АППАРАТНЫЕ СРЕДСТВА ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ

Часть 2

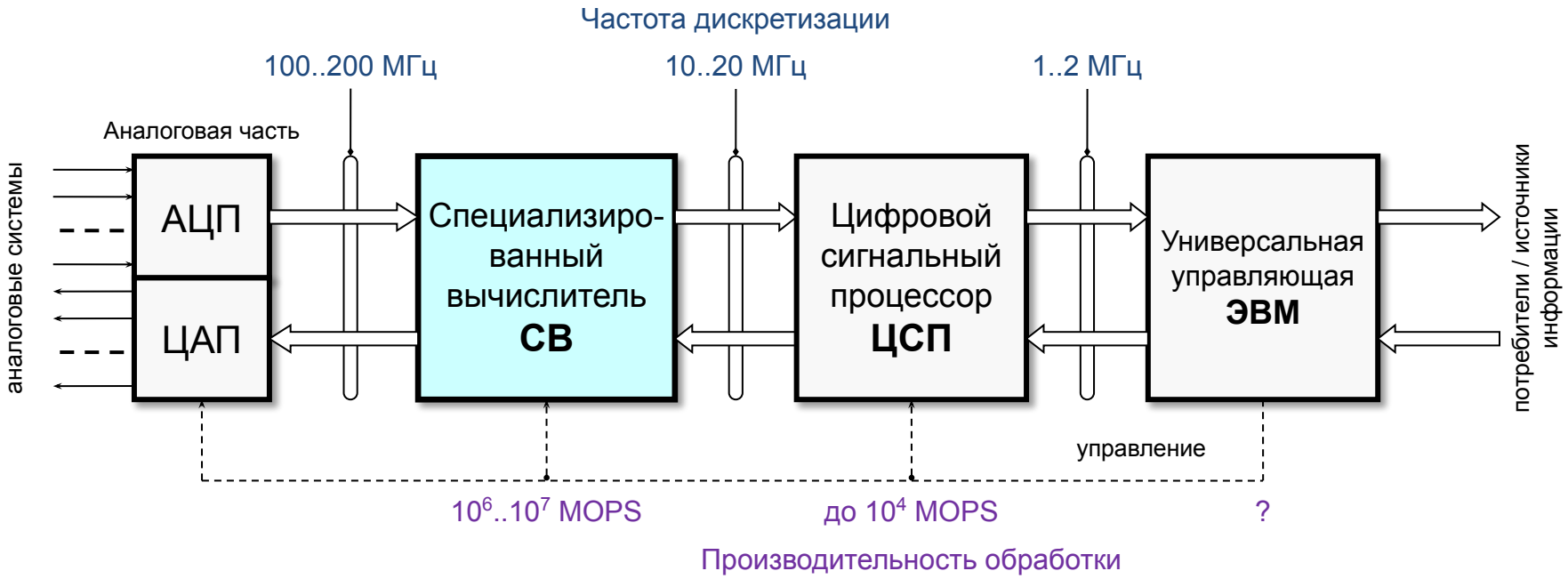
Литература

1. Справочник по радиоэлектронным системам. В 2-х томах. Т.1. Захаров В.Н., Кривицкий Б.Х., Мамаев Н.С. и др.; Под ред. Б.Х. Кривицкого – М.: Энергия, 1979 –352 с., ил.
2. Miles Murdocca PRINCIPLES OF COMPUTER ARCHITECTURE *An Integrated Approach* http://www.cs.rutgers.edu/~murdocca/
3. Сверхбольшие интегральные схемы и современная обработка сигналов: Пер. с англ./Под ред. С. Гуна, Х. Уайтхауса, Т. Кайлата. – М.: Радио и связь, 1989.– 472 с.
4. Труды института инженеров по электротехнике и радиоэлектронике. Тематический выпуск ЦИФРОВАЯ ОБРАБОТКА СИГНАЛОВ. ТИИЭР т. 63, №4 апрель 1975. – М.: Мир, 1975 –195 с.
5. Steven W. Smith The Scientist and Engineer's Guide to Digital Signal Processing, California Technical Publishing, San Diego, California. 1999.
6. Оппенгейм А.В., Шафер Р.В. Цифровая обработка сигналов: Пер. с англ./ Под ред. С.Я.Шаца.– М.: Связь, 1979.– 416с.
7. Хемминг Р.В. Цифровые фильтры: Пер. с англ./ Под ред. А.М.Трахтмана. – М.: Сов. радио, 1980.– 224 с., ил.
8. Сергиенко А.Б. Цифровая обработка сигналов: – СПб.: Питер, 2002.–608с.:ил.
9. Стешенко В. Школа разработки аппаратуры цифровой обработки сигналов на ПЛИС. Chip News,1999, №8–10, 2000, № 1, 3–5.
10. Стешенко В. Б. Школа схемотехнического проектирования устройств обработки сигналов. Компоненты и технологии, № 3–6, 2000
11. http://www.andraka.com/files/crdcsrvy.pdf
12. Application Note 73 (Implementing FIR Filters in FLEX Device) v.1.01. Altera Corporation, 1998
13. LogiCore Digital Down Converter v1.0. Xilinx Inc., 2002
14. LogiCore Cascaded Integrator-Comb (CIC) Filter v2.0. Xilinx Inc., 2001
15. LogiCore Distributed Arithmetic FIR Filter v7.0. Xilinx Inc., 2002
16. LogiCore MAC FIR v2.0. Xilinx Inc., 2002
17. LogiCore CORDIC v1.1. Xilinx Inc., 2002
18. AD6620. 65 MSPS Digital Receive Signal Processor. Analog Devices, Inc., 1998
19. GC4016 Multi-Standard Quad DDC Chip Data Sheet. Rev.1.0. Graychip, Inc., 2001
20. CORDIC Core Specification. Rev.0.3 http://www.opencores.org, 2001
21. A Technical Tutorial on Digital Signal Syntesis. Analog Devices, Inc. 1999
22. High Speed Design Techniques. Section 6 in ANALOG DEVICES TECHNICAL REFERENCE BOOKS. Analog Devices, Inc. 1996
23. А.И.Солонина, Д.А.Улахович, С.М.Арбузов, Е.Б.Соловьева. Основы цифровой обработки сигналов. СПб, БХВ-Петербург, 2005

Программируемые логические интегральные схемы (ПЛИС)

Структура и особенности применения

Система цифровой обработки

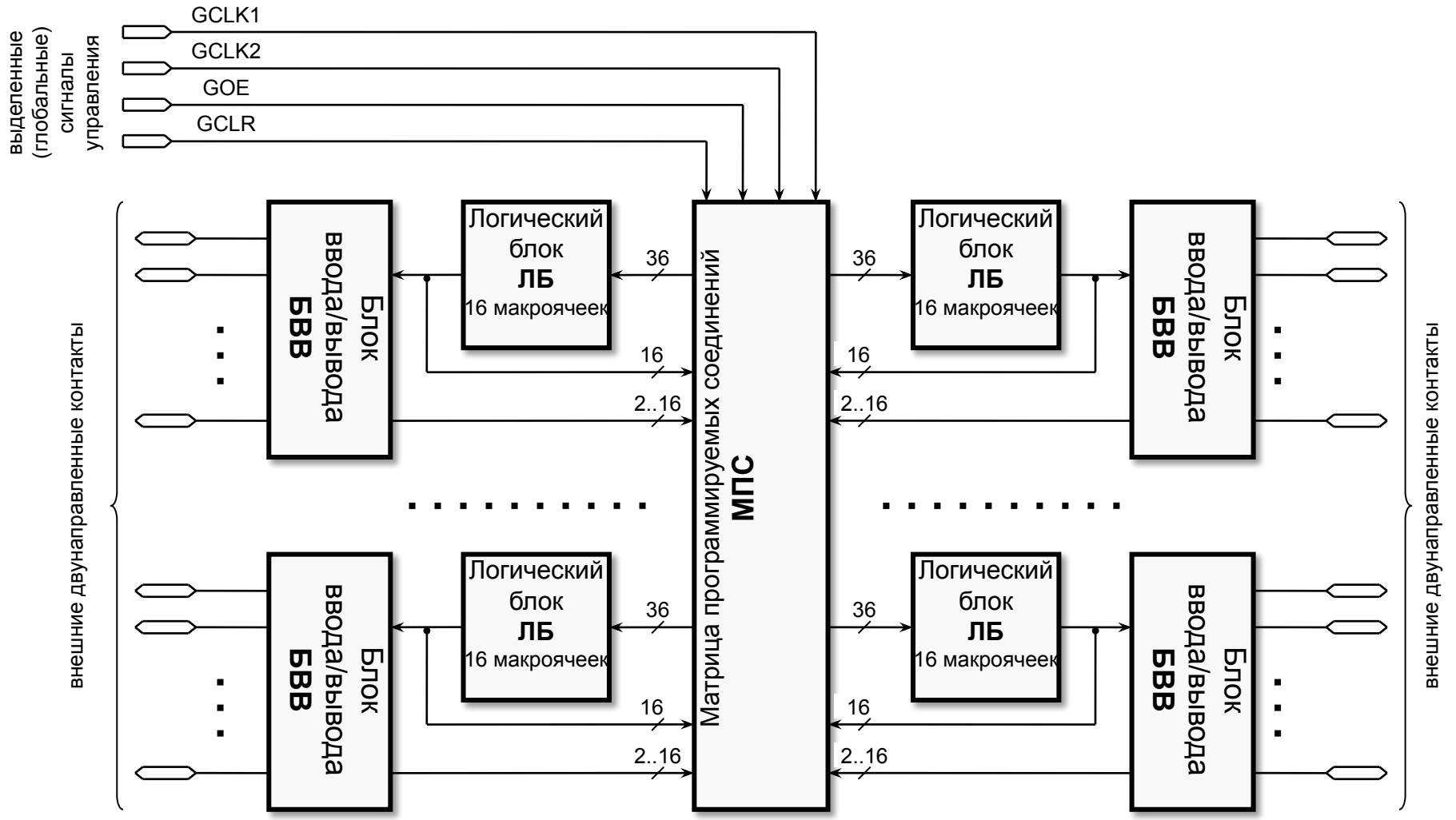


Для реализации СВ оптимально использование микросхем программируемой логики или ПЛИС (программируемые логические интегральные схемы). Привлекательность данной технологии заключается в предоставляемой конечному пользователю возможности быстрого создания цифровых устройств с произвольной внутренней структурой. В ПЛИС используются соединения, коммутируемые программируемыми ключами. Для задания этих соединений в ПЛИС существует теневая (конфигурационная) память, хранящая таблицу соединений.

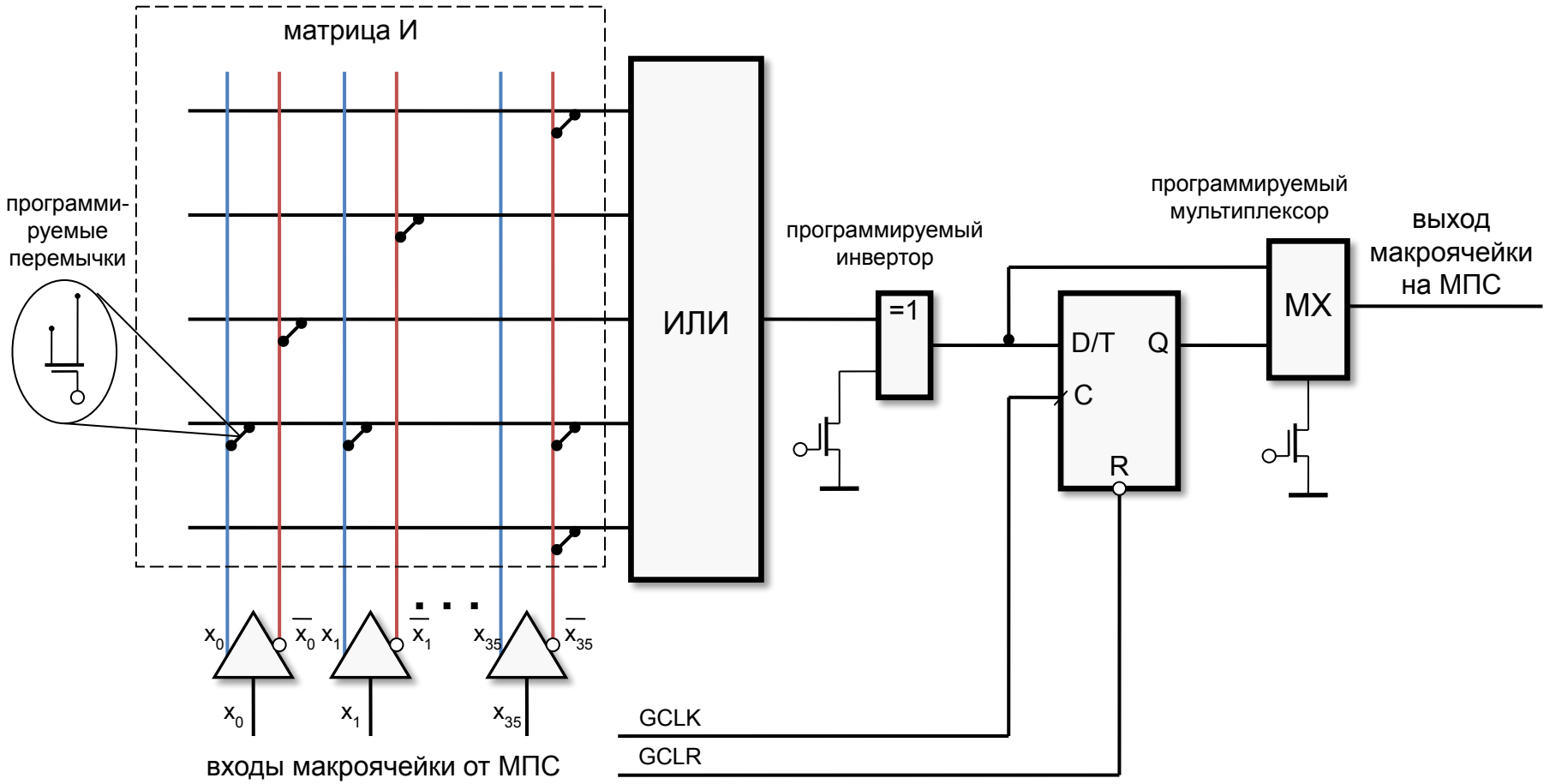
В настоящее время наиболее распространенные серии ПЛИС имеют следующую архитектуру:

- **CPLD** (**C**omplex **P**rogrammable **L**ogic **D**evice) устройства, использующие для хранения конфигурации энергонезависимую память (Flash или EEPROM);
- **FPGA** (**F**ield **P**rogrammable **G**ate **A**rray) устройства, использующие для хранения конфигурации энергозависимую память, которая требует инициализации после включения питания.

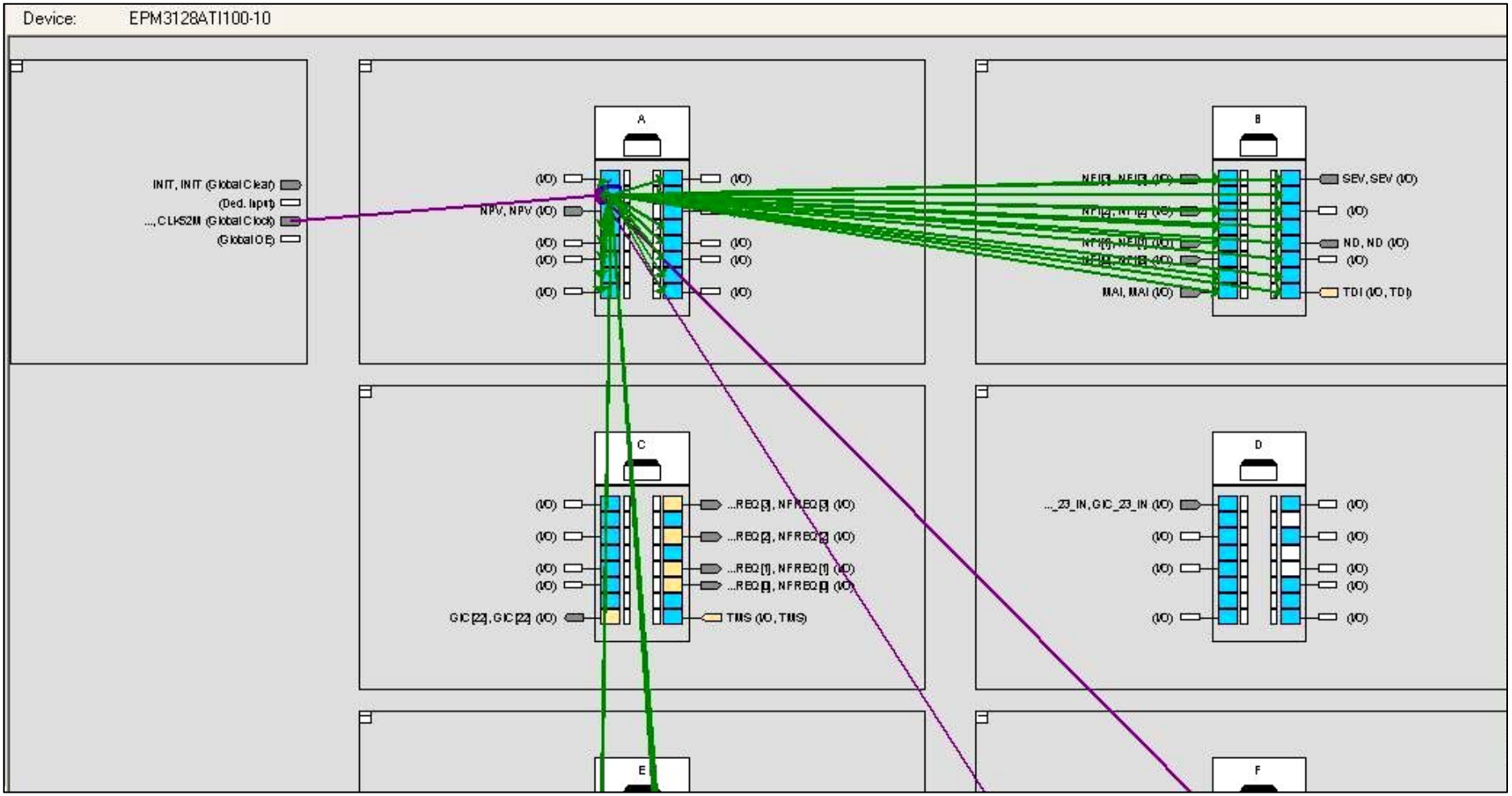
Общая структура CPLD



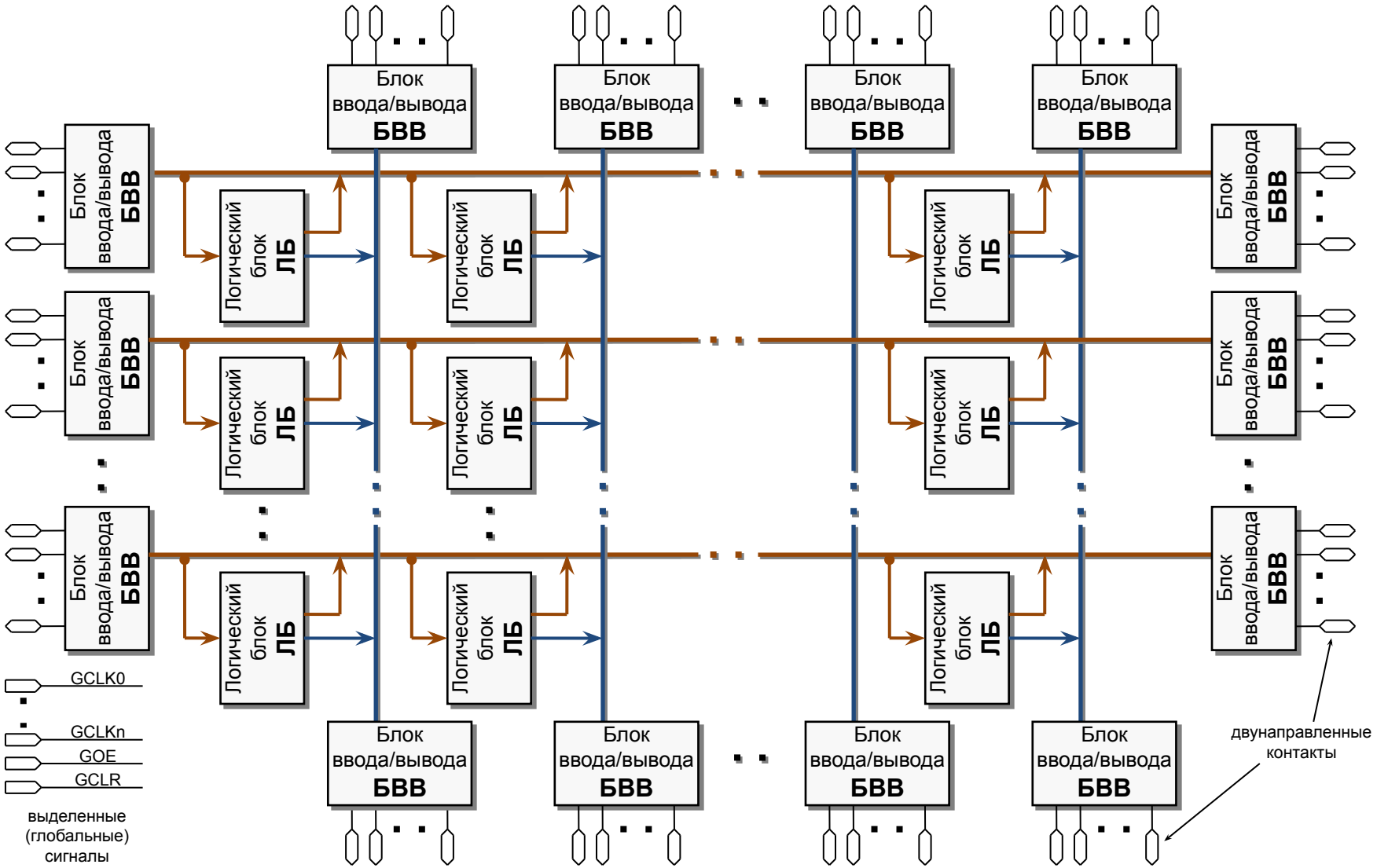
Макроячейка CPLD



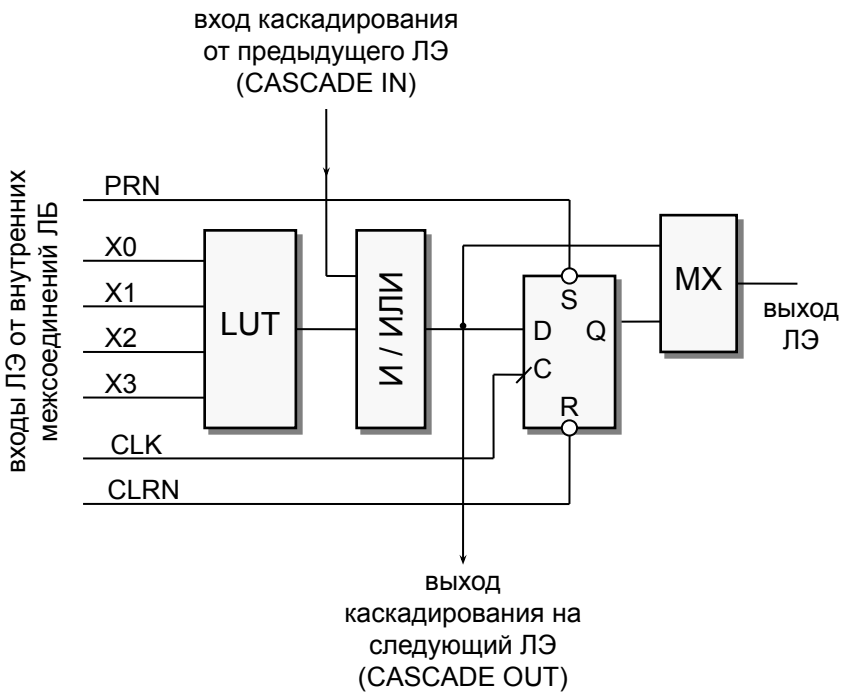
Структура CPLD (Timing Closure Floorplan)



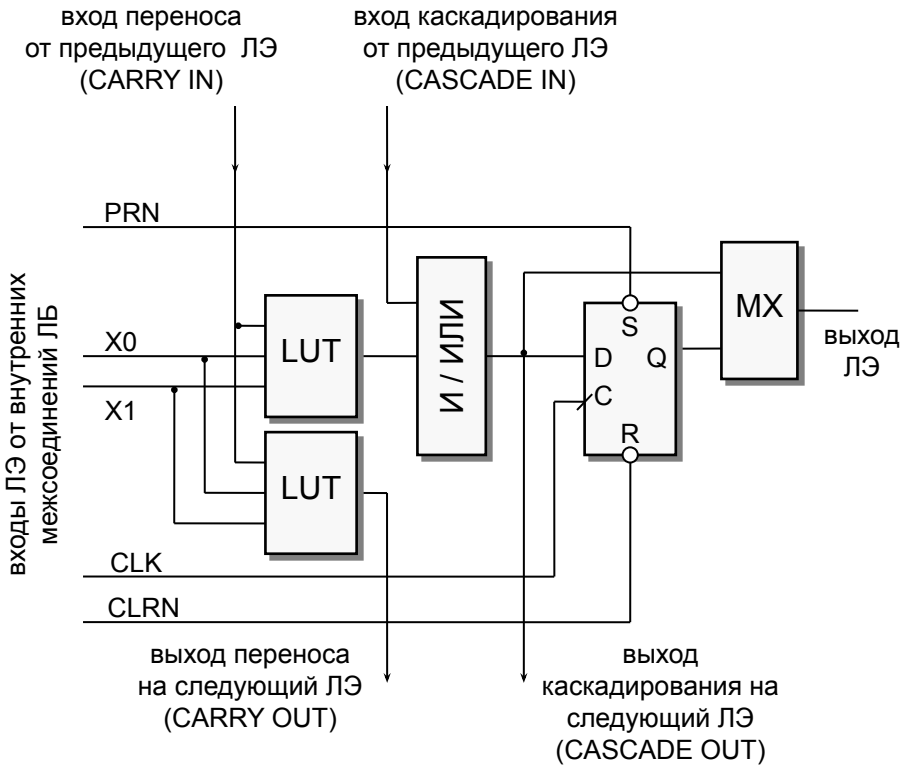
Общая структура FPGA



Логический элемент FPGA

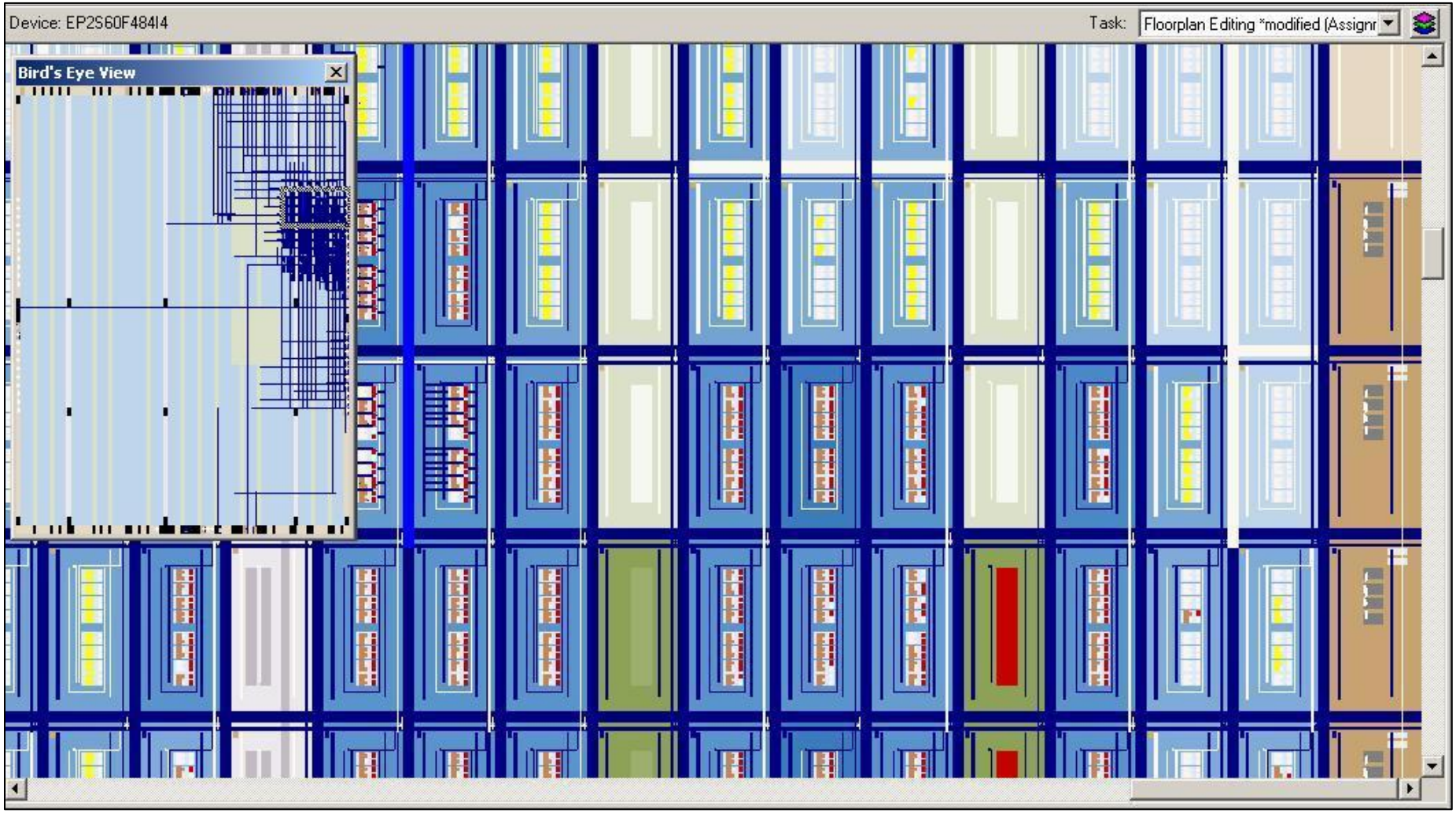


Нормальный режим

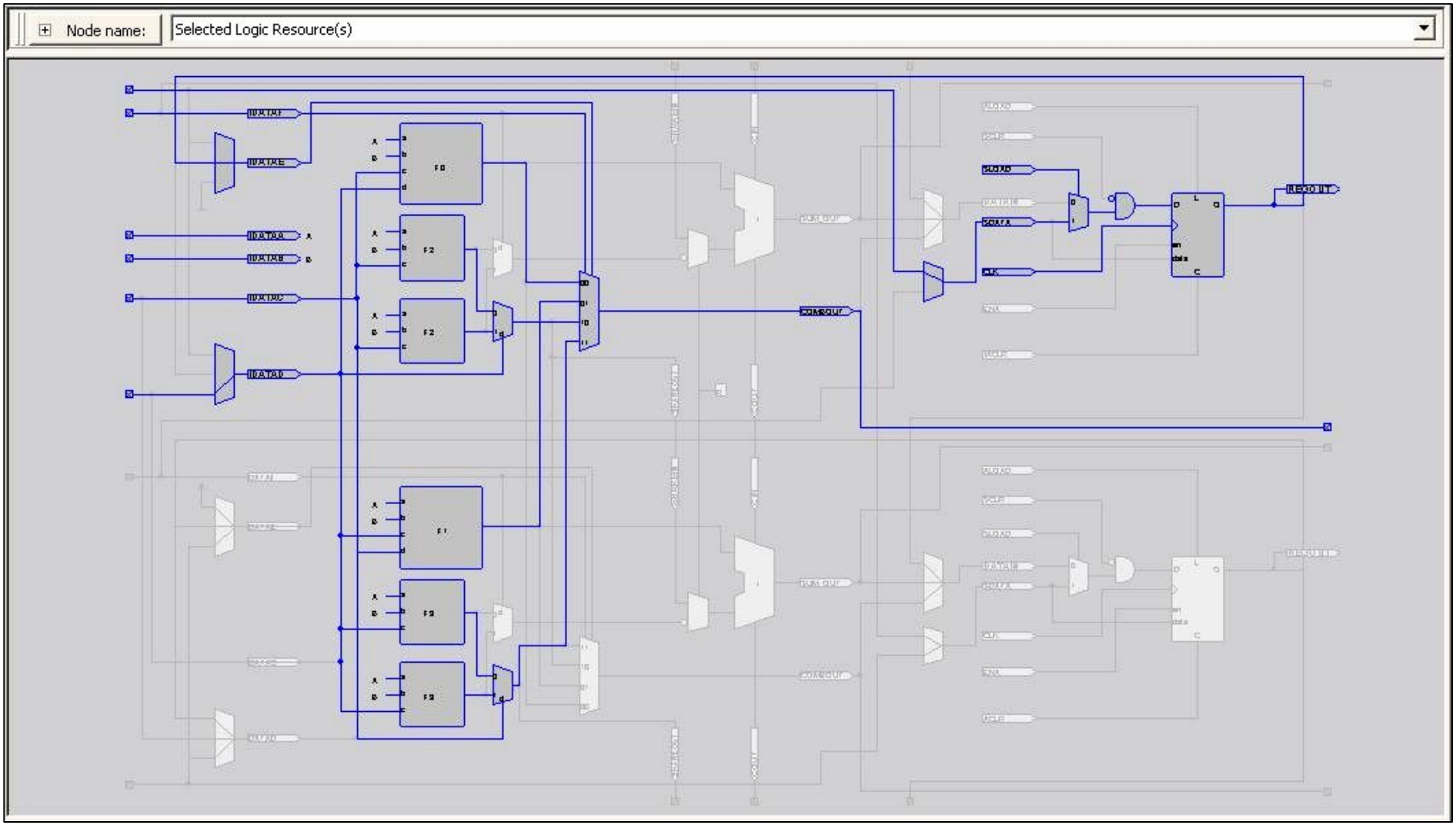


Арифметический режим

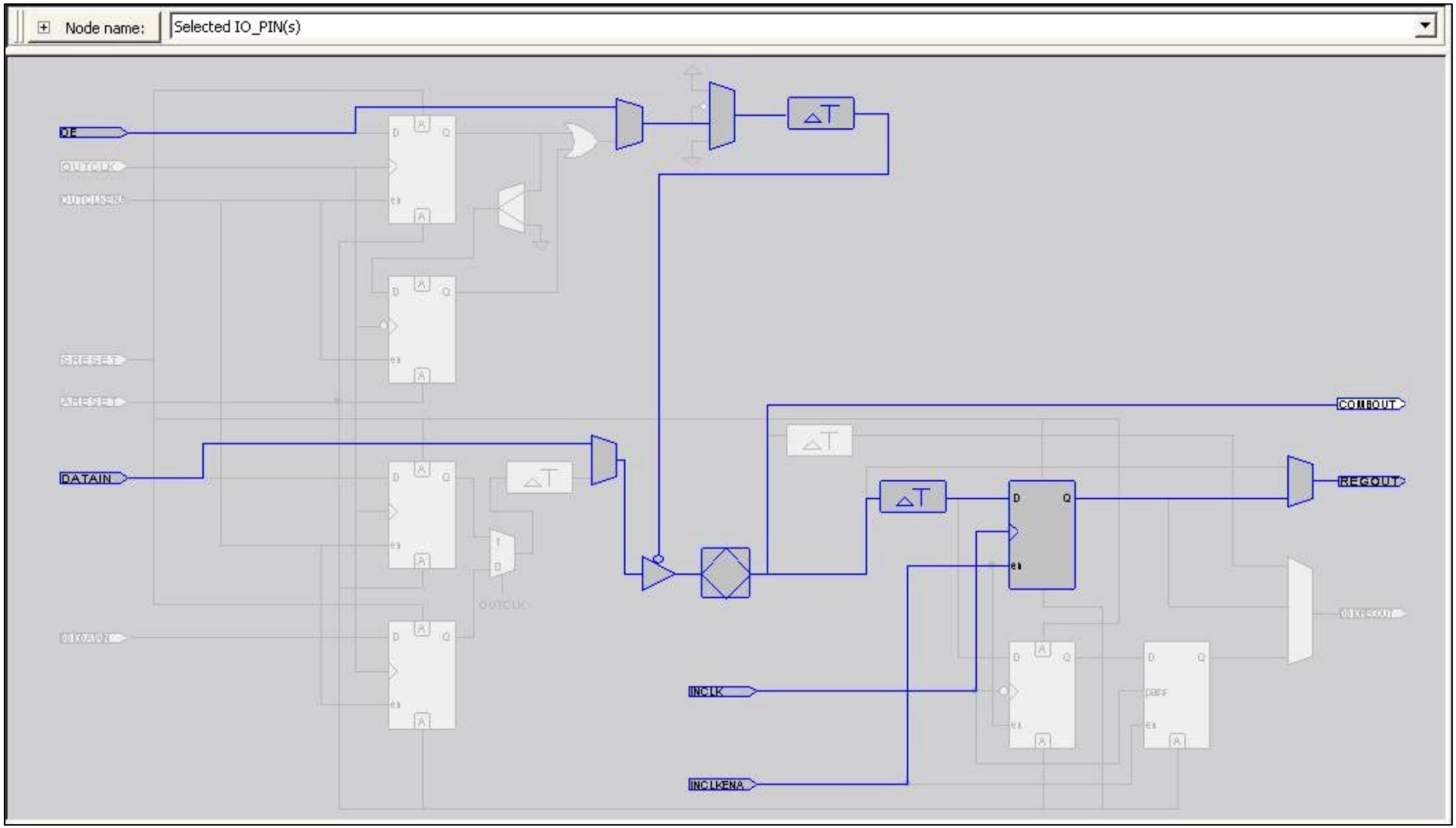
Общая структура FPGA (Chip Planner)



Логическая ячейка FPGA (Resource Property Editor)



Блок ввода-вывода FPGA (Resource Property Editor)



Реализация логических функций в FPGA и CPLD

Таблица истинности

Номер набора	x_2	x_1	x_0	y
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

реализация в FPGA

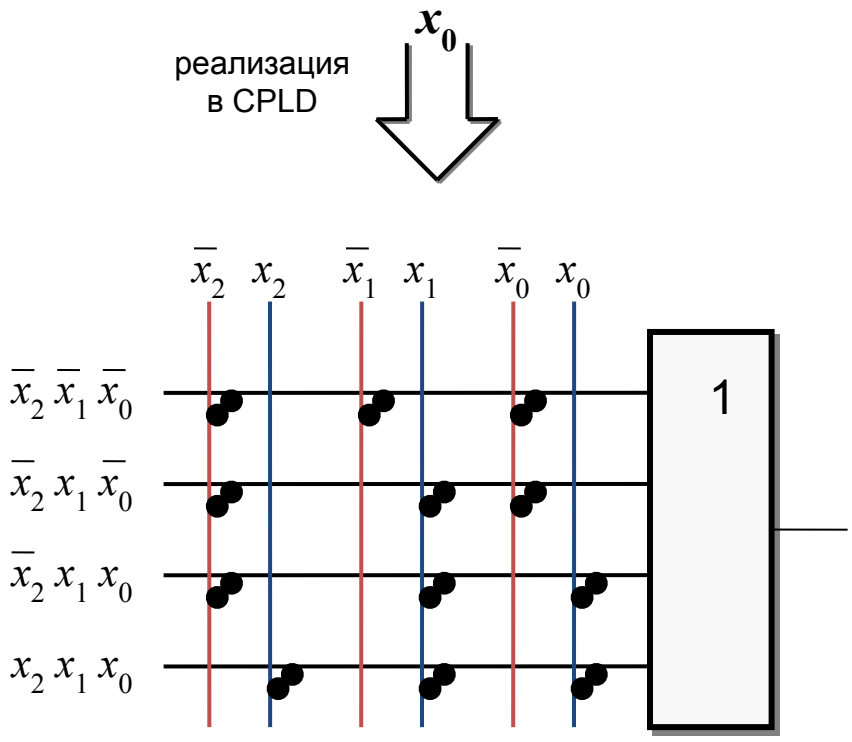
запись таблицы истинности непосредственно в LUT FPGA, содержимое Y записывается в ячейку памяти по адресу X

описания функции эквивалентны

Логическая функция в СДНФ

$$y = \bar{x}_2\,\bar{x}_1\,\bar{x}_0 \vee \bar{x}_2\,x_1\,\bar{x}_0 \vee x_2\,x_1\,x_0 \vee x_2\,x_1$$

реализация в CPLD

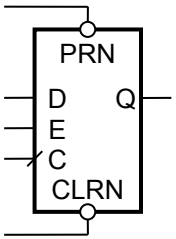


Основной элемент памяти CPLD/FPGA – D-триггер

C (Clock, CLK) – вход тактовых (синхронизирующих) импульсов

D (Data) – информационный вход (вход данных), синхронный

E (Enable) – вход разрешения тактирования, синхронный, H-активный



PRN (PReset) – вход предустановки, асинхронный, L-активный

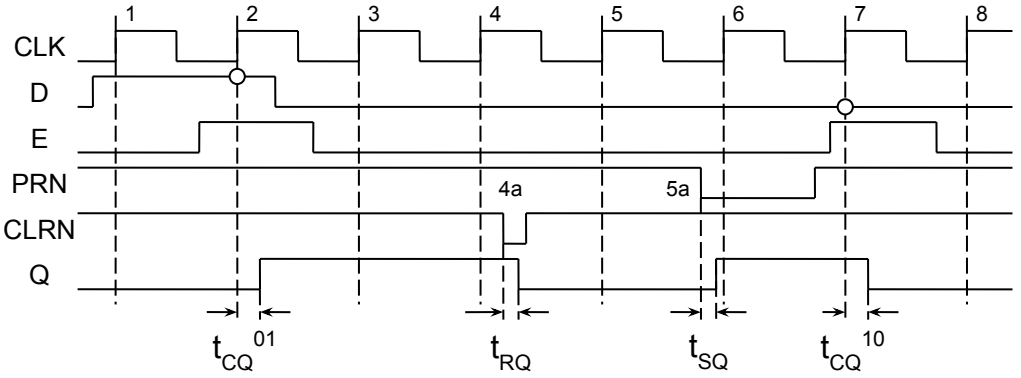
CLRN (CLeaR) – вход сброса, асинхронный, L-активный

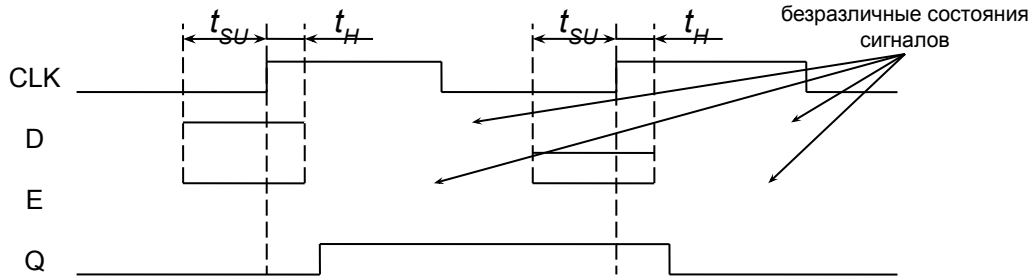
Q – выход триггера

Таблица переходов D-триггера

C	E	D	PRN	CLRN	Q(n)
↑	0	X	1	1	Q(n-1)
↑	1	0	1	1	0
↑	1	1	1	1	1
X	X	X	0	1	1
X	X	X	1	0	0

Основные временные соотношения D-триггера





t_{CQ}^{01} – время задержки переключения выхода из 0 в 1 относительно переднего фронта сигнала синхронизации CLK

t_{CQ}^{10} – время задержки переключения выхода из 1 в 0 относительно переднего фронта сигнала синхронизации CLK

t_{RQ} – время задержки переключения выхода в 0 относительно асинхронного сигнала сброса CLRN

t_{SQ} – время задержки переключения выхода в 1 относительно асинхронного сигнала установки PRN

t_{SU} – время предустановки (*SetUp*) управляющего сигнала относительно фронта синхронизации CLK

t_H – время удержания (*Hold*) управляющего сигнала относительно фронта синхронизации CLK

Арифметические блоки устройств ЦОС

Одноразрядный полный сумматор

Таблица истинности

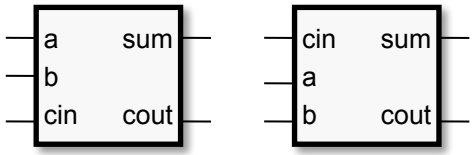
Входы			Выходы	
a	b	cin	sum	cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Логические функции выходов

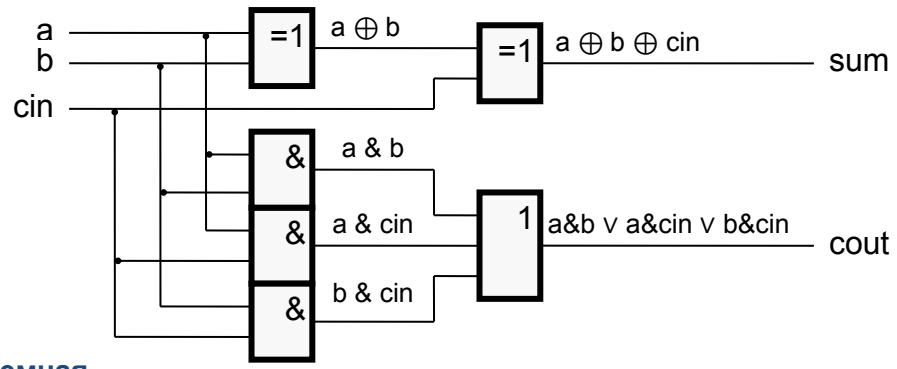
sum = a ⊕ b ⊕ cin

cout = a&b ∨ a&cin ∨ b&cin

Графические обозначения



Реализация



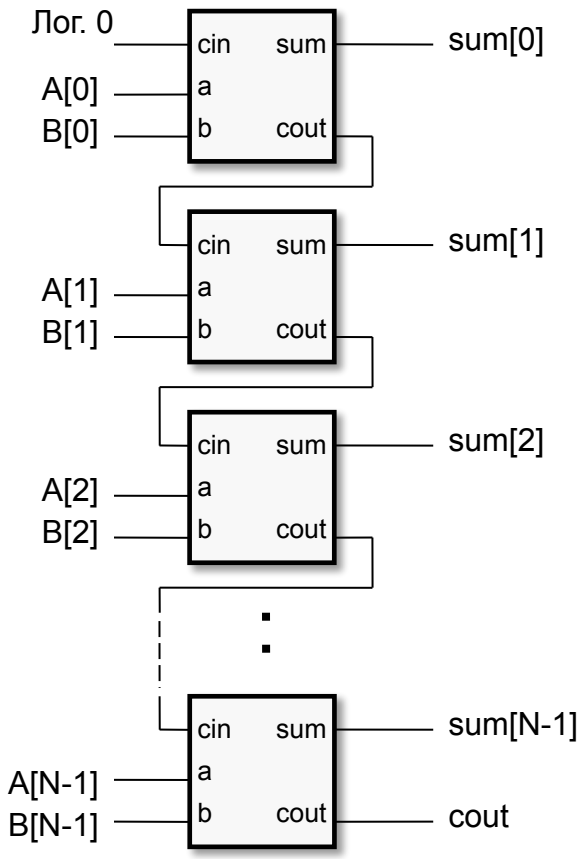
Схемная

На VHDL

```
-- Full Adder
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY adder IS
     PORT (  a, b, cin : IN STD_LOGIC;
             cout,sum  : OUT STD_LOGIC);
END adder;
architecture struct of adder is
begin
     sum <= a xor b xor cin;
     cout <= (a and b) or (a and cin) or (b and cin);
end struct;
```

Многоразрядный параллельный сумматор

Схемная и VHDL реализации многоразрядного параллельного сумматора с последовательным переносом



Логические функции выходов

$sum[i] = a[i] \oplus b[i] \oplus c[i]$

$c[i+1] = a[i]\&b[i] \vee a[i]\&c[i] \vee b[i]\&c[i],$
$c[0] = 0, \; cout = c[i+1]$

```
-- 32-bit Full Adder
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
entity add_32 is
generic(N: natural:=32);
port( A, B : IN  STD_LOGIC_VECTOR(N-1 DOWNTO 0);
      sum  : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0);
      cout : OUT STD_LOGIC);
end add_32;
architecture struct of add_32 is
  component
  adder
      PORT ( a, b, cin : IN  STD_LOGIC;
             cout, sum : OUT STD_LOGIC);
  end component;
  component
  add
      PORT ( a, b : IN  STD_LOGIC;
             cout, sum : OUT STD_LOGIC);
  end component;
signal cc : STD_LOGIC_VECTOR(0 to N-1);
begin
adder32: for i in 0 to N-1 generate
  first_bit:
      if(i=0) generate
      fistr_cell:  add
            port map( A(i), B(i), cc(i), sum(i));
      end generate first_bit;
  middle_bit:
      if(i>0) generate
      middle_cell: adder
            port map(A(i),B(i),cc(i-1),cc(i), sum(i));
      end generate middle_bit;
end generate adder32;
cout <= cc(N-1);
end struct;
```

Многоразрядный последовательный сумматор

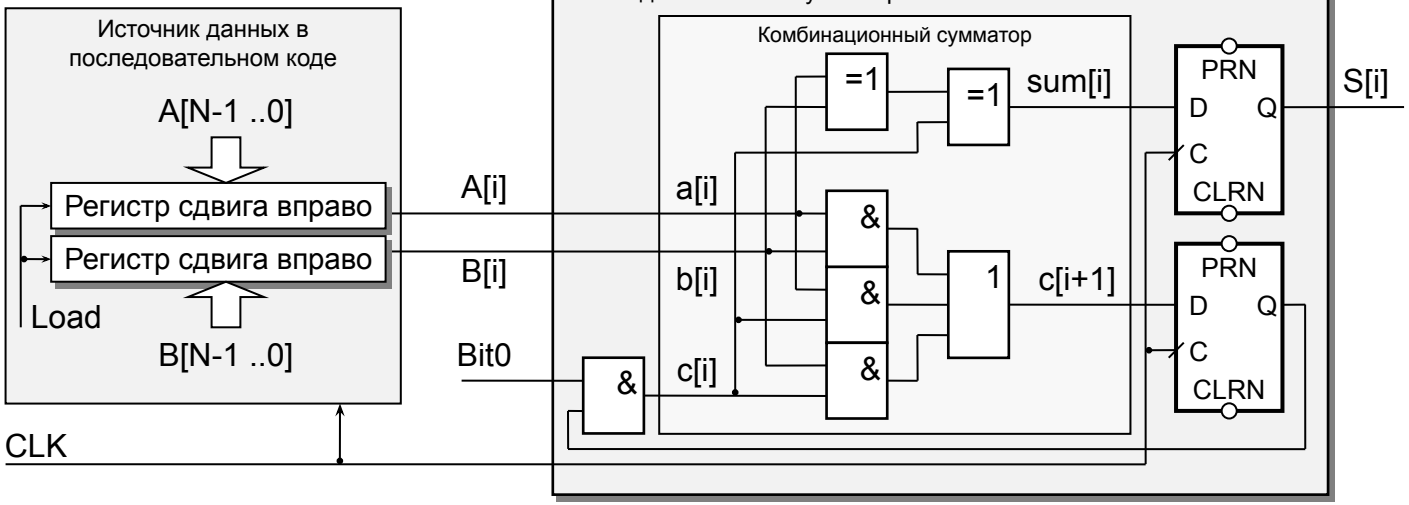
Пример сложения

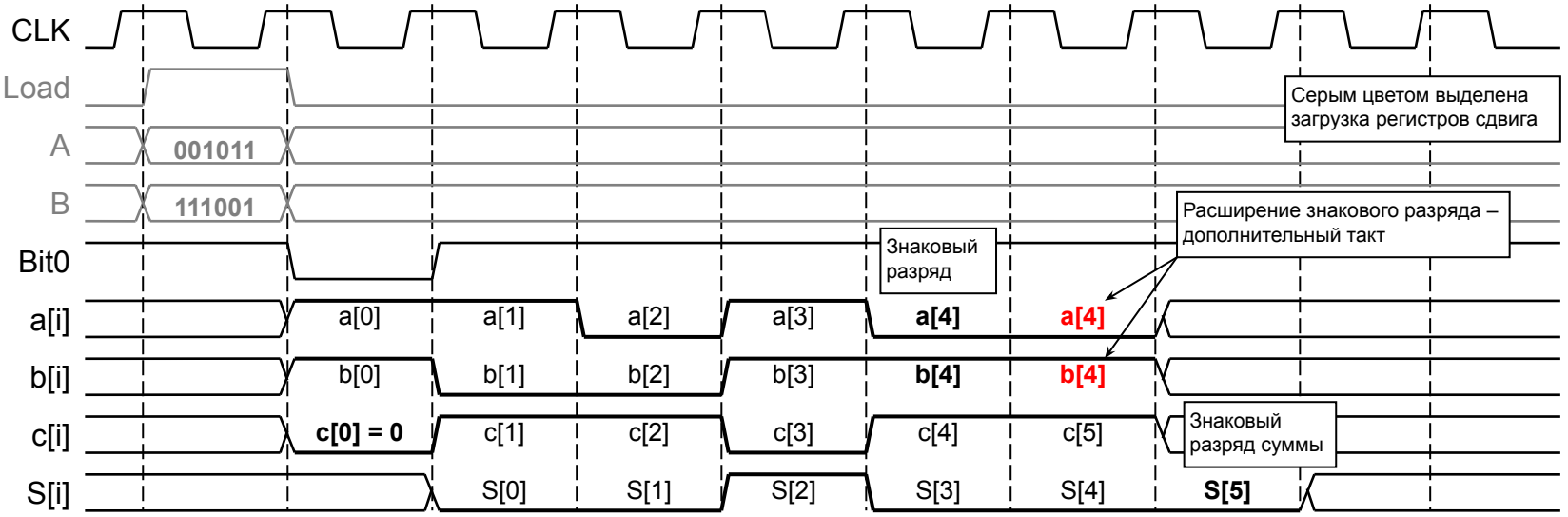
A = 0 01011(+11)

B = 1 11001 (–7)

A+B = X 0 00100 (+4)

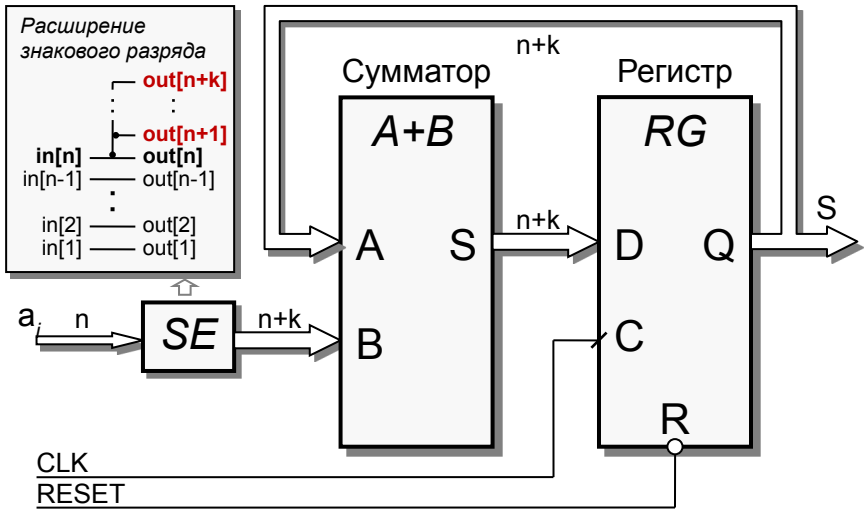
Возникающий при сложении перенос учитывается в дополнительном такте (расширение знакового разряда), все последующие переносы отбрасываются





