ТИПОВЫЕ

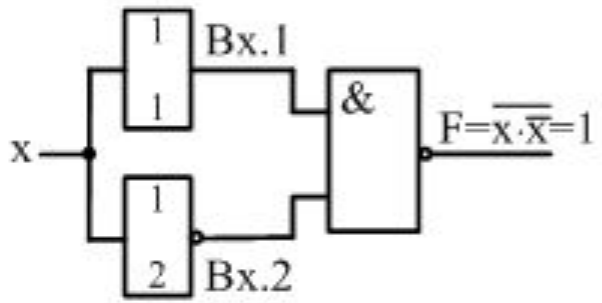
КЦУ

Как известно, функциональные узлы цифровых устройств (ЦУ) делятся на *комбинационные* и *последовательностные*.

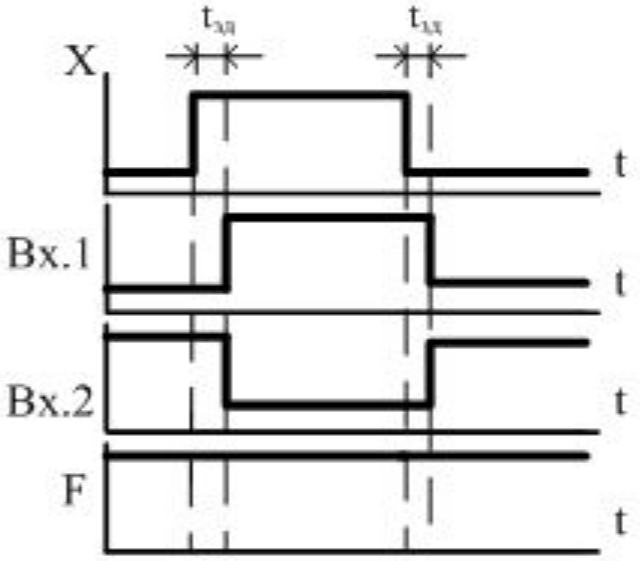
Выходные сигналы КЦУ зависят только от текущего значения входных сигналов (аргументов). Предыдущие значения аргументов значения не имеют.

При поступлении входных сигналов в КЦУ начинаются переходные процессы. После их завершения на выходах КЦУ устанавливаются выходные сигналы, на которые характер переходных процессов влияния не оказывает. С этой точки зрения переходные процессы в КЦУ не опасны. Но в цифровых устройствах КЦУ работают совместно с ПЦУ, что кардинально меняет ситуацию. *Во время переходных процессов на выходе КЦУ появляются временные сигналы, не предусмотренные таблицей истинности и называемые рисками*. После окончания переходных процессов они исчезают, и выходные сигналы КЦУ приобретают значения, предусмотренные логическими функциями, описывающими работу устройства. Однако во время переходных процессов риски могут быть восприняты элементами памяти ПЦУ, необратимое изменение состояния которых может радикально изменить работу цифрового устройства, несмотря на исчезновение сигналов рисков на выходах КЦУ после завершения переходных процессов. *Это явление называется «опасными состязаниями»* и появляется из-за того, что к выходному логическому элементу сигналы поступают неодновременно из-за различных задержек сигналов в разных цепях схемы.

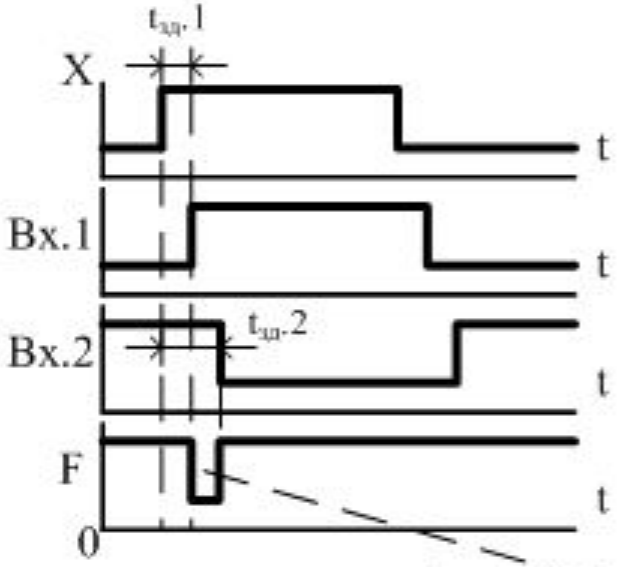
Различают *статические и динамические риски*. Статические риски – это кратковременное изменение сигнала, который должен был бы оставаться неизменным (единичным или нулевым). Если согласно логике работы КЦУ состояние выхода должно измениться, но вместо однократного перехода происходят многократные, то имеют место динамические риски. При динамических рисках первый и последний переходы всегда совпадают с алгоритмическими, предусмотренными логикой работы схемы. Статические риски такого свойства не имеют и считаются более опасными.



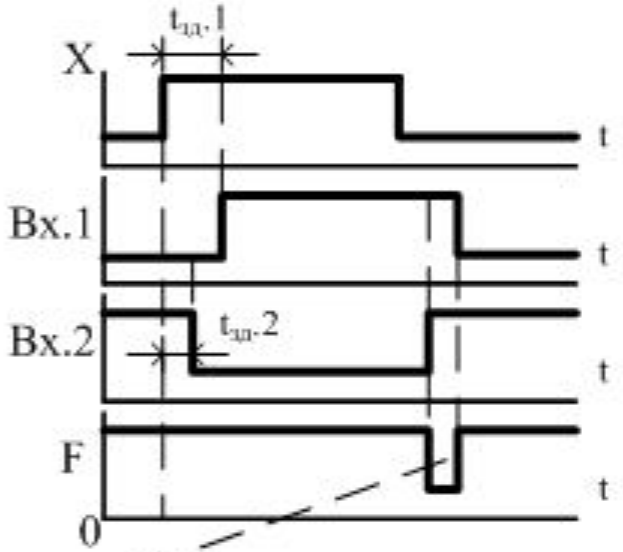
а)



б)

