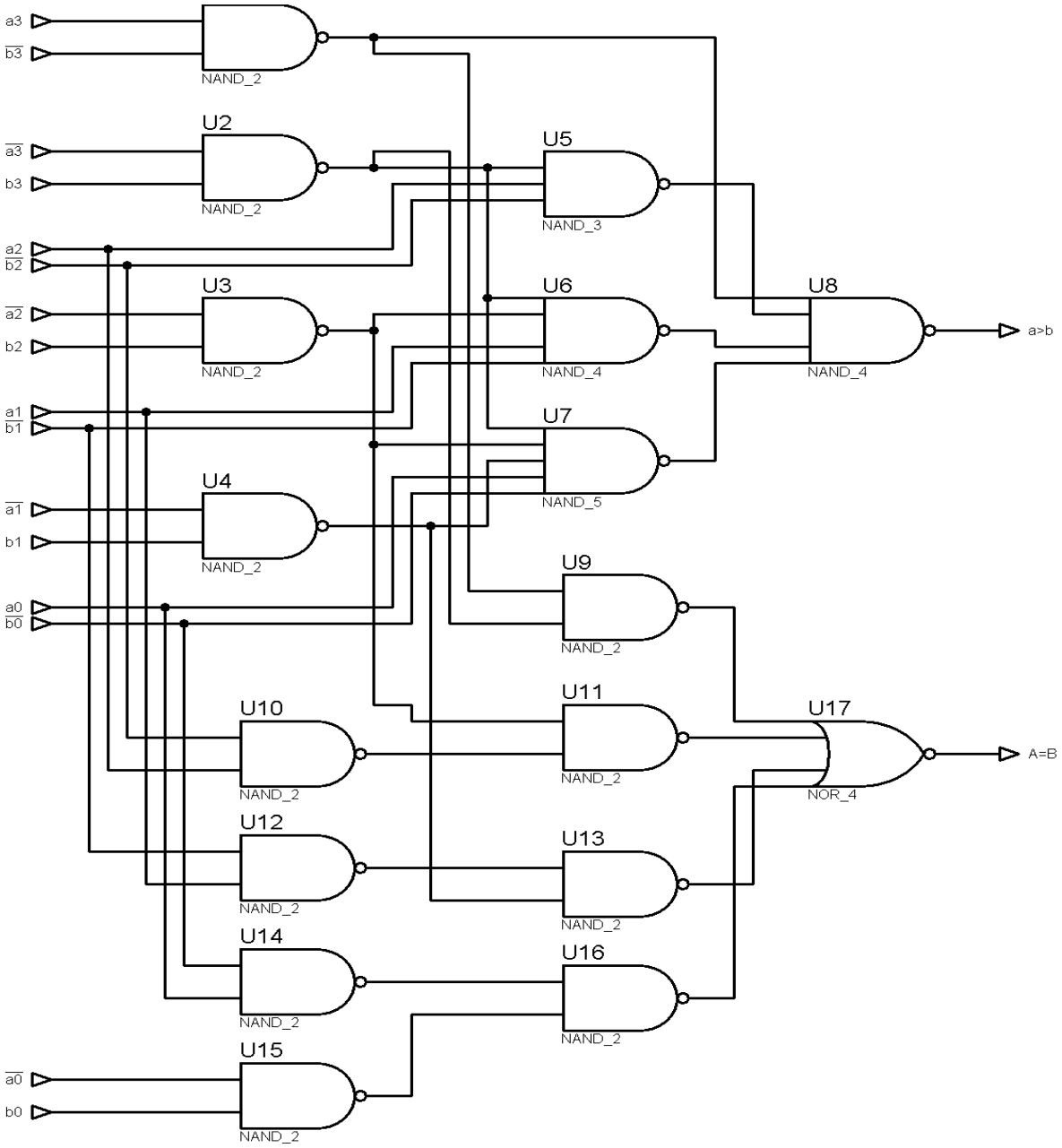
Пример построения автомата с помощью мультиплексоров



Компаратор

Компаратором называется комбинационная схема, определяющая отношение между двумя словами.





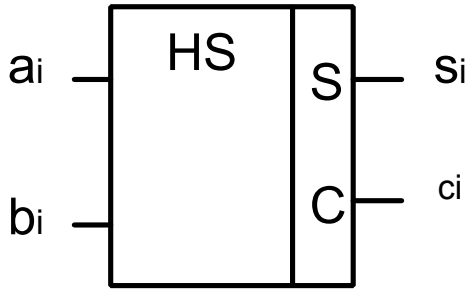
Сумматоры

Сумматором называется узел ЭВМ, выполняющий арифметическое сложение кодов чисел.

$$S_i = a_i \bar{b}_i \bar{c}_{i-1} \cup \bar{a}_i b_i \bar{c}_{i-1} \cup \bar{a}_i \bar{b}_i c_{i-1} \cup a_i b_i c_{i-1}$$

$$c_i = a_i b_i \bar{c}_{i-1} \cup a_i \bar{b}_i c_{i-1} \cup \bar{a}_i b_i c_{i-1} \cup a_i b_i c_{i-1} = a_i b_i \cup a_i c_{i-1} \cup b_i c_{i-1}$$

Полусумматор выполняет арифметическое сложение кодов двух чисел.



Сумматор выполняет арифметическое сложение кодов двух чисел с учетом переноса в младший разряд.