Варианты многоразрядных сумматоров

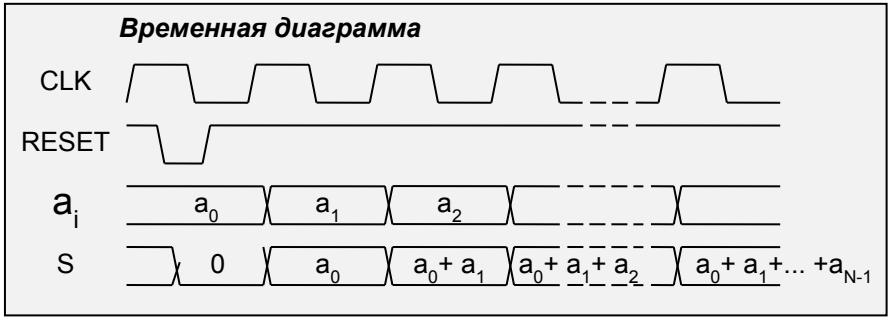
Аккумулятор (накапливающий сумматор)



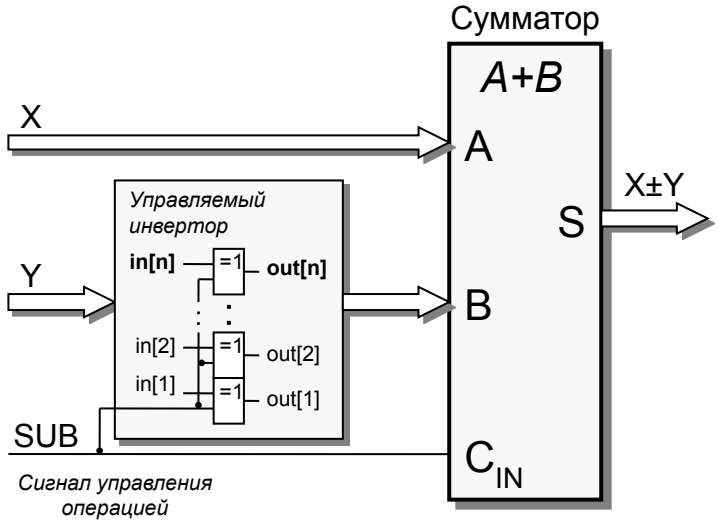
$$S = \sum_{i=0}^{N-1} a_i$$

$k = [log_2 N]$,
где [.] – операция округления
до большего целого

Временная диаграмма



Устройство сложения / вычитания



Для чисел в дополнительном коде

$-Y = \overline{Y} + 1$

независимо от знака числа

SUB	Вход A	Вход B	Вход C_{IN}	Выход S
0	X	Y	0	**X + Y**
1	X	$\overline{Y}$	1	X + ($\overline{Y}$+1) = **X – Y**

Умножение чисел без знака

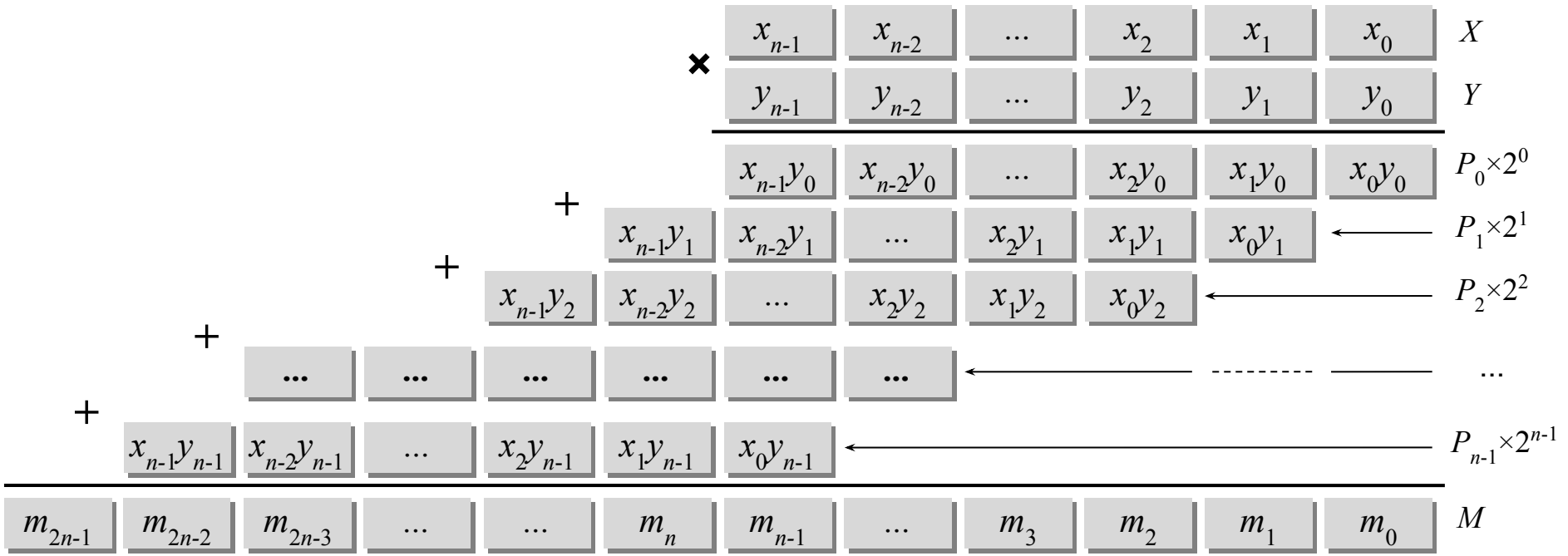
$X = x_{n-1}, x_{n-2}, \ldots x_0$, $Y = y_{n-1}, y_{n-2}, \ldots y_0$, где x_i, y_i – двоичные цифры

Произведение $X \times Y = M = m_{2n-1}, m_{2n-2}, \ldots m_{n-1}, \ldots m_0 = \sum_{i=0}^{n-1} P_i \times 2^i = \sum_{i=0}^{n-1} (x_{n-1}, x_{n-2}, \ldots x_0) \times y_i \times 2^i$, где 2^i – вес разряда y_i

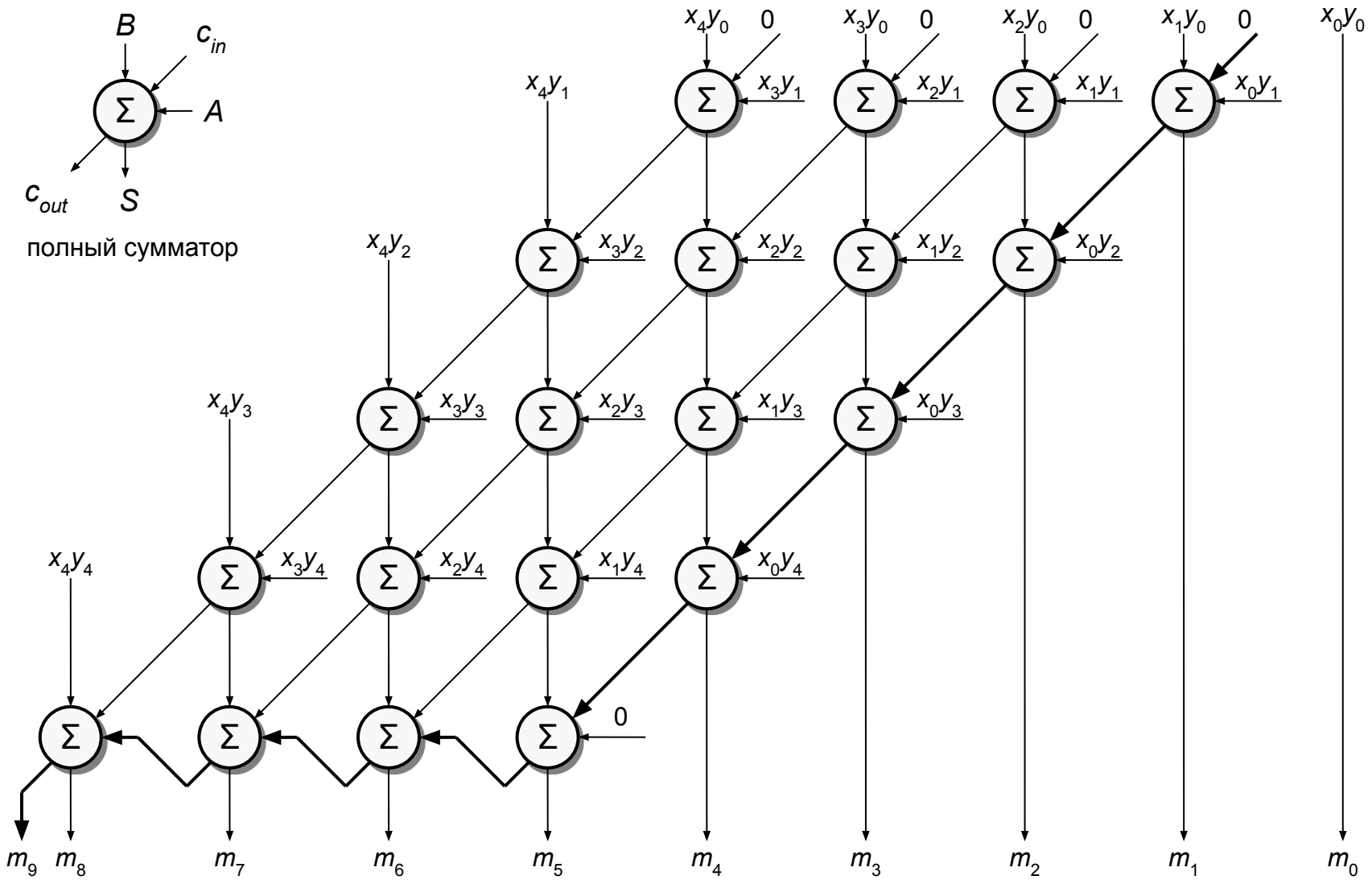
Частичное произведение $P_i = p_{n-1}, p_{n-2}, \ldots p_0 = (x_{n-1}, x_{n-2}, \ldots x_0) \times y_i = x_{n-1}y_i, x_{n-2}y_i, \ldots x_0y_i$, где x_jy_i – элементарные произведения

Элементарное произведение $p_i = x_j \times y_i$ вычисляется двухвходовым логическим элементом И,
для одноразрядных двоичных чисел арифметическое и логическое умножение эквивалентны $x_j \times y_i \equiv x_j \,\&\, y_i$

Алгоритм умножения "столбиком" для положительных чисел



Структура умножителя для чисел без знака

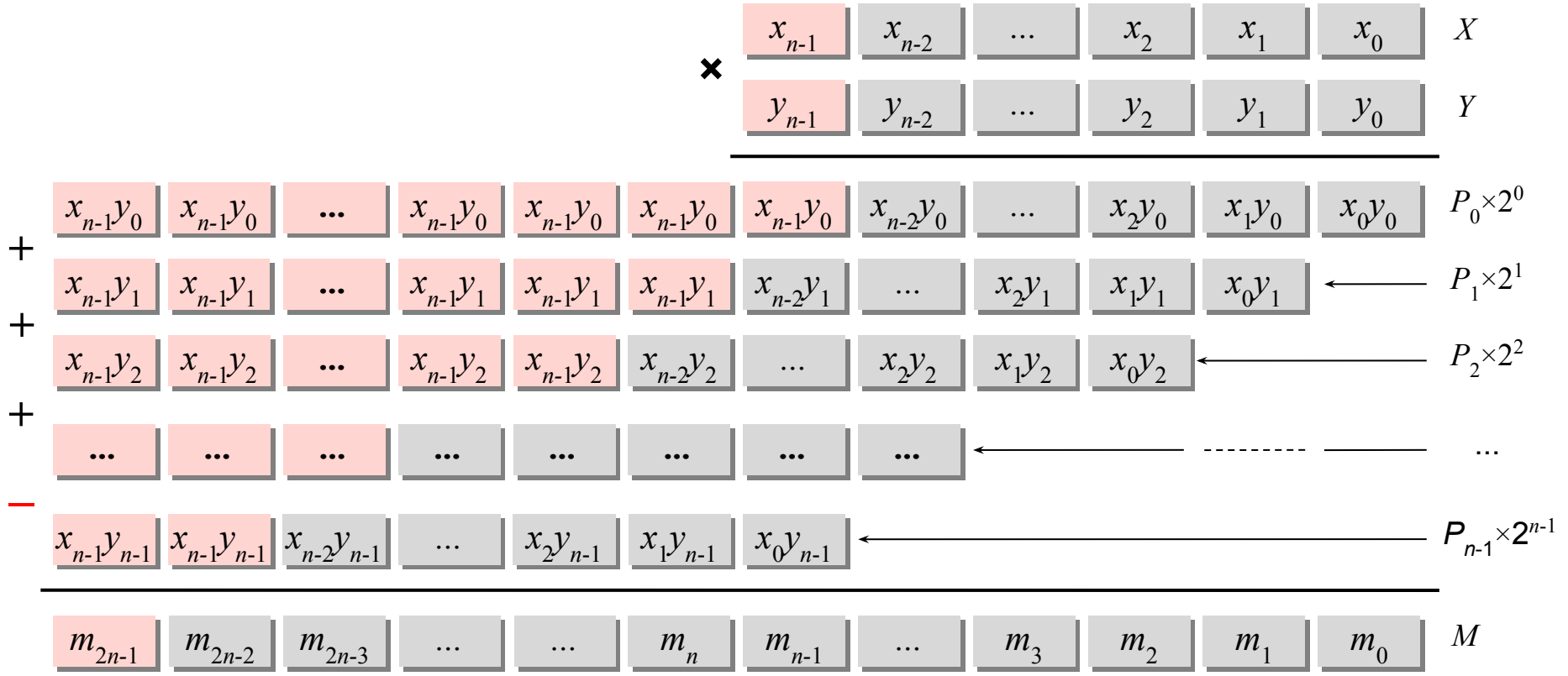


Умножение чисел со знаком

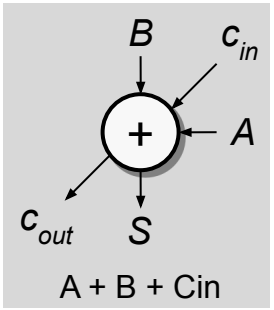
Для чисел со знаком в дополнительном коде $X = \boldsymbol{x_{n-1}}, x_{n-2}, \ldots, x_1, x_0 = \boldsymbol{x_{n-1}} \times (-2^{n-1}) + x_{n-2} \times 2^{n-2} + \ldots + x_1 \times 2^1 + x_0 \times 2^0$

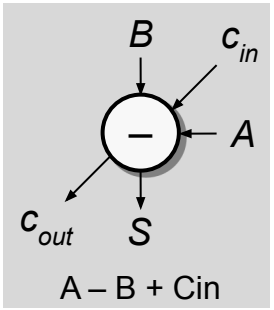
Знаковый разряд числа имеет вес -2^{n-1}

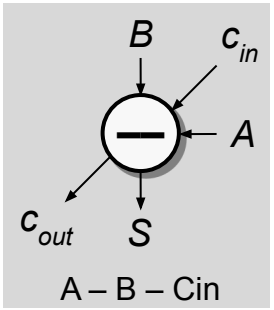
Алгоритм умножения "столбиком" для чисел со знаком

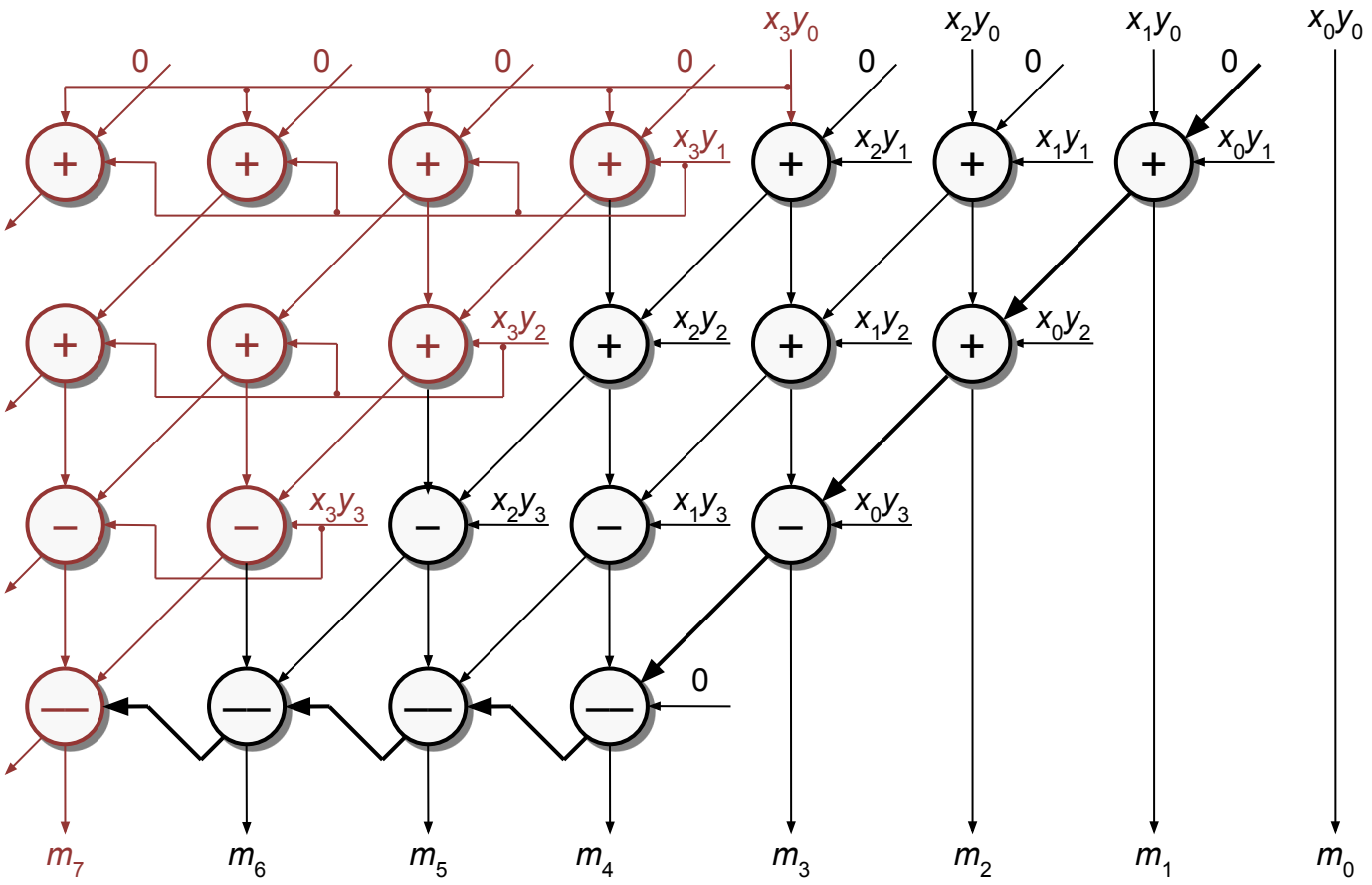


Структура умножителя для чисел со знаком









Примеры умножения чисел со знаком

$a = -3 = 1101_2$

$b = -7 = 1001_2$

				×	1	1	0	1	
					1	0	0	1	
	1	1	1	1	1	1	0	1	→ P_0
+	0	0	0	0	0	0	0		→ P_1
+	0	0	0	0	0	0			→ P_2
−	1	1	1	0	1				→ P_3
	0	0	0	1	0	1	0	1	→ M

$M = 00010101_2 = +21$

$a = -3 = 1101_2$

$b = +7 = 0111_2$

				×	1	1	0	1	
					0	1	1	1	
	1	1	1	1	1	1	0	1	→ P_0
+	1	1	1	1	1	0	1		→ P_1
+	1	1	1	1	0	1			→ P_2
−	0	0	0	0	0				→ P_3
	1	1	1	0	1	0	1	1	→ M

$M = 11101011_2 = -21$

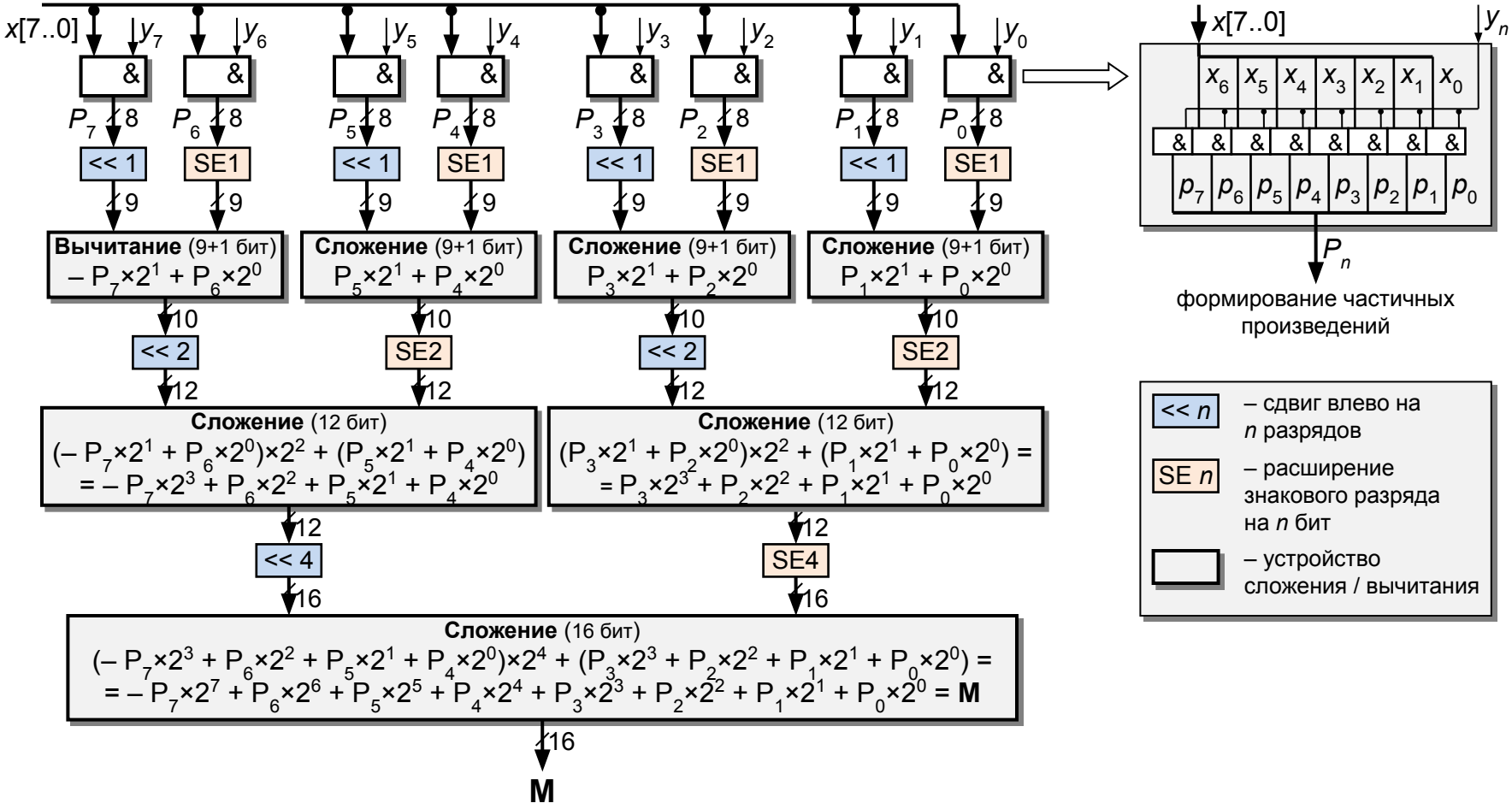
Умножитель с деревом сумматоров

$$M = -P_7 \cdot 2^7 + P_6 \cdot 2^6 + P_5 \cdot 2^5 + P_4 \cdot 2^4 +$$
$$+ P_3 \cdot 2^3 + P_2 \cdot 2^2 + P_1 \cdot 2^1 + P_1 \cdot 2^1 =$$

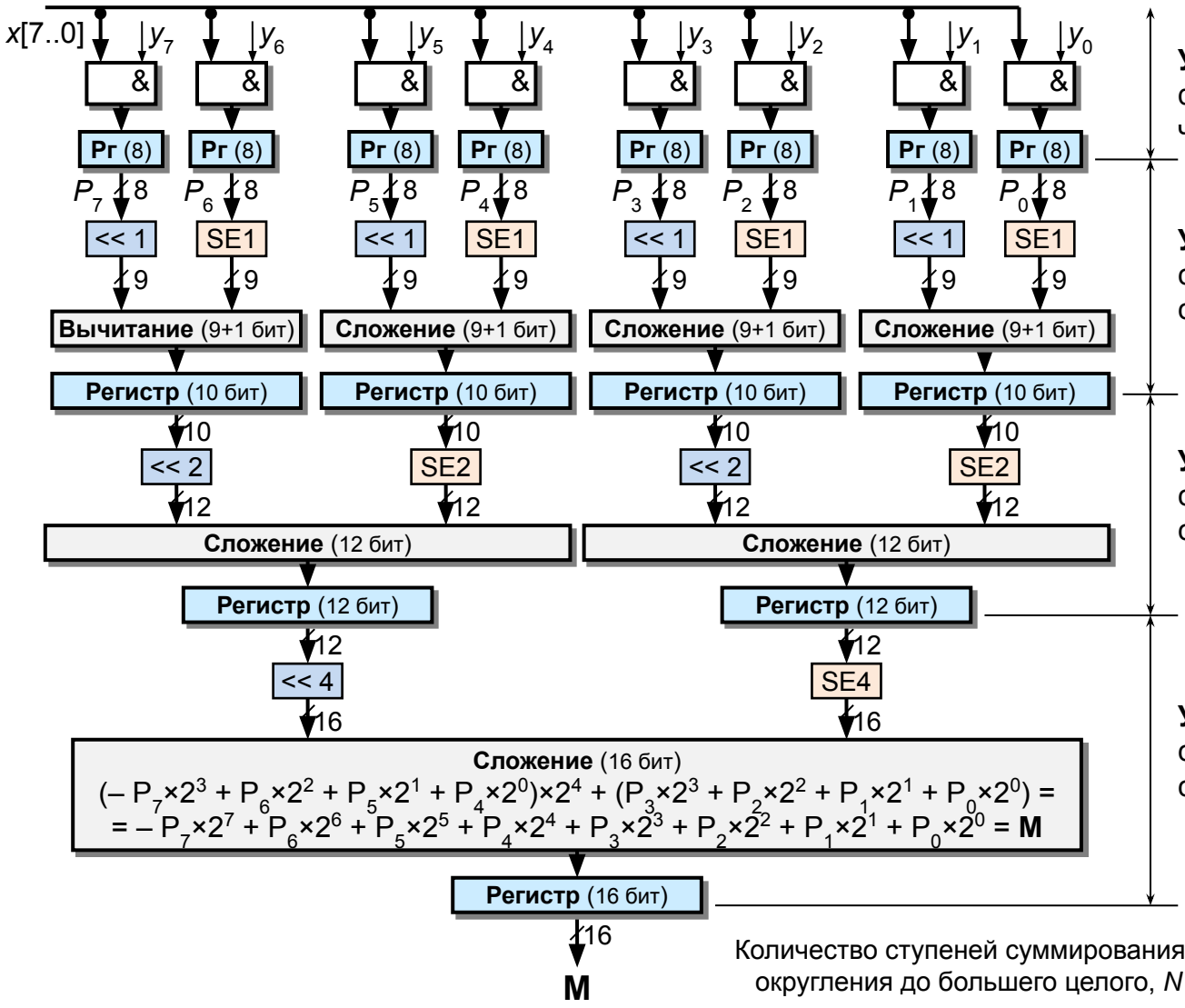
$$= (-P_7 \cdot 2^1 + P_6 \cdot 2^0) \times 2^6 + (P_5 \cdot 2^1 + P_4 \cdot 2^0) \times 2^4 +$$
$$+ (P_3 \cdot 2^1 + P_2 \cdot 2^0) \times 2^2 + (P_1 \cdot 2^1 + P_0 \cdot 2^0) \times 2^0 =$$

$$= [(-P_7 \cdot 2^1 + P_6 \cdot 2^0) \times 2^2 + (P_5 \cdot 2^1 + P_4 \cdot 2^0) \times 2^0] \times 2^4 +$$
$$+ [(P_3 \cdot 2^1 + P_2 \cdot 2^0) \times 2^2 + (P_1 \cdot 2^1 + P_0 \cdot 2^0) \times 2^0] \times 2^0$$

Умножитель с деревом сумматоров (структура)



Конвейерный умножитель



Уровень 1 – формирование частичных произведений

Уровень 2 – сумматоры первой ступени

Уровень 3 – сумматоры второй ступени

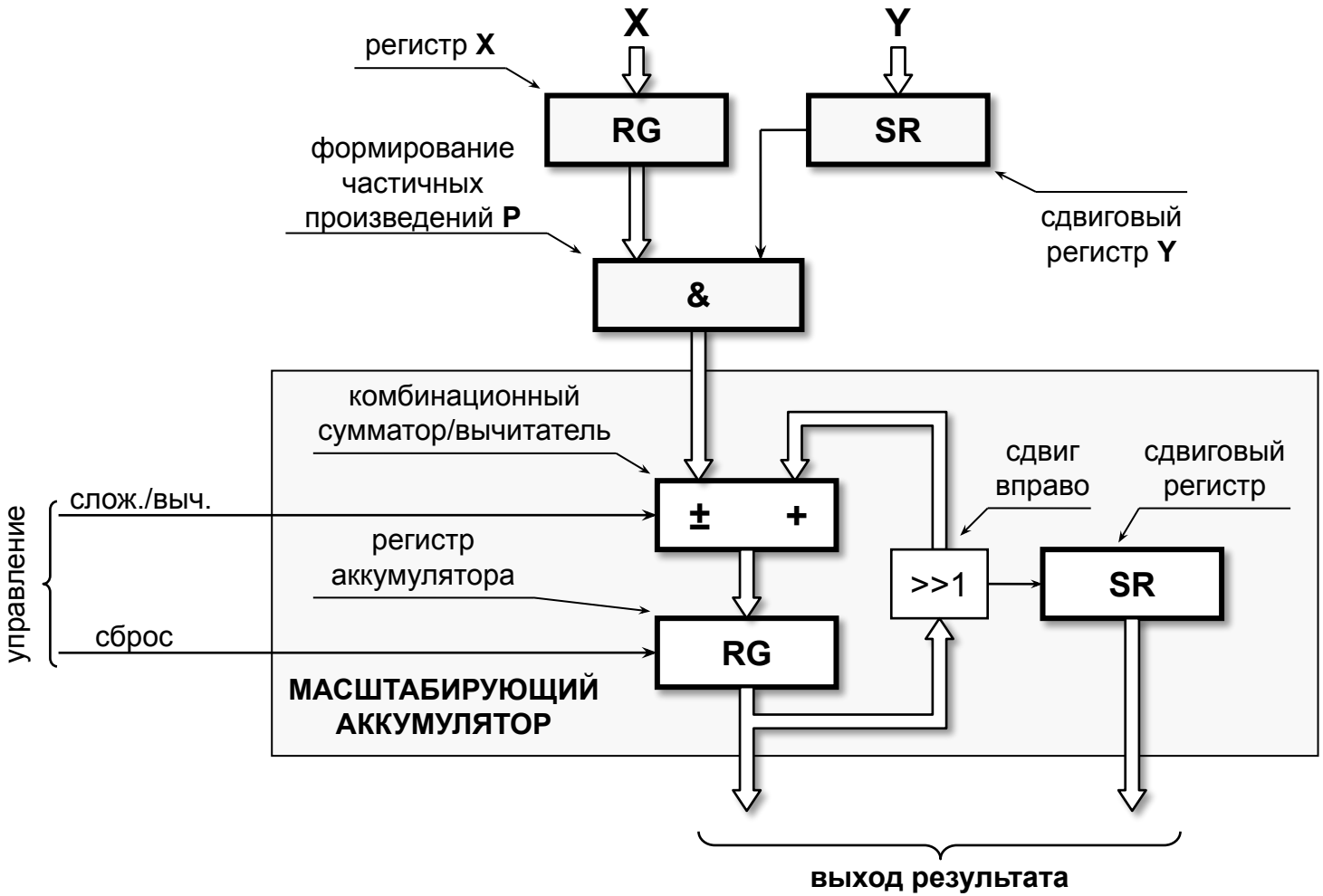
Уровень 4 – сумматоры третьей ступени

Количество ступеней суммирования $S = [\log_2 N]$, где [.] – операция округления до большего целого, N – разрядность множителя y.

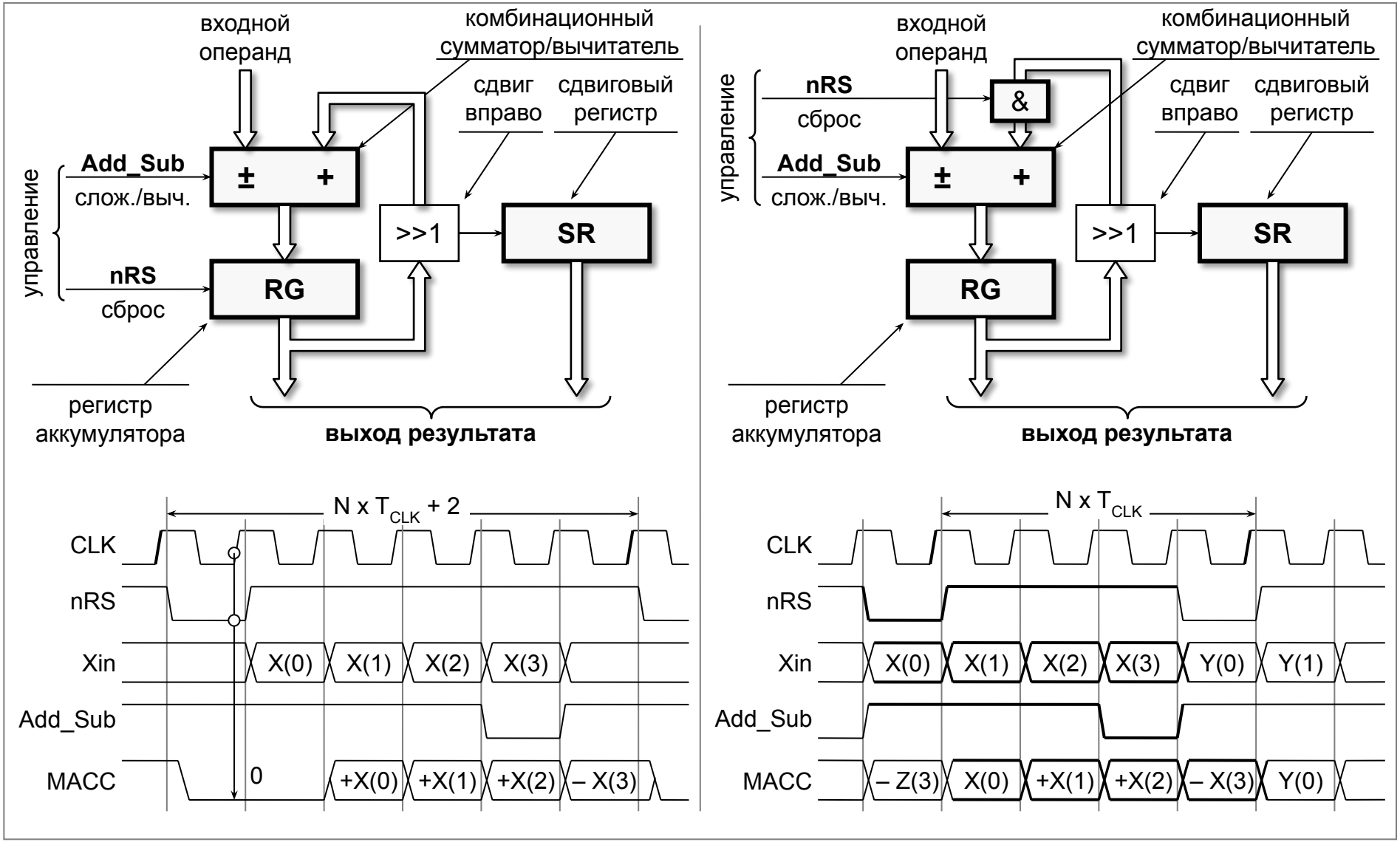
Умножитель с масштабирующим аккумулятором

Произведение M = a x b = 00001001_2 = +9

Умножитель с масштабирующим аккумулятором



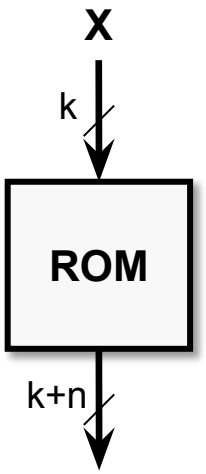
Управление масштабирующим аккумулятором



Умножитель на константу (принцип построения)

Вариант 1

$\mathbf{X} = x_{k-1}, x_{k-2}, \ldots, x_{k/2}, x_{k/2-1}, \ldots, x_0$



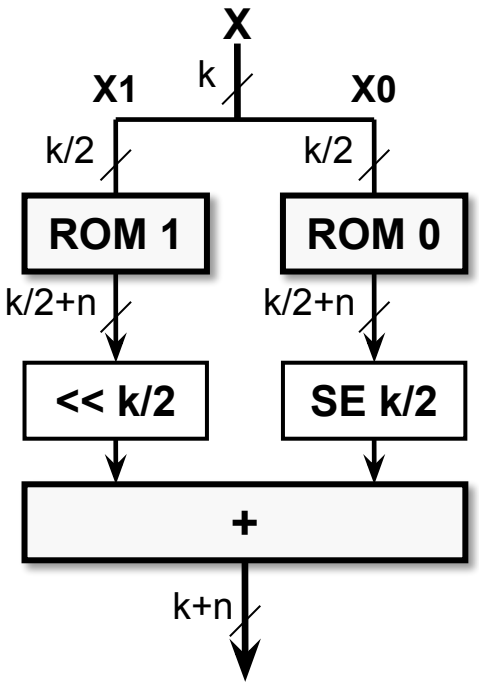
Y = X × Const

Пример
Разрядность константы **n**

k = 16, n = 16
Объем памяти ROM
V = 2^{16} x 32 = **256 КБайт**

Вариант 2

$\mathbf{X} = x_{k-1}, x_{k-2}, \ldots, x_{k/2}, x_{k/2-1}, \ldots, x_0 =$
$= (x_{k-1}, x_{k-2}, \ldots, x_{k/2}) \times 2^{k/2} + (x_{k/2-1}, \ldots, x_0) \times 2^0 =$
$= X1 \times 2^{k/2} + X0 \times 2^0$



Y = X × Const

Пример
Разрядность константы **n**

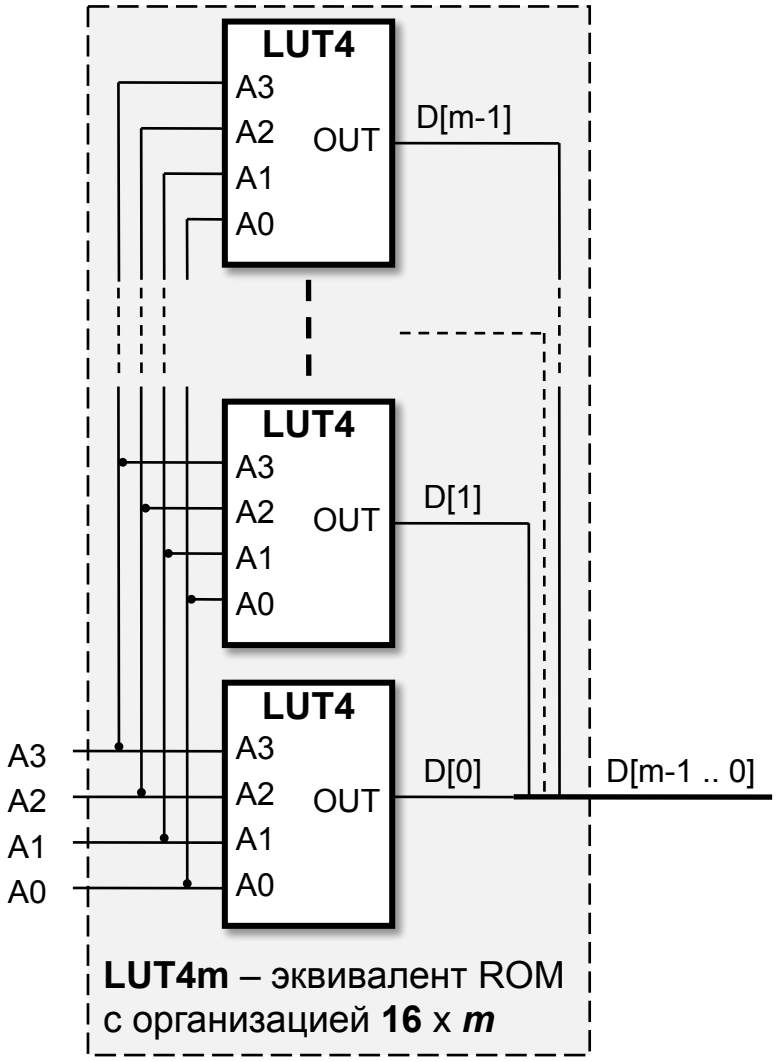
k = 16, n = 16
Объем памяти ROM 0
$V_0 = 2^8$ x 24 = 768 Байт

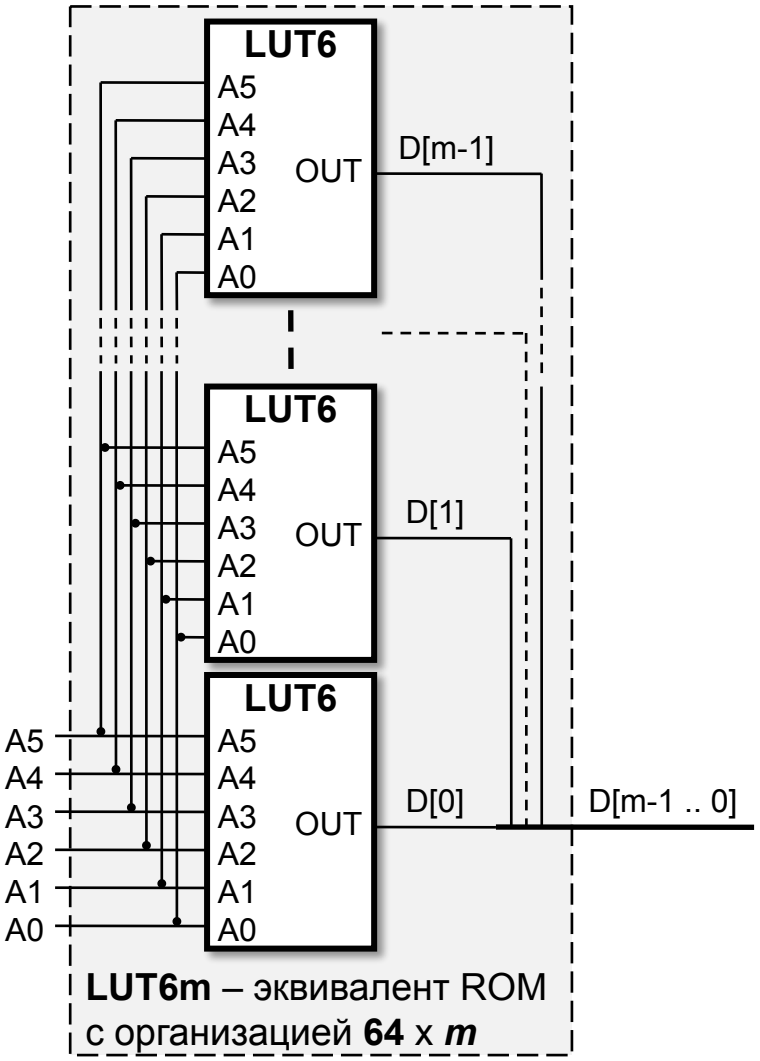
Объем памяти ROM 1
$V_1 = 2^8$ x 24 = 768 Байт

Общий объем памяти
V = **1536 Байт**

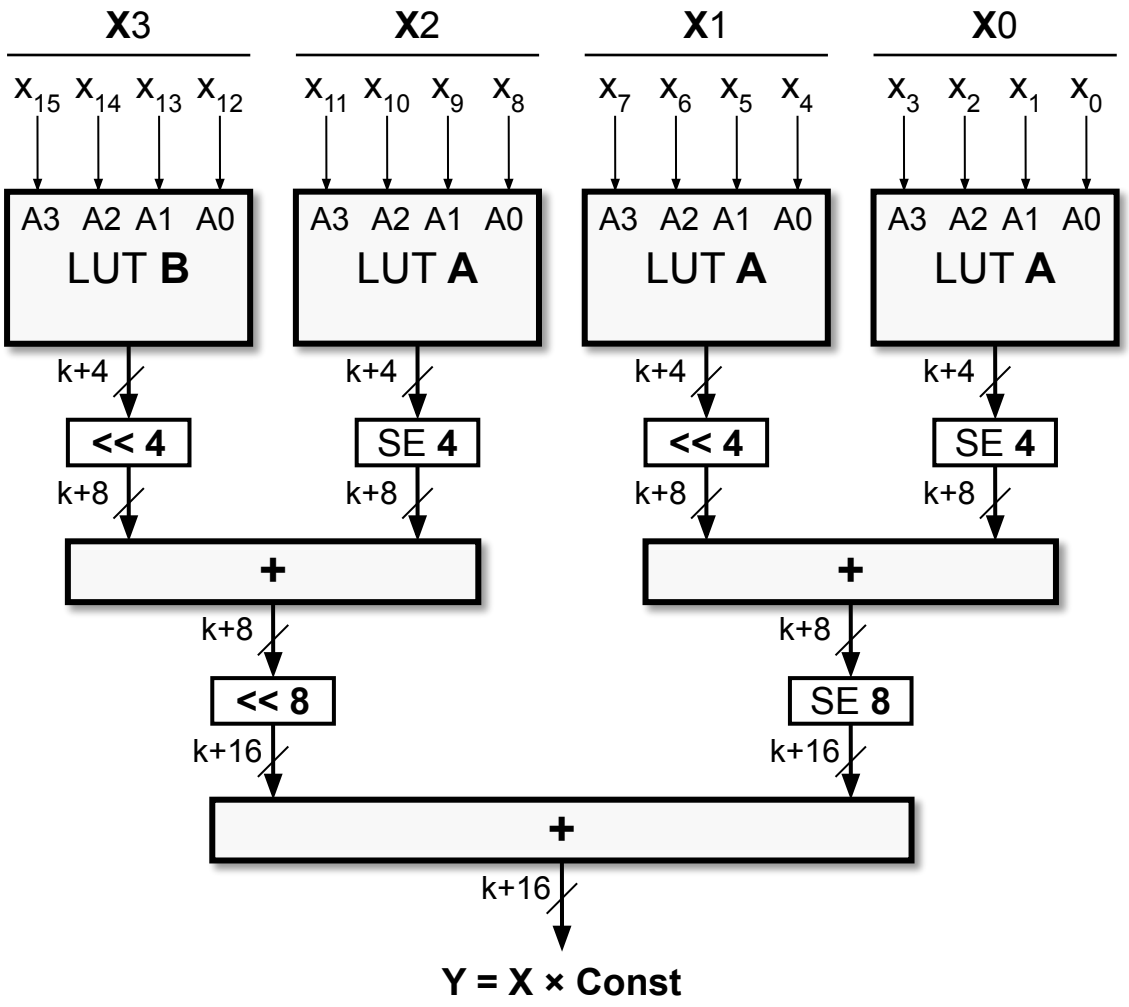
Замечание.
Память ROM 1 учитывает знак операнда X1

Умножитель на константу (ROM на LUT)





Умножитель на константу (структура)



Для операндов произвольной разрядности
$X = x_{n-1}, x_{n-2}, \ldots, x_1, x_0$ и
$Y = y_{k-1}, y_{k-2}, \ldots, y_1, y_0$, где
Y – константа, потребуется [n/4] таблиц LUT с организацией 16 × (k+4) и [log2 (n/4)] ступеней суммирования.
[.] – операция округления до большего целого.

Пример
Разрядность операнда X равна 16 (n=16).
Операнд разбивается на 4 группы по 4 разряда X0, X1, X2, X3.
Группы X0, X1 и X2 – числа без знака, группа X3 – число со знаком (учитывается вес знакового разряда x_{15}).
Разряды каждой группы являются адресами таблиц LUT.
Для групп X0, X1 и X2 используются таблицы LUT A, для группы X3 – таблица LUT B.

Умножитель на константу (содержимое LUT)

ТАБЛИЦА **LUT A** (адрес – число без знака)		
АДРЕС		СОДЕРЖИМОЕ ЯЧЕЙКИ
код	значение	
0000	+0	+0 × Const
0001	+1	+1 × Const
0010	+2	+2 × Const
0011	+3	+3 × Const
0100	+4	+4 × Const
0101	+5	+5 × Const
0110	+6	+6 × Const
0111	+7	+7 × Const
1000	**+8**	**+8 × Const**
1001	**+9**	**+9 × Const**
1010	**+10**	**+10 × Const**
1011	**+11**	**+11 × Const**
1100	**+12**	**+12 × Const**
1101	**+13**	**+13 × Const**
1110	**+14**	**+14 × Const**
1111	**+15**	**+15 × Const**

ТАБЛИЦА **LUT B** (адрес – число со знаком)		
АДРЕС		СОДЕРЖИМОЕ ЯЧЕЙКИ
код	значение	
0000	+0	+0 × Const
0001	+1	+1 × Const
0010	+2	+2 × Const
0011	+3	+3 × Const
0100	+4	+4 × Const
0101	+5	+5 × Const
0110	+6	+6 × Const
0111	+7	+7 × Const
1000	**– 8**	**– 8 × Const**
1001	**– 7**	**– 7 × Const**
1010	**– 6**	**– 6 × Const**
1011	**– 5**	**– 5 × Const**
1100	**– 4**	**– 4 × Const**
1101	**– 3**	**– 3 × Const**
1110	**– 2**	**– 2 × Const**
1111	**– 1**	**– 1 × Const**

Умножитель на константу (пример 1)

Const = 2047 (7FFh)

ТАБЛИЦА **LUT A** (адрес – число без знака)	
АДРЕС	СОДЕРЖИМОЕ ЯЧЕЙКИ (hex)
0000	0000
0001	07FF
0010	0FFE
0011	17FD
0100	1FFC
0101	27FB
0110	2FFA
0111	37F9
1000	**3FF8**
1001	**47F7**
1010	**4FF6**
1011	**57F5**
1100	**5FF4**
1101	**67F3**
1110	**6FF2**
1111	**77F1**

ТАБЛИЦА **LUT B** (адрес – число со знаком)	
АДРЕС	СОДЕРЖИМОЕ ЯЧЕЙКИ (hex)
0000	0000
0001	07FF
0010	0FFE
0011	17FD
0100	1FFC
0101	27FB
0110	2FFA
0111	37F9
1000	**C008**
1001	**C807**
1010	**D006**
1011	**D805**
1100	**E004**
1101	**E803**
1110	**F002**
1111	**F801**

X = 32570 (7F3Ah)

```
    4 F F 6   1010 (LUT A)
+ 1 7 F D     0011 (LUT A)
+ 7 7 F 1     1111 (LUT A)
+ 3 7 F 9     0111 (LUT B)
-------------
  3 F 9 5 0 C 6
```

3F950C6h = 66670790 = **32570 × 2047**

X = –30192 (8A10h)

```
    0 0 0 0   0000 (LUT A)
+ 0 7 F F     0001 (LUT A)
+ 4 F F 6     1010 (LUT A)
+ C 0 0 8     1000 (LUT B)
-------------
  C 5 0 F 5 F 0
```

C50F5F0h = –61803024 = **–30192 × 2047**

Умножитель на константу (пример 2)

Const = –47 (FD1h)

ТАБЛИЦА **LUT A** (адрес – число без знака)	
АДРЕС	СОДЕРЖИМОЕ ЯЧЕЙКИ (hex)
0000	0000
0001	FFD1
0010	FFA2
0011	FF73
0100	FF44
0101	FF15
0110	FEE6
0111	FEB7
1000	**FE88**
1001	**FE59**
1010	**FE2A**
1011	**FDFB**
1100	**FDCC**
1101	**FD9D**
1110	**FD6E**
1111	**FD3F**

ТАБЛИЦА **LUT B** (адрес – число со знаком)	
АДРЕС	СОДЕРЖИМОЕ ЯЧЕЙКИ (hex)
0000	0000
0001	FFD1
0010	FFA2
0011	FF73
0100	FF44
0101	FF15
0110	FEE6
0111	FEB7
1000	**0178**
1001	**0149**
1010	**011A**
1011	**00EB**
1100	**00BC**
1101	**008D**
1110	**005E**
1111	**002F**

X = 32570 (7F3Ah)

	F	F	F	F	E	2	A	1010 (LUT A)
+	F	F	F	F	7	3		0011 (LUT A)
+	F	F	D	3	F			1111 (LUT A)
+	F	E	B	7				0111 (LUT B)
	F	E	8	A	4	5	A	

FE8A45Ah = –1530790 = **32570 × (– 47)**

X = –30192 (8A10h)

	0	0	0	0	0	0	0	0000 (LUT A)
+	F	F	F	F	D	1		0001 (LUT A)
+	F	F	E	2	A			1010 (LUT A)
+	0	1	7	8				1000 (LUT B)
	0	1	5	A	7	5	0	

1419024h = –61803024 = –**30192 × (–47)**

Устройство деления (алгоритм)

Для выполнения деления используется алгоритм с восстановлением остатка, позволяющий получить одновременно частное и остаток. Операция выполняется над двоичными положительными числами с фиксированной точкой. В случае чисел со знаком ввиду сложности алгоритма, рекомендуется исходные операнды сделать положительными, а результат деления откорректировать в зависимости от знака исходных операндов. Алгоритм содержит n итераций (по количеству разрядов делимого X), каждая из которых включает несколько шагов.