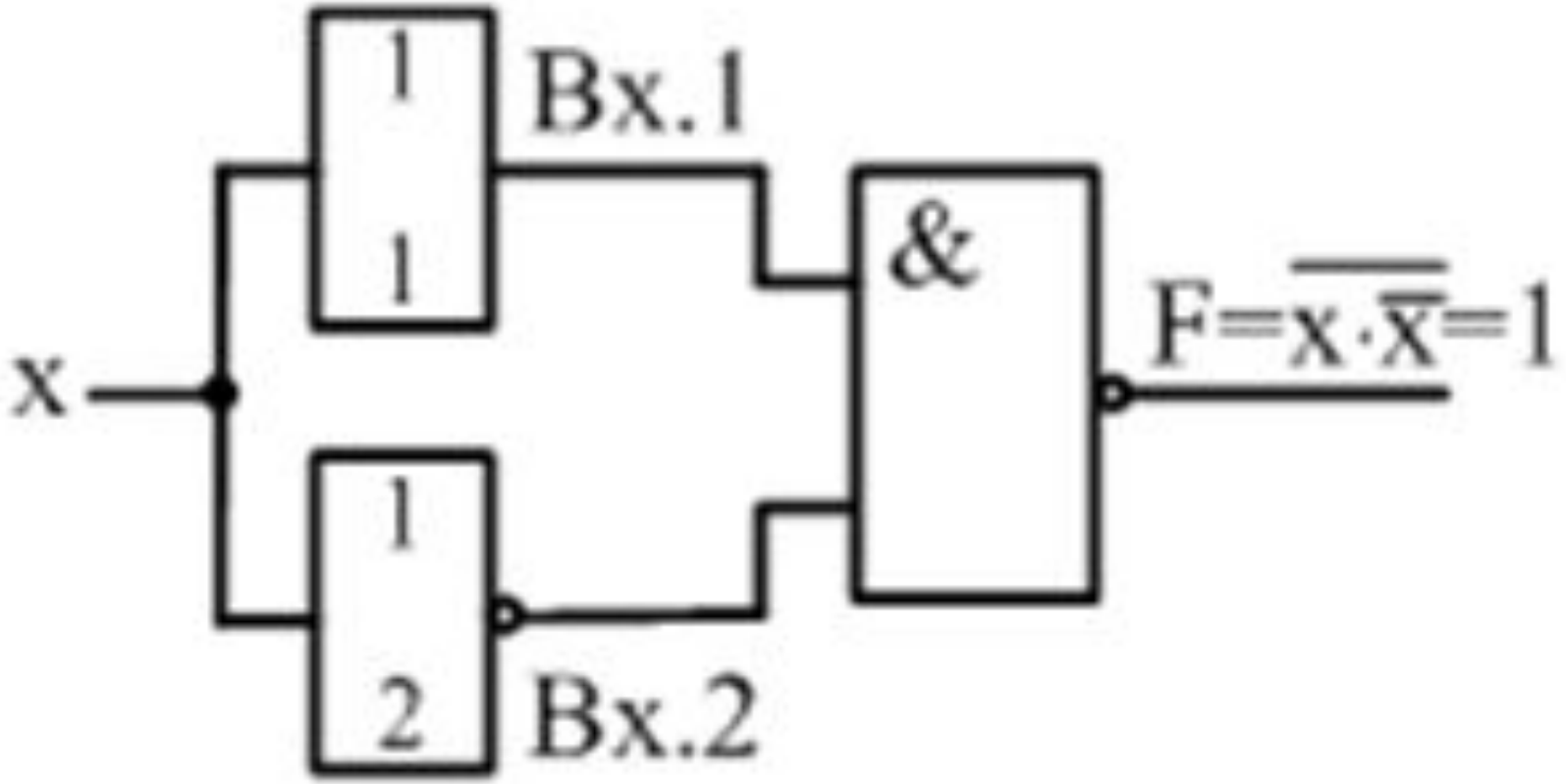


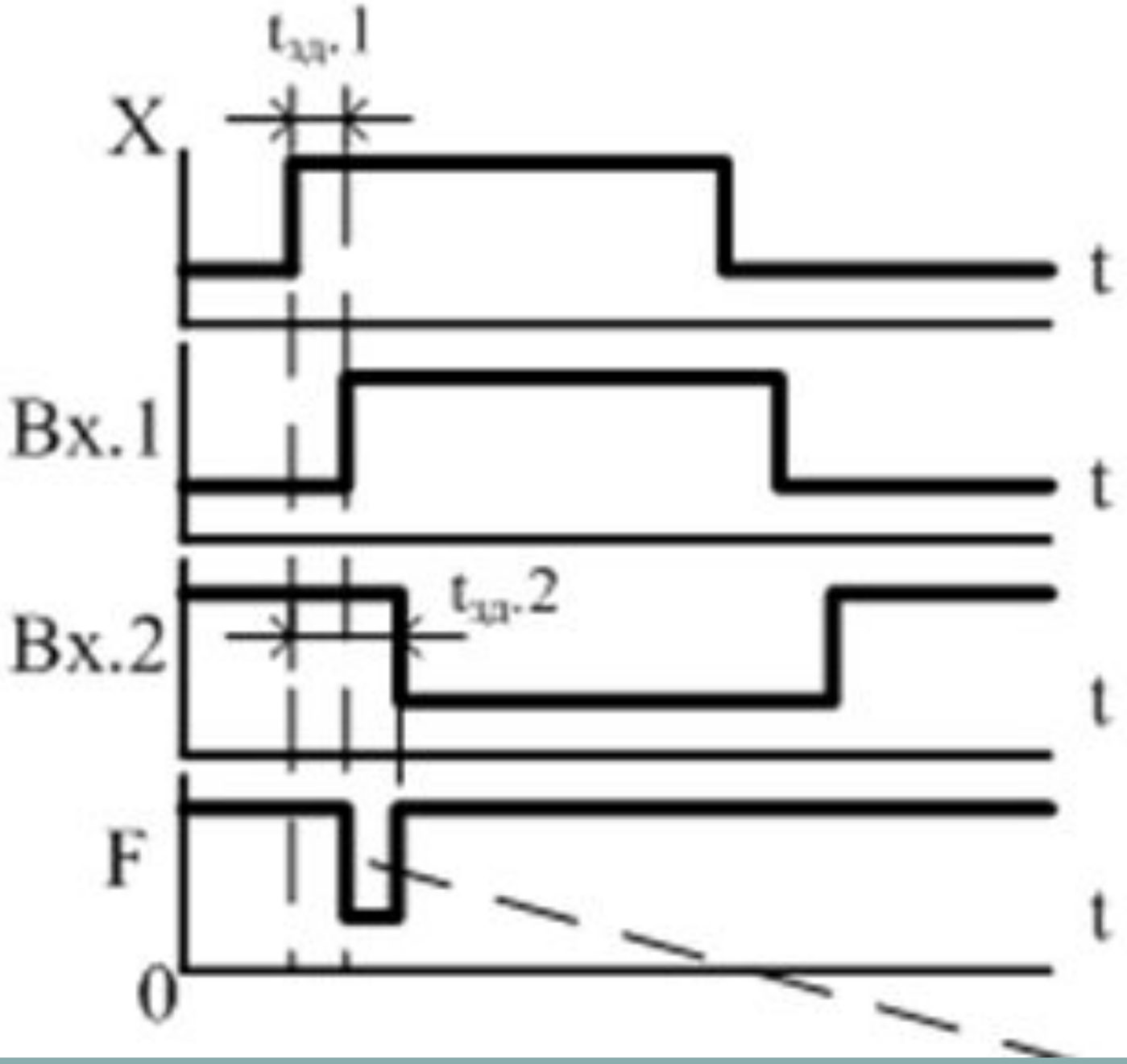
в)



г)

Сигналы статического риска


1
1
Bx.1
1
2
Bx.2
&
x
F=x·x=1


$t_{зд.1}$
X
t
Вх.1
t
$t_{зд.2}$
Вх.2
t
F
t
0

Для исключения возможных сбоев в работе ЦУ из-за «опасных состязаний» имеются два пути.

- *Первый путь* состоит в синтезе схем, свободных от рисков, и требует сложного анализа процессов в схеме и введения избыточных логических элементов для выравнивания задержек в различных цепях схемы с целью исключения рисков. Этот путь редко используется на практике.
- *Второй путь*, основной для современной цифровой схемотехники, предусматривает *запрещение восприятия сигналов КЦУ элементами памяти ПЦУ на время переходных процессов*. Прием информации с выходов КЦУ разрешается только специальным *сигналом синхронизации*, подаваемым на элементы памяти после окончания переходных процессов в КЦУ. Таким образом, исключается воздействие ложных сигналов на элементы памяти. Такие ЦУ называются *синхронными*.

Для определения временного интервала, на котором проходят переходные процессы, следует оценить задержки на путях распространения сигналов от входов к выходам КЦУ. В общем случае нужно оценить задержку сигнала на самом коротком и на самом длинном путях.

Способы схемотехнической реализации логических функций.

Логические блоки, собираемые из логических элементов некоторого базиса (SLC, Small Logic Cells).

- Синтез КЦУ на логических блоках SLC, т.е. на вентильном уровне, является самым традиционным и изученным (термином «вентиль» называют базовые логические элементы, например, элементы И-НЕ с двумя-тремя входами);

Логические блоки в виде последовательности матриц логических элементов И и ИЛИ (PLA, Programmable Logic Array; PAL, Programmable Array Logic).

- Логические блоки с матрицами элементов И и ИЛИ воспроизводят системы логических функций и имеют параметры: число входов (число аргументов воспроизводимых функций), число выходов (число функций) и число термов (конъюнкций). Если сложность логической функции превышает возможности логического блока, то функцию следует минимизировать с целью сокращения числа термов;

Универсальные логические блоки на основе мультиплексоров

- (рассмотрены ниже после ознакомления с мультиплексорами);

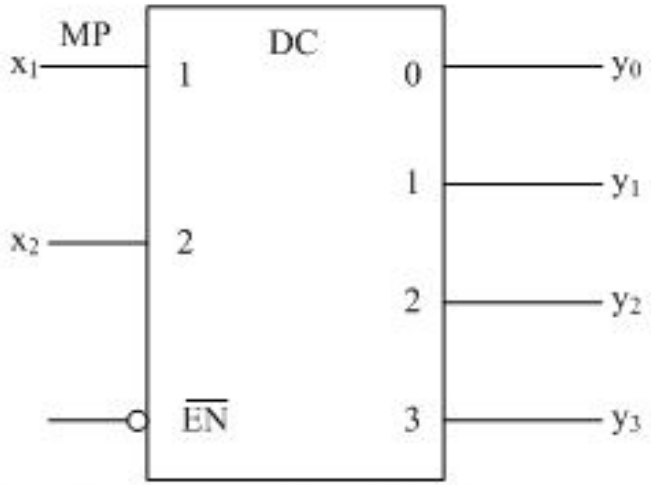
Логические блоки табличного типа (LUT$_S$, Look-Up Tables).

В этом случае совершенная дизъюнктивная нормальная форма (СДНФ) является окончательным выражением логической функции. Табличный блок представляет собой память, в которой имеется столько ячеек, сколько необходимо для хранения всех значений функций, т.е. 2^n, где n – число аргументов функции. Набор аргументов является адресом той ячейки, в которой хранится значение функции на этом наборе. СДНФ как раз и содержит все адреса, по которым нужно хранить единичные значения функции. Если логическая функция выражена в какой-либо сокращенной форме, то ее следует перевести в СДНФ. Если требуется воспроизвести n функций, то в каждой ячейке следует хранить n бит (по одному биту для каждой функции).

Дешифраторы

- *Дешифратором* называется КЦУ, которое служит для преобразования n-разрядных слов из двоичного позиционного кода в двоичный унитарный код. *Унитарным* называется двоичный код, в котором каждое слово содержит единицу только в одном из своих разрядов, а в остальных разрядах имеет нули. Таким образом, в зависимости от входного двоичного кода на выходе дешифратора возбуждается одна и только одна из выходных цепей.

  Из сказанного следует, что двоичный дешифратор, имеющий n входов, должен иметь 2^n выходов. Такой дешифратор называется полным. Если часть входных наборов не используется, то дешифратор называется *неполным*, и у него число выходов меньше 2^n. Например, неполный дешифратор, имеющий четыре входа и десять выходов, называется *десятичным*. Такой дешифратор является частным случаем полного двоичного дешифратора на четыре входа и шестнадцать выходов и выполняет дешифрацию двоичнокодированных десятичных цифр.



Условное графическое обозначение двоичного дешифратора со входом разрешения

Таблица истинности двоичного дешифратора

Логические аргументы			Логические функции			
$\overline{EN}$	x_2	x_1	y_0	y_1	y_2	y_3
1	x	x	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1
Примечание – Знаком «x» обозначен произвольный сигнал (0 или 1).						

По данным таблицы 1 запишем систему логических функций (конъюнкций) в СДНФ, описывающих работу дешифратора:

$$\left.\begin{aligned} y_0 &= \overline{x_1}\cdot\overline{x_2}\cdot\overline{EN};\\ y_1 &= x_1\cdot\overline{x_2}\cdot\overline{EN};\\ y_2 &= \overline{x_1}\cdot x_2\cdot\overline{EN};\\ y_3 &= x_1\cdot x_2\cdot\overline{EN}. \end{aligned}\right\} \qquad (1)$$

