Глава 5

Методические материалы

5.1. ЛАБОРАТОРНЫЕ РАБОТЫ

ЛАБОРАТОРНАЯ РАБОТА 1
Описание и моделирование системы логических функций

Задание

По таблице истинности системы логических функций составить VHDL-модель и выполнить моделирование на всех наборах значений входных переменных.

VHDL-модель системы функций должна быть **компактной**: функции желательно минимизировать, а также учитывать возможность их инверсной реализации.

Рекомендуемый порядок выполнения работы

1. Определить для каждой функции f системы форму ее реализации — прямую f или инверсную $\overline{f}$.

2. Минимизировать представление функции, применив любой известный метод минимизации, например, с помощью карт Карно, метода Квайна, диаграмм двоичного выбора и т. д.

3. Составить VHDL-модель, употребив логические операторы и операторы назначения сигналов.

4. Составить тестирующую программу, порядок подачи тестирующие воздействий должен соответствовать порядку наборов из левой части таблицы истинности.

Требования к оформлению отчета

1. В отчете должно быть приведена **таблица истинности** и соответствующая ей **VHDL-модель.**

2. В отчете должна содержаться **тестирующая программа** для всех наборов входных переменных, соответствующих таблице истинности.

3. В отчете должны содержаться **временные диаграммы,** соответствующие тестирующей программе.

Вариант 1

x1	x2	x3	x4	y1	y2	y3
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	0	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	0	0

Вариант 2

x1	x2	x3	x4	y1	y2	y3
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	1	0	1
0	1	0	0	0	0	0
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	1	0	0
1	0	0	0	1	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

Вариант 3

x1	x2	x3	x4	y1	y2	y3
0	0	0	0	1	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	1	1	1
0	1	0	0	0	0	0
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	0	0
1	0	0	1	0	1	1
1	0	1	0	0	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1
1	1	1	1	1	1	0

Вариант 4

x1	x2	x3	x4	y1	y2	y3
0	0	0	0	1	0	0
0	0	0	1	1	0	1
0	0	1	0	1	0	0
0	0	1	1	0	0	0
0	1	0	0	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
0	1	1	1	0	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	0	1	1	1	0	1
1	1	0	0	0	0	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	0	0	0

Вариант 5

x1	x2	x3	x4	y1	y2	y3
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	1	0
0	0	1	1	1	1	1
0	1	0	0	0	0	0
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	1	1	0
1	0	0	1	1	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	1	1	1
1	1	0	1	0	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Вариант 6

x1	x2	x3	x4	y1	y2	y3
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	0	0	0
1	0	1	1	0	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	0	1	0

Вариант 7

x1	x2	x3	x4	y1	y2	y3
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	0	1	1	0	0	1
0	1	0	0	0	0	0
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	0	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1
1	1	1	1	1	0	0

Вариант 8

x1	x2	x3	x4	y1	y2	y3
0	0	0	0	0	0	1
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	1	1
0	1	1	0	0	1	1
0	1	1	1	1	0	1
1	0	0	0	0	1	1
1	0	0	1	0	1	1
1	0	1	0	1	0	1
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

Вариант 9

x1	x2	x3	x4	y1	y2	y3
0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	0	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	0	0	1
1	1	1	1	1	1	0

Вариант 10

x1	x2	x3	x4	y1	y2	y3
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	1	1	1
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	1	0	0
0	1	1	0	1	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	1

Вариант 11

x1	x2	x3	x4	y1	y2	y3
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	1

Вариант 12

x1	x2	x3	x4	y1	y2	y3
0	0	0	0	1	0	0
0	0	0	1	1	1	1
0	0	1	0	1	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	1
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	1	0
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

Вариант 13

x1	x2	x3	x4	y1	y2	y3
0	0	0	0	0	1	0
0	0	0	1	0	1	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	1	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	0	1	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	0	1	0
1	0	1	1	0	1	1
1	1	0	0	0	1	1
1	1	0	1	0	1	0
1	1	1	0	0	1	1
1	1	1	1	0	1	0

Вариант 14

x1	x2	x3	x4	y1	y2	y3
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	0	0	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1
1	1	1	1	1	1	0

Вариант 15

x1	x2	x3	x4	y1	y2	y3
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	0	1
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	1	0
1	0	0	0	1	0	0
1	0	0	1	1	0	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	1	1	1
1	1	0	1	1	1	0
1	1	1	0	1	1	1
1	1	1	1	1	1	1

ЛАБОРАТОРНАЯ РАБОТА 2
Описание и моделирование нерегулярных логических схем

Задание

Для заданной нерегулярной логической схемы:

- составить структурное VHDL-описание;
- выполнить моделирование на всех наборах значений входных переменных;
- построить таблицу истинности системы логических функций, реализуемых схемой;
- найти критический путь в схеме.

Рекомендуемый порядок выполнения работы

1. Составить VHDL-модель каждого из типов элементов, входящих в схему. Если в схеме есть элементы одинакового типа, то составляется *одна модель* для всех элементов данного типа. Модель элемента должна соответствовать задержке, указанной в табл. 5.1. При графическом изображении логического элемента на схеме будет указываться его тип (библиотечное имя) и имена входных и выходных полюсов.

2. Составить VHDL-модель схемы в целом.

3. Составить тестирующую программу для всех наборов значений входных переменных.

4. Провести моделирование и получить временную диаграмму.

5. По временной диаграмме записать таблицу истинности системы логических функций, реализуемых схемой.

6. Для каждого тестирующего набора определить задержку схемы.

7. Найти критический путь на схеме — путь с наибольшей суммарной задержкой элементов.

Требования к оформлению отчета

1. В отчете должна быть нарисована логическая **схема**. При этом обозначения сигналов, элементов схемы должны *соответствовать* описанию на языке VHDL.

2. В отчете должен содержаться **VHDL-код** схемы и **тестирующая программа.**

3. VHDL-код и тест должны быть в *отдельных* файлах и содержать *комментарии:* автор разработанной VHDL-модели, номер варианта;

4. В отчете должны содержаться **временные диаграммы,** соответствующие тестирующей программе.

5. В отчете должна содержаться **таблица истинности** системы логических функций, реализуемых схемой.

6. На логической схеме должен быть отмечен **критический путь.**

7. В отчете должно быть указано значение задержки схемы, соответствующее задержке критического пути.

Таблица 5.1

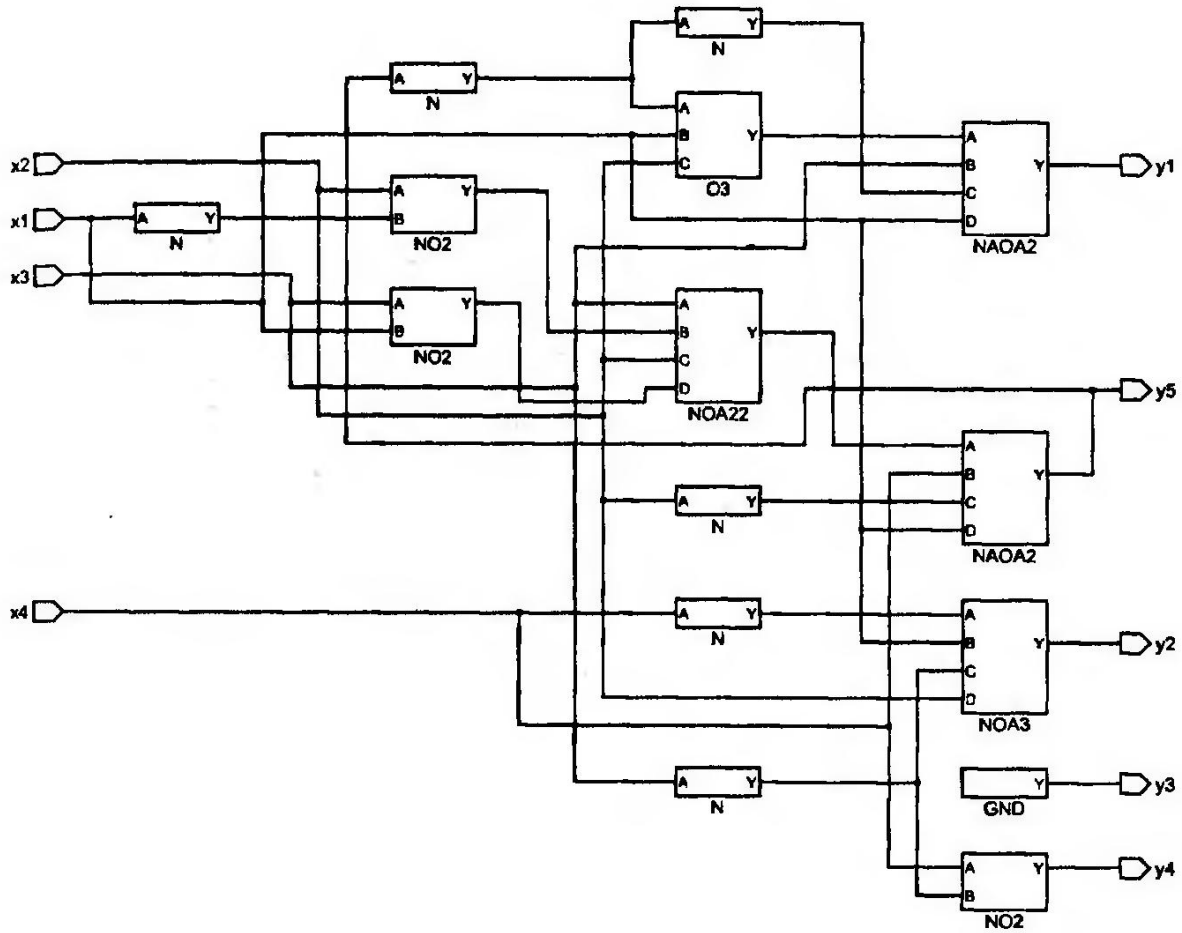
Логические элементы

Имя элемента	Функция элемента	Задержка (ns)
GND	$y = 0$	1
VCC	$y = 1$	1
N	$y = \overline{A}$	1
A2	$y = AB$	2
A3	$y = ABC$	3
A4	$y = ABCD$	4
A6	$y = ABCDEF$	6
A8	$y = ABCDEFGH$	8
EX2	$y = A\overline{B} \vee \overline{A}B$	5
MX2	$y = A\overline{V} \vee BV$	3
NA2	$y = \overline{AB}$	2

Окончание таблицы 5.1

NA3	$y = \overline{ABC}$	3
NA3O2	$y = \overline{AB(C \vee D)}$	4
NA4	$y = \overline{ABCD}$	5
NAO2	$y = \overline{A(B \vee C)}$	3
NAO22	$y = \overline{(A \vee B)(C \vee D)}$	3
NAO3	$y = \overline{A(B \vee C \vee D)}$	5
NAOA2	$y = \overline{A(B \vee (CD)}$	4
NEX2	$y = AB \vee \overline{AB}$	5
NMX2	$y = \overline{(A \vee \overline{V})(B \vee V)}$	6
NMX4	$y = \overline{AV_1V_2} \vee \overline{BV_1}V_2 \vee \overline{C}V_1\overline{V_2} \vee \overline{D}V_1V_2$	8
NO2	$y = \overline{A \vee B}$	3
NO3	$y = \overline{A \vee B \vee C}$	4
NO3A2	$y = \overline{A \vee B \vee DC}$	5
NO4	$y = \overline{A \vee B \vee C \vee D}$	5
NOA2	$y = \overline{A \vee BC}$	3
NOA22	$y = \overline{AB \vee CD}$	4
NOA3	$y = \overline{A \vee BCD}$	5
NOAO2	$y = \overline{A \vee B(C \vee D)}$	4
O2	$y = A \vee B$	2
O3	$y = A \vee B \vee C$	3
O4	$y = A \vee B \vee C \vee D$	4
O6	$y = A \vee B \vee C \vee D \vee E \vee F$	6
O8	$y = A \vee B \vee C \vee D \vee E \vee F \vee G \vee H$	8

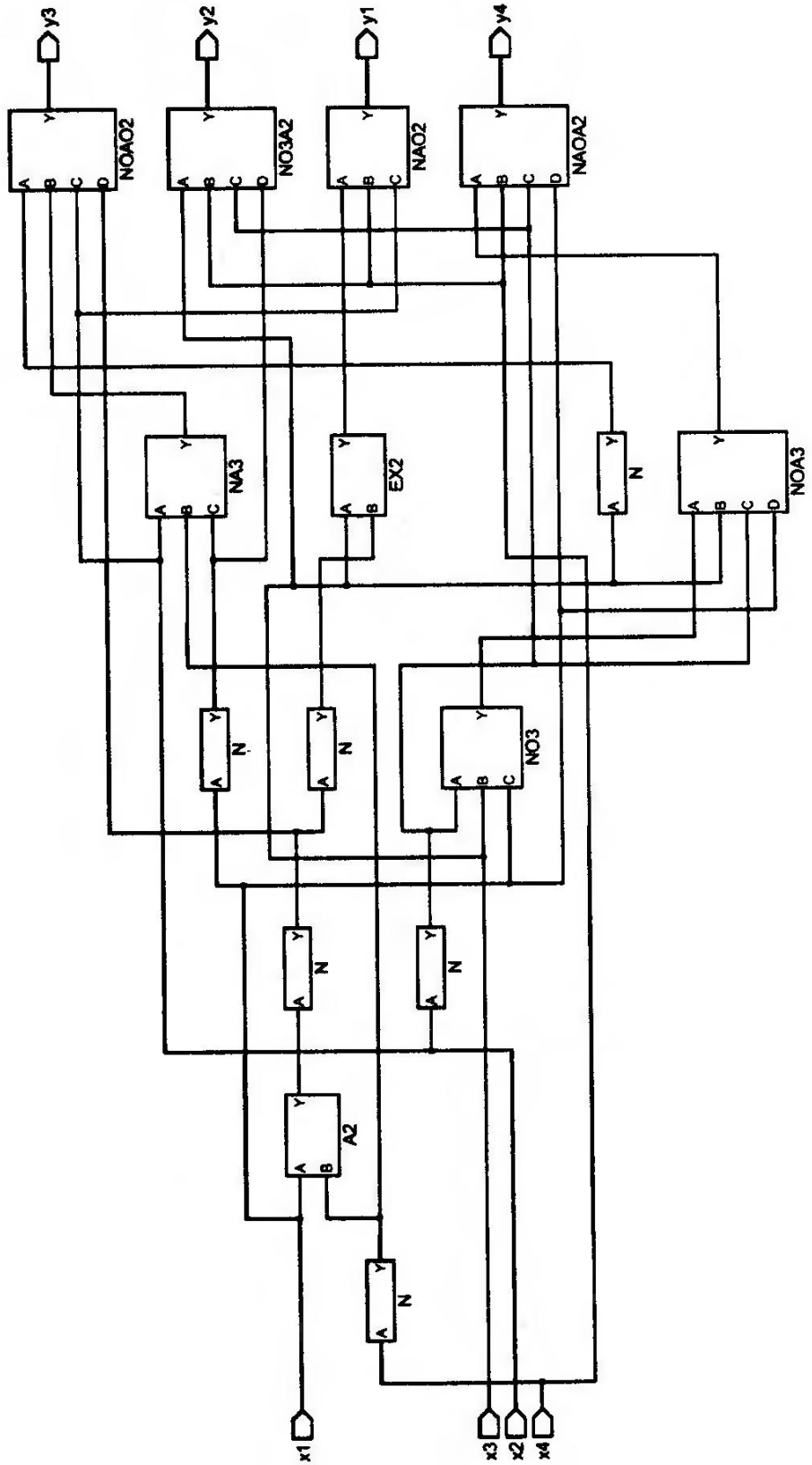
Вариант 1


x2
x1
x3
x4
N
N
N
N
N
N
NO2
NO2
O3
NOA22
NAOA2
NAOA2
NOA3
GND
NO2
y1
y5
y2
y3
y4

Ответ к варианту 1

x1	x2	x3	x4	y1	y2	y3	y4	y5
0	0	0	0	1	0	0	0	1
0	0	0	1	1	1	0	0	0
0	0	1	0	1	0	0	1	1
0	0	1	1	0	1	0	0	0
0	1	0	0	1	0	0	0	1
0	1	0	1	1	1	0	0	1
0	1	1	0	0	0	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	0	0	0	0
1	0	0	1	1	1	0	0	0
1	0	1	0	0	0	0	1	1
1	0	1	1	0	1	0	0	1
1	1	0	0	0	0	0	0	1
1	1	0	1	1	0	0	0	0
1	1	1	0	0	0	0	1	1
1	1	1	1	0	1	0	0	0

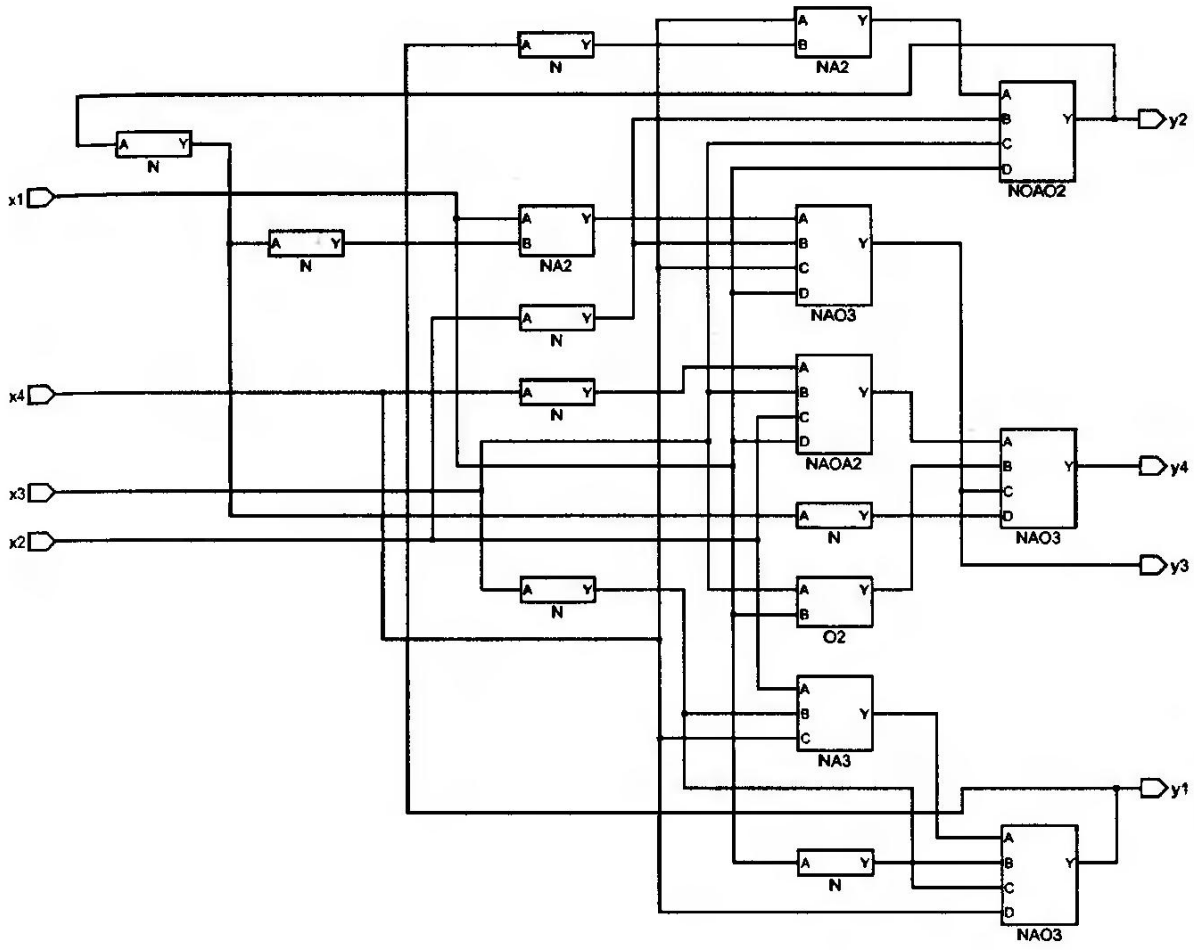
Вариант 2


y3
y2
y1
y4
NOAO2
NO3A2
NAO2
NAOA2
NA3
EX2
N
NOA3
N
N
NO3
N
N
A2
N
x1
x3
x2
x4

Ответ к варианту 2

x1	x2	x3	x4	y1	y2	y3	y4
0	0	0	0	1	0	0	1
0	0	0	1	1	0	0	0
0	0	1	0	1	0	0	1
0	0	1	1	0	0	0	0
0	1	0	0	1	1	0	1
0	1	0	1	1	0	0	1
0	1	1	0	0	0	1	1
0	1	1	1	0	0	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	1	1
1	0	1	1	0	0	0	1
1	1	0	0	0	1	0	1
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	1
1	1	1	1	0	0	0	0

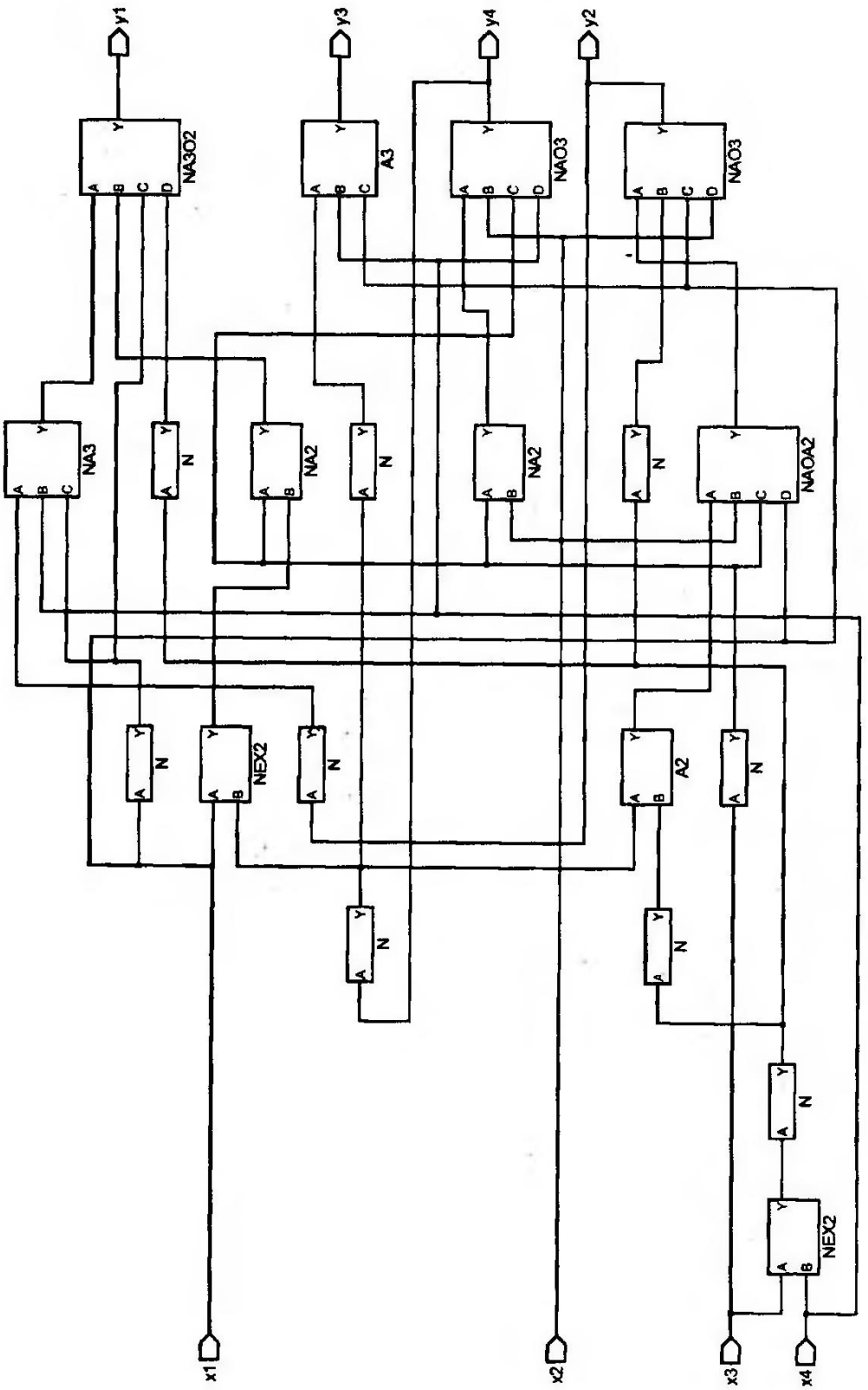
Вариант 3



Ответ к варианту 3

x1	x2	x3	x4	y1	y2	y3	y4
0	0	0	0	0	0	0	1
0	0	0	1	0	1	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	0	0
0	1	0	0	0	0	1	0
0	1	0	1	1	0	0	1
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
1	0	1	0	1	0	0	1
1	0	1	1	0	0	0	0
1	1	0	0	0	0	0	1
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	1
1	1	1	1	0	1	1	0

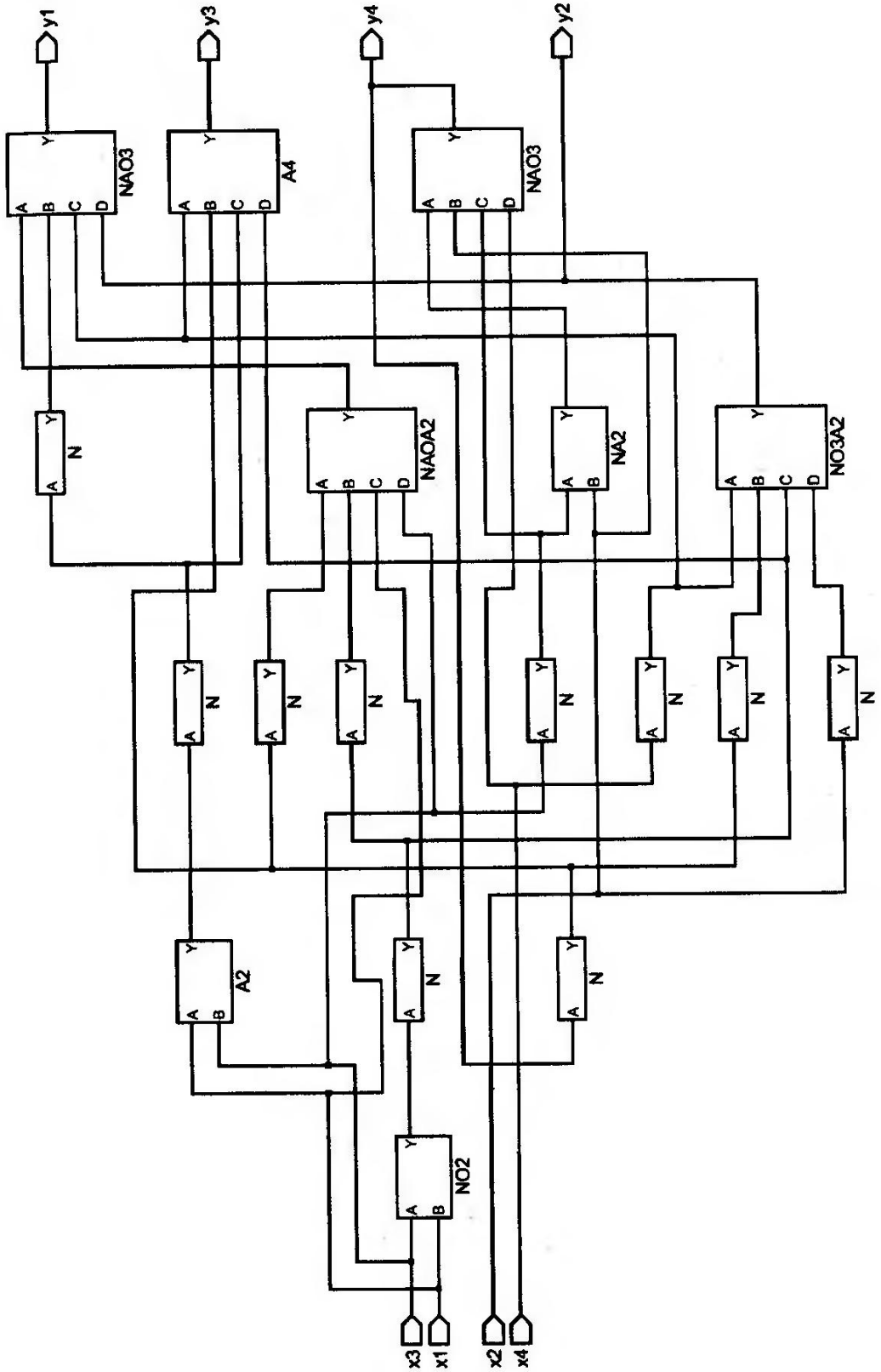
Вариант 4


y1
y3
y4
y2
NA3O2
A3
NAO3
NAO3
NA3
N
NA2
N
NA2
N
NAOA2
N
NEX2
N
A2
N
N
N
NEX2
x1
x2
x3
x4

Ответ к варианту 4

x1 x2 x3 x4	y1 y2 y3 y4
0 0 0 0	0 0 0 0
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 1
0 0 1 1	1 0 0 0
0 1 0 0	1 0 0 1
0 1 0 1	1 0 0 1
0 1 1 0	0 0 0 0
0 1 1 1	0 1 0 0
1 0 0 0	1 1 0 0
1 0 0 1	1 0 0 0
1 0 1 0	1 0 0 1
1 0 1 1	0 0 0 0
1 1 0 0	0 0 0 1
1 1 0 1	1 0 1 1
1 1 1 0	1 0 0 0
1 1 1 1	0 1 0 0