Начальная установка

Приравниваем частичный остаток $X'(0) = X$

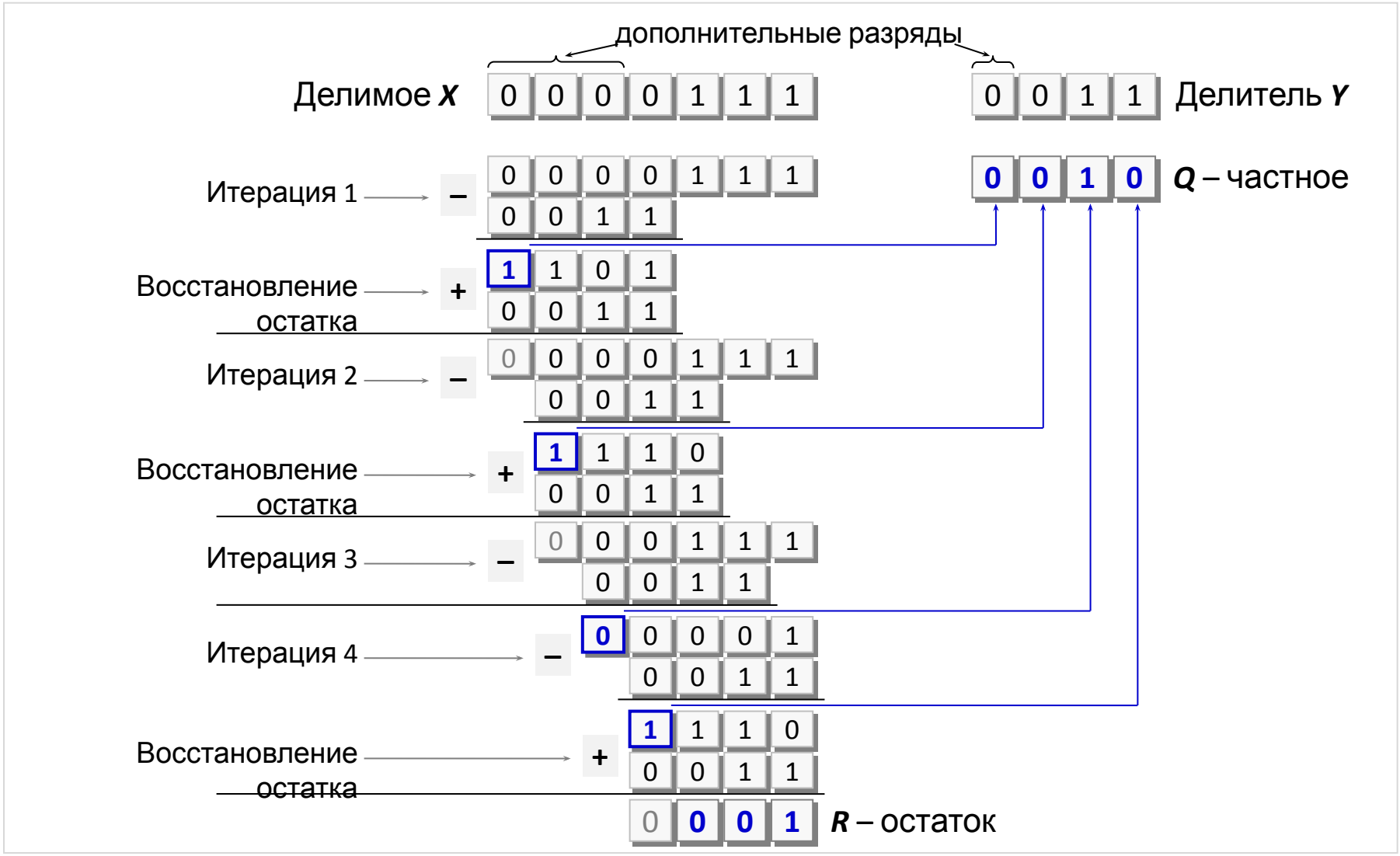
Итерация 1 (i=1)

Шаг 1. Определяем, содержится ли число $Y \times 2^{n-1-i}$ в частичном остатке $X'(i-1)$, для чего выполняем операцию вычитания $X'(i-1) - Y \times 2^{n-1-i}$ и анализируем знак полученной разности. Если результат положителен (знаковый разряд 0 – число содержится в частичном остатке), записываем цифру частного $Q_{n-1}=1$. Если результат вычитания отрицателен (знаковый разряд 1 – число в частичном остатке не содержится), записываем цифру частного $Q_{n-1}=0$.

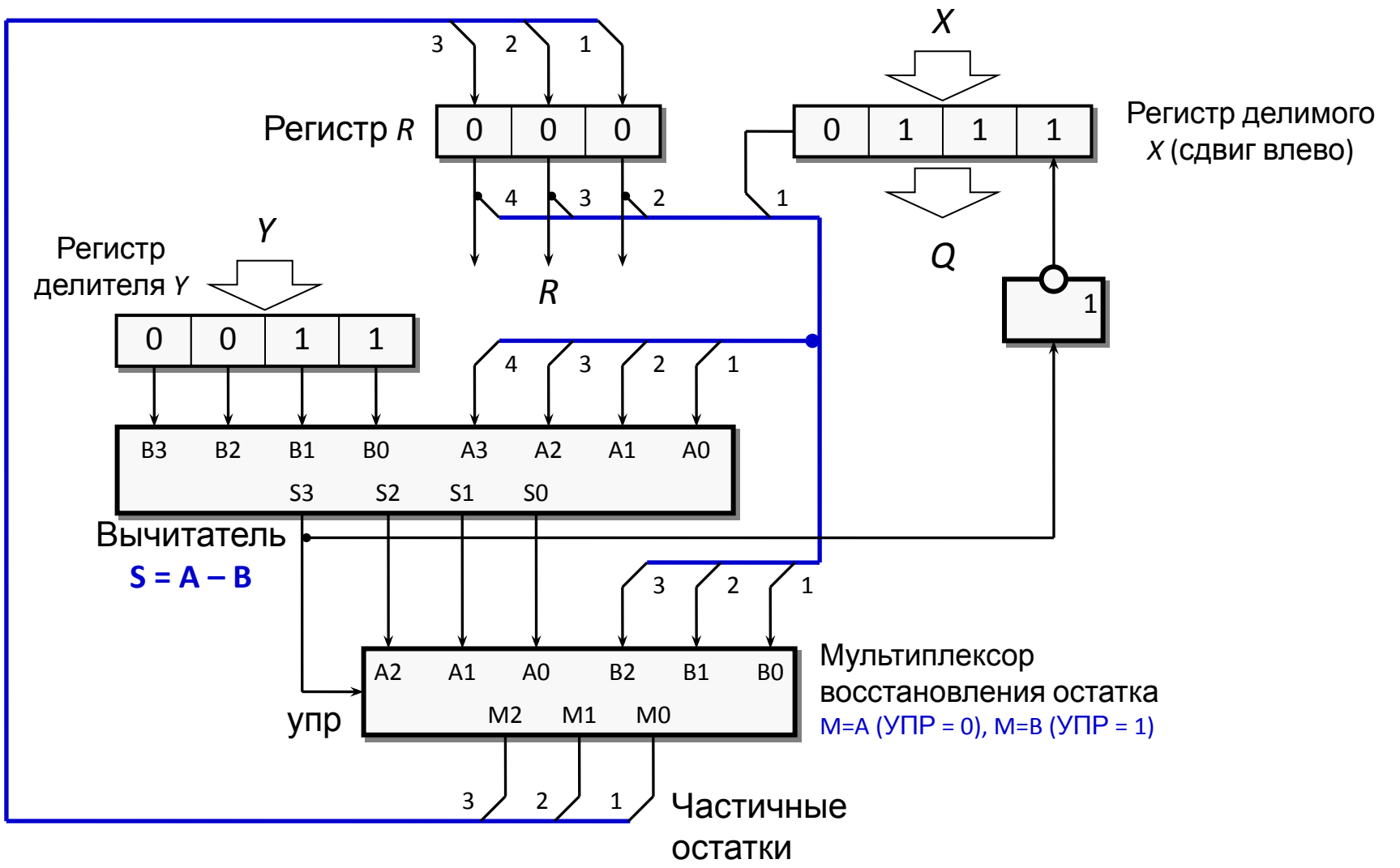
Шаг2. Если разность $X'(i-1) - Y \times 2^{n-1-i}$ положительна, формируем новый частичный остаток, равный этой разности $X'(i) = X'(i-1) - Y \times 2^{n-1-i}$. Если разность отрицательна, то $X'(i) = X'(i-1) - Y \times 2^{n-1-i} + Y \times 2^{n-1-i} = X'(i-1)$. Этот шаг называется ***восстановлением остатка*** (то, что вычли, снова прибавили).

Итерации 2...n. Повторяются шаги 1 и 2, формируется последняя цифра частного Q_0 и остаток операции деления $R = X'(n-1)$.

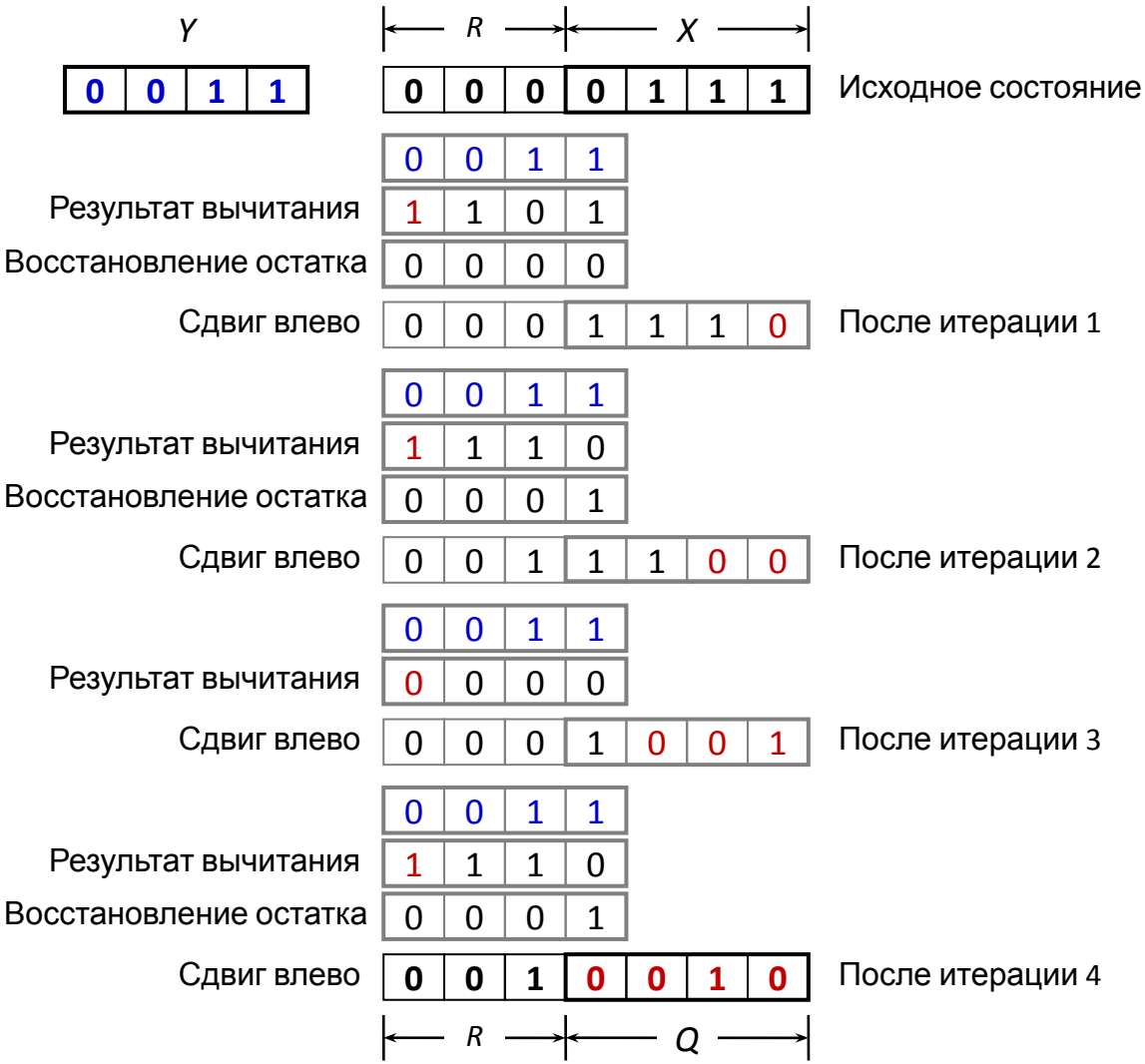
Устройство деления (выполнение операции)



Устройство деления (структура)



Устройство деления (пример)



Вычислительные блоки цифровых фильтров

Общая структура КИХ-фильтра

Уравнения

k – порядок фильтра;
N = k+1 – размерность импульсной характеристики

Сверточная сумма

$$y(n) = \sum_{i=0}^{k} x(n-i)h(i)$$

Векторная форма

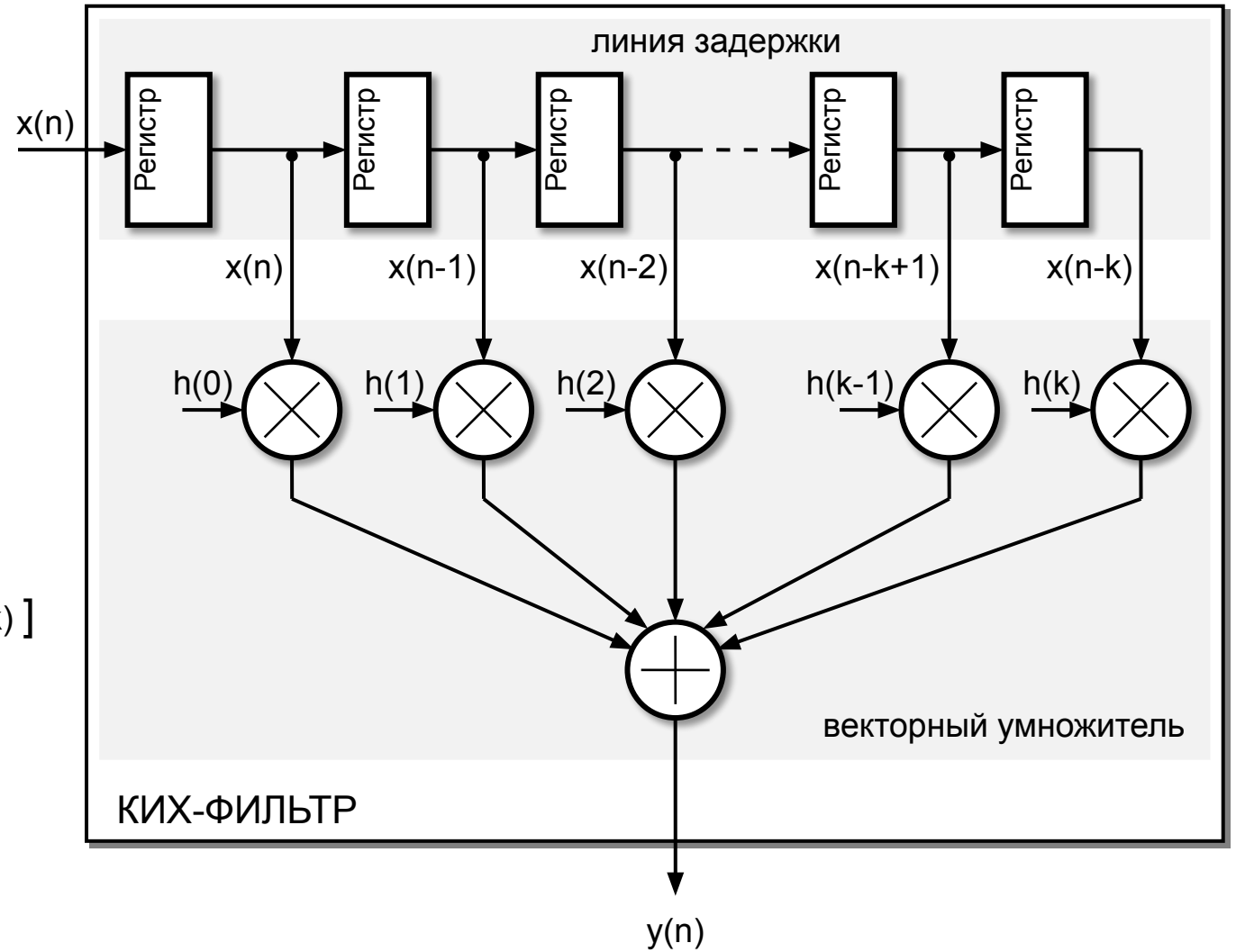
Y = X × H

Вектор – строка задержанных отсчетов сигнала

X = [x(n) x(n-1) ... x(n-k)]

Вектор – столбец коэффициентов фильтра

$$\mathbf{H} = \begin{bmatrix} h(0) \\ h(1) \\ h(2) \\ \dots \\ h(k) \end{bmatrix}$$



Симметрия коэффициентов N - четное

N = 8

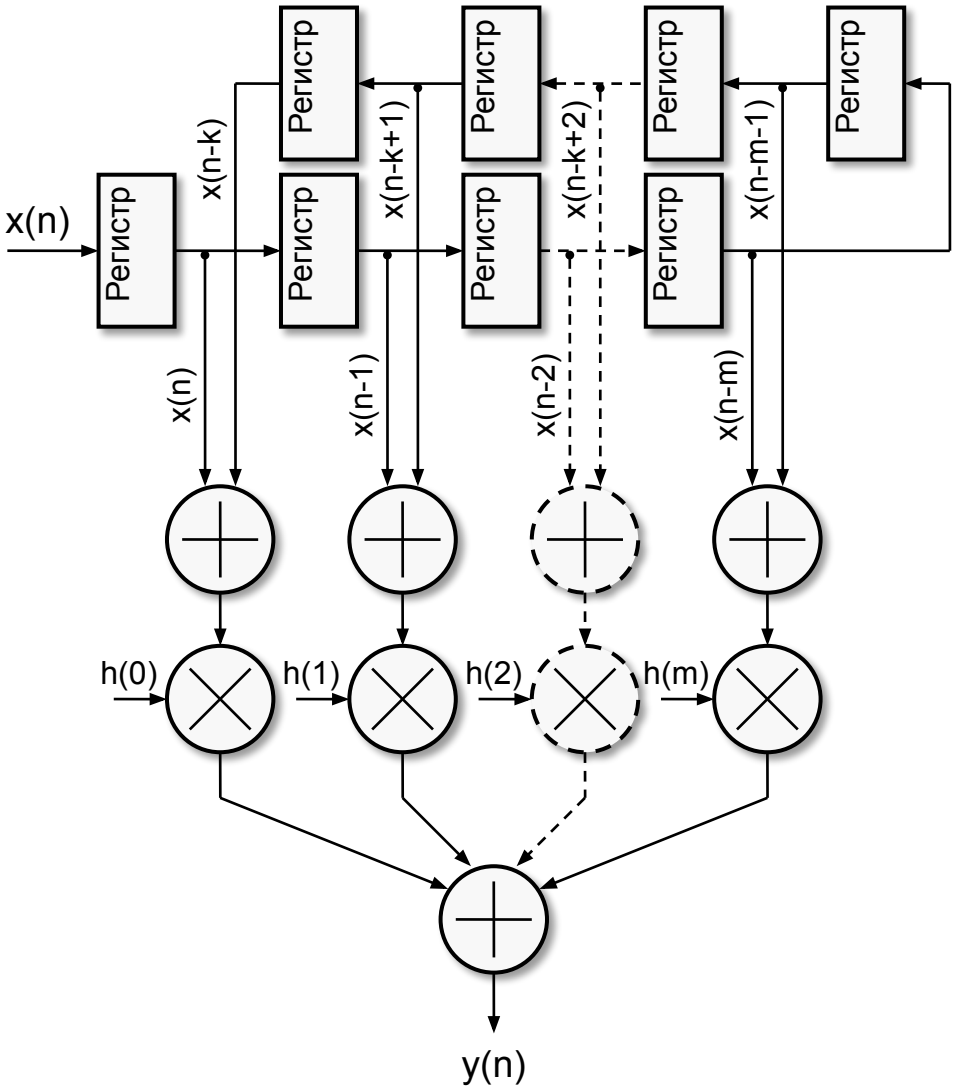
h(0) = ±h(7), h(1) = ±h(6), h(2) = ±h(5), h(3) = ±h(4)

y(7) = h(0)*x(7) + h(1)*x(6) + h(2)*x(5) + h(3)*x(4) +
+ h(4)*x(3) + h(5)*x(2) + h(6)*x(1) + h(7)*x(0) =
= h(0)*x(7) + h(1)*x(6) + h(2)*x(5) + h(3)*x(4) +
± h(3)*x(3) ± h(2)*x(2) ± h(1)*x(1) ± h(0)*x(0) =
= h(0)*[x(7)±x(0)] + h(1)*[x(6)±x(1)] +
+ h(2)*[x(5)±x(2)] + h(3)*[x(4)±x(3)]

N произвольное ЧЕТНОЕ

h(0) = ±h(N-1), ...,
h(m) = ±h(N-m-1), ...,
h(N/2-1) = ±h(N/2)

y(N-1) = h(0)*[x(N-1)±x(0)] + h(1)*[x(N-2)±x(1)] +
+ . . . +
+ h(m)*[x(N-m-1)±x(m)] +
+ . . . +
+ h(N/2-1)*[x(N/2)±x(N/2-1)]



Симметрия коэффициентов N - нечетное

N = 7

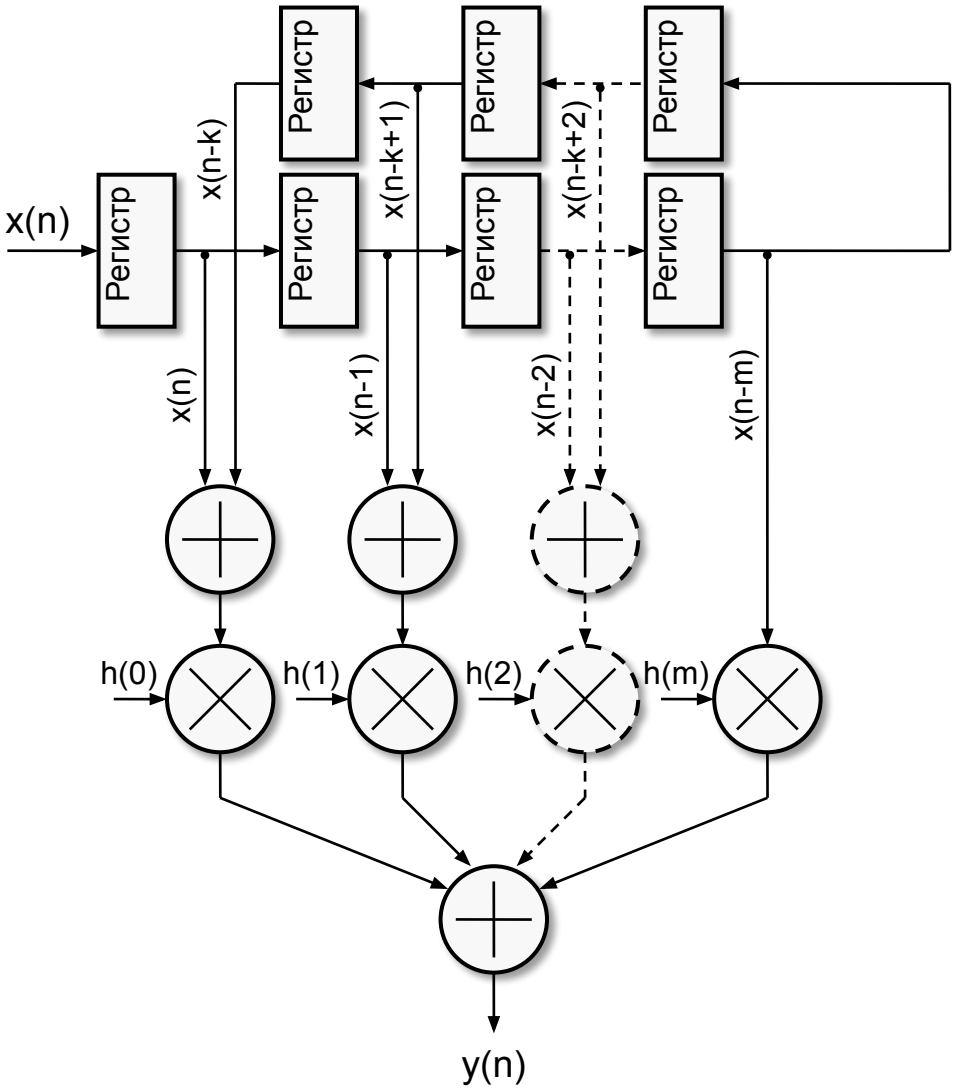
h(0) = ±h(6), h(1) = ±h(5), h(2) = ±h(4), h(3)

y(7) = h(0)*x(6) + h(1)*x(5) + h(2)*x(4) + **h(3)*x(3)** +
+ h(4)*x(2) + h(5)*x(1) + h(6)*x(0) =
= h(0)*x(6) + h(1)*x(6) + h(2)*x(4) + **h(3)*x(3)** +
± h(3)*x(2) ± h(2)*x(1) ± h(1)*x(0) =
= h(0)*[x(6)±x(0)] + h(1)*[x(5)±x(1)] +
+ h(2)*[x(4)±x(2)] + **h(3)*[x(3)]**

N произвольное НЕЧЕТНОЕ

h(0) = ±h(N-1), ..., h(m) = ±h(N-m-1), ..., **h((N-1)/2)**

y(N-1) = h(0)*[x(N-1)±x(0)] + h(1)*[x(N-2)±x(1)] +
+ . . . +
+ h(m)*[x(N-m-1)±x(m)] +
+ . . . +
+ **h((N-1)/2)*[x((N-1)/2)]**



Векторный умножитель – распределенная арифметика

Симметрия коэффициентов фильтра

$s(0) = x(n)\pm x(n-k);\ \ s(1) = x(n-1)\pm x(n-k+1);\ s(2) = x(n-2)\pm x(n-k+2);\ \ s(3) = x(n-3)\pm x(n-k+3);$

Векторный умножитель с размерностью вектора 4

$$y(n)= s(0)\times h(0) + s(1)\times h(1) + s(2)\times h(2) + s(3)\times h(3)$$

Распределенная (поразрядная или битовая) **арифметика**

$s_i(n)$ – *i*-й двоичный разряд операнда *n*

$$\begin{aligned} y(n) = & [\, -s_{m-1}(0)\times 2^{m-1} + s_{m-2}(0)\times 2^{m-2} + \ldots + s_1(0)\times 2^1 + s_0(0)\times 2^0 \,] \times h(0) + \\ & + [\, -s_{m-1}(1)\times 2^{m-1} + s_{m-2}(1)\times 2^{m-2} + \ldots + s_1(1)\times 2^1 + s_0(1)\times 2^0 \,] \times h(1) + \\ & + [\, -s_{m-1}(2)\times 2^{m-1} + s_{m-2}(2)\times 2^{m-2} + \ldots + s_1(2)\times 2^1 + s_0(2)\times 2^0 \,] \times h(2) + \\ & + [\, -s_{m-1}(3)\times 2^{m-1} + s_{m-2}(3)\times 2^{m-2} + \ldots + s_1(3)\times 2^1 + s_0(3)\times 2^0 \,] \times h(3) \end{aligned}$$

$$\begin{aligned} y(n) = & -[\, s_{m-1}(0)\times h(0) + s_{m-1}(1)\times h(1) + s_{m-1}(2)\times h(2) + s_{m-1}(3)\times h(3) \,] \times 2^{m-1} + \\ & + [\, s_{m-2}(0)\times h(0) + s_{m-2}(1)\times h(1) + s_{m-2}(2)\times h(2) + s_{m-2}(3)\times h(3) \,] \times 2^{m-2} + \\ & + \ldots + \\ & + [\, s_1(0)\times h(0) + s_1(1)\times h(1) + s_1(2)\times h(2) + s_1(3)\times h(3) \,] \times 2^1 + \\ & + [\, s_0(0)\times h(0) + s_0(1)\times h(1) + s_0(2)\times h(2) + s_0(3)\times h(3) \,] \times 2^0 \end{aligned}$$

Примечание. Выражение в [.] вычисляется с помощью таблицы LUT.

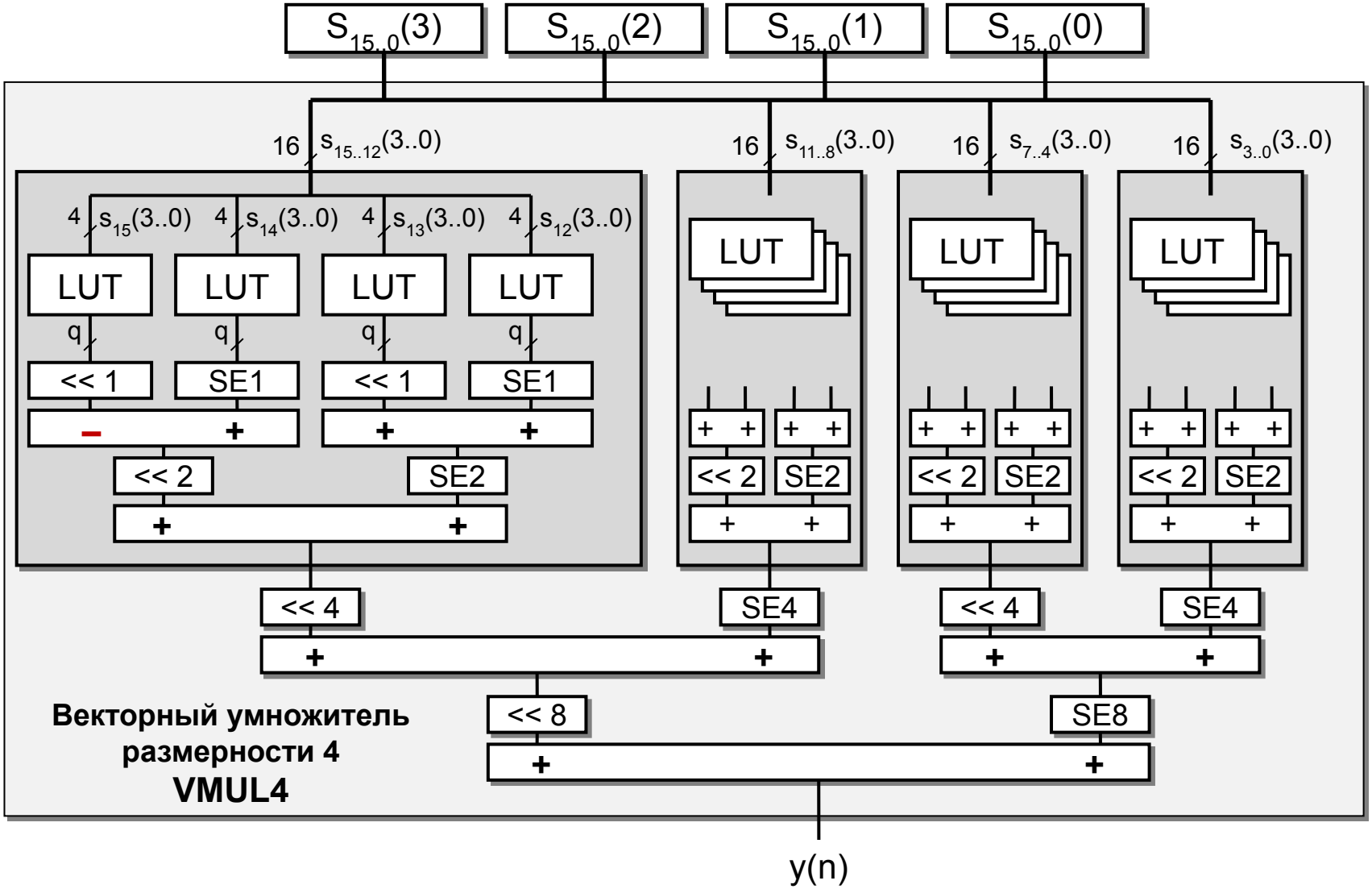
Содержимое LUT векторного умножителя фильтра

Вычисляется выражение

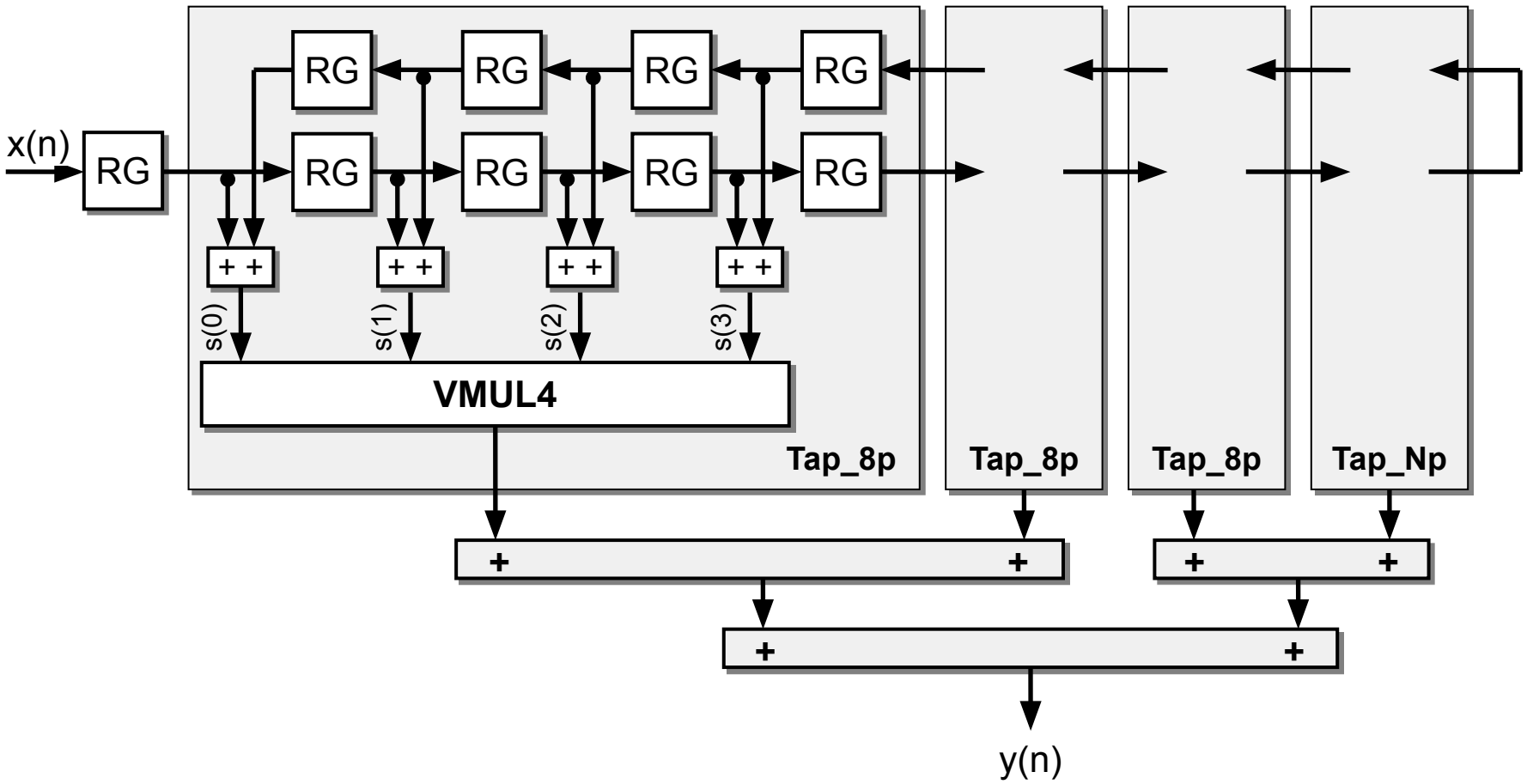
LUT = [$s_m(0)\times h(0) + s_m(1)\times h(1) + s_m(2)\times h(2) + s_m(3)\times h(3)$],
где $s_m(n)$ – *m*-й двоичный разряд операнда *n*

№ ячейки	Разряд (адрес LUT)				СОДЕРЖИМОЕ ЯЧЕЙКИ
	$s_m(3)$	$s_m(2)$	$s_m(1)$	$s_m(0)$	
0	0	0	0	0	0×h(3) + 0×h(2) + 0×h(1) + 0×h(0) = **0**
1	0	0	0	**1**	0×h(3) + 0×h(2) + 0×h(1) + 1×h(0) = **h(0)**
2	0	0	**1**	0	0×h(3) + 0×h(2) + 1×h(1) + 0×h(0) = **h(1)**
3	0	0	**1**	**1**	0×h(3) + 0×h(2) + 1×h(1) + 1×h(0) = **h(1) + h(0)**
4	0	**1**	0	0	0×h(3) + 1×h(2) + 0×h(1) + 0×h(0) = **h(2)**
5	0	**1**	0	**1**	0×h(3) + 1×h(2) + 0×h(1) + 1×h(0) = **h(2) + h(0)**
6	0	**1**	**1**	0	0×h(3) + 1×h(2) + 1×h(1) + 0×h(0) = **h(2) + h(1)**
7	0	**1**	**1**	**1**	0×h(3) + 1×h(2) + 1×h(1) + 1×h(0) = **h(2) + h(1) + h(0)**
8	**1**	0	0	0	1×h(3) + 0×h(2) + 0×h(1) + 0×h(0) = **h(3)**
9	**1**	0	0	**1**	1×h(3) + 0×h(2) + 0×h(1) + 1×h(0) = **h(3) + h(0)**
10	**1**	0	**1**	0	1×h(3) + 0×h(2) + 1×h(1) + 0×h(0) = **h(3) + h(1)**
11	**1**	0	**1**	**1**	1×h(3) + 0×h(2) + 1×h(1) + 1×h(0) = **h(3) + h(1) + h(0)**
12	**1**	**1**	0	0	1×h(3) + 1×h(2) + 0×h(1) + 0×h(0) = **h(3) + h(2)**
13	**1**	**1**	0	**1**	1×h(3) + 1×h(2) + 0×h(1) + 1×h(0) = **h(3) + h(2) + h(0)**
14	**1**	**1**	**1**	0	1×h(3) + 1×h(2) + 1×h(1) + 0×h(0) = **h(3) + h(2) + h(1)**
15	**1**	**1**	**1**	**1**	1×h(3) + 1×h(2) + 1×h(1) + 1×h(0) = **h(3) + h(2) + h(1) + h(0)**

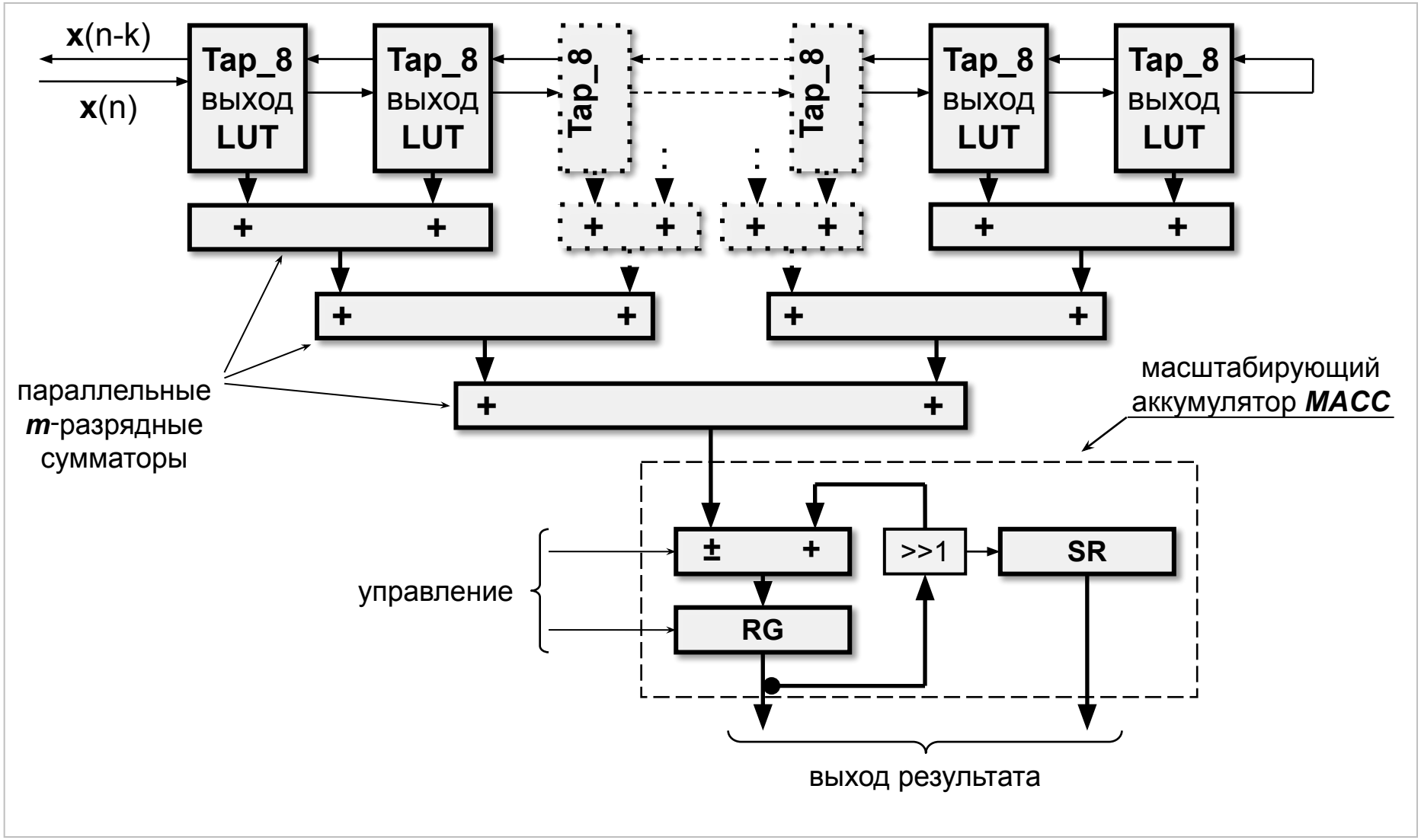
Векторный умножитель – параллельная распределенная арифметика



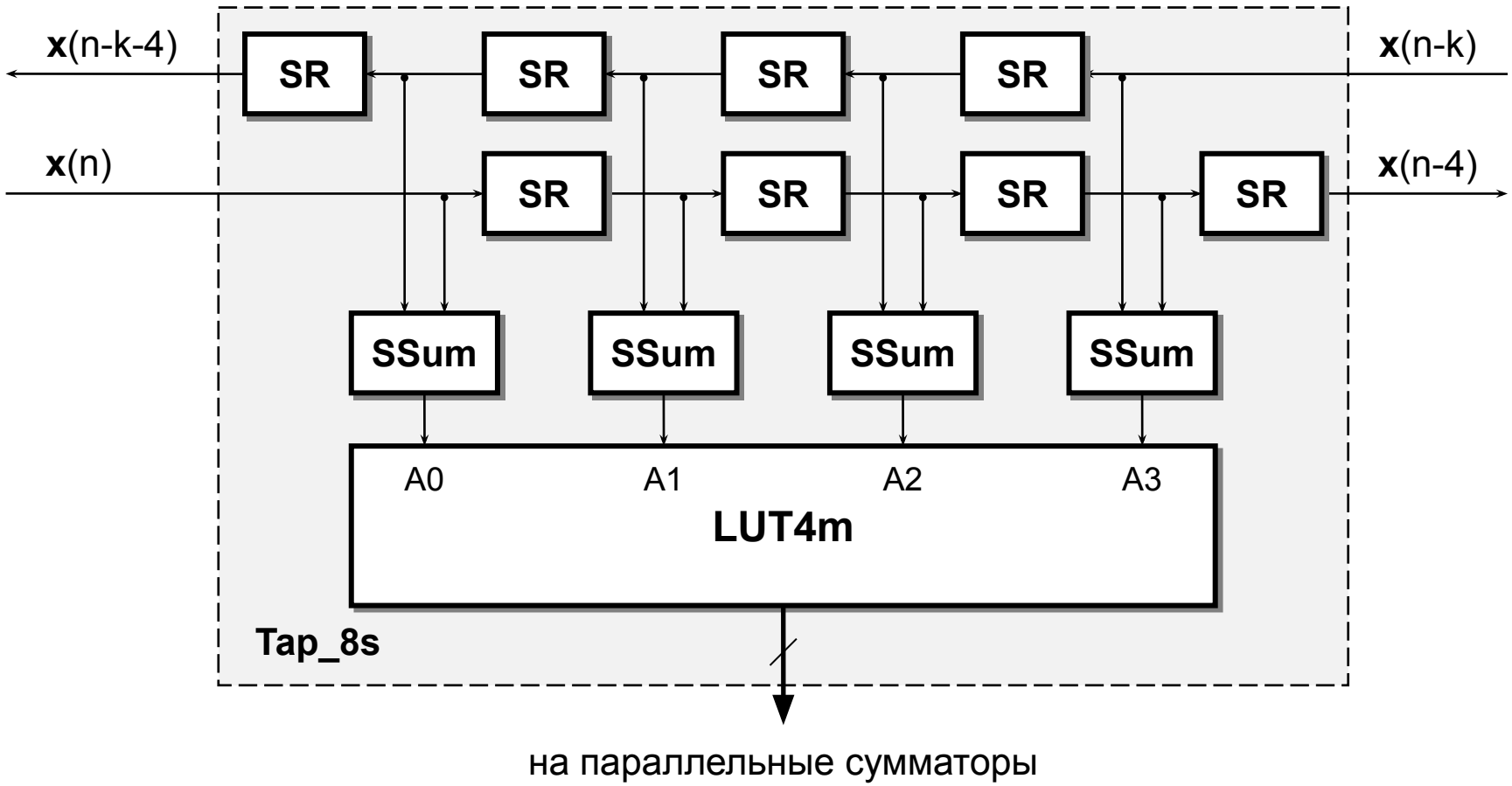
КИХ-фильтр – параллельная распределенная арифметика



КИХ-фильтр – последовательная распределенная арифметика

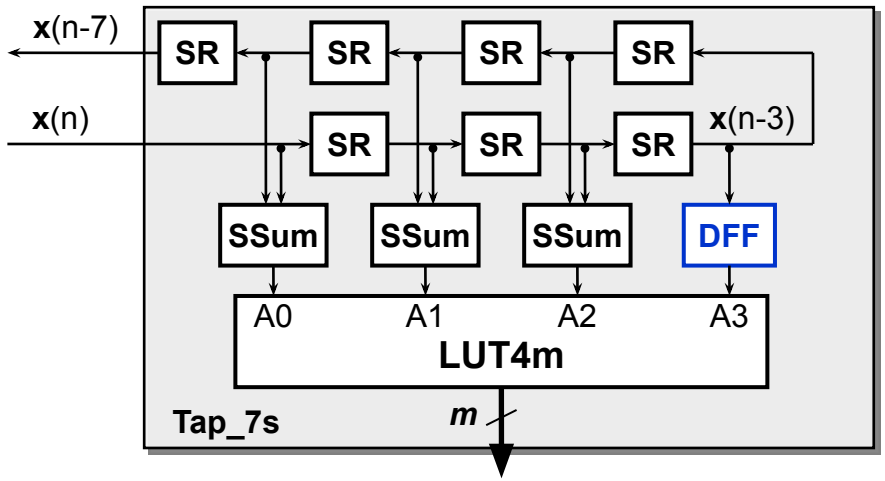


Последовательная распределенная арифметика – TAP_8s

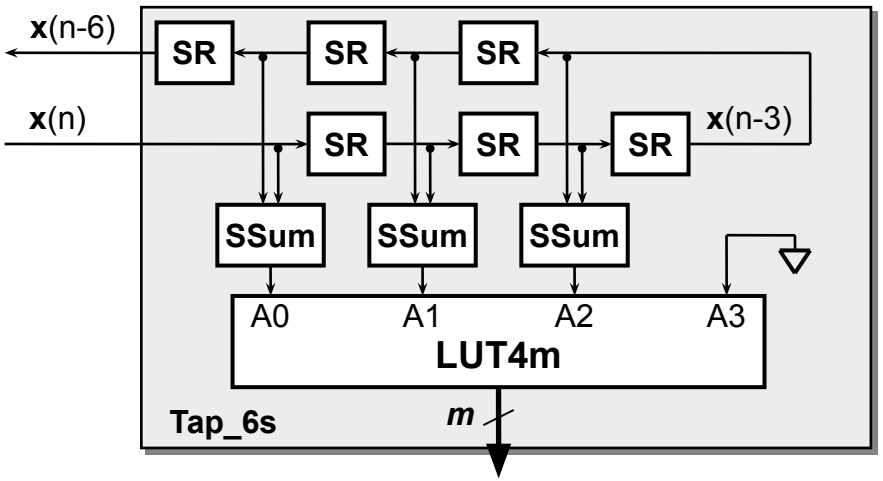


Последовательная распределенная арифметика – примеры

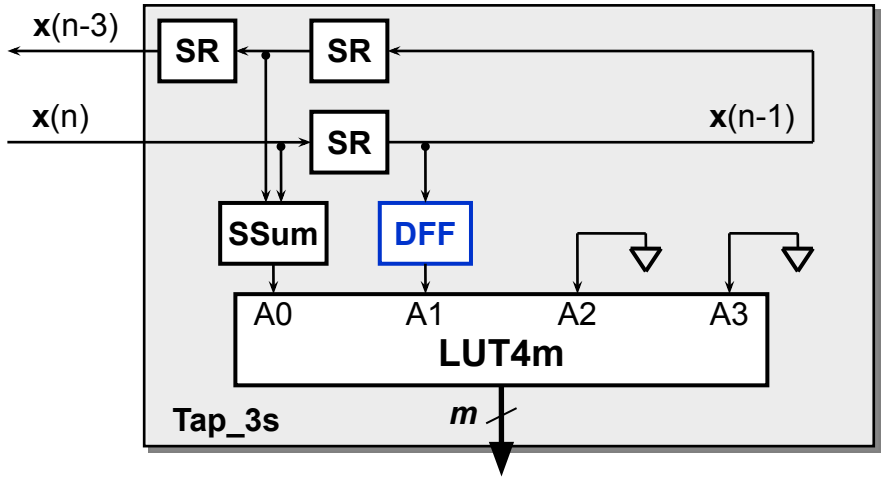
TAP_Ns



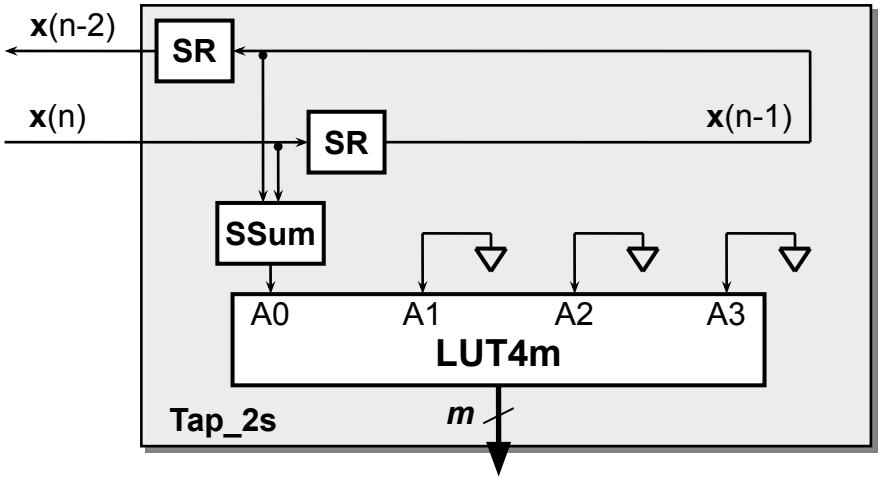
на параллельные сумматоры



на параллельные сумматоры

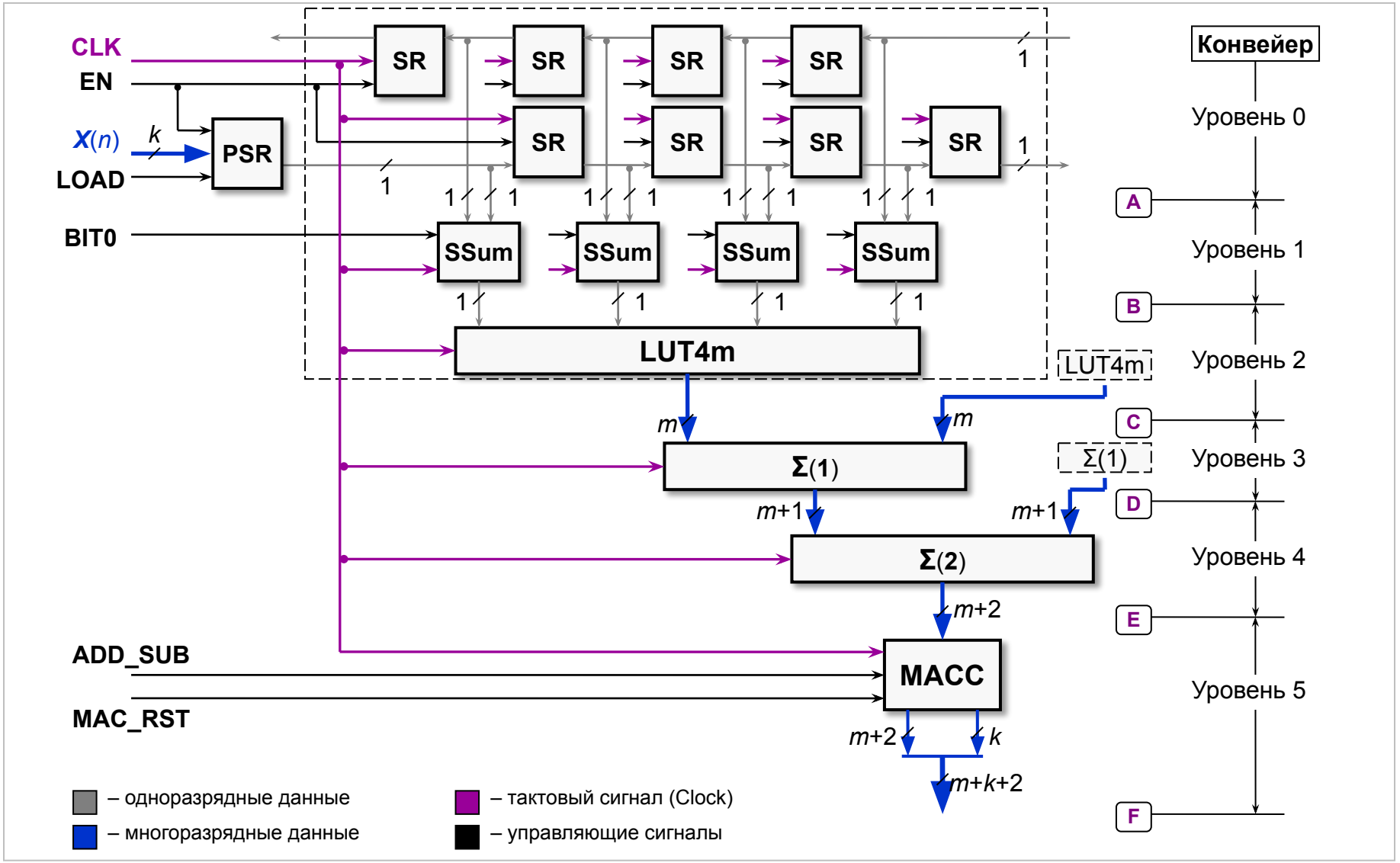


на параллельные сумматоры

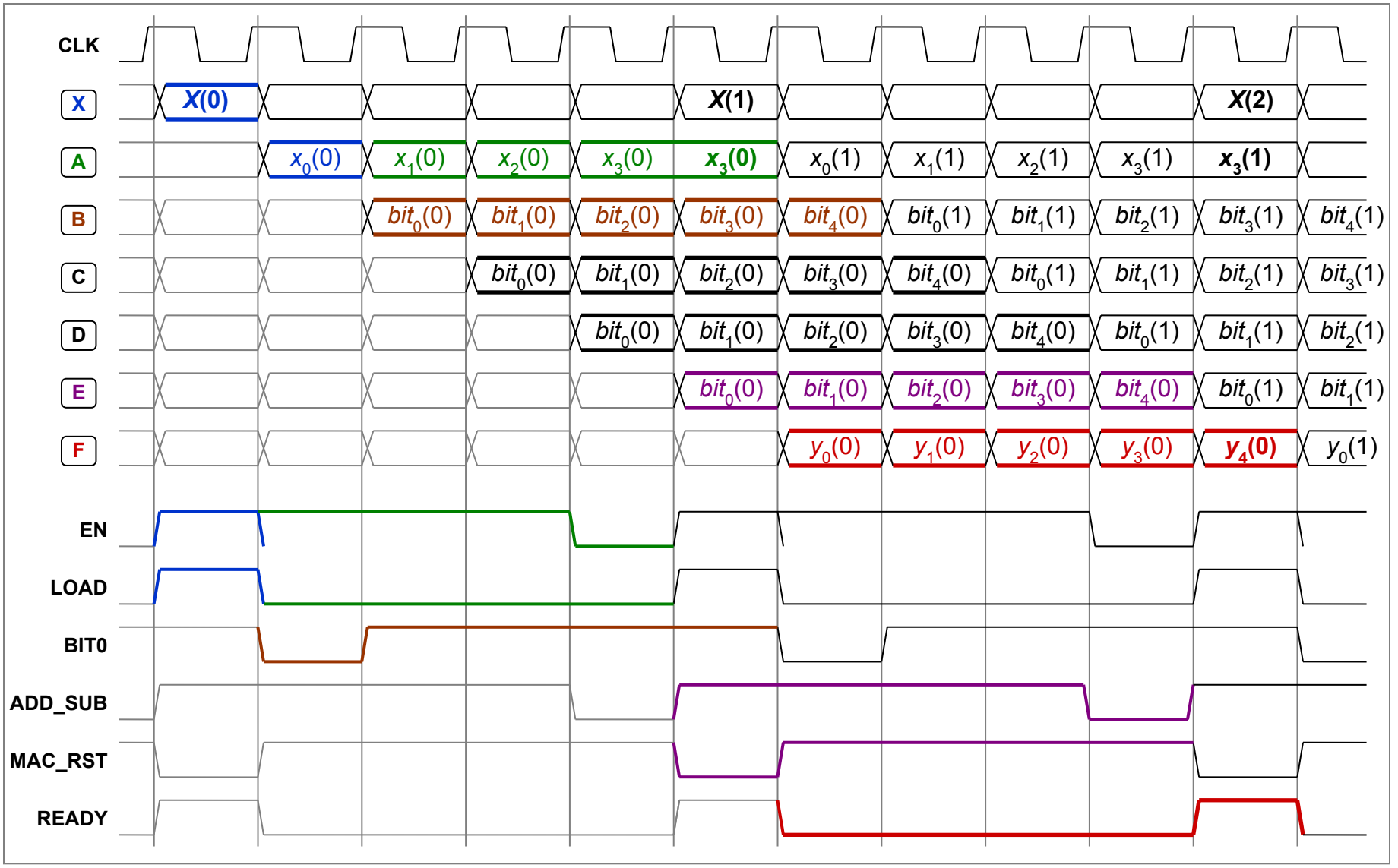


на параллельные сумматоры

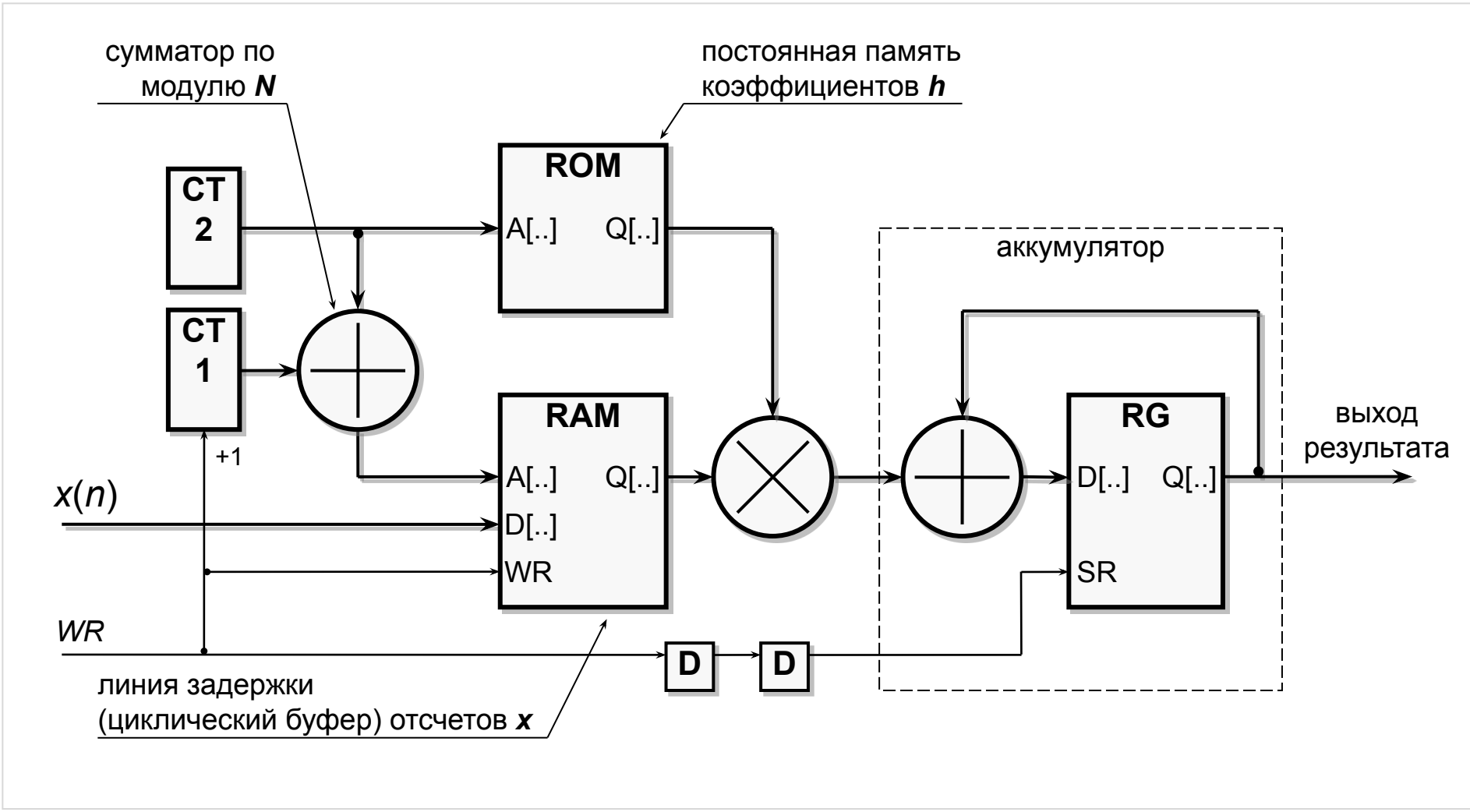
Потоки данных фильтра – последовательно-параллельная обработка