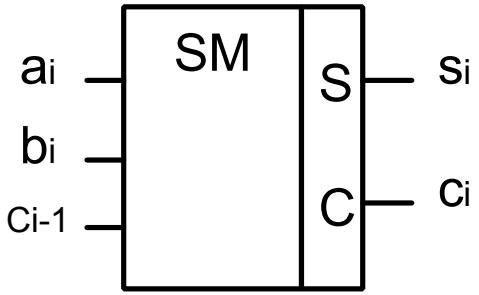


Схема одноразрядного сумматора

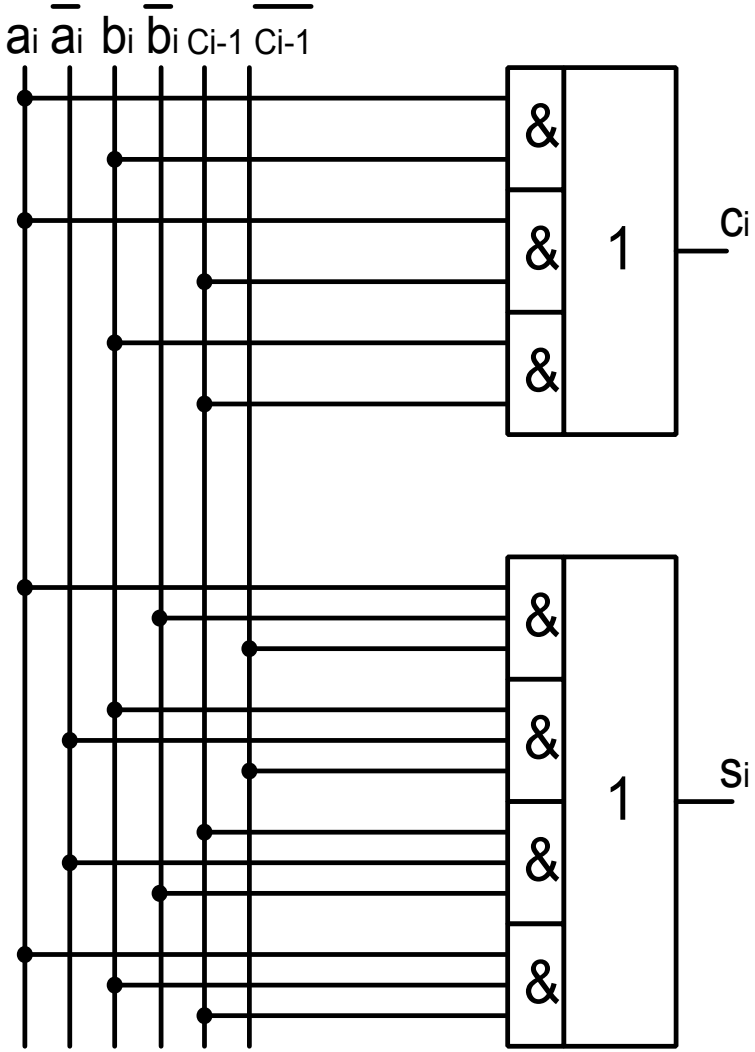


Схема параллельного сумматора с последовательным переносом

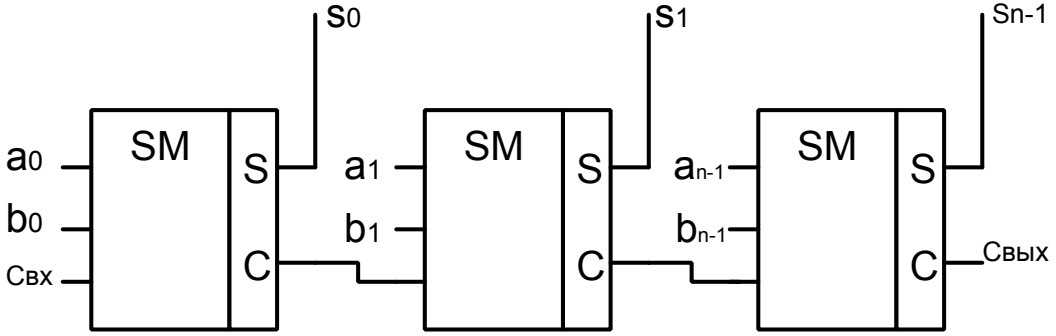