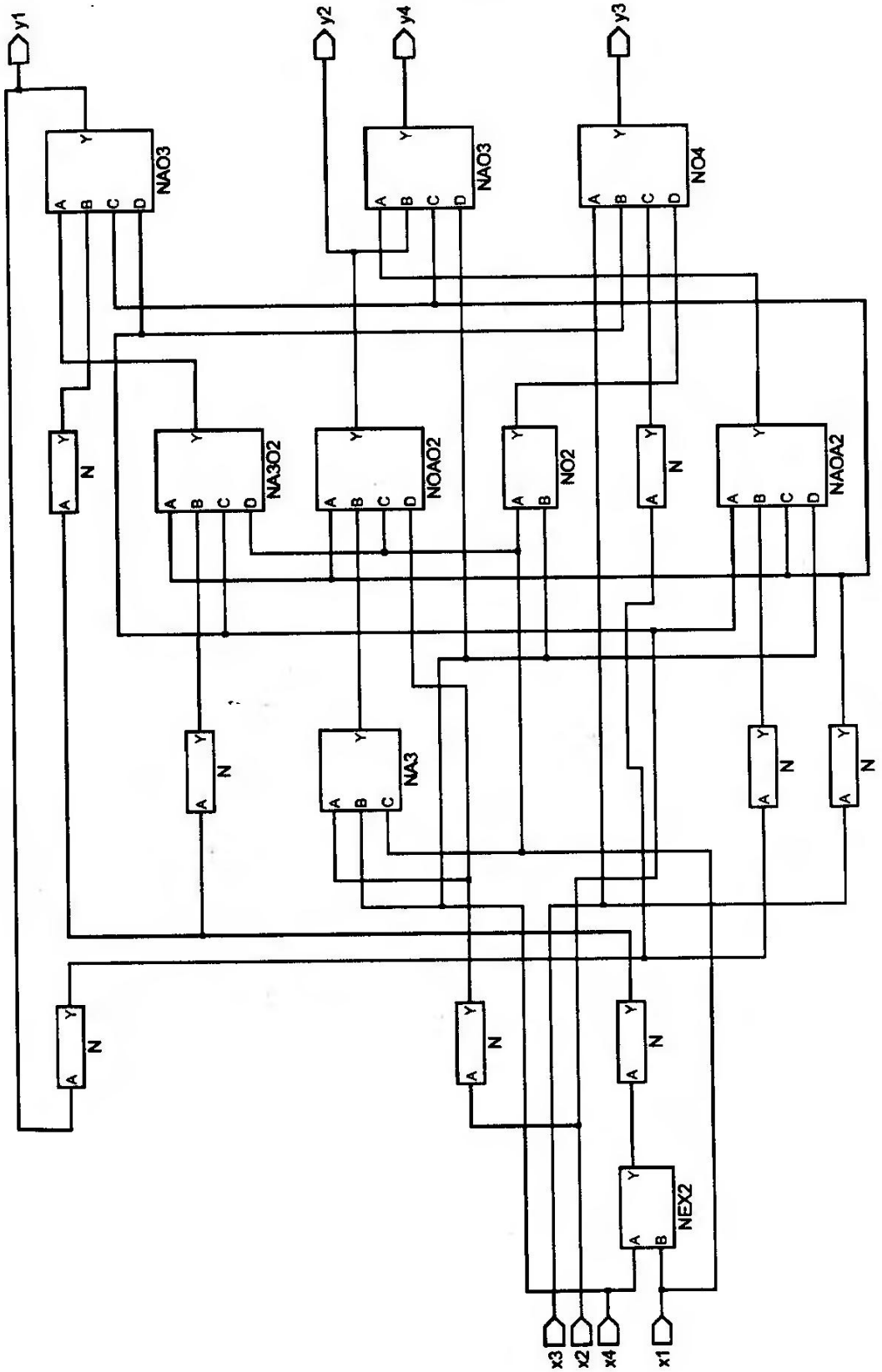
Вариант 5



Ответ к варианту 5

x1	x2	x3	x4	y1	y2	y3	y4
0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0
0	0	1	0	0	0	0	1
0	0	1	1	1	0	0	0
0	1	0	0	1	0	0	1
0	1	0	1	1	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	1	0	0
1	0	0	0	0	0	1	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	1
1	0	1	1	0	0	0	0
1	1	0	0	0	0	0	1
1	1	0	1	1	0	0	1
1	1	1	0	0	0	0	0
1	1	1	1	0	1	0	0

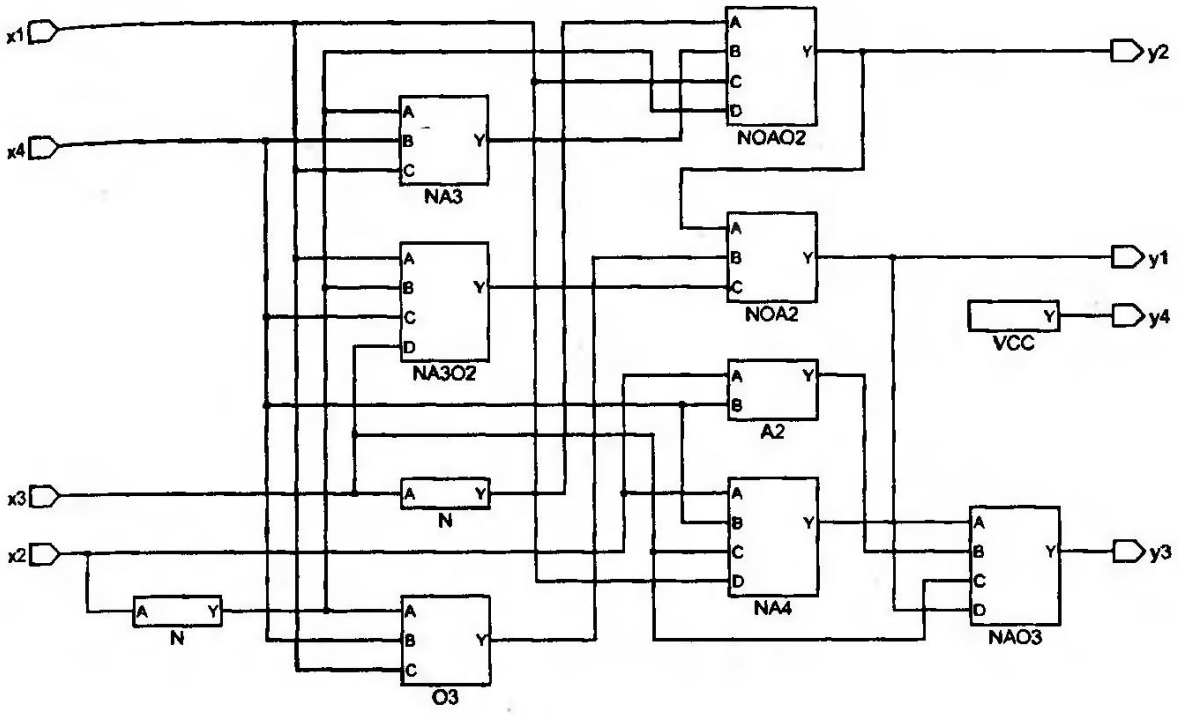
Вариант 6


y1
y2
y4
y3
NAO3
NAO3
NO4
N
NA3O2
NOAO2
NO2
N
NAOA2
N
NA3
N
N
N
N
NEX2
x3
x2
x4
x1

Ответ к варианту 6

x1 x2 x3 x4	y1 y2 y3 y4
0 0 0 0	0 0 0 0
0 0 0 1	0 0 1 0
0 0 1 0	0 0 0 1
0 0 1 1	1 0 0 0
0 1 0 0	1 0 0 1
0 1 0 1	0 0 0 1
0 1 1 0	0 1 0 0
0 1 1 1	0 1 0 0
1 0 0 0	0 0 1 0
1 0 0 1	1 0 0 0
1 0 1 0	1 0 0 1
1 0 1 1	0 1 0 0
1 1 0 0	0 0 0 0
1 1 0 1	1 0 0 1
1 1 1 0	0 0 0 1
1 1 1 1	0 0 0 0

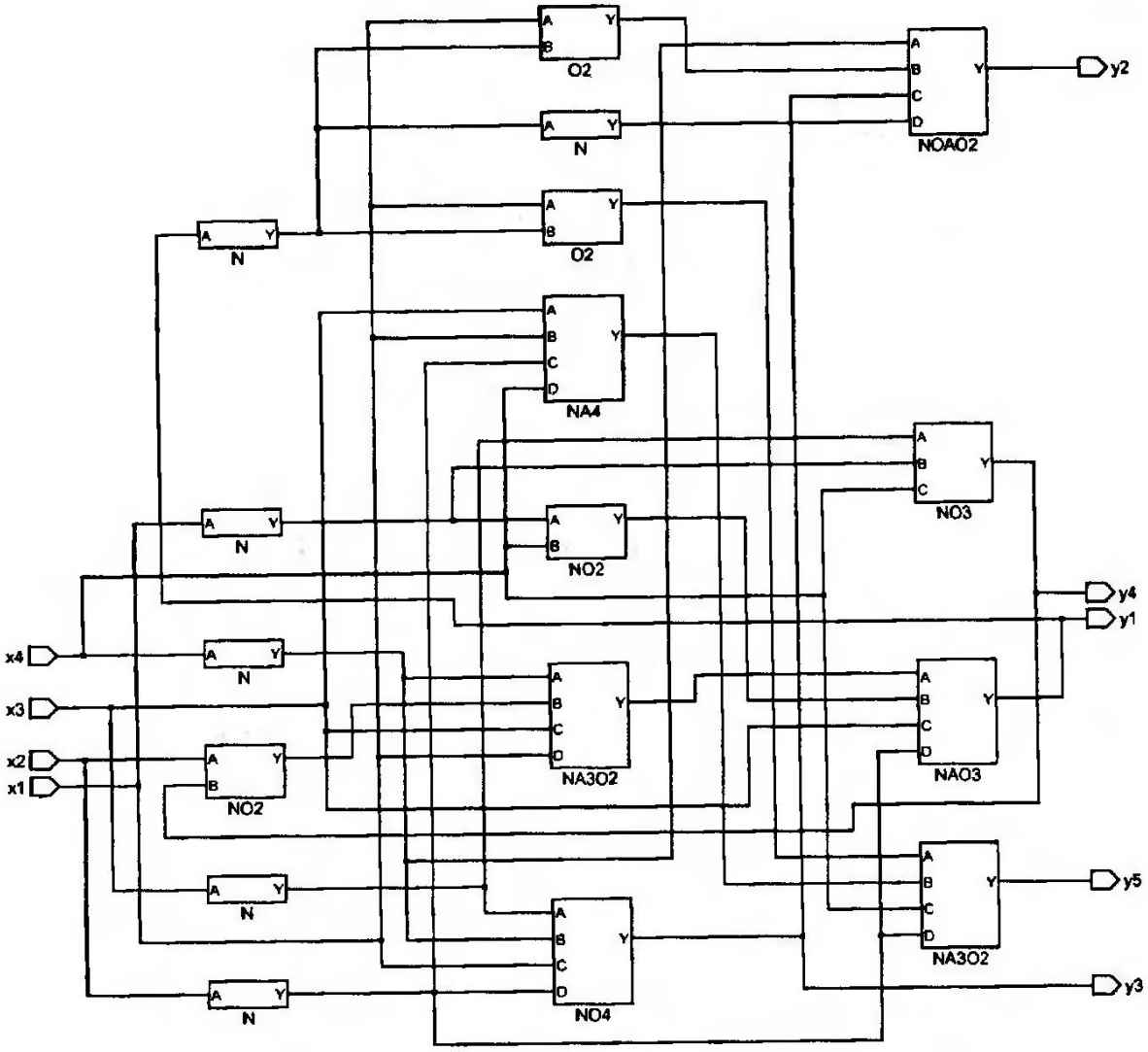
Вариант 7



Ответ к варианту 7

x1	x2	x3	x4	y1	y2	y3	y4
0	0	0	0	0	0	1	1
0	0	0	1	0	0	1	1
0	0	1	0	0	0	0	1
0	0	1	1	0	0	0	1
0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	1
1	0	0	0	0	0	1	1
1	0	0	1	1	0	0	1
1	0	1	0	1	0	0	1
1	0	1	1	0	1	0	1
1	1	0	0	0	0	1	1
1	1	0	1	0	0	0	1
1	1	1	0	0	0	0	1
1	1	1	1	0	0	1	1

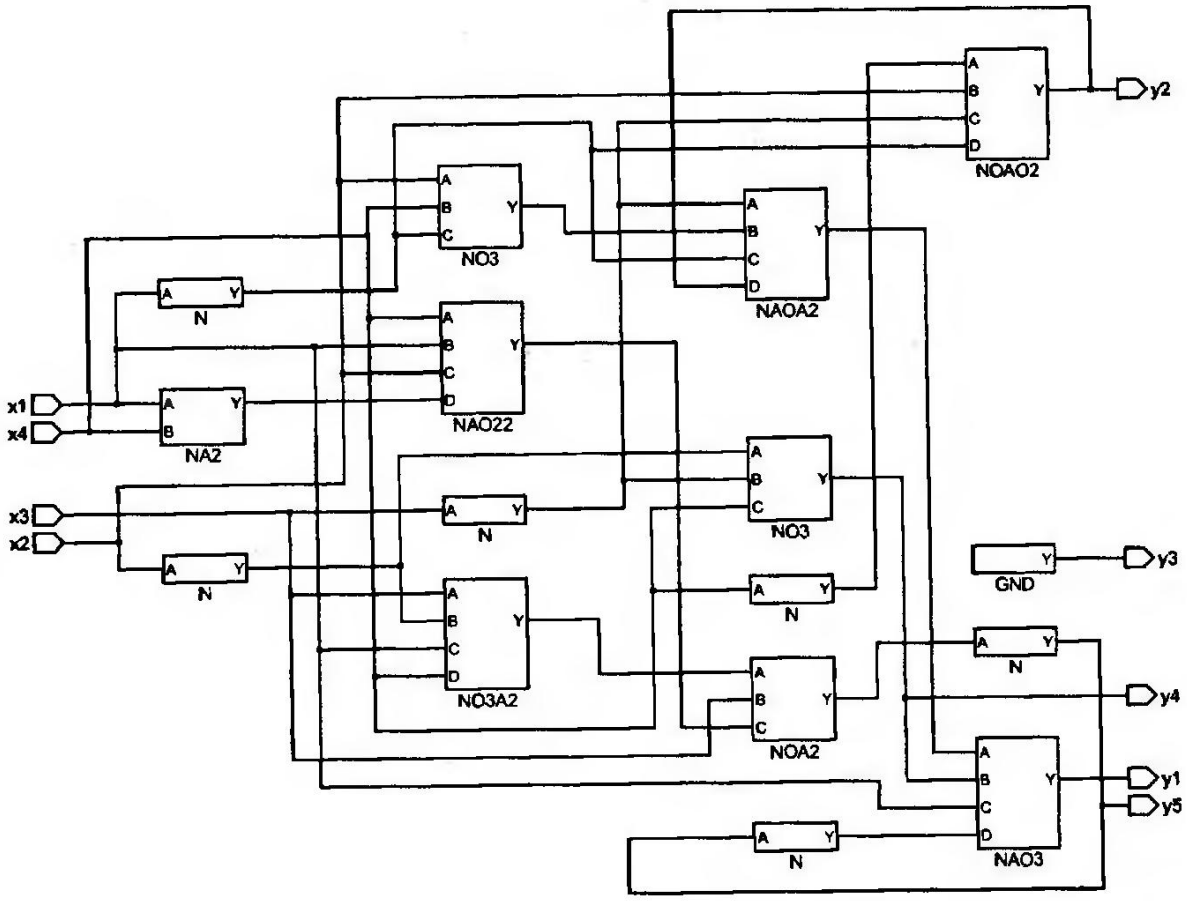
Вариант 8


O2
N
NOA02
y2
O2
N
NA4
NO3
N
NO2
y4
y1
x4
N
x3
NA3O2
NAO3
x2
NO2
x1
N
y5
NA3O2
N
NO4
y3

Ответ к варианту 8

x1	x2	x3	x4	y1	y2	y3	y4	y5
0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0
0	0	1	0	1	0	0	0	1
0	0	1	1	0	1	0	0	0
0	1	0	0	1	0	0	0	1
0	1	0	1	1	1	0	0	1
0	1	1	0	0	0	0	0	1
0	1	1	1	0	0	1	0	0
1	0	0	0	1	0	0	0	0
1	0	0	1	0	1	0	0	0
1	0	1	0	0	0	0	1	0
1	0	1	1	0	1	0	0	1
1	1	0	0	0	0	0	0	1
1	1	0	1	1	0	0	0	0
1	1	1	0	0	0	0	1	1
1	1	1	1	0	1	0	0	0

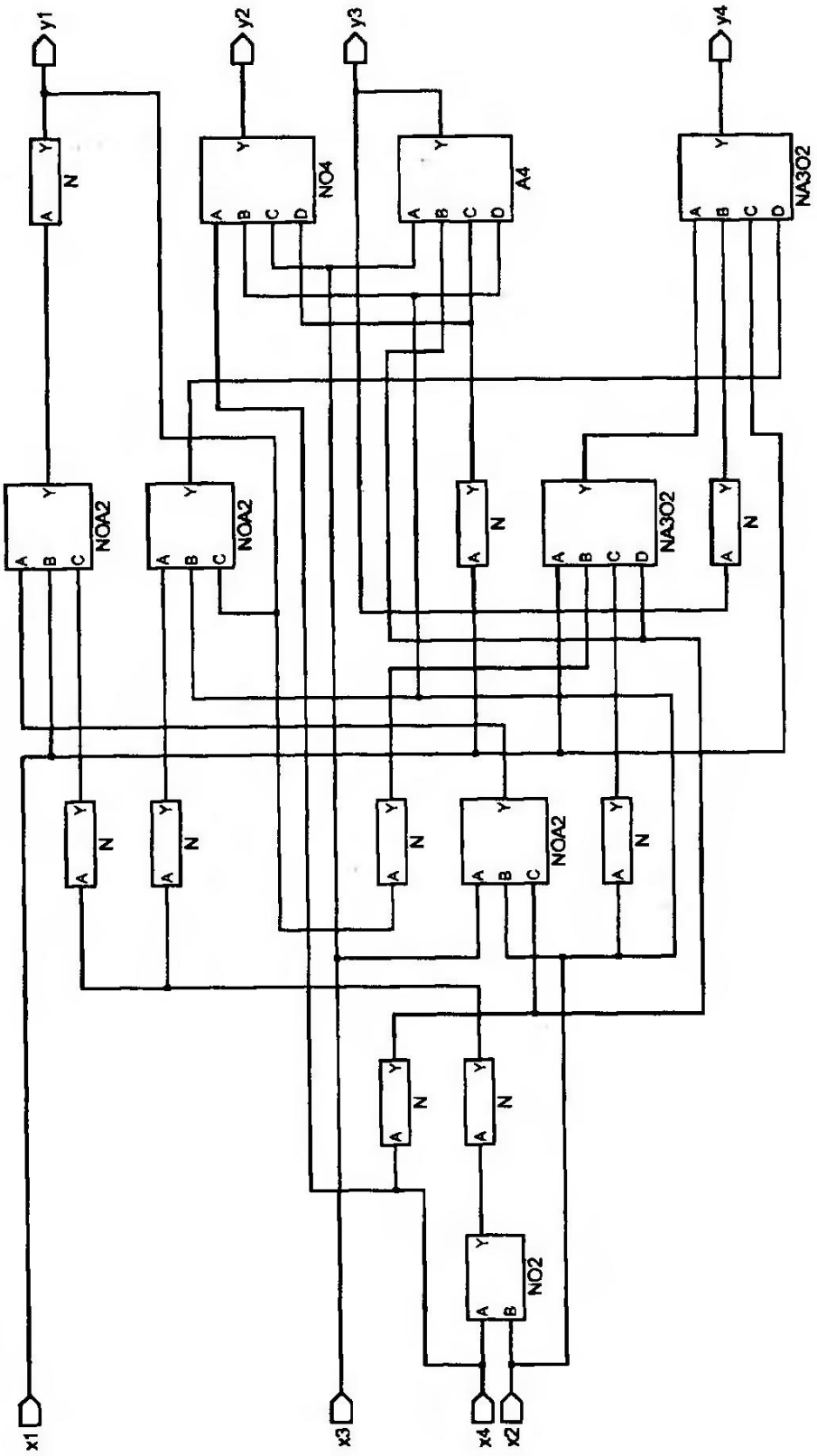
Вариант 9


x1
x4
x3
x2
NA2
N
NO3
NAO22
NAOA2
NOAO2
NO3A2
NOA2
NAO3
GND
y1
y2
y3
y4
y5

Ответ к варианту 9

x1	x2	x3	x4	y1	y2	y3	y4	y5
0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0
0	0	1	0	1	0	0	0	1
0	0	1	1	0	1	0	0	0
0	1	0	0	1	0	0	0	1
0	1	0	1	1	0	0	0	1
0	1	1	0	0	0	0	1	1
0	1	1	1	0	0	0	0	0
1	0	0	0	1	0	0	0	0
1	0	0	1	0	1	0	0	0
1	0	1	0	0	0	0	0	0
1	0	1	1	0	1	0	0	1
1	1	0	0	0	0	0	0	1
1	1	0	1	0	0	0	0	0
1	1	1	0	0	0	0	1	0
1	1	1	1	0	1	0	0	0

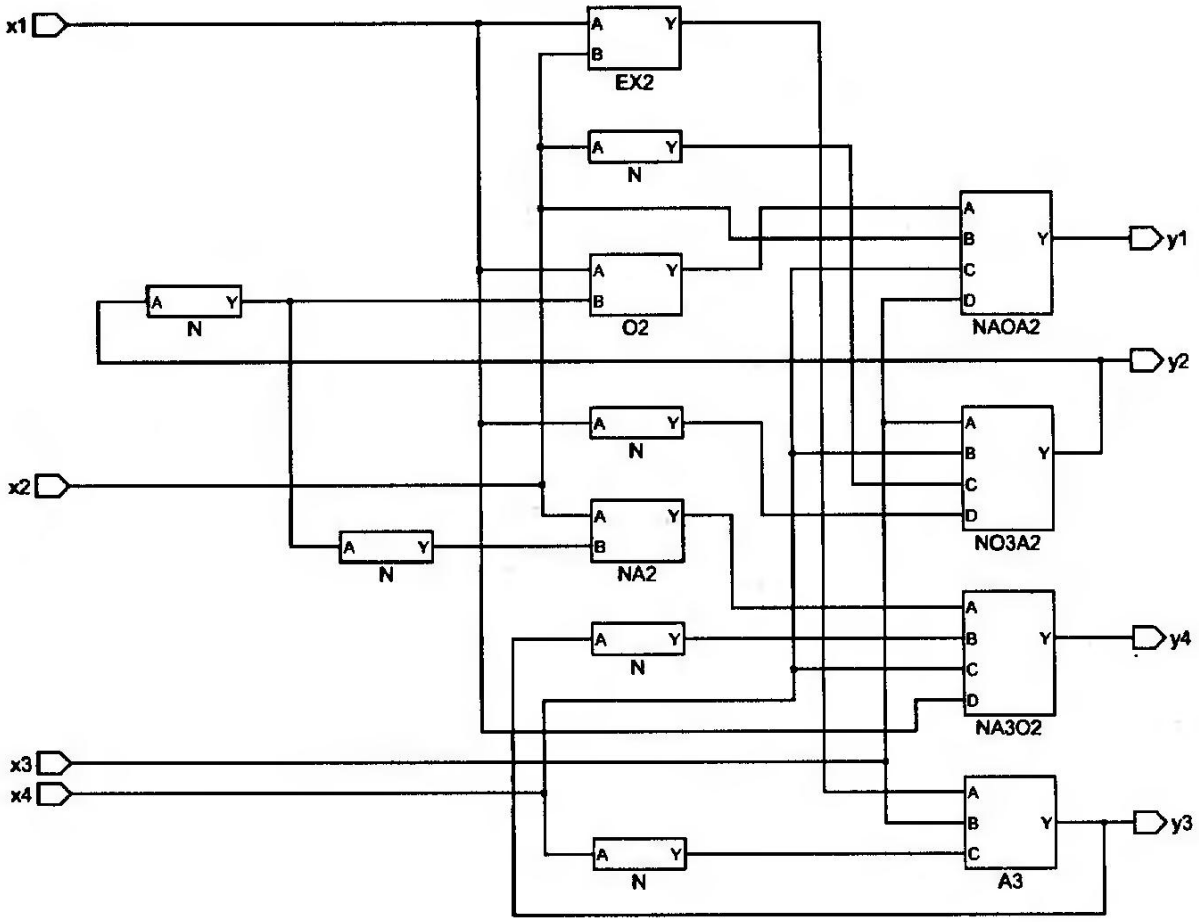
Вариант 10


y1
y2
y3
y4
N
NO4
A4
NA3O2
NOA2
NOA2
N
NA3O2
N
N
N
N
NOA2
N
N
N
NO2
x1
x3
x4
x2

Ответ к варианту 10

x1	x2	x3	x4	y1	y2	y3	y4
0	0	0	0	1	0	0	1
0	0	0	1	1	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	1-	0	0	1
0	1	1	0	0	0	1	1
0	1	1	1	0	0	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	0	0	0	1
1	1	0	0	0	0	0	1
1	1	0	1	1	0	0	0
1	1	1	0	0	0	0	1
1	1	1	1	0	0	0	0