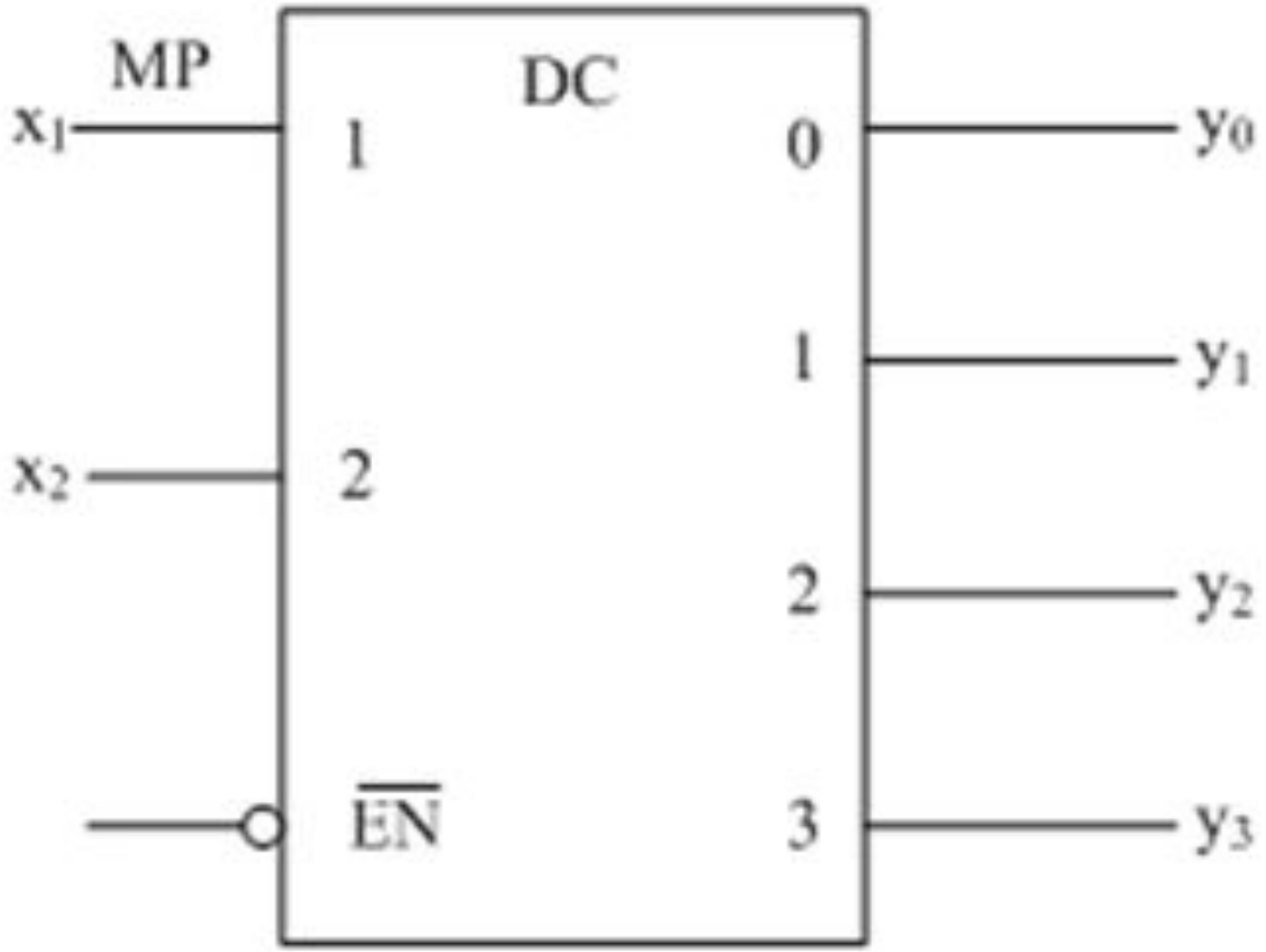
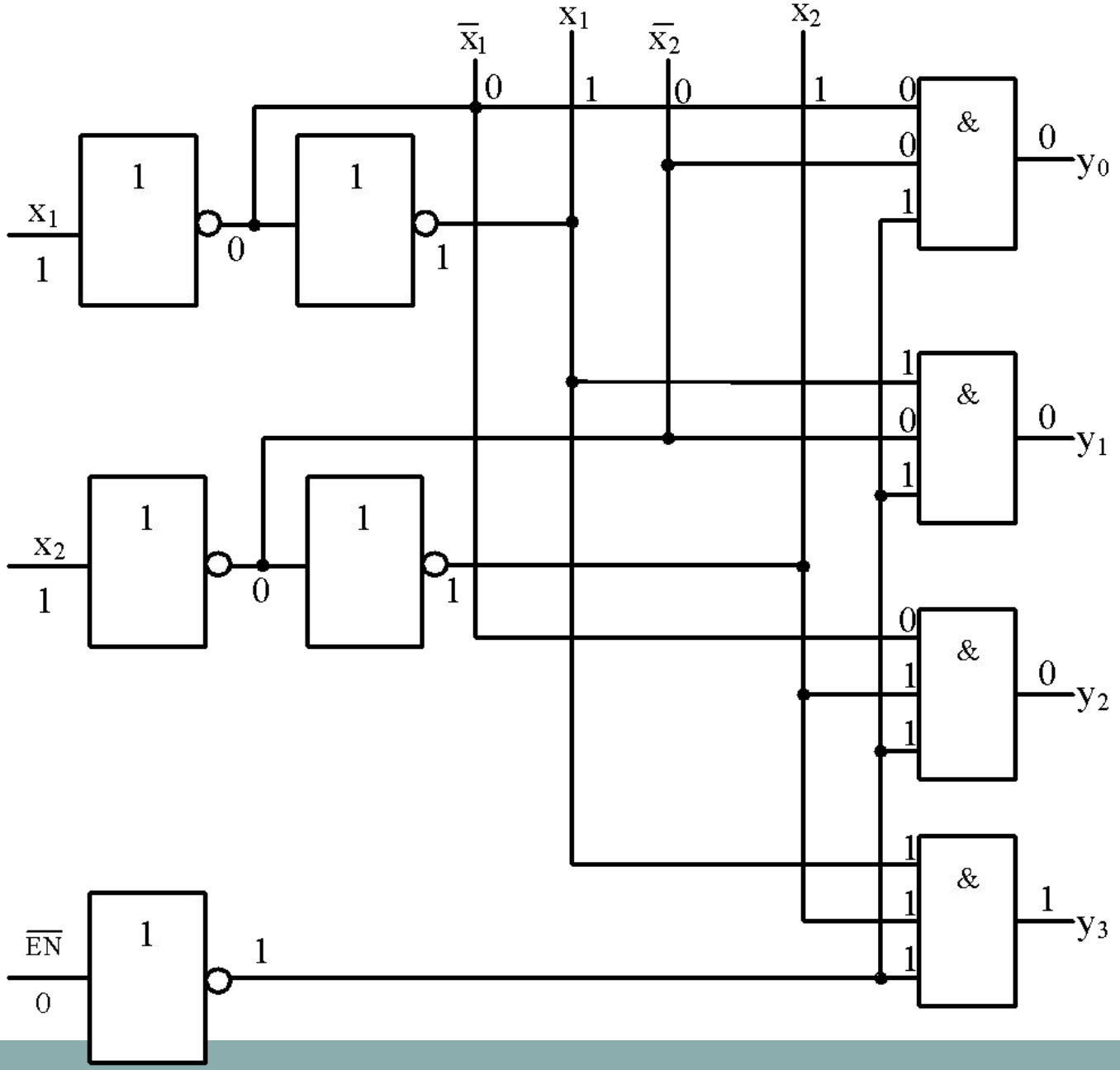

MP
DC
x1
x2
1
2
EN
0
1
2
3
y0
y1
y2
y3

Таблица истинности двоичного дешифратора

Логические аргументы			Логические функции			
$\overline{EN}$	x_2	x_1	y_0	y_1	y_2	y_3
1	x	x	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1

Примечание – Знаком «x» обозначен произвольный сигнал (0 или 1).

$$\left.\begin{aligned} y_0 &= \overline{x_1} \cdot \overline{x_2} \cdot \overline{EN}; \\ y_1 &= x_1 \cdot \overline{x_2} \cdot \overline{EN}; \\ y_2 &= \overline{x_1} \cdot x_2 \cdot \overline{EN}; \\ y_3 &= x_1 \cdot x_2 \cdot \overline{EN}. \end{aligned}\right\}$$


$\overline{x_1}$
x_1
$\overline{x_2}$
x_2
0
1
0
1
1
1
x_1
1
0
1
1
1
x_2
1
0
1
1
1
$\overline{EN}$
0
1
1
&
&
&
&
0
0
0
1
1
0
0
1
1
0
0
1
1
1
y_0
y_1
y_2
y_3

Рассмотрим методику построения двухступенчатого дешифратора при n = 4. В качестве исходного примем классическое описание одноступенчатого дешифратора системой логических функций, заданных в СДНФ:

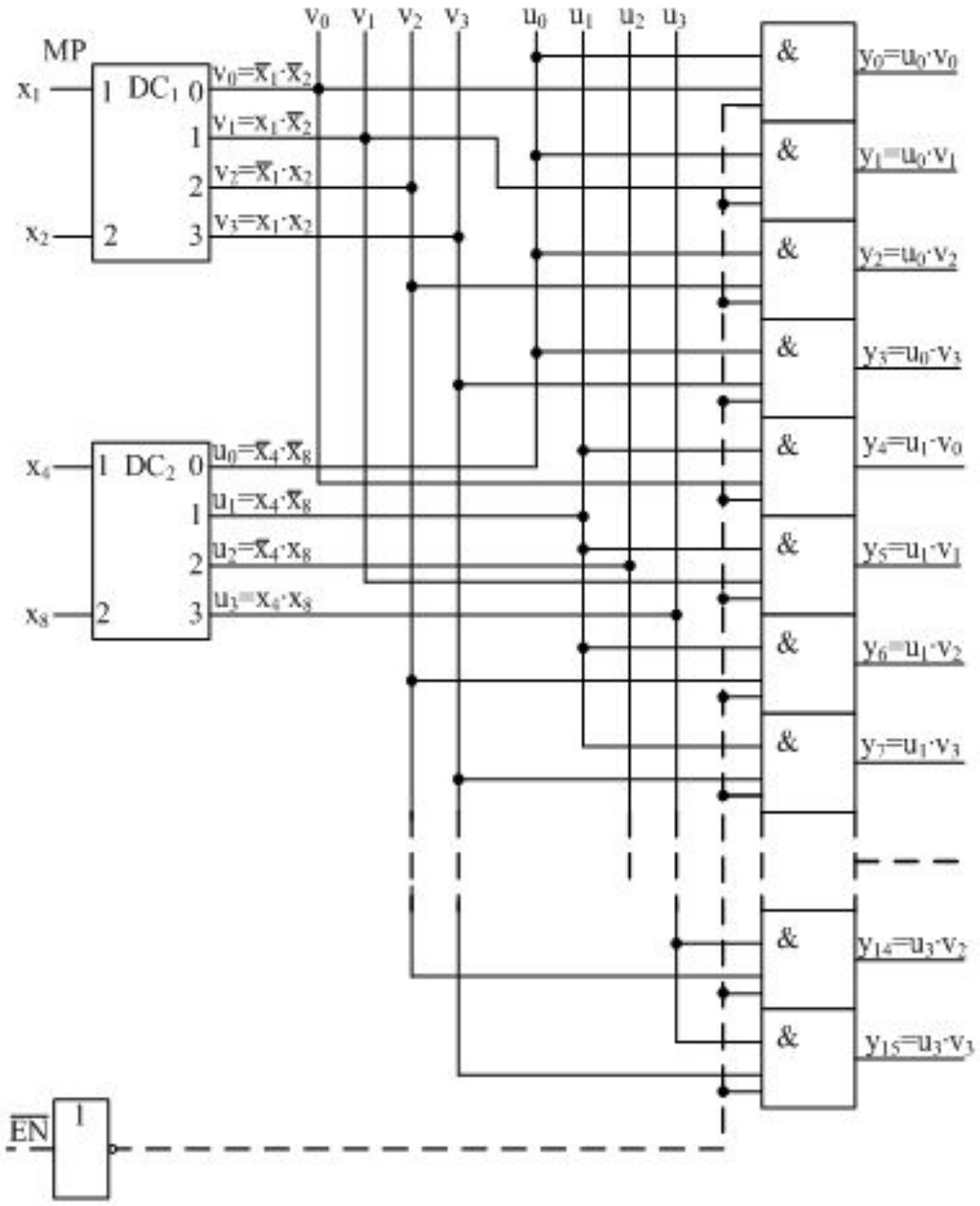
$$
\begin{aligned}
y_0 &= \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_4} \cdot \overline{x_8}; & y_8 &= \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_4} \cdot x_8; \\
y_1 &= x_1 \cdot \overline{x_2} \cdot \overline{x_4} \cdot \overline{x_8}; & y_9 &= x_1 \cdot \overline{x_2} \cdot \overline{x_4} \cdot x_8; \\
y_2 &= \overline{x_1} \cdot x_2 \cdot \overline{x_4} \cdot \overline{x_8}; & y_{10} &= \overline{x_1} \cdot x_2 \cdot \overline{x_4} \cdot x_8; \\
y_3 &= x_1 \cdot x_2 \cdot \overline{x_4} \cdot \overline{x_8}; & y_{11} &= x_1 \cdot x_2 \cdot \overline{x_4} \cdot x_8; \\
y_4 &= \overline{x_1} \cdot \overline{x_2} \cdot x_4 \cdot \overline{x_8}; & y_{12} &= \overline{x_1} \cdot \overline{x_2} \cdot x_4 \cdot x_8; \\
y_5 &= x_1 \cdot \overline{x_2} \cdot x_4 \cdot \overline{x_8}; & y_{13} &= x_1 \cdot \overline{x_2} \cdot x_4 \cdot x_8; \\
y_6 &= \overline{x_1} \cdot x_2 \cdot x_4 \cdot \overline{x_8}; & y_{14} &= \overline{x_1} \cdot x_2 \cdot x_4 \cdot x_8; \\
y_7 &= x_1 \cdot x_2 \cdot x_4 \cdot \overline{x_8}; & y_{15} &= x_1 \cdot x_2 \cdot x_4 \cdot x_8.
\end{aligned}
$$

Введем новые обозначения:

$$\left.\begin{array}{ll} u_0 = \overline{x_4} \cdot \overline{x_8}; & v_0 = \overline{x_1} \cdot \overline{x_2}; \\ u_1 = x_4 \cdot \overline{x_8}; & v_1 = x_1 \cdot \overline{x_2}; \\ u_2 = \overline{x_4} \cdot x_8; & v_2 = \overline{x_1} \cdot x_2; \\ u_3 = x_4 \cdot x_8; & v_3 = x_1 \cdot x_2. \end{array}\right\}$$