Потоки данных фильтра и сигналы управления



КИХ-фильтр на основе циклического буфера – структура

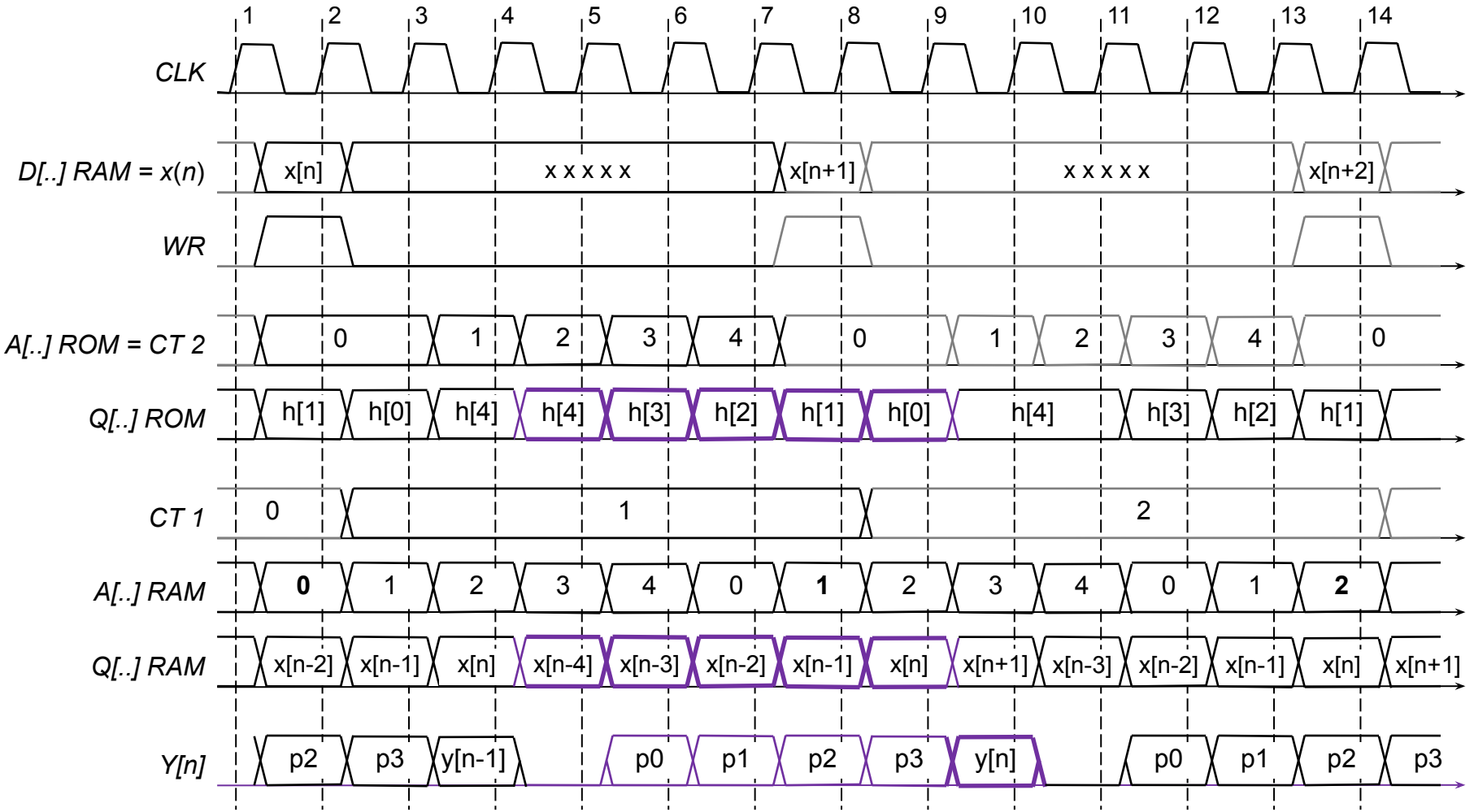


КИХ-фильтр на основе циклического буфера – работа

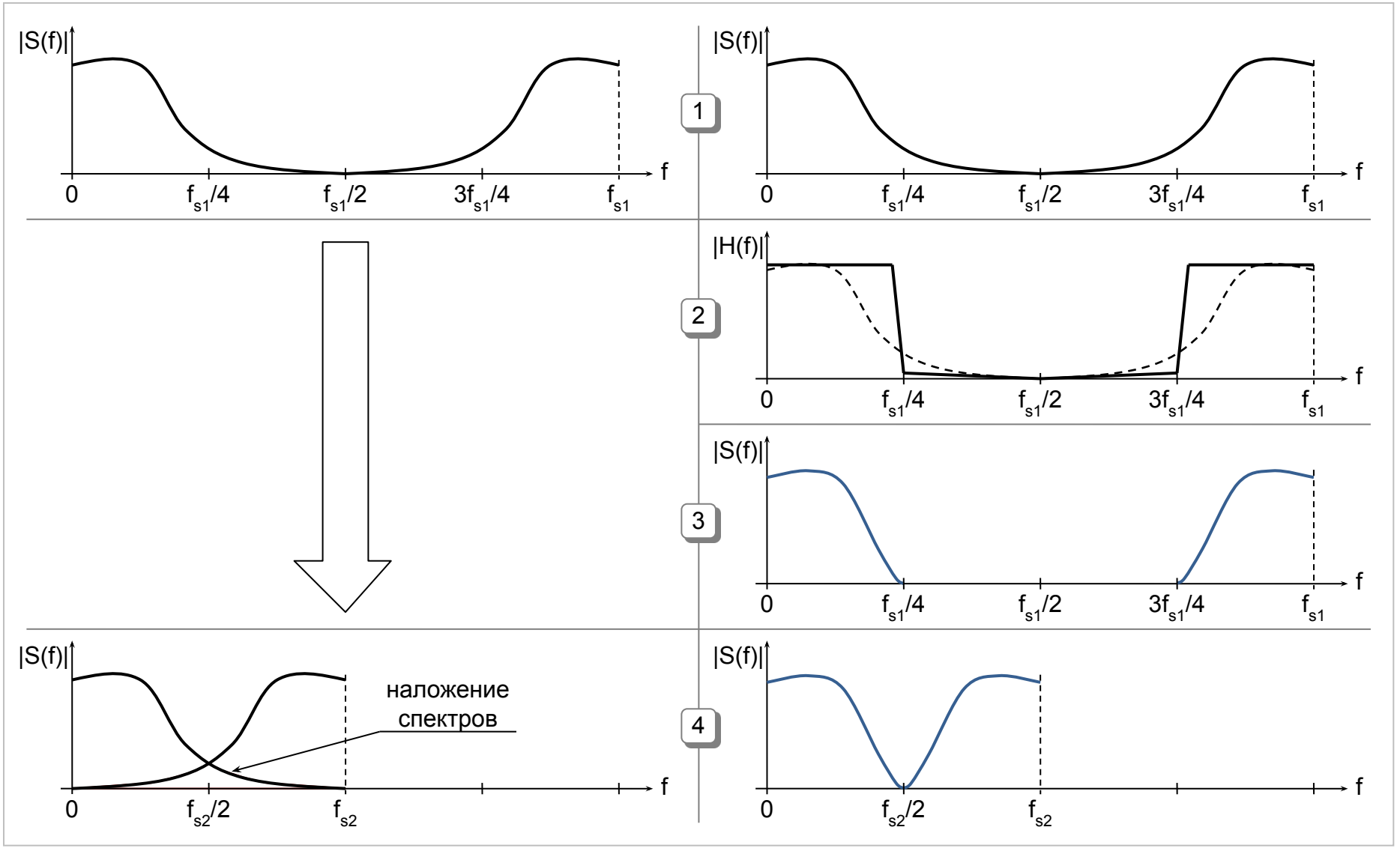
Таблица Адресация памяти для случая $N = 5$

		Запись	**Номер такта считывания (CT2)**				
Время			0	1	2	3	4
	Адрес ROM		0	1	2	3	4
	Содержимое ROM		$h(4)$	$h(3)$	$h(2)$	$h(1)$	$h(0)$
n	Адрес RAM	**0**	1	2	3	4	0
	Содержимое RAM	$\boldsymbol{x(n)}$	0	0	0	0	$x(n)$
n+1	Адрес RAM	**1**	2	3	4	0	1
	Содержимое RAM	$\boldsymbol{x(n+1)}$	0	0	0	$x(n)$	$x(n+1)$
n+2	Адрес RAM	**2**	3	4	0	1	2
	Содержимое RAM	$\boldsymbol{x(n+2)}$	0	0	$x(n)$	$x(n+1)$	$x(n+2)$
n+3	Адрес RAM	**3**	4	0	1	2	3
	Содержимое RAM	$\boldsymbol{x(n+3)}$	0	x(n)	$x(n+1)$	$x(n+2)$	$x(n+3)$
n+4	Адрес RAM	**4**	0	1	2	3	4
	Содержимое RAM	$\boldsymbol{x(n+4)}$	$x(n)$	x(n+1)	$x(n+2)$	$x(n+3)$	$x(n+4)$
n+5	Адрес RAM	**0**	1	2	3	4	0
	Содержимое RAM	$\boldsymbol{x(n+5)}$	$x(n+1)$	$x(n+2)$	$x(n+3)$	$x(n+4)$	$x(n+5)$
n+6	Адрес RAM	**1**	2	3	4	0	1
	Содержимое RAM	$\boldsymbol{x(n+6)}$	$x(n+2)$	$x(n+3)$	$x(n+4)$	$x(n+5)$	$x(n+6)$
. . .							

КИХ-фильтр на основе циклического буфера – диаграмма



Полифазный децимирующий фильтр (спектры)



Полифазный децимирующий фильтр (уравнения)

Коэффициент децимации **R = 3**

y(n+ 0) = h(0)*x(n+ 0) + h(1)*x(n - 1) + h(2)*x(n - 2) + h(3)*x(n - 3) + h(4)*x(n - 4) + h(5)*x(n - 5) + h(6)*x(n - 6) + ...
y(n+ 3) = h(0)*x(n+ 3) + h(1)*x(n+ 2) + h(2)*x(n+ 1) + h(3)*x(n+ 0) + h(4)*x(n - 1) + h(5)*x(n - 2) + h(6)*x(n - 3) + ...
y(n+ 6) = h(0)*x(n+ 6) + h(1)*x(n+ 5) + h(2)*x(n+ 4) + h(3)*x(n+ 3) + h(4)*x(n+ 2) + h(5)*x(n+ 1) + h(6)*x(n+ 0) + ...
y(n+ 9) = h(0)*x(n+ 9) + h(1)*x(n+ 8) + h(2)*x(n+ 7) + h(3)*x(n+ 6) + h(4)*x(n+ 5) + h(5)*x(n+ 4) + h(6)*x(n+ 3) + ...
y(n+12) = h(0)*x(n+12) + h(1)*x(n+11) + h(2)*x(n+10) + h(3)*x(n+ 9) + h(4)*x(n+ 8) + h(5)*x(n+ 7) + h(6)*x(n+ 6) + ...
y(n+15) = h(0)*x(n+15) + h(1)*x(n+14) + h(2)*x(n+13) + h(3)*x(n+12) + h(4)*x(n+11) + h(5)*x(n+10) + h(6)*x(n+ 9) + ...
y(n+18) = h(0)*x(n+18) + h(1)*x(n+17) + h(2)*x(n+16) + h(3)*x(n+15) + h(4)*x(n+14) + h(5)*x(n+13) + h(6)*x(n+12) + ...
y(n+21) = h(0)*x(n+21) + h(1)*x(n+20) + h(2)*x(n+19) + h(3)*x(n+18) + h(4)*x(n+17) + h(5)*x(n+16) + h(6)*x(n+15) + ...

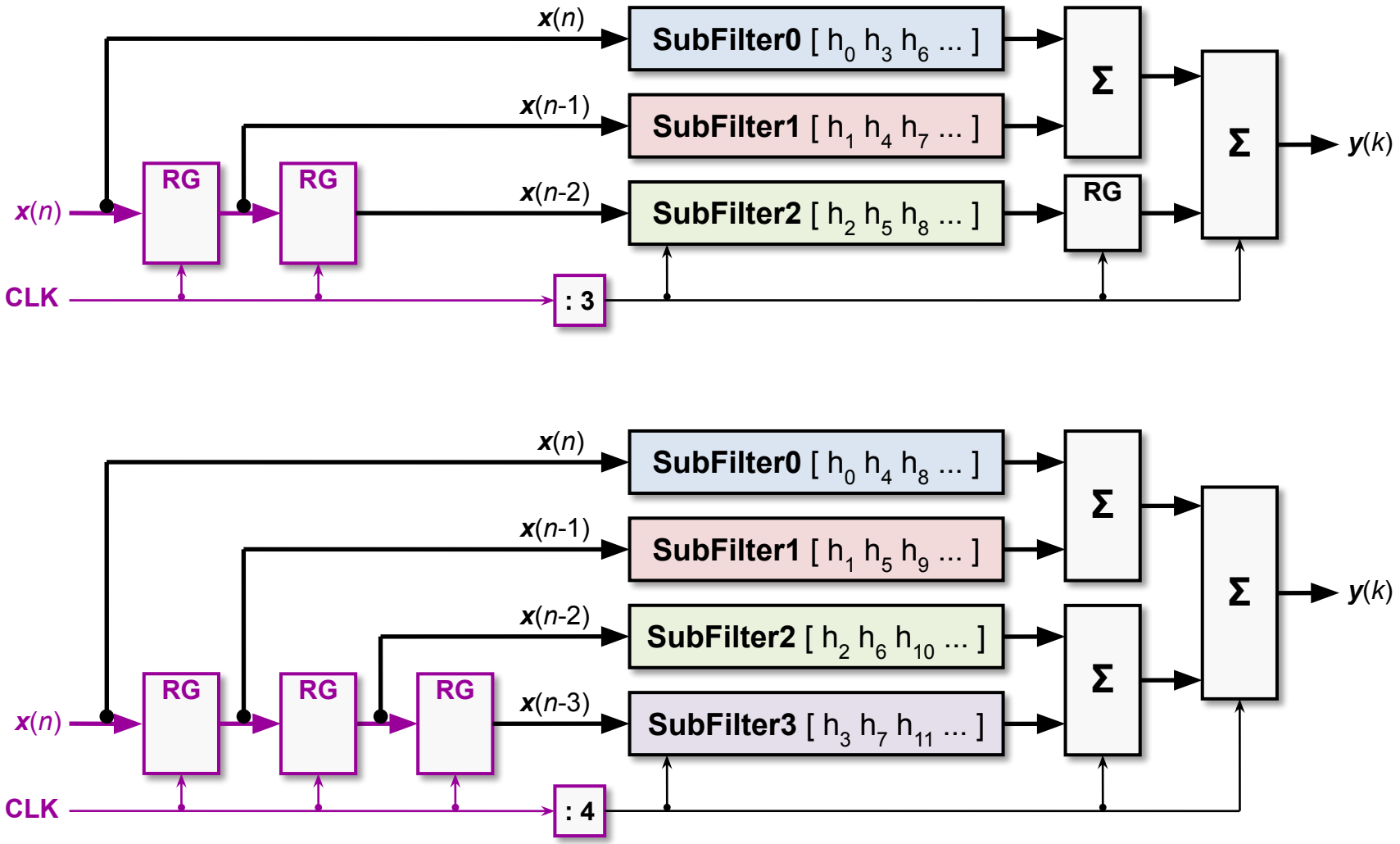
SubFilter2
SubFilter1
SubFilter0

Коэффициент децимации **R = 4**

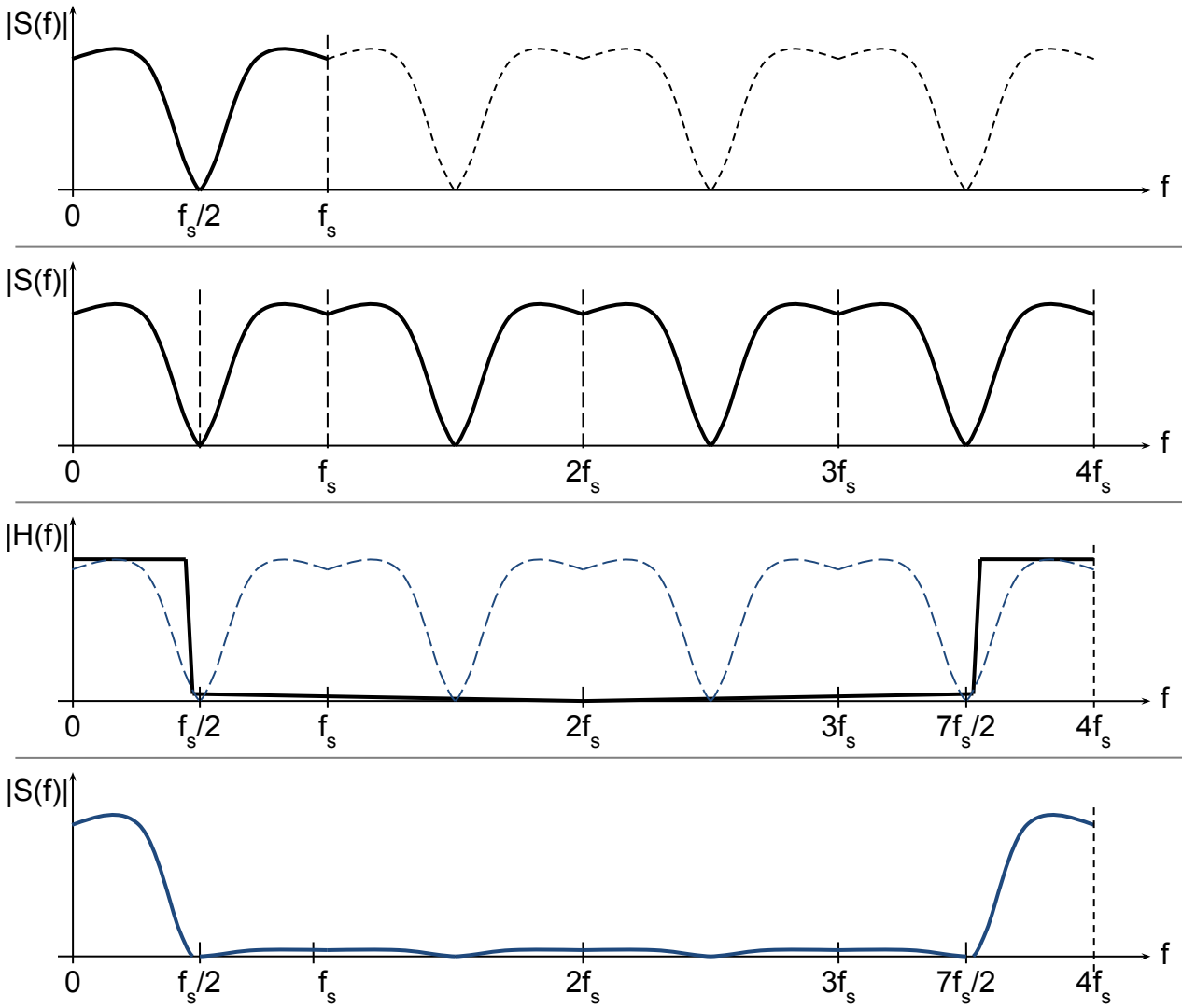
y(n+ 0) = h(0)*x(n+ 0) + h(1)*x(n - 1) + h(2)*x(n - 2) + h(3)*x(n - 3) + h(4)*x(n - 4) + h(5)*x(n - 5) + h(6)*x(n - 6) + ...
y(n+ 4) = h(0)*x(n+ 4) + h(1)*x(n+ 3) + h(2)*x(n+ 2) + h(3)*x(n+ 1) + h(4)*x(n+ 0) + h(5)*x(n - 1) + h(6)*x(n - 2) + ...
y(n+ 8) = h(0)*x(n+ 8) + h(1)*x(n+ 7) + h(2)*x(n+ 6) + h(3)*x(n+ 5) + h(4)*x(n+ 4) + h(5)*x(n+ 3) + h(6)*x(n+ 2) + ...
y(n+12) = h(0)*x(n+12) + h(1)*x(n+11) + h(2)*x(n+10) + h(3)*x(n+ 9) + h(4)*x(n+ 8) + h(5)*x(n+ 7) + h(6)*x(n+ 6) + ...
y(n+16) = h(0)*x(n+16) + h(1)*x(n+15) + h(2)*x(n+14) + h(3)*x(n+13) + h(4)*x(n+12) + h(5)*x(n+11) + h(6)*x(n+10) + ...
y(n+20) = h(0)*x(n+20) + h(1)*x(n+19) + h(2)*x(n+18) + h(3)*x(n+17) + h(4)*x(n+16) + h(5)*x(n+15) + h(6)*x(n+14) + ...
y(n+24) = h(0)*x(n+24) + h(1)*x(n+23) + h(2)*x(n+22) + h(3)*x(n+21) + h(4)*x(n+20) + h(5)*x(n+19) + h(6)*x(n+18) + ...
y(n+28) = h(0)*x(n+28) + h(1)*x(n+27) + h(2)*x(n+26) + h(3)*x(n+25) + h(4)*x(n+24) + h(5)*x(n+23) + h(6)*x(n+22) + ...

SubFilter3
SubFilter2
SubFilter1
SubFilter0

Полифазный децимирующий фильтр (структуры)



Полифазный интерполирующий фильтр (спектры)



1. Спектр исходного дискретного сигнала с частотой дискретизации $\mathbf{f_s}$
2. Спектр сигнала после дополнения нулями (три нуля между каждой парой отсчетов, **M=4**)
3. Частотная характеристика интерполирующего фильтра
4. Спектр выходного сигнала после интерполяции с M=4 (частота $\mathbf{4f_s}$)

Полифазный интерполирующий фильтр (уравнения)

Коэффициент интерполяции **M = 3**

y(k+ 0) = **h(0)*x(n+0)** + h(1)*0 + h(2)*0 + **h(3)*x(n - 1)** + h(4)*0 + h(5)*0 + **h(6)*x(n - 2)** + h(7)*0 + h(8)*0 + **h(9)*x(n - 3)** + ...

y(k+ 1) = h(0)*0 + **h(1)*x(n+0)** + h(2)*0 + h(3)*0 + **h(4)*x(n - 1)** + h(5)*0 + h(6)*0 + **h(7)*x(n - 2)** + h(8)*0 + h(9)*0 + ...

y(k+ 2) = h(0)*0 + h(1)*0 + **h(2)*x(n+0)** + h(3)*0 + h(4)*0 + **h(5)*x(n - 1)** + h(6)*0 + h(7)*0 + **h(8)*x(n - 2)** + h(9)*0 + ...

y(k+ 3) = **h(0)*x(n+1)** + h(1)*0 + h(2)*0 + **h(3)*x(n+0)** + h(4)*0 + h(5)*0 + **h(6)*x(n - 1)** + h(7)*0 + h(8)*0 + **h(9)*x(n - 2)** + ...

y(k+ 4) = h(0)*0 + **h(1)*x(n+1)** + h(2)*0 + h(3)*0 + **h(4)*x(n+0)** + h(5)*0 + h(6)*0 + **h(7)*x(n - 1)** + h(8)*0 + h(9)*0 + ...

y(k+ 5) = h(0)*0 + h(1)*0 + **h(2)*x(n+1)** + h(3)*0 + h(4)*0 + **h(5)*x(n+0)** + h(6)*0 + h(7)*0 + **h(8)*x(n - 1)** + h(9)*0 + ...

y(k+ 6) = **h(0)*x(n+2)** + h(1)*0 + h(2)*0 + **h(3)*x(n+1)** + h(4)*0 + h(5)*0 + **h(6)*x(n+0)** + h(7)*0 + h(8)*0 + **h(9)*x(n - 1)** + ...

y(k+ 7) = h(0)*0 + **h(1)*x(n+2)** + h(2)*0 + h(3)*0 + **h(4)*x(n+1)** + h(5)*0 + h(6)*0 + **h(7)*x(n + 0)** + h(8)*0 + h(9)*0 + ...

y(k+ 8) = h(0)*0 + h(1)*0 + **h(2)*x(n+2)** + h(3)*0 + h(4)*0 + **h(5)*x(n+1)** + h(6)*0 + h(7)*0 + **h(8)*x(n+0)** + h(9)*0 + ...

y(k+ 0) = h(0)*x(n+0) + h(3)*x(n - 1) + h(6)*x(n - 2) + h(9)*x(n - 3) + h(12)*x(n - 4) + ...

y(k+ 9) = **h(0)*x(n+3)** + h(1)*0 + h(2)*0 + **h(3)*x(n+2)** + h(4)*0 + h(5)*0 + **h(6)*x(n+1)** + h(7)*0 + h(8)*0 + **h(9)*x(n+0)** + ...

y(k+ 1) = h(1)*x(n+0) + h(4)*x(n - 1) + h(7)*x(n - 2) + h(10)*x(n- 3) + h(13)*x(n - 4) + ...

y(k+ 2) = h(2)*x(n+0) + h(5)*x(n - 1) + h(8)*x(n - 2) + h(11)*x(n- 3) + h(14)*x(n - 4) + ...

y(k+ 3) = h(0)*x(n+1) + h(3)*x(n+ 0) + h(6)*x(n - 1) + h(9)*x(n - 2) + h(12)*x(n - 3) + ...

y(k+ 4) = h(1)*x(n+1) + h(4)*x(n+ 0) + h(7)*x(n - 1) + h(10)*x(n- 2) + h(13)*x(n - 3) + ...

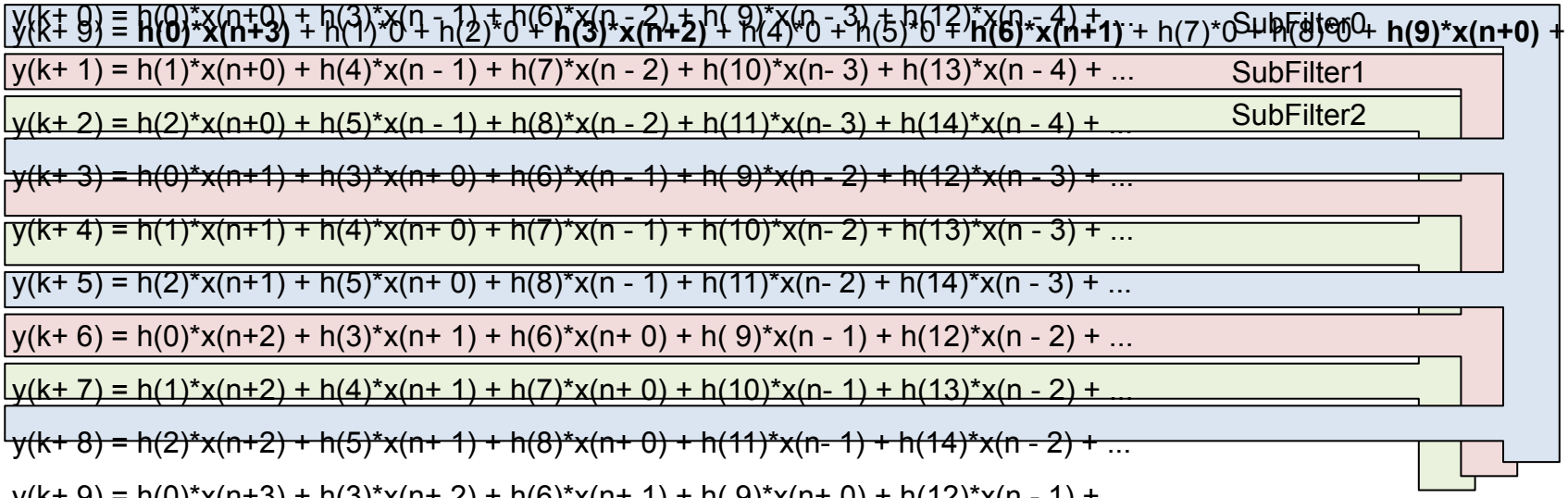
y(k+ 5) = h(2)*x(n+1) + h(5)*x(n+ 0) + h(8)*x(n - 1) + h(11)*x(n- 2) + h(14)*x(n - 3) + ...

y(k+ 6) = h(0)*x(n+2) + h(3)*x(n+ 1) + h(6)*x(n+ 0) + h(9)*x(n - 1) + h(12)*x(n - 2) + ...

y(k+ 7) = h(1)*x(n+2) + h(4)*x(n+ 1) + h(7)*x(n+ 0) + h(10)*x(n- 1) + h(13)*x(n - 2) + ...

y(k+ 8) = h(2)*x(n+2) + h(5)*x(n+ 1) + h(8)*x(n+ 0) + h(11)*x(n- 1) + h(14)*x(n - 2) + ...

y(k+ 9) = h(0)*x(n+3) + h(3)*x(n+ 2) + h(6)*x(n+ 1) + h(9)*x(n+ 0) + h(12)*x(n - 1) + ...



Полифазный интерполирующий фильтр (уравнения)

Коэффициент интерполяции **M = 4**

y(k+ 0) = **h(0)*x(n+0)** + h(1)*0 + h(2)*0 + h(3)*0 + **h(4)*x(n - 1)** + h(5)*0 + h(6)*0 + h(7)*0 + **h(8)*x(n - 2)** + h(9)*0 + ...

y(k+ 1) = h(0)*0 + **h(1)*x(n+0)** + h(2)*0 + h(3)*0 + h(4)*0 + **h(5)*x(n - 1)** + h(6)*0 + h(7)*0 + h(8)*0 + **h(9)*x(n - 2)** + ...

y(k+ 2) = h(0)*0 + h(1)*0 + **h(2)*x(n+0)** + h(3)*0 + h(4)*0 + h(5)*0 + **h(6)*x(n - 1)** + h(7)*0 + h(8)*0 + h(9)*0 + ...

y(k+ 3) = h(0)*0 + h(1)*0 + h(2)*0 + **h(3)*x(n+0)** + h(4)*0 + h(5)*0 + h(6)*0 + **h(7)*x(n - 1)** + h(8)*0 + h(9)*0 + ...

y(k+ 4) = **h(0)*x(n+1)** + h(1)*0 + h(2)*0 + h(3)*0 + **h(4)*x(n+0)** + h(5)*0 + h(6)*0 + h(7)*0 + **h(8)*x(n-1)** + h(9)*0 + ...

y(k+ 5) = h(0)*0 + **h(1)*x(n+1)** + h(2)*0 + h(3)*0 + h(4)*0 + **h(5)*x(n+0)** + h(6)*0 + h(7)*0 + h(8)*0 + **h(9)*x(n - 1)** + ...

y(k+ 6) = h(0)*0 + h(1)*0 + **h(2)*x(n+1)** + h(3)*0 + h(4)*0 + h(5)*0 + **h(6)*x(n+0)** + h(7)*0 + h(8)*0 + h(9)*0 + ...

y(k+ 7) = h(0)*0 + h(1)*0 + h(2)*0 + **h(3)*x(n+1)** + h(4)*0 + h(5)*0 + h(6)*0 + **h(7)*x(n+0)** + h(8)*0 + h(9)*0 + ...

y(k+ 8) = **h(0)*x(n+2)** + h(1)*0 + h(2)*0 + h(3)*0 + **h(4)*x(n+1)** + h(5)*0 + h(6)*0 + h(7)*0 + **h(8)*x(n+0)** + h(9)*0 + ...

y(k+ 9) = h(0)*0 + **h(1)*x(n+2)** + h(2)*0 + h(3)*0 + h(4)*0 + **h(5)*x(n+1)** + h(6)*0 + h(7)*0 + h(8)*0 + **h(9)*x(n+0)** + ...

y(k+ 0) = h(0)*x(n+0) + h(4)*x(n - 1) + h(8)*x(n - 2) + h(12)*x(n - 3) + h(16)*x(n - 4) + ...

y(k+ 1) = h(1)*x(n+0) + h(5)*x(n - 1) + h(9)*x(n - 2) + h(13)*x(n - 3) + h(17)*x(n - 4) + ...

y(k+ 2) = h(2)*x(n+0) + h(6)*x(n - 1) + h(10)*x(n- 2) + h(14)*x(n - 3) + h(18)*x(n - 4) + ...

y(k+ 3) = h(3)*x(n+0) + h(7)*x(n - 1) + h(11)*x(n- 2) + h(15)*x(n - 3) + h(19)*x(n - 4) + ...

y(k+ 4) = h(0)*x(n+1) + h(4)*x(n+ 0) + h(8)*x(n - 1) + h(12)*x(n - 2) + h(16)*x(n - 3) + ...

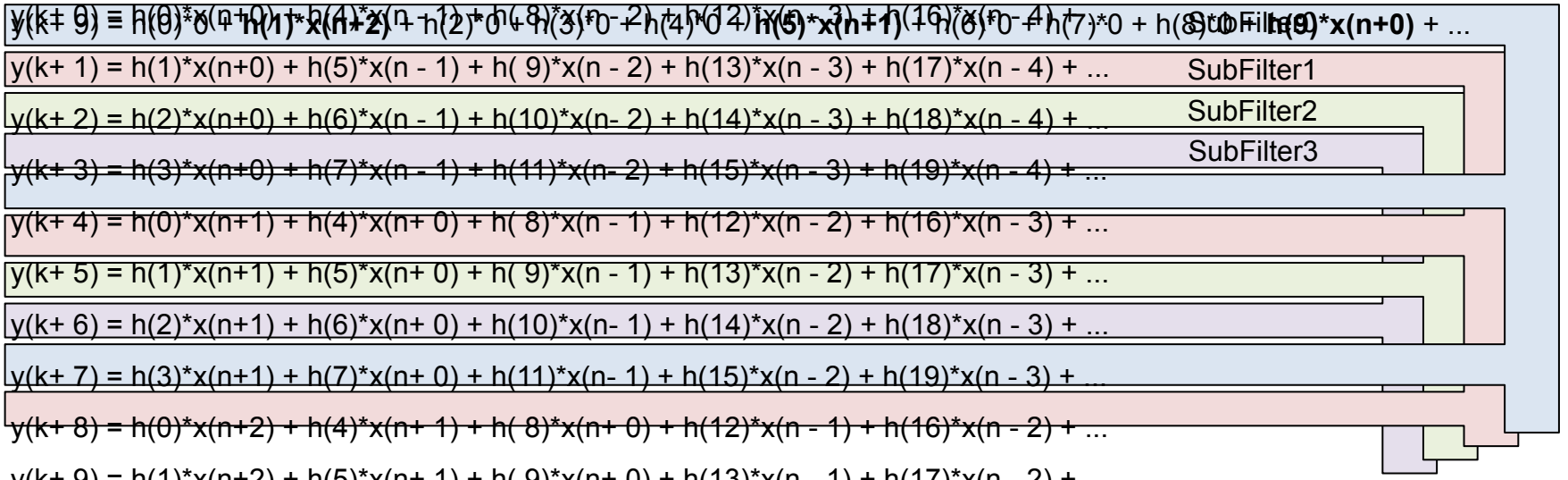
y(k+ 5) = h(1)*x(n+1) + h(5)*x(n+ 0) + h(9)*x(n - 1) + h(13)*x(n - 2) + h(17)*x(n - 3) + ...

y(k+ 6) = h(2)*x(n+1) + h(6)*x(n+ 0) + h(10)*x(n- 1) + h(14)*x(n - 2) + h(18)*x(n - 3) + ...

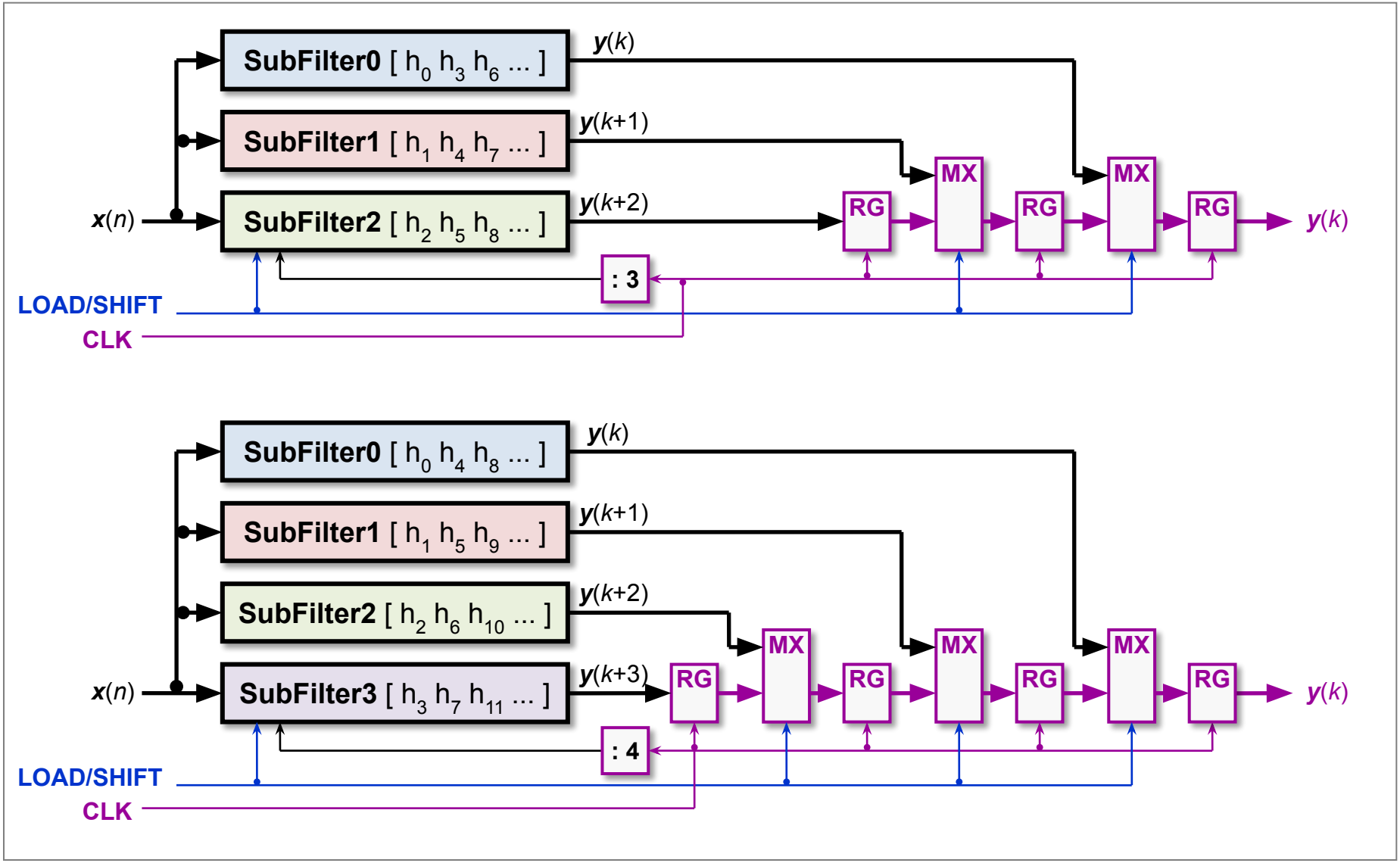
y(k+ 7) = h(3)*x(n+1) + h(7)*x(n+ 0) + h(11)*x(n- 1) + h(15)*x(n - 2) + h(19)*x(n - 3) + ...

y(k+ 8) = h(0)*x(n+2) + h(4)*x(n+ 1) + h(8)*x(n+ 0) + h(12)*x(n - 1) + h(16)*x(n - 2) + ...

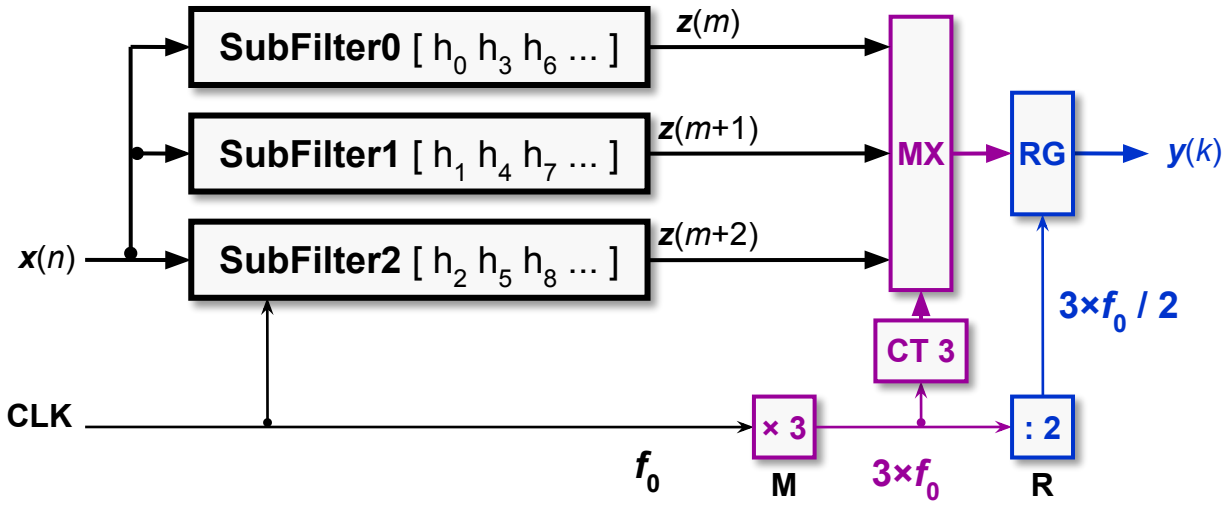
y(k+ 9) = h(1)*x(n+2) + h(5)*x(n+ 1) + h(9)*x(n+ 0) + h(13)*x(n - 1) + h(17)*x(n - 2) + ...

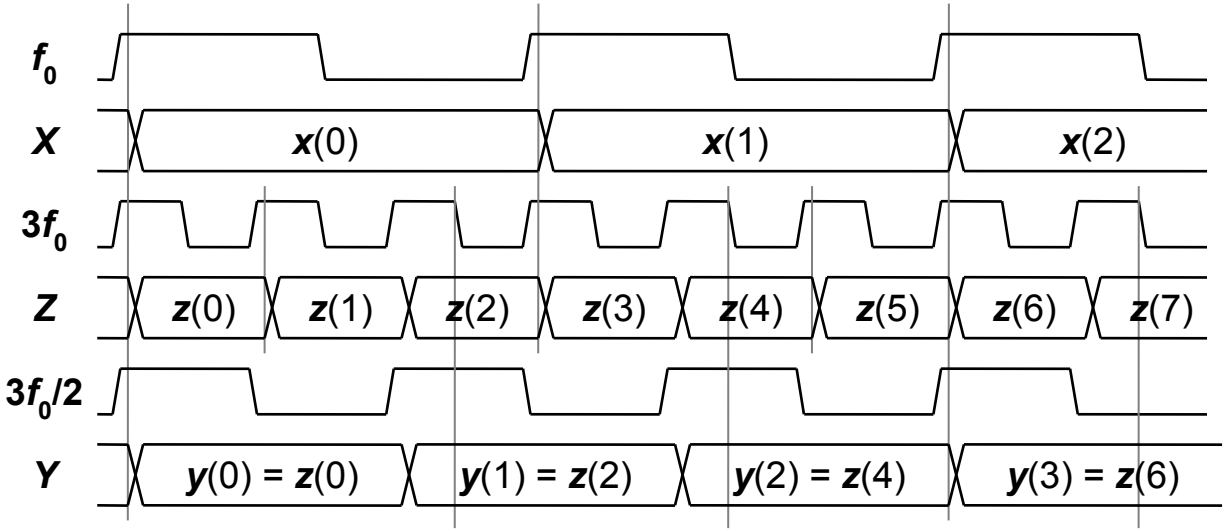


Полифазный интерполирующий фильтр (структуры)

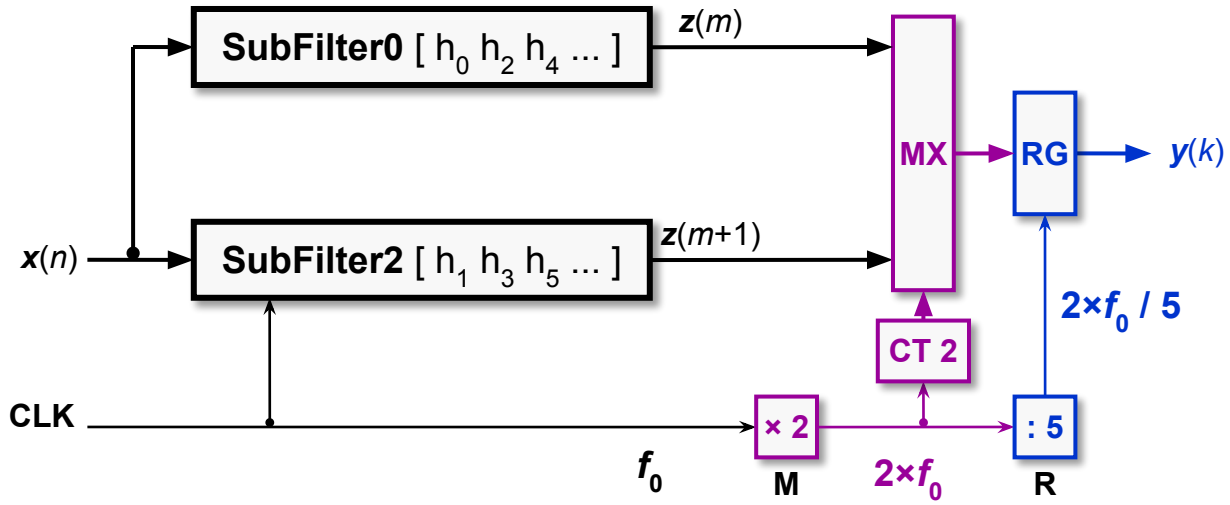


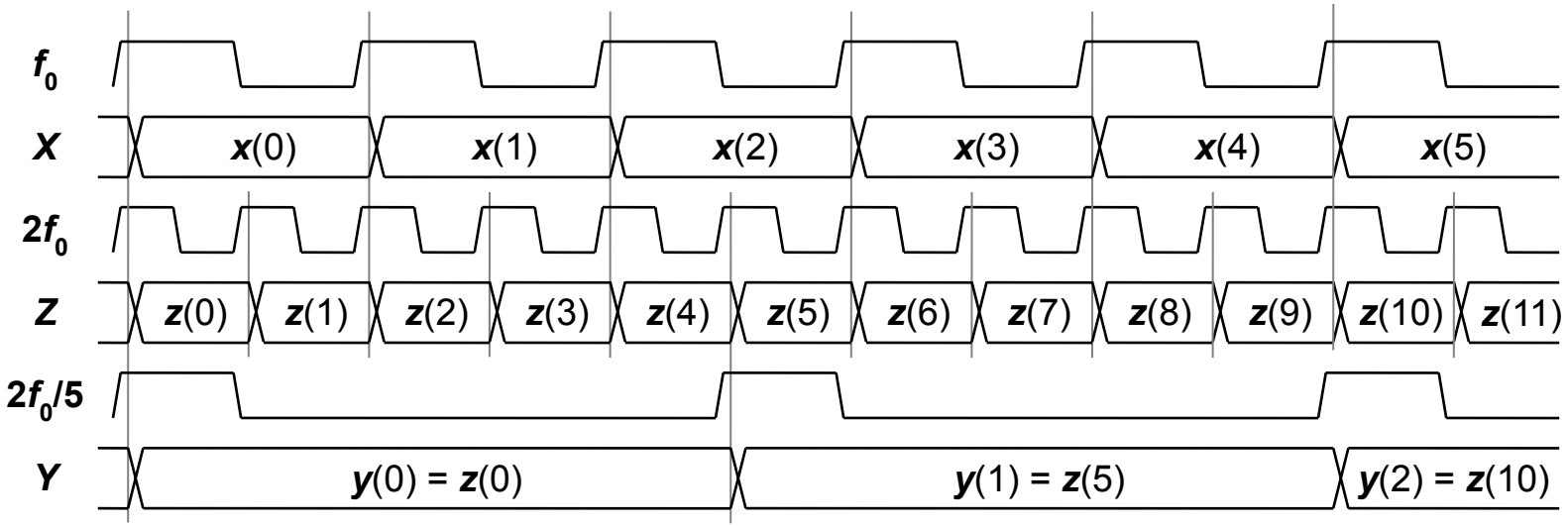
Полифазный фильтр передискретизации M=3, R=2, M/R = 1.5



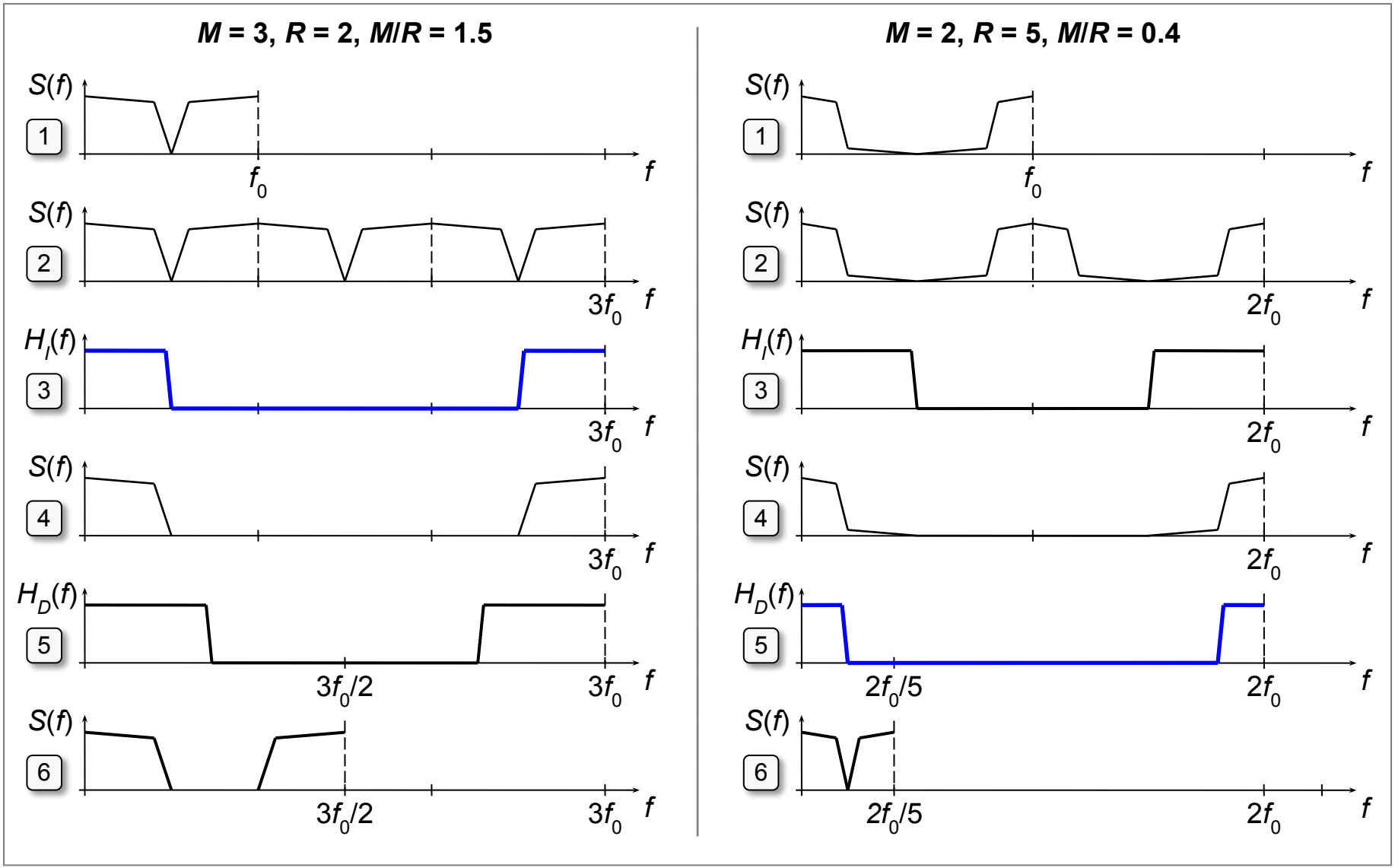


Полифазный фильтр передискретизации M=2, R=5, M/R = 0.4





Полифазные фильтры передискретизации (преобразования спектров)