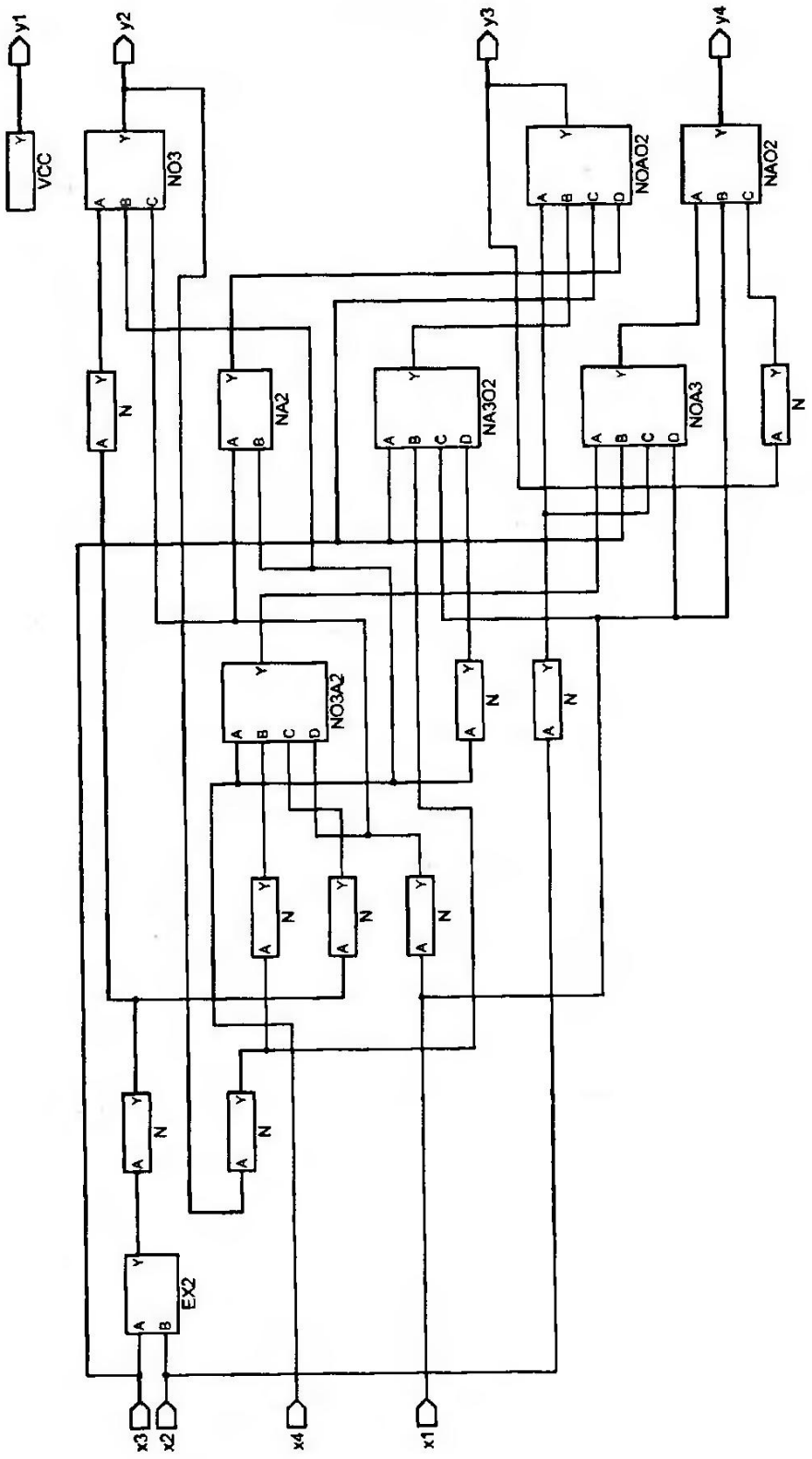
Вариант 11



Ответ к варианту 11

x1	x2	x3	x4	y1	y2	y3	y4
0	0	0	0	1	0	0	1
0	0	0	1	1	0	0	0
0	0	1	0	1	0	0	1
0	0	1	1	0	0	0	0
0	1	0	0	1	1	0	1
0	1	0	1	0	0	0	0
0	1	1	0	0	0	1	1
0	1	1	1	0	0	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	1	1
1	0	1	1	0	0	0	0
1	1	0	0	0	1	0	1
1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0

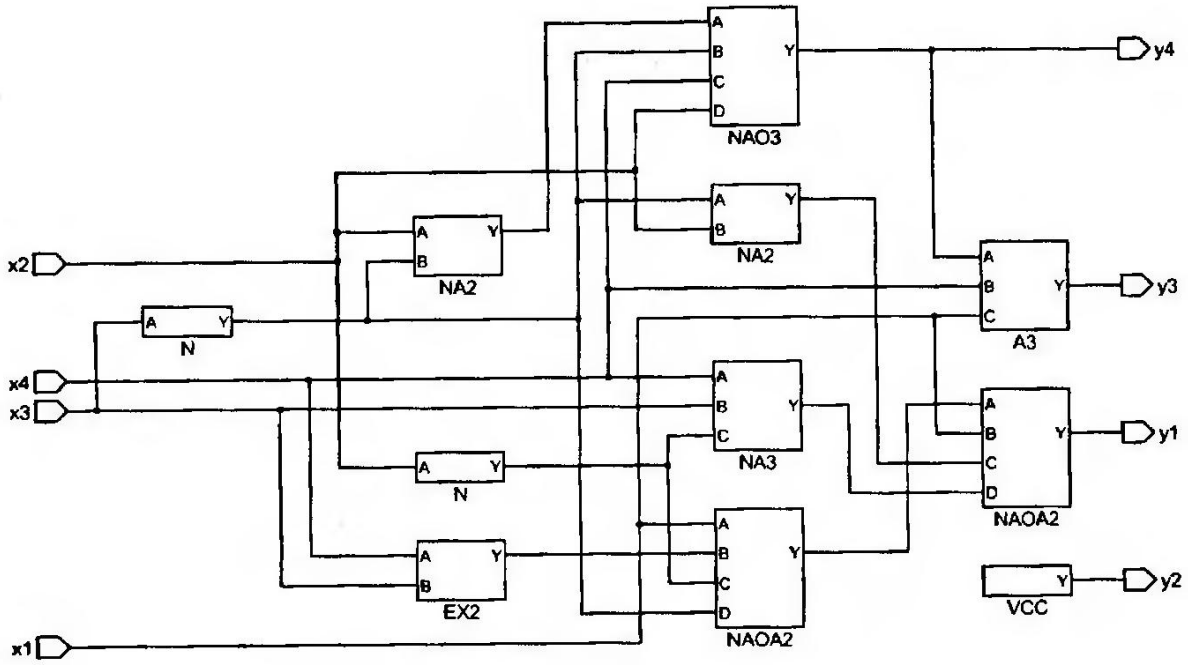
Вариант 12


y1
y2
y3
y4
VCC
NO3
NOAO2
NAO2
N
NA2
NA3O2
NOA3
NO3A2
EX2
x3
x2
x4
x1

Ответ к варианту 12

x1	x2	x3	x4	y1	y2	y3	y4
0	0	0	0	1	0	0	1
0	0	0	1	1	0	0	0
0	0	1	0	1	0	0	0
0	0	1	1	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1
1	1	0	0	1	0	0	1
1	1	0	1	1	0	0	0
1	1	1	0	1	1	0	0
1	1	1	1	1	0	1	0

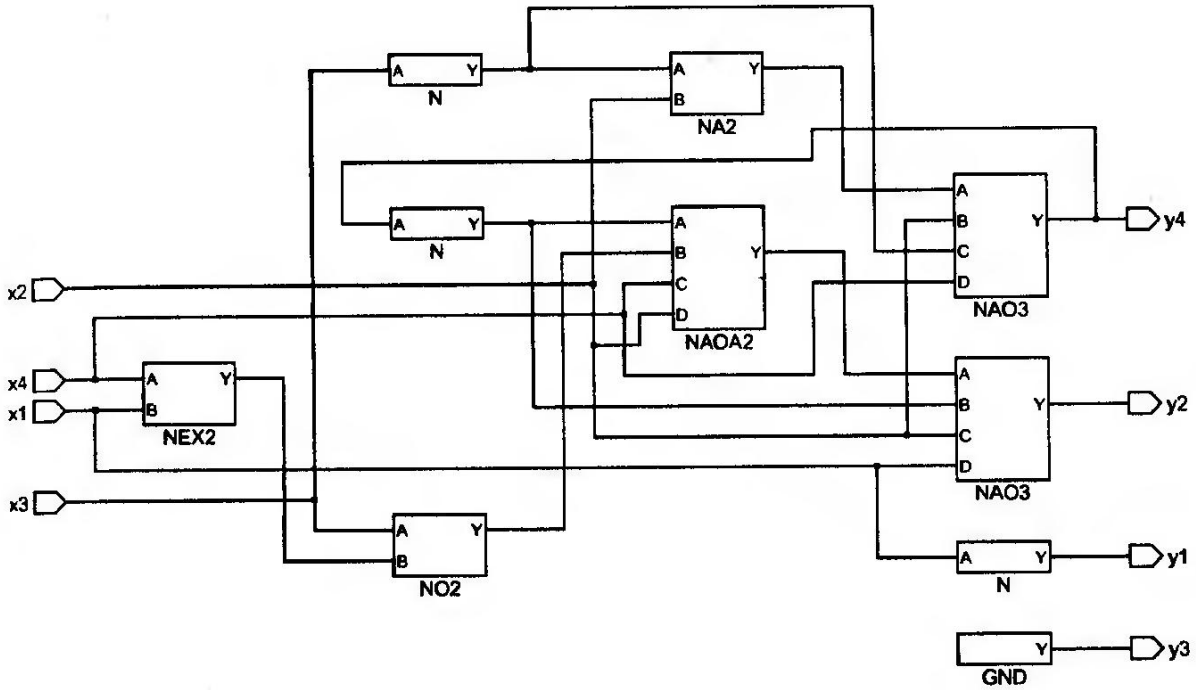
Вариант 13



Ответ к варианту 13

x1 x2 x3 x4	y1 y2 y3 y4
0 0 0 0	0 1 0 0
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 1
0 0 1 1	1 1 0 0
0 1 0 0	1 1 0 1
0 1 0 1	1 1 0 1
0 1 1 0	0 1 0 0
0 1 1 1	0 1 0 0
1 0 0 0	1 1 0 0
1 0 0 1	1 1 0 0
1 0 1 0	1 1 0 1
1 0 1 1	0 1 0 0
1 1 0 0	0 1 0 1
1 1 0 1	1 1 1 1
1 1 1 0	1 1 0 0
1 1 1 1	0 1 0 0

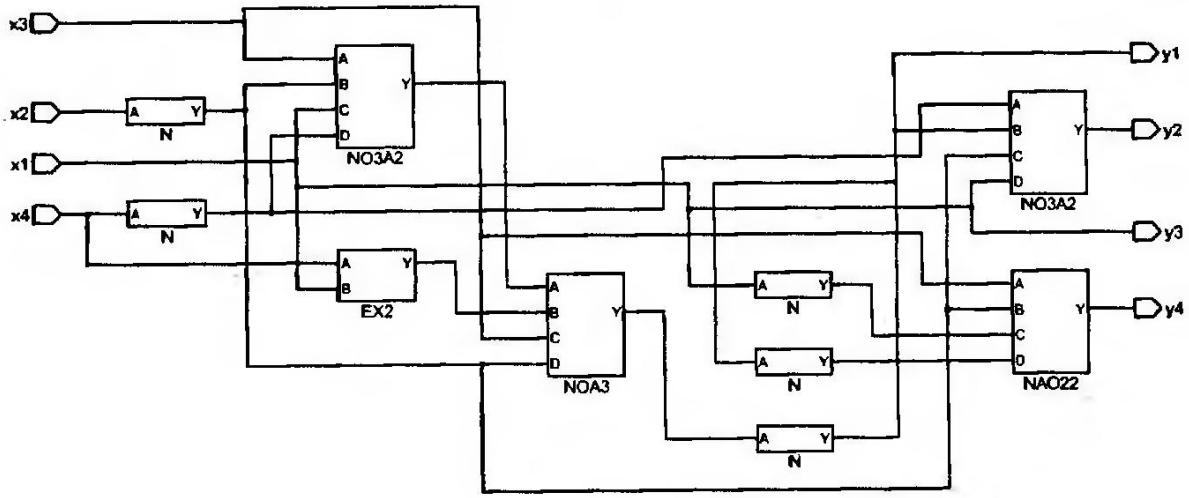
Вариант 14



Ответ к варианту 14

x1	x2	x3	x4	y1	y2	y3	y4
0	0	0	0	1	0	0	0
0	0	0	1	1	1	0	0
0	0	1	0	1	1	0	1
0	0	1	1	1	0	0	0
0	1	0	0	1	0	0	1
0	1	0	1	1	0	0	1
0	1	1	0	1	0	0	0
0	1	1	1	1	1	0	0
1	0	0	0	0	1	0	0
1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	1
1	0	1	1	0	0	0	0
1	1	0	0	0	0	0	1
1	1	0	1	0	0	0	1
1	1	1	0	0	0	0	0
1	1	1	1	0	1	0	0

Вариант 15



Ответ к варианту 15

x1	x2	x3	x4	y1	y2	y3	y4
0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0
0	0	1	0	0	0	0	0
0	0	1	1	1	0	0	0
0	1	0	0	1	0	0	1
0	1	0	1	1	0	0	1
0	1	1	0	0	0	0	0
0	1	1	1	0	1	0	0
1	0	0	0	0	0	1	0
1	0	0	1	0	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	0	0	1	0
1	1	0	0	0	0	1	1
1	1	0	1	1	0	1	1
1	1	1	0	0	0	1	0
1	1	1	1	0	1	1	0

ЛАБОРАТОРНАЯ РАБОТА 3
Описание и моделирование регулярных (систолических) схем

Задание

Провести описание (двумя способами) логической схемы с регулярным соединением элементов и сравнить результаты моделирования.

Способ 1. Составить иерархическое описание ограниченного фрагмента регулярной схемы.

Первый (нижний) уровень иерархии составляют функциональные описания логических элементов.

Второй (средний) уровень иерархии должны составлять две подсхемы. Например, в варианте 1 они выделены штриховой линией. Аналогично для остальных вариантов.

Третий (верхний) уровень иерархии — это описание схемы в целом.

Способ 2. Составить параметризованное описание регулярной схемы, т. е. описать регулярную схему с использованием операторов **generate, generic** для *произвольной разрядности* N.

Рекомендуемый порядок выполнения работы

1. Ввести имена входов и выходов схемы с использованием типа **BIT_VECTOR.**
2. Составить VHDL-модель каждого из **типов** элементов, входящих в схему.
3. Составить иерархическое VHDL-модель схемы в целом для конкретного значения N (обычно N = 4).
4. Написать тестирующую программу и провести моделирование иерархического описания, получить временную диаграмму.
5. Провести описание регулярной схемы с использованием операторов **generate, generic** для *произвольной разрядности* N.
6. Написать тестирующую программу для моделирования по способу 2 при том же значении выбранного в п. 3 параметра N.
7. Сравнить результаты моделирования по способу 1 и способу 2 (результаты должны быть одинаковыми).

8. Проверить модель по способу 2 при других значениях параметра N (N = 2, 3, 5).

Требования к оформлению отчета

1. В отчете должна быть нарисована регулярная логическая **схема.** При этом обозначения сигналов, элементов схемы должны ***соответствовать*** описанию на языке VHDL.

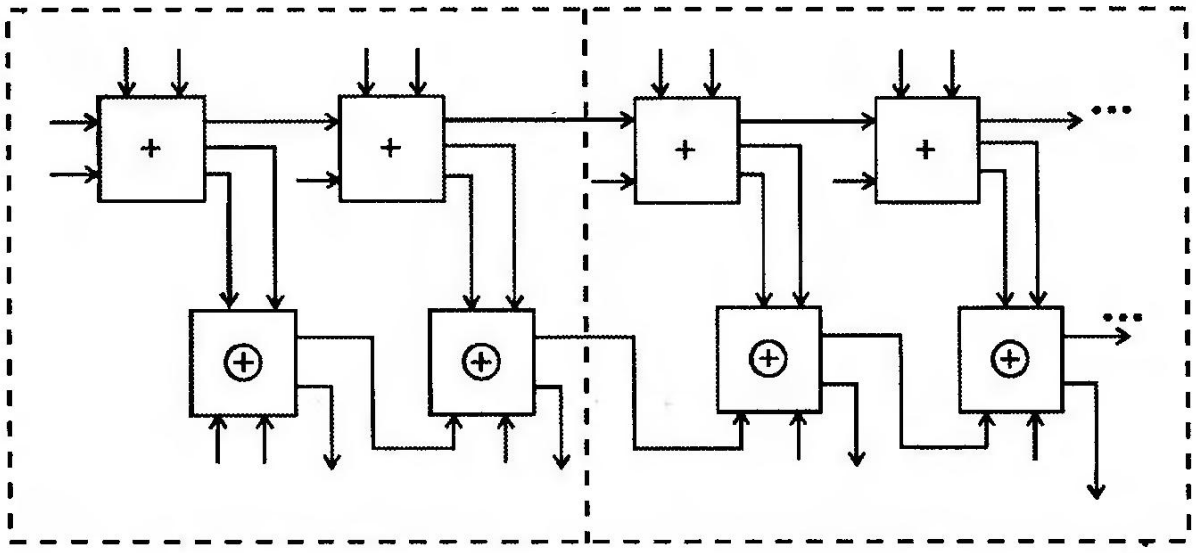
2. В отчете должен содержаться **VHDL-коды** схемы и **тестирующие программы** по способу 1 и способу 2.

3. VHDL-коды и тесты должны содержать **комментарии**:

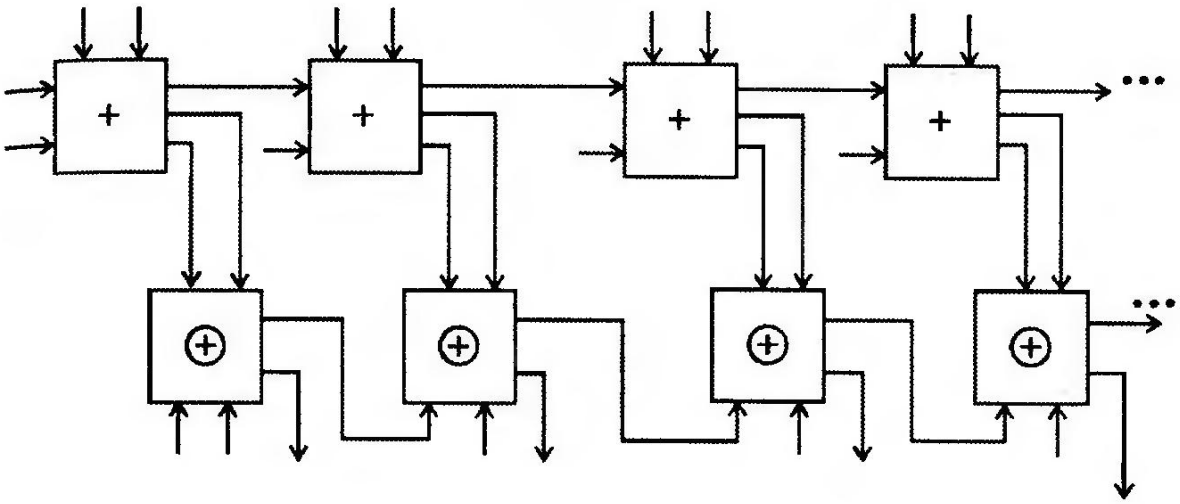
автор разработанной VHDL-модели;
номер варианта;
разрядность схемы;

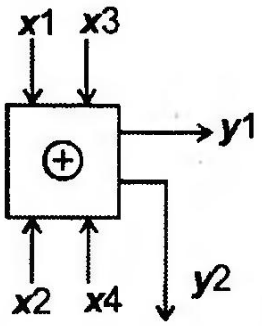
4. В отчете должны содержаться **временные диаграммы,** соответствующие тестирующим программам.

Вариант 1



Регулярная схема (N = 4) для иерархического описания (способ 1)

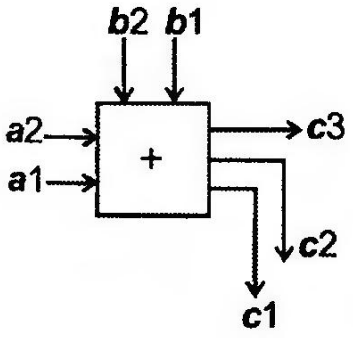




Логический элемент

$y1 = x1 \oplus x2$

$y2 = x3 \oplus x4$



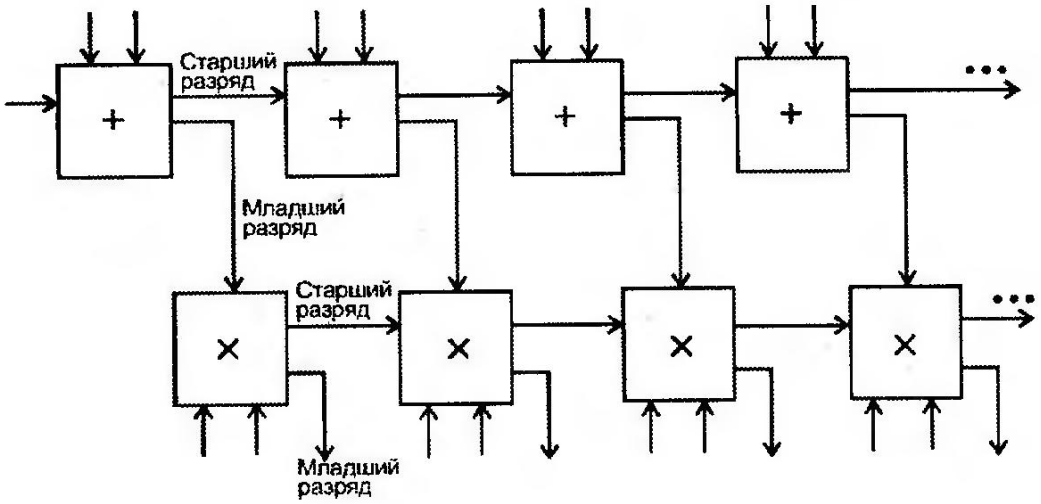
Сумматор

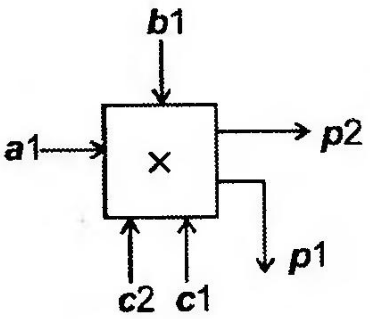
(*a*2, *a*1) + (*b*2, *b*1) = (*c*3, *c*2, *c*1)

*a*2, *b*2, *c*3 - старший разряд

Регулярная схема (N = 4) для параметризованного описания (способ 2)

Вариант 2

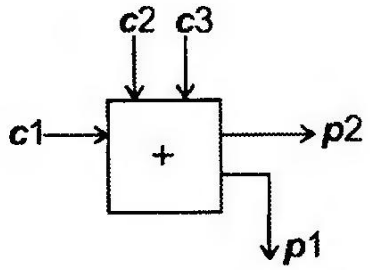




Умножение трех чисел

(*a*1)× (*b*1)× (*c*2, *c*1)=(*p*2, *p*1)

*c*2, *p*2 - старший разряд

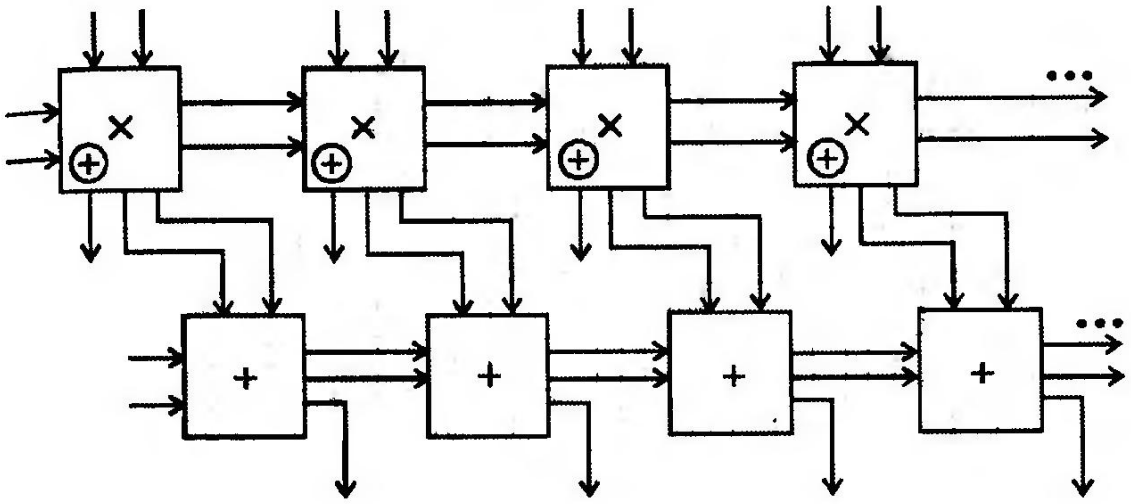


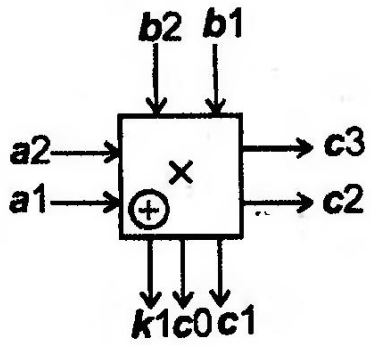
Сумматор трех чисел

(*c*1)+ (*c*2)+(*c*3) =(*p*2, *p*1)