Схема параллельного сумматора с параллельным переносом

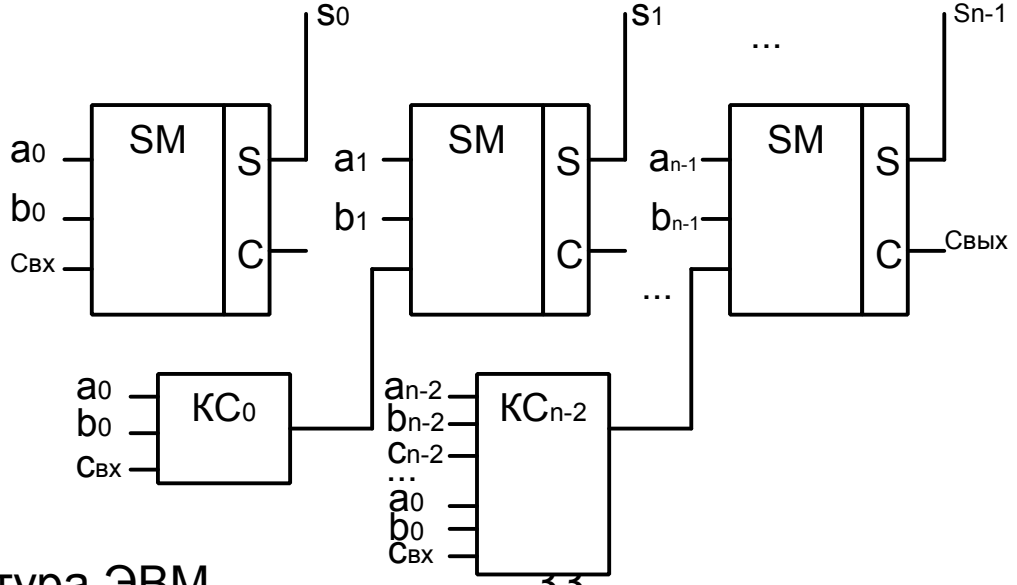
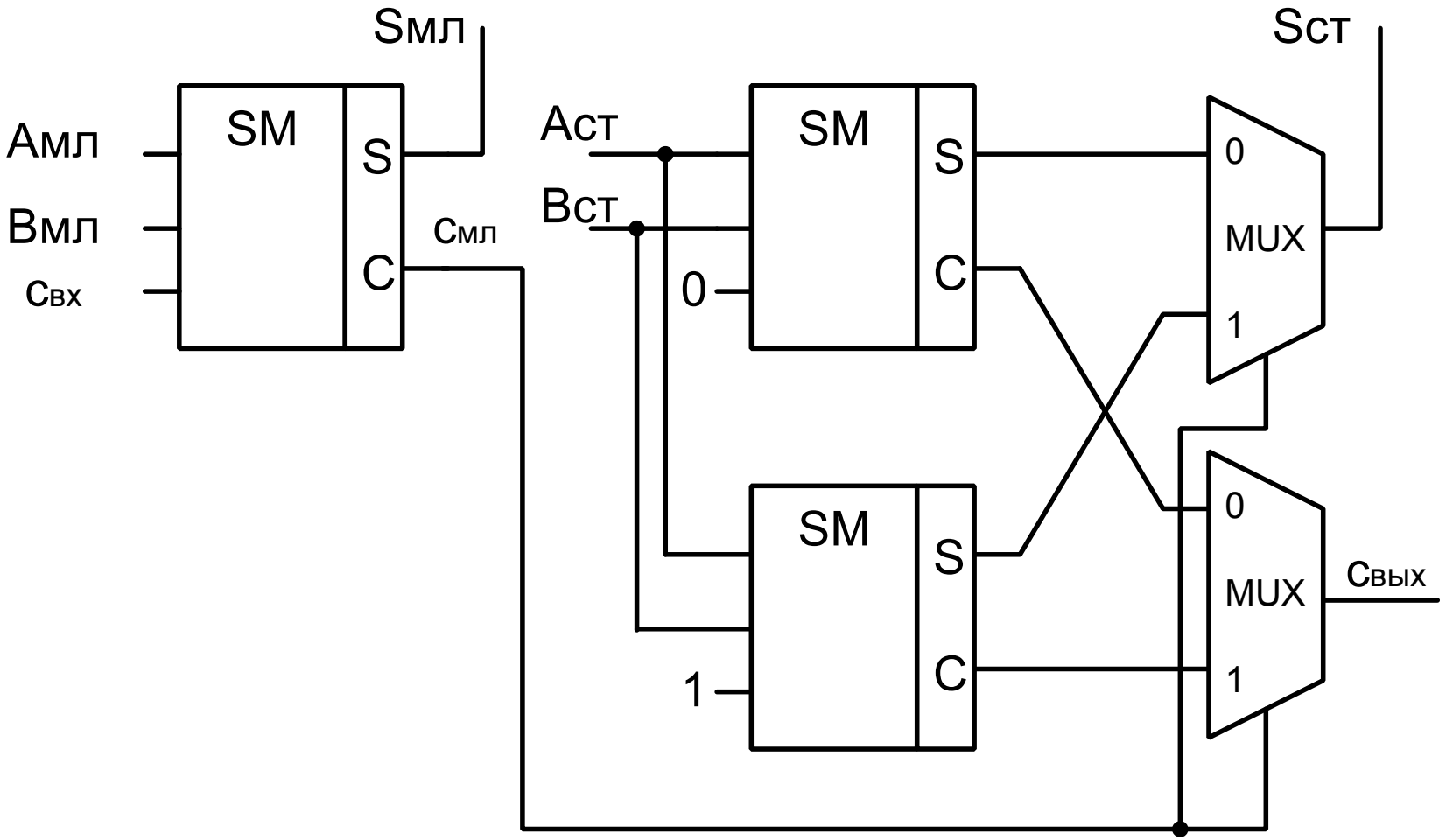
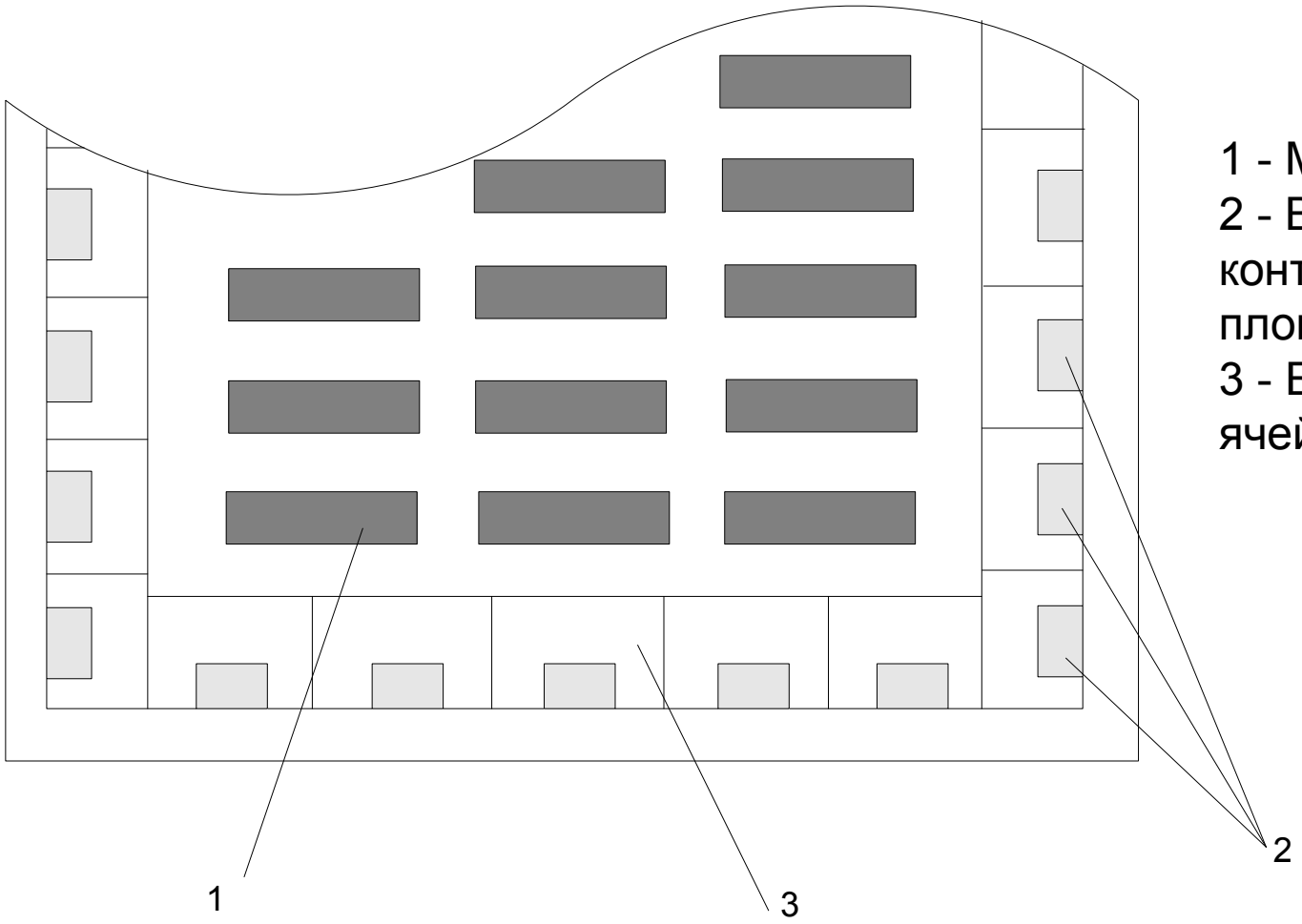