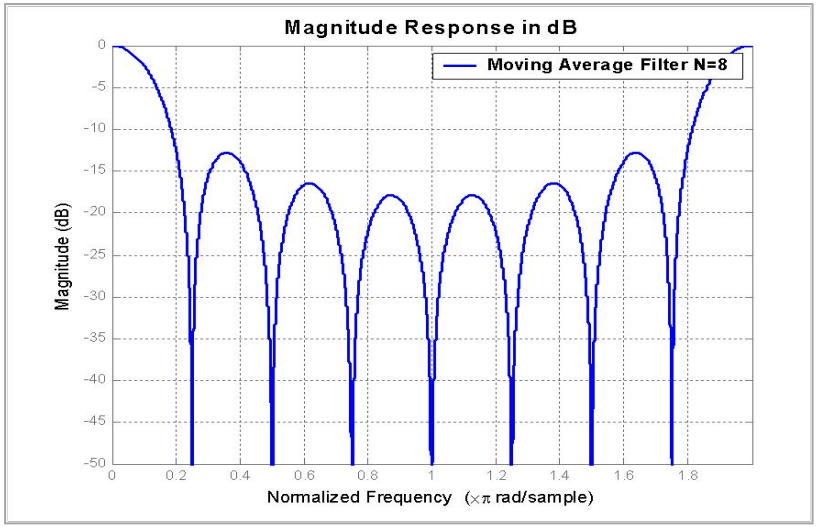
Фильтр с прямоугольным окном (уравнения)

Уравнение фильтра

$$y(n) = \sum_{k=0}^{N-1} x(n-k)$$

соответствует передаточной функции

$$H(z) = \sum_{k=0}^{N-1} z^{-k}$$

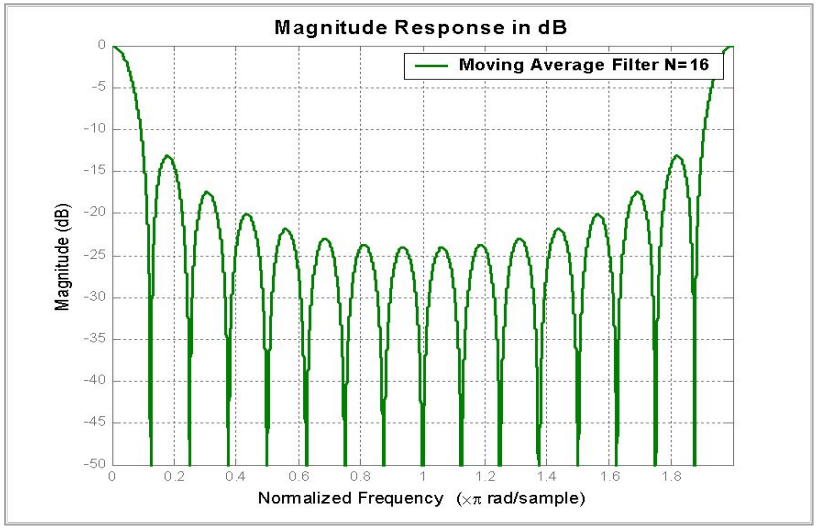


Частотная характеристика

$$H(f) = \frac{\sin(N\pi f)}{\sin(\pi f)}$$

содержит N–1 нулей на частотах $i\frac{F_S}{N}$,

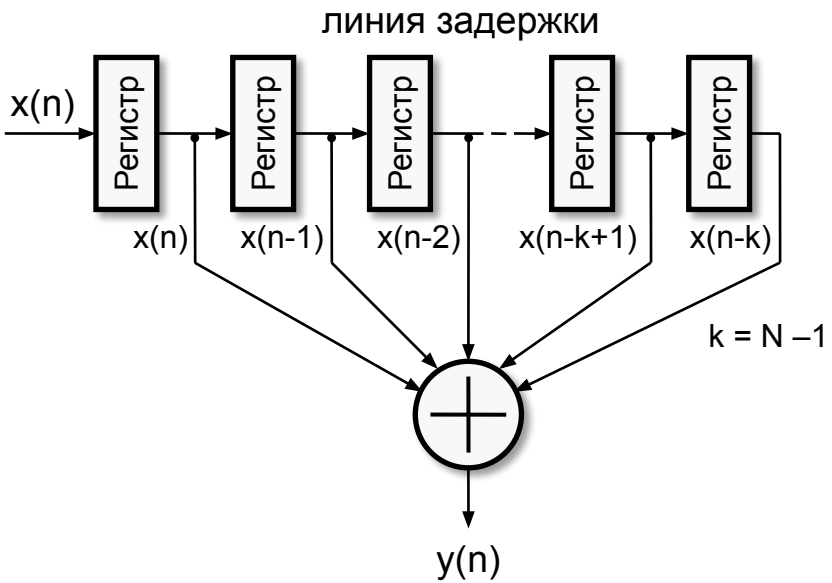
где F_S – частота дискретизации,
$i = 1 \ldots N - 1$



Фильтр с прямоугольным окном (структуры)

НЕРЕКУРСИВНАЯ РЕАЛИЗАЦИЯ

$$y(n) = x(n) + x(n-1) + \ldots + x(n-(N\text{-}1))$$



РЕКУРСИВНАЯ РЕАЛИЗАЦИЯ

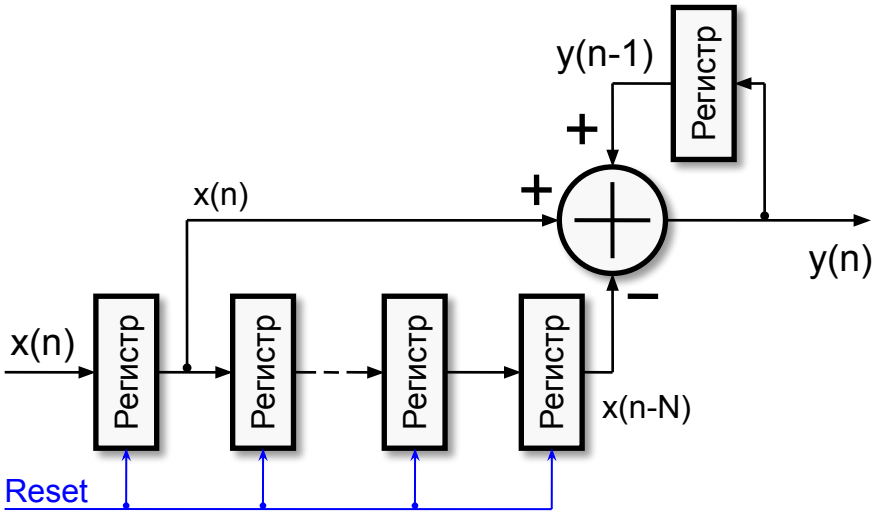
$$H(z) = \sum_{k=0}^{N-1} z^{-k} = \frac{1 - z^{-N}}{1 - z^{-1}}$$

$y(N\text{-}1) = x(N\text{-}1)+x(N\text{-}2)+ \ldots +x(1)+x(0)$

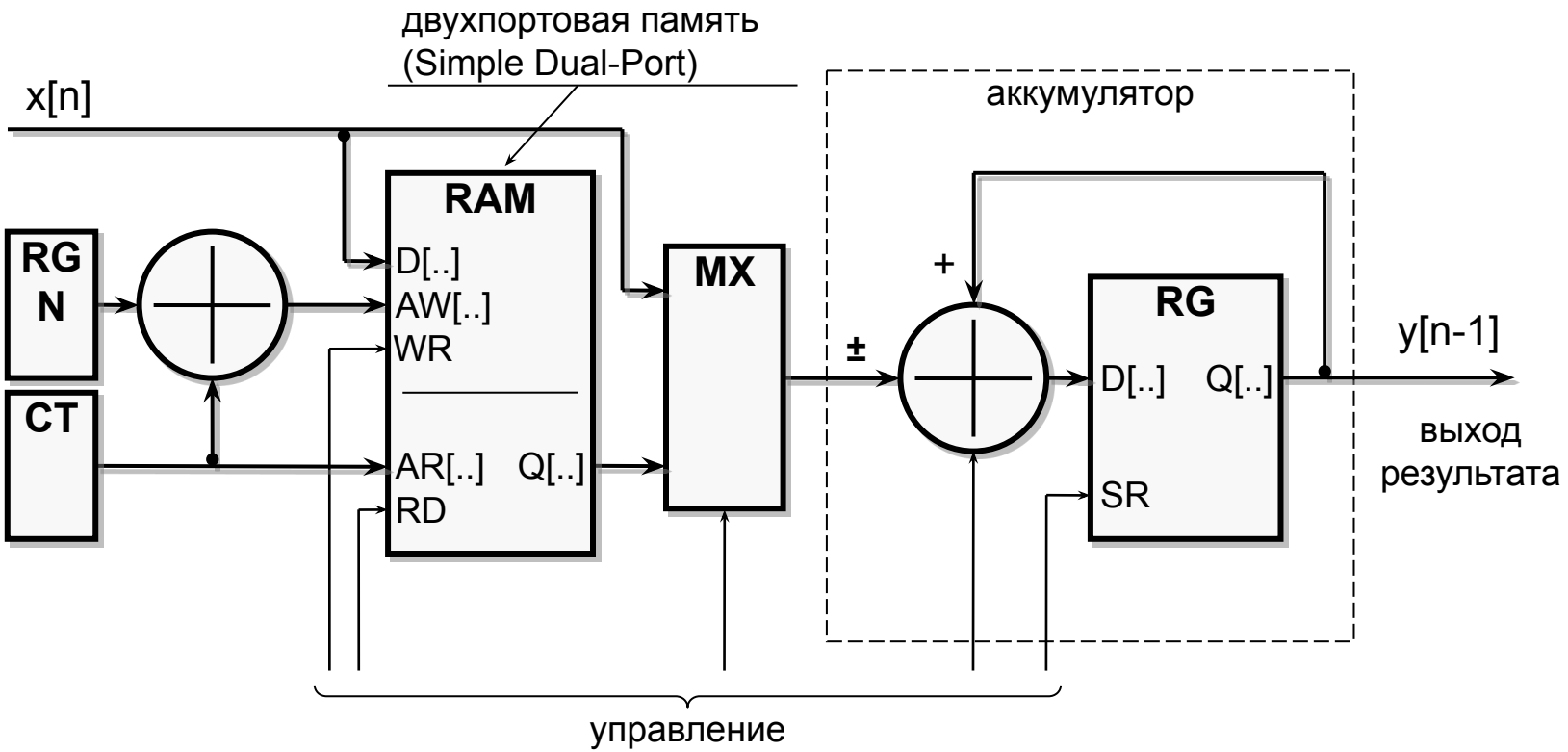
$y(N) = x(N)+x(N\text{-}1)+ \ldots +x(2)+x(1) = x(N) + y(N\text{-}1) – x(0)$

$y(N+1) = x(N+1)+x(N)+ \ldots +x(3)+x(2) = x(N+1)+y(N) – x(1)$

$$y(n) = x(n) + y(n\text{-}1) – x(n\text{-}N)$$



Программируемый фильтр с прямоугольным окном



Структурные элементы CIC-фильтров

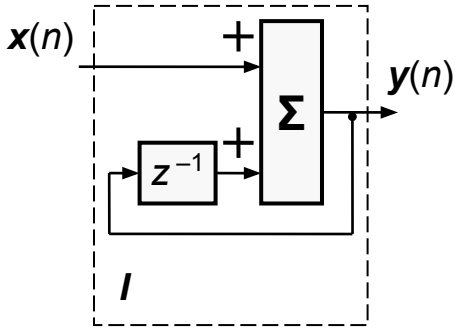
а) Идеальный интегратор

Разностное уравнение

$$y[n] = y[n-1] + x[n]$$

Функция передачи

$$H_I(z) = \frac{1}{1 - z^{-1}}$$



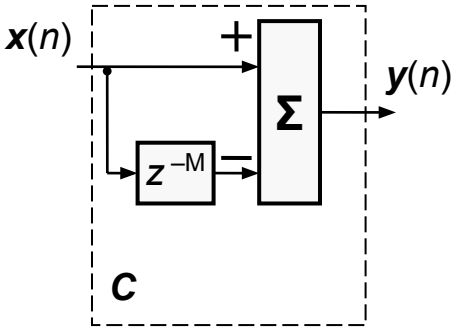
б) Идеальный дифференциатор

Разностное уравнение

$$y[n] = x[n] - x[n-M]$$

Функция передачи

$$H_C(z) = 1 - z^{-M}$$



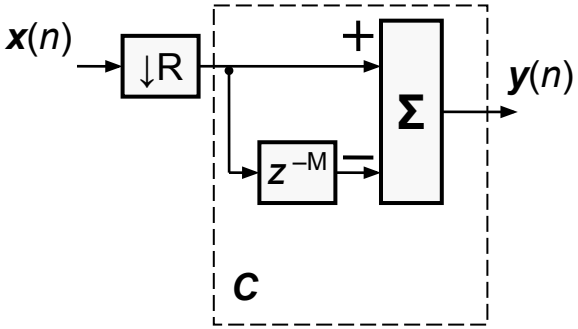
в) Дифференциатор с прореживанием

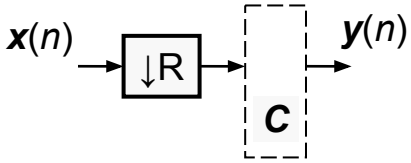
Разностное уравнение

$$y[n] = x[n] - x[n-RM]$$

Функция передачи

$$H_{CR}(z) = 1 - z^{-RM}$$





Структура децимирующего CIC-фильтра

Функция передачи *N*-каскадного децимирующего фильтра с прореживанием *R* и дифференциальной задержкой *M*

$$H(z) = \left[\frac{1 - z^{-RM}}{1 - z^{-1}}\right]^N = \left[\frac{1}{1 - z^{-1}}\right]^N \cdot [1 - z^{-RM}]^N = [H_I(z)]^N \cdot [H_{CR}(z)]^N$$

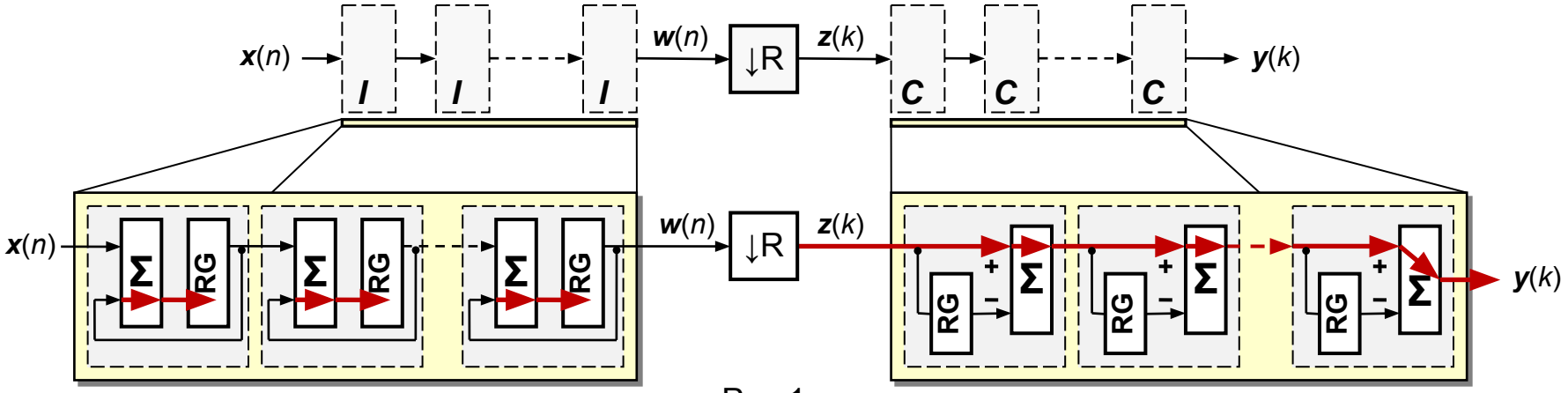


Рис.1

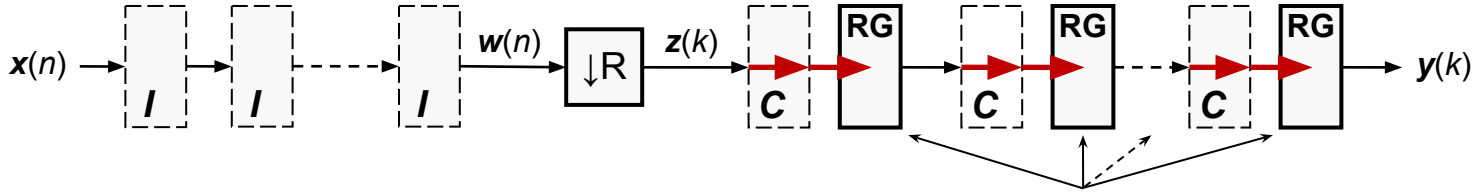
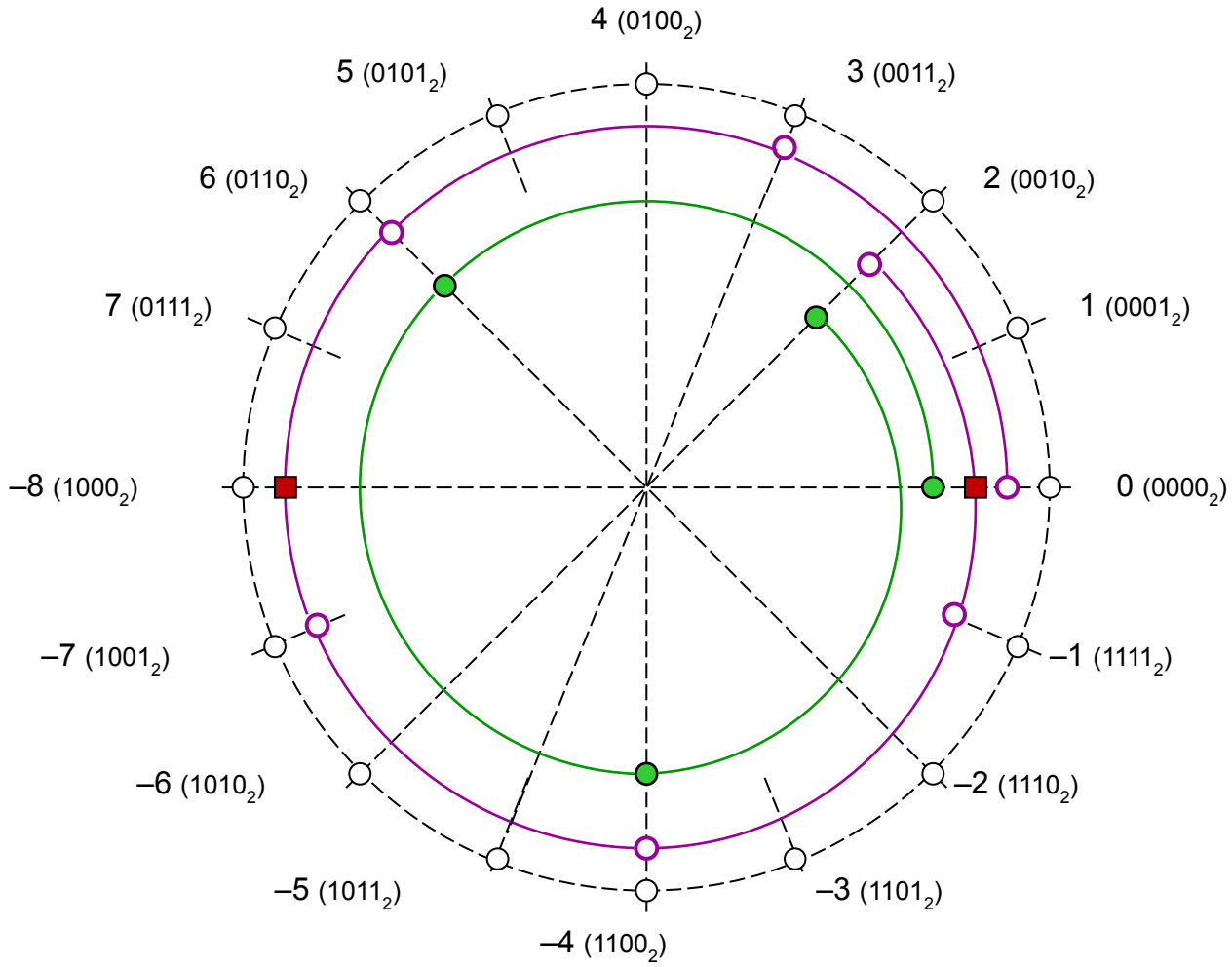


Рис.2

Компенсация переполнения в CIC-дециматоре



Разрядность и усиление CIC-фильтра

Разрядность операций CIC-фильтра и его коэффициент усиления определяются формулами

$$B_{max} = [N \log_2 RM + B_0]$$

$$G = \frac{(RM)^N}{2^{[N \log_2 RM]}}$$

где B_{max} – максимальная разрядность операций;

B_0 – разрядность входных отсчетов;

N – число каскадов;

R – коэффициент децимации;

M – параметр дифференциальной задержки;

G – коэффициент усиления;

[.] – операция округления до большего целого.

Пример 1

Для

$B_0 = 12$; $N = 5$; $R = 24$; $M = 2$

получаем

$\boldsymbol{B_{max}} = 40$; $\boldsymbol{G} = 0.94922$

Пример 2

Для

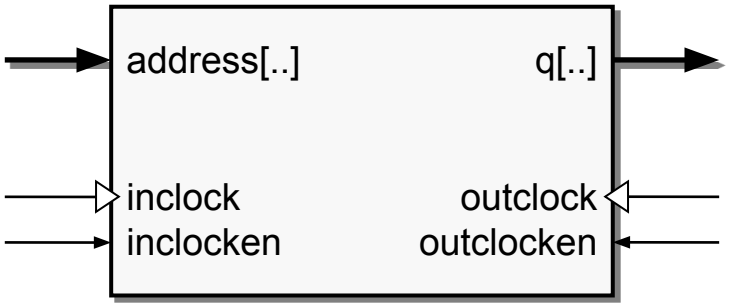
$B_0 = 12$; $N = 2$; $R = 24$; $M = 2$

получаем

$\boldsymbol{B_{max}} = 24$; $\boldsymbol{G} = 0.5625$

Структурные неоднородности ПЛИС – память и DSP блоки

Память ПЛИС (ROM Memory Mode)



address[..] – шина адреса

inclock – сигнал тактовой частоты входа

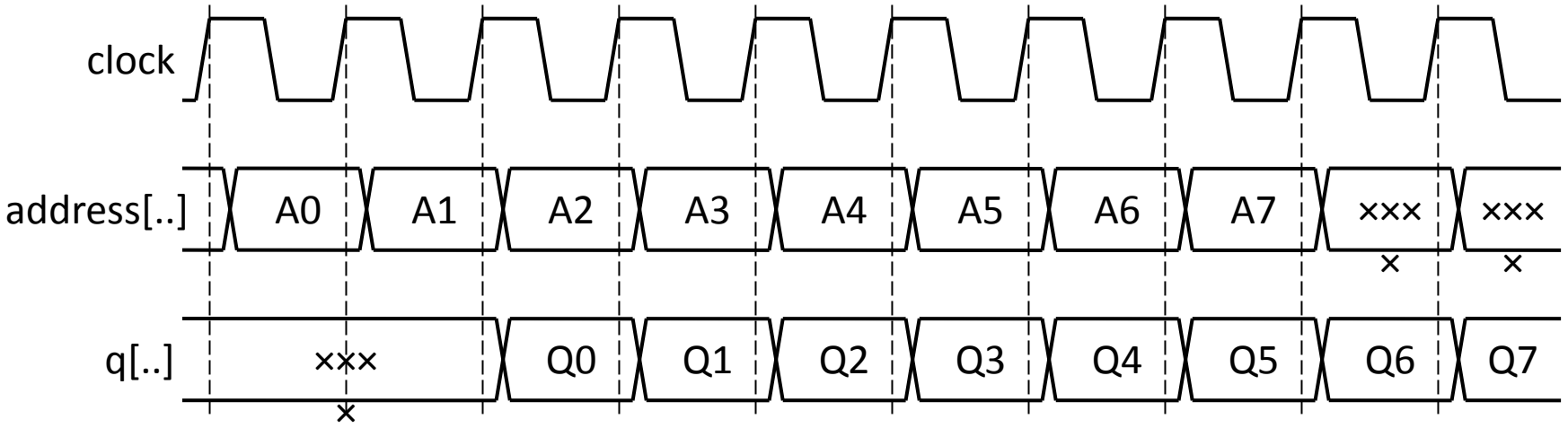
inclocken – разрешение входного тактирования

q[..] – шина выходных данных

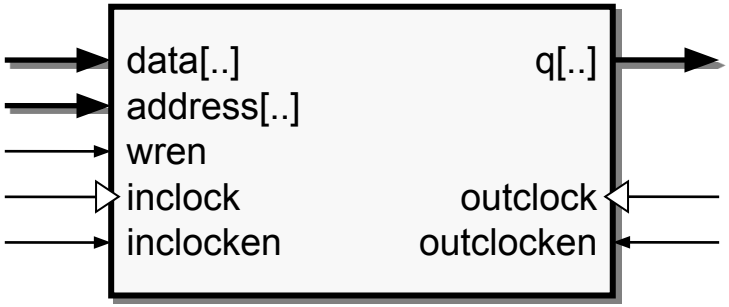
outclock – сигнал тактовой частоты выхода

outclocken – разрешение выходного тактирования

inclock = outclock = clock, inclocken = outclocken = '1'



Память ПЛИС (Single-Port Memory Mode)



data[..] – шина входных данных

address[..] – шина адреса

wren – сигнал разрешения записи

inclock – сигнал тактовой частоты входа

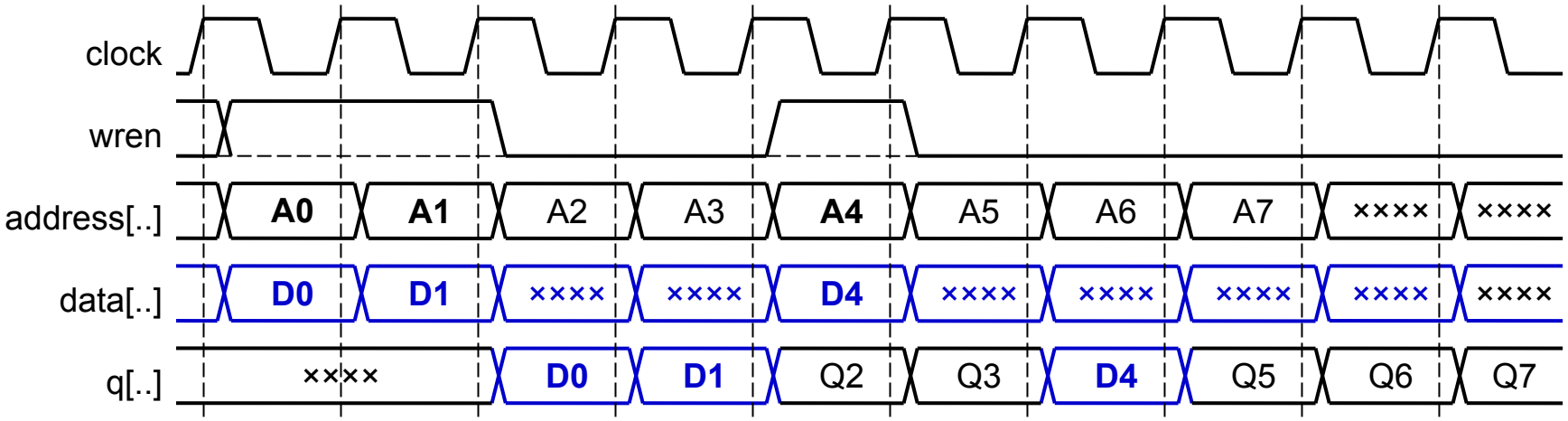
inclocken – разрешение входного тактирования

q[..] – шина выходных данных

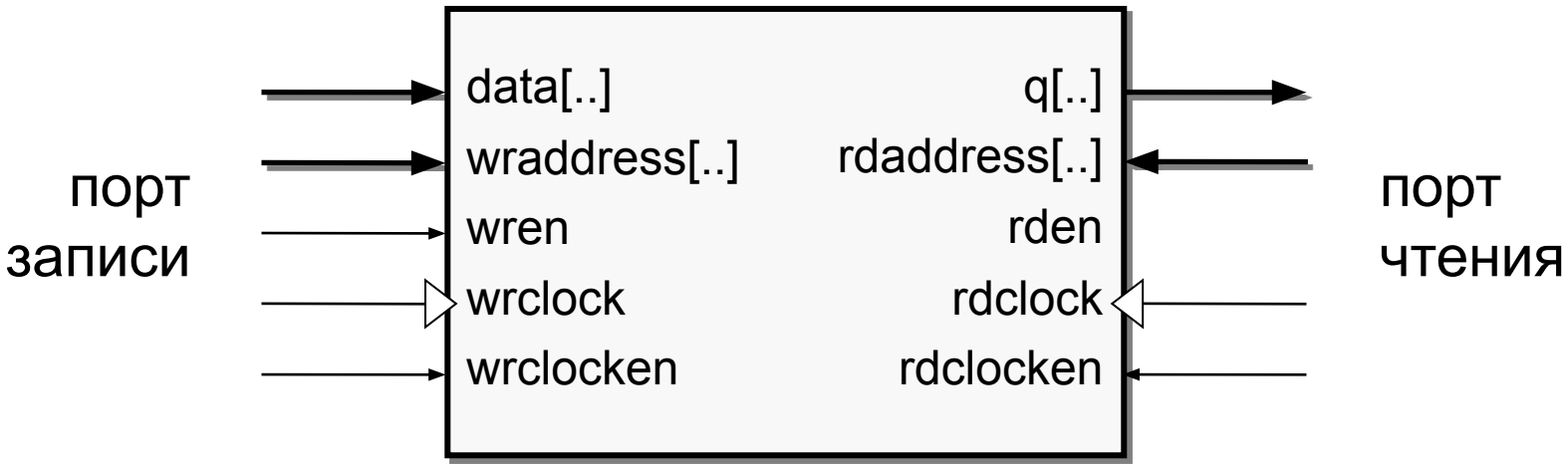
outclock – сигнал тактовой частоты выхода

outclocken – разрешение выходного тактирования

inclock = outclock = clock, inclocken = outclocken = '1'



Память ПЛИС (Simple Dual-Port Memory Mode)



data[..] – шина входных данных

wraddress[..] – шина адреса записи

wren – сигнал разрешения записи

wrclock – сигнал тактовой частоты записи

wrclocken – сигнал разрешения тактовой частоты записи

q[..] – шина выходных данных

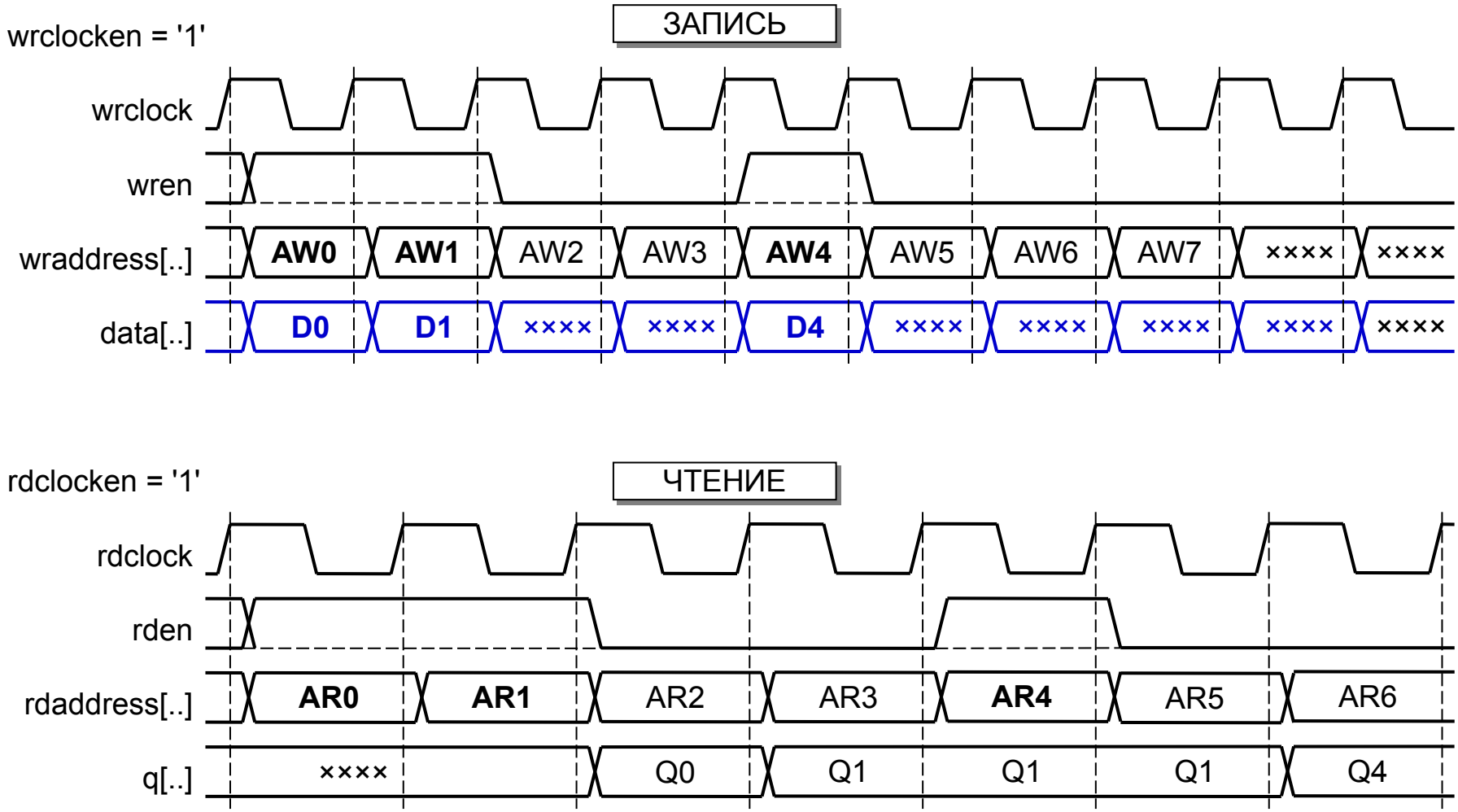
rdaddress[..] – шина адреса чтения

rden – сигнал разрешения записи

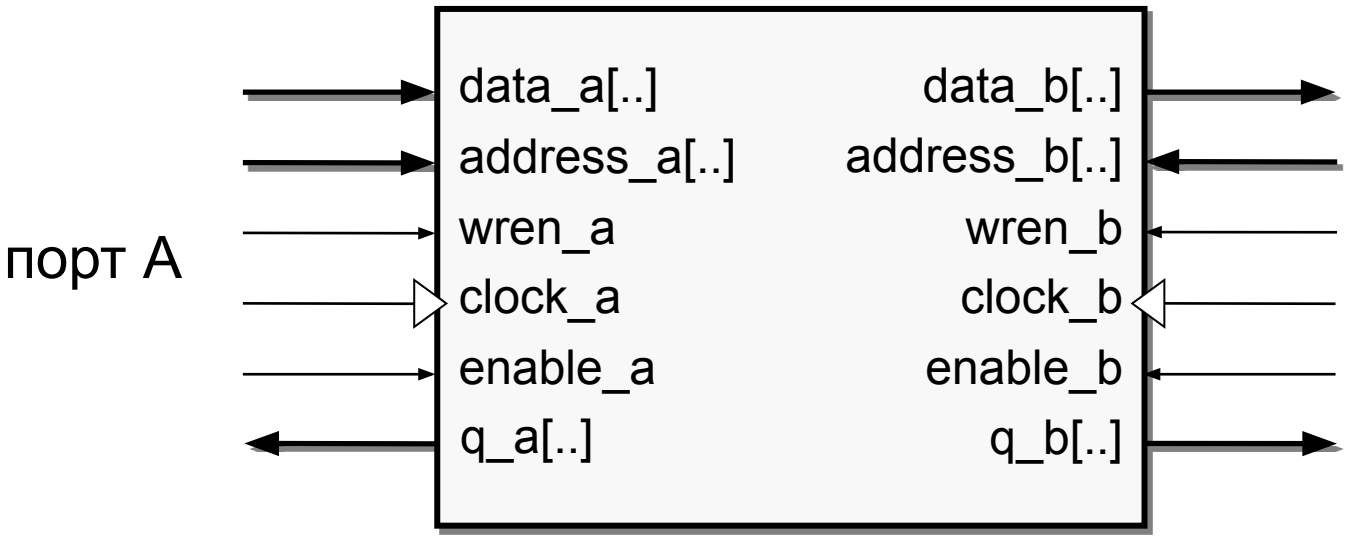
rdclock – сигнал тактовой частоты записи

rdclocken – сигнал разрешения тактовой частоты чтения

Память ПЛИС (Simple Dual-Port Memory Mode) – диаграммы



Память ПЛИС (True Dual-Port Memory Mode)



data_a[..] – шина входных данных

address_a[..] – шина адреса

wren_a – сигнал разрешения записи

clock_a – сигнал тактовой частоты

enable_a – сигнал разрешения тактирования

q_a[..] – шина выходных данных

data_b[..] – шина входных данных

address_b[..] – шина адреса

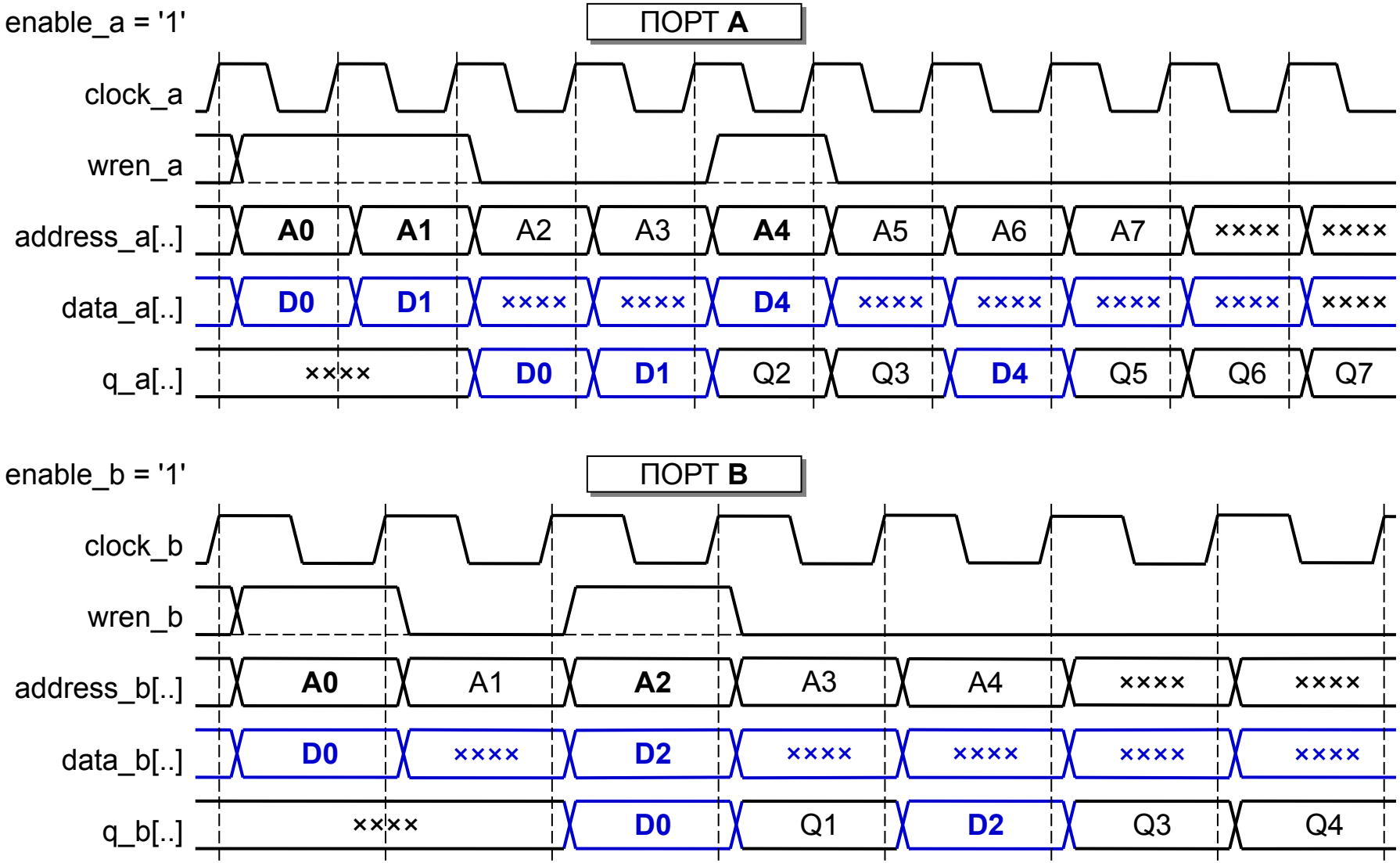
wren_b – сигнал разрешения записи

clock_b – сигнал тактовой частоты

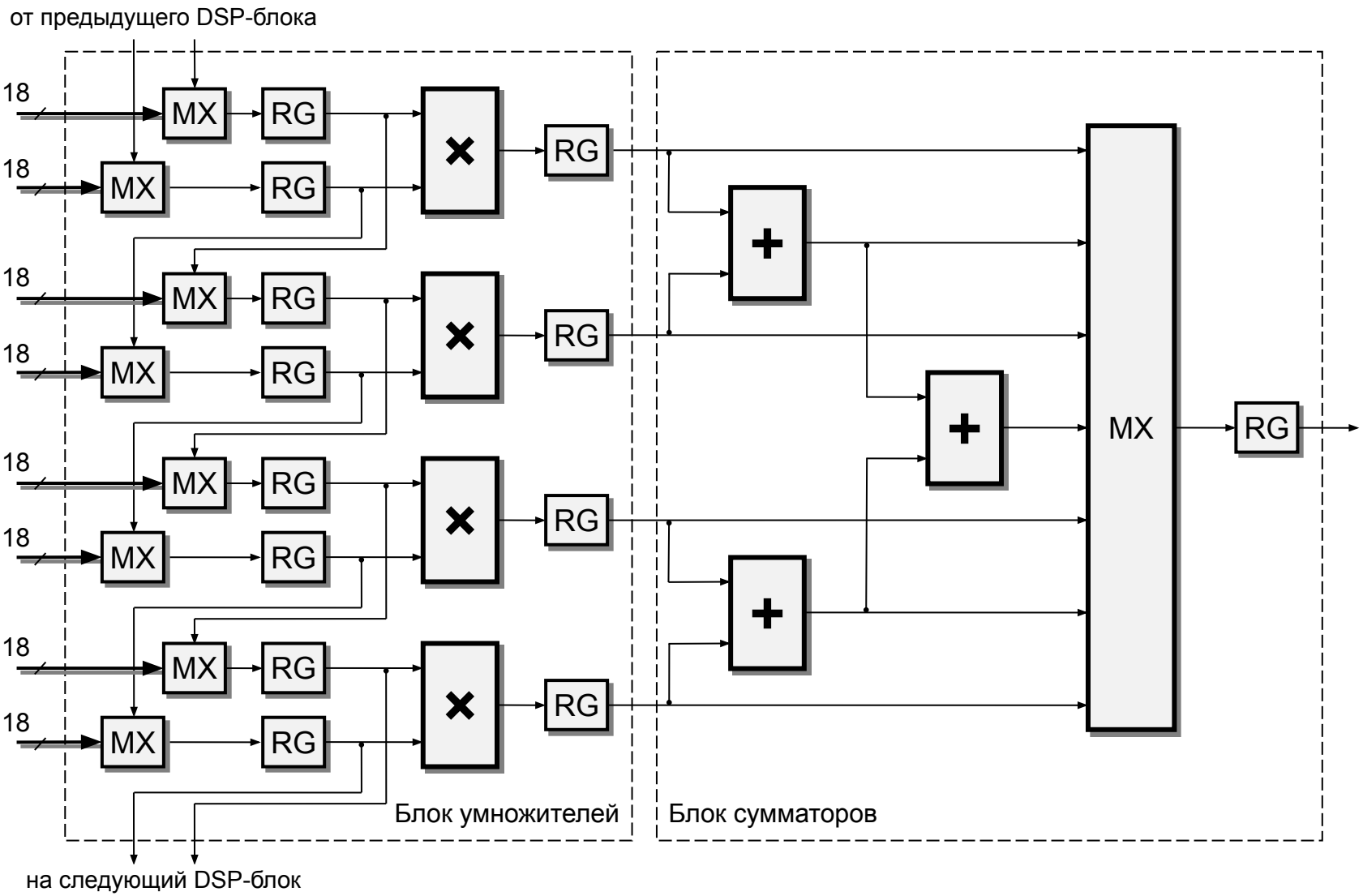
enable_b – сигнал разрешения тактирования

q_b[..] – шина выходных данных

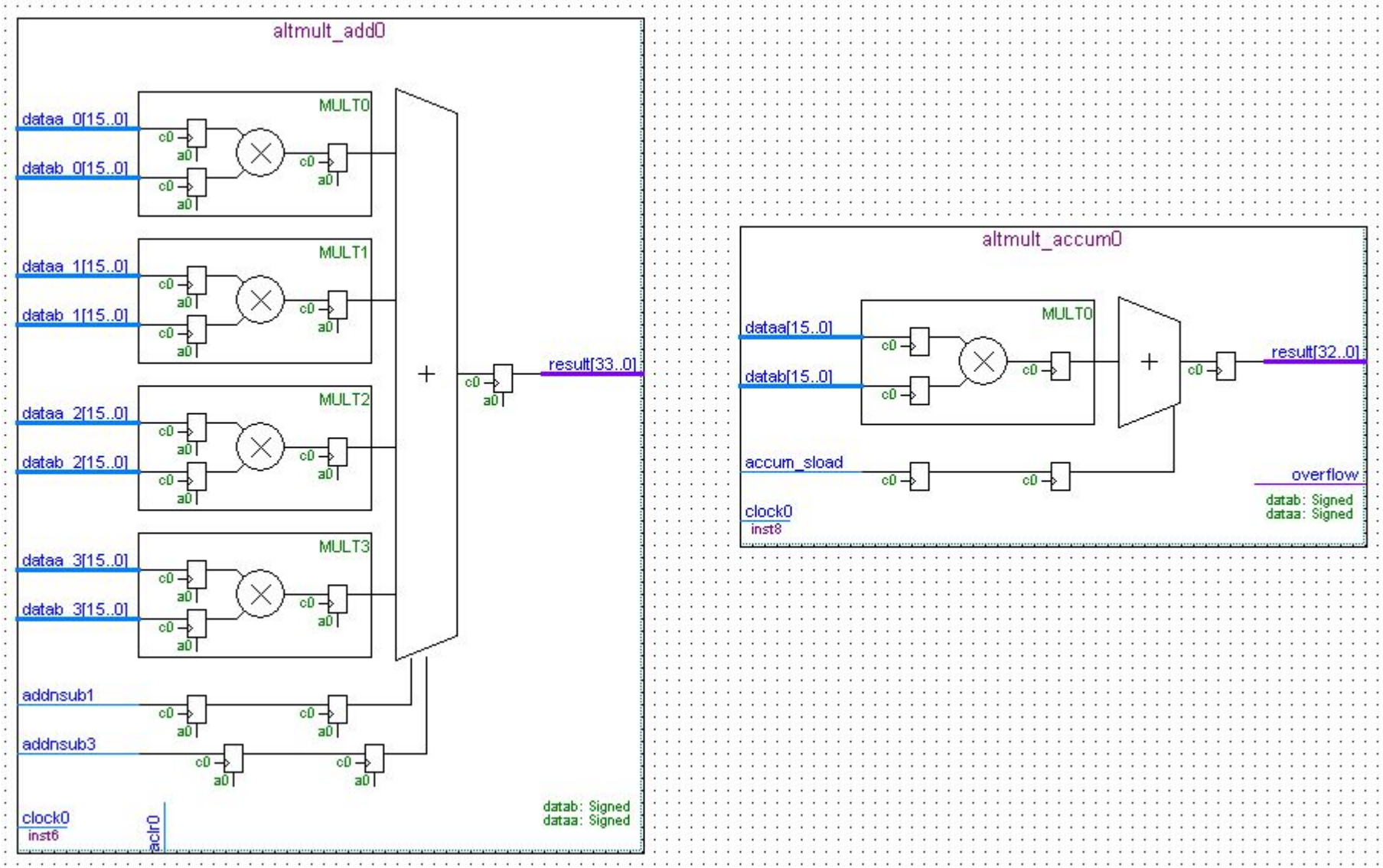
Память ПЛИС (True Dual-Port Memory Mode) – диаграммы



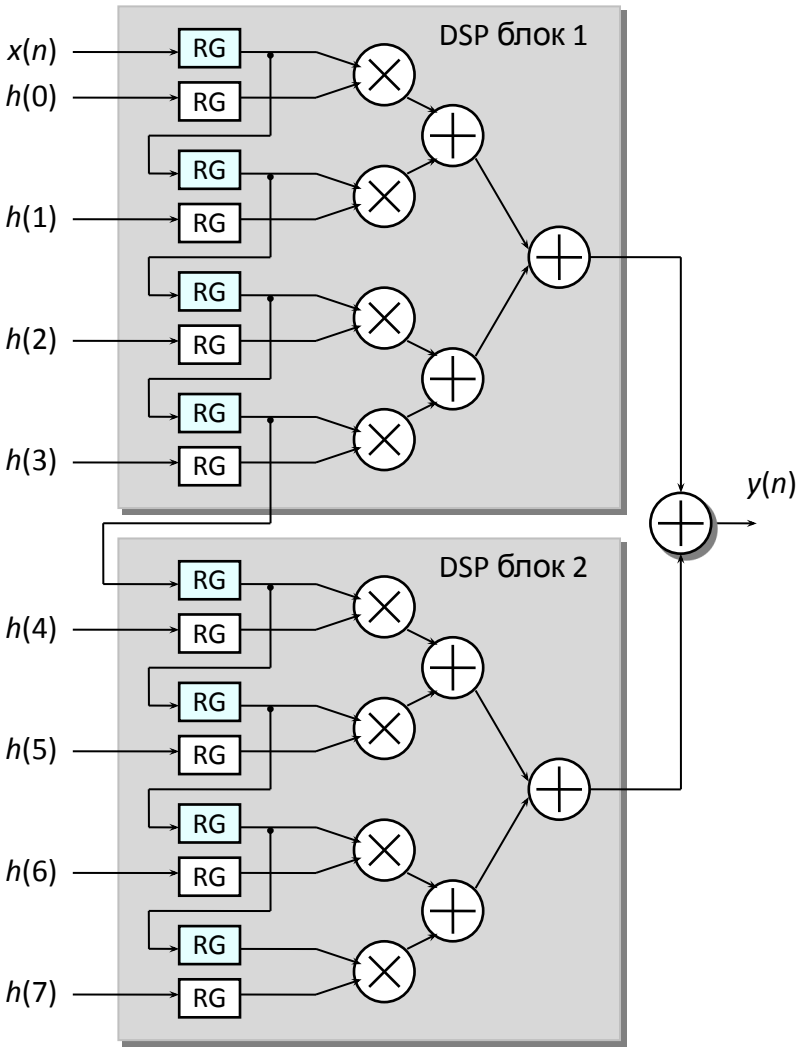
DSP-блоки ПЛИС

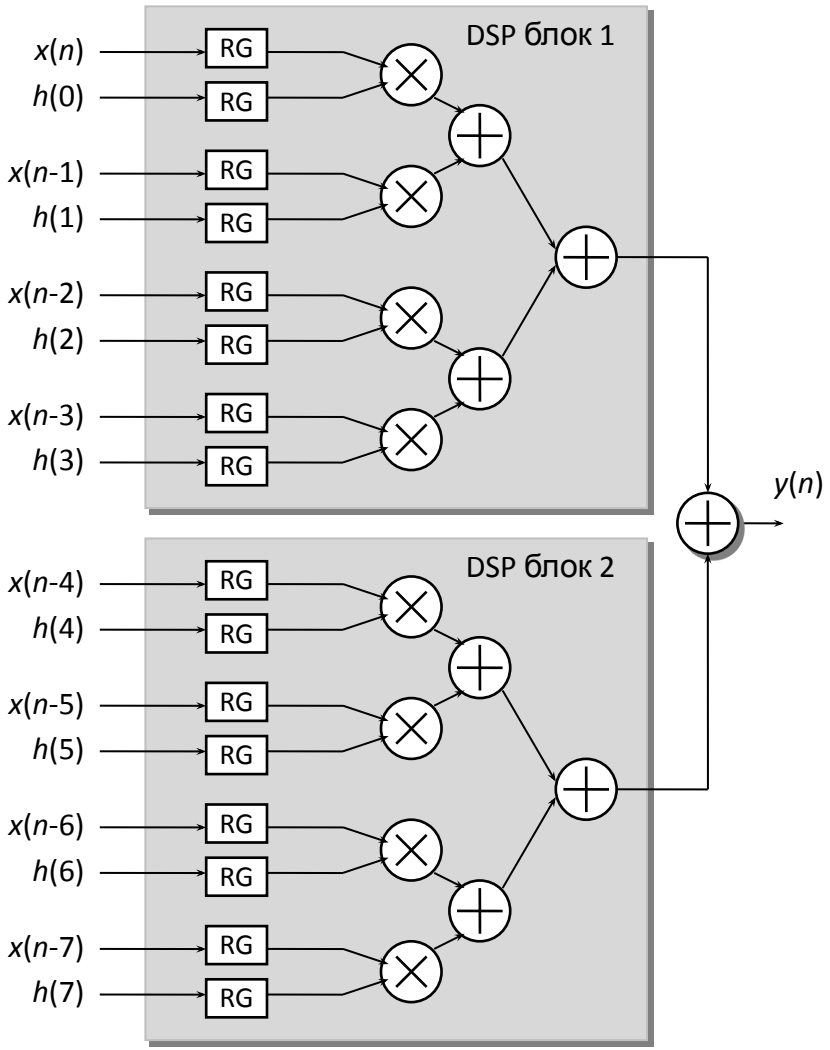


Пример мегафункций DSP-блоков ПЛИС

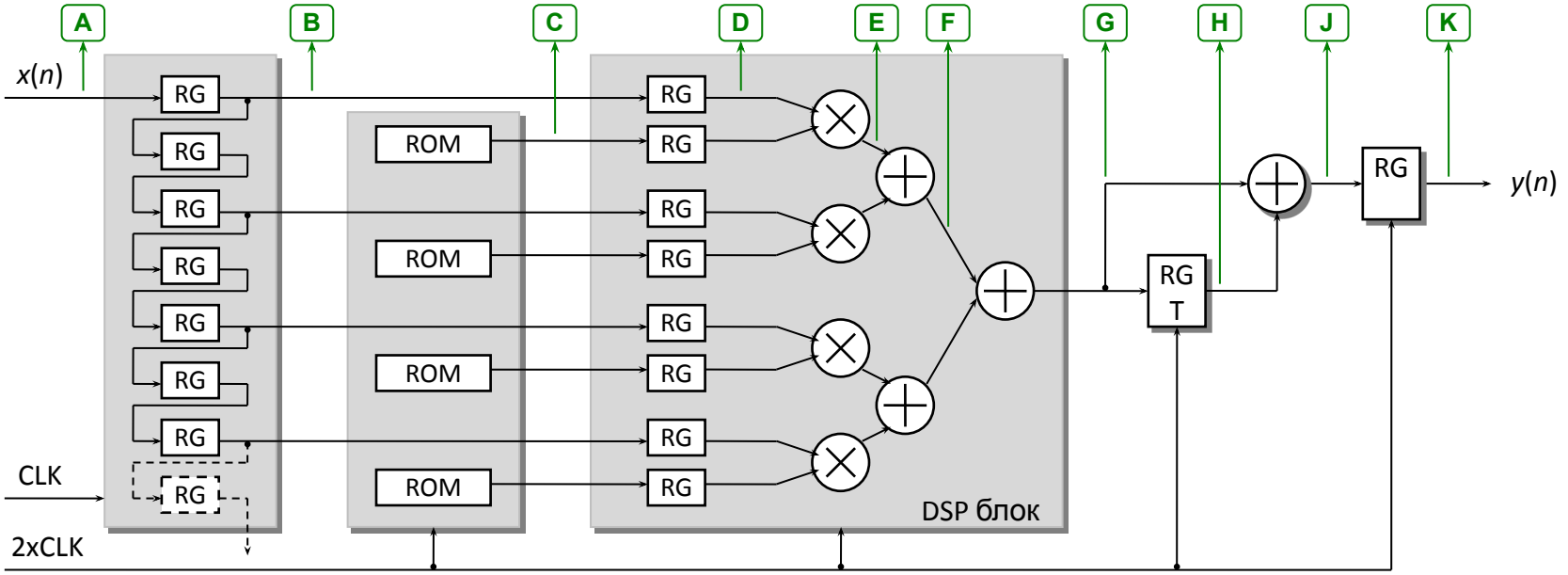


Использование RAM и DSP блоков (FIR)