*p*2 - старший разряд

Вариант 3



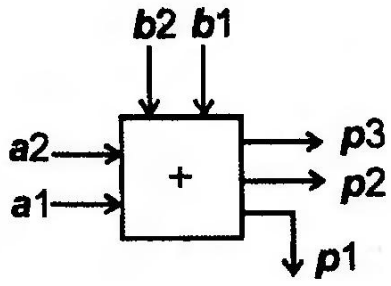


Умножитель

(**a**2, **a**1)×(**b**2, **b**1) = (**c**3, **c**2, **c**1, **c**0)

a2, **b**2, **c**3 - старший разряд

k1 = **b**2 ⊕ **b**1

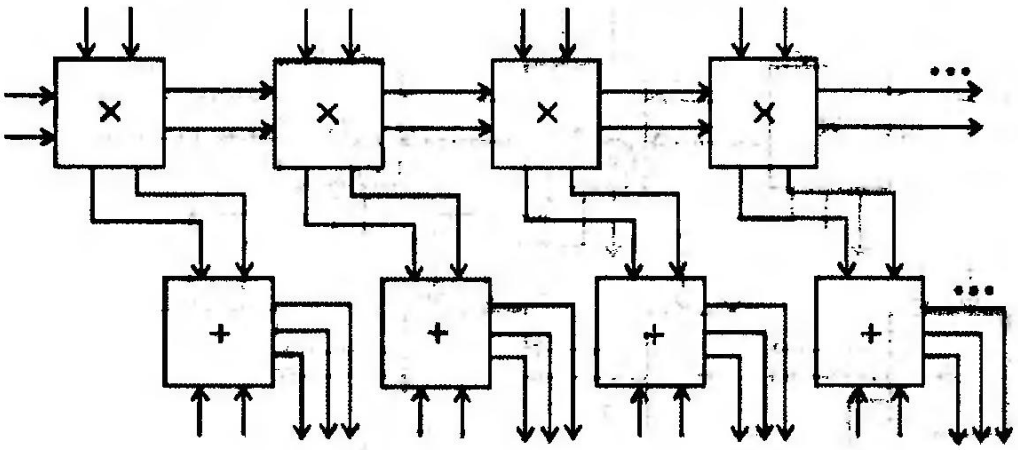


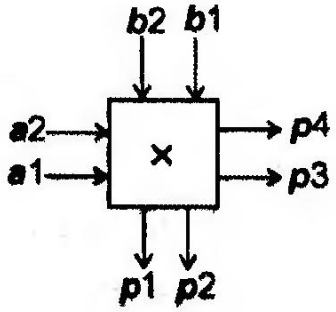
Сумматор

(**a**2, **a**1)+(**b**2, **b**1)= (**p**3, **p**2, **p**1)

a2, **b**2, **p**3 - старший разряд

Вариант 4

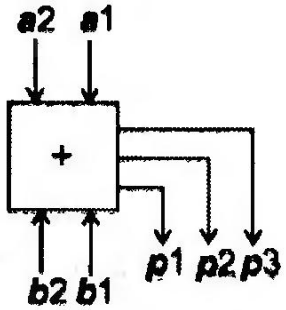




Умножитель

(a2, a1)×(b2, b1) = (p4, p3, p2, p1)

p4 - старший разряд, p1 - младший разряд

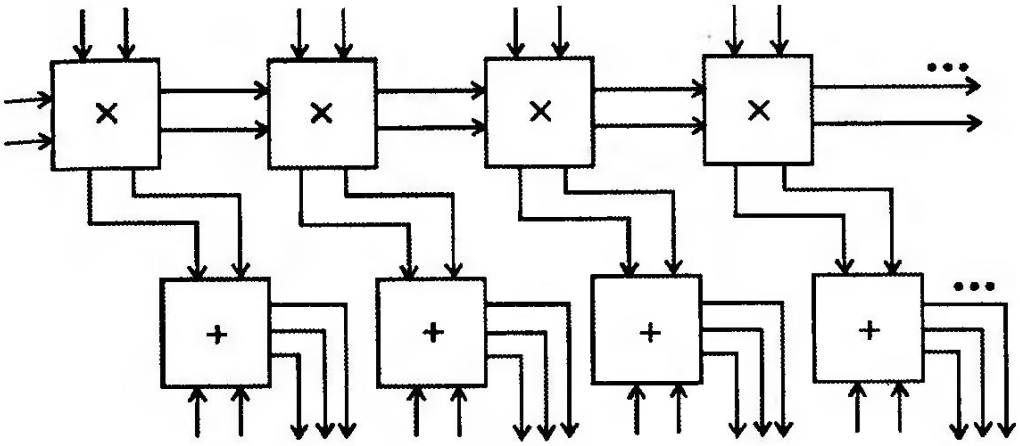


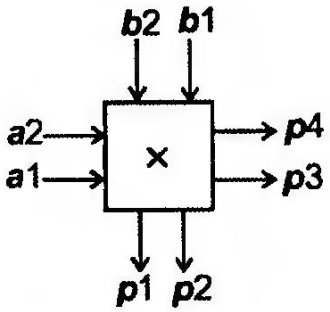
Сумматор

(a2, a1)+(b2, b1)= (p3, p2, p1)

p3 - старший разряд

Вариант 5

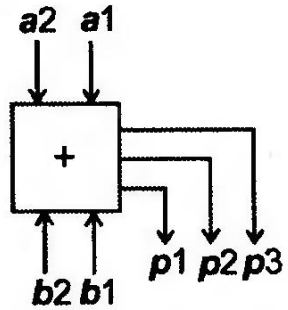




Умножитель

(***a***2, ***a***1)×(***b***2, ***b***1) = (***p***4, ***p***3, ***p***2, ***p***1)

p4 - старший разряд, ***p***1 - младший разряд

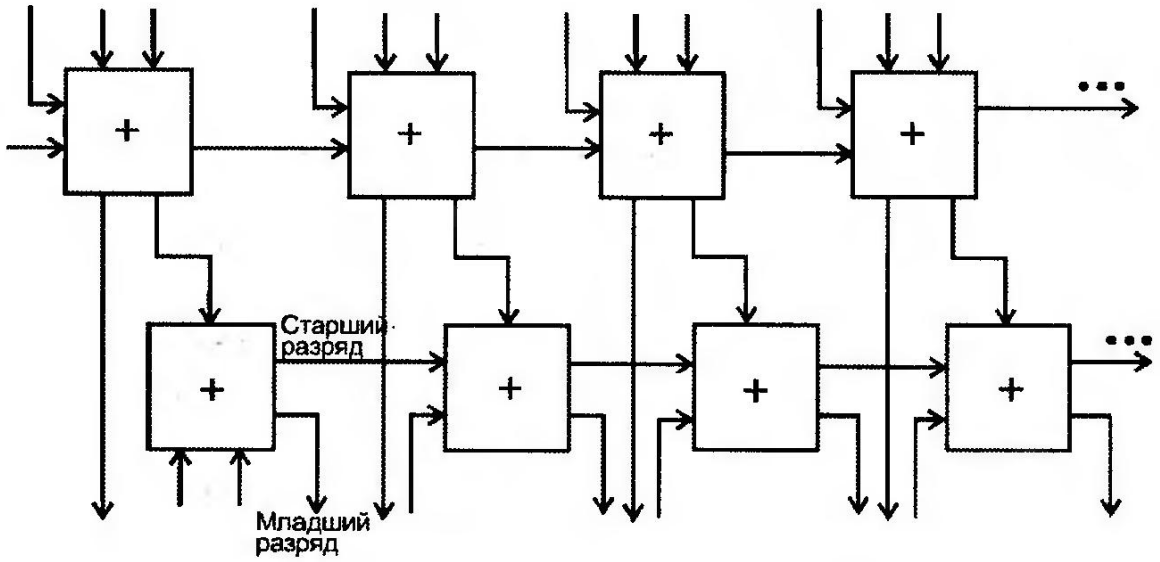


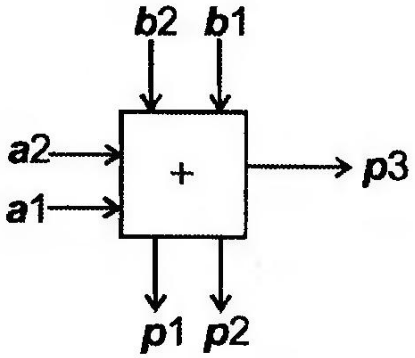
Сумматор

(***a***2, ***a***1)+(***b***2, ***b***1)= (***p***3, ***p***2, ***p***1)

p3 - старший разряд

Вариант 6

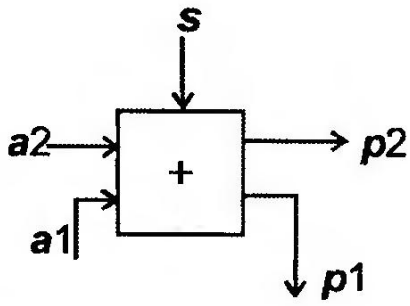




Сумматор

(**a**2, **a**1)+(**b**2, **b**1)=(**p**3, **p**2, **p**1)

a2, **b**2, **p**3 - старший разряд

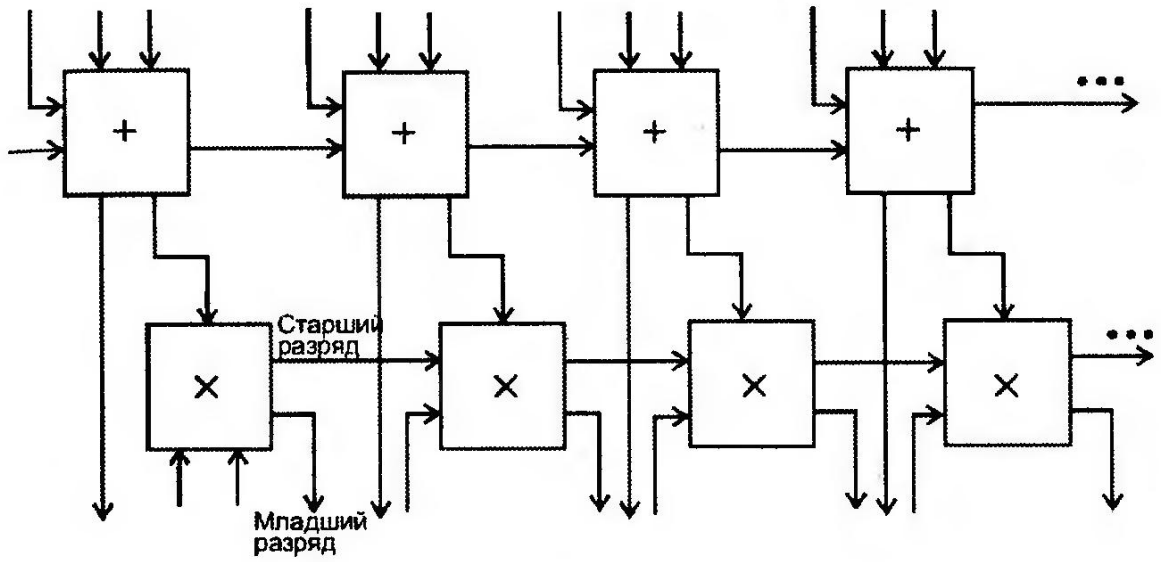


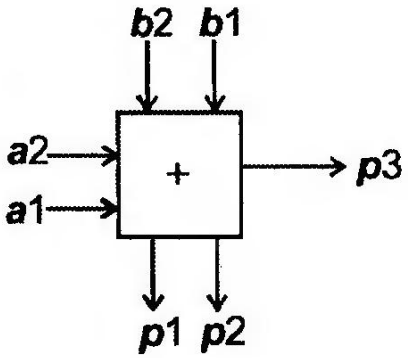
Сумматор

(**a**2, **a**1) + **s** = (**p**2, **p**1)

a2, **p**2 - старший разряд

Вариант 7

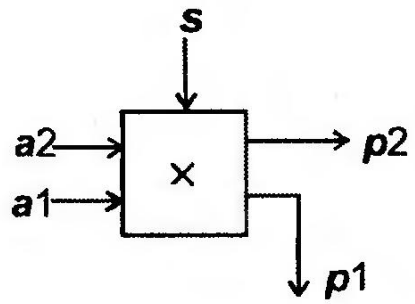




Сумматор

(*a*2, *a*1)+(*b*2, *b*1)=(*p*3, *p*2, *p*1)

*a*2, *b*2, *p*3 - старший разряд

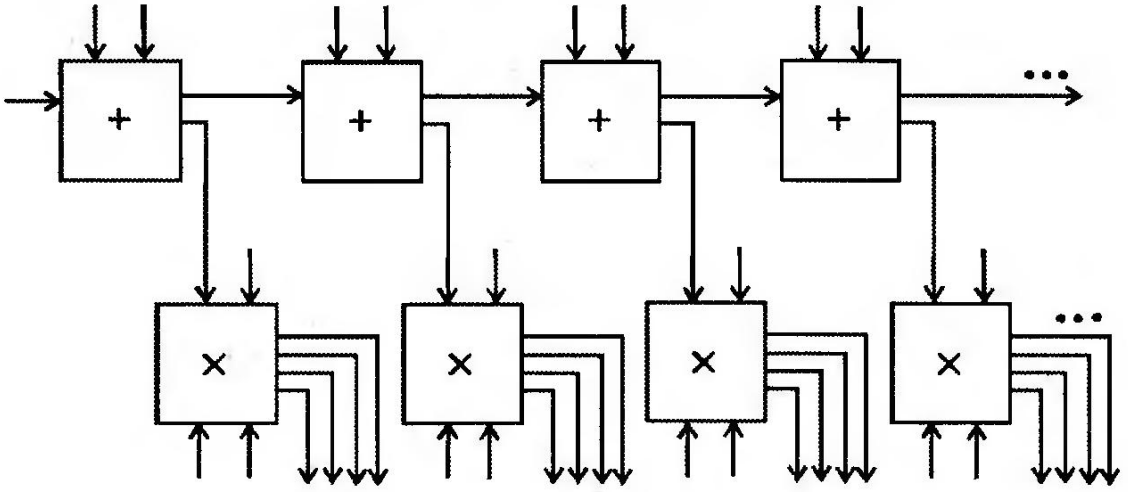


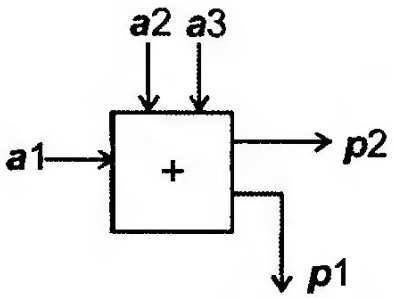
Умножитель

(*a*2, *a*1)× *s* = (*p*2, *p*1)

*a*2, *p*2 - старший разряд

Вариант 8

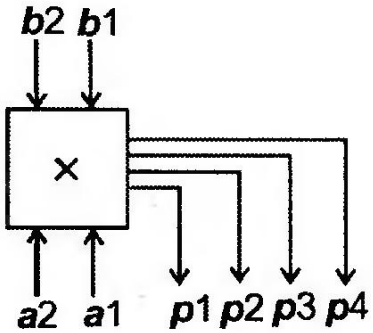




Сложение трех чисел

(*a*1) + (*a*2) + (*a*3) = (*p*2, *p*1)

*p*2 - старший разряд

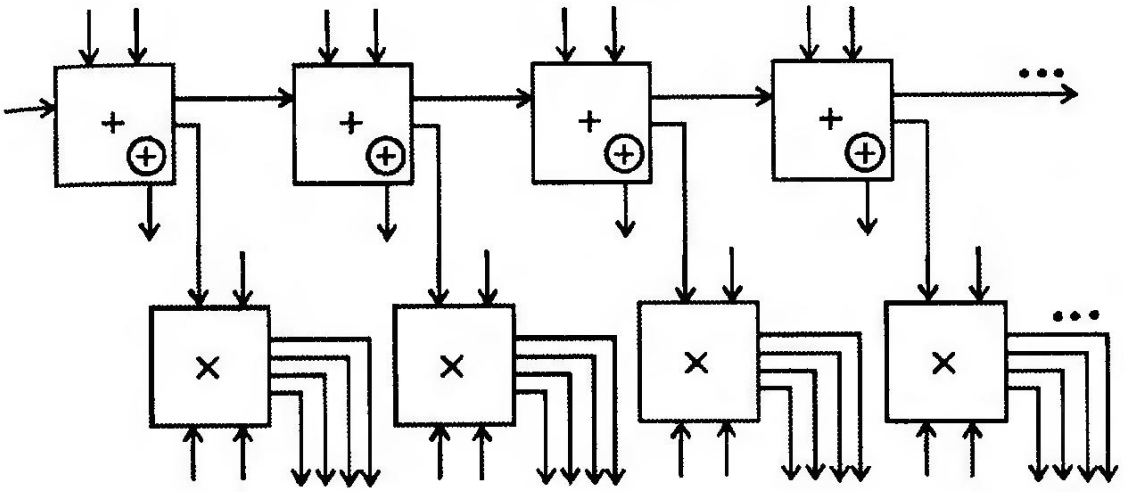


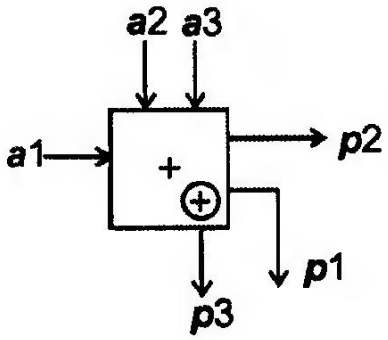
Умножитель

(*a*2, *a*1) × (*b*2, *b*1) = (*p*4, *p*3, *p*2, *p*1)

*p*4 - старший разряд

Вариант 9



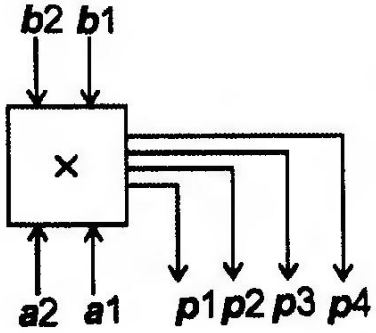


Сложение трех чисел

($a1$)+ ($a2$) + ($a3$) = ($p2$, $p1$)

$p3 = a2 \oplus a3$

$p2$ - старший разряд

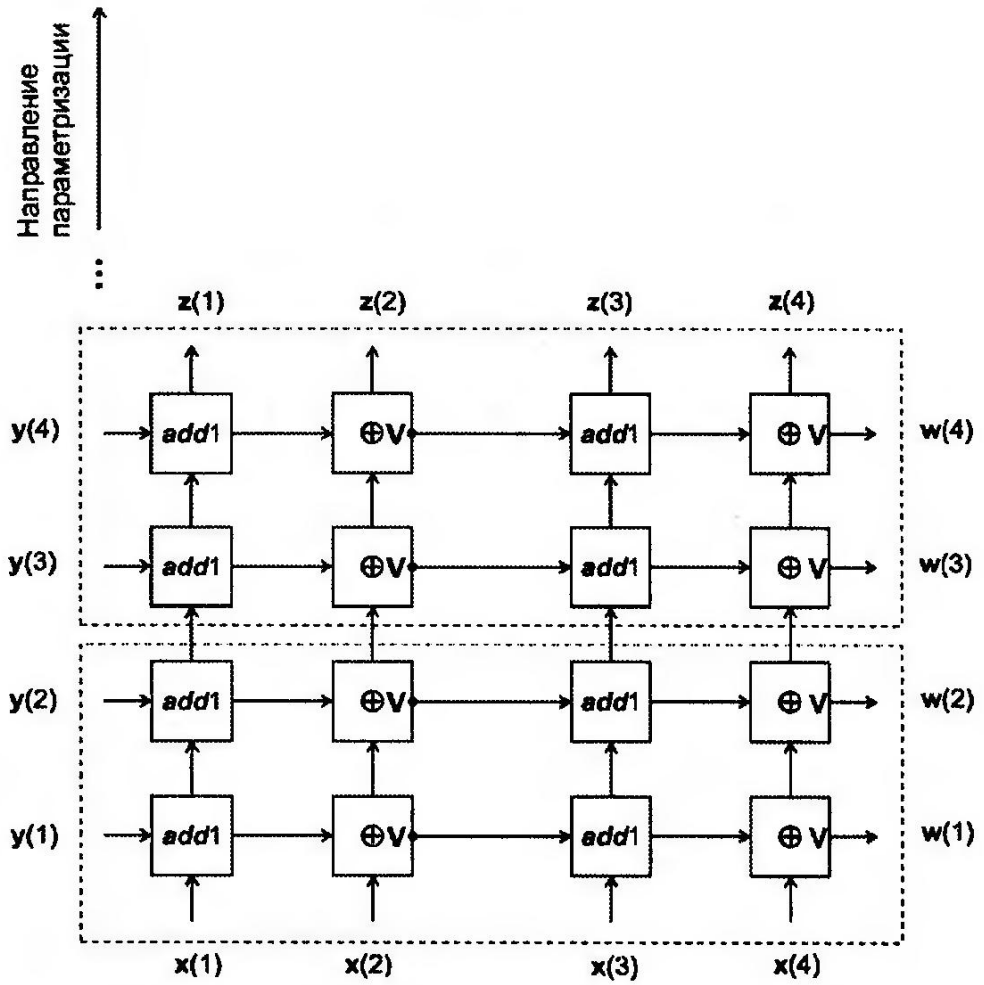


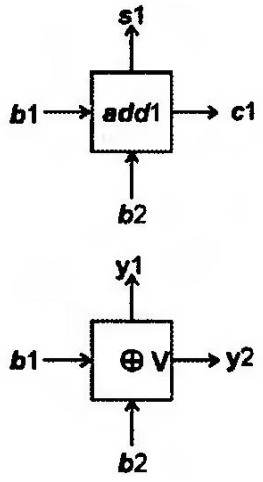
Умножитель

($a2$, $a1$)× ($b2$, $b1$)= ($p4$, $p3$, $p2$, $p1$)

$p4$ - старший разряд

Вариант 10





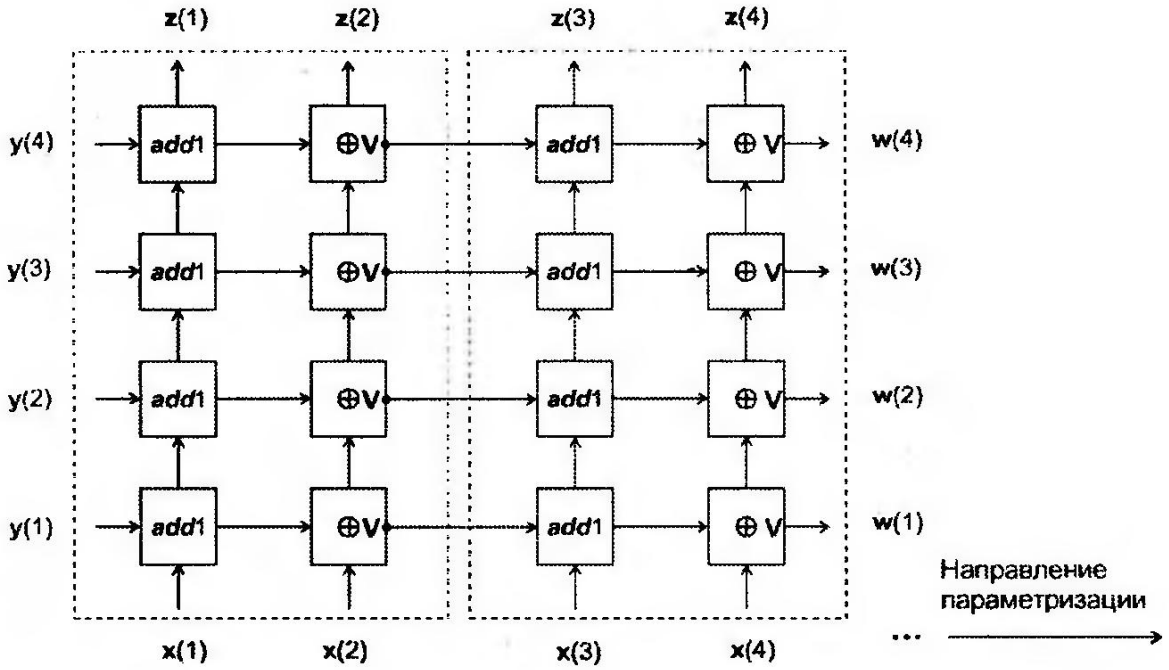
Полусумматор

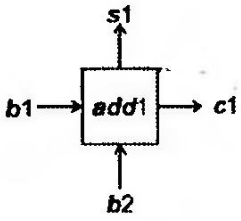
(*b*1) + (*b*2) = (*c*1, *s*1)

*y*1 = *b*1 ⊕ *b*2

*y*2 = *b*1 V *b*2

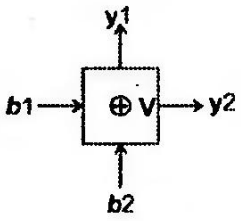
Вариант 11





Полусумматор

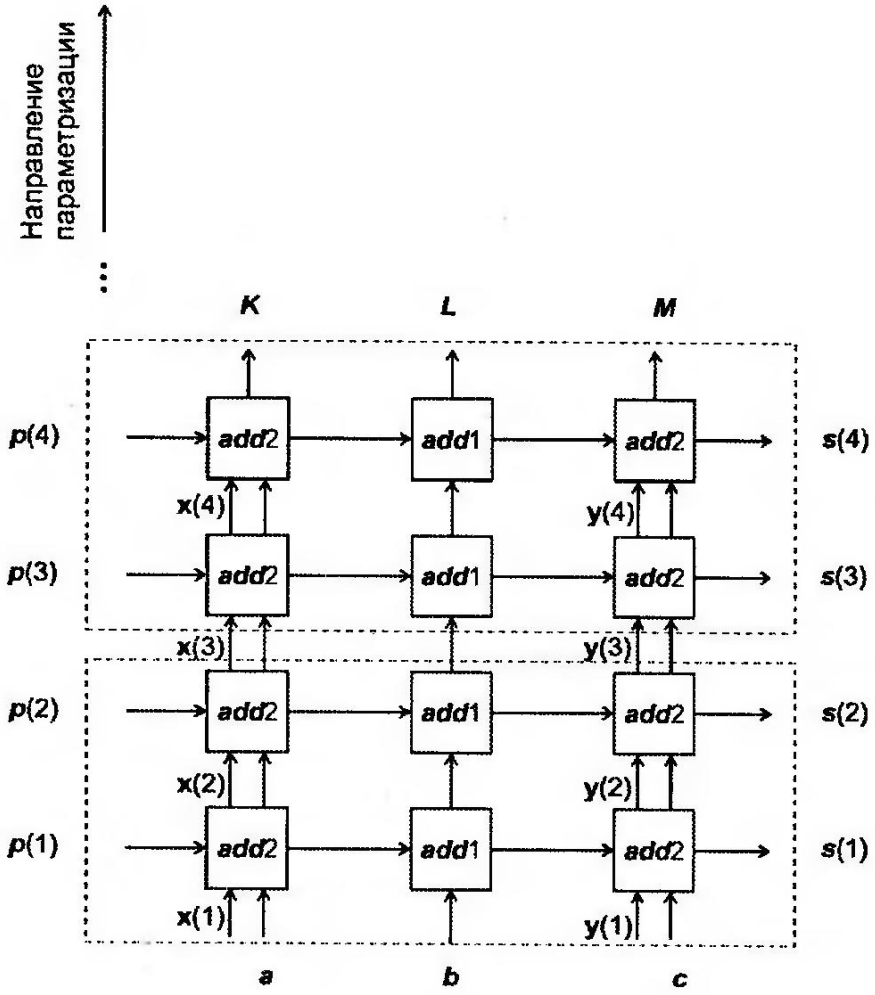
(*b*1) + (*b*2) = (*c*1, *s*1)

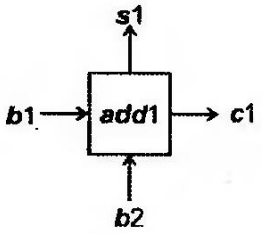


y1 = *b*1 ⊕ *b*2

y2 = *b*1 V *b*2

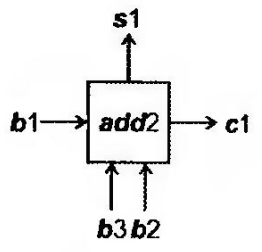
Вариант 12





Полусумматор

(*b*1) + (*b*2) = (*c*1, *s*1)

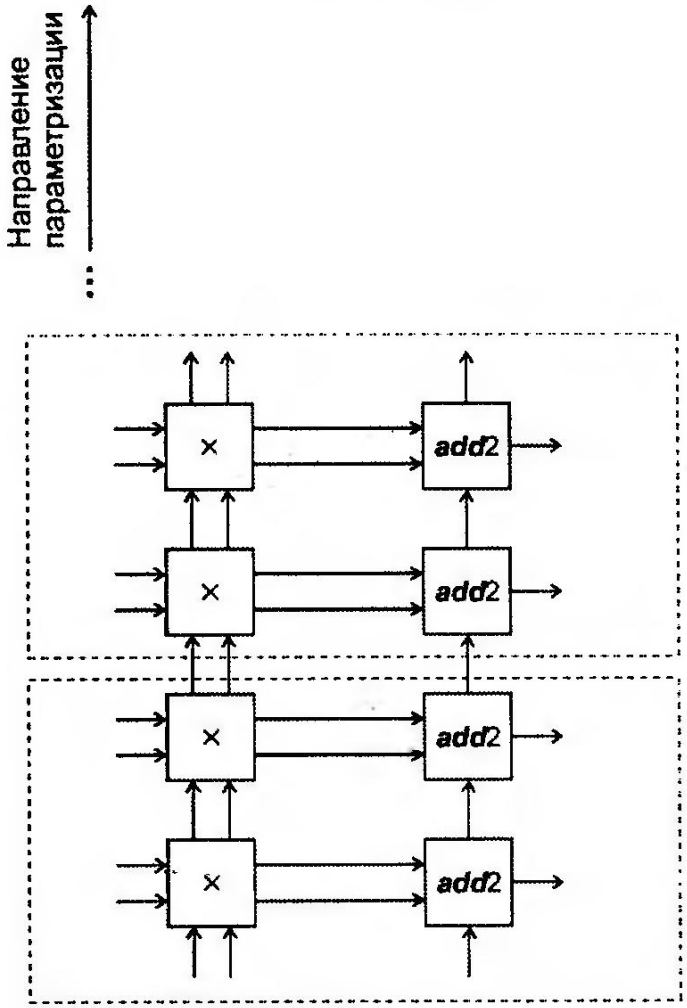


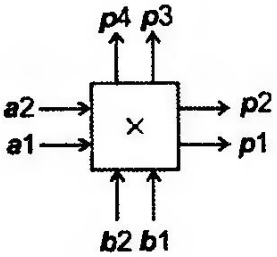
Сумматор *add2*

(*b*1) + (*b*2) + (*b*3) = (*c*1, *s*1)