Схема сумматора с условным переносом

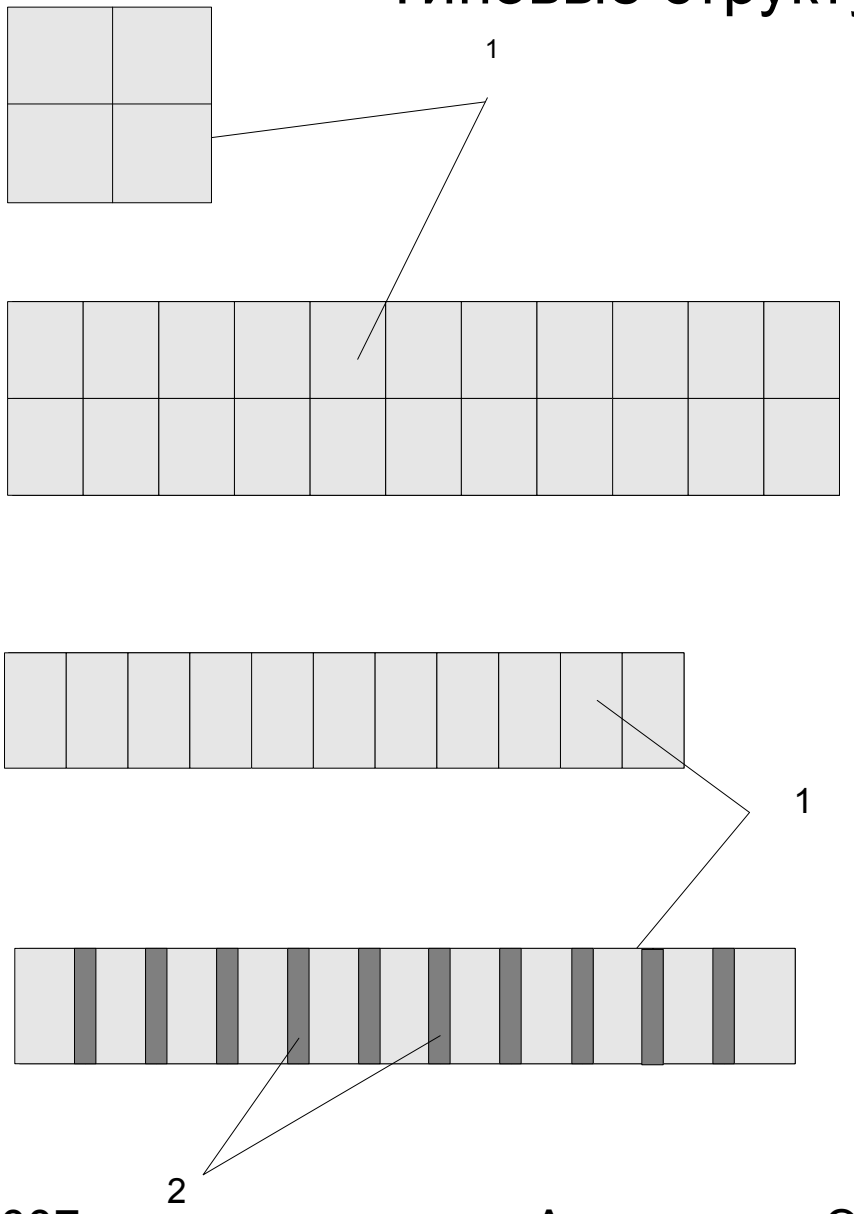


Структура базовых матричных кристаллов



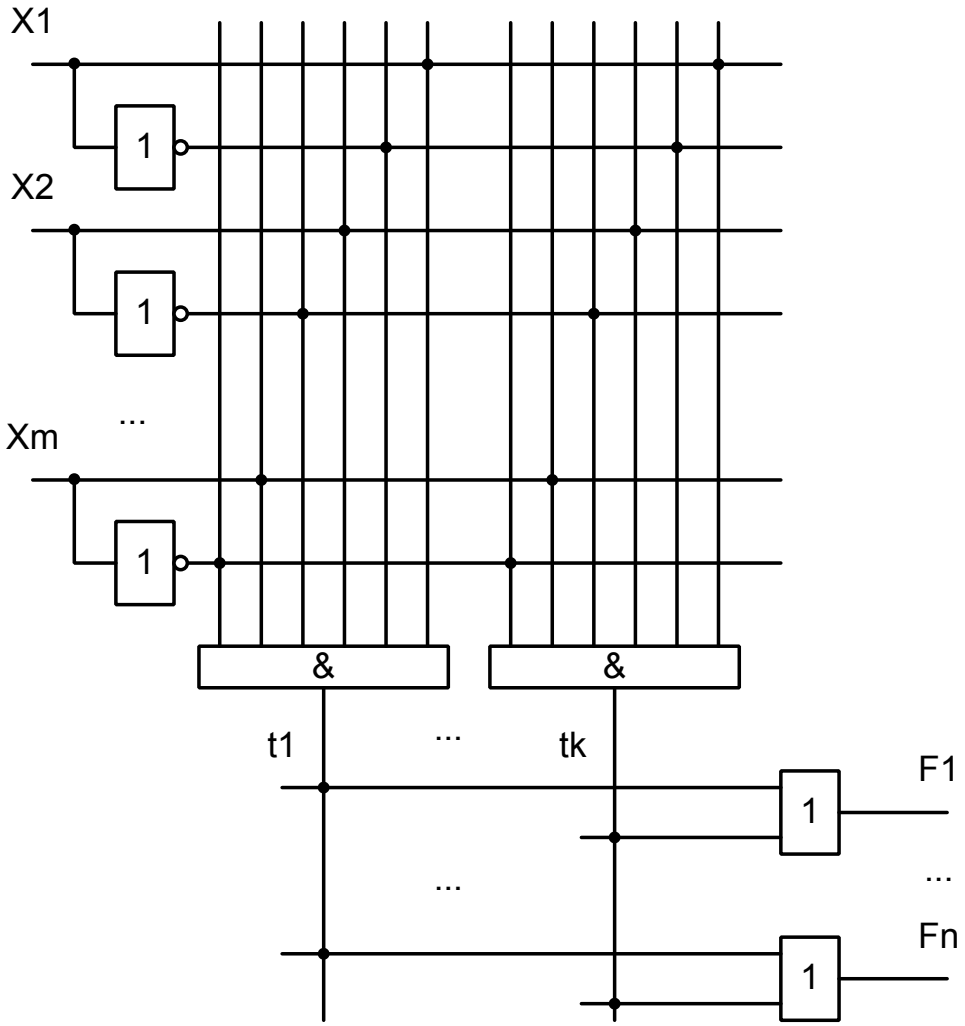
1 - Макроячейки;
2 - Внешние контактные площадки;
3 - Буферные ячейки.

Типовые структуры макроячеек

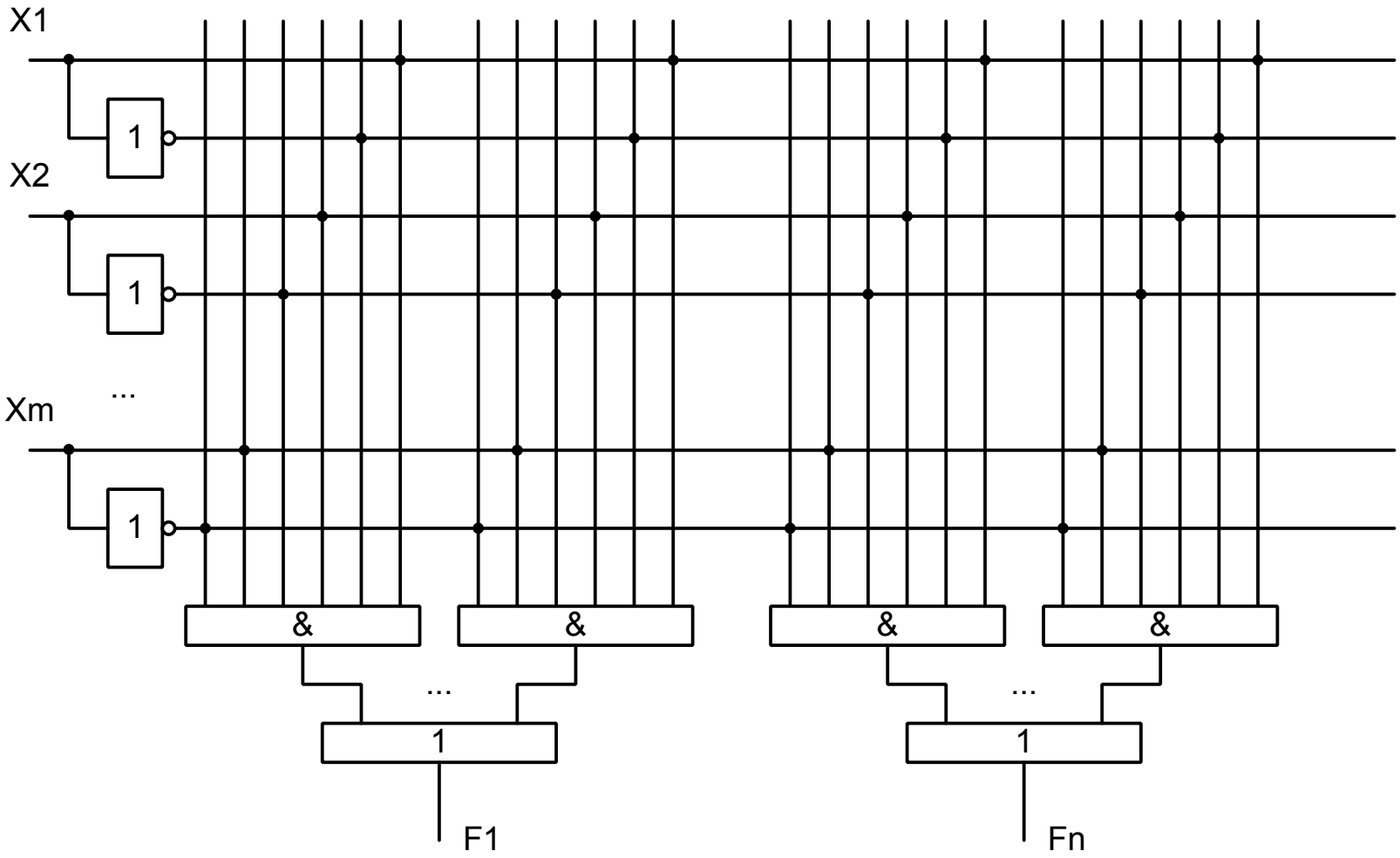


1 - Базовые ячейки (БЯ);
2 - Промежутки между БЯ для прокладки трасс (транзитные соединения).