Подставим функции в равенства Получим

$$\left.\begin{array}{ll} y_0 = u_0 \cdot v_0; & y_8 = u_2 \cdot v_0; \\ y_1 = u_0 \cdot v_1; & y_9 = u_2 \cdot v_1; \\ y_2 = u_0 \cdot v_2; & y_{10} = u_2 \cdot v_2; \\ y_3 = u_0 \cdot v_3; & y_{11} = u_2 \cdot v_3; \\ y_4 = u_1 \cdot v_0; & y_{12} = u_3 \cdot v_0; \\ y_5 = u_1 \cdot v_1; & y_{13} = u_3 \cdot v_1; \\ y_6 = u_1 \cdot v_2; & y_{14} = u_3 \cdot v_2; \\ y_7 = u_1 \cdot v_3; & y_{15} = u_3 \cdot v_3. \end{array}\right\}$$


MP
x1
x2
DC1
1
2
0
1
2
3
v0=x̄1·x̄2
v1=x1·x̄2
v2=x̄1·x2
v3=x1·x2
x4
x8
DC2
u0=x̄4·x̄8
u1=x4·x̄8
u2=x̄4·x8
u3=x4·x8
v0 v1 v2 v3
u0 u1 u2 u3
&
y0=u0·v0
y1=u0·v1
y2=u0·v2
y3=u0·v3
y4=u1·v0
y5=u1·v1
y6=u1·v2
y7=u1·v3
y14=u3·v2
y15=u3·v3
EN
1

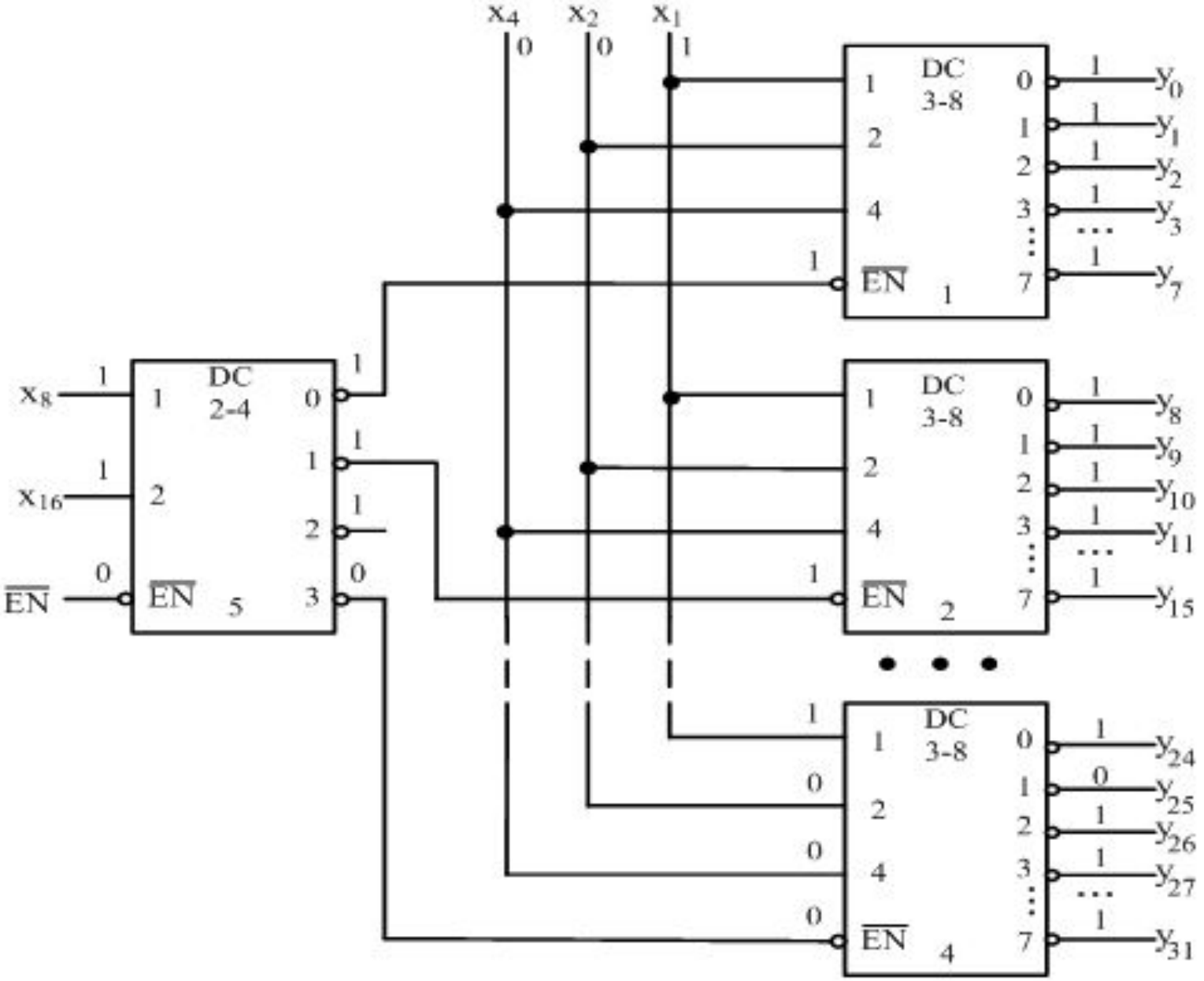


Схема наращивания разрядности двоичного дешифратора

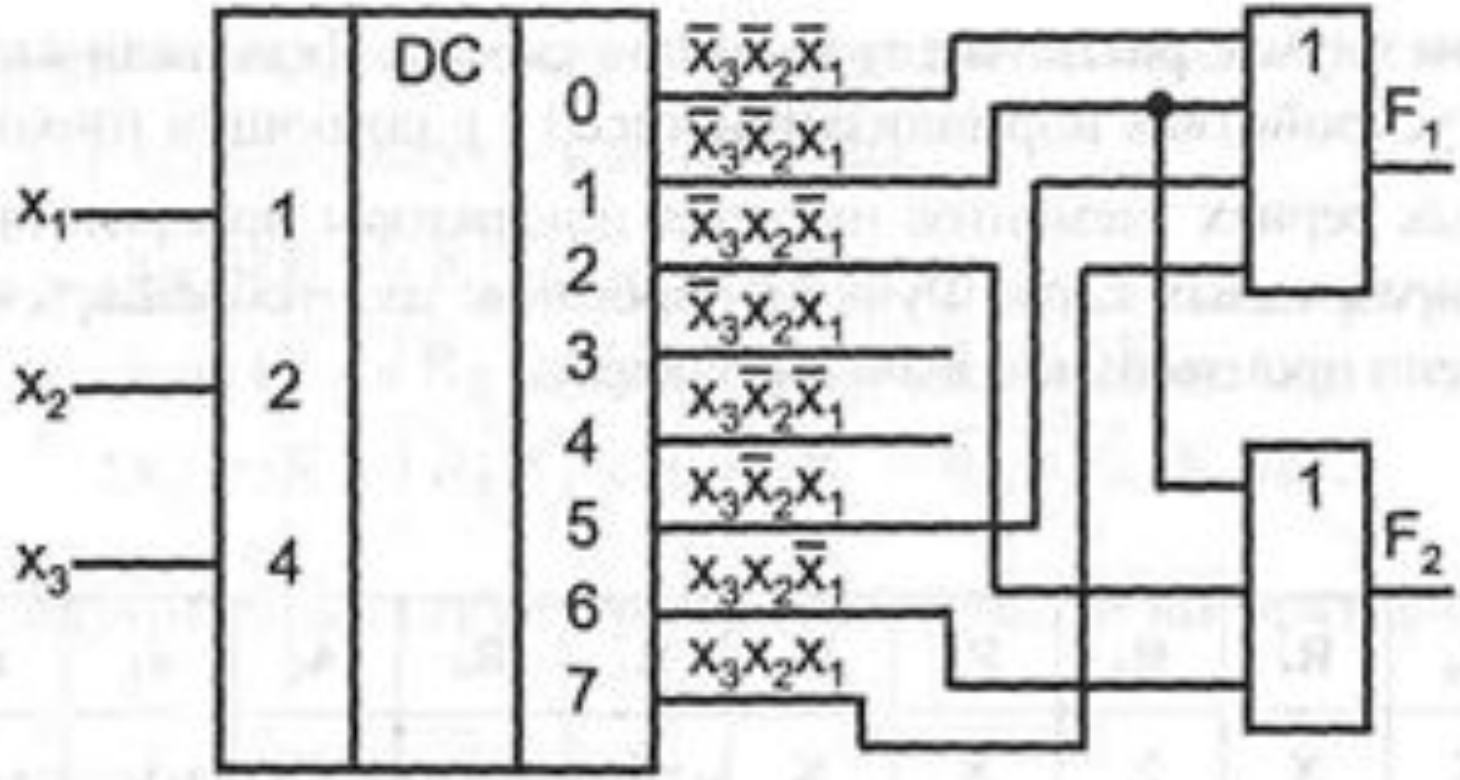


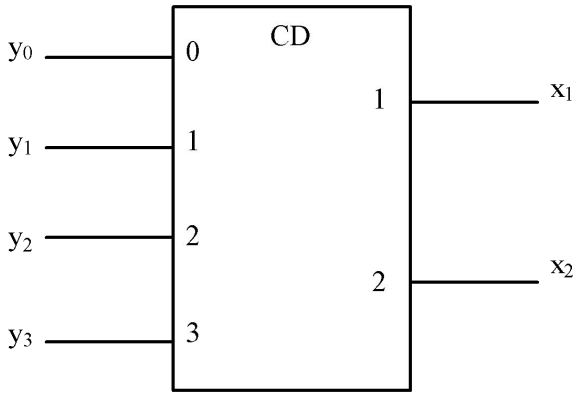
Схема воспроизведения произвольных логических функций с помощью дешифратора и дизъюнкторов

$$F_1 = \bar{x}_3\bar{x}_2 \vee x_3x_1 \text{ и } F_2 = \bar{x}_3\bar{x}_2x_1 \vee x_2\bar{x}_1.$$

Шифраторы

- ***Шифратором*** называется КЦУ, которое преобразует двоичные слова из унитарного кода в позиционный.