*s*1 - сумма, *c*1 - перенос

Вариант 13

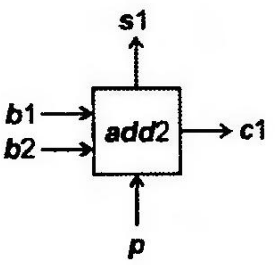




Умножитель

(*a*2, *a*1)×(*b*2, *b*1) = (*p*4, *p*3, *p*2, *p*1)

*a*2, *b*2, *p*4 - старшие разряды

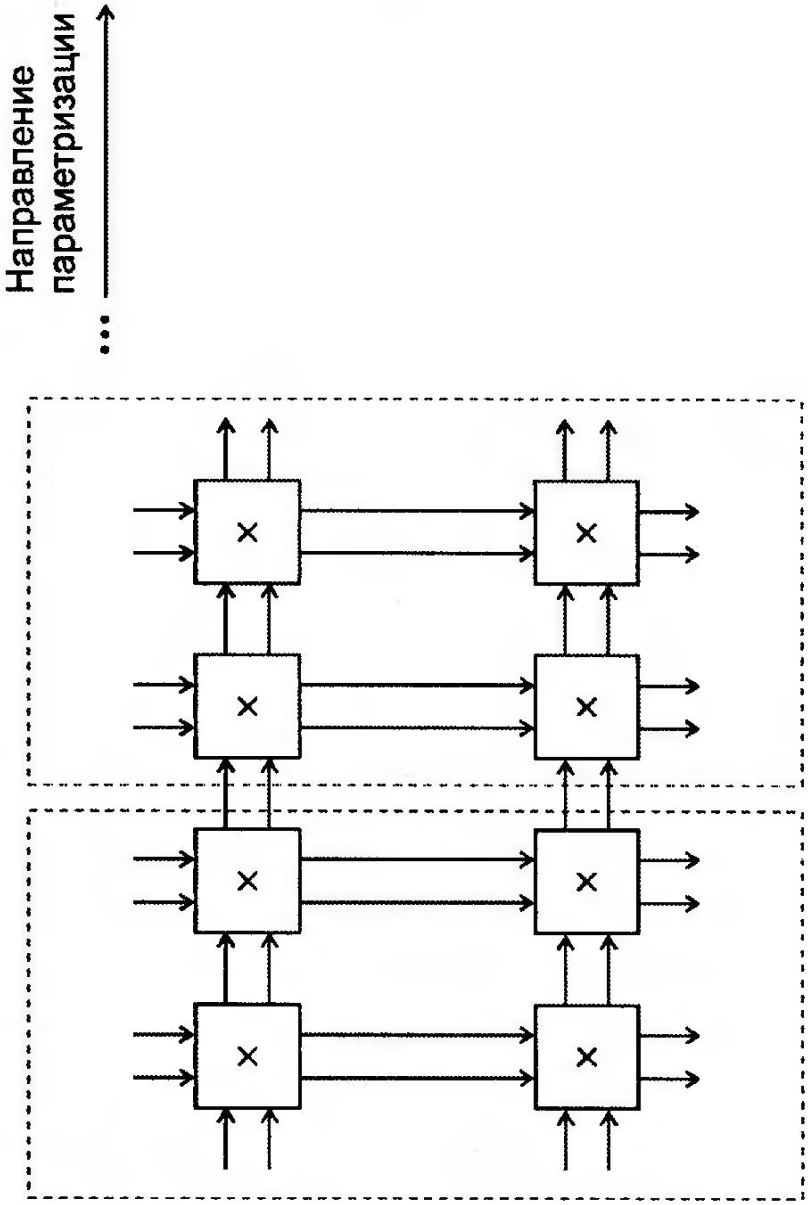


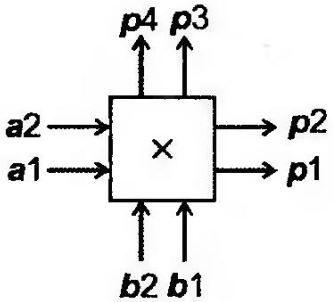
Сумматор *add*2

(*b*1) + (*b*2) + (*p*) = (*c*1, *s*1)

*s*1 - сумма, *c*1 - перенос

Вариант 14



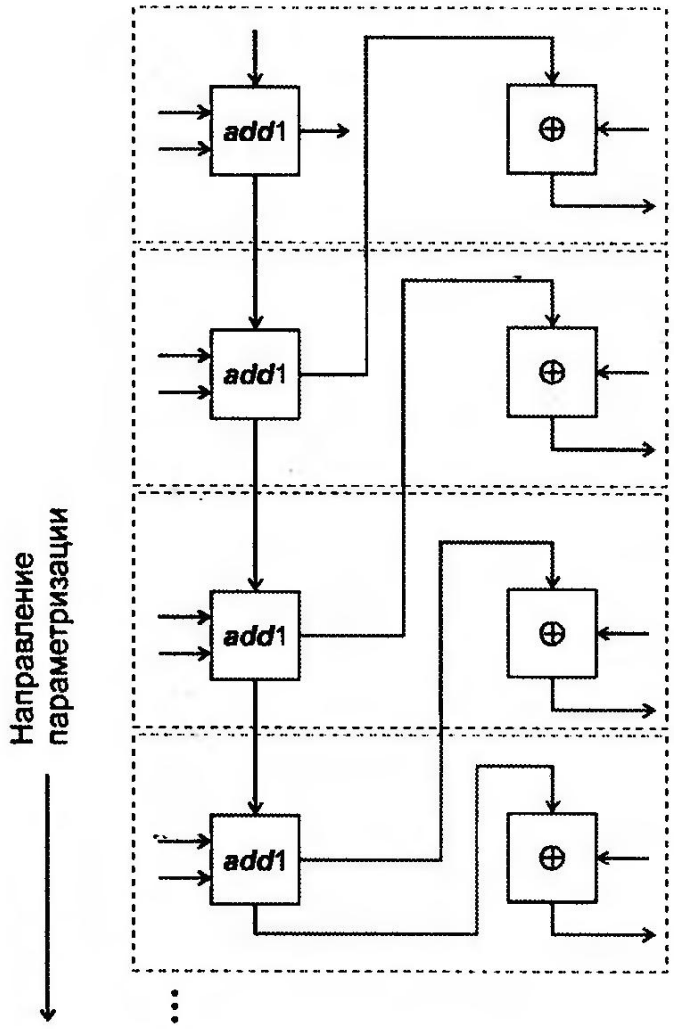


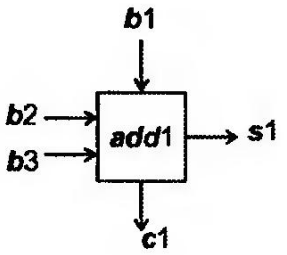
Умножитель

(*a*2, *a*1)×(*b*2, *b*1) = (*p*4, *p*3, *p*2, *p*1)

*a*2, *b*2, *p*4 - старшие разряды

Вариант 15

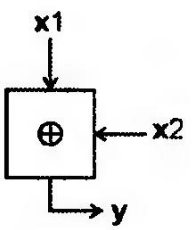




Полусумматор

(*b*1) + (*b*2) + (*b*3) = (*c*1, *s*1)

*s*1 - сумма, *c*1 - перенос



y = x1 ⊕ x2

ЛАБОРАТОРНАЯ РАБОТА 4
Описание и моделирование триггеров и конечных автоматов

Задание

1. С использованием девятизначного алфавита STD_LOGIC составить VHDL-модель и провести моделирование триггера двумя способами:

- по логической схеме триггера (структурное описание);
- по таблице функционирования триггера (алгоритмическое описание).

Сравнить результаты моделирования.

2. С использованием перечислимых типов составить VHDL-модель конечного автомата и провести моделирование двумя способами:

- с помощью моделирующей программы;
- с помощью скрипта (в этом случае предполагается использование системы моделирования ModelSim).

Обозначения в таблицах функционирования триггеров

Через «–» обозначено *любое* из (0, 1) значение сигнала; в квадратных скобках показывается *изменение* сигнала (обычно синхросигнала), например, через [01] обозначается передний фронт сигнала (сигнал меняется из 0 в 1); через [10] обозначается задний фронт сигнала (сигнал меняется из 1 в 0); символ N обозначает *неизменяемое* (предыдущее) значение сигнала; символ ^ является знаком инверсии (отрицания).

Рекомендуемый порядок выполнения лабораторной работы

1. Составить структурное (с использованием оператора **port map**) описание триггера. Использовать задержки логических элементов 2 ns – 5 ns.

2. Составить алгоритмическое описание. Использовать оператор **process** и соответствующие задержки сигналов.

3. Составить тестирующую программу.

4. Провести моделирование обоих описаний в одной тестирующей программе.

5. Сравнить результаты моделирования.

6. Составить описание конечного автомата.

7. Составить тестирующую программу для моделирования конечного автомата и провести моделирование.

8. Составить скрипт и провести моделирование автомата с помощью скрипта (без тестирующей программы).

Требования к оформлению отчета

1. В отчете должна быть нарисована логическая **схема** триггера. При этом обозначения сигналов, элементов схемы должны ***соответствовать*** описанию на языке VHDL.

2. В отчете должен содержаться **VHDL-код** структурной схемы триггера и **VHDL-код** алгоритмического описания триггера.

4. VHDL-код и тестирующая программа должны быть в **отдельных** файлах и содержать **комментарии**:

- автор разработанной VHDL-модели;
- литературный источник;
- номер варианта;

4. В отчете должны содержаться **временные диаграммы**, соответствующие тестирующей программе.

5. В отчете должна содержаться **VHDL-модель** конечного автомата.

6. В отчете должна быть **тестирующая программа** для конечного автомата.

7. В отчете должен быть **скрипт** для моделирования конечного автомата.

Вариант 1

1. Триггер: двухступенчатый D-триггер со сбросом (рис. 5.1).

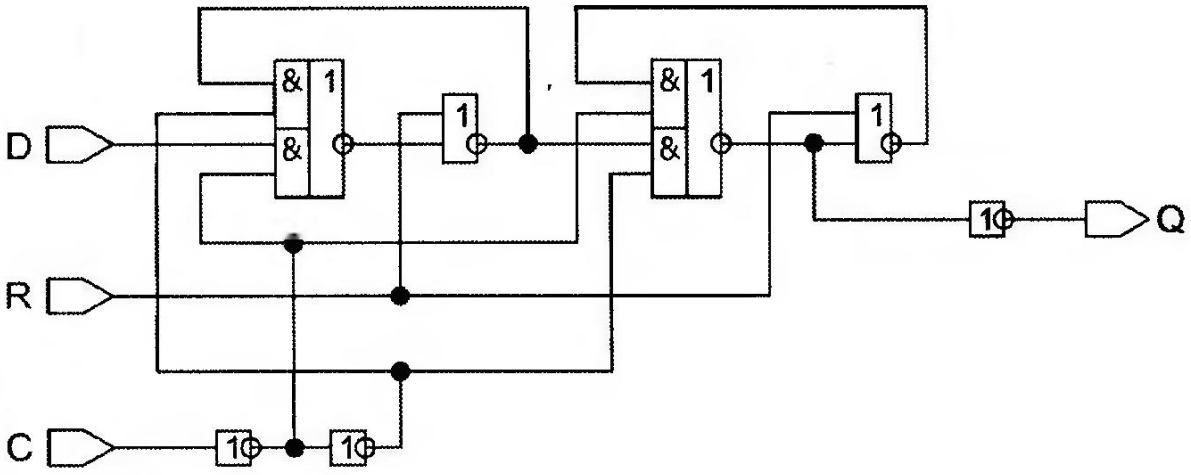


Рис. 5.1. Двухступенчатый D-триггер со сбросом

Таблица функционирования
двухступенчатого D-триггера со сбросом

R	D	C	Q
1	-	[--]	0
0	-	[01]	(D)
0	-	[1–]	N
0	-	[00]	N

2. Конечный автомат Мили.

Алфавит внутренних состояний A = $\{a_1, a_2, a_3, a_4\}$. Начальное состояние автомата a_1. Входной алфавит Z = $\{z_1, z_2, z_3\}$. Выходной алфавит W = $\{w_1, w_2, w_3, w_4, w_5\}$.

Таблица функционирования
конечного автомата Мили

Входные сигналы	Состояния			
	a_1	a_2	a_3	a_4
z_1	a_2/w_1	a_2/w_1	a_1/w_2	a_1/w_4
z_2	a_4/w_5	a_3/w_3	a_4/w_4	a_3/w_5
z_3	a_3/w_2	a_3/w_3	a_1/w_4	a_3/w_5

Вариант 2

1. Триггер: двухступенчатый D-триггер с установкой (рис. 5.2).

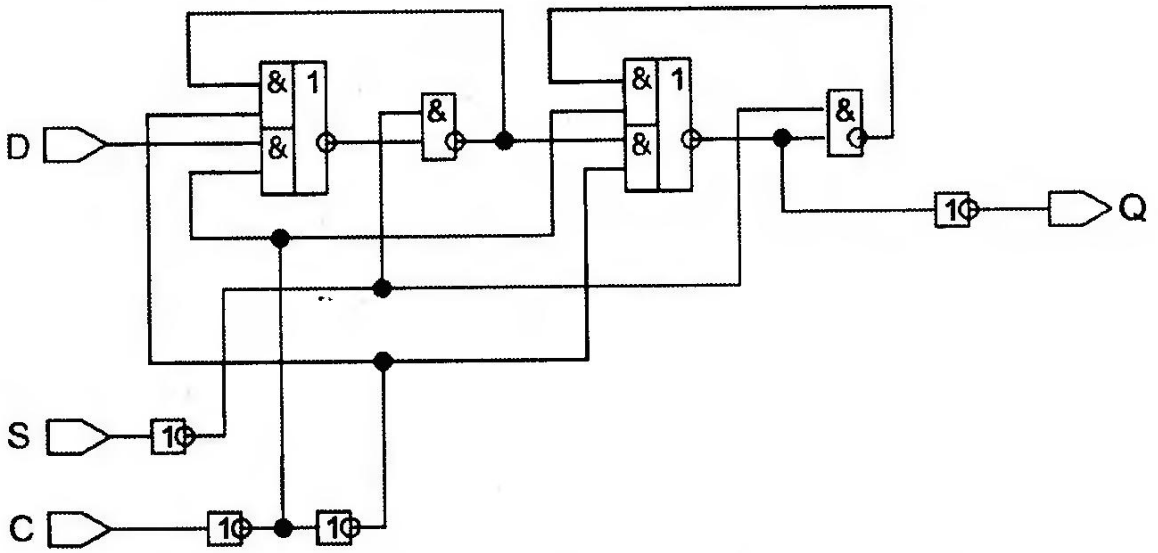


Рис. 5.2. Двухступенчатый D-триггер с установкой

Таблица функционирования
двухступенчатого D-триггера с установкой

S	D	C	Q
1	–	[--]	1
0	–	[01]	(D)
0	–	[1–]	N
0	–	[00]	N

2. Конечный автомат Мили.

Алфавит внутренних состояний A = {a_1, a_2, a_3, a_4}. Начальное состояние автомата a_1. Входной алфавит Z = {z_1, z_2, z_3}. Выходной алфавит W = {w_1, w_2 ,w_3 ,w_4, w_5}.

Таблица функционирования
конечного автомата Мили

Входные сигналы	Состояния			
	a_1	a_2	a_3	a_4
z_1	a_3/w_4	a_2/w_1	a_1/w_2	a_1/w_4
z_2	a_4/w_5	a_2/w_3	a_4/w_3	a_3/w_3
z_3	a_3/w_5	a_3/w_4	a_1/w_4	a_3/w_1

Вариант 3

1. Триггер: двухступенчатый D-триггер с инверсным выходом (рис. 5.3).

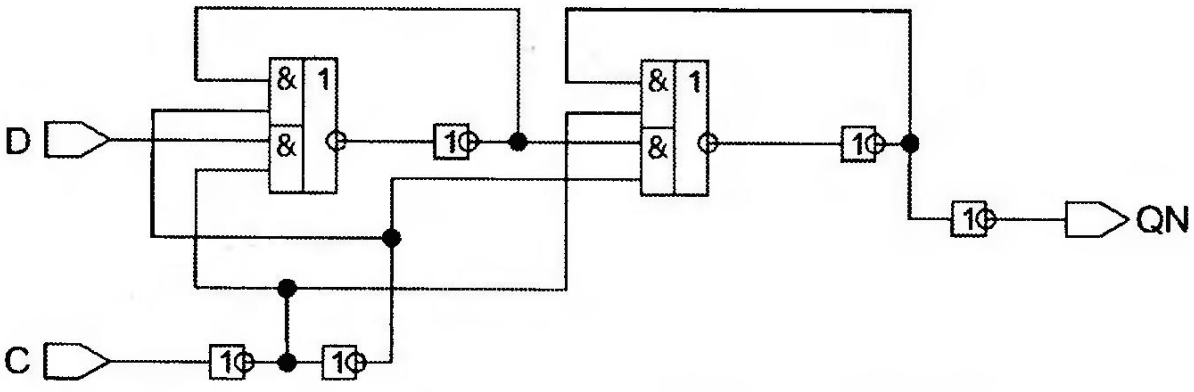


Рис. 5.3. Двухступенчатый D-триггер с инверсным выходом

Таблица функционирования
двухступенчатого D-триггера с инверсным выходом

D	C	QN
–	[01]	^(D)
–	[1–]	N
–	[00]	N

2. Конечный автомат Мили.

Алфавит внутренних состояний A = {a_1, a_2, a_3, a_4}. Начальное состояние автомата a_1. Входной алфавит Z = {z_1, z_2, z_3}. Выходной алфавит W = {w_1, w_2 ,w_3 ,w_4, w_5}.

Таблица функционирования
конечного автомата Мили

Входные сигналы	Состояния			
	a_1	a_2	a_3	a_4
z_1	a_4/w_4	a_2/w_1	a_1/w_2	a_4/w_4
z_2	a_4/w_5	a_1/w_3	a_3/w_4	a_3/w_3
z_3	a_1/w_2	a_2/w_4	a_1/w_4	a_4/w_1

Вариант 4

1. Триггер: D-триггер (рис. 5.4).

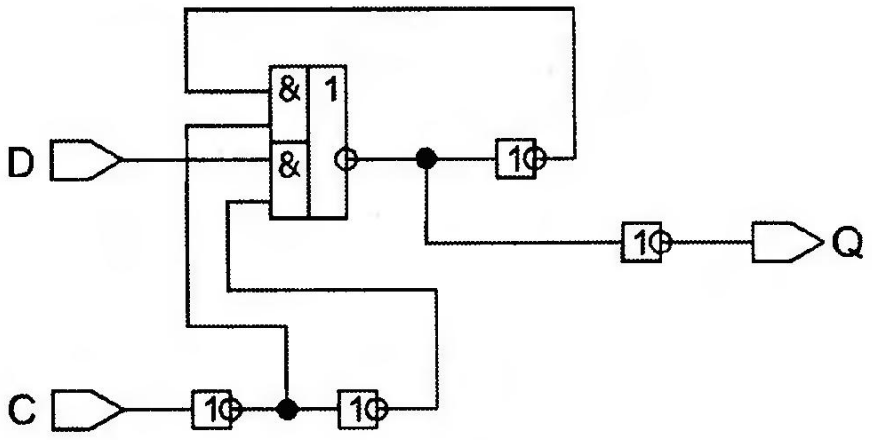


Рис. 5.4. D-триггер

Таблица функционирования
D-триггера

D	C	Q
–	1	(D)
–	0	N

2. Конечный автомат Мили.

Алфавит внутренних состояний A = {a_1, a_2, a_3, a_4}. Начальное состояние автомата a_1. Входной алфавит Z = {z_1, z_2, z_3}. Выходной алфавит W = {w_1, w_2 ,w_3 ,w_4, w_5}.

Таблица функционирования
конечного автомата Мили

Входные сигналы	Состояния			
	a_1	a_2	a_3	a_4
z_1	a_1/w_4	a_3/w_1	a_1/w_3	a_1/w_4
z_2	a_2/w_5	a_1/w_3	a_4/w_3	a_3/w_3
z_3	a_3/w_5	a_4/w_5	a_1/w_4	a_4/w_1

Вариант 5

1. Триггер: D-триггер со сбросом (рис. 5.5).

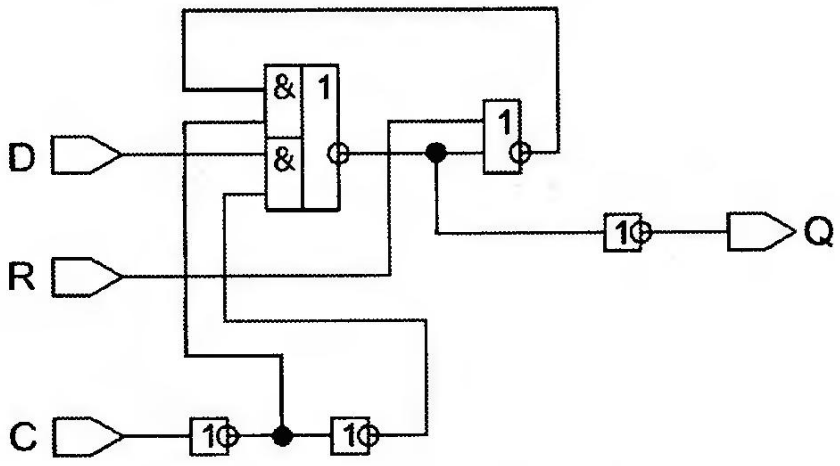


Рис. 5.5. D-триггер со сбросом

Таблица функционирования
D-триггера со сбросом

R	D	C	Q
1	–	0	0
0	–	1	(D)
0	–	0	N

2. **Конечный автомат Мили.**

Алфавит внутренних состояний A = {a_{11}, a_2, a_3, a_4}. Начальное состояние автомата a_1. Входной алфавит Z = {z_1, z_2, z_3}. Выходной алфавит W = {w_1, w_2 ,w_3 ,w_{44}, w_5}.

Таблица функционирования
конечного автомата Мили

Входные сигналы	Состояния			
	a_{11}	a_2	a_3	a_4
z_1	a_3/w_5	a_2/w_1	a_2/w_2	a_{11}/w_5
z_2	a_4/w_{44}	a_2/w_5	a_4/w_3	a_3/w_3
z_3	a_3/w_5	a_{11}/w_{44}	a_{11}/w_{44}	a_{11}/w_5

Вариант 6

1. Триггер: D-триггер с установкой (рис. 5.6).

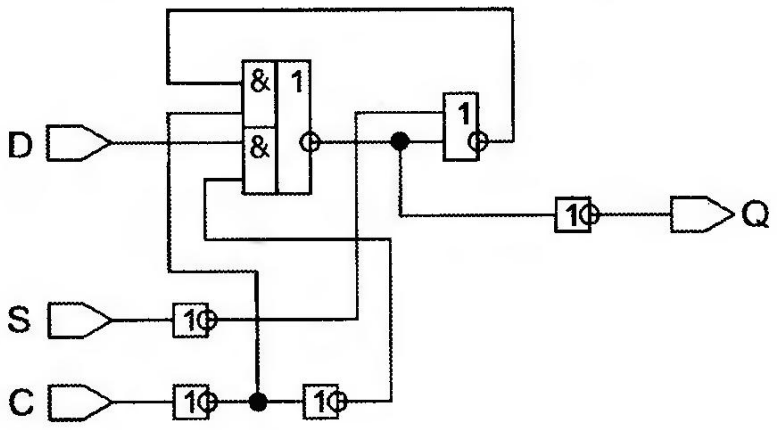


Рис. 5.6. D-триггер с установкой

Таблица функционирования
D-триггера с установкой

S	D	C	Q
1	–	0	1
0	–	1	(D)
0	–	0	N

2. Конечный автомат Мили.

Алфавит состояний A = {a_1, a_{22}, a_3, a_4}. Начальное состояние автомата a_1. Входной алфавит Z = {z_1 , z_2 , z_3}. Выходной алфавит W = {w_1, w_2 ,w_{33} ,w_4, w_5}.

Таблица функционирования
конечного автомата Мили

Входные сигналы	Состояния			
	a_1	a_{22}	a_3	a_4
z_1	a_3/w_4	a_{22}/w_1	a_1/w_2	a_1/w_4
z_2	a_4/w_4	a_{22}/w_{33}	a_4/w_{33}	a_{22}/w_{33}
z_3	a_3/w_5	a_1/w_4	a_1/w_4	a_3/w_1

Вариант 7

1. Триггер: D-триггер со сбросом и установкой (рис. 5.7).

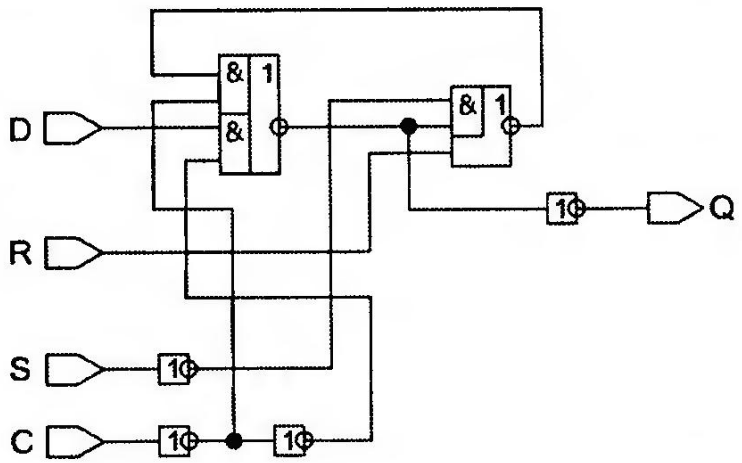


Рис. 5.7. D-триггер со сбросом и установкой

Таблица функционирования
D-триггера со сбросом и установкой

R	S	D	C	Q
1	–	–	0	0
0	1	–	0	1
0	0	–	1	(D)
0	0	–	0	N

2. **Конечный автомат Мили.**

Алфавит состояний A = {a_1, a_{22}, a_3, a_4}. Начальное состояние автомата a_1. Входной алфавит Z = {z_1, z_2, z_{33}}. Выходной алфавит W = {w_1, w_{22} ,w_3 ,w_{45}, w_5}.

Таблица функционирования
конечного автомата Мили

Входные сигналы	Состояния			
	a_1	a_{22}	a_3	a_4
z_1	a_3 /w_{22}	a_{22} /w_1	a_1 /w_1	a_1 /w_{45}
z_2	a_4 /w_5	a_4 /w_3	a_4 /w_{22}	a_3 /w_3
z_{33}	a_3 /w_{22}	a_3 /w_{45}	a_{22} /w_{45}	a_3 /w_1

Вариант 8

1. Триггер: D-триггер с инверсным выходом (рис. 5.8).