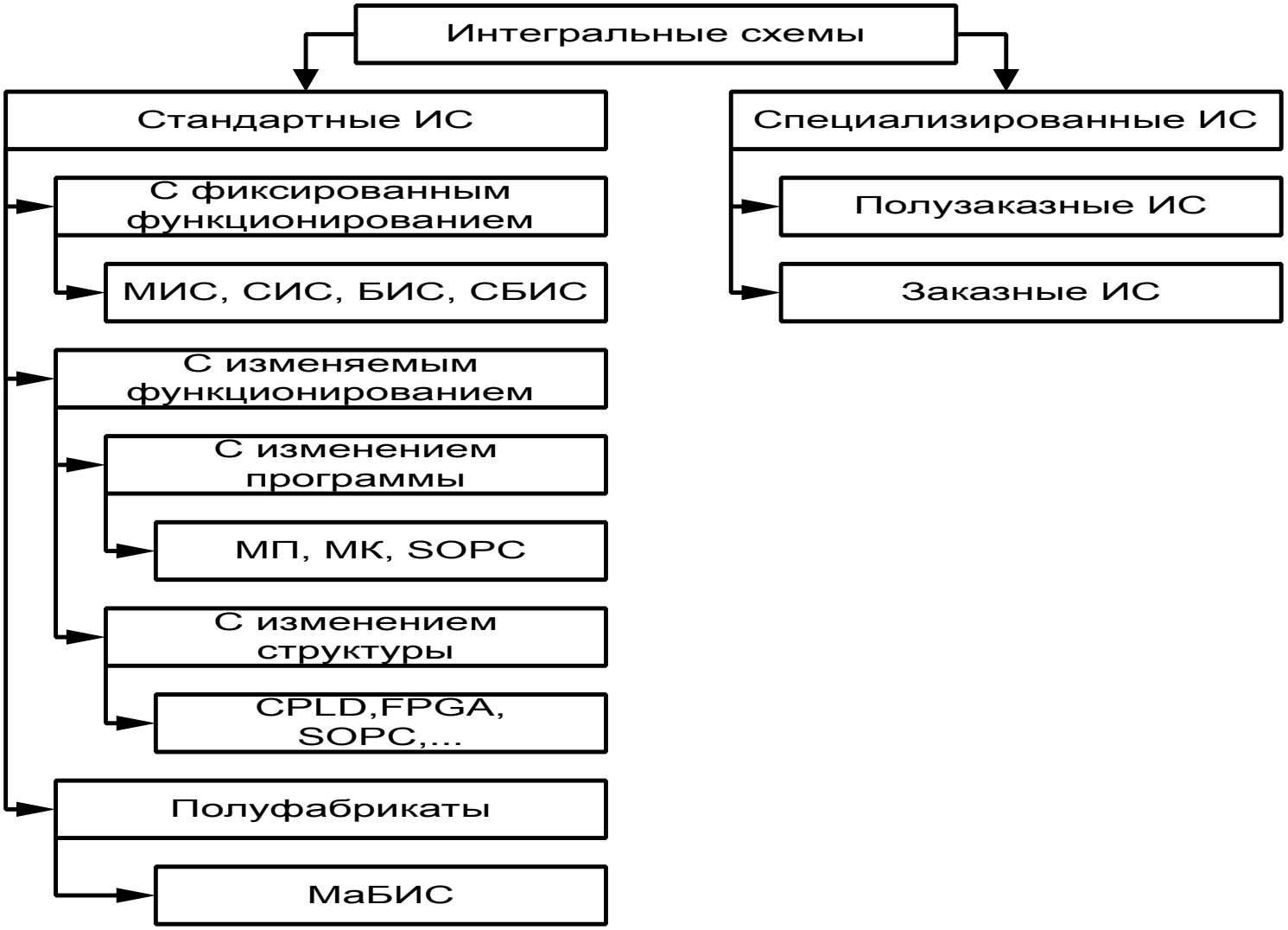
Программируемые логические матрицы



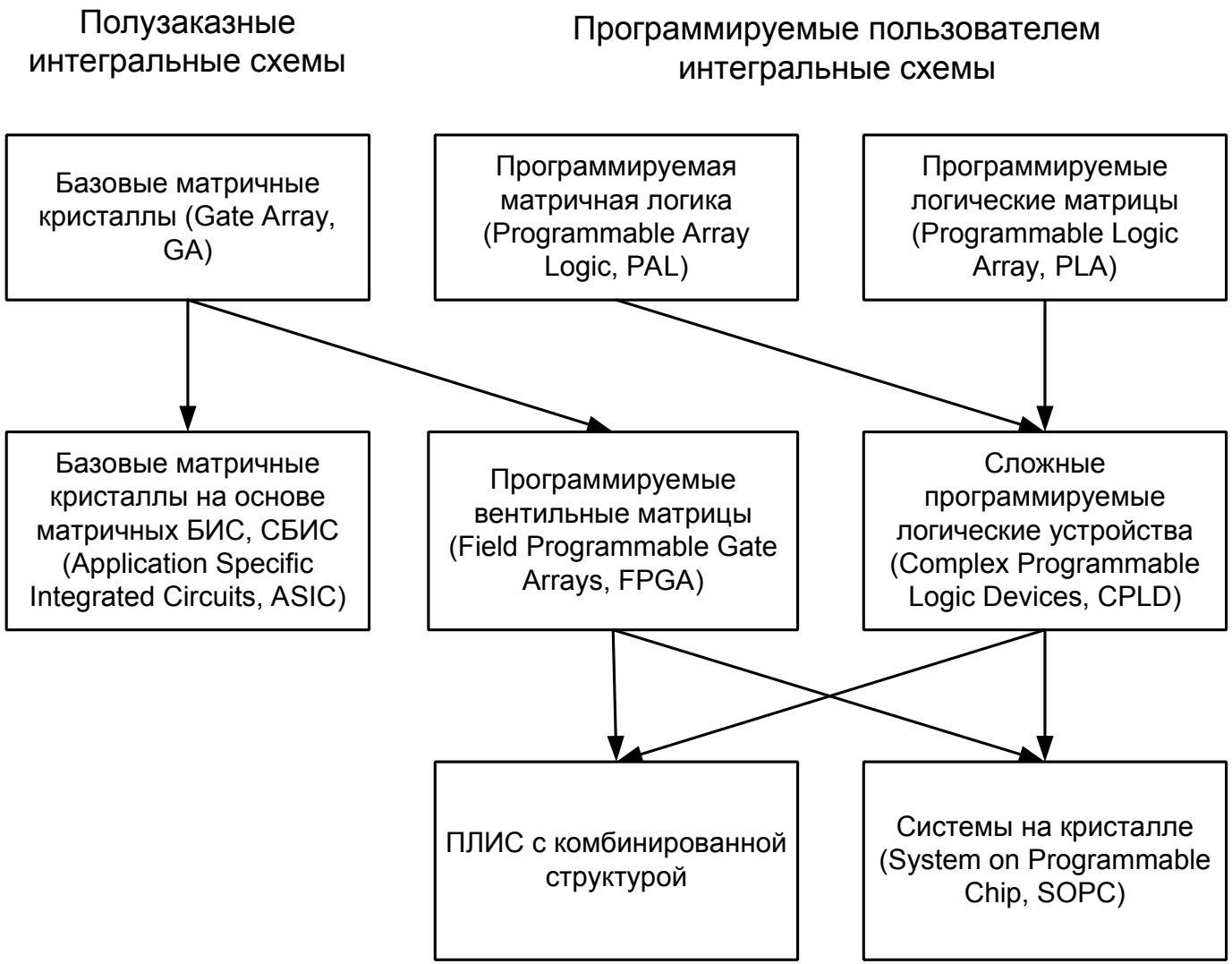
Программируемая матричная логика



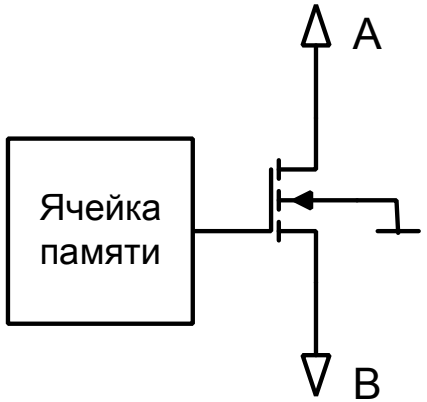
Классификация ИС по способу обеспечения функциональности