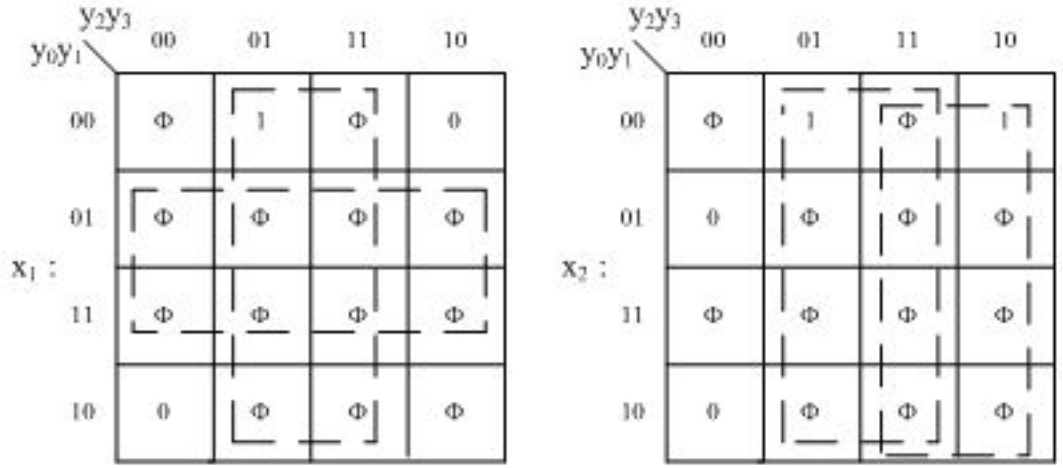
  Таким образом, шифрация является операцией, обратной дешифрации. При возбуждении одного из входов шифратора на его выходе формируется двоичный код номера возбужденной входной линии. Двоичный шифратор имеет 2^n входов и n выходов.
- ***Приоритетные шифраторы*** выполняют более сложную операцию. При работе ЭВМ и в других устройствах часто решается задача определения приоритетного претендента на использование какого-либо ресурса. Несколько конкурентов выставляют свои запросы на обслуживание, которые не могут быть удовлетворены одновременно. Нужно выбрать того, кому предоставляется право первоочередного обслуживания. Простейший вариант решения указанной задачи – присвоение каждому источнику запросов фиксированного уровня приоритета. Например, группа восьми запросов R_7-R_0 (R от англ Request) формируется так, что высший приоритет имеет источник R_7, а далее уровень приоритета уменьшается от номера к номеру. Самый младший приоритет у источника R_0: он будет обслуживаться только при отсутствии всех других запросов. Если имеется одновременно несколько запросов, то обслуживается запрос с наибольшим номером *Приоритетный шифратор вырабатывает на выходе двоичный номер старшего запроса.*



Логические аргументы				Логические функции	
y_0	y_1	y_2	y_3	x_2	x_1
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

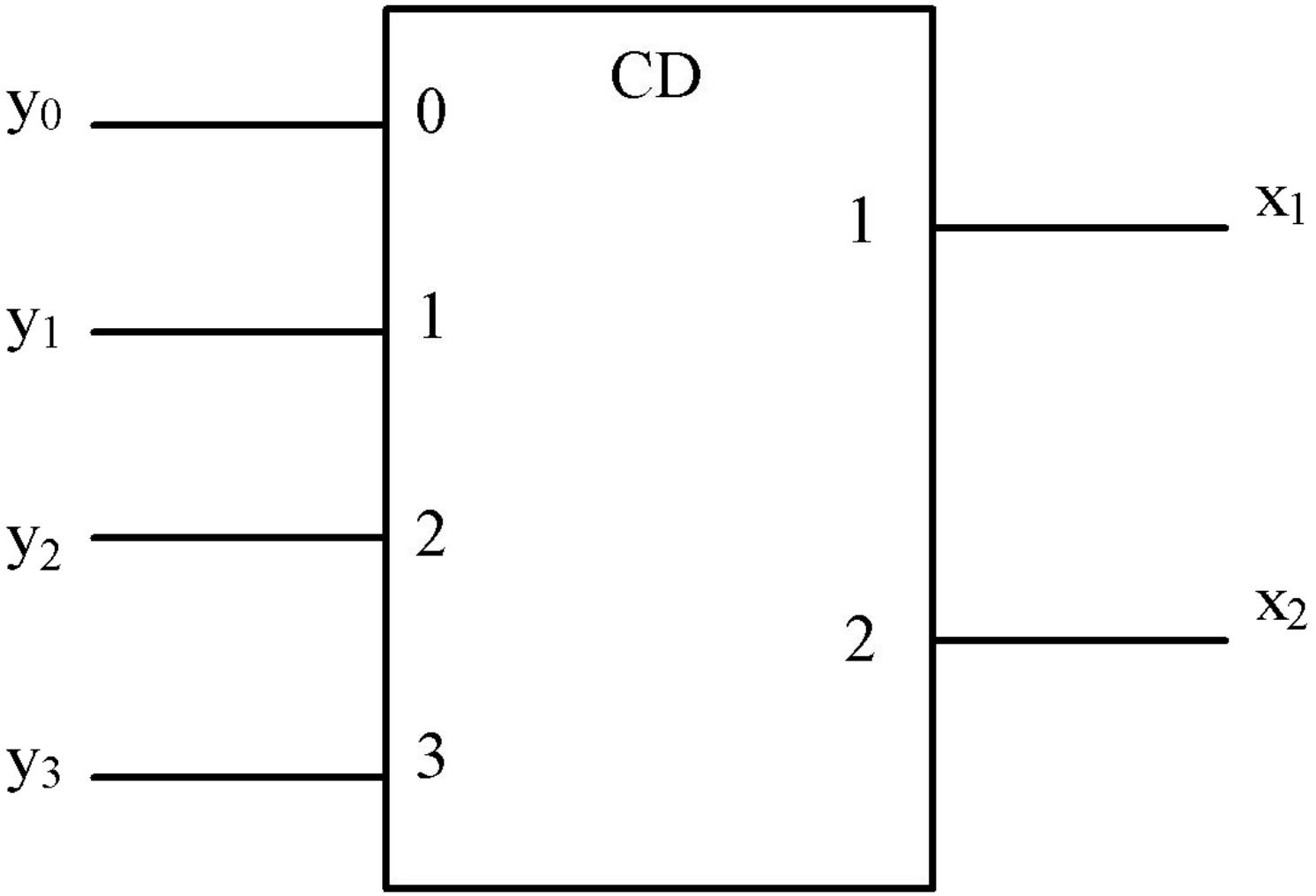
$$\left.\begin{aligned} x_1 &= \overline{y_0}\cdot y_1\cdot\overline{y_2}\cdot\overline{y_3} \vee \overline{y_0}\cdot\overline{y_1}\cdot\overline{y_2}\cdot y_3; \\ x_2 &= \overline{y_0}\cdot\overline{y_1}\cdot y_2\cdot\overline{y_3} \vee \overline{y_0}\cdot\overline{y_1}\cdot\overline{y_2}\cdot y_3. \end{aligned}\right\}$$

Очевидно, что в таблице двенадцать наборов аргументов запрещены, следовательно, значения функций x_1 и x_2 на этих наборах являются неопределенными. Как известно, наличие неопределенностей может привести к улучшению минимизации функций. Построим карты Карно для функций x_1 и x_2



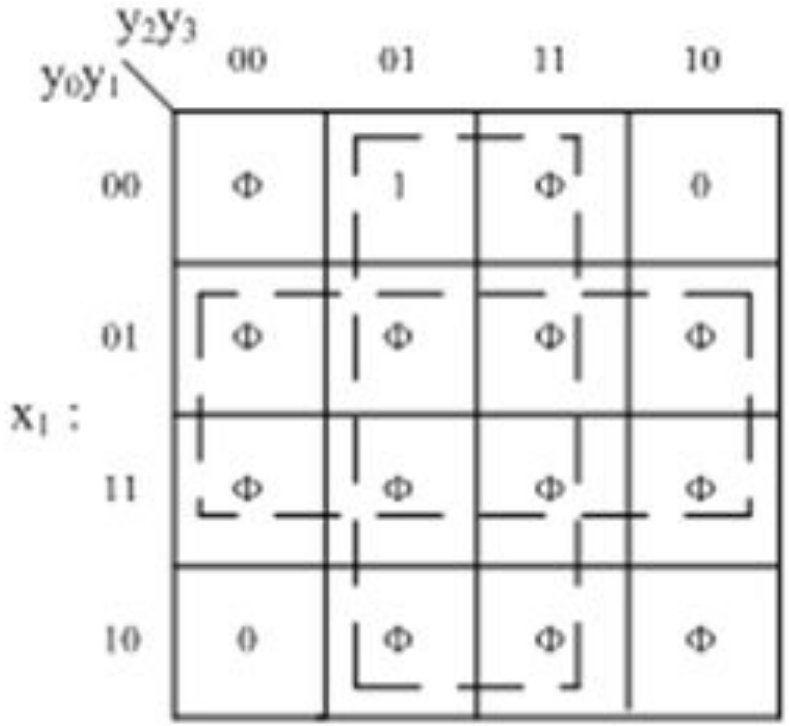
Карты Карно позволяют получить минимальные формы исходных функций:

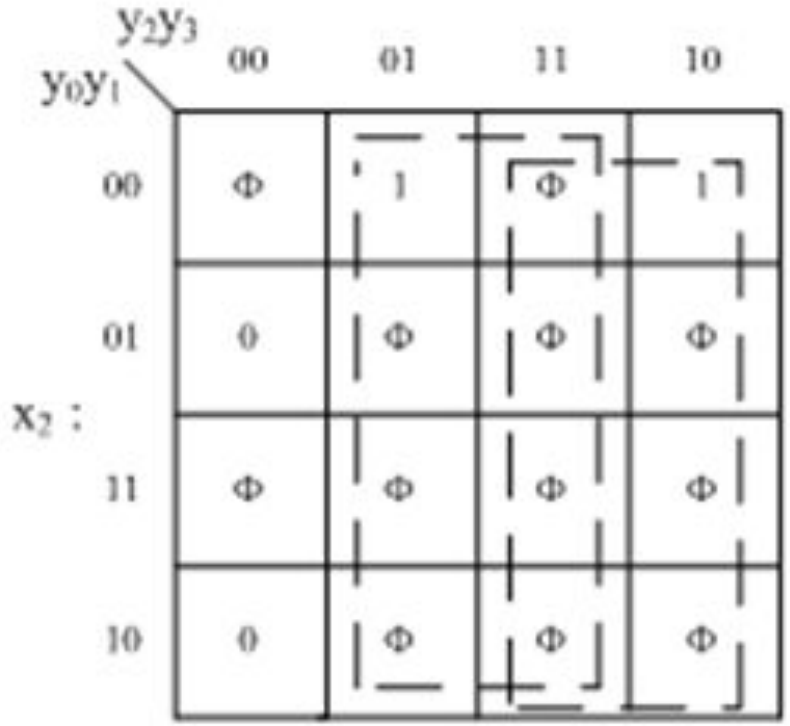
$$\left.\begin{aligned} x_1 &= y_1 \vee y_3; \\ x_2 &= y_2 \vee y_3. \end{aligned}\right\}$$


CD
y0
y1
y2
y3
0
1
2
3
1
2
x1
x2

Логические аргументы				Логические функции	
y_0	y_1	y_2	y_3	x_2	x_1
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