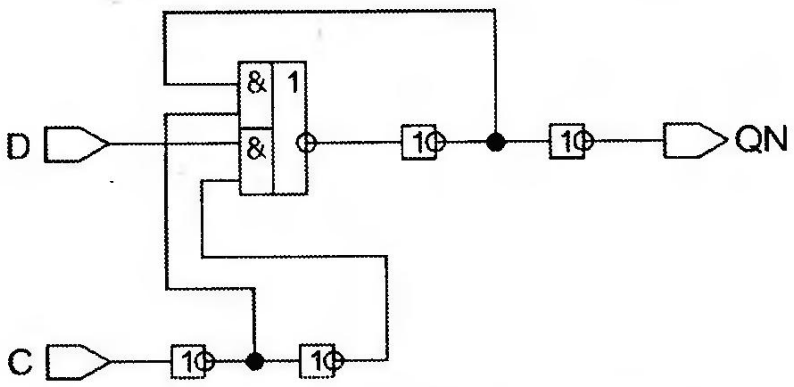


Рис. 5.8. D-триггер с инверсным выходом

Таблица функционирования
D-триггера с инверсным выходом

D	C	QN
–	1	^(D)
–	0	N

2. Конечный автомат Мили.

Алфавит состояний A = {a_{11}, a_2, a_3, a_{44}}. Начальное состояние автомата a_1. Входной алфавит Z = {z_1, z_2, z_3}. Выходной алфавит W = {w_1, w_2 ,w_3 ,w_{44}, w_5}.

Таблица функционирования конечного автомата Мили				
Входные сигналы	Состояния			
	a_{11}	a_2	a_3	a_{44}
z_1	a_3 /w_{44}	a_2 /w_1	a_{11} /w_2	a_{11} /w_{44}
z_2	a_{44} /w_5	a_2 /w_5	a_{44} /w_3	a_3 /w_2
z_3	a_3 /w_2	a_{44} /w_3	a_{11} /w_{44}	a_3 /w_1

Вариант 9

1. Триггер: D-триггер с инверсным синхросигналом (рис. 5.9).

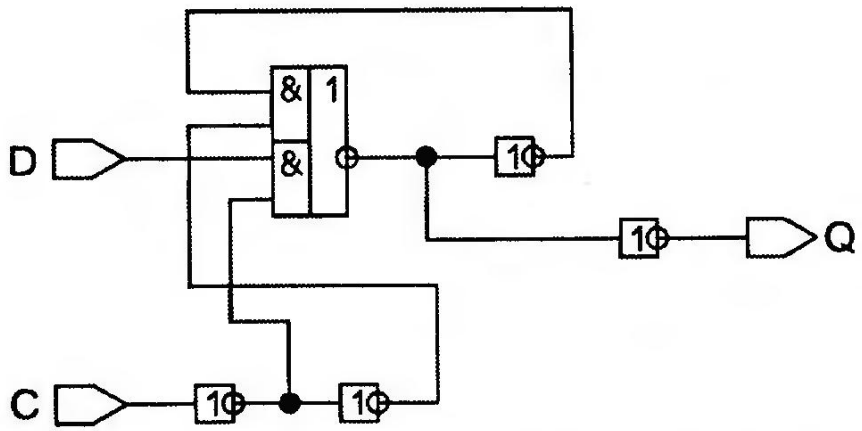


Рис. 5.9. D-триггер с инверсным синхросигналом

Таблица функционирования
D-триггера с инверсным синхросигналом

D	C	Q
–	0	(D)
–	1	N

2. Конечный автомат Мура.

Алфавит состояний A = $\{a_{11}, a_2, a_3\}$. Начальное состояние автомата a_1. Входной алфавит Z = $\{z_1, z_{22}, z_3\}$. Выходной алфавит W = = $\{w_{11}, w_{22}, w_{33}\}$.

Таблица функционирования
конечного автомата Мура

Входные сигналы	Состояния			
	a_{11}	a_2	a_3	
z_1	a_3	a_2	a_{11}	
z_{22}	a_{11}	a_2	a_3	
z_3	a_3	a_{11}	a_3	
	w_{33}	w_{11}	w_{22}	Выходные сигналы

Вариант 10

1. Триггер: D-триггер со сбросом и инверсным синхросигналом (рис. 5.10).

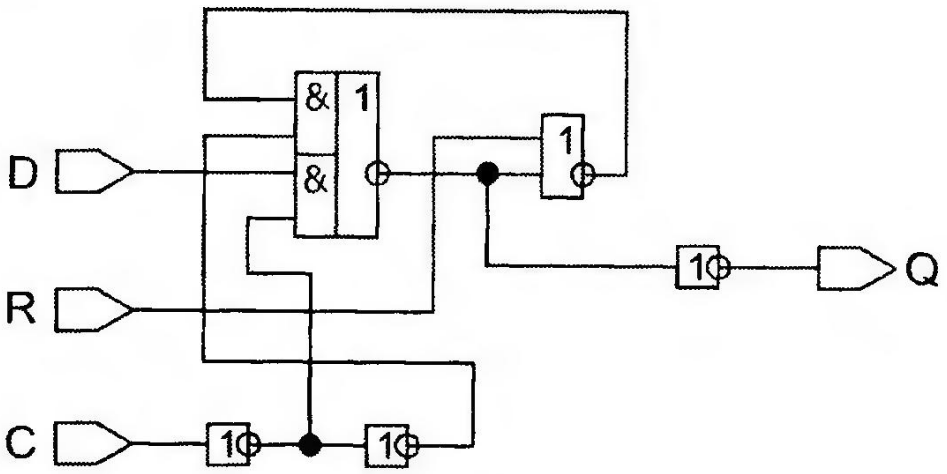


Рис. 5.10. D-триггер со сбросом и инверсным синхросигналом

Таблица функционирования
D-триггера со сбросом и инверсным синхросигналом

R	D	C	Q
1	–	1	0
0	–	0	(D)
0	–	1	N

2. Конечный автомат Мура.

Алфавит состояний B = {b_1, b_{22}, b_3}. Начальное состояние автомата b_1. Входной алфавит Q = {q_1, q_{22}, q_3}. Выходной алфавит Y = = {y_1, y_2 ,y_3}.

Таблица функционирования
конечного автомата Мура

Входные сигналы	Состояния			
	b_1	b_{22}	b_3	
q_1	b_{22}	b_1	b_1	
q_{22}	b_3	b_{22}	b_{22}	
q_3	b_1	b_3	b_3	
	y_2	y_3	y_1	Выходные сигналы

Вариант 11

1. Триггер: Двухступенчатый Т-триггер (рис. 5.11).

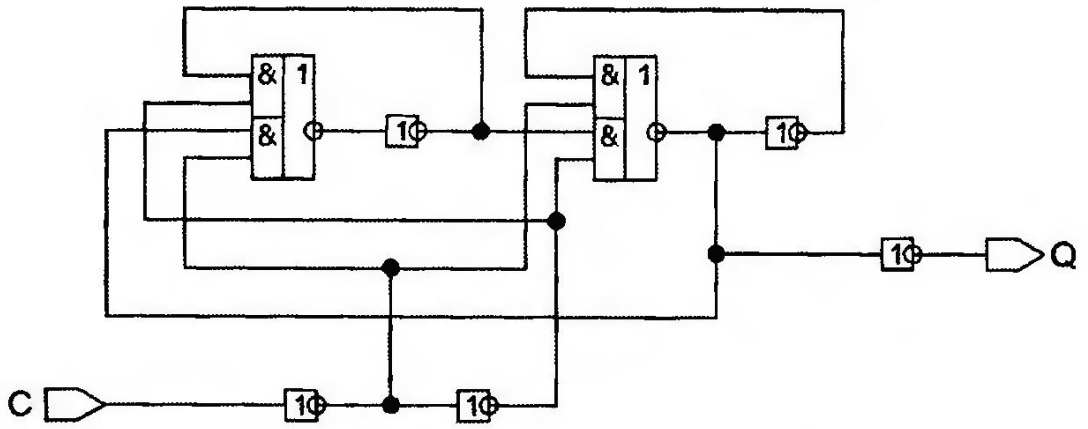


Рис. 5.11. Двухступенчатый Т-триггер

Таблица функционирования
двухступенчатого Т-триггера

C	Q
[01]	^(Q)
[1–]	N
[00]	N

2. Конечный автомат Мура.

Алфавит состояний $C = \{c_1, c_2, c_{33}\}$. Начальное состояние автомата c_1. Входной алфавит $Q = \{q_1, q_2, q_3\}$. Выходной алфавит $Y = \{y_{11}, y_2, y_3\}$.

Таблица функционирования
конечного автомата Мура

Входные сигналы	Состояния			
	c_1	c_2	c_{33}	
q_1	c_2	c_2	c_1	
q_2	c_2	c_1	c_2	
q_3	c_{33}	c_{33}	c_{33}	
	y_{11}	y_3	y_2	Выходные сигналы

Вариант 12

1. Триггер: двухступенчатый Т-триггер с установкой (рис. 5.12).

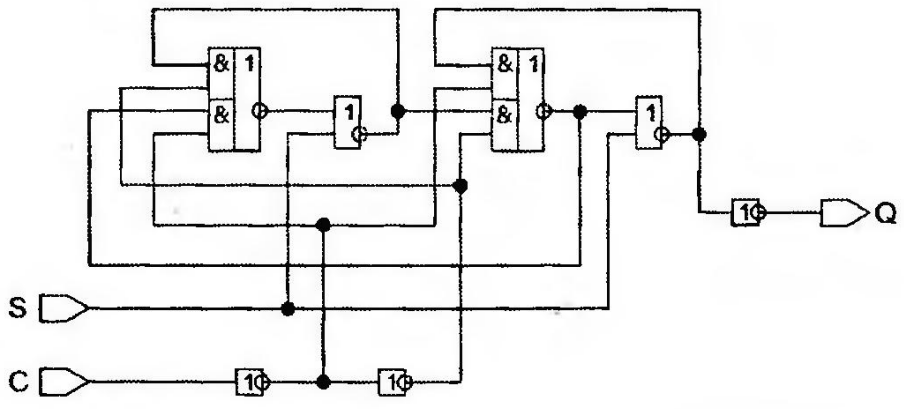


Рис. 5.12. Двухступенчатый Т-триггер с установкой

Таблица функционирования
двухступенчатого Т-триггера с установкой

S	C	Q
1	[--]	1
0	[01]	^(Q)
0	[1–]	N
0	[00]	N

2. Конечный автомат Мура.

Алфавит состояний R = {r_1, r_2, r_3}. Начальное состояние автомата r_1. Входной алфавит Q = {q_1, q_2}. Выходной алфавит W = {w_1, w_2 ,w_3}.

Таблица функционирования
конечного автомата Мура

Входные сигналы	Состояния			
	r_1	r_2	r_3	
q_1	r_1	r_3	r_1	
q_2	r_3	r_2	r_3	
	w_3	w_1	w_2	Выходные сигналы

Вариант 13

1. Триггер: RS-триггер (рис. 5.13).

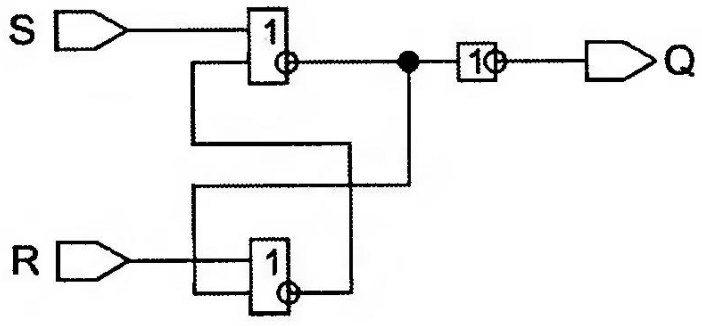


Рис. 5.13. RS-триггер

Таблица функционирования
RS-триггера

R	S	Q
0	0	N
-	1	1
1	0	0

2. Конечный автомат Мура.

Алфавит состояний T = {t_1, t_{22}, t_3}. Начальное состояние автомата t_1. Входной алфавит Q = {q_{11}, q_2, q_3}. Выходной алфавит Y = = {y_1, y_2 ,y_3}.

Таблица функционирования
конечного автомата Мура

Входные сигналы	Состояния			
	t_1	t_{22}	t_3	
q_{11}	t_3	t_{22}	t_1	
q_2	t_{22}	t_1	t_{22}	
q_3	t_1	t_{22}	t_3	
	y_3	y_2	y_1	Выходные сигналы

Вариант 14

1. Триггер: RS-триггер с инверсным выходом (рис. 5.14).

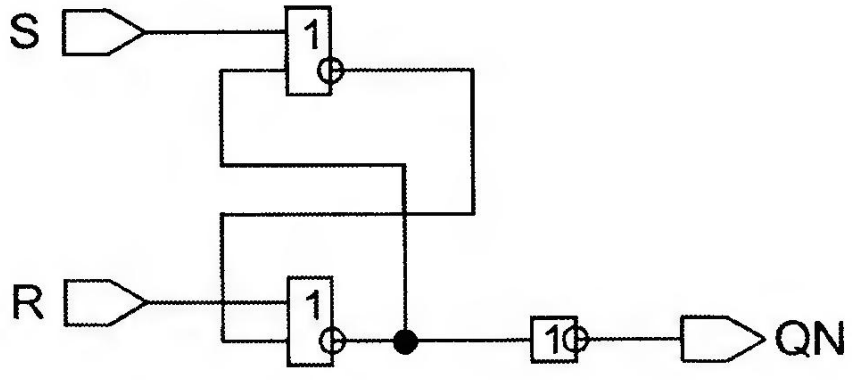


Рис. 5.14. RS-триггер с инверсным выходом

Таблица функционирования
RS-триггера с инверсным выходом

R	S	QN
0	0	N
0	1	0
1	–	1

2. Конечный автомат Мура.

Алфавит состояний S = {s_{11}, s_{22}, s_3}. Начальное состояние автомата s_1. Входной алфавит Z = {q_1, q_2, q_{33}}. Выходной алфавит Y = = {y_1, y_2 ,y_3}.

Таблица функционирования
конечного автомата Мура

Входные сигналы	Состояния			
	s_{11}	s_{22}	s_3	
q_1	s_{11}	s_{11}	s_{22}	
q_2	s_3	s_{11}	s_{22}	
q_{33}	s_{22}	s_{22}	s_3	
	y_2	y_3	y_1	Выходные сигналы

Вариант 15

1. Триггер: двухступенчатый D-триггер (рис. 5.15).

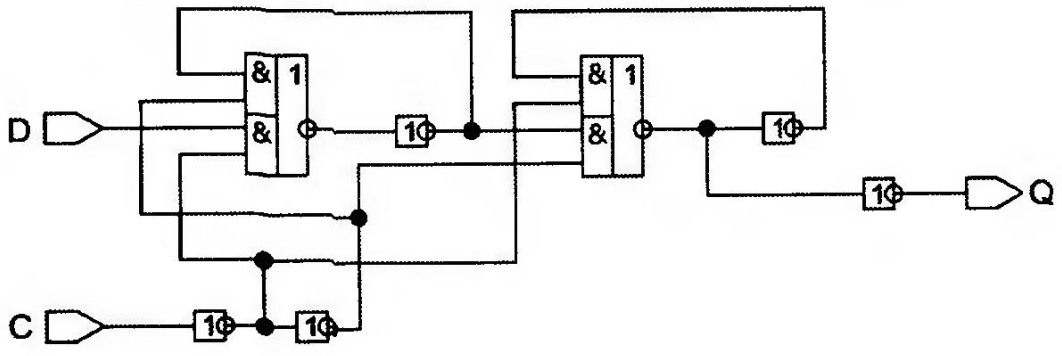


Рис. 5.15. Двухступенчатый D-триггер

Таблица функционирования
двухступенчатого D-триггера

D	C	Q
–	[01]	(D)
–	[1–]	N
–	[00]	N

2. Конечный автомат Мура.

Алфавит состояний $R = \{r_1, r_2, r_3\}$. Начальное состояние автомата r_1. Входной алфавит $Q = \{q_1, q_2, q_3\}$. Выходной алфавит $W = \{w_1, w_2, w_3\}$.

Таблица функционирования
конечного автомата Мура

Входные сигналы	Состояния			
	r_1	r_2	r_3	
q_1	r_1	r_3	r_1	
q_2	r_3	r_2	r_3	
q_3	r_2	r_2	r_2	
	w_3	w_1	w_2	Выходные сигналы

ЛАБОРАТОРНАЯ РАБОТА 5
Функции и процедуры

Задание

Написать на языке VHDL требуемые функции и процедуры, провести их моделирование. Требуемые собственные типы данных определить в пакете.

Рекомендуемый порядок выполнения лабораторной работы

1. Разработать математический алгоритм решения задачи.
2. Определить требуемые типы данных в пакете.
3. Изучить отличия оформления функций и процедур в языке VHDL.
4. Написать тексты функций и процедур.

5. Написать тестирующую программу.

6. Провести моделирование и получить временные диаграммы.

Требования к оформлению отчета по лабораторной работе 5

1. В отчете должно быть приведено исходное задание и разработанные **VHDL-программы.**

2. В отчете должна содержаться **тестирующая программа.**

3. В отчете должен содержаться VHDL-пакет (в отдельном файле).

4. В отчете должны содержаться **временные диаграммы,** соответствующие тестирующей программе.

Вариант 1

Написать VHDL-коды и провести тестирование двух функций и двух процедур.

1. Функция (процедура) преобразования двоичного представления числа в десятичное представление. Это функция (процедура) преобразования типа BIT_VECTOR в тип INTEGER.

Первый разряд двоичного представления является старшим. Число разрядов бит-вектора равно 10.

2. Функция (процедура) преобразования двоичного представления числа в десятичное представление. Это функция (процедура) преобразования типа BIT_VECTOR в тип INTEGER.

Последний (правый) разряд двоичного представления является старшим. Число разрядов бит-вектора равно 10.

Вариант 2

Написать VHDL-коды и провести тестирование двух функций и двух процедур.

1. Функция (процедура) преобразования двоичного представления числа в восьмеричное представление. Это функция (процедура) преобразования типа BIT_VECTOR в тип INTEGER.

Первый разряд двоичного представления является старшим. Число разрядов бит-вектора равно 12.

2. Функция (процедура) преобразования двоичного представления числа в восьмеричное представление. Это функция (процедура) преобразования типа BIT_VECTOR в тип INTEGER.

Последний (правый) разряд двоичного представления является старшим. Число разрядов бит-вектора равно 12.

Вариант 3

Написать VHDL-коды и провести тестирование двух функций.

1. Функция преобразования шестнадцатеричного представления числа в восьмеричное представление.

Число разрядов в восьмеричном представлении равно 4.

2. Функция преобразования восьмеричного представления числа в шестнадцатеричное представление.

Число разрядов в восьмеричном представлении равно 4.

Вариант 4

Написать VHDL-коды и провести тестирование двух функций и двух процедур.

1. Функция (процедура) преобразования двоичного представления числа в дополнительный код.

Число разрядов в двоичном представлении числа равно 8.

2. Функция (процедура) преобразования десятичного представления числа в шестнадцатеричное представление.

Число разрядов в десятичном представлении числа равно 3.

Вариант 5

Написать VHDL-коды и провести тестирование двух функций и двух процедур.

1. Функция (процедура) преобразования двоично-десятичного кода числа в двоичный.

Первый (левый) разряд двоичного представления числа является старшим. Число разрядов в двоичном представлении числа равно 8.

2. Функция (процедура) преобразования двоично-десятичного кода числа в двоичный.

Последний (правый) разряд двоичного представления числа является старшим. Число разрядов в двоичном представлении числа равно 8.

Вариант 6

1. Написать и провести тестирование **функции и процедуры** нахождения максимального m_{max} и минимального m_{min} элемента в одноразмерном массиве M целых положительных чисел. Выдать элементы m_{max}, m_{min} и номера их позиций в массиве.

2. Написать и провести тестирование **функции и процедуры** нахождения минимального элемента m_{min} в столбце j двухразмерного массива (матрицы) M, элементами которого являются целые положительные числа, выбираемые из множества {0, 1, 2, ..., 100}. Выдать элемент m_{min} и номер строки i, в которой он находится. В пакете определить соответствующий тип для элементов матрицы.

Вариант 7

1. Написать и провести тестирование **функции и процедуры** транспонирования булевой (двоичной) матрицы. Соответствующий тип матрицы определить в пакете.

2. Написать и провести тестирование **функции и процедуры** транспонирования матрицы, элементами которой являются целые положительные числа, выбираемые из множества {0, 1, 2, ..., 100}.

В пакете определить соответствующий тип для элементов матрицы. Тип матрицы также определить в пакете.

Вариант 8

Разработать в виде функции и процедуры алгоритмическое описание системы, осуществляющей перемножение ($C = A \times B$) двух матриц A, B размерности $N \times N$, элементами которых являются целые положительные числа, выбираемые из множества {0, 1, 2, ..., 100}.

Входные порты системы — матрицы A, B.

Выходной порт системы — матрица C.

В пакете определить соответствующий тип для матриц A, B и соответствующий тип для матрицы C. В пакете определить также соответствующий тип для элементов матриц A, B, C.

Вариант 9

Написать и провести тестирование **функции и процедуры** поиска максимального элемента m_{ij} в заданном столбце j двухразмерного массива (матрицы) M, элементами которого являются натуральные числа, выбираемые из множества {0, 1, 2, ..., 100}.

Выходные данные для процедуры: элемент m_{ij}, номер i строки и номер j столбца найденного элемента m_{ij}. Соответствующий тип матрицы определить в пакете. Соответствующий тип для элементов матрицы также определить в пакете.

Вариант 10

Написать и провести тестирование одной **функции** и одной **процедуры** упорядочения по **возрастанию** элементов массива. Элементами массива являются натуральные числа, выбираемые из множества {0, 1, 2, ..., 100}. Для каждого элемента упорядоченного массива выдать номер позиции в исходном массиве.

В пакете определить соответствующий тип для элементов матрицы. Тип массива также определить в пакете.

Вариант 11

Написать и провести тестирование одной **функции** и одной **процедуры** поиска такого столбца j в двухразмерном массиве (матрице), сумма элементов которого является **максимальной**.

Написать и провести тестирование одной **функции** и одной **процедуры** поиска такого столбца j в двухразмерном массиве (матрице), сумма элементов которого является **минимальной**.

Элементами массива являются натуральные числа, выбираемые из множества {0, 1, 2, ..., 100}. В пакете определить соответствующий тип для элементов массива (матрицы). Тип массива также определить в пакете.

Вариант 12

Написать и провести тестирование двух функций и двух процедур.

1. Функция (процедура) поиска такой строки *j* двухразмерного массива (матрицы), сумма элементов которой (строки) является **максимальной.**

Элементами массива являются натуральные числа, выбираемые из множества {0, 1, 2, ..., 100}. Выдать номер *j* строки. Выдать сумму элементов найденной строки *j*. В пакете определить соответствующий тип для элементов массива (матрицы). Тип массива также определить в пакете.

2. Функция (процедура) преобразования шестнадцатеричного представления числа в десятичное представление.

Вариант 13

Написать и провести тестирование двух функций и двух процедур.

1. Функция (процедура) нахождения в двоичной матрице B строки *j* с **минимальным** суммарным числом S единиц.

Выдать номер *j* строки и суммарное число S единиц. Тип матрицы размерности (N × M) определить в пакете.

2. Функция (процедура) нахождения в двоичной матрице B столбца *i* с **максимальным** суммарным числом S единиц.

Выдать номер *i* строки и суммарное число S единиц. Тип матрицы размерности (N × N) определить в пакете.

Вариант 14

Написать VHDL-коды и провести тестирование двух функций и двух процедур.

1. Функция (процедура) нахождения в двоичной матрице B строки *j* с **максимальным** суммарным числом S единиц. Выдать номер *j* строки и суммарное число S единиц.

Тип матрицы размерности (N × M) определить в пакете.

2. Функция (процедура) нахождения в двоичной матрице B столбца I с **минимальным** суммарным числом S единиц. Выдать номер I строки и суммарнос число S единиц.

Тип матрицы размерности (N × N) определить в пакете.

Вариант 15

Написать VHDL-коды и провести тестирование двух функций и двух процедур.

1. Функция и процедура транспонирования булевой (двоичной) матрицы размерности N × M.

Соответствующий тип матрицы определить в пакете.

2. Написать и провести тестирование **функции и процедуры** упорядочения по **убыванию** элементов одноразрядного массива M. Элементами массива M являются натуральные числа, выбираемые из множества {0, 1, 2, …, 100}.

Для каждого элемента упорядоченного массива выдать номер позиции в исходном массиве M. Выдать сумму элементов массива M, подсчитав ее с помощью отдельно написанной функции. Тип массива определить в пакете. Тип элементов массива определить в пакете.

5.2. КОНТРОЛЬНАЯ РАБОТА

Вариант 1

Задача 1. Составить VHDL-код, представляющий схему (рис. 5.16), в виде *структурного* описания с компонентом F1 (рис. 5.17). Использовать операторы **port map** при составлении структурного описания. Использовать обозначения входов, выходов компонента F1, приведенного на рис. 5.17.

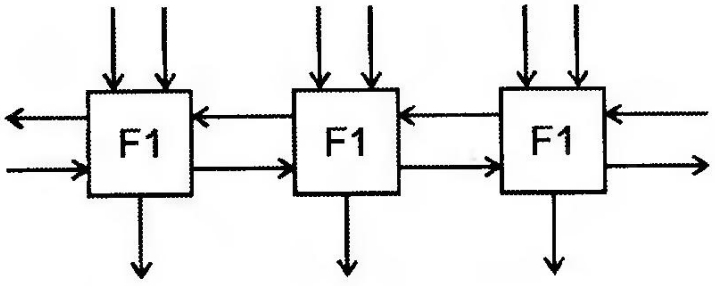


Рис. 5.16. Логическая схема

Задача 2. Составить VHDL-код, описывающий функционирование схемы (рис. 5.16), употребив параллельные операторы процесса и назначения сигнала.

Задача 3. Описать регулярную логическую схему (рис. 5.17), употребив параллельный оператор **generate** и положив, что схема состоит из 32 компонент F1. Использовать обозначения входов, выходов компонента F1 приведенного на рис. 5.17.

Задача 4. Можно ли в языке VHDL проверить предыдущее значение сигнала clock, имеющего тип **integer**?

Задача 5. Могут ли употребляться переменные для передачи информации между процессами? Какие объекты языка VHDL служат для передачи информации между процессами?

Задача 6. Правильно ли, что все операторы внутри процесса выполняются один за другим?

Задача 7. Правильно ли, что все процессы внутри архитектурного тела выполняются один за другим?

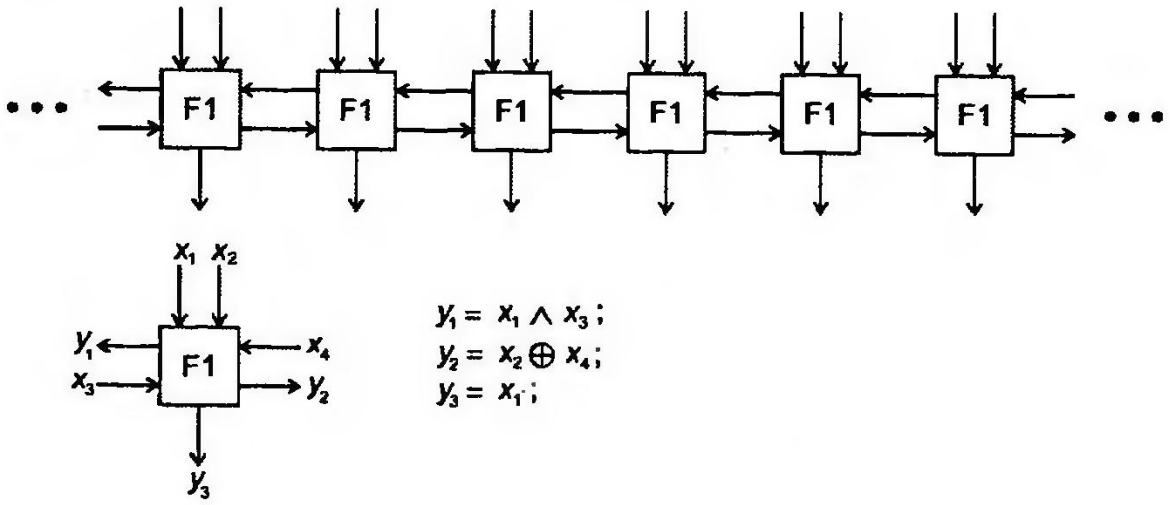


Рис. 5.17. Параметризуемая регулярная логическая схема

Вариант 2

Задача 1. Составить VHDL-код, представляющий схему (рис. 5.18), в виде *структурного* описания с компонентами F1,F2 (рис. 5.19). При составлении структурного описания использовать операторы **port map**. Использовать обозначения входов, выходов компонентов F1, F2, приведенных на рис. 5.19.