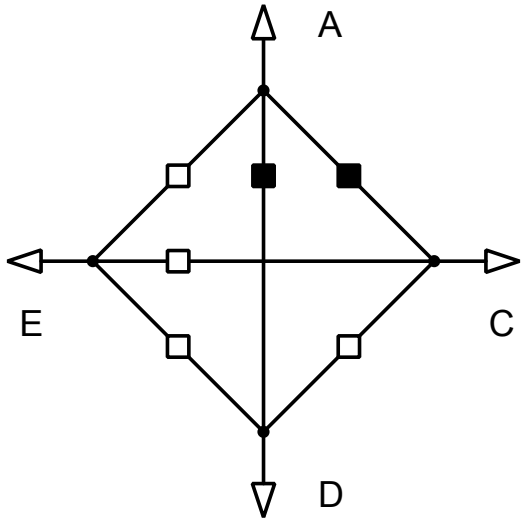


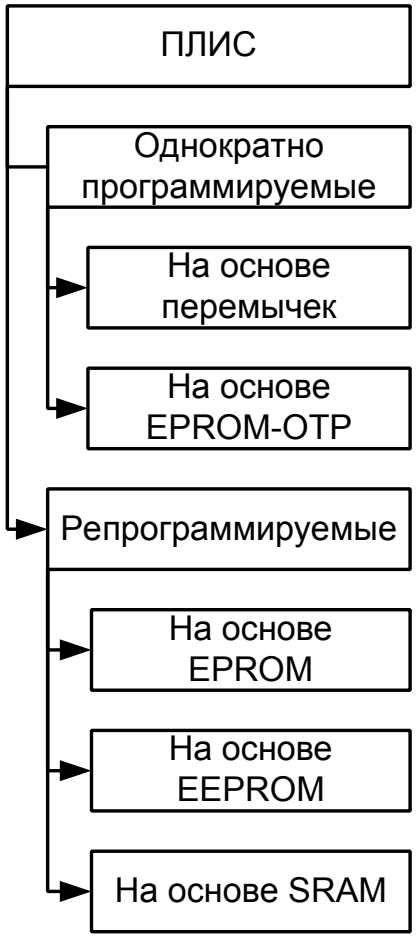
Эволюция ПЛИС



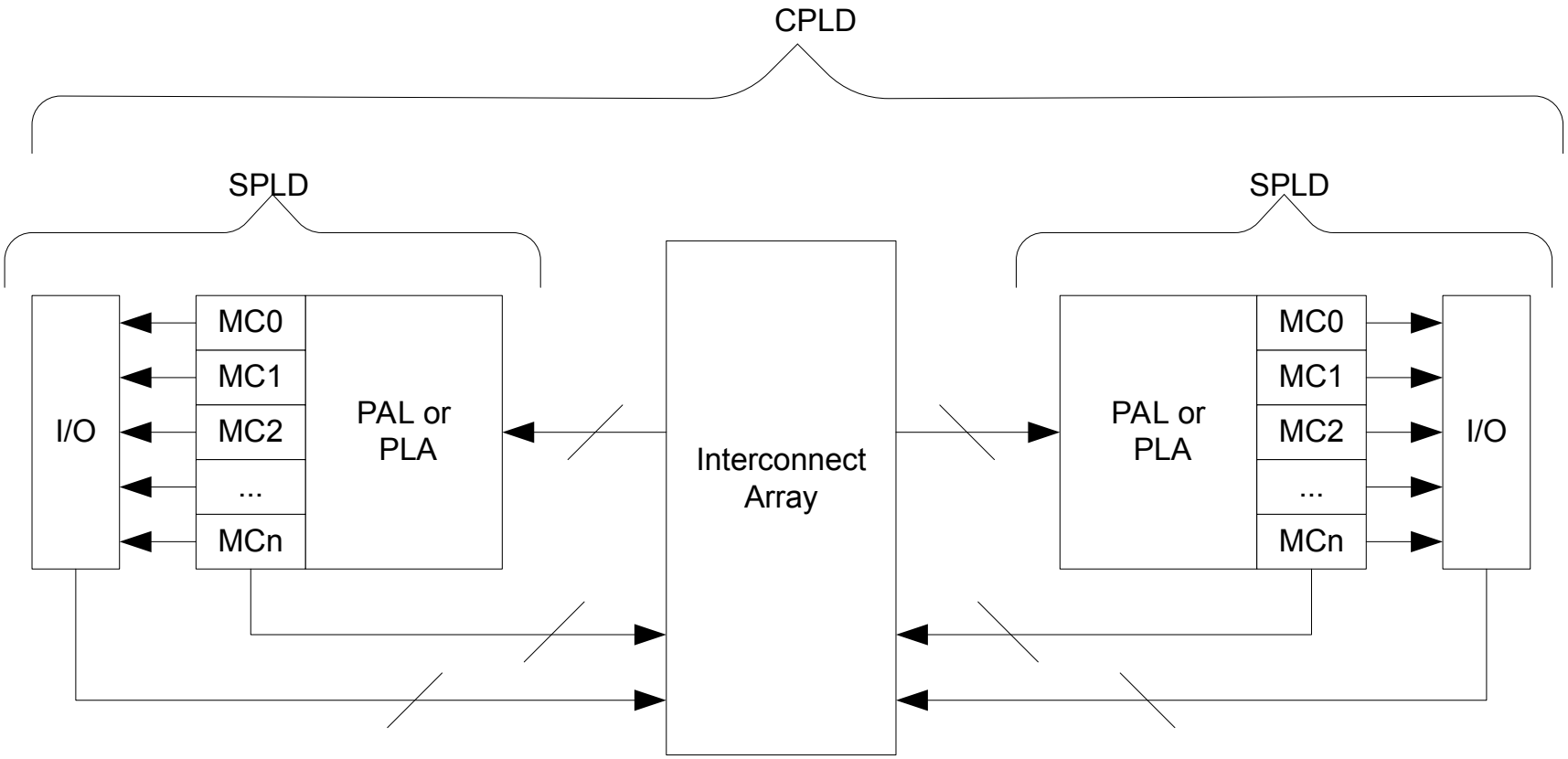
Классификация ПЛИС по типу программируемых связей



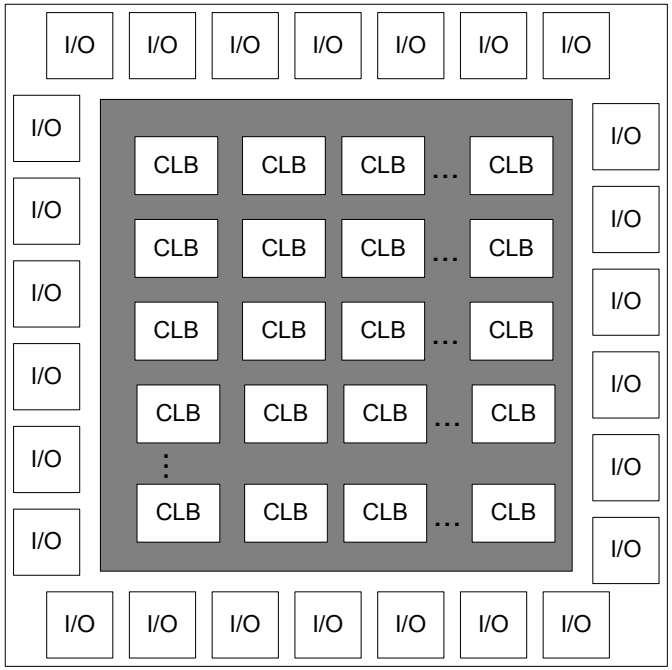




Архитектура сложных программируемых логических устройств (CPLD)



Программируемые вентильные матрицы (FPGA)