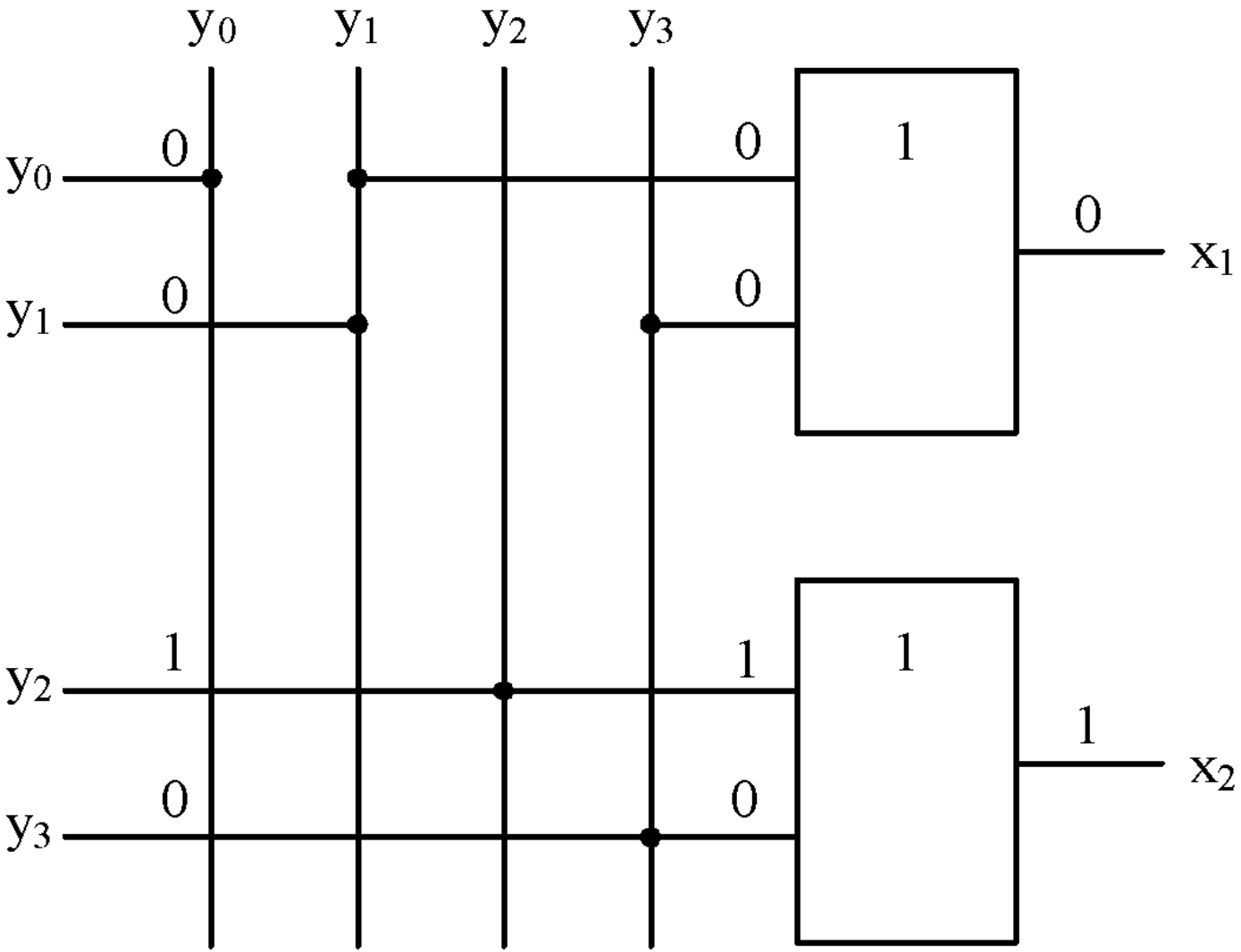
$$\left.\begin{aligned} x_1 &= \overline{y_0}\cdot y_1\cdot\overline{y_2}\cdot\overline{y_3}\vee\overline{y_0}\cdot\overline{y_1}\cdot\overline{y_2}\cdot y_3;\\ x_2 &= \overline{y_0}\cdot\overline{y_1}\cdot y_2\cdot\overline{y_3}\vee\overline{y_0}\cdot\overline{y_1}\cdot\overline{y_2}\cdot y_3. \end{aligned}\right\}$$

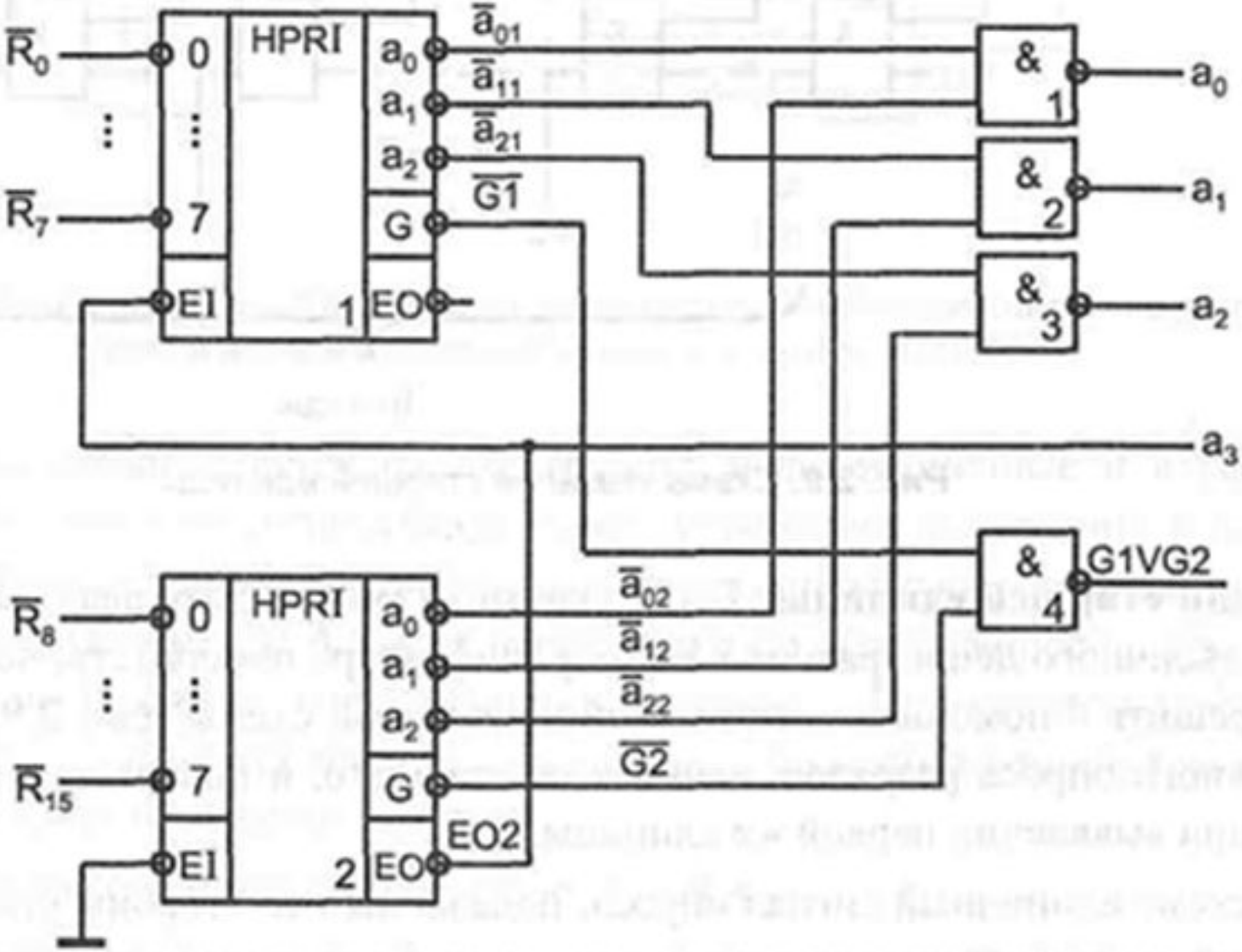




Карты Карно позволяют получить минимальные формы исходных функций:

$$\left.\begin{aligned} x_1 &= y_1 \vee y_3; \\ x_2 &= y_2 \vee y_3. \end{aligned}\right\}$$


y0
y1
y2
y3
0
0
1
0
0
1
1
0
1
1
0
0
1
x1
x2

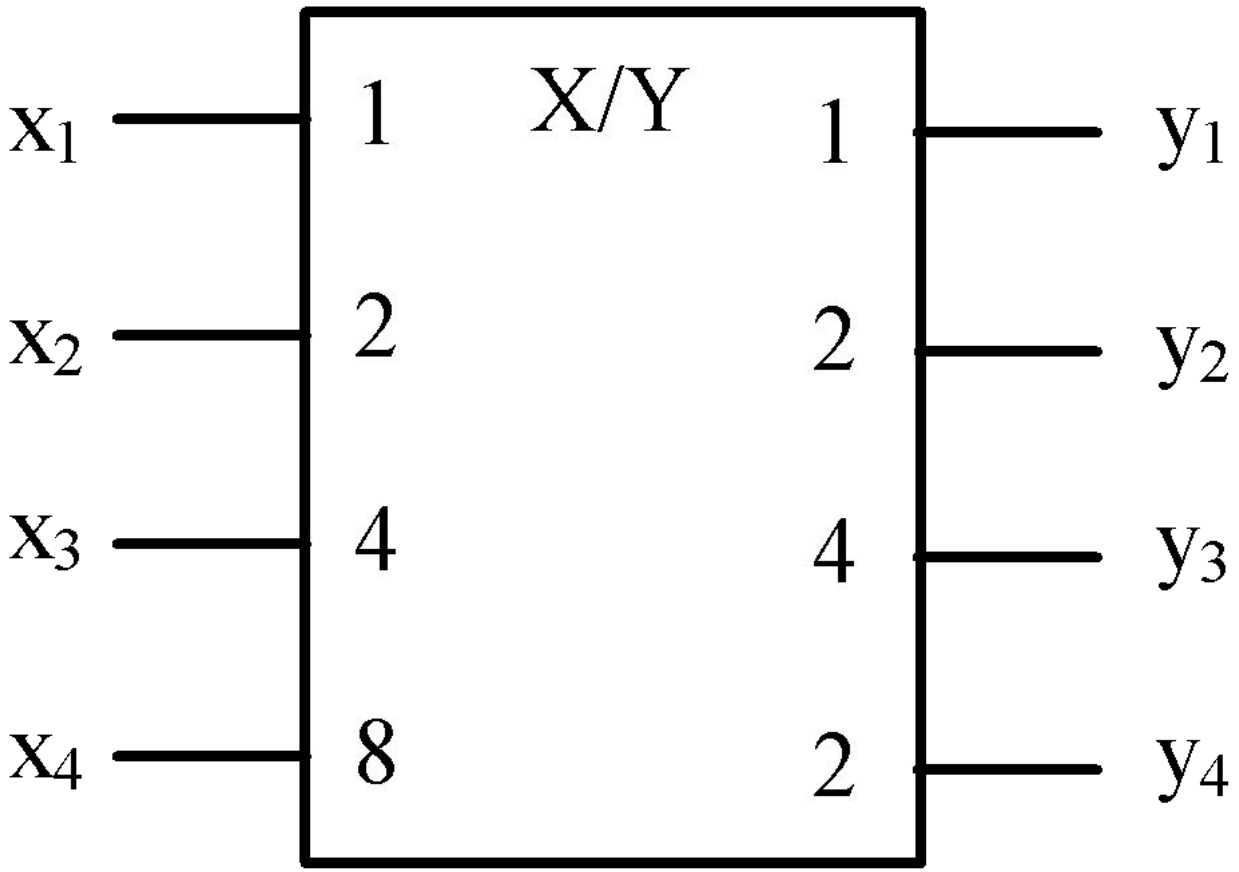

$\overline{R}_0$
$\overline{R}_7$
0
7
EI
HPRI
1
a_0
a_1
a_2
G
EO
$\overline{a}_{01}$
$\overline{a}_{11}$
$\overline{a}_{21}$
$\overline{G1}$
&
1
2
3
a_0
a_1
a_2
a_3
$\overline{R}_8$
$\overline{R}_{15}$
$\overline{a}_{02}$
$\overline{a}_{12}$
$\overline{a}_{22}$
$\overline{G2}$
EO2
4
G1VG2

EI — сигнала разрешения работы данного шифратора; EO — сигнала, вырабатываемого на выходе данного шифратора при отсутствии запросов на его входах для разрешения работы следующего (младшего) шифратора при наращивании размерности шифраторов; G — сигнала, отмечающего наличие запросов на входе данного шифратора; $R_7 ... R_0$ — запросов на входах шифратора; $a_2 ... a_0$ — значений разрядов выходного двоичного кода, формирующего номер старшего запроса. Все перечисленные сигналы формируются при условии EI = 1 (работа шифратора разрешена). При EI = 0 независимо от состояний входов запросов все выходные сигналы шифратора становятся нулевыми.

Преобразователи кодов

- ***Преобразователем кодов (ПК)*** называют КЦУ, которое преобразует n-разрядные двоичные входные слова в m-разрядные двоичные выходные слова. Иногда ПК называют *n, m-преобразователями*.
- Рассмотренные ранее дешифраторы и шифраторы являются ПК некоторых частных видов, например, их можно использовать для преобразования чисел из одной позиционной системы счисления в другую.