Задача 2. Составить VHDL-код, описывающий функцио-нирование схемы (рис. 5.18), употребив параллельные операторы процесса и назначения сигнала.

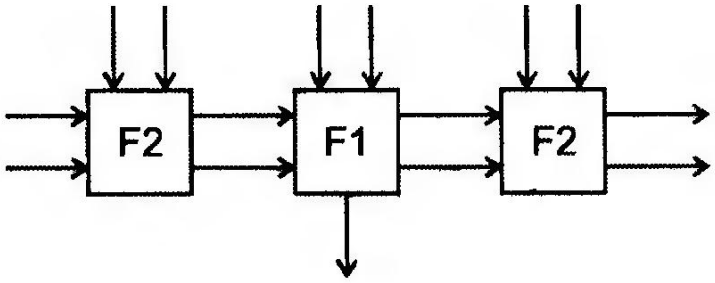


Рис. 5.18. Логическая схема

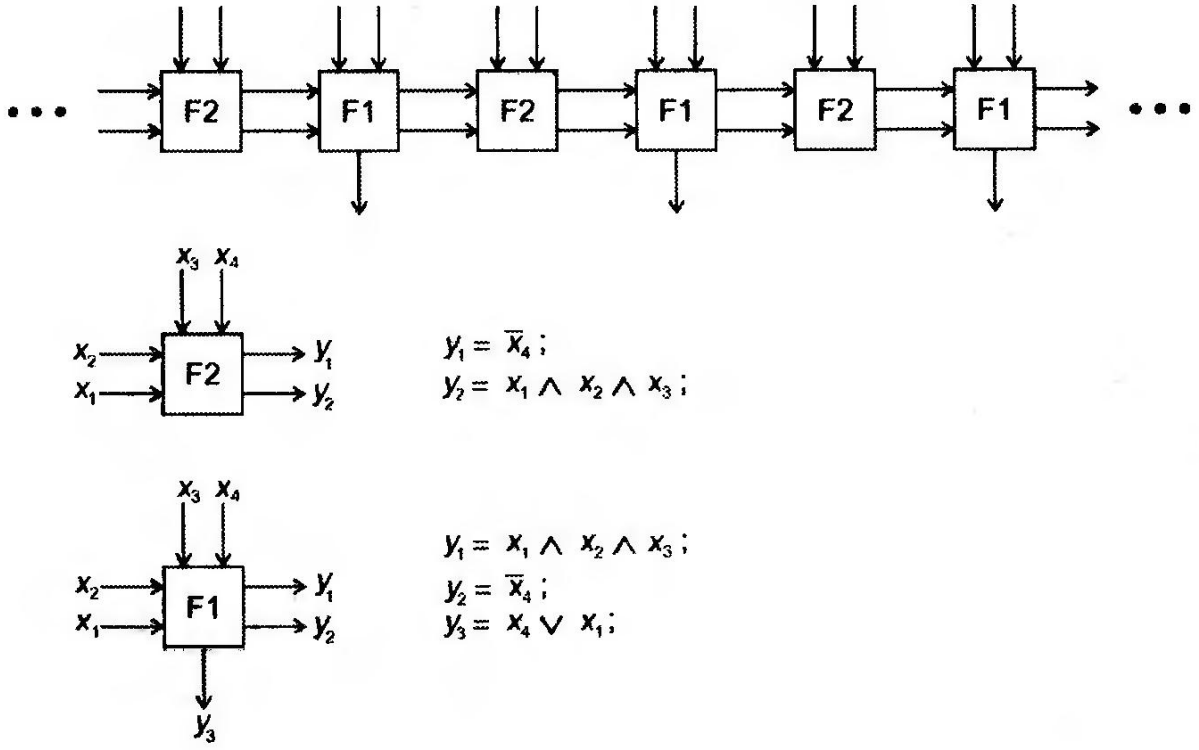


Рис. 5.19. Параметризуемая регулярная логическая схема

Задача 3. Описать регулярную логическую схему (рис. 5.19), употребив параллельный оператор **generate** и положив, что схема состоит из 8 компонент F1 и 8 компонент F2. Использовать обозначения входов, выходов компонентов F1, F2, приведенных на рис. 5.19.

Задача 4. Можно ли декларировать сигналы внутри процессов?

Задача 5. Может ли процесс в языке VHDL иметь список сигналов запуска и операторы **wait** внутри оператора процесса?

Задача 6. Укажите типы данных, возвращаемых операторами сравнения?

Задача 7. Можно ли в языке VHDL узнать (проверить) предыдущее значение переменной типа **integer**?

Вариант 3

Задача 1. Составить VHDL-код, представляющий схему (рис. 5.20), в виде *структурного* описания с компонентами F1, F2 (рис. 5.21). При составлении структурного описания использовать операторы port map. Использовать обозначения входов, выходов компонентов F1, F2, приведенных на рис. 5.21.

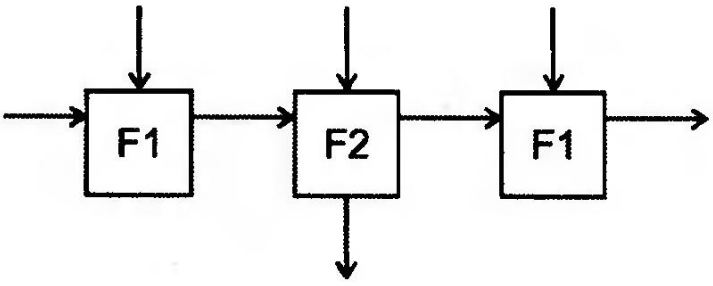


Рис. 5.20. Логическая схема

Задача 2. Составить VHDL-код, описывающий функцио-нирование схемы (рис. 5.20), употребив параллельный операторы процесса и назначения сигнала.

Задача 3. Описать регулярную логическую схему (рис. 5.21), употребив параллельный оператор **generate** и положив, что схема состоит из 10 компонент F1 и 10 компонент F2. Использовать обозначения входов, выходов компонентов F1, F2, приведенных на рис. 5.21.

Задача 4. Результат выполнения какого из трех операторов a), b), c) отличается от двух других?

a) Z <= **not** X **and not** Y; b) Z <= **not** (X **or** Y);

c) Z <= (**not** X) **xor** Y;

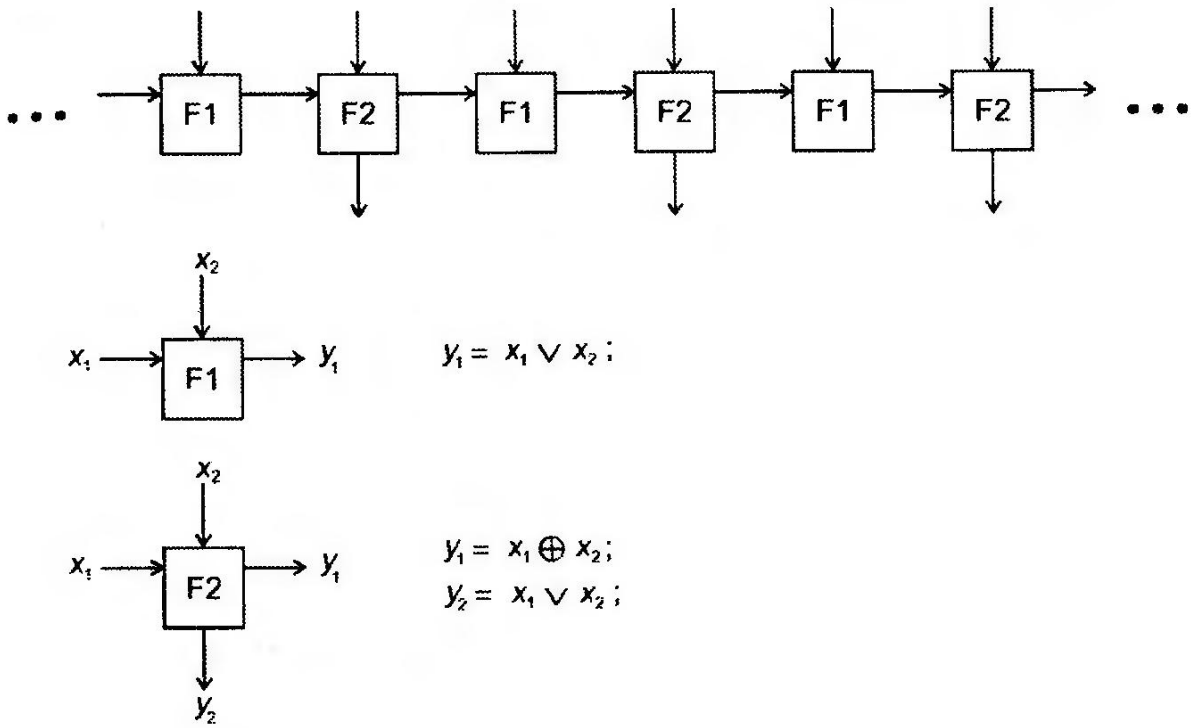


Рис. 5.21. Параметризуемая регулярная логическая схема

Задача 5. В каких разделах VHDL-кода может встретиться:

- оператор назначения сигнала;
- оператор присвоения значения переменной?

Задача 6.. Какая часть VHDL-кода содержит последовательные операторы. Выберите правильный ответ:

a) процесс (перед ключевым словом **begin**);

b) архитектурное тело;

c) процесс (после ключевого слова **begin**);

d) пакет.

Задача 7. Правильно ли, что структурное описание состоит из компонент и сигналов?

Вариант 4

Задача 1. Составить VHDL-код, представляющий схему (рис. 5.22), в виде *структурного* описания с компонентами F1,F2 (рис. 5.23). При составлении структурного описания использовать операторы **port map**. Использовать обозначения входов, выходов компонентов F1, F2, приведенных на рис. 5.23.

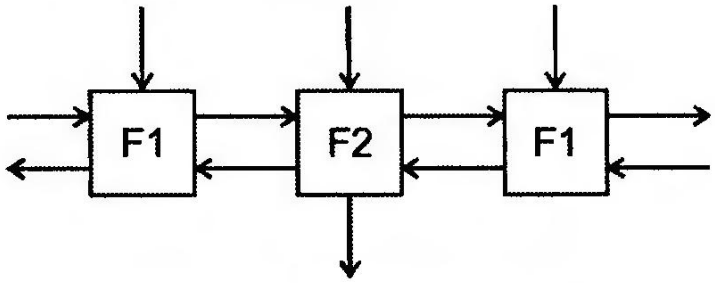


Рис. 5.22. Логическая схема

Задача 2. Составить VHDL-код, описывающий функционирование схемы (рис. 5.22), употребив параллельные операторы процесса и назначения сигнала.

Задача 3. Описать регулярную логическую схему (рис. 5.23), употребив параллельный оператор **generate** и положив, что схема состоит из 10 компонент F1 и 10 компонент F2. Использовать обозначения входов, выходов компонентов F1, F2, приведенных на рис. 5.23.

Задача 4. Укажите правильные и неправильные идентификаторы

a) Decoder_1; b) _Decoder_1; c) 2FFT; d) sig_N; e) Not_Ack; f) Not-Ask; g) Sig_#N; h) FFT

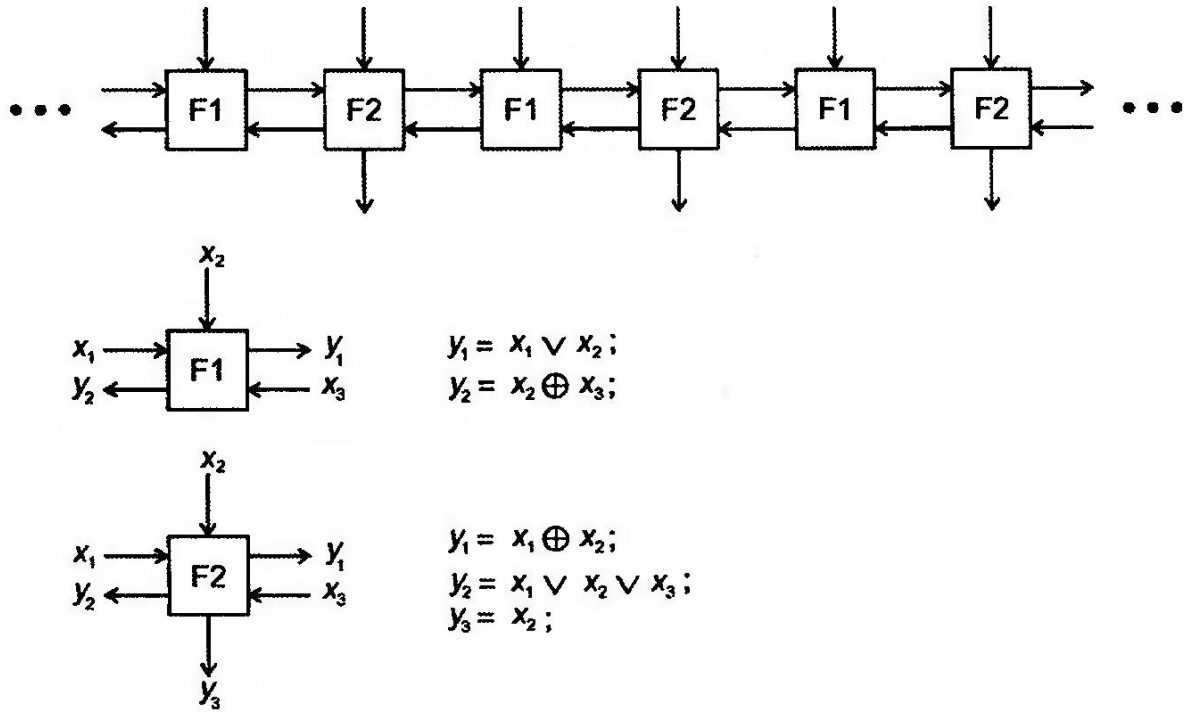


Рис. 5.23. Параметризуемая регулярная логическая схема

Задача 5. Для каждого из утверждений a, b ответьте, является оно правильным или ложным:

a) все элементы массива должны быть одного и того же типа;

b) значение true типа boolean эквивалентно значению 0 типа bit;

Задача 6. В какой части VHDL-кода должны быть декларированы локальные сигналы архитектурного тела? Выберите правильный ответ:

a)в списке портов архитектурного тела;

b) в конфигурации;

c) в архитектурном теле после ключевого слова **begin**;

d) в архитектурном теле перед ключевым словом **begin**.

Задача 7. Какая часть VHDL-кода содержит последовательные операторы. Выберите правильный ответ:

a) процесс (перед ключевым словом **begin**);

b) архитектурное тело;

c) процесс (после ключевого слова **begin**);

d) пакет.

Вариант 5

Задача 1. Составить VHDL-код, представляющий схему (рис. 5.24), в виде *структурного* описания с компонентами F1,F2 (рис. 5.25). При составлении структурного описания использовать операторы **port map**. Использовать обозначения входов, выходов компонентов F1, F2, приведенных на рис. 5.25.

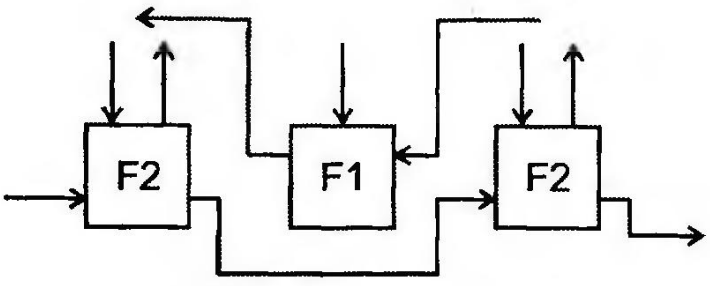


Рис. 5.24. Логическая схема

Задача 2. Составить VHDL-код, описывающий функционирование схемы (рис. 5.24), употребив параллельные операторы процесса и назначения сигнала.

Задача 3. Описать регулярную логическую схему (рис. 5.25), употребив параллельный оператор **generate** и положив, что схема состоит из 10 компонент F1 и 10 компонент F2. Использовать обозначения входов, выходов компонентов F1, F2, приведенных на рис. 5.25.

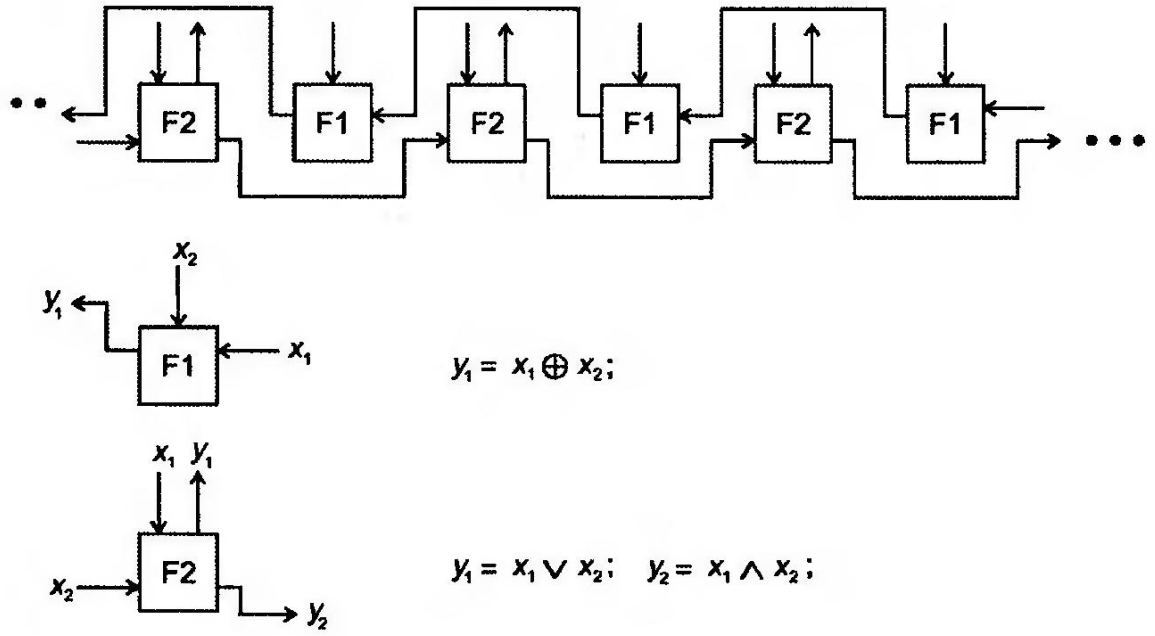


Рис. 5.25. Параметризуемая регулярная логическая схема

Задача 4. Найдите ошибки (выпишите строки) в следующем VHDL-коде:

```
entity EXP is end EXP;
architecture RTL of EXP is
  signal A, B, C, D, E, F : bit;   signal X, Y, Z, S, T, R : bit;
begin
  R <= A and B and C;
  S <= B or C or D;
  Y <= C nor E nor F;
  X <= A and not B and not C;
  Z <= F nand E nand B;
end RTL;
```

Задача 5. В какой части VHDL-кода необходимо указывать тип сигнала, который декларируется? Выберите правильный ответ:

a) когда сигнал декларируется;

b) когда сигнал употребляется первый раз в коде;

c) в пакете;

d) нет необходимости декларировать тип сигнала.

Задача 6. Правильно ли утверждение:

Структурное VHDL-описание может быть иерархичным.

Задача 7. Объясните, как Вы понимаете запись на языке VHDL

```
x <= y <= z ;
```

Является ли она корректной? Почему?

5.3. ТЕСТОВЫЕ (ДА/НЕТ) ВОПРОСЫ

1. Декларация **entity** определяет только имя проектируемой цифровой системы и спецификацию ее портов ввода/вывода?
2. Структурное описание может быть иерархическим?
3. Описание всей проектируемой цифровой системы может быть не иерархическим?
4. Допускается не более восьми уровней иерархии описания VHDL-проекта?
5. Одна декларация **entity** может соответствовать нескольким архитектурным **(architecture)** описаниям?
6. Каждая цифровая система имеет интерфейс (**entity**)?
7. Прежде чем использовать пакет **STANDARD**, декларация **entity** должна быть обязательно предопределена операторами **library**, **use**?
8. Каждая цифровая система, описываемая в VHDL, описывается парой **entity** и **architecture**?
9. Пакет **(package)** — это компактный способ хранения логически связанных описаний **entity** и **architecture**?
10. Одна декларация **architecture** может соответствовать нескольким декларациям **entity**?

11. Конфигурация (**configuration**) является модулем (VHDL-файлом) проекта?
12. Конфигурация является объектом языка VHDL?
13. Пакет является модулем проекта?
14. Пакет является объектом языка VHDL?
15. Сигнал является объектом языка VHDL?
16. Сигнал является модулем проекта?
17. Константа является объектом языка VHDL?
18. Константа является модулем проекта?
19. Переменная является объектом языка VHDL?
20. Переменная является модулем проекта?
21. Имеется только три объекта языка VHDL: сигнал, переменная, константа?
22. Имеется только пять модулей проекта (VHDL-файлов) цифровой системы: **entity, architecture, configuration, package, package body** (тело пакета)?
23. Существует ли возможность узнать предыдущее значение переменной?
24. Порядок нумерации битов в векторе типа bit_vector всегда одинаков?
25. Внутренние сигналы системы определяются в декларации **entity**?
26. Существует ли возможность узнать предыдущее значение сигнала?
27. Выходные сигналы системы могут определяться в разделе деклараций архитектурного тела?
28. Внутренние сигналы системы определяются в разделе деклараций архитектурного тела?
29. Декларация входного сигнала состоит из имени сигнала, типа, направления (**in**)?
30. Декларация выходного сигнала состоит из имени сигнала, типа, направления (**out**)?
31. Декларация внутреннего сигнала состоит из имени сигнала и типа?
32. Все сигналы, описанные в **entity**, являются видимыми во всех архитектурах, связанных с **entity**?

33. Шина и вектор два обозначения одного и того же понятия (концепции) в VHDL?
34. Левая граница порядка битов в векторе всегда должна быть меньше его правой границы?
35. Каждый порт должен иметь спецификацию своего направления (**in**,**out**, **inout**)?
36. Символом & обозначается логическая операция И?
37. Все сигналы системы описываются в ее декларации **entity**?
38. Имя декларации **entity** может быть записано после оператора **end** этой декларации **entity**?
39. Порт должен быть либо только входным, либо только выходным?
40. Значение настраиваемого параметра (**generic**) может динамически меняться в процессе моделирования?
41. Комментарий начинается и заканчивается двойным дефисом (--)?
42. Имеется ли пакете **STANDARD** декларация типа byte?
43. Настраиваемый параметр (**generic**) позволяет задать ввод постоянной величины (константы) в проектируемую систему извне ее?
44. Настраиваемый параметр может быть использован для спецификации портов, например, для задания ширины шин?
45. Имея в виду модельное время, можно ли сказать, что оператор a <= b; выполняется мгновенно?
46. Имея в виду физическое время, можно ли сказать, что система моделирования выполняет оператор a <= b; мгновенно?
47. Каждый настраиваемый параметр должен быть определен вместе со своим направлением?
48. Каждая декларация **entity** должна заканчиваться ключевым словом **end**?
49. Порт является сигналом?
50. Может ли переменная типа **integer** быть входным портом?
51. Может ли сигнал типа **string** быть выходным портом?
52. Использование комментариев при описании порта является обязательным требованием стандартов языка VHDL?

53. Начальное значение порта может быть задано?
54. Начальное значение переменной может быть задано?
55. В соответствии с пожеланием проектировщика оператор назначения сигнала a <= b; может быть записан вправо, т. е. записан в виде b => a; ?
56. В выражениях могут использоваться константы?
57. Процедура может быть декларирована в пакете?
58. Может ли сигнал A типа **real** передать свое значение сигналу B типа **integer** с помощью оператора B <= A **after** 5 ns; ?
59. Может ли значение true быть присвоено сигналу типа bit?
60. Может ли значение '0' быть назначено сигналу типа **boolean**?
61. Может ли значение '0' быть назначено сигналу типа **integer**?
62. Может ли значение 0 быть назначено сигналу типа **real**?
63. Может ли значение 0 быть назначено сигналу типа **integer**?
64. Внутренние состояния конечного автомата могут определяться как перечислимый тип?
65. Все элементы массива должны иметь одинаковый тип?
66. В VHDL можно умножать действительное число на целое?
67. В VHDL можно умножать действительное число на время (тип **time**)?
68. Через 0.0 обозначается значение нуля для типа **real**?
69. Можно ли декларировать двумерный массив, элементами которого являются натуральные числа, не большие 1000?
70. Если длительность сигнала больше чем его задержка, то инерционная модель задержки даст тот же результат, что и транспортная модель задержки?
71. Атрибут clk'event сигнала clk имеет тип boolean?
72. Атрибут S'left массива S целых чисел задает левое значение номера числа (индекса) в массиве?
73. Может ли пользователь определить собственный атрибут?
74. В VHDL есть предопределенные атрибуты?
75. Сигналы имеют атрибуты?
76. Константы имеют атрибуты?
77. Переменные имеют атрибуты?

78. Подтип является подмножеством значений типа?
79. Константе может быть назначено новое значение, если новое значение равно предыдущему?
80. В языке VHDL значение «истина» типа **boolean** равно значению '1' типа **bit**?
81. При определении декларации константы достаточно описать только ее значение, так как тип константы определяется значением?
82. Логические операторы могут выполняться только над операндами типа **bit**?
83. Логические операторы могут выполняться только над операндами типа **boolean**?
84. Логические операторы могут выполняться только над операндами типа **std_logic**?
85. Оператор ожидания **wait** в начале процесса играет ту же самую роль, как если бы он был расположен в процессе перед словом **end**?
86. Условный оператор **if** должен быть завершен одним ключевым словом **endif**?
87. Булево условие в цикле типа **while** проверяется в начале каждой итерации?
88. Выполнение процесса, не имеющего списка сигналов запуска, заканчивается, когда достигается ключевое слово **end**?
89. Процесс со списком чувствительности должен не содержать ни одного оператора ожидания?
90. Сигналы и переменные одного и того же типа могут быть присвоены один другому?
91. В **entity** обязательно должны быть описаны порты (**port**)?
92. Параметр (**generic**) должен быть константой?
93. Верно ли то, что по умолчанию рабочая библиотека проекта имеет имя WORK?
94. Можно ли перед декларацией **entity** ссылаться на несколько пакетов?
95. Можно ли в тексте конфигурации использовать другую конфигурацию?

96. Можно ли в перед описанием пакета ссылаться на другой пакет?
97. Константа может передать свое значение переменной того же типа?
98. Выражение в заголовке оператора **case** может быть дискретного типа?
99. Символом & обозначается операция конкатенации?
100. Процесс содержит последовательные операторы?
101. Пакет есть спецификация последовательных операторов?
102. Модельное время в VHDL описывается типом **integer**?
103. Модельное время в VHDL описывается типом **real**?
104. Блок (**block**) содержит последовательные операторы?
105. Оператор блока (**block**) может иметь охранное выражение?
106. Пакет может иметь охранное выражение?
107. Имя процесса специфицируется после слова **process**?
108. Счетчик в цикле типа **for** есть переменная, которую нужно декларировать в начале процесса, в котором цикл употребляется?
109. Сигналам назначаются значения тогда, когда все процессы в данном такте моделирования выполнены?
110. При декларации возрастающего диапазона употребляется ключевое слово **downto**?
111. Ключевое слово **to** употребляется при декларации убывающего диапазона?
112. Для сигналов типа **real** никогда не нужна разрешающая функция?
113. Идентификатор @mail является корректным?
114. Идентификатор 8mail является корректным?
115. Идентификаторы Abc7, aBC7 являются различными?
116. Строковые литералы 'b', 'B' являются различными?
117. Функция может быть декларирована в пакете?
118. Переменные могут употребляться для хранения промежуточных данных внутри процессов?
119. Рекомендуется ли использовать сигналы для хранения промежуточных данных внутри процессов?
120. Условный оператор **if** должен быть завершен ключевыми словами **end if**?