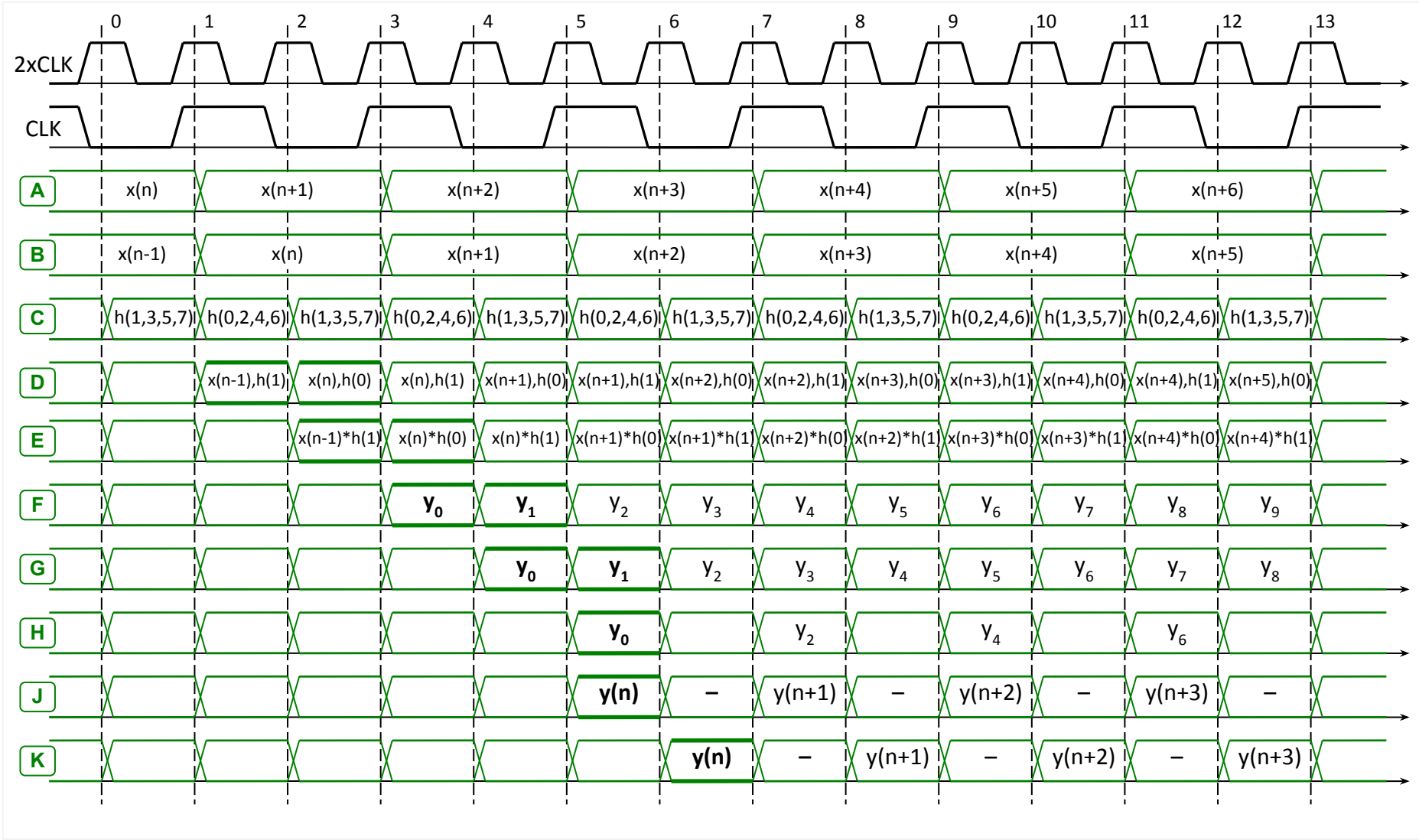


Использование RAM и DSP блоков (TDM FIR)

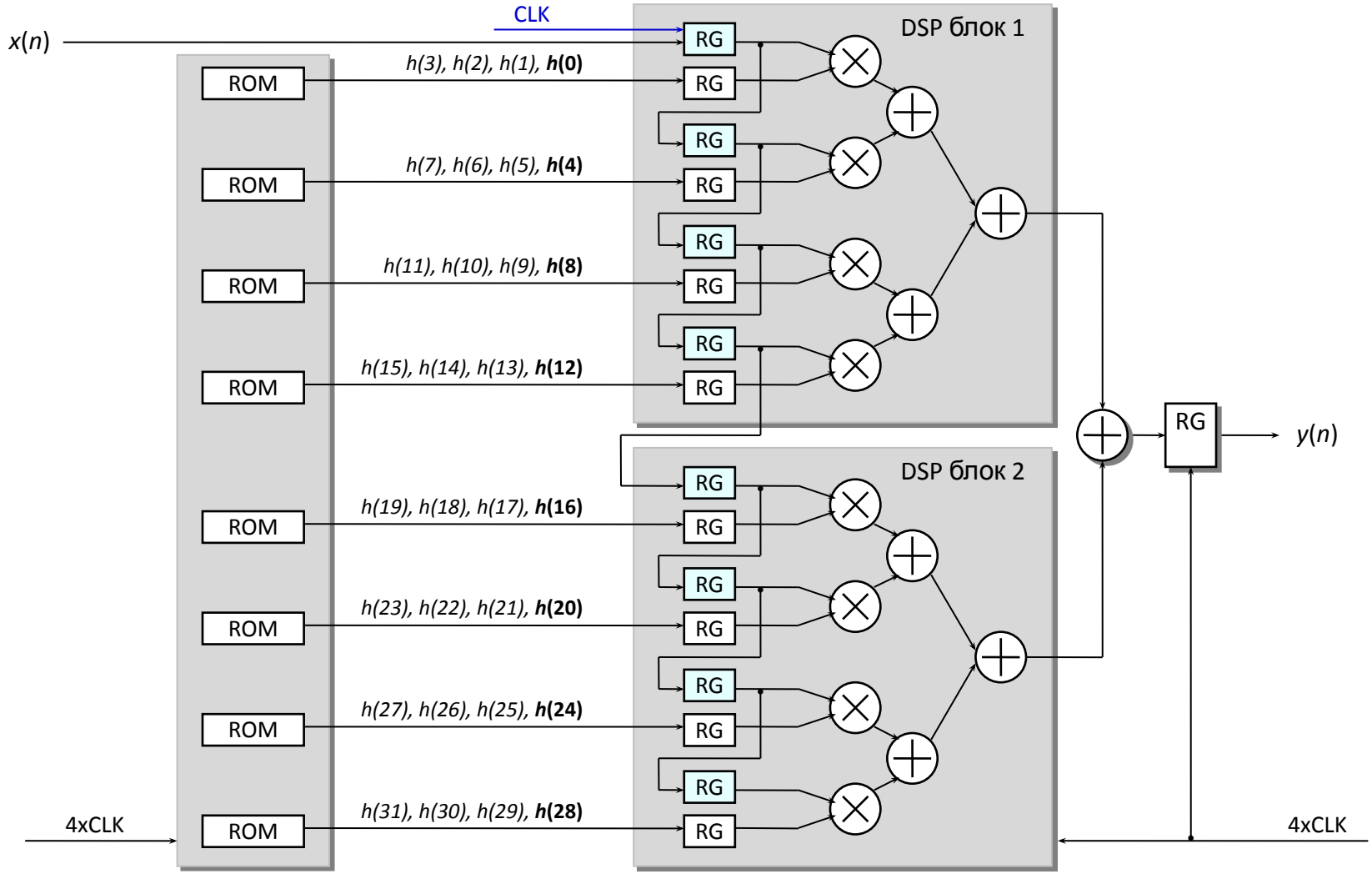


N	Выход DSP блока	Операция	Выход
0	$y_0 = x(n-1)h(1) + x(n-3)h(3) + x(n-5)h(5) + x(n-7)h(7)$	Запись в RG T	–
1	$y_1 = x(n)h(0) + x(n-2)h(2) + x(n-4)h(4) + x(n-6)h(6)$	Результат	$y(n) = y_0+y_1$
2	$y_2 = x(n)h(1) + x(n-2)h(3) + x(n-4)h(5) + x(n-6)h(7)$	Запись в RG T	–
3	$y_3 = x(n+1)h(0) + x(n-1)h(2) + x(n-3)h(4) + x(n-5)h(6)$	Результат	$y(n+1) = y_2+y_3$
4	$y_4 = x(n+1)h(1) + x(n-1)h(3) + x(n-3)h(5) + x(n-5)h(7)$	Запись в RG T	–
5	$y_5 = x(n+2)h(0) + x(n)h(2) + x(n-2)h(4) + x(n-4)h(6)$	Результат	$y(n+2) = y_4+y_5$

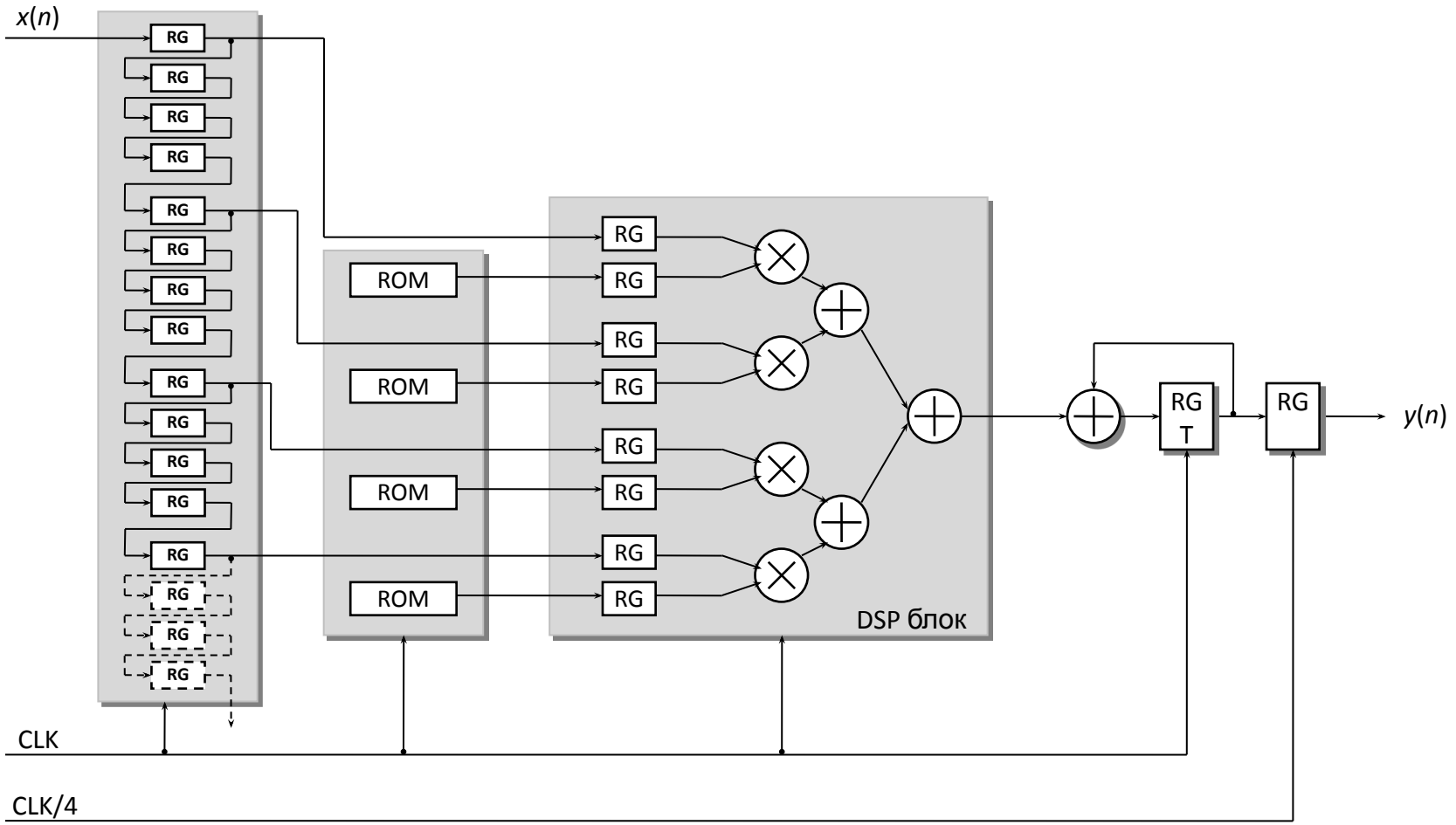
Использование RAM и DSP блоков (TDM FIR - диаграммы)



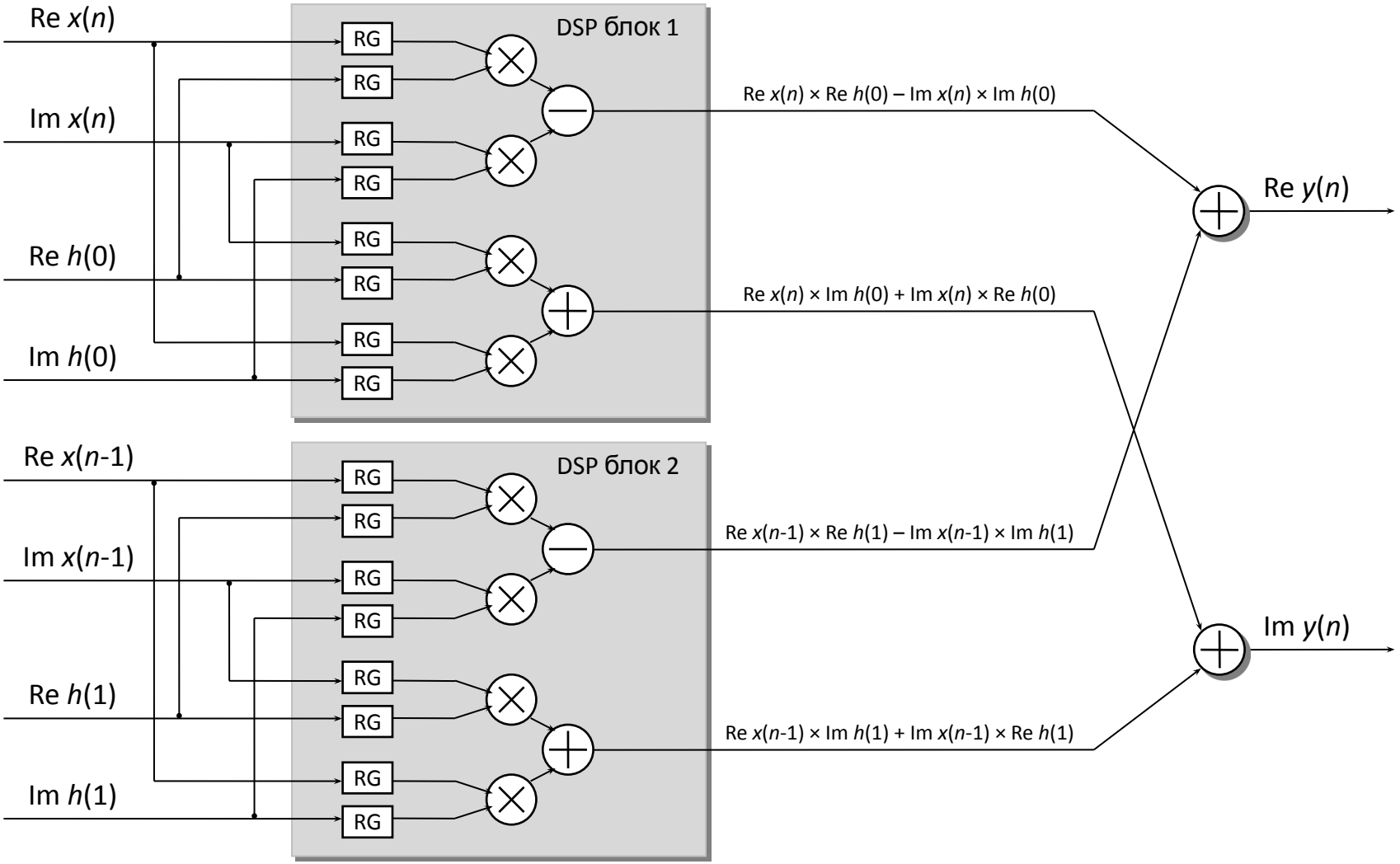
Использование RAM и DSP блоков (интерполятор)



Использование RAM и DSP блоков (дециматор)

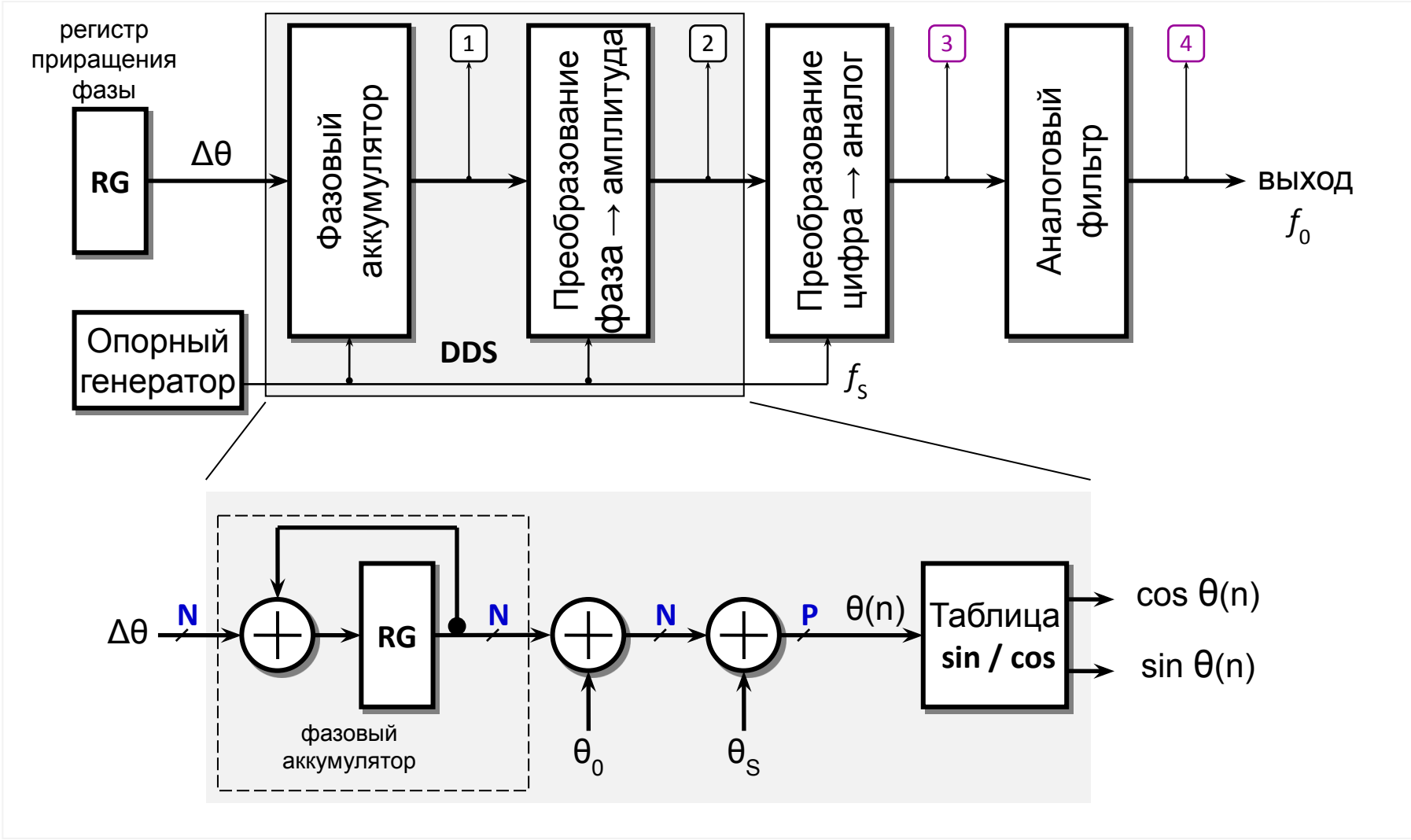


Использование RAM и DSP блоков (комплексный Tap-2 FIR)

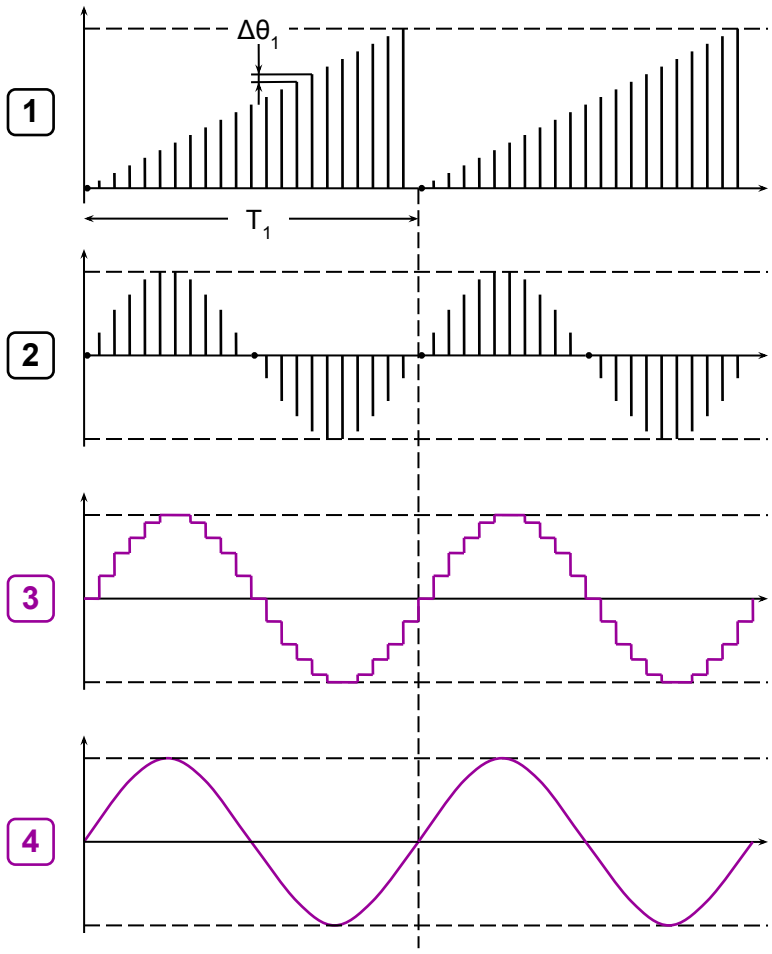


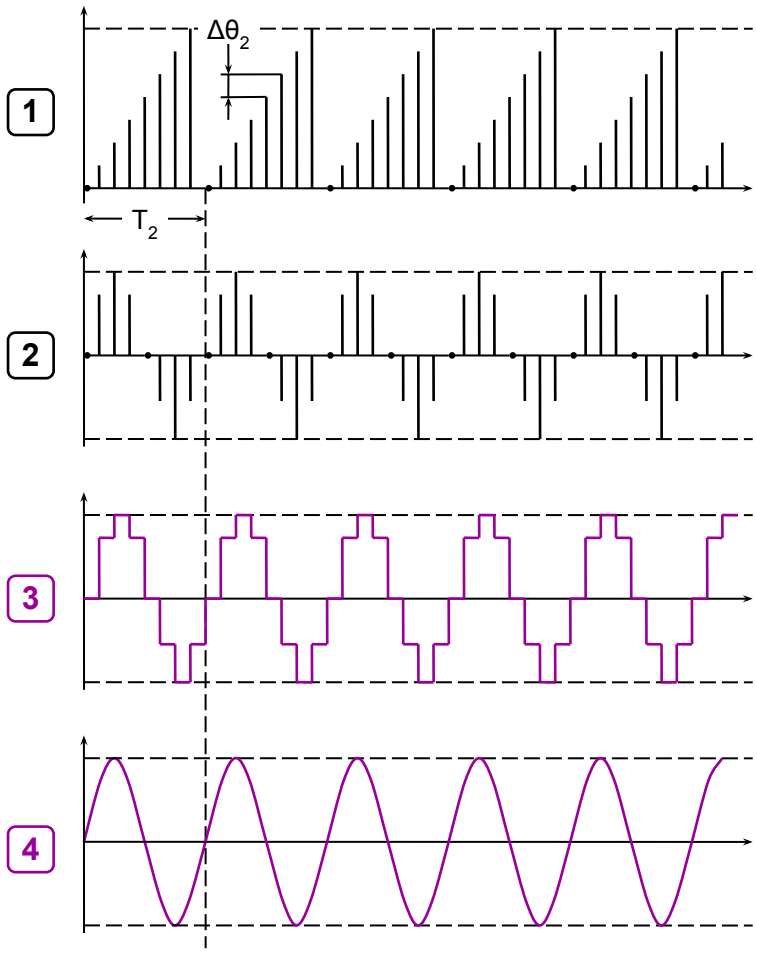
Цифровой синтез сигналов

Цифровой синтезатор сигналов (DDS) – структура



Цифровой синтезатор сигналов (DDS) – диаграммы





Цифровой синтезатор сигналов (DDS) – формулы

Разрядность фазового аккумулятора

$$N = \left\lceil \log_2 \frac{f_S}{\Delta f} \right\rceil$$

Точность установки частоты

$$\Delta f = \frac{f_S}{2^N}$$

Код приращения фазы

$$\Delta\theta = \left\lceil \frac{f_0 \times 2^N}{f_S} \right\rceil$$

Значение выходной частоты

$$f_0 = \frac{\Delta\theta \times f_S}{2^N}$$

Обозначения:
f_S – частота опорного генератора;
f_0 – выходная (формируемая) частота;
Δf – точность установки частоты;
N – разрядность фазового аккумулятора;
$\Delta\theta$ – код приращения фазы.

Пример

Сформировать синусоидальный сигнал f_0 частотой **17 325 761** Гц с точностью **0,5** Гц. Частота опорного генератора f_S равна **100** МГц

Разрядность фазового аккумулятора

$$N = \left\lceil \log_2 \frac{f_S}{\Delta f} \right\rceil = \left\lceil \log_2 \frac{10^8}{0{,}5} \right\rceil = 28$$

Код приращения фазы

$$\Delta\theta = \left\lceil \frac{f_0 \times 2^N}{f_S} \right\rceil = \left\lceil \frac{17325761 \times 2^{28}}{10^8} \right\rceil = 46\,508\,486$$

Точное значение выходной частоты

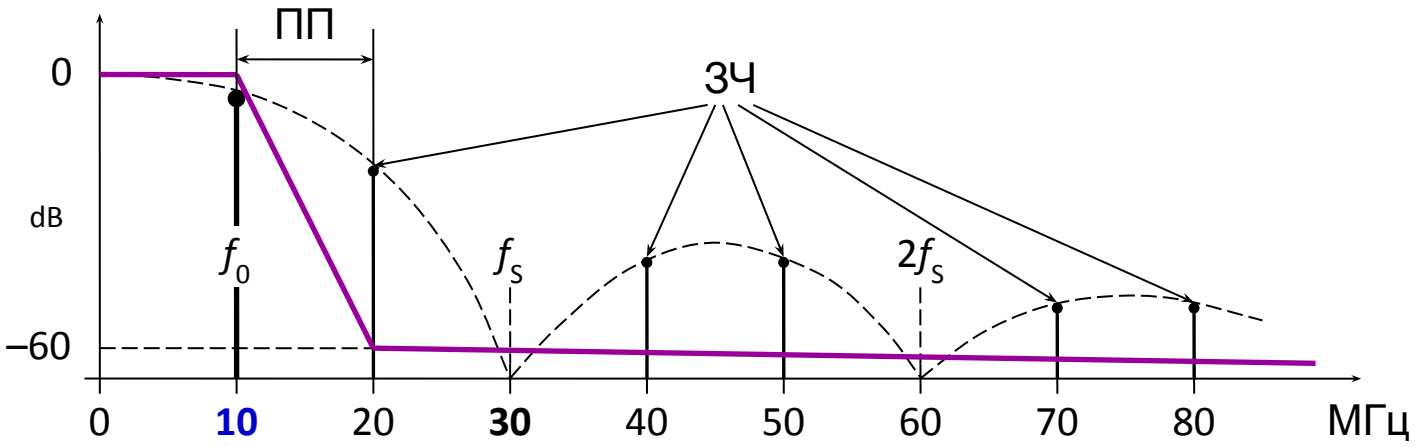
$$f_0 = \frac{\Delta\theta \times f_S}{2^N} = \frac{46\,508\,486 \times 10^8}{2^{28}} = 17\,325\,761{,}1692$$

Ошибка выходной частоты

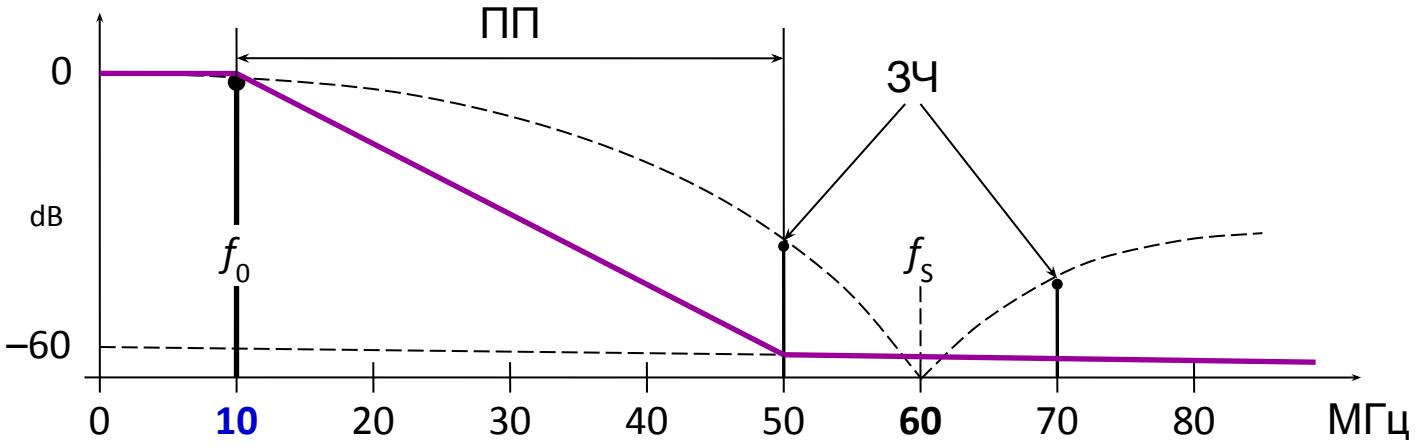
0,1692 Гц

Цифровой синтезатор сигналов (DDS) – фильтрация

f_0 = 10 МГц
f_S = 30 МГц
ПП = 10 МГц



f_0 = 10 МГц
f_S = 60 МГц
ПП = 40 МГц



Цифровые модуляторы (слайд 1)

Гармонический сигнал

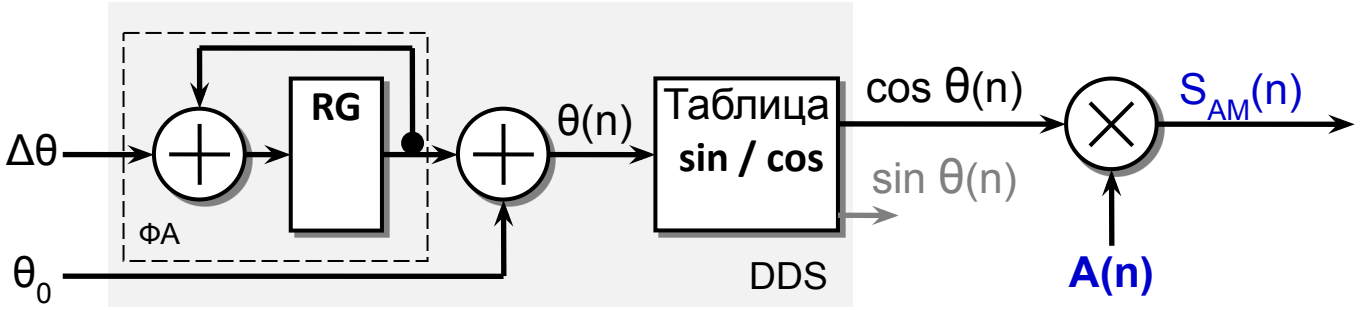
$$x(n) = Re\left\{exp\left(j\frac{2\pi f_0}{f_S}n + \varphi_0\right)\right\} = \cos\left(\frac{2\pi f_0}{f_S}n + \varphi_0\right),$$ где

круговая частота $\omega_0(n) = 2\pi\frac{\Delta\theta}{2^N}\cdot n$ и начальная фаза $\varphi_0 = 2\pi\frac{\theta_0}{2^N}$

Сигнал с модуляцией по амплитуде $S_{AM}(n) = A(n)\cdot\cos\frac{2\pi(\Delta\theta\cdot n + \theta_0)}{2^N}$, где

$\theta(n) = \Delta\theta\cdot n + \theta_0$ – код полной фазы колебания

Структура АМ-модулятора

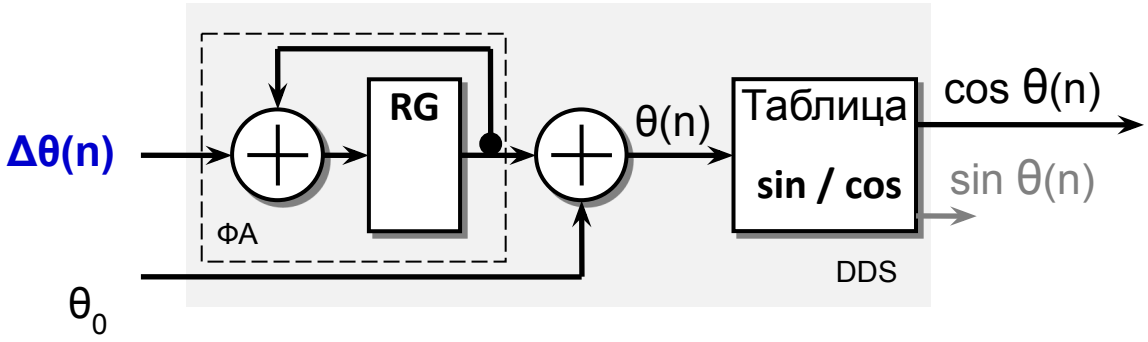


Цифровые модуляторы (слайд 2)

Частотная модуляция

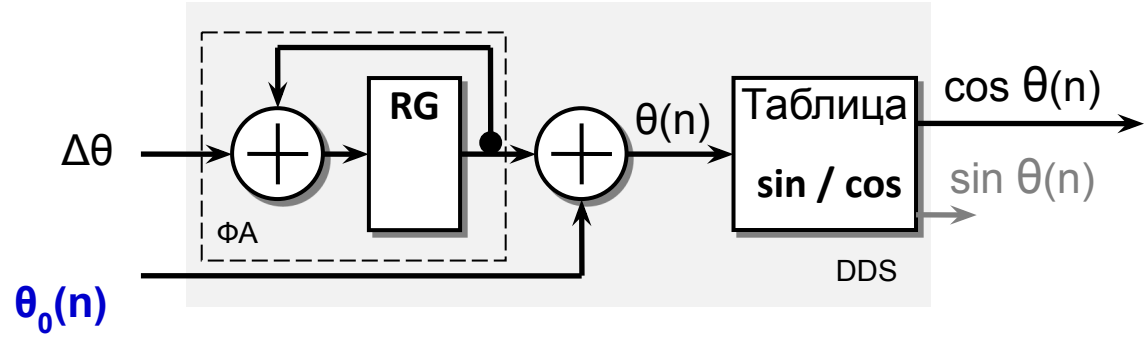
$$S_{\text{ЧМ}}(n) = \cos\frac{2\pi(\theta_0 + \sum \Delta\theta(n))}{2^N}$$

Структура ЧМ-модулятора



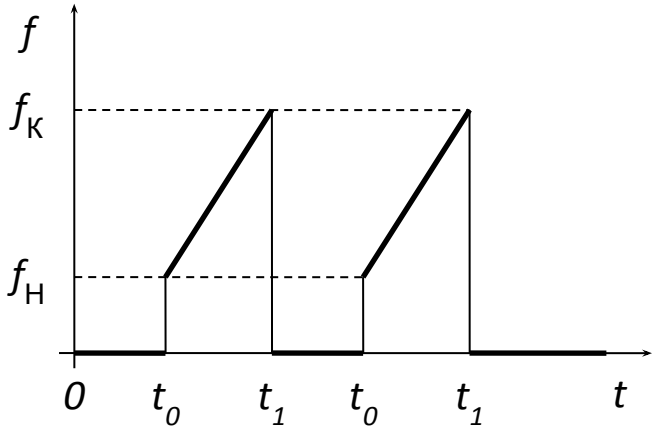
Фазовая модуляция $S_{\text{ФМ}}(n) = \cos\frac{2\pi(\Delta\theta \cdot n + \theta_0(n))}{2^N}$

Структура ФМ-модулятора



Цифровые модуляторы (слайд 3)

Изменение частоты ЛЧМ-сигнала



Начальный и конечный коды приращения фазы

$$\Delta\theta_Н = \frac{f_Н \times 2^N}{f_S} \qquad \Delta\theta_К = \frac{f_К \times 2^N}{f_S}$$

Дискретная длительность ЛЧМ-импульса

$$K = [(t_1 - t_0)f_S]$$

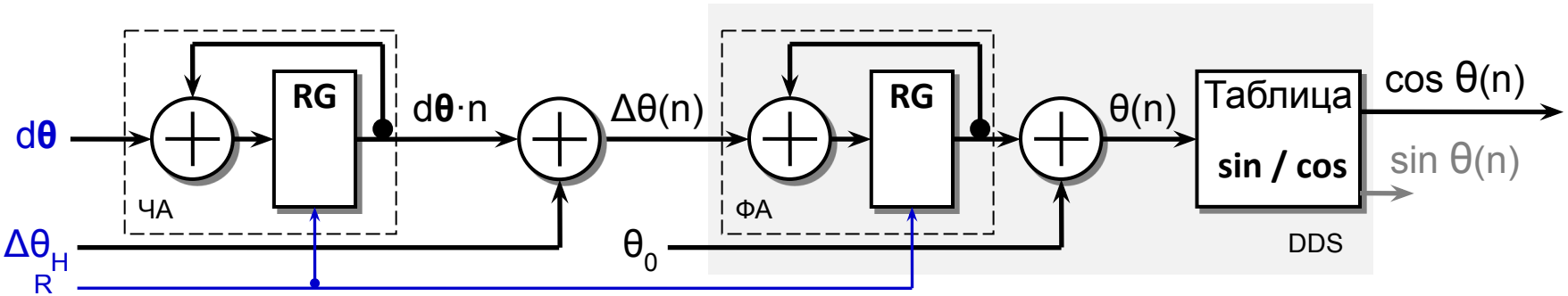
Код приращения частоты

$$d\theta = \frac{\theta_К - \theta_Н}{K}$$

Полная фаза DDS

$$\Delta\theta(n) = \Delta\theta_Н + d\theta \cdot n \text{ для } n = n_0 \ldots (n_0 + K)$$

Структура ЛЧМ-модулятора



Цифровые модуляторы (слайд 4)

Исходные данные:

начальная и конечная частота ЛЧМ-сигнала $f_{\text{Н}} = 9\ \text{МГц} \quad f_{\text{К}} = 11\ \text{МГц}$

системная частота $f_S = 100\ \text{МГц}$

длительность сигнала $t_1 - t_0 = 10\ \text{мкс}$

разрядность фазового аккумулятора $N = 24$

Расчет

$$\Delta\theta_{\text{Н}} = \frac{9 \times 2^{24}}{100} = 1\,509\,949 \qquad \Delta\theta_{\text{К}} = \frac{11 \times 2^{24}}{100} = 1\,845\,494$$

$$K = 10 \cdot 10^{-6} \times 100 \cdot 10^{6} = 1000$$

$$d\theta = \frac{1\,845\,494 - 1\,509\,949}{1000} = \frac{335\,545}{1000} = 336$$

Точные значения начальной и конечной частоты

$$F_{\text{Н}} = \frac{1\,509\,949 \times 100 \cdot 10^{6}}{2^{24}} = 8\,999\,997{,}4\ \text{Гц}$$

$$F_{\text{К}} = \frac{(1\,509\,949 + 336\,000) \times 100 \cdot 10^{6}}{2^{24}} = 11\,002\,713{,}4\ \text{Гц}$$

Отклонения частот от заданных

$$\Delta F_{\text{Н}} = 2{,}6\ \text{Гц} \qquad \Delta F_{\text{К}} = 2\,713{,}4\ \text{Гц}$$

Цифровые модуляторы (слайд 5)

Модуляция несущей частоты одновременно по амплитуде и фазе

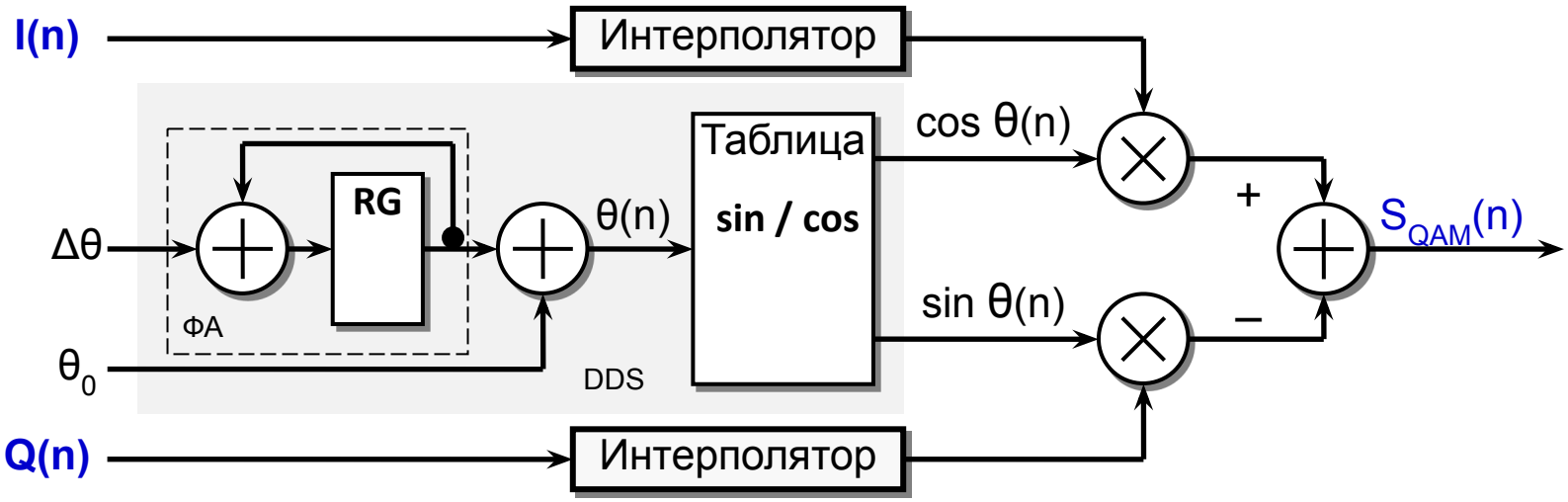
$$s(t) = A(t)\cos\left(\omega_0 t + \varphi(t)\right)$$

$$s(t) = A(t)\cos\omega_0 t\cos\varphi(t) - A(t)\sin\omega_0 t\sin\varphi(t)$$

эквивалентна амплитудной модуляции ее квадратурных компонент квадратурными компонентами модулирующего сигнала

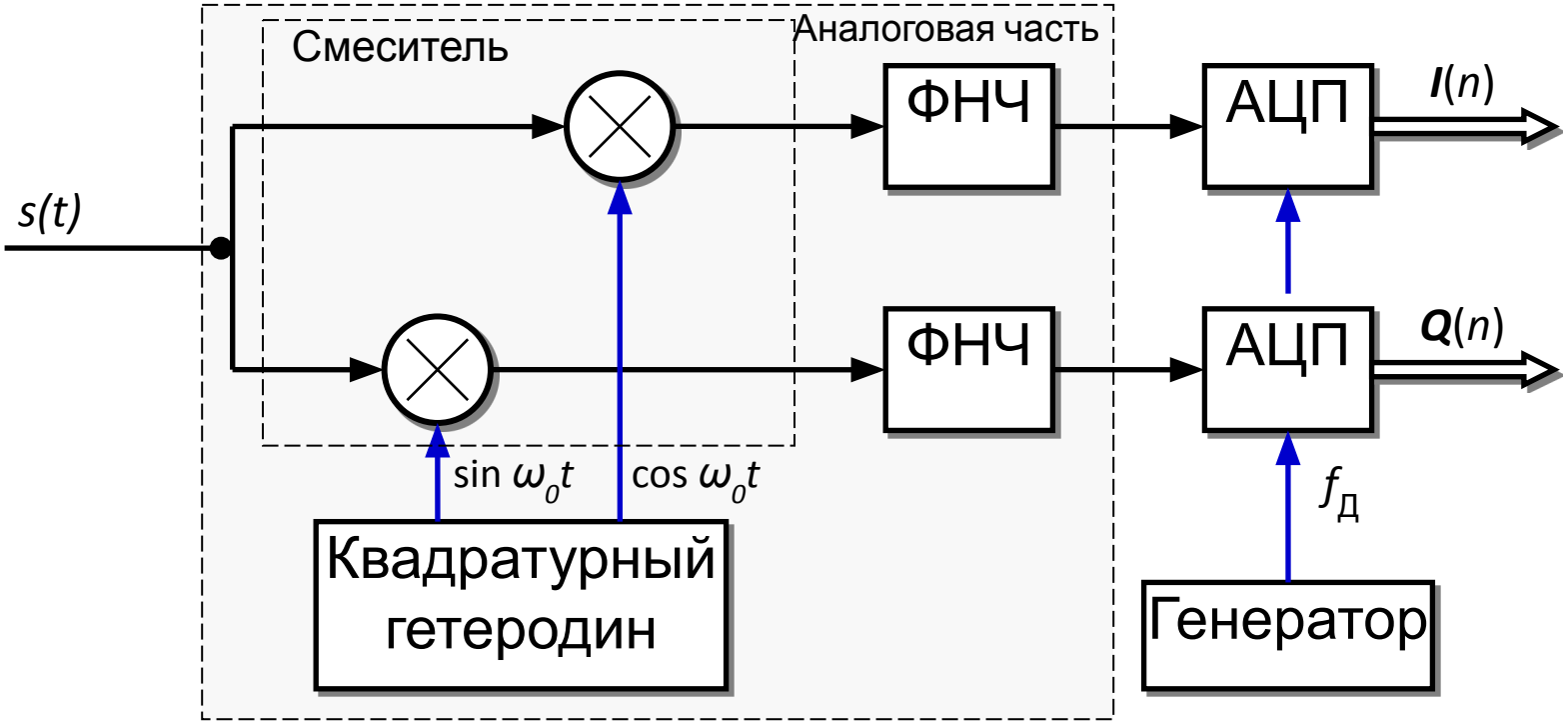
$$s(t) = I(t)\cos\omega_0 t - Q(t)\sin\omega_0 t \quad I(t) = A(t)\cos\varphi(t) \quad Q(t) = A(t)\sin\varphi(t)$$

Структура QAM-модулятора



Вычислители ЦОС с переносом спектра

Квадратурная дискретизация узкополосных сигналов (вариант 1)



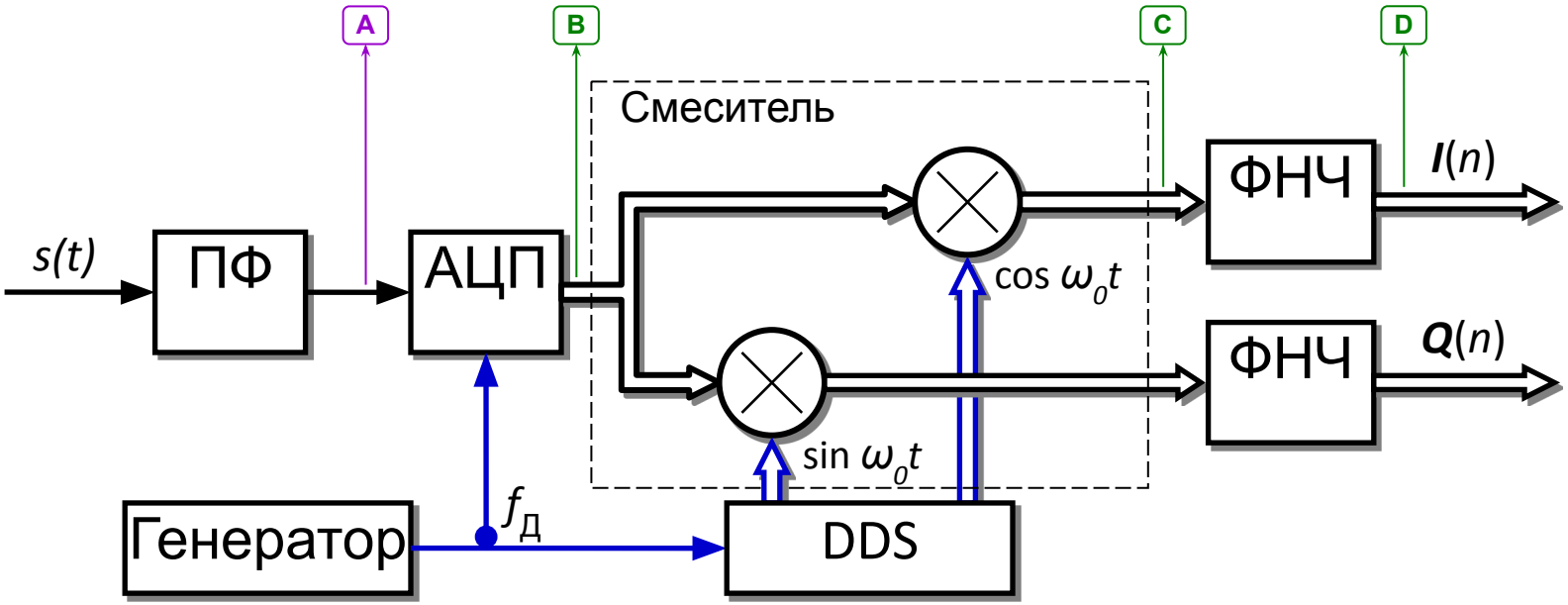
Недостатки

- Трудно реализуемый квадратурный гетеродин
- Трудно реализуемый смеситель с высокой идентичностью каналов
- Аналоговые ФНЧ должны иметь абсолютно одинаковые характеристики, что недостижимо

Достоинства

- Низкая частота дискретизации в АЦП

Квадратурная дискретизация узкополосных сигналов (вариант 2)



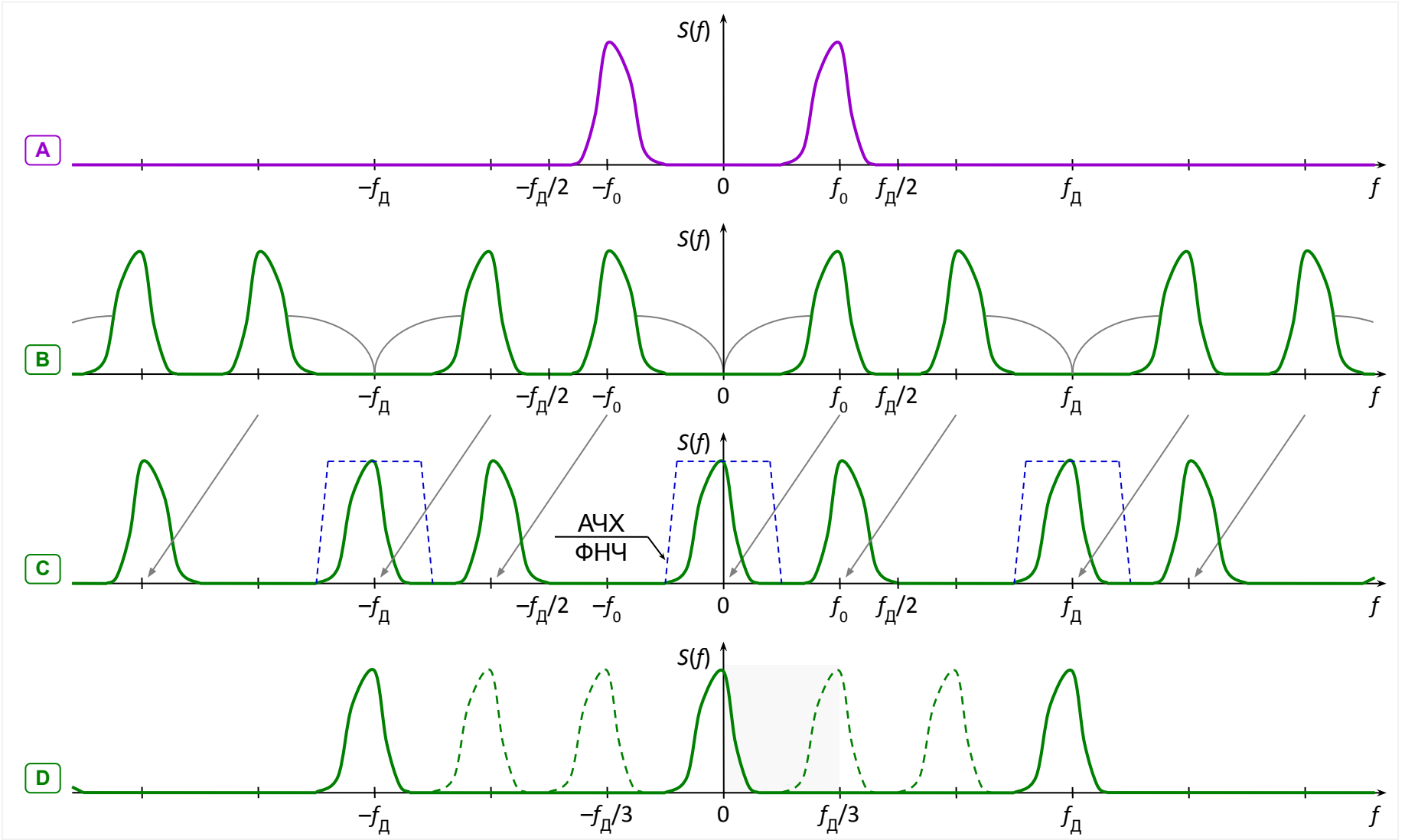
Недостатки

- Высокая частота дискретизации в АЦП

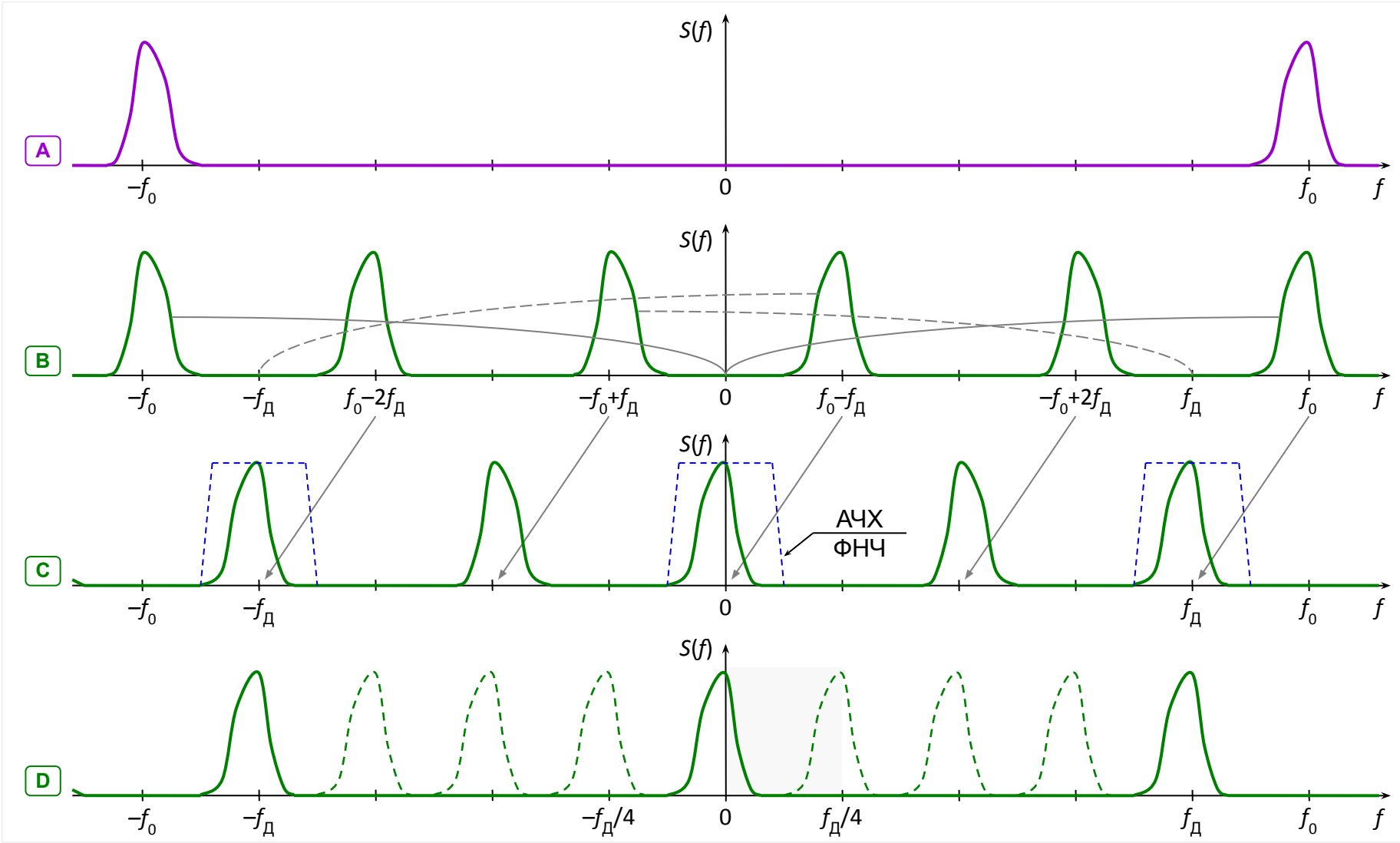
Достоинства

- Высокая точность и стабильность квадратурного гетеродина (DDS)
- Абсолютная идентичность каналов смесителя и ФНЧ

Квадратурная дискретизация – спектры



Квадратурная дискретизация – субдискретизация

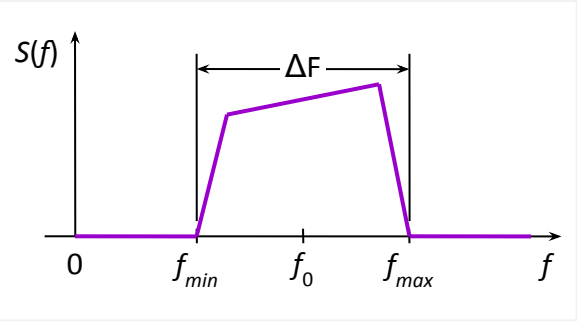


Субдискретизация (слайд 1)

Полоса сигнала $\Delta F = f_{max} - f_{min}$

Средняя частота сигнала (несущая)

$$f_{ср} = f_0 = \frac{f_{max} - f_{min}}{2} = f_{min} + \frac{\Delta F}{2}$$



Узкополосный сигнал $\frac{\Delta F}{f_0} \ll 1$ или иногда $\frac{f_{max}}{f_{min}} \leq 2$

Условие неперекрывающихся при дискретизации копий спектров $f_Д \geq 2f_{max}$

Для узкополосных сигналов (обобщенная теорема Котельникова) $\frac{2f_{max}}{q} \leq f_Д \leq \frac{2f_{min}}{q-1}$

где ***q*** имеет смысл НОМЕРА ДИАПАЗОНА ЧАСТОТ и может принимать только целые значения

$q = 1, 2, \ldots \left\lfloor \frac{f_{max}}{f_{max} - f_{min}} \right\rfloor$,где $\lfloor x \rfloor$ – операция округления до целого, не превосходящего ***x***

Для $q = 1 \rightarrow \frac{2f_{max}}{1} \leq f_Д \leq \frac{2f_{min}}{0} \rightarrow 2f_{max} \leq f_Д \leq \infty$ – теорема Котельникова

Субдискретизация (слайд 2)

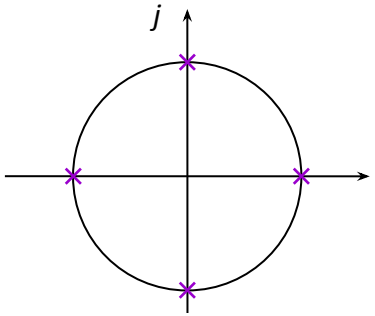
Если несущая частота одной из копий спектра

$$f_0' = \frac{f_Д}{4}$$

гетеродин смесителя, формирующий опорный сигнал

$$g(n) = e^{\pm j \cdot 2\pi n f_0'/f_Д} = e^{\pm j \cdot n\frac{\pi}{2}} = \cos n\frac{\pi}{2} \pm j \cdot \sin n\frac{\pi}{2}$$

сильно упрощается (исключаются умножители).