Условное графическое обозначение
преобразователя кода 8421 в код 2421



Условное графическое обозначение преобразователя кода 8421 в код 2421

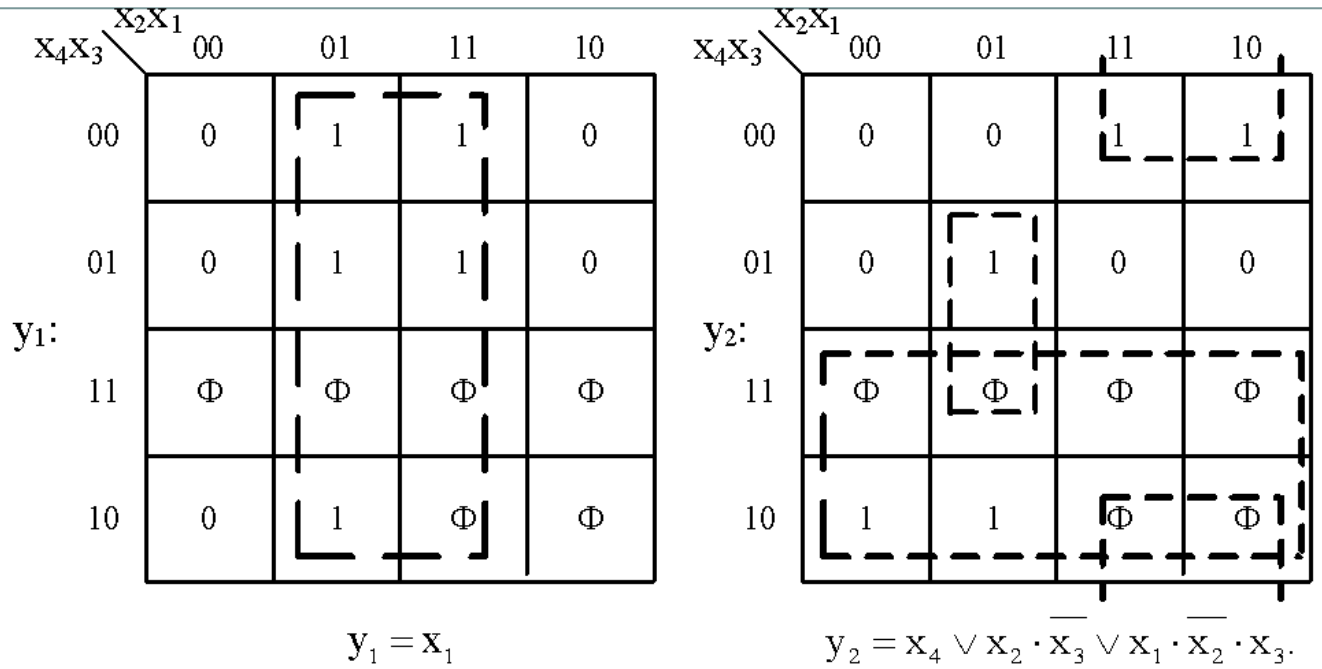
Таблица истинности преобразователя кода 8421 в код 2421

Логические аргументы				Промежуточная переменная	Логические функции			
x_4	x_3	x_2	x_1	z_i	y_4	y_3	y_2	y_1
0	0	0	0	z_0	0	0	0	0
0	0	0	1	z_1	0	0	0	1
0	0	1	0	z_2	0	0	1	0
0	0	1	1	z_3	0	0	1	1
0	1	0	0	z_4	0	1	0	0
0	1	0	1	z_5	1	0	1	1
0	1	1	0	z_6	1	1	0	0
0	1	1	1	z_7	1	1	0	1
1	0	0	0	z_8	1	1	1	0
1	0	0	1	z_9	1	1	1	1

Имея таблицу истинности, можно использовать три подхода к синтезу ПК:

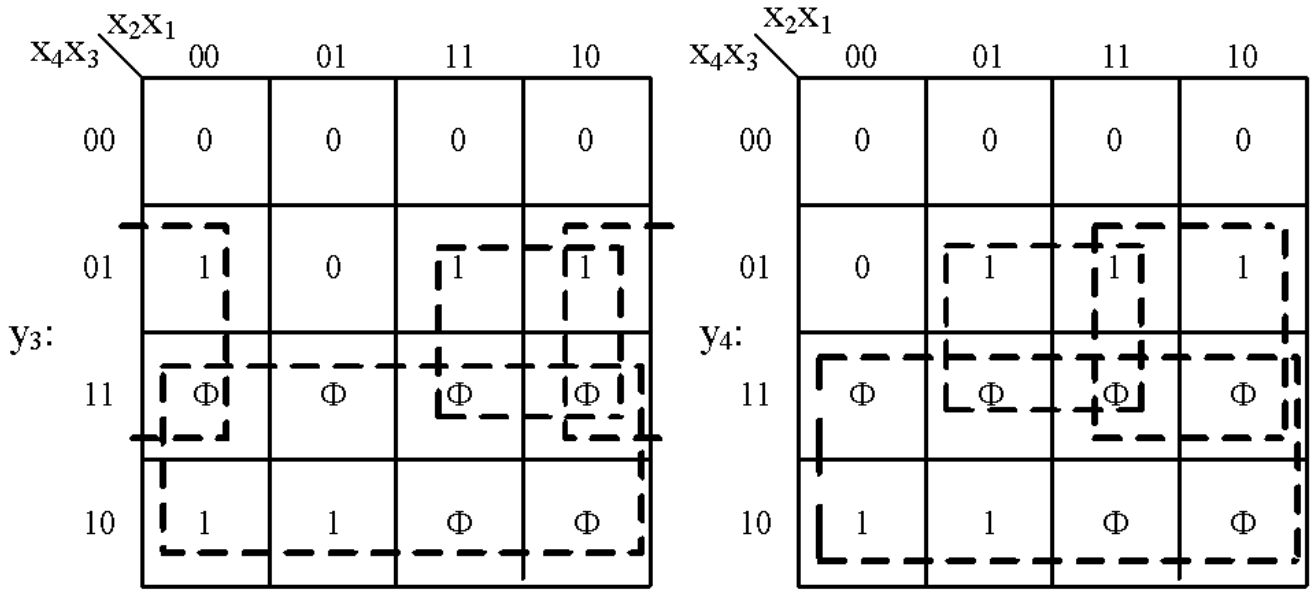
- преобразователь синтезируется как однокомпонентная минимизированная комбинационная схема с нерегулярной структурой (по общим правилам синтеза КЦУ);
- преобразователь синтезируется как слабо минимизированная комбинационная схема с частично регулярной структурой (на основе шифратора и дешифратора);
- преобразователь синтезируется как неминимизированная комбинационная схема с регулярной структурой (на основе постоянного запоминающего устройства).

Рассмотрим *первый традиционный подход*. По данным таблицы заполним карты Карно



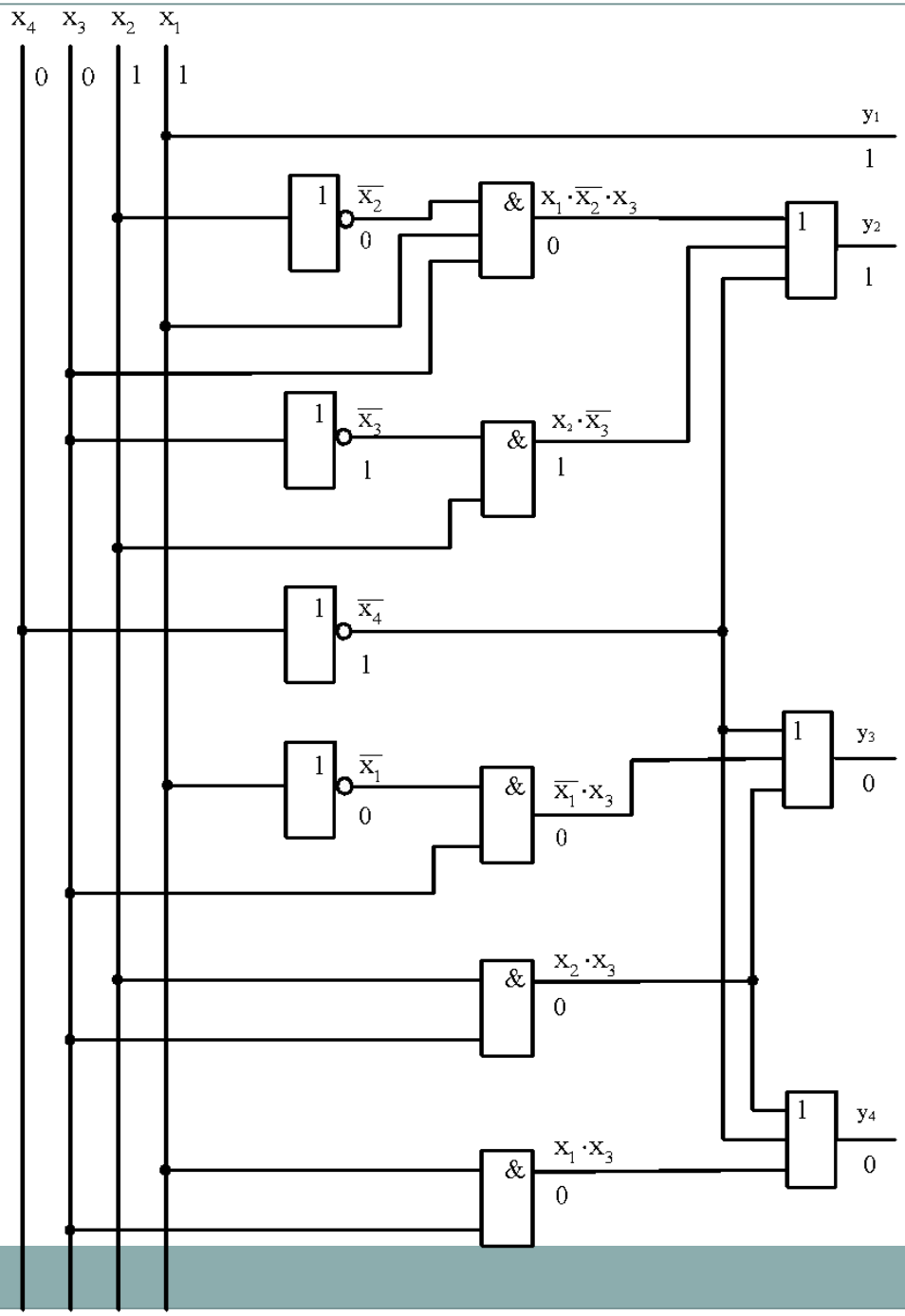
$y_1 = x_1$

$y_2 = x_4 \vee x_2 \cdot \overline{x_3} \vee x_1 \cdot \overline{x_2} \cdot x_3.$