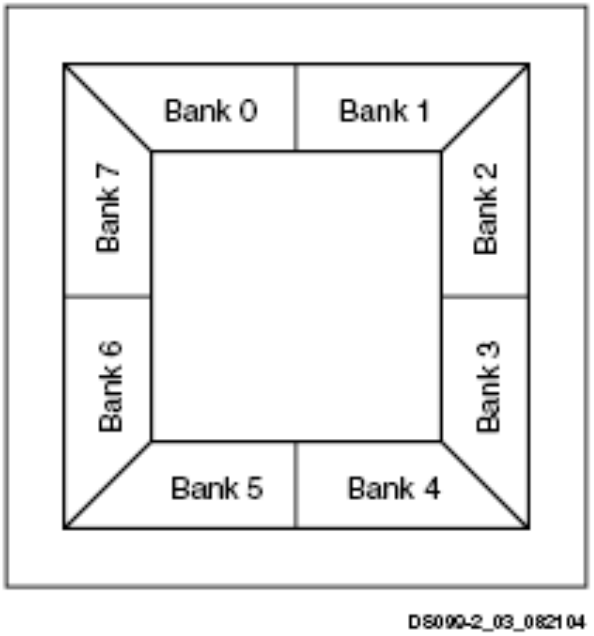
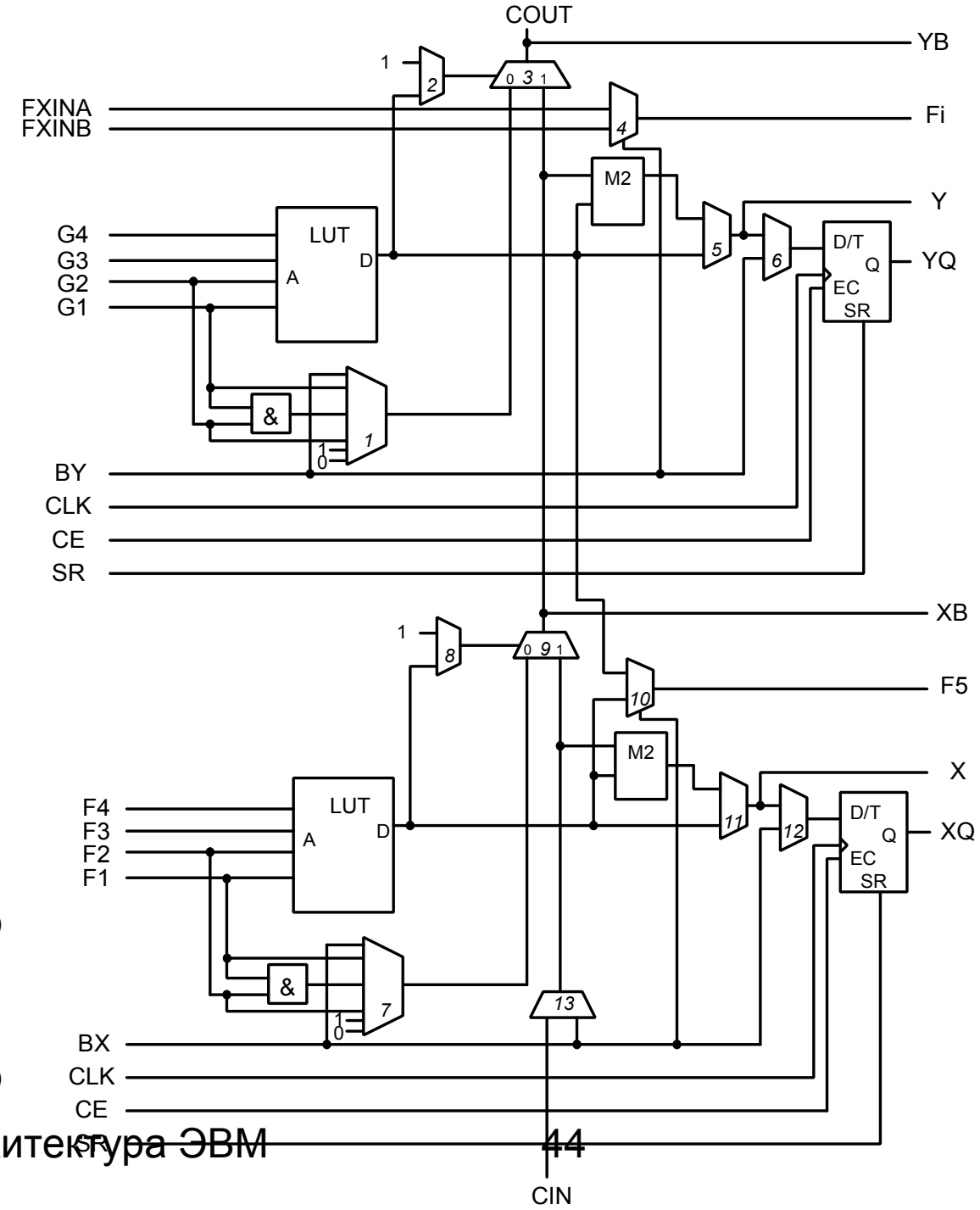


Figure 4: **Spartan-3 I/O Banks (top view)**

Структура блока типа SLICEL

D = Ai xor Bi,
M7 = Ai and Bi
S = D xor CIN

F1	F2	D	M7	CIN	S	COUT
0	0	0	0	0	0	0
0	1	1	0	0	1	0(CIN)
1	0	1	0	0	1	0(CIN)



V.II Основы языка VHDL

(Very high speed integration circuits Hardware Description Language)
Стандарт VHDL-87, Стандарт VHDL-93, Стандарт VHDL-AMS

Язык VHDL используется для:

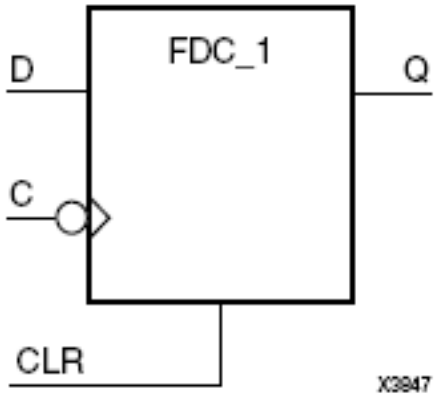
- описания поведения цифровых устройств во времени и при изменении входных воздействий;
- описания структуры цифровых устройств с различной степенью детализации (на системном и блочном уровнях, на уровне регистровых передач, на уровне вентилей);
- моделирования цифровых устройств;
- описания тестовых воздействий при моделировании устройств;
- автоматизации преобразования исходного описания схемы в описание на более низком уровне (вплоть до вентильного уровня).

Стили описания:

- **поведенческий стиль**, при котором для описания проекта используются причинно-следственные связи между событиями на входах устройства и событиями на его выходах (без уточнения структуры);
- **структурный стиль описания**, при котором устройство представляется в виде иерархии взаимосвязанных простых устройств (подобно стилю, принятому в схемотехнике);
- **потоковый стиль описания** устройства основан на использовании логических уравнений, каждое из которых преобразует один или несколько входных информационных потоков в выходные потоки.

Примеры описания устройств на языках VHDL и Verilog

Динамический D-триггер (flip-flop with negative edge clock).
VHDL описание




```
library ieee;
use ieee.std_logic_1164.all;

entity registers_2 is
port(C, D, CLR : in std_logic;
Q : out std_logic);
end registers_2;

architecture archi of registers_2 is
begin
process (C, CLR)
begin
      if (CLR = '1')then
            Q <= '0';
      elsif (C'event and C='0')then
            Q <= D;
      end if;
end process;
end archi;
```

V.III Основы языка Verilog HDL

Verilog был разработан фирмой Gateway Design Automation в 1984 г

Стандарт Verilog LRM (Language Reference Manual), IEEE1364-1995 принят в 1995 году

Verilog и VHDL

•VHDL обладает большей универсальностью и может быть использован не только для описания моделей цифровых электронных схем, но и для других моделей.

•Из-за своих расширенных возможностей VHDL проигрывает в эффективности и простоте, то есть на описание одной и той же конструкции в Verilog потребуется в 3–4 раза меньше символов (ASCII), чем в VHDL.

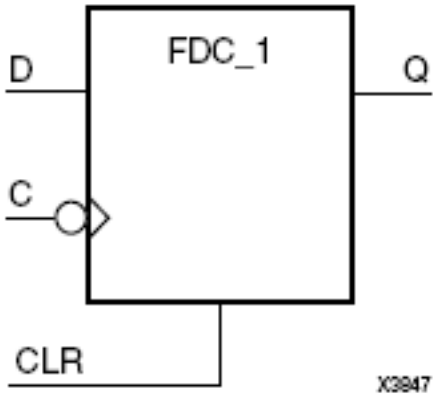
•Как и VHDL, Verilog изначально предназначался для моделирования цифровых систем и как средство описания синтезируемых проектов стал использоваться с 1987 г. В настоящее время ведущие пакеты синтеза систем на ПЛИС, такие как продукты фирм Synopsis, Caddence, Mentor Graphics, многих производителей ПЛИС, поддерживают синтез с описания на языке Verilog.