121. Только сигналы могут употребляться как переносчики информации между процессами?
122. Выборочное назначения сигнала представляет то же действие, что и условное назначение сигнала, только они записаны различным способом?
123. Прежде чем использовать пакет STD_LOGIC_1164, декларация **entity** должна быть обязательно предопределена операторами **library, use**?
124. Типы **std_logic** и **std_logic_vector** являются промышленным стандартом для логических сигналов и могут применяться для сигналов с многими источниками, так как эти типы являются «разрешимыми»?
125. Архитектура есть множество параллельных операторов, взаимодействующих между собой и находящихся под влиянием друг друга?
126. Операторы процесса (**process**) не могут употребляться вместе с операторами (<=) назначения сигнала в одной и той же архитектуре?
127. Оператор **null** назначает нулевое значение сигнала типа bit?
128. Оператор **next** позволяет перейти к выполнению следующего (по порядку записи) оператора архитектурного тела?
129. Оператор **next** позволяет перейти к выполнению следующего (по порядку записи) оператора блока?
130. Оператор **next** позволяет перейти к выполнению следующей итерации цикла?
131. Оператор **exit** позволяет завершить выполнение операторов архитектурного тела?
132. Можно ли с помощью оператора **assert** выдать сообщение без проверки условия?
133. Можно ли с помощью оператора **wait** выразить бесконечное ожидание?
134. Оператор конкретизации компонента (**port map**) может употребляться как в архитектурном теле (как параллельный оператор), так и в процессе (как последовательный оператор)?
135. Локальная переменная процесса может изменить свое значение несколько раз во время выполнения процесса?

136. Тип **bit** как стандартный логический тип не нуждается в разрешающей функции и может употребляться для сигналов с многими источниками?
137. Все процессы в архитектурном теле являются активными все время, когда архитектура активна?
138. Конструкция **if** ... **then** может употребляться внутри процесса?
139. Конструкция **if** ... **then** может употребляться внутри функции?
140. Цикловой параметр (счетчик числа итераций цикла) требуется декларировать снаружи цикла, записываемого с ключевым словом **for**?
141. Процедура имеет имя?
142. Функция имеет имя?
143. Функция может содержать последовательный оператор ожидания (**wait**)?
144. Оператор **wait** приостанавливает выполнение процесса?
145. Оператор **case** может употребляться внутри процедуры?
146. Метка компонента в операторе **port map** (конкретизации компонента) является необязательной?
147. Оператор **port map** может быть употреблен в функции?
148. Оператор **port map** может быть употреблен в процедуре?
149. Оператор **port map** может быть употреблен в теле оператора **generate**?
150. Оператор **xor** может выполняться над операндами типа bit_vector?
151. Оператор **and** не может выполняться над операндами типа boolean?
152. Оператор **or** может выполняться над операндами типа std_logic?
153. Оператор **or** может выполняться над операндами типа std_logic_vector?
154. Символом : (двоеточие) в VHDL обозначается операция деления?
155. Символом = в VHDL обозначается логическая операция эквивалентность?
156. Символом ** в VHDL обозначается операция извлечения квадратного корня?

157. Может ли в одном операторе **port map** быть назначение на одни порты одиночных бит, на другие порты — векторов?
158. Структурное описание состоит из компонент и сигналов?
159. Всегда ли все компоненты иерархического описания должны быть специфицированы, употребляя только поведенческое описание?
160. Оператор сдвига sll сдвига влево может быть оформлен как функция пакета?
161. Тип bit является перечислимым типом?
162. Тип boolean является перечислимым типом?
163. Дельта-задержка равна одной пикосекунде модельного времени?
164. Дельта-задержка равна одной фемтосекунде модельного времени?
165. Величина дельта-задержки определяется пользователем?
166. Дельта-задержка равна нулю модельного времени?
167. Имея в виду дельта-задержки, можно ли сказать, что задержка выполнения оператора a <= b; составляет две дельта-задержки?
168. В разделе деклараций архитектурного тела может находиться только одна функция?
169. В разделе деклараций архитектурного тела может находиться только одна процедура?
170. В пакете может находиться только две функции?
171. В теле пакета может находиться более 100 процедур?
172. В позиционном соответствии портов сигналы располагаются в том же порядке как порты в **entity**?
173. Компоненты, декларируемые внутри архитектуры, специфицируются полностью, т. е. с их интерфейсом и архитектурой?
174. В операторе **port map** символы => или <= (соответствия) употребляются в зависимости от направления порта (для входа символы =>, для выхода символы <=)?
175. Операторы **block** могут быть вложенными?
176. При вызове процедуры в архитектурном теле метка обязательна?
177. Метка процесса в архитектурном теле обязательна?
178. Все конструкции VHDL-описания цифровой системы реализуются автоматически при синтезе логическими элементами либо логическими подсхемами?

179. Оператор сложения целых чисел реализуется при синтезе логической схемой?
180. Оператор сложения вещественных чисел может быть реализован при синтезе логической схемой?
181. В тестирующей (головной VHDL-программе) есть описания входных и выходных портов?
182. В тестирующей программе могут тестироваться описания несвязанных между собой цифровых систем?
183. Проверяемое выражение clk = '1' в операторе **if** (clk = '1') имеет тип boolean?
184. Если сигнал clk имеет тип бит, то выражение (clk"event **and** clk = '1') также имеет тип bit?
185. Выражение (clk'event **and** clk = '1') соответствует заднему фронту сигнала clk?

5.4. КОНТРОЛЬНЫЕ ЗАДАЧИ

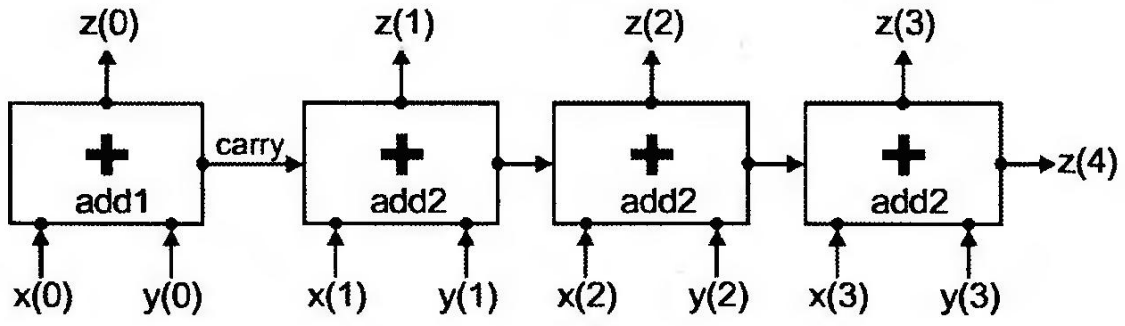
Задача 1.

Провести условное моделирование и нарисовать временные диаграммы поведения цифровой системы absd:

```
entity absd is
        generic (Tpw :Time:= 50 ns;
                 Tps :Time:= 30 ns);
        port (ph1, ph2 : out bit);
end absd;
architecture beh of absd is
        constant clock_period : Time := 2 * (Tpw + Tps);
begin
ccc: process
begin
                ph1 <= '1', '0' after Tpw;
        ph2 <= '1' after Tpw + Tps, '0' after Tpw + Tps + Tpw;
wait for clock_period;
        end process ccc;
end beh;
```

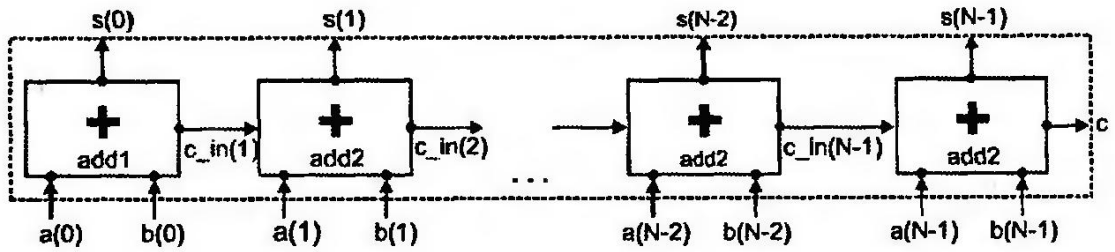
Задача 2.

Составить структурное описание 4-разрядного сумматора.



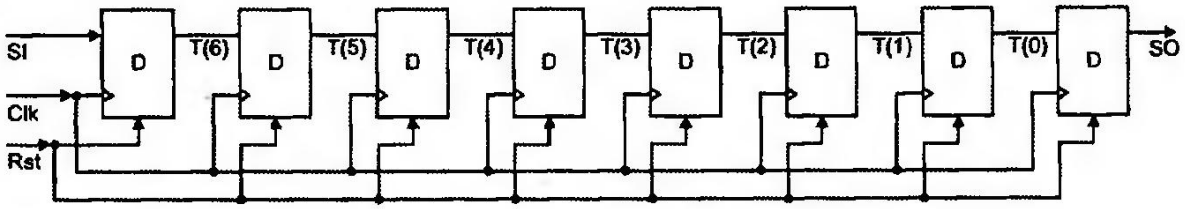
Задача 3.

Составить структурное описание N-разрядного сумматора. Использовать оператор **generate**.



Задача 4.

Составить структурное описание сдвигового регистра. Использовать оператор **generate**.



Задача 5.

Составить функциональное описание D-триггера.

Задача 6.

Написать функциональную VHDL-модель конечного автомата, заданного таблицей. Ввести сигнал синхронизации clk, переключать состояния по переднему фронту сигнала clk. Выходные сигналы автомата: y1, y2, y3, y4.

Совмещенная таблица переходов
и выходов конечного автомата

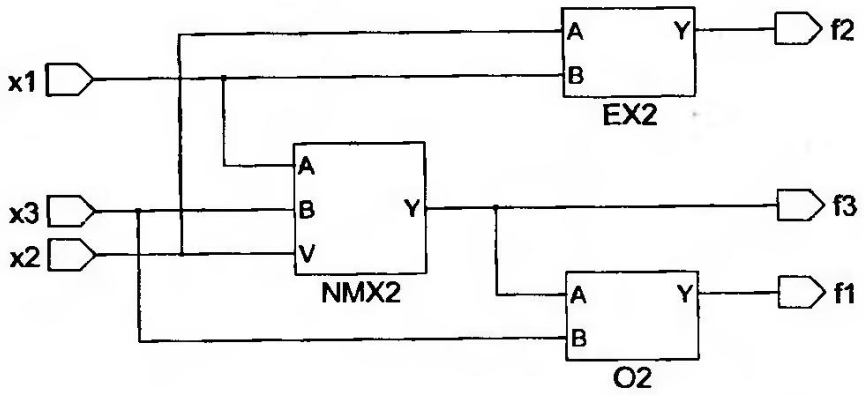
Входные сигналы	Состояния	
	d_1	d_2
x_1	d_2/y_1	d_1/y_2
x_2	d_1/y_3	d_1/y_4
x_3	d_1/y_2	d_1/y_4

Задача 7.

Написать на VHDL функцию преобразования двоичного представления числа в десятичное представление. Это функция преобразования типа BIT_VECTOR в тип INTEGER. Первый разряд двоичного представления является старшим.

Задача 8.

Составить смешанное описание схемы, используя для описания элементов: для EX2 — оператор процесса; для O2 — оператор назначения сигнала; для NMX2 — оператор создания экземпляра компонента.



Элемент O2 — двухвходовый дизъюнктор, реализующий функцию $y = A \vee B$;

элемент EX2 — исключающее ИЛИ (функция $y = A\overline{B} \vee \overline{A}B$);

элемент NMX2 реализует функцию $y = \overline{(A \vee \overline{V})(B \vee V)}$.

Задача 9.

Нарисовать логическую схему, полученную в результате синтеза по следующему VHDL-коду:

```
package pack is
type my_int1 is range 0 to 2;
end pack;
package body pack is
end pack;

use work.pack.all;
entity type_my_int is
port(
x1, x2 : in my_int1;
y : out my_int1);
end type_my_int;
architecture str of type_my_int is
begin
y <= x1 + x2;
end str;
```

Задача 10.

Нарисовать логическую схему, полученную в результате синтеза по VHDL-коду:

```
entity const_log is
port (x1, x2 : in bit_vector (0 to 4);
y : out bit_vector (0 to 4));
end const_log;
architecture beh of const_log is
constant b : bit_vector (0 to 4) := "01010";
begin
y <= (x1 and x2) or b;
end beh;
```

Задача 11.

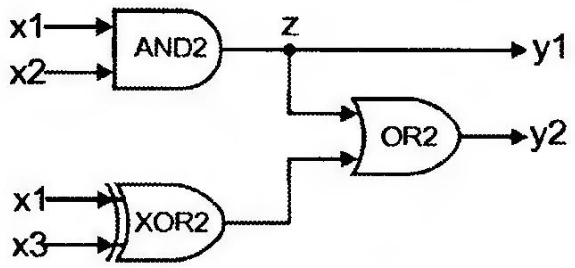
Нарисуйте логическую схему, полученную в результате синтеза по VHDL-коду:

```
entity circ is
port (a: in bit_vector (0 to 3);
    m: out bit_vector (0 to 3));
end circ;

architecture example of circ is
begin
process (a)
variable b: bit;
begin
b := '1';
for i in 0 to 3 loop
                b:= a(3 – i) or b;
                m( i ) <= b;
end loop;
end process;
end example;
```

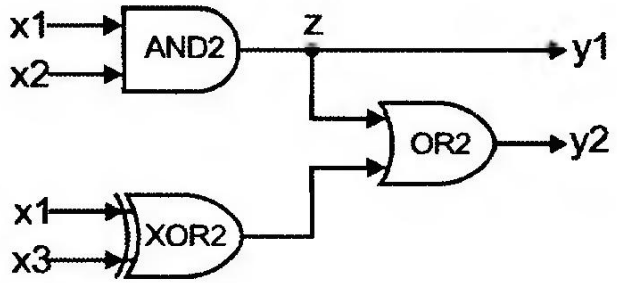
Задача 12.

Составить смешанное описание схемы, используя для описания элементов: AND2 — оператор процесса; OR2 — оператор назначения сигнала; XOR2 — оператор создания экземпляра компонента.



Задача 13.

Составить описание схемы, используя функции и процедуры для описания логических элементов. Функцию двухвходового логического элемента XOR2 написать без использования логического оператора **xor**.



Задача 14.

Написать VHDL-код функции нахождения максимального элемента в массиве, элементами которого являются натуральные числа. Написать процедуру для решения такой же задачи.

Задача 15.

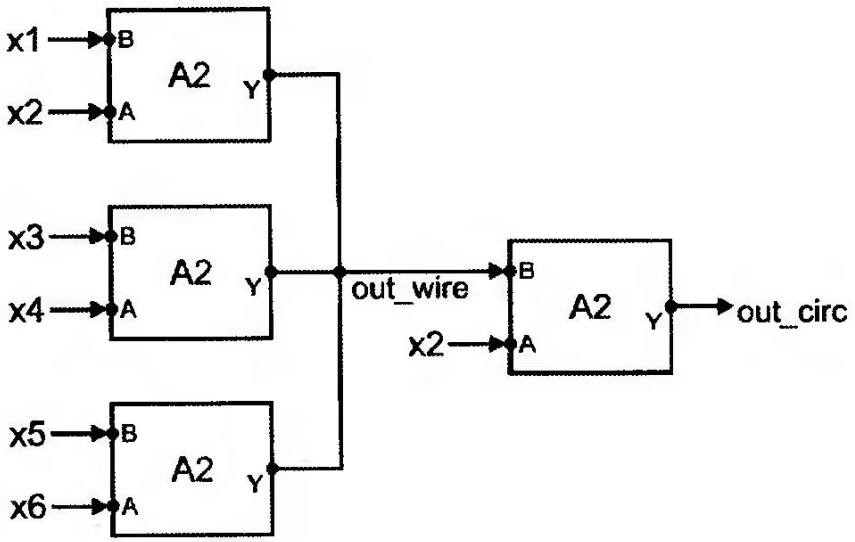
Написать VHDL-код процедуры транспонирования булевой (двоичной) матрицы размерности N (N строк, N столбцов).

Задача 16.

Написать VHDL-код цифровой системы, осуществляющей перемножение двух матриц размерности N × N, элементами которых являются целые положительные числа в диапазоне 0, 100.

Задача 17.

Используя тип std_logic, написать структурное описание схемы. Элемент A2 — двухвходовый конъюнктор.



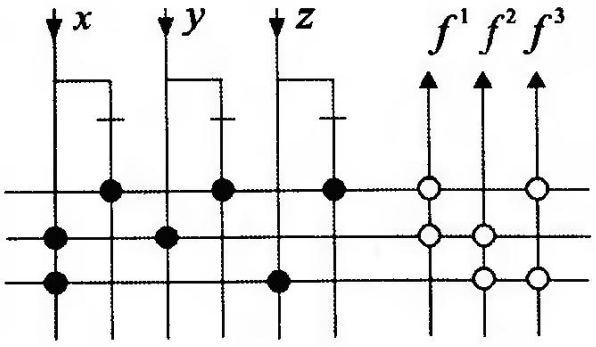
Задача 18.

Составить VHDL-модель схемы ПЗУ, хранящей 8 слов, каждое из которых состоит из 3 бит:

011
110
111
000
010
011
110
110

Задача 19.

Составить VHDL-модель схемы программируемой логической матрицы



Задача 20.

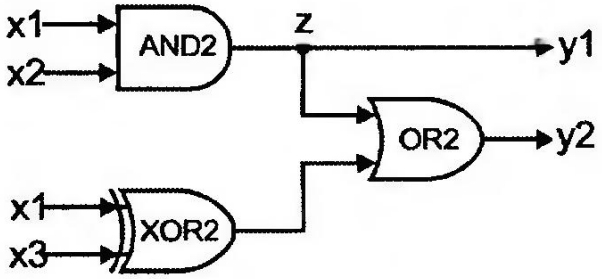
Составить VHDL-модель схемы мультиплексора с тремя управляющими входами.

Задача 21.

Составить VHDL-модель схемы дешифратора с тремя входными полюсами. Привести таблицу истинности дешифратора.

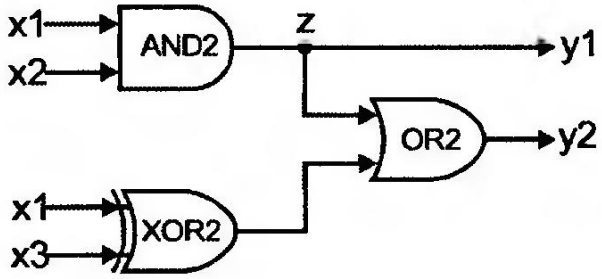
Задача 22.

Составить описание схемы, используя для каждого элемента процесс со своим списком чувствительности.



Задача 23.

Составить описание схемы, используя для каждого элемента отдельное описание в виде **entity**.



Задача 24.

Нарисовать логическую схему, полученную в результате синтеза по следующему VHDL-коду

```
entity arithm_1 is
port (x1, x2 : in integer range 0 to 2;
y : out integer range 0 to 4);
end arithm_1;

architecture beh of arithm_1 is
begin
y <= x1 + x2;
end beh;
```

Задача 25.

Нарисовать логическую схему, полученную в результате синтеза по следующему VHDL-коду:

```
entity arithm_1 is
port (x1, x2 : in integer range 0 to 2;
y : out integer range 0 to 4);
end arithm_1;
architecture beh of arithm_1 is
begin
y <= x1 * x2;
end beh;
```

5.5. ЛИТЕРАТУРА ПО ЯЗЫКУ VHDL

1. *Авдеев Н. А., Бибило П. Н.* Пакет numeric_std языка VHDL // Информационные технологии. 2005. № 9. С. 26–35.
2. *Авдеев Н. А., Бибило П. Н.* Реализация VHDL-функций пакета numeric_std логическими схемами //Информационные технологии. 2006. № 1. С. 9–18.
3. *Армстронг Дж. Р.* Моделирование цифровых систем на языке VHDL: Пер с англ. М.: Мир, 1992. 175 с.
4. *Бибило П. Н.* Основы языка VHDL. Второе издание. М.: Солон-Р, 2002. 224 с.
5. *Бибило П. Н.* Синтез логических схем с использованием языка VHDL. М.: Солон-Р, 2002. 384 с.
6. *Бибило П. Н.* Системы проектирования интегральных схем на основе языка VHDL. StateCAD, ModelSim, LeonardoSpectrum. М.: СОЛОН-Пресс, 2005. 384 с.
7. *Бибило П. Н.* Логическое перепроектирование схем, реализованных на FPGA, в схемы на базовых матричных кристаллах // Информационные технологии. 2004. № 1. С. 10 –17.
8. *Бибило П. Н.* Проектирование конечных автоматов в САПР WebPack ISE фирмы Xilinx // Информационные технологии. 2004. № 3. С. 8–13.
9. *Бибило П. Н., Кочанов Д. А.* Получение VHDL-моделей схем для вычисления функций, заданных в табличной форме // Информатика. 2004. № 3. С. 29–38.
10. *Грушвицкий Р. И., Мурсаев А. Х., Угрюмов Е. П.* Проектирование систем на микросхемах программируемой логики. СПб.: БХВ-Петербург, 2002. 608 *с.*
11. *Зотов Ю. В.* Проектирование цифровых устройств на основе ПЛИС фирмы XILINX в САПР WebPack ISE. М.: Горячая линия — Телеком, 2003. 624 с.
12. *Зотов Ю. В.* ModelSim — система HDL-моделирования цифровых устройств // Компоненты и технологии. 2002. № 6.
13. *Косткин М. Д., Лохов А. Л.* Комплексное проектирование FPGA/ASIC с помощью пакета FPGA Advantage. Часть 2. ModelSim // Информационные технологии. 2002. № 11. С. 45–48.
14. *Палагин А. В., Опанасенко В. Н., Сахарин В. Г.* Опыт проектирования цифровых устройств на базе ПЛИС с использованием HDL-технологии // Управляющие системы и машины. 2004. № 6. С. 11–20.
15. *Перельройзен Е. З.* Проектируем на VHDL. М.: СОЛОН Пресс, 2004. 448 с.

16. *Поляков А. К.* Языки VHDL и VERILOG в проектировании цифровой аппаратуры. М.: СОЛОН-Пресс, 2003. 320 с.
17. *Сергиенко А. М.* VHDL для проектирования вычислительных устройств. К.: ЧП «Корнейчук», ООО «ТИД „ДС“», 2003. 208 с.
18. *Стешенко В. Б.* EDA. Практика автоматизированного проектирования радиоэлектронных устройств. М.: Нолидж, 2002. 768 с.
19. *Суворова Е. А., Шейнин Ю. Е.* Проектирование цифровых систем на VHDL. СПб.: БХВ-Петербург, 2003. 576 с.
20. *Уэйкерли Дж. Ф.* Проектирование цифровых устройств. Т. 2. М.: Постмаркет, 2002. 528 с.
21. VHDL для моделирования, синтеза и формальной верификации аппаратуры / Пер с англ. М: Радио и связь, 1995. 360 с.
22. *Шарипова Н. Н., Овчинников Н. В.* Об одном подходе к разработке VHDL-описаний и моделированию встроенных систем. Микроэлектроника. 1999. Т.28. № 1. С. 62–67.
23. VHDL'92. Новые свойства языка описания аппаратуры VHDL / Пер с англ. М.: Радио и связь, 1995. 256 с.
24. IEEE Standard VHDL Language Reference Manual, IEEE Std 1076-1987.
25. IEEE Standard VHDL Language Reference Manual, IEEE Std 1076-1993.
26. IEEE Standard VHDL Analog and Mixed-Signal Extensions, IEEE Std 1076.1-1999.
27. *Zwolinski M.* Projektowanie uklladow cyfrowych z wykorzystaniem jezyka VHDL. Warszawa, 2002. 368 c.

5.6. ИНТЕРНЕТ-РЕСУРСЫ

1. Учебный русскоязычный сайт по VHDL

`www.bsuir.unibel.by/vhdl/`

На сайте размещены:

- Справочная система по операторам и ключевым словам.
- Обучающая система Evita-VHDL (на анлийском языке).
- Обучающая система Evita-VHDL (на русском языке).
- Тестовые (да/нет) вопросы.
- Лабораторные работы.
- Примеры VHDL-программ.
- Полный синтаксис языка VHDL.
- Список опубликованной (на русском языке) литературы по языку VHDL.
- Ссылки на другие сайты.

2. Список сайтов по языкам проектирования, системам автоматизированного проектирования и современной элементной базе:

`vhdl.org` — интернет-сервер международного форума пользователей VHDL (VHDL International Users' Forum-VIUF),

`eda.org` — сайт EDA Industry Working Groups, содержащий ссылки на рабочие группы VHDL, ресурсы по VHDL, стандарты IEEE, а также интернациональные конференции и форумы;

`eda.org/rassp/` — курсы, учебники по VHDL;

`www.dsol.ru/stud/` — книги по элементной базе и САПР;

`eda.org/comp.lang.vhdl/` — часто задаваемые вопросы по VHDL;

`tech-www.informatik.uni-hamburg.de` — сайт, содержащий множество VHDL-моделей;

`aldec.com` — сайт разработчиков обучающих программ (компания Aldec). Компания Aldec разработала обучающую систему VHDL-Evita.

`en.wikipedia.org/wiki/VHDL` — начальные сведения по VHDL в электронной энциклопедии;

`tech-www.informatik.uni-hamburg.de` — сайт, содержащий VHDL-модели;

`freemodelfoundry.com` — сайт, содержащий VHDL-модели;

`synopsys.com` — программное обеспечение САПР;

`mentor.com` — программное обеспечение САПР;

`cadence.com` — программное обеспечение САПР;

`vhdl-ams.org` — стандарт языка VHDL'99;

`xilinx.com` — элементная база, программное обеспечение САПР;

`altera.com` — элементная база, программное обеспечение САПР;

`samsung.com` — элементная база;

`intel.com` — элементная база;

`cypress.com` — элементная база.

Литература

1. *Армстронг Дж. Р.* Моделирование цифровых систем на языке VHDL / Пер с англ. М.: Мир, 1992. 175 с.
2. *Баранов С. И., Скляров В. А.* Цифровые устройства на программируемых БИС с матричной структурой. М.: Радио и связь, 1986. 279 с.
3. *Бибило П. Н.* Кремниевая компиляция заказных СБИС. Минск: Ин-т техн. кибернетики АН Беларуси, 1996. 268 с.
4. *Закревский А. Д.* Параллельные алгоритмы логического управления. М.: УРСС, 2003. 200 с.
5. *Соловьев В. В., Васильев А. Г.* Программируемые логические интегральные схемы и их применение. Минск: Беларуская навука, 1998. 270 с.
6. VHDL для моделирования, синтеза и формальной верификации аппаратуры / Пер с англ. М: Радио и связь, 1995. 360 с.
7. VHDL'92. Новые свойства языка описания аппаратуры VHDL/Пер с англ. М.: Радио и связь, 1995. 256 с.
8. *Asheden P. J.* The VHDL Cookbook. University of Adelaide, South Australia. 1990.
9. *Chang K. C.* Digital design and modeling with VHDL and synthesis. IEEE Computer Society Press. 1997.
10. IEEE Standard VHDL Language Reference Manual, IEEE Std 1076-1987.
11. IEEE Standard VHDL Language Reference Manual, IEEE Std 1076-1993.