$g(n)$ = 1 + 0·j
$g(n+1)$ = 0 + 1·j
$g(n+2)$ = –1 + 0·j
$g(n+3)$ = 0 – 1·j
$g(n+4)$ = 1 + 0·j
$g(n+5)$ = 0 + 1·j
. . .

Частоту дискретизации

$$f_Д = 4f_0' = f_{Допт}$$

будем называть "оптимальной". Новая центральная частота копии спектра для диапазона частот номер ***q***

$$f_0' = \frac{f_0}{2q-1}$$

Условие: частота f_0' не должна быть иррациональным числом

Пример выбора частоты дискретизации

Исходные данные $f_{min} = 63$ МГц $f_{max} = 77$ МГц

Получаем $f_0 = 70$ МГц $\Delta F = 14$ МГц

Определяем количество допустимых диапазонов частот дискретизации

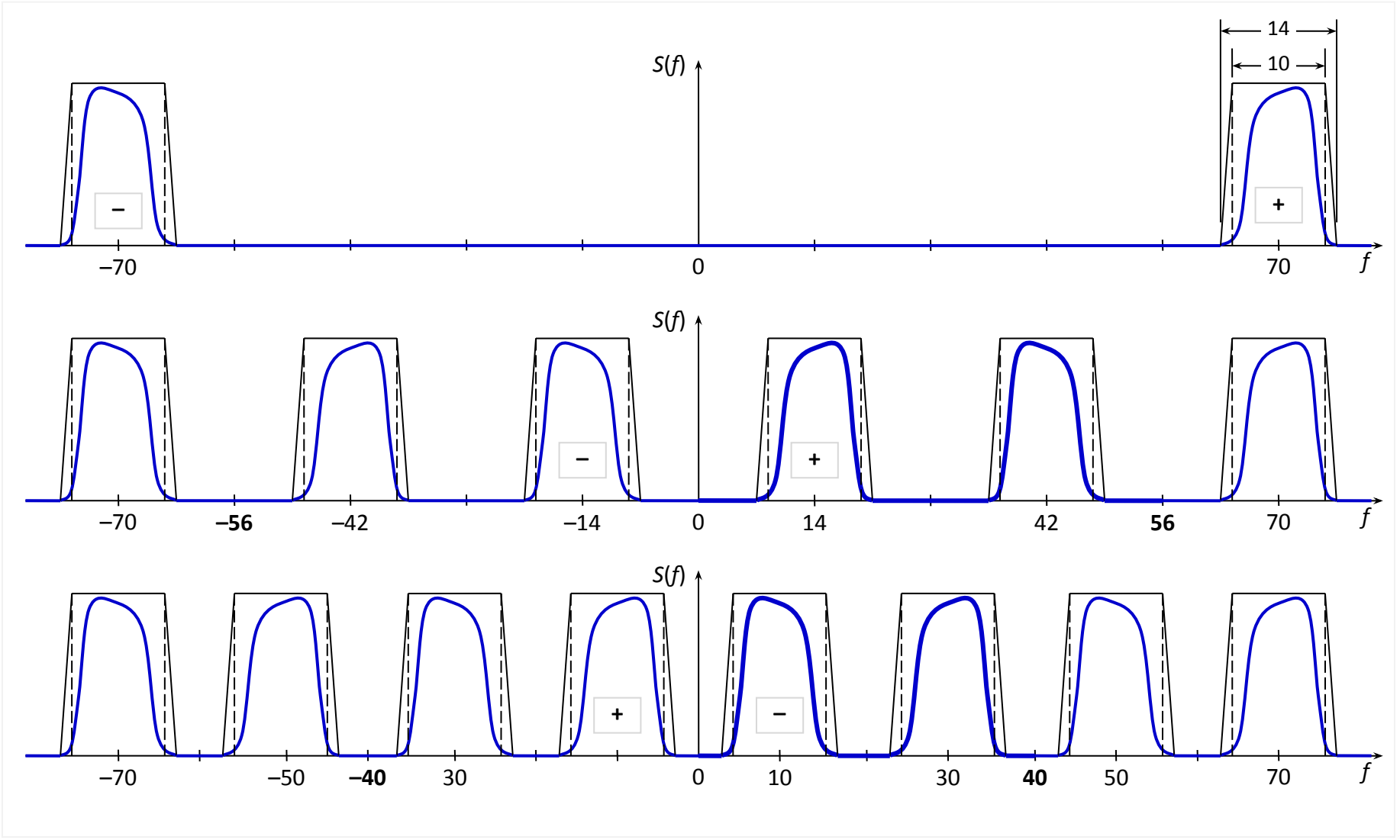
$$q_{max} = \left\lfloor \frac{f_{max}}{\Delta F} \right\rfloor = \left\lfloor \frac{77}{14} \right\rfloor = [5.5] = 5$$

Строим таблицу для ***q*** = 1 ... 5

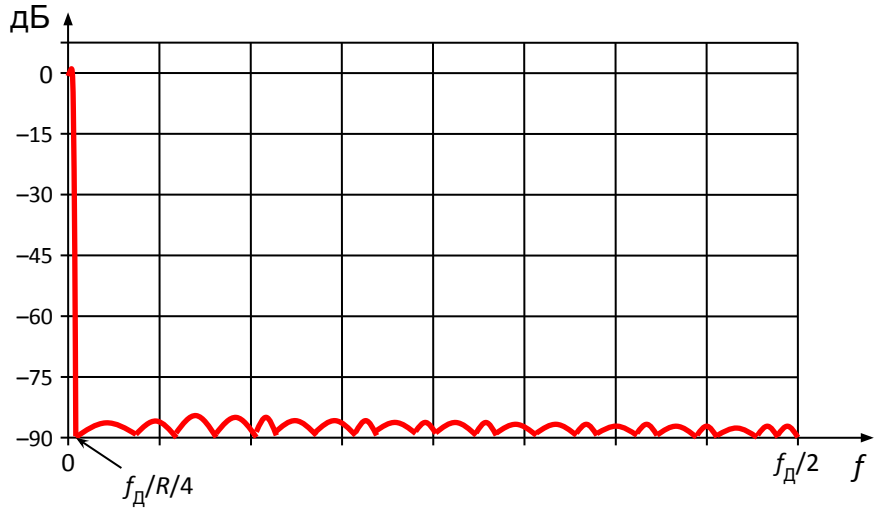
q	$f_Д$ (МГц) $\frac{2f_{max}}{q} \le f_Д \le \frac{2f_{min}}{q-1}$	f_0' (МГц) $f_0' = \frac{f_0}{2q-1}$	$f_{Допт}$=4 f_0' (МГц)	
1	154 ... ∞	70	280	допустимо
2	77 ... 126	$23\,^1/_3$	$93\,^1/_3$	иррациональное число
3	51.333 ... 63	14	**56**	оптимально
4	38.5 ... 42	10	**40**	оптимально (инверсия спектра)
5	30.8 ... 31.5	$7\,^7/_9$	$31\,^1/_9$	иррациональное число

Получаем две "оптимальных" частоты дискретизации **40** и **56** МГц.

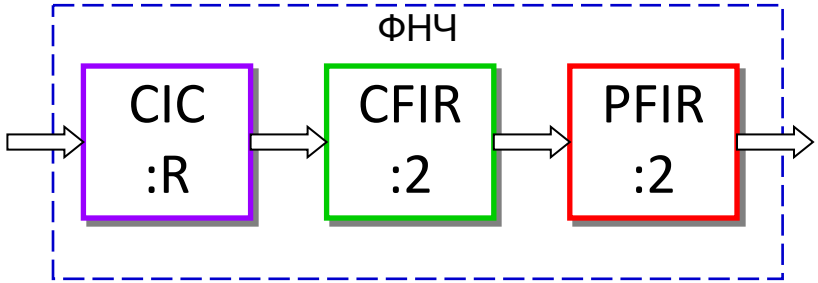
Пример выбора частоты дискретизации – спектры

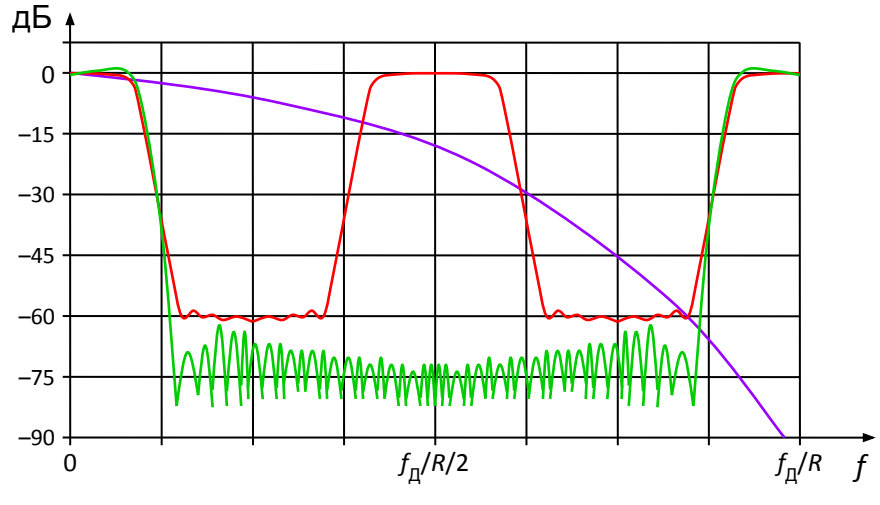


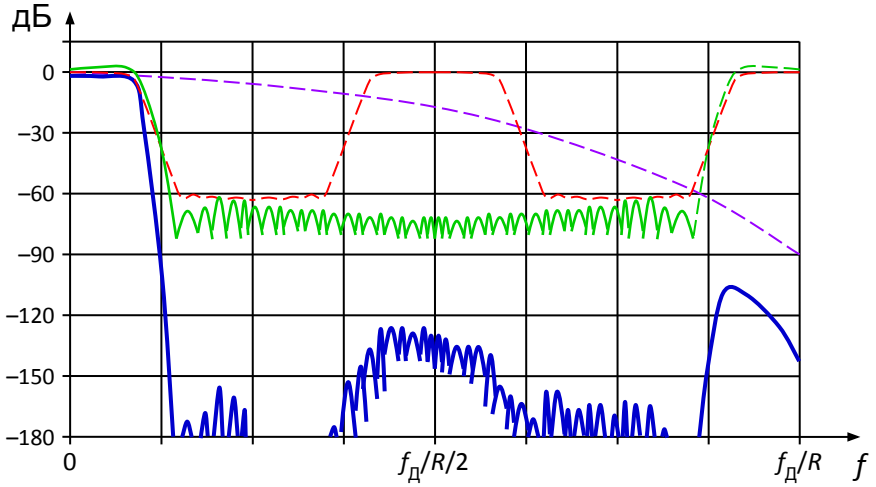
Многокаскадный децимирующий фильтр



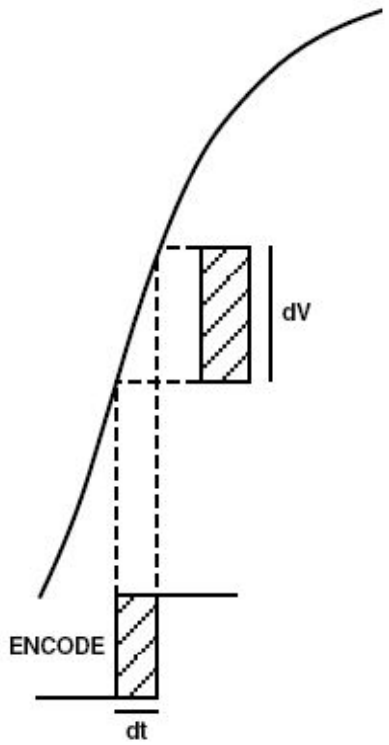
ОЧЕНЬ БОЛЬШОЙ ПОРЯДОК ФИЛЬТРА



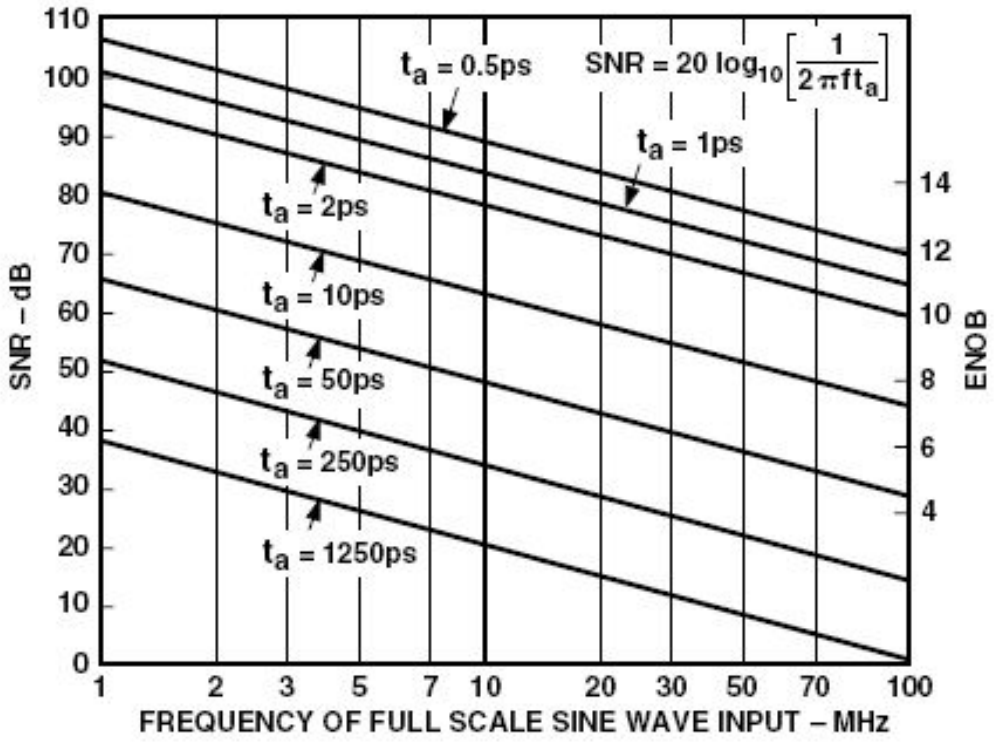




Требования к стабильности частоты дискретизации



Шум, вызванный нестабильностью фронта такта дискретизации (джиттером)

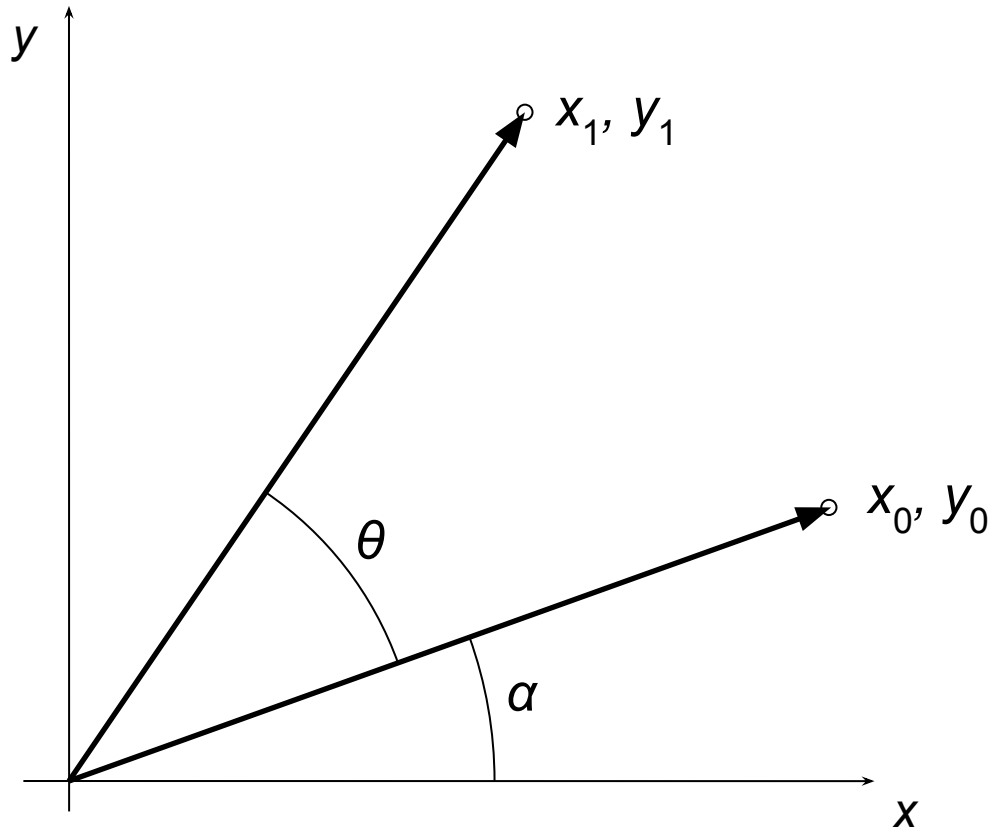


Изменение отношения сигнал/шум (***SNR***) и эффективного количества разрядов АЦП (***ENOB***) в зависимости от джиттера (t_a)

Специальные вычисления в ЦОС

Арифметические основы CORDIC (слайд 1)

Алгоритм координатного вращения (***CO****ordinat* ***R****otation* ***DI****gital* ***C****omputer*) **CORDIC**



$$x_1 = x_0 \cos\theta - y_0 \sin\theta$$

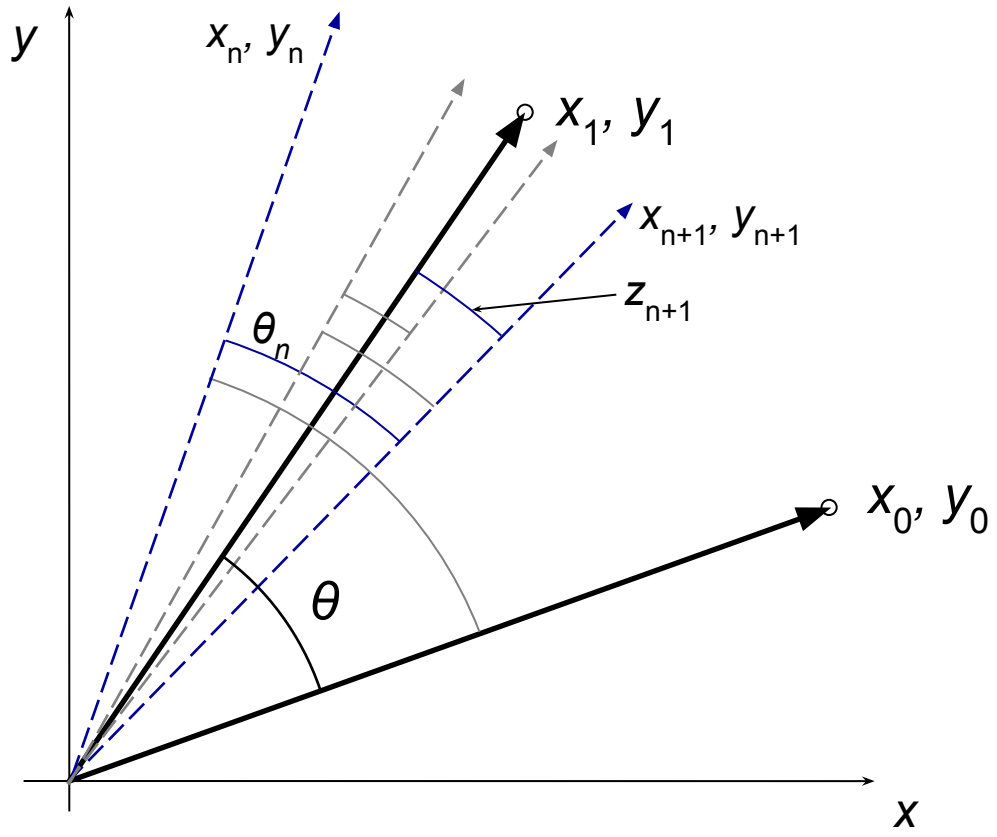
$$y_1 = x_0 \sin\theta + y_0 \cos\theta$$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

$$x_1 = A\cos(\alpha+\theta) = A(\cos\alpha \cdot \cos\theta - \sin\alpha \cdot \sin\theta) = x_0 \cos\theta - y_0 \sin\theta$$

$$y_1 = A\sin(\alpha+\theta) = A(\cos\alpha \cdot \sin\theta + \sin\alpha \cdot \cos\theta) = x_0 \sin\theta + y_0 \cos\theta$$

Арифметические основы CORDIC (слайд 2)



$$\theta = \sum_{n=0}^{\infty} S_n \theta_n$$

$$S_n = \{-1; +1\}$$

$$z_{n+1} = z_n - S_n \theta_n$$

$$z_{n+1} = \theta - \sum_{i=0}^{n} S_i \theta_i$$

$$z_0 = \theta$$

Арифметические основы CORDIC (слайд 3)

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} \cos\theta_n & -\sin\theta_n \\ \sin\theta_n & \cos\theta_n \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix}$$

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \cos\theta_n \begin{bmatrix} 1 & -\operatorname{tg}\theta_n \\ \operatorname{tg}\theta_n & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix}$$

$$\theta_n = \operatorname{arctg} 2^{-n}$$

Ряд $\theta = \sum_{n=0}^{\infty} S_n \theta_n$ сходится для $\theta \in \left[-\frac{\pi}{2}; \frac{\pi}{2}\right]$

при соответствующем выборе угла поворота S_n

Арифметические основы CORDIC (слайд 4)

Подставляя $\operatorname{tg}\theta_n = \operatorname{tg}(\operatorname{arctg} 2^{-n}) = 2^{-n}$

получим $$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \cos\theta_n \begin{bmatrix} 1 & -S_n 2^{-n} \\ S_n 2^{-n} & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix}$$

где $\cos\theta_n = \cos(\operatorname{arctg} 2^{-n}) = const$

Это позволяет исключить масштабирование на каждой итерации алгоритма, и выполнить его только на завершающем шаге. Общий **масштабирующий коэффициент** алгоритма ***K*** определяется

$$K = \prod_{n=0}^{\infty} \cos(\operatorname{arctg} 2^{-n}) \approx 0.6072529350$$

Величина $P = \frac{1}{K} \approx 1.64676$ называется **радиусом вращения**

Арифметические основы CORDIC (слайд 5)

Последовательность *Sn* определяет направление поворота вектора на каждой итерации. Значения этой последовательности определяются в ходе вычислений, поэтому алгоритм *CORDIC* называют ***алгоритмом с динамическим выбором направления вращения***.

Формы алгоритма

ПРЯМАЯ ($z \to 0$)

$$S_n = \begin{cases} +1, z_n \geq 0 \\ -1, z_n < 0 \end{cases}$$

$$\begin{cases} x_{n+1} = x_n - mS_n 2^{-n} y_n \\ y_{n+1} = y_n + S_n 2^{-n} x_n \\ z_{n+1} = z_n - S_n \varepsilon_n \end{cases}$$

ИНВЕРСНАЯ ($y \to 0$)

$$S_n = \begin{cases} +1, y_n \geq 0 \\ -1, y_n < 0 \end{cases}$$

$$\begin{cases} x_{n+1} = x_n + mS_n 2^{-n} y_n \\ y_{n+1} = y_n - S_n 2^{-n} x_n \\ z_{n+1} = z_n + S_n \varepsilon_n \end{cases}$$

Параметр ***m*** определяет тип функции CORDIC, а ***εn*** – набор (обычно таблица) заранее вычисленных констант, определяемых типом функции.

Функции алгоритма CORDIC

Тригонометрические:

$m = 1$

$\varepsilon_n = \operatorname{arctg} 2^{-n}$

$K = \prod_{n=0}^{\infty} \cos(\operatorname{arctg} 2^{-n}) \approx 0.6072529350$

$P = \frac{1}{K} \approx 1.64676$

$n = 0 \dots i$

Гиперболические:

$m = -1$

$\varepsilon_n = \operatorname{arcth} 2^{-n}$

$K = \prod_{n=0}^{\infty} \cosh(\operatorname{arcth} 2^{-n}) \approx 1.2051$

$P = \frac{1}{K} \approx 0.8299$

$n = 1 \dots i$

Линейные: $m = 0 \;; \varepsilon_n = 2^{-n} \;; K = 1 \;; P = 1 \;; n = 1 \dots i$

Преобразования алгоритма в общем виде

$$[x, y, z] \rightarrow [x', y', z']$$

Применение алгоритма CORDIC (слайд 1)

Тригонометрические функции

Прямая форма

$$[x, y, z] \rightarrow [P(x\cos z - y\sin z), P(y\cos z + x\sin z), 0]$$

Частные случаи прямой формы

$$[K, 0, a] \rightarrow [\cos a, \sin a, 0]$$

$$[x, 0, z] \rightarrow [Px\cos z, Px\sin z, 0]$$

Инверсная форма

$$[x, y, z] \rightarrow \left[P\sqrt{x^2 + y^2}, 0, z + \operatorname{arctg} {}^{y}/_{x}\right]$$

Частные случаи инверсной формы

$$[1, a, 0] \rightarrow \left[P\sqrt{1 + a^2}, 0, \operatorname{arctg} a\right]$$

$$[x, y, 0] \rightarrow \left[P\sqrt{x^2 + y^2}, 0, \operatorname{arctg} {}^{y}/_{x}\right]$$

Применение алгоритма CORDIC (слайд 2)

Гиперболические функции

Прямая форма

$$[x, y, z] \to [P(x\cosh z + y\sinh z), P(y\cosh z + x\sinh z), 0]$$

Частные случаи прямой формы

$$[K, 0, a] \to [\cosh a, \sinh a, 0]$$

$$[K, K, a] \to [\exp a, \exp a, 0]$$

Инверсная форма

$$[x, y, z] \to \left[P\sqrt{x^2 - y^2}, 0, z + \operatorname{arcth} {}^{y}/_{x}\right]$$

Частные случаи инверсной формы

$$[1, a, 0] \to \left[P\sqrt{1 - a^2}, 0, \operatorname{arcth} a\right]$$

$$[x, y, 0] \to \left[P\sqrt{x^2 - y^2}, 0, \operatorname{arcth} {}^{y}/_{x}\right]$$

$$[a + 1, a - 1, 0] \to \left[2P\sqrt{a}, 0, \frac{\ln a}{2}\right]$$

$$\left[a + \left(\frac{K}{2}\right)^2, a - \left(\frac{K}{2}\right)^2, 0\right] \to \left[\sqrt{a}, 0, \frac{\ln\left(a \times \left({}^{2}/_{K}\right)^2\right)}{2}\right]$$

$$\left[a + \left(\frac{K}{2}\right)^2, a - \left(\frac{K}{2}\right)^2, -\ln\left(\frac{K}{2}\right)\right] \to \left[\sqrt{a}, 0, \frac{\ln a}{2}\right]$$

Применение алгоритма CORDIC (слайд 3)

Линейные функции

Прямая форма

$$[x, y, z] \rightarrow [x, y + x \times z, 0]$$

Частный случай прямой формы

$$[x, 0, z] \rightarrow [x, x \times z, 0]$$

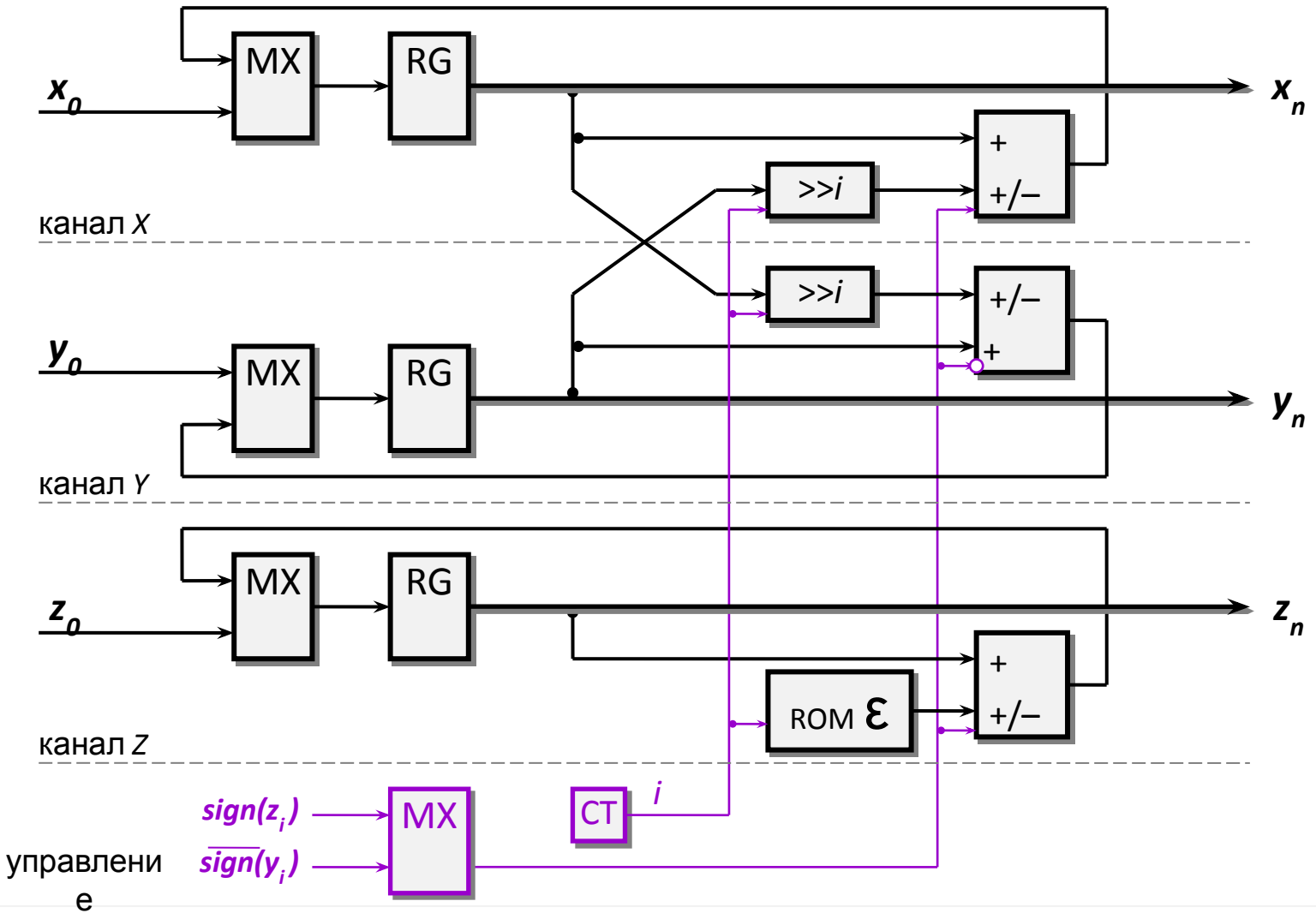
Инверсная форма

$$[x, y, z] \rightarrow \left[x, 0, z + \frac{y}{x}\right]$$

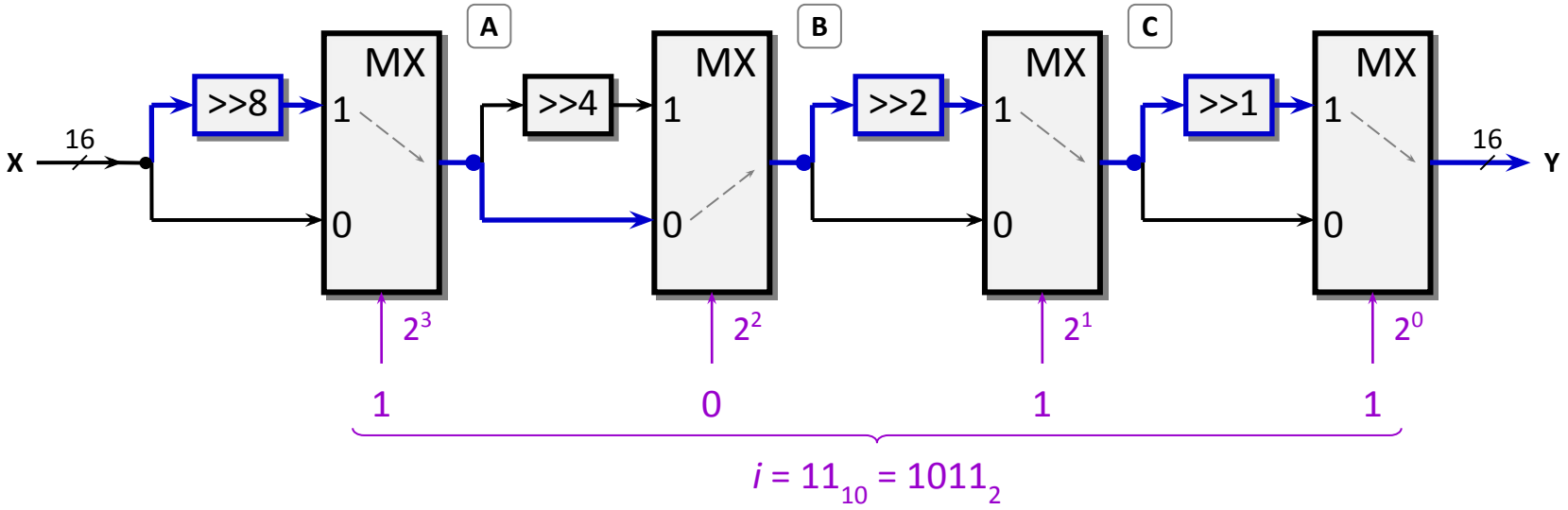
Частный случай инверсной формы

$$[x, y, 0] \rightarrow \left[x, 0, \frac{y}{x}\right]$$

Структура вычислителя CORDIC (ядро – CORE)



Структура вычислителя CORDIC (блок аппаратного сдвига)



Примеры арифметического сдвига вправо

	$i = 1011_2$	$i = 0010_2$	$i = 0000_2$
X	$1110\ 0000\ 1111\ 0101_2$	$1110\ 0000\ 1111\ 0101_2$	$1110\ 0000\ 1111\ 0101_2$
A	$1111\ 1111\ 1110\ 0000_2$	$1110\ 0000\ 1111\ 0101_2$	$1110\ 0000\ 1111\ 0101_2$
B	$1111\ 1111\ 1110\ 0000_2$	$1110\ 0000\ 1111\ 0101_2$	$1110\ 0000\ 1111\ 0101_2$
C	$1111\ 1111\ 1111\ 1000_2$	$1111\ 1000\ 0011\ 1101_2$	$1110\ 0000\ 1111\ 0101_2$
Y	$1111\ 1111\ 1111\ 1100_2$	$1111\ 1000\ 0011\ 1101_2$	$1110\ 0000\ 1111\ 0101_2$

Содержимое ROM ядра CORDIC

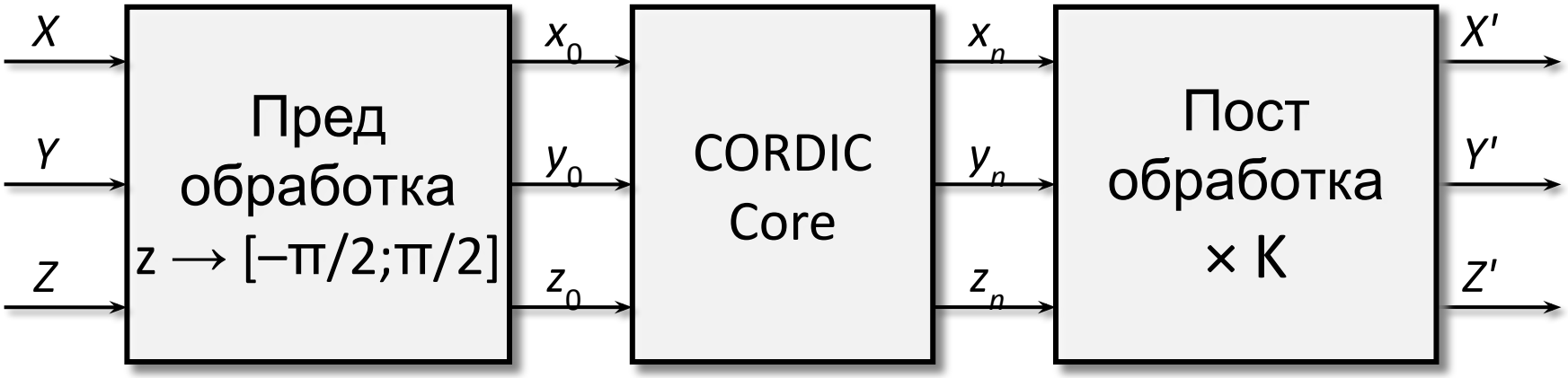
Содержимое постоянной памяти арктангенсов *ROM* вычисляется заранее любым доступным методом. При переходе к конечной разрядной сетке значение арктангенса умножается на коэффициент, который равен 2^{k-1}, где k – выбранная разрядность. Множитель учитывает знак числа.

Пример содержимого *ROM* для 16-разрядного представления арктангенса приведен в **таблице**. В этом случае угол величиной **π** радиан соответствует коду 2^{15}=32768

Таблица

N	Арктангенс(рад)	Dec	Hex
0	0.7853981634	8192	0x2000
1	0.4636476090	4836	0x12E4
2	0.2449786631	2555	0x09FB
3	0.1243549945	1297	0x0511
4	0.0624188100	0651	0x028B
5	0.0312398334	0325	0x0145
6	0.0156237286	0162	0x00A2
7	0.0078123411	0081	0x0051
8	0.0039062301	0040	0x0028
9	0.0019531225	0020	0x0014
10	0.0009765622	0010	0x000A
11	0.0004882812	0005	0x0005
12	0.0002441406	0002	0x0002
13	0.0001220703	0001	0x0001
14	0.0000610352	0000	0x0000
15	0.0000305176	0000	0x0000

Полная структура вычислителя CORDIC



Отображение исходного вектора в правую координатную полуплоскость (коррекция *Z*), в соответствии с областью сходимости ряда θ, ограниченной интервалом ± π/2.

Вращение вектора. Для тригонометрических функций следует учитывать радиус вращения P≈1.64676, что требует дополнительного разряда в арифметических блоках.

При необходимости, **коррекция** результатов вычислений

Предобработка CORDIC

Прямая форма

Вариант 1

$$-\frac{\pi}{2} \le \theta \le \frac{\pi}{2}$$

$$z_0 = \theta$$

$$x_0 = x$$

$$y_0 = y$$

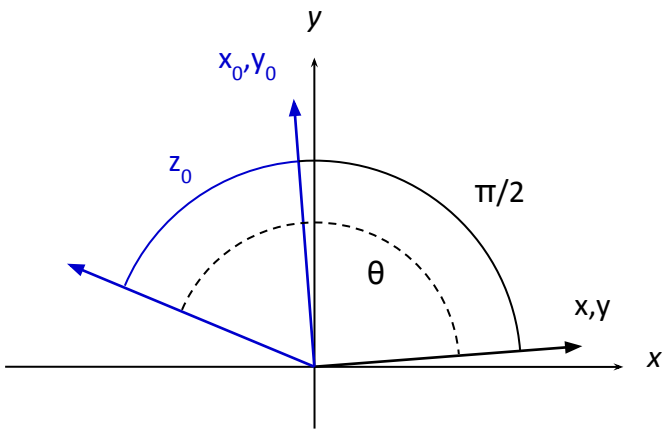
Вариант 2

$$\frac{\pi}{2} < \theta < \pi$$

$$z_0 = \theta - \frac{\pi}{2}$$

$$x_0 = -y$$

$$y_0 = x$$



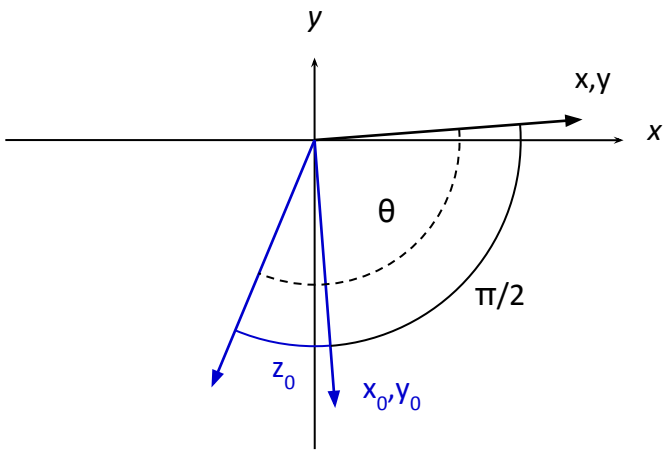
Вариант 3

$$-\pi < \theta < -\frac{\pi}{2}$$

$$z_0 = \theta + \frac{\pi}{2}$$

$$x_0 = y$$

$$y_0 = -x$$



Предобработка CORDIC

Инверсная форма

Вариант 1

$x > 0$

$z_0 = 0$

$x_0 = x$

$y_0 = y$

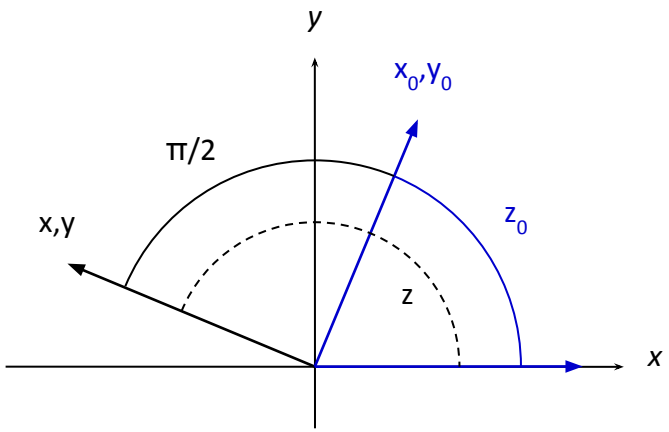
Вариант 2

$x < 0, y > 0$

$z_0 = \frac{\pi}{2}$

$x_0 = y$

$y_0 = -x$



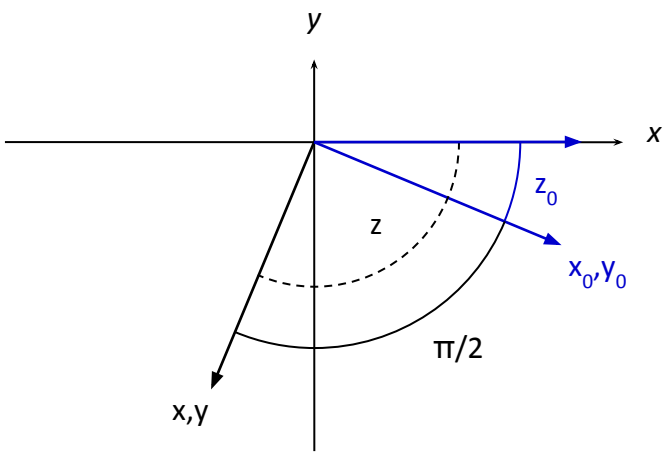
Вариант 3

$x < 0, y < 0$

$z_0 = -\frac{\pi}{2}$

$x_0 = -y$

$y_0 = x$



Точность вычислений CORDIC

CORDIC Пример 1. Перевод полярных координат в декартовы (1)

Исходные данные:

$$\rho = 0.999, \varphi = -\pi/3$$

Переход в 16-разрядную сетку:

$$\rho = [0.999 \times 2^{14}] = 16367$$

$$\varphi = \left[(-\pi/3) \times \frac{2^{15}}{\pi}\right] = -10922$$

Инициализация вычислений:

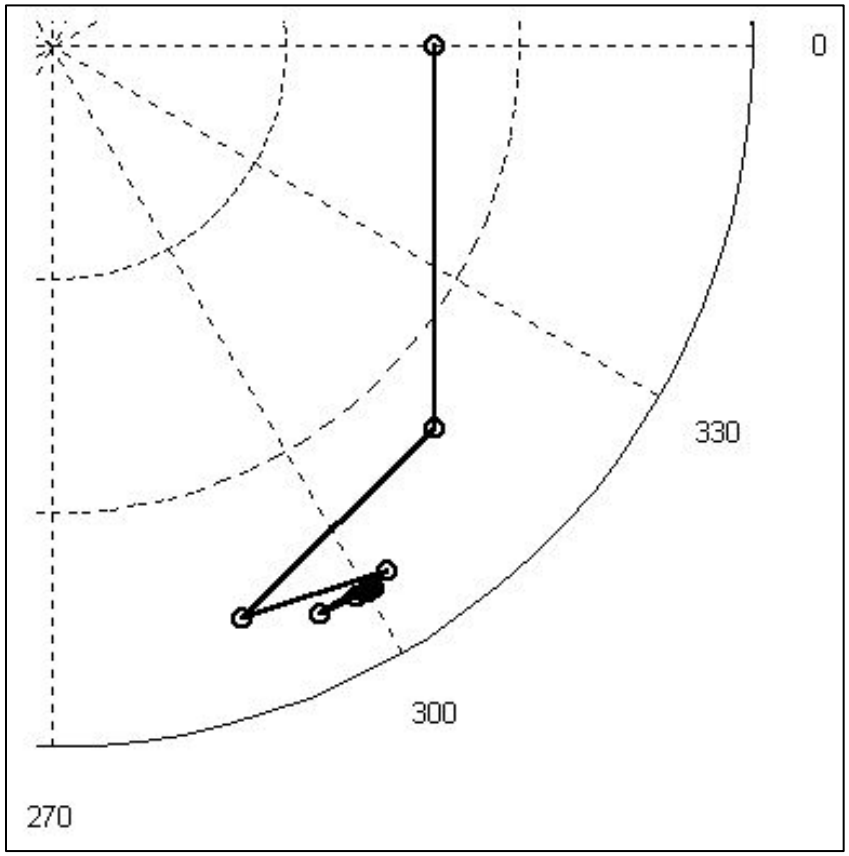
$$x_0 = \rho, y_0 = 0, z_0 = \varphi$$

Значения переменных на итерациях алгоритма показаны в **таблице**, процесс вычислений управляется по знаку переменной ***z*** (прямая форма).

Таблиц

n	z_n	x_n	y_n	s_n
0	-10922	16367	0	-1
1	-2730	16367	-16367	-1
2	2106	8183	-24550	1
3	-449	14321	-22505	-1
4	848	11507	-24295	1
5	197	13026	-23576	1
6	-128	13763	-23169	-1
7	34	13400	-23384	1
8	-47	13583	-23280	-1
9	-7	13492	-23333	-1
10	13	13446	-23359	1
11	3	13469	-23346	1
12	-2	13481	-23340	-1
13	0	13475	-23343	1
14	-1	13478	-23342	-1
15	-1	**13476**	**-23342**	-1

CORDIC Пример 1. Перевод полярных координат в декартовы (2)



Графическое представление поворота вектора

Интерпретация

$$x = 13476 \times K/2^{14} = \frac{13476 \times 0.607253}{16384} = 0.499471$$

$$y = -23342 \times K/2^{14} = \frac{-23342 \times 0.607253}{16384} = -0.865142$$

Проверка

$$\rho \cos\varphi = 0.999 \times \cos\left(-\frac{\pi}{3}\right) = 0.999 \times 0.5 = 0.499500 \approx x$$

$$\rho \sin\varphi = 0.999 \times \sin\left(-\frac{\pi}{3}\right) = 0.999 \times (-0.866) = -0.865159 \approx y$$

Вычислительная ошибка

$$\Delta x = |0.499500 - 0.499471| = 0.000029$$

$$\Delta y = |-0.865159 - (-0.865142)| = 0.000017$$

CORDIC Пример 2. Перевод декартовых координат в полярные (1)

Исходные данные:

$$x = 0.99, y = 0.49$$

Переход в 16-разрядную сетку:

$$x = [0.99 \times 2^{14}] = 16220$$

$$y = [0.49 \times 2^{14}] = 8028$$

Инициализация вычислений:

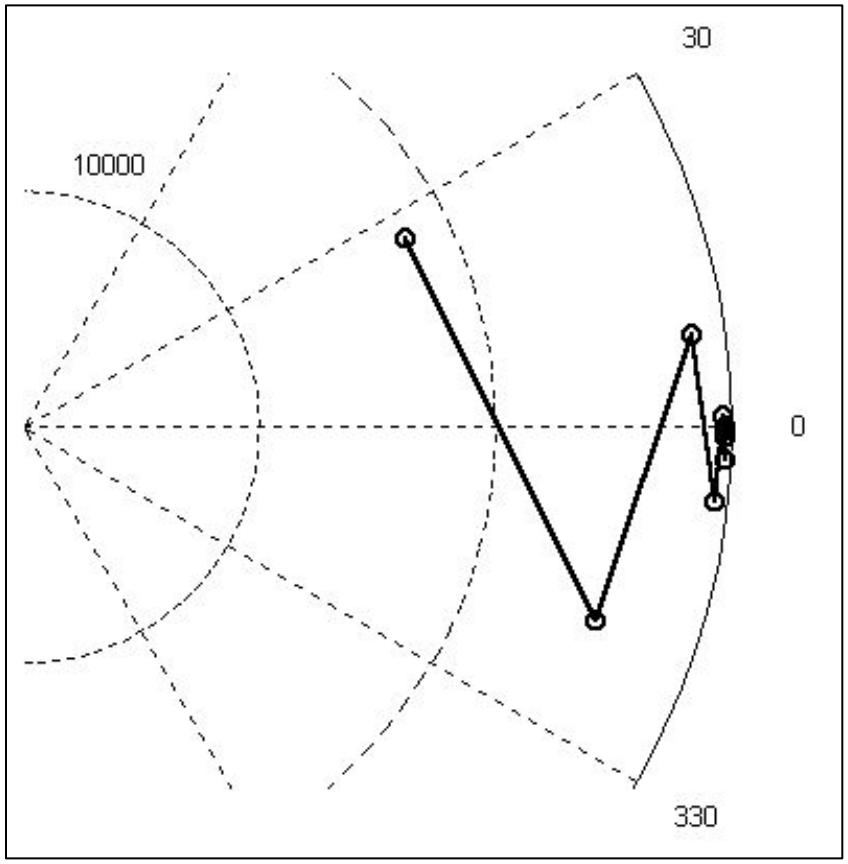
$$x_0 = x, y_0 = y, z_0 = 0$$

Значения переменных на итерациях алгоритма показаны в таблице, управление сумматорами – по знаку переменной ***y*** (инверсная форма).