•Создавать свои типы данных в Verilog нельзя. Основной тип данных для синтезируемых описаний: целое — integer (32-битовое со знаком).

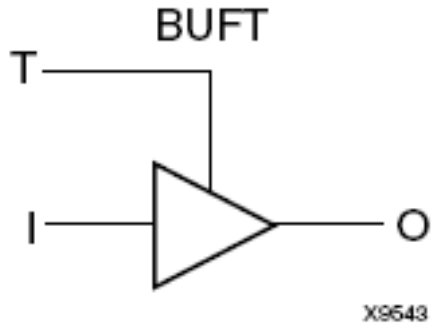
•В Verilog могут быть использованы специфические объекты (UDP, Specify-блоки), не имеющие аналогов в VHDL, а также многочисленные функции PLI (Program Language Interface).

Динамический D-триггер (flip-flop with positive edge clock). Verilog описание

```
module v_registers_2 (C, D, CLR, Q);

input C, D, CLR;
output Q;
reg Q;

always @(negedge C or posedge CLR)
begin
      if (CLR)
            Q <= 1'b0;
      else
            Q <= D;
      end
endmodule
```



Буфер с третьим состоянием (buft). VHDL описание



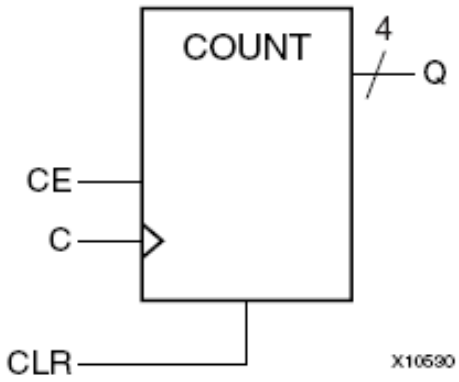

```
entity three_st_1 is
port(T : in std_logic;
I : in std_logic;
O : out std_logic);
end three_st_1;

architecture archi of three_st_1 is
begin
	process (I, T)
	begin
		if (T='0') then
			O <= I;
		else
			O <= 'Z';
		end if;
	end process;
	-- Variant 2
	-- O <= I when (T='0') else 'Z';
end archi;
```

Буфер с третьим состоянием (buft). Verilog описание

```
module v_three_st_1 (T, I, O);

input T, I;
output O;
reg O;

always @(T or I)
begin
	if (~T)
		O = I;
	else
		O = 1'bZ;
end

//Variant 2
assign O = (~T) ? I: 1'bZ;

endmodule
```

Счетчик с разрешением счета (Counter). VHDL описание

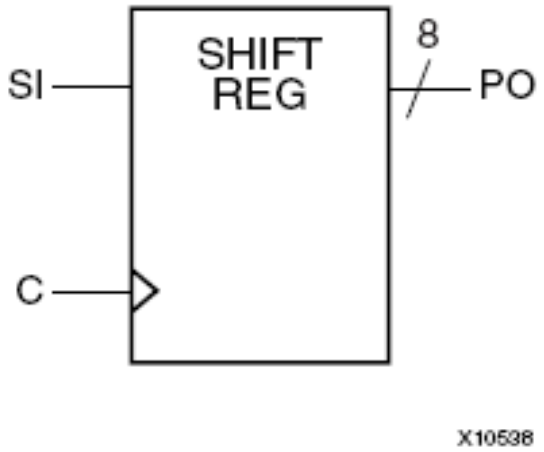
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counters_5 is
port(C, CLR, CE : in std_logic;
Q : out std_logic_vector(3 downto 0));
end counters_5;

architecture archi of counters_5 is
signal tmp: std_logic_vector(3 downto 0);
begin
        process (C, CLR)
        begin
                if (CLR='1') then
                        tmp <= "0000";
                elsif (C'event and C='1') then
                        if (CE='1') then
                                tmp <= tmp + 1;
                        end if;
                end if;
        end process;
        Q <= tmp;
end archi;
```



Счетчик с разрешением счета (Counter). Verilog описание

```
module v_counters_5 (C, CLR, CE, Q);

input C, CLR, CE;
output [3:0] Q;
reg [3:0] tmp;

always @(posedge C or posedge CLR)
begin
      if (CLR)
            tmp <= 4'b0000;
      else if (CE)
            tmp <= tmp + 1'b1;
      end
      assign Q = tmp;
endmodule
```

Сдвиговый регистр с последовательной загрузкой (Shift register with serial in and parallel out).
VHDL описание



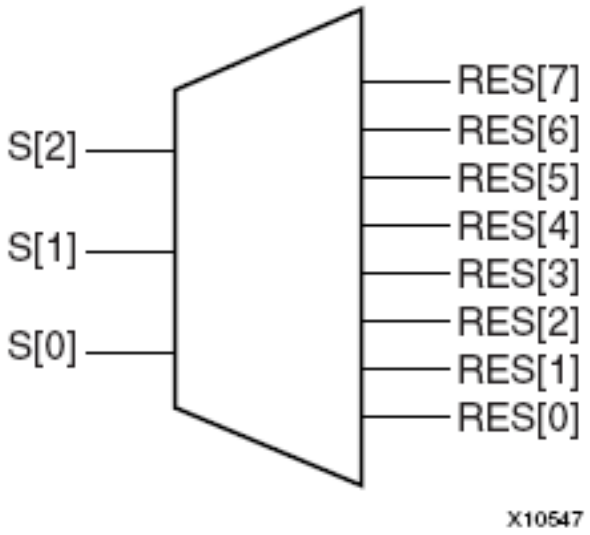

```
library ieee;
use ieee.std_logic_1164.all;

entity shift_registers_5 is
port(C, SI : in std_logic;
PO : out std_logic_vector(7 downto 0));
end shift_registers_5;

architecture archi of shift_registers_5 is
signal tmp: std_logic_vector(7 downto 0);
begin
      process (C)
      begin
            if (C'event and C='1') then
                  tmp <= tmp(6 downto 0)& SI;
            end if;
      end process;
      PO <= tmp;
end archi;
```

Сдвиговый регистр с последовательной загрузкой (Shift register with serial in and parallel out). Verilog описание

```
module v_shift_registers_5 (C, SI, PO);
input C,SI;
output [7:0] PO;
reg [7:0] tmp;

always @(posedge C)
      tmp <= {tmp[6:0], SI};

assign PO = tmp;

endmodule
```

Дешифратор (Decoder). VHDL описание

```
library ieee;
use ieee.std_logic_1164.all;

entity decoders_1 is
port (sel: in std_logic_vector (2 downto 0);
res: out std_logic_vector (7 downto 0));
end decoders_1;

architecture archi of decoders_1 is
begin
        res <= "00000001" when sel = "000" else
           "00000010" when sel = "001" else
           "00000100" when sel = "010" else
           "00001000" when sel = "011" else
           "00010000" when sel = "100" else
           "00100000" when sel = "101" else
           "01000000" when sel = "110" else
           "10000000";
end archi;
```



Дешифратор (Decoder). Verilog описание

```
module v_decoders_1 (sel, res);

input [2:0] sel;
output [7:0] res;
reg [7:0] res;

always @(sel or res)
begin
      case (sel)
            3'b000 : res = 8'b00000001;
            3'b001 : res = 8'b00000010;
            3'b010 : res = 8'b00000100;
            3'b011 : res = 8'b00001000;
            3'b100 : res = 8'b00010000;
            3'b101 : res = 8'b00100000;
            3'b110 : res = 8'b01000000;
            default : res = 8'b10000000;
      endcase
end
endmodule
```

Архитектура ПЛИС типа SOPC

Варианты реализации библиотечных блоков:

Soft - ядра.
Firm - ядра.
Hard – ядра.

Назначение ядер:

Память (ОЗУ, FIFO, кэш- память, …).
АЛУ (умножители, …).
Интерфейсная логика (JTAG, PCI, SPI, UART, …).

МП и МК.