Таблиц

a n	y_n	x_n	z_n	s_n
0	8028	16220	0	1
1	-8192	24248	8192	-1
2	3932	28344	3356	1
3	-3154	29327	5911	-1
4	511	29722	4614	1
5	-1346	29753	5265	-1
6	-417	29796	4940	-1
7	48	29803	4778	1
8	-184	29803	4859	-1
9	-68	29804	4819	-1
10	-10	29805	4799	-1
11	19	29806	4789	1
12	5	29806	4794	1
13	-2	29806	4796	-1
14	1	29807	4795	1
15	0	**29807**	**4795**	1

CORDIC Пример 2. Перевод декартовых координат в полярные (2)



Графическое представление поворота вектора

Интерпретация

$$\rho = x = 29807 \times K/2^{14} = \frac{29807 \times 0.607253}{16384} = 1.104760$$

$$\varphi = y = 4796 \times \pi/2^{15} = \frac{4795 \times \pi}{32768} = 0.459715$$

Проверк

$$\sqrt{x^2 + y^2} = \sqrt{0.99^2 + 0.49^2} = 1.104627 \approx \rho$$

$$\text{arctg}\,\frac{y}{x} = \text{arctg}\,\frac{0.49}{0.99} = \text{arctg}\,0.4949949 = 0.459599 \approx \varphi$$

Вычислительная ошибка:

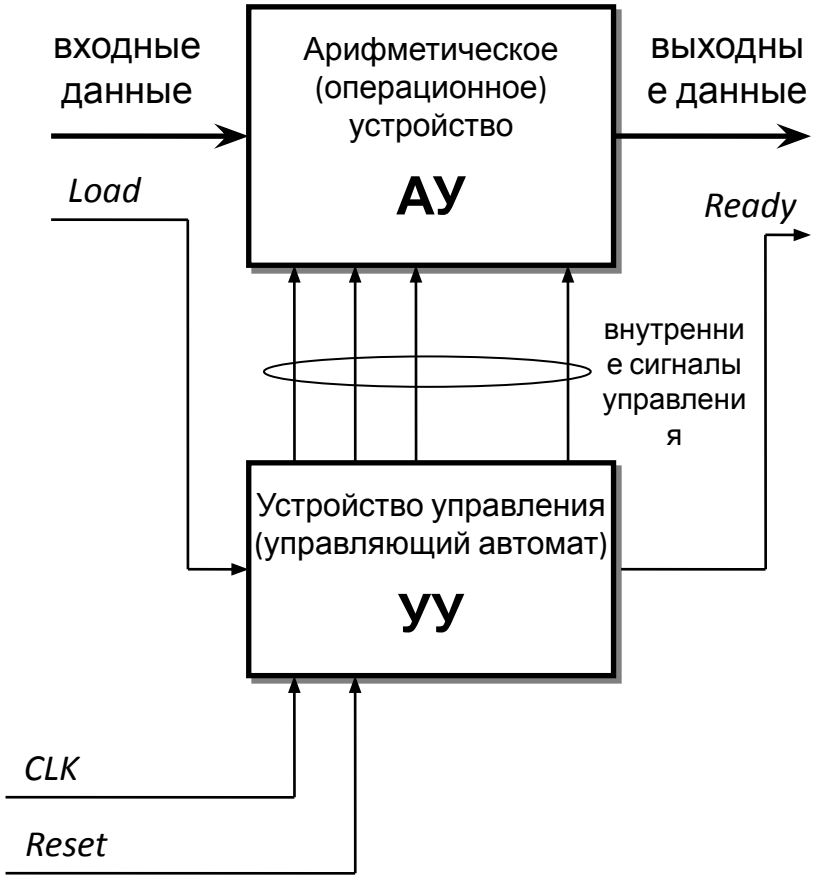
$$\Delta\rho = |1.104627 - 1.104760| = 0.000133$$

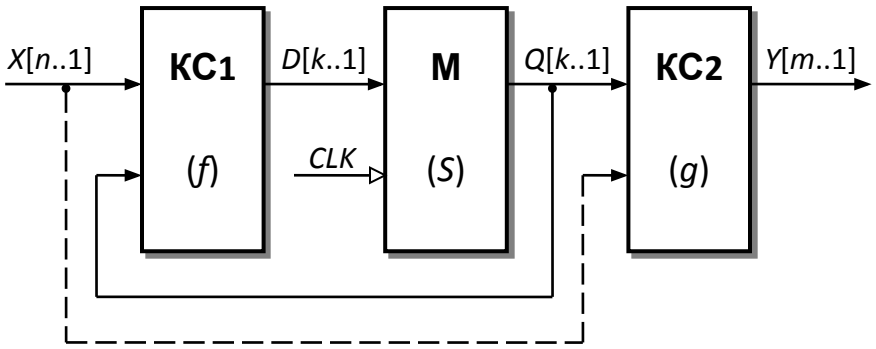
$$\Delta\varphi = |0.459599 - 0.459715| = 0.000116$$

0.000116 радиан = 0.00667 градуса = 0.4 минуты

Синхронизация и управление в вычислителях устройств ЦОС

Управляющие (конечные) автоматы





***M* = {*S*, *X*, *Y*, *f*, *g*},**

где *S* – множество состояний автомата ($S_0 \ldots S_{k-1}$);

X – множество входных векторов (сигналов);

Y – множество выходных векторов (сигналов);

f – функция переходов;

g – функция выходов.

Q – текущее состояние автомата (*Q* принадлежит *S*);

D – следующее состояние автомата (*D* принадлежит *S*);

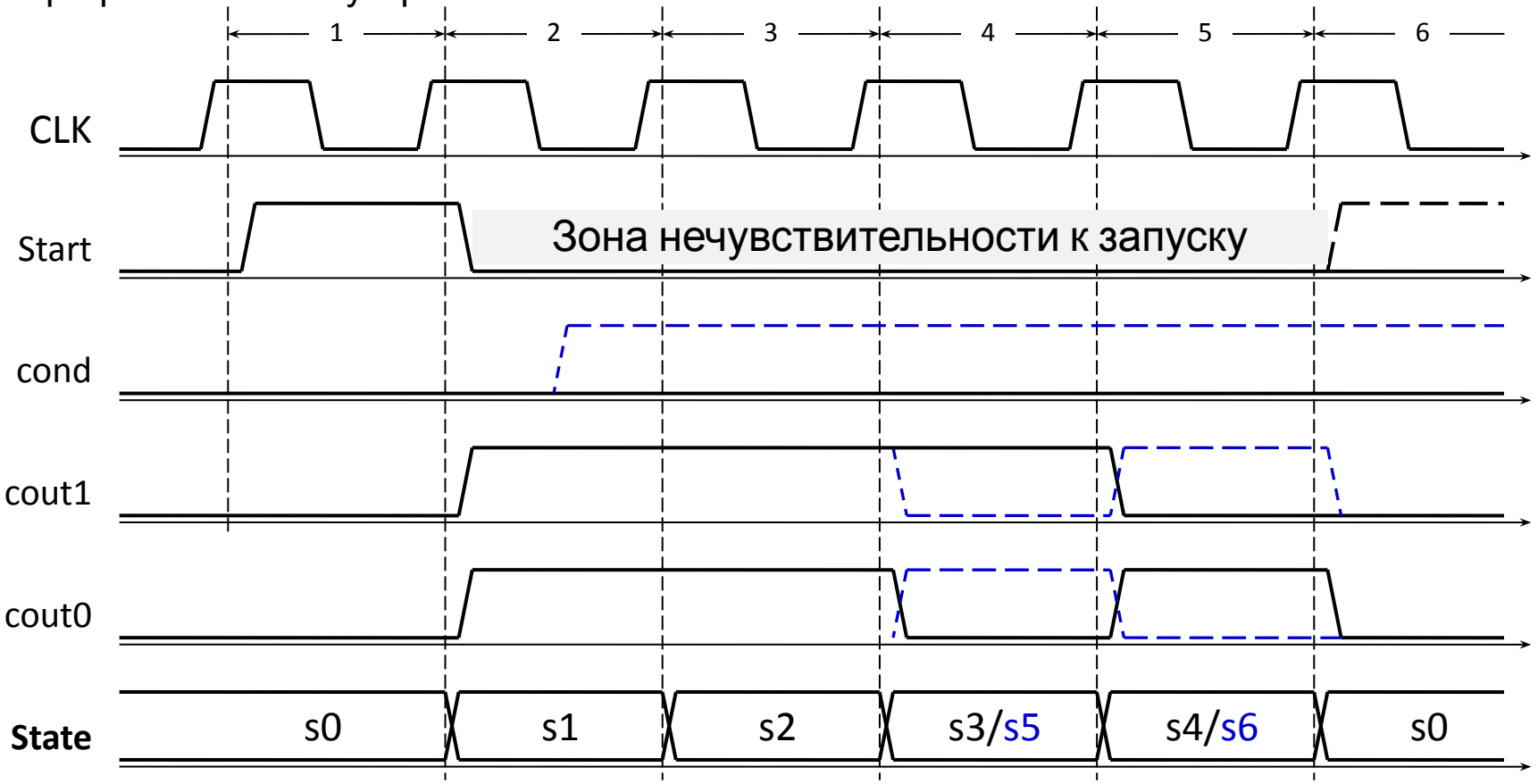
D = f(X,Q);

Y = g(Q) для автомата Мура и

Y = g(X,Q) для автомата Мили

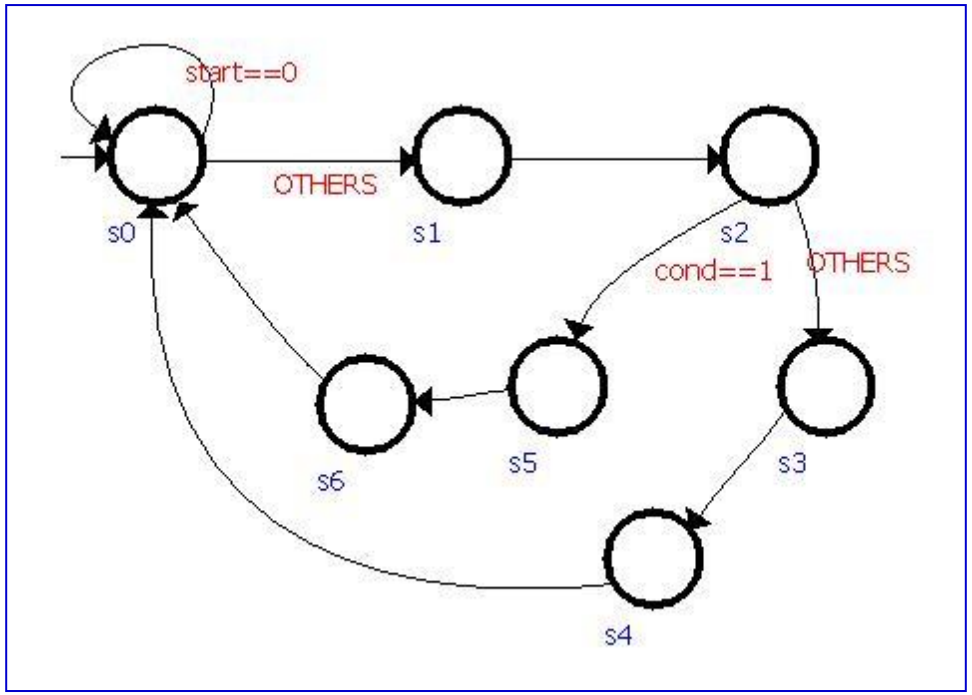
Управляющий автомат – исходные данные

Исходными данными для проектирования управляющего автомата является временная диаграмма сигналов, необходимая для правильной работы арифметического устройства



Управляющий автомат – граф и функция выходов

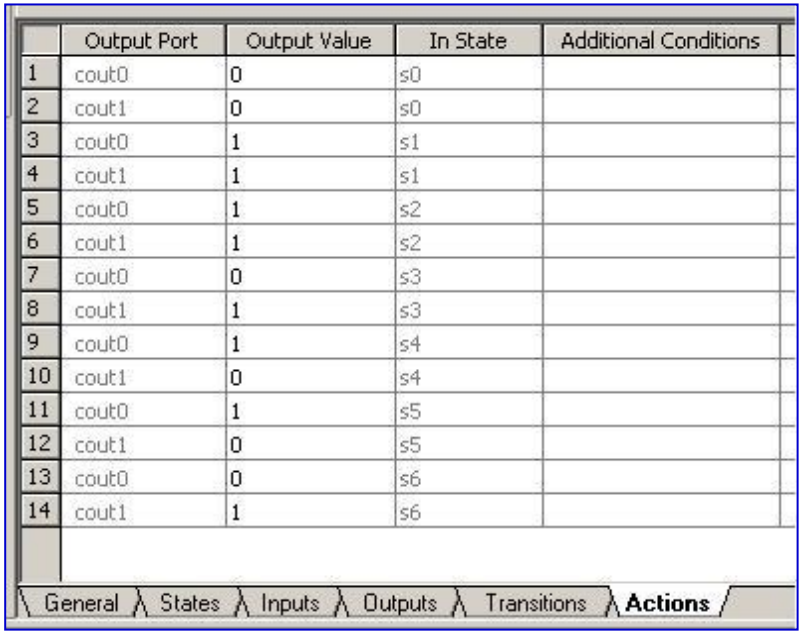
Граф управляющего



Вершины графа – состояния управляющего автомата, пути – переходы между состояниями (условные или безусловные).

Граф управляющего автомата строится с помощью ***State Tool*** и ***Transition Tool*** соответствующего редактора ***Quartus II***.

Определение функции выходов



	Output Port	Output Value	In State	Additional Conditions
1	cout0	0	s0	
2	cout1	0	s0	
3	cout0	1	s1	
4	cout1	1	s1	
5	cout0	1	s2	
6	cout1	1	s2	
7	cout0	0	s3	
8	cout1	1	s3	
9	cout0	1	s4	
10	cout1	0	s4	
11	cout0	1	s5	
12	cout1	0	s5	
13	cout0	0	s6	
14	cout1	1	s6	

General | States | Inputs | Outputs | Transitions | **Actions**

Для каждого выхода **cout0** и **cout1** задаются их значения в каждом из состояний **s0** – **s6** управляющего автомата

(функция ***State Machine Table*** редактора ***Quartus II***)

Управляющий автомат – VHDL код

1

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY SM2 IS
   PORT (
      reset : IN STD_LOGIC := '0';
      clock : IN STD_LOGIC;
      start : IN STD_LOGIC := '0';
      cond : IN STD_LOGIC := '0';
      cout0 : OUT STD_LOGIC;
      cout1 : OUT STD_LOGIC
   );
END SM2;

ARCHITECTURE BEHAVIOR OF SM2 IS
   TYPE type_fstate IS (s0,s1,s2,s3,s4,s5,s6);
   SIGNAL fstate : type_fstate;
   SIGNAL reg_fstate : type_fstate;

BEGIN
   PROCESS (clock,reset,reg_fstate)
   BEGIN
      IF (reset='1') THEN
         fstate <= s0;
      ELSIF (clock='1' AND clock'event) THEN
         fstate <= reg_fstate;
      END IF;
```

2

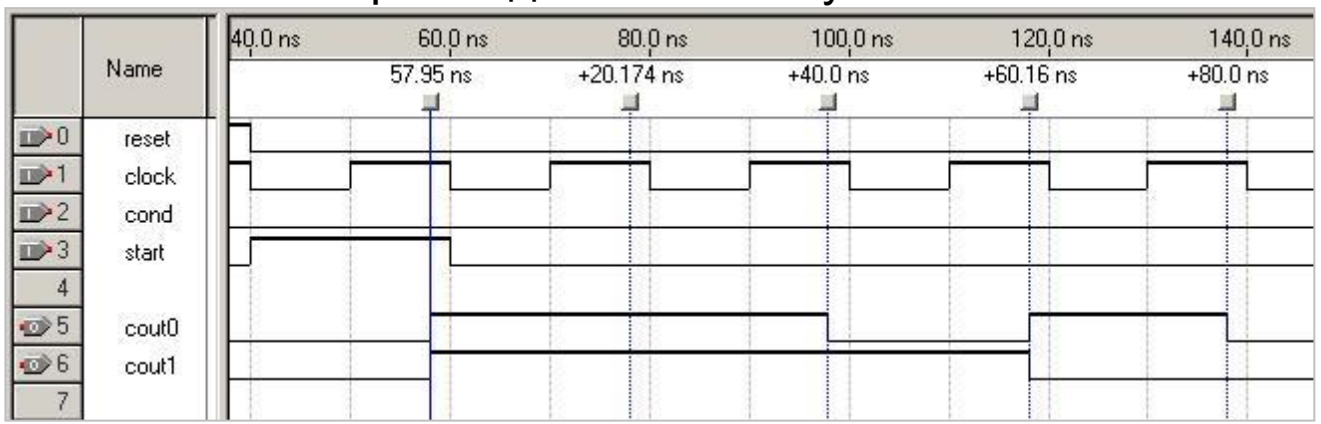
```
   PROCESS (fstate,start,cond)
   BEGIN
      CASE fstate IS

         WHEN s0 =>
            IF ((start = '0')) THEN
               reg_fstate <= s0;
            ELSE
               reg_fstate <= s1;
            END IF;
            cout1 <= '0';
            cout0 <= '0';

         WHEN s1 =>
            reg_fstate <= s2;
            cout1 <= '1';
            cout0 <= '1';

         WHEN s2 =>
            IF ((cond = '1')) THEN
               reg_fstate <= s5;
            ELSE
               reg_fstate <= s3;
            END IF;
            cout1 <= '1';
            cout0 <= '1';
```

3

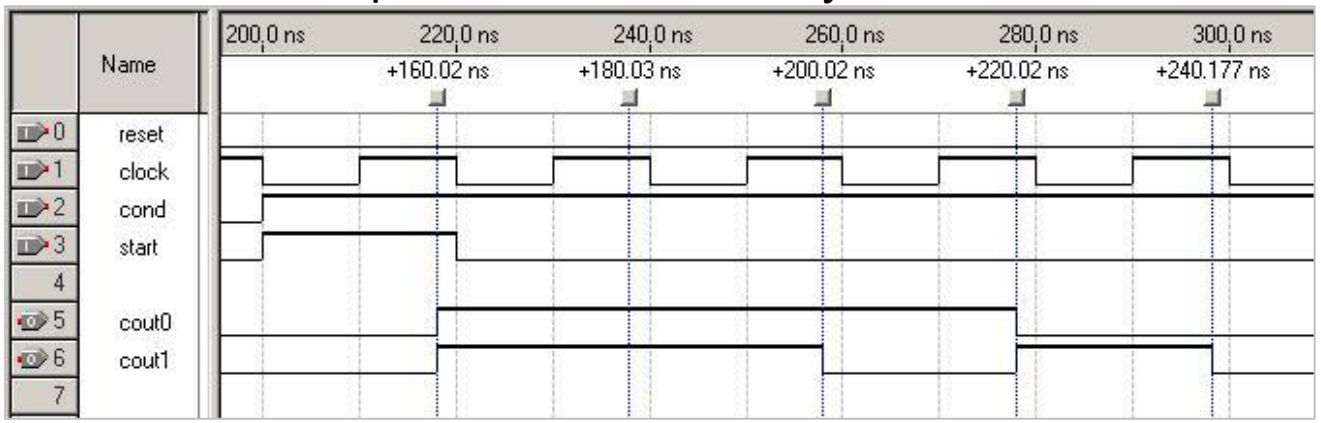
```
         WHEN s3 =>
            reg_fstate <= s4;
            cout1 <= '1';
            cout0 <= '0';

         WHEN s4 =>
            reg_fstate <= s0;
            cout1 <= '0';
            cout0 <= '1';

         WHEN s5 =>
            reg_fstate <= s6;
            cout1 <= '0';
            cout0 <= '1';

         WHEN s6 =>
            reg_fstate <= s0;
            cout1 <= '1';
            cout0 <= '0';

         WHEN OTHERS =>
            cout0 <= 'X';
            cout1 <= 'X';
            report "Reach undefined state";
      END CASE;
   END PROCESS;
END BEHAVIOR;
```

Управляющий автомат – результат симуляции

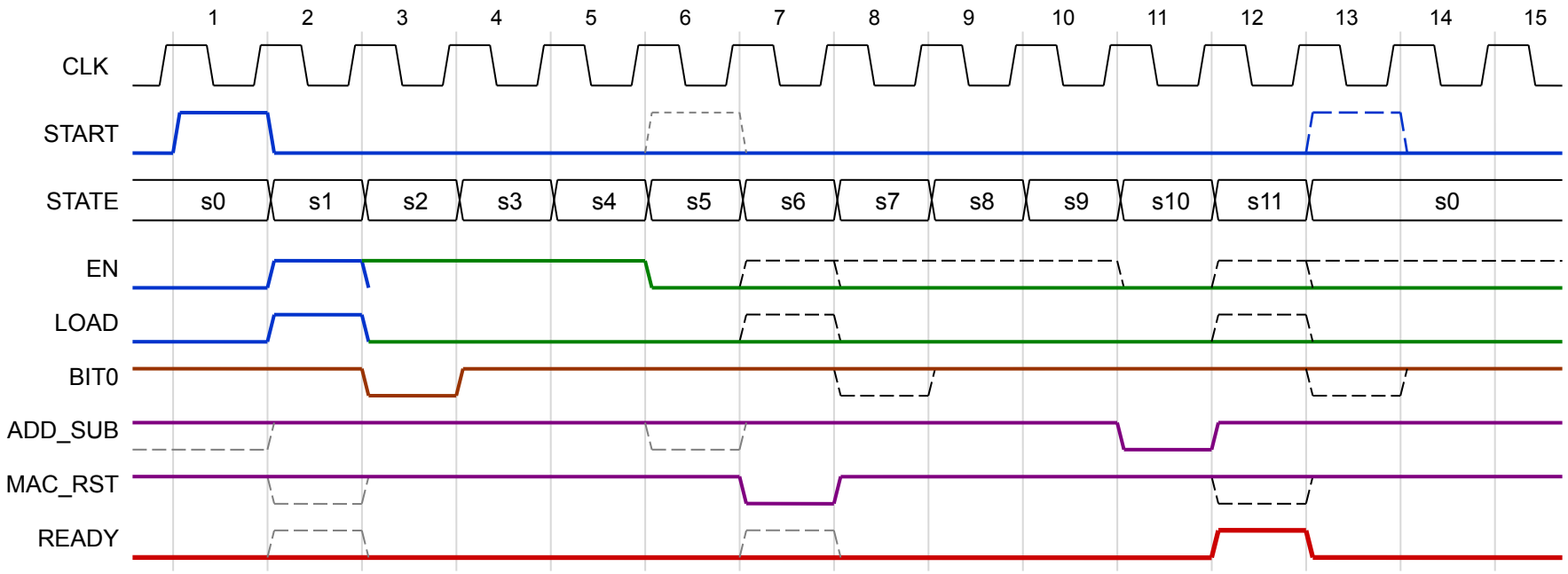
Моделирование работы управляющего автомата в симуляторе *Quartus II* при входном сигнале условия ***cond = 0***

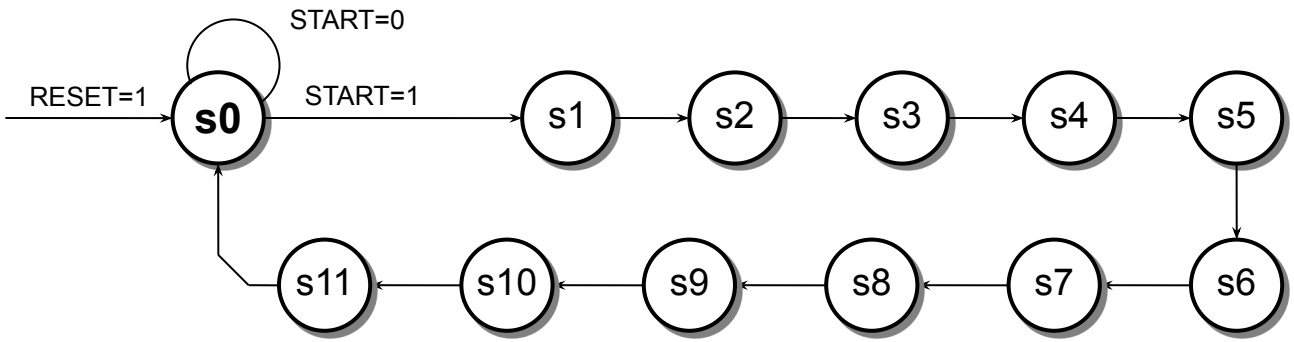


Моделирование работы управляющего автомата в симуляторе *Quartus II* при входном сигнале условия ***cond = 1***

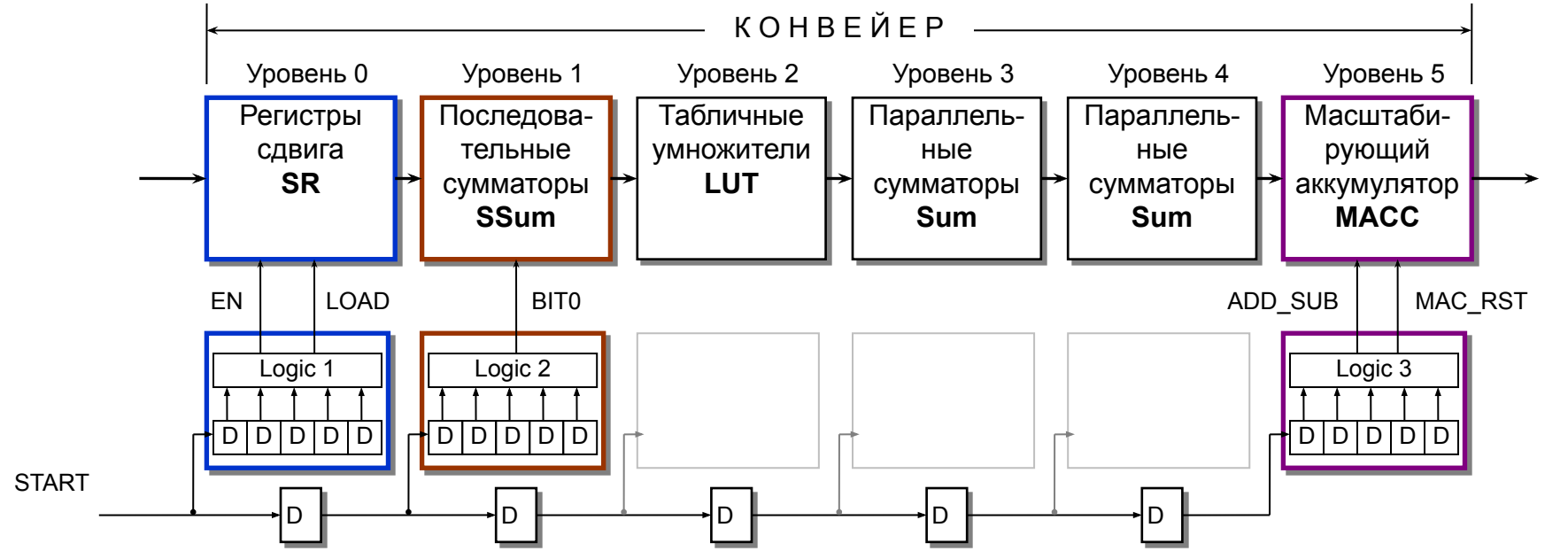


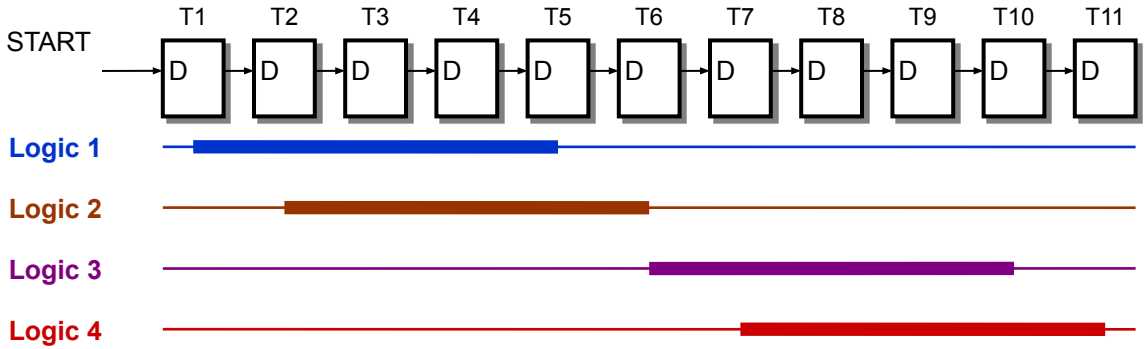
Пример управляющего автомата (УА) ЦФ с РА (вариант 1)



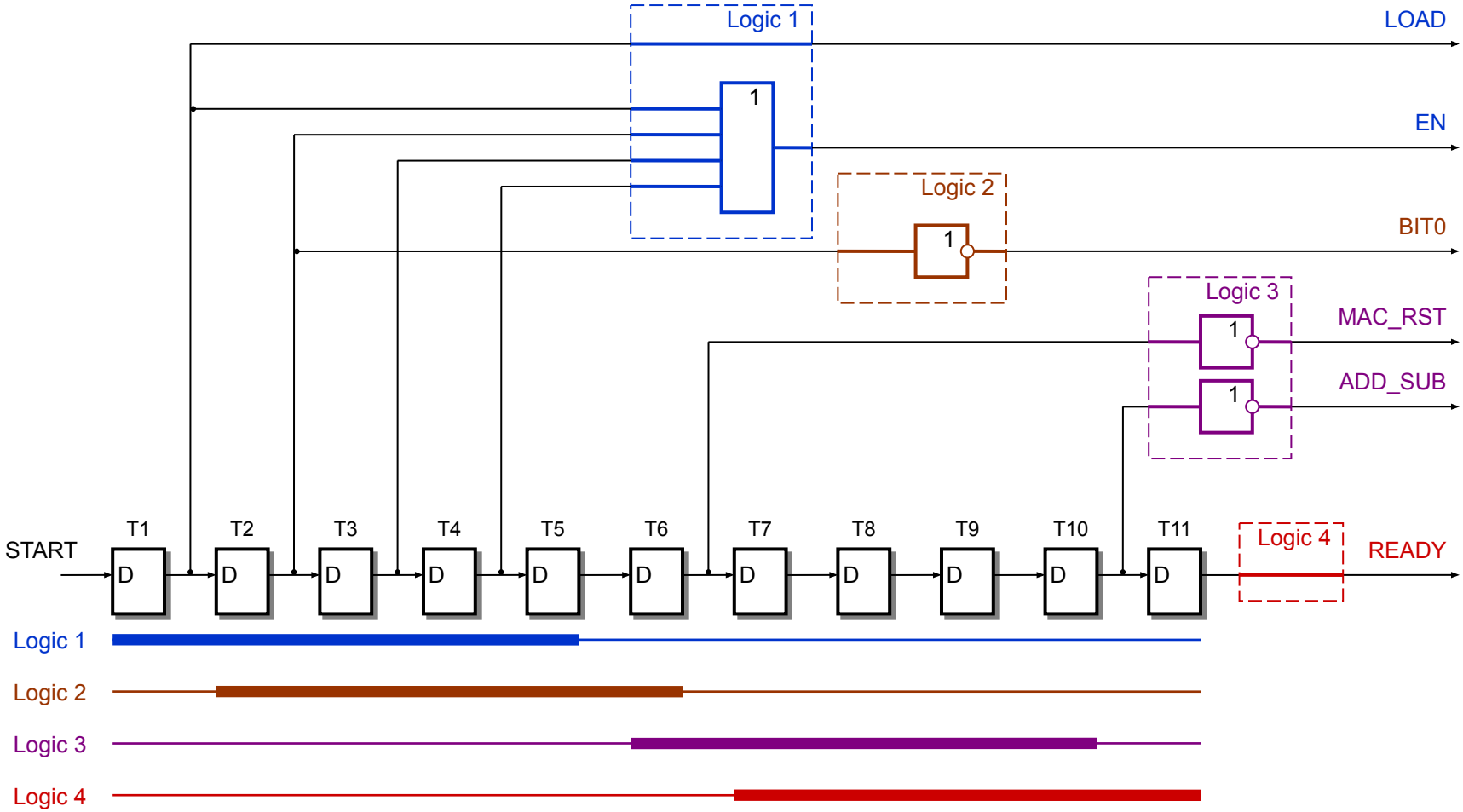


Пример УА ЦФ с РА (вариант 2)

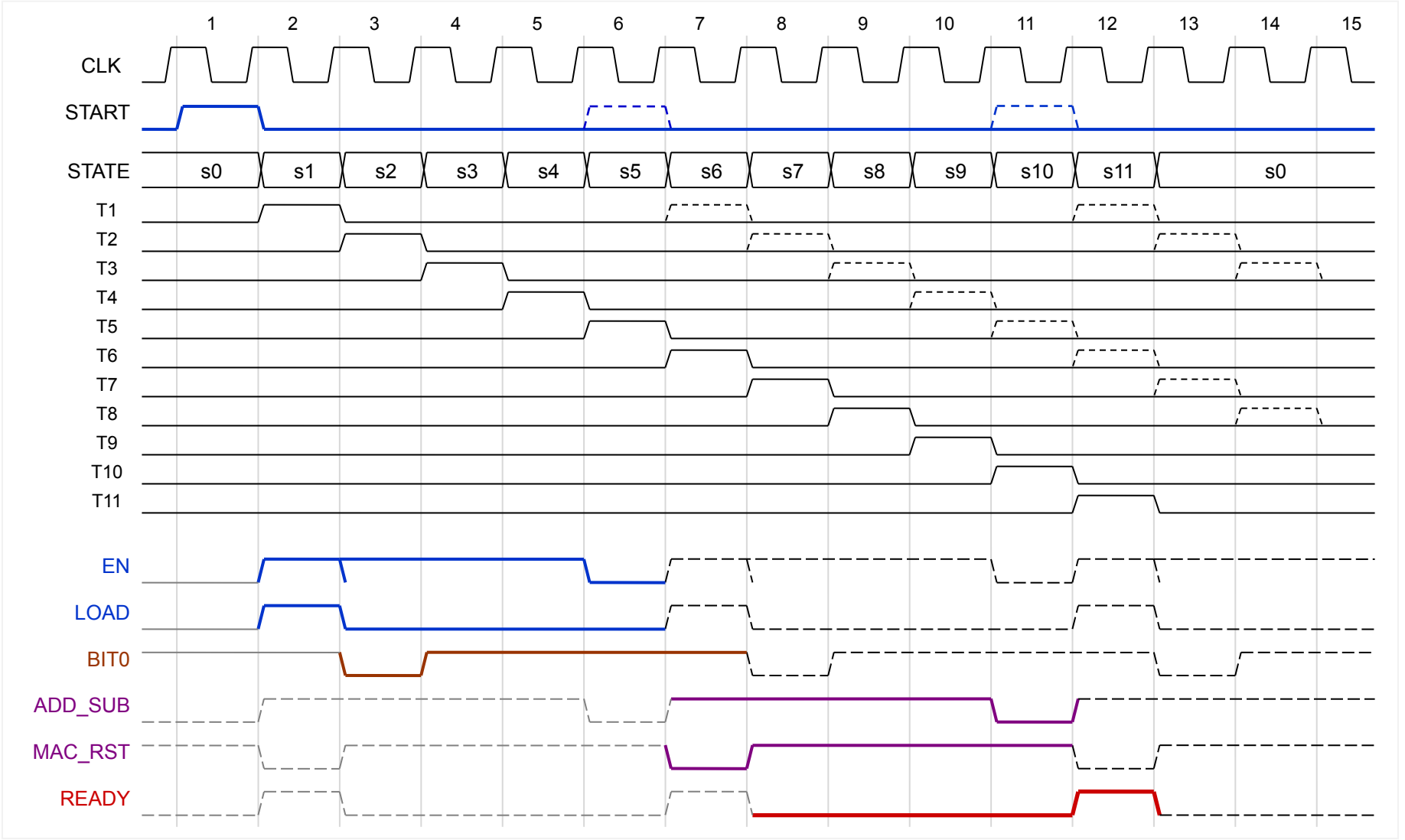




Пример УА ЦФ с РА на регистре сдвига

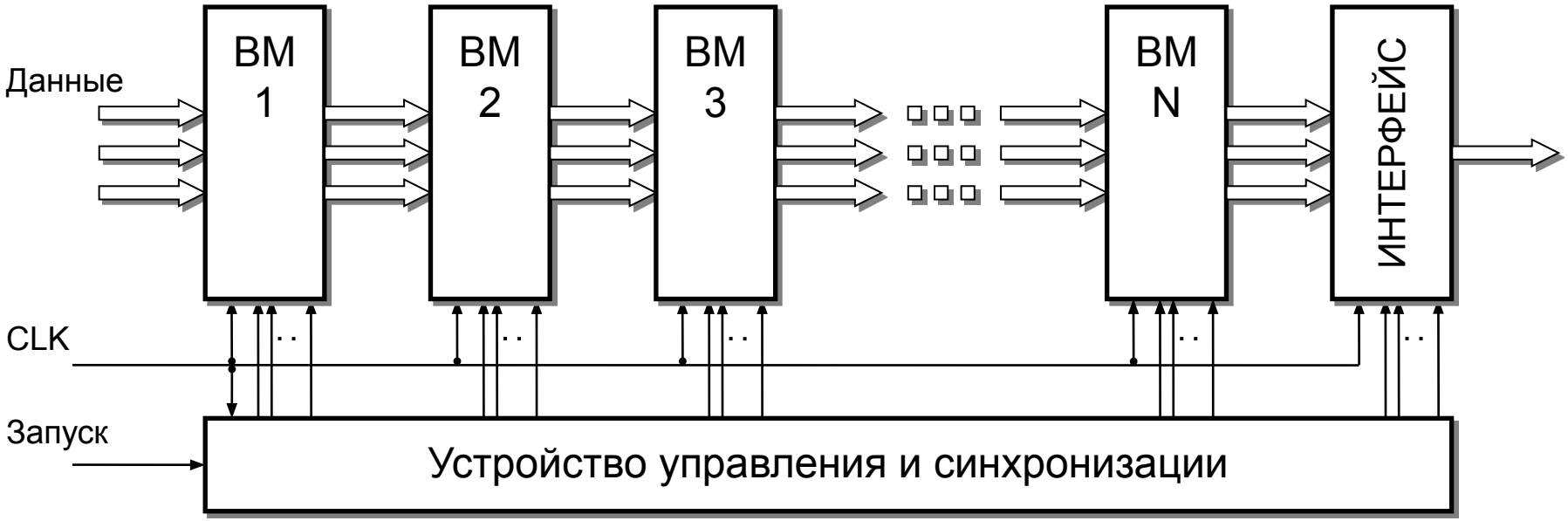


Временные диаграммы УА ЦФ с РА на регистре сдвига



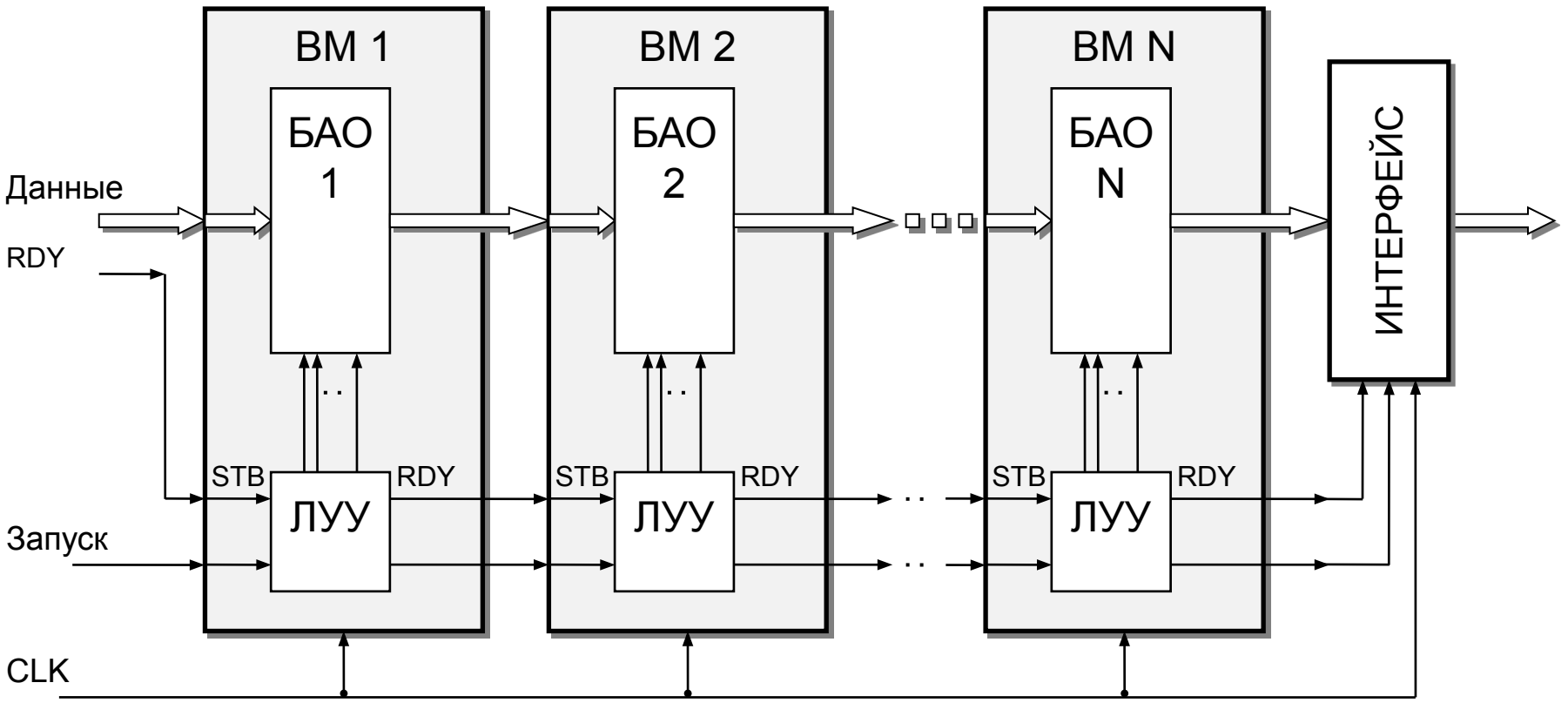
Глобальное управление конвейером обработки

Структура системы ЦОС – вычислительный конвейер



ВМ – вычислительный модуль

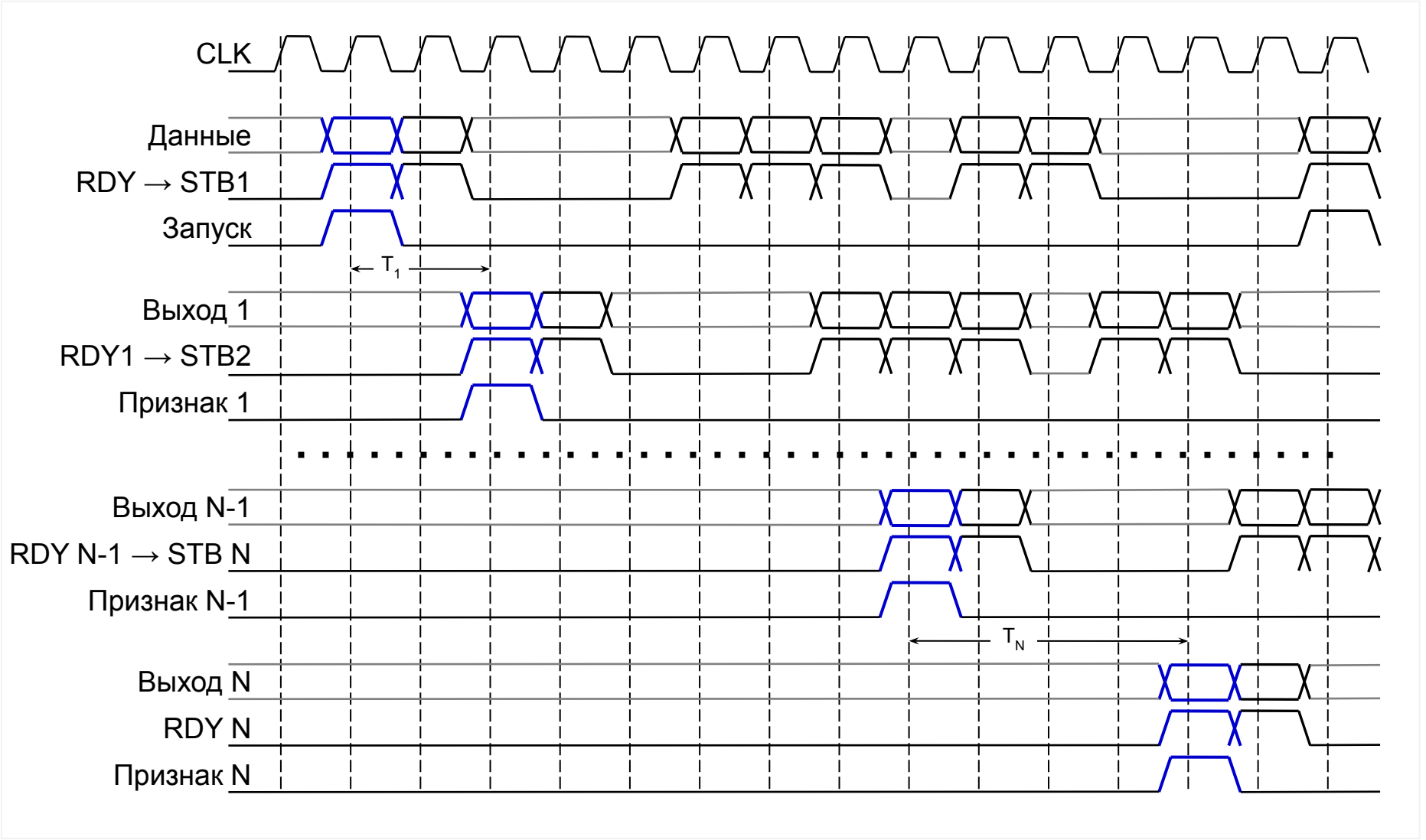
Распределенное (локальное) управление



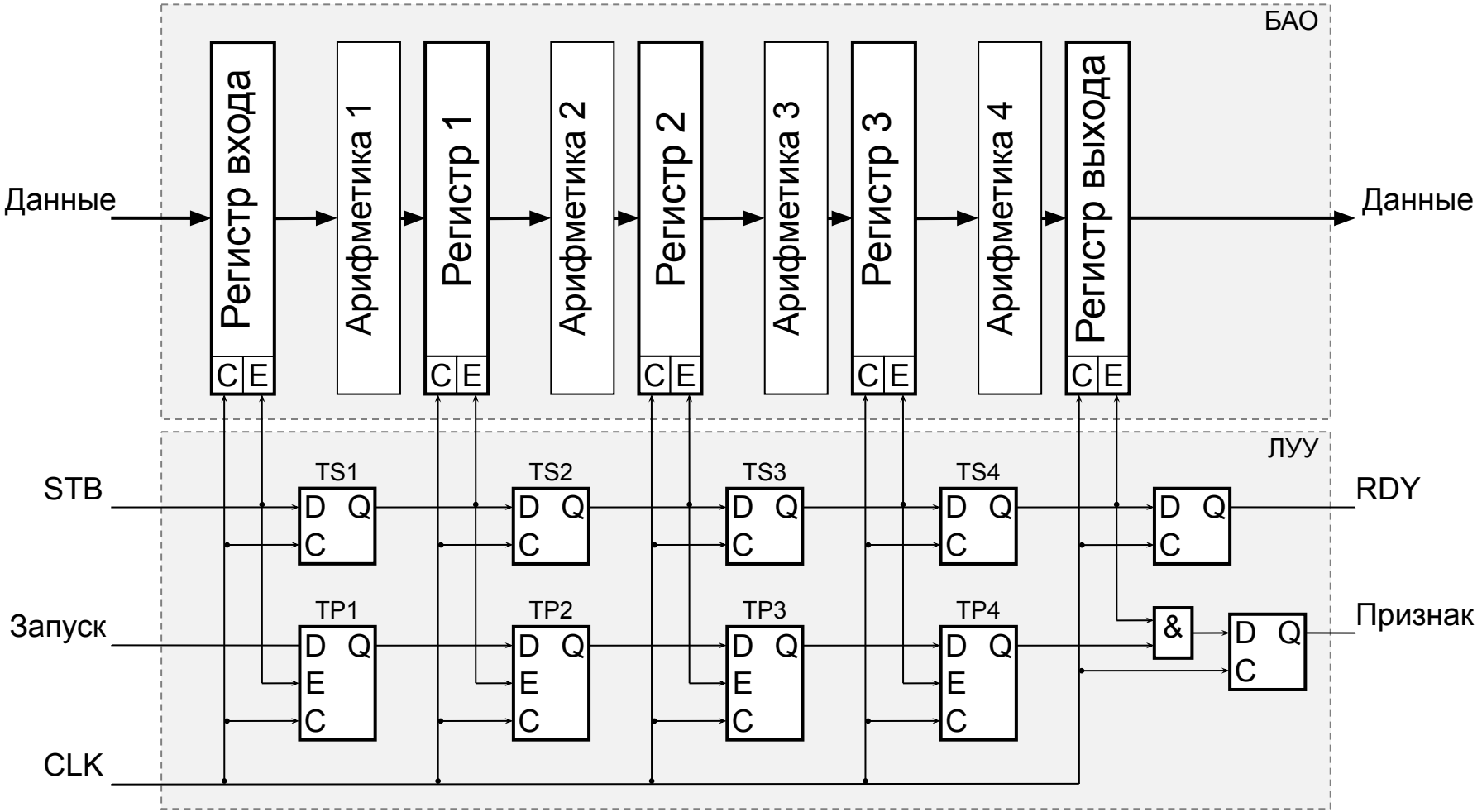
ВМ – вычислительный модуль
БАО – блок арифметических операций
ЛУУ – локальное устройство управления

STB – строб данных (признак готовности входных данных)
RDY – готовность данных (признак готовности выходных данных)

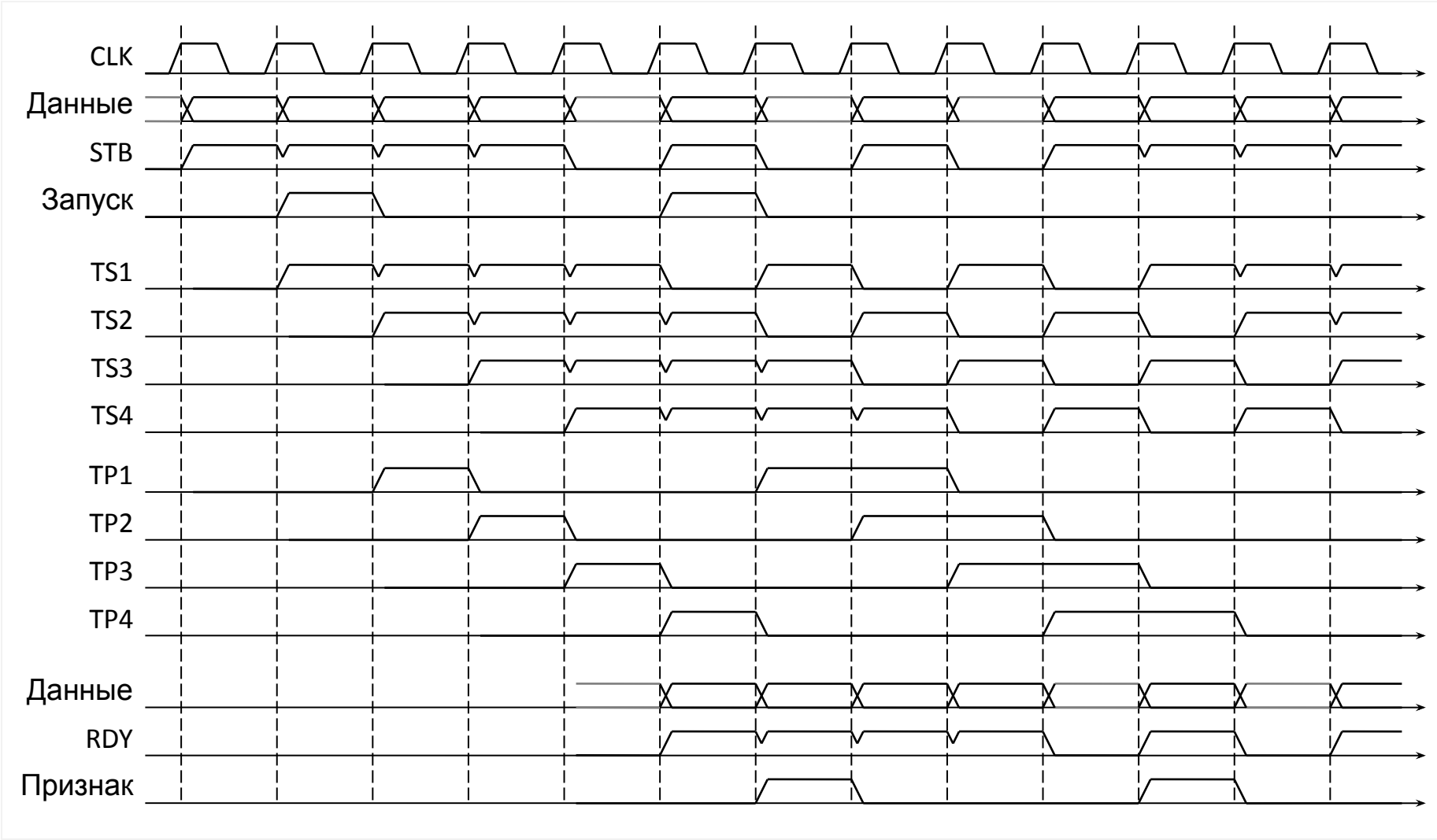
Диаграммы распределенного управления



Локальное управление вычислительного модуля



Диаграммы локального управления ВМ



Вычислители ЦОС с плавающей точкой

Заключение