$y_3 = x_4 \vee x_2 \cdot x_3 \vee \overline{x_1} \cdot x_3.$

$y_4 = x_4 \vee x_1 \cdot x_3 \vee x_2 \cdot x_3.$

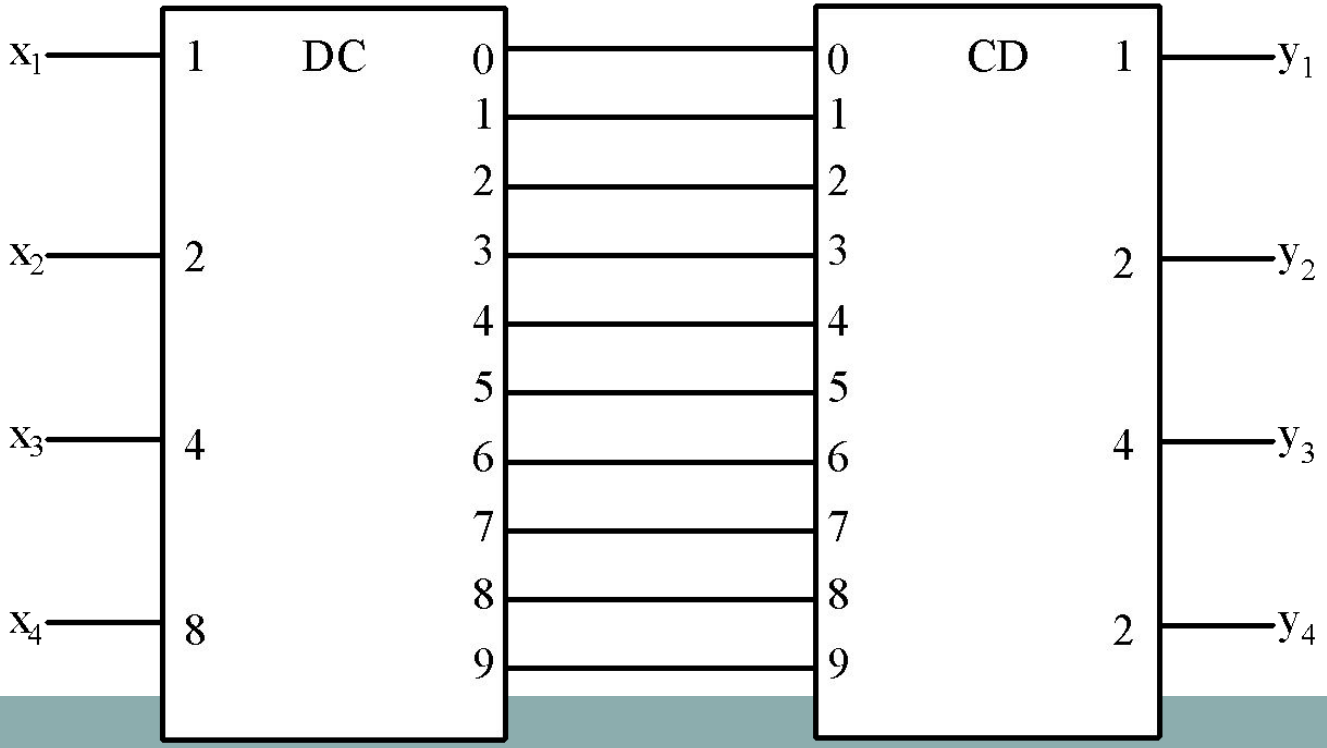

x4 x3 x2 x1
0 0 1 1
y1
1
1
x2
0
&
x1 · x2 · x3
0
1
y2
1
1
x3
1
&
x2 · x3
1
1
x4
1
1
x1
0
&
x1 · x3
0
1
y3
0
&
x2 · x3
0
1
y4
0
&
x1 · x3
0

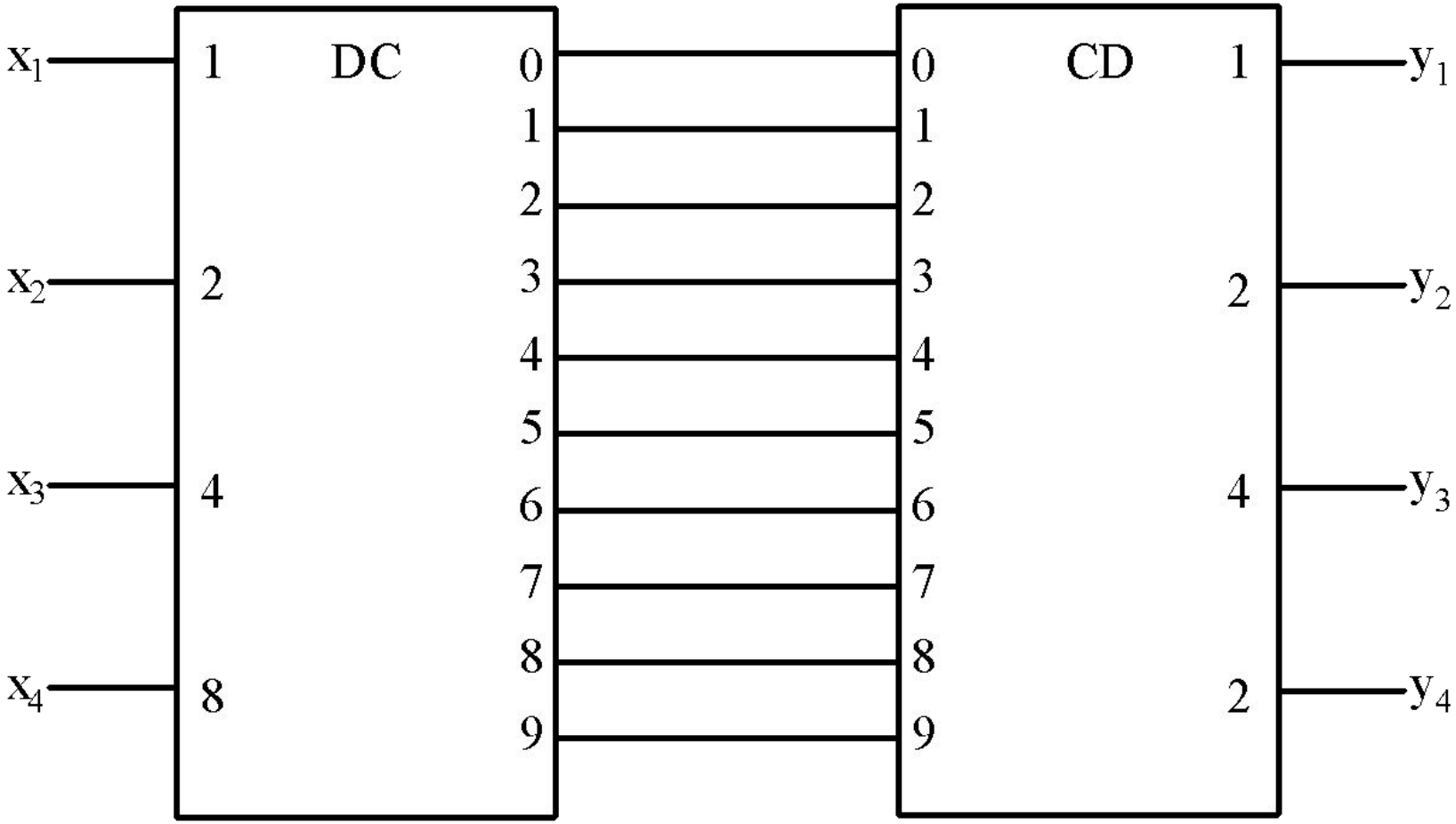
- К достоинству первого подхода относится экономичность синтезируемой схемы по аппаратурным затратам, исчисляемым в условных транзисторах. Это означает, что схема будет занимать небольшую часть площади кристалла. Оценка схемы на рисунке 12 дает величину $E_{ПК\ (1)} = 23$ условных транзистора.
- К недостатку этого подхода можно отнести то, что схема получилась нерегулярной (с неравным числом конъюнкторов в цепи каждого выхода, с перекрещивающимися связями), что делает ее нетехнологичной при изготовлении, неудобной для тестовых проверок.

- *Второй подход* позволяет повысить регулярность структур ПК за счет некоторого увеличения аппаратурных затрат.

  Схема ПК в данном случае приобретает двухкомпонентную структуру вида «десятичный дешифратор-шифратор». Она несколько сложнее предыдущей схемы , но значительно проще для обозрения.

  К недостатку полученной схемы следует отнести ее специализированность, что снижает массовость выпуска подобных схем и приводит к относительно высокой цене изделия.

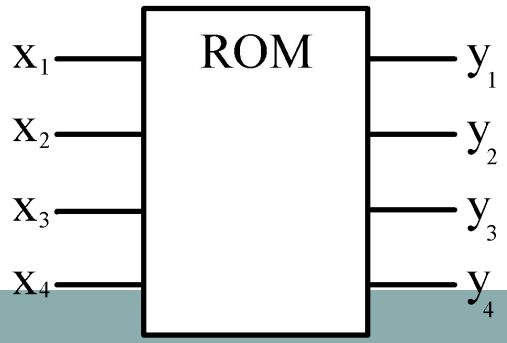


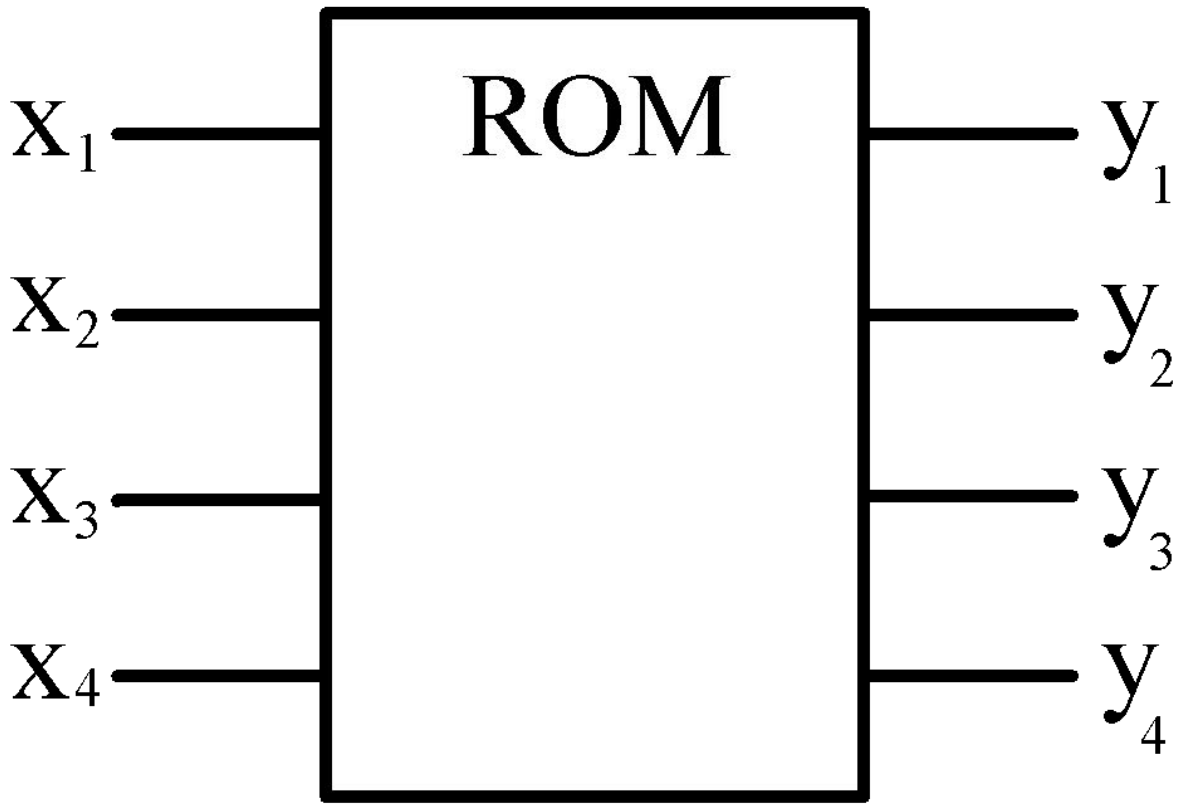

x1
x2
x3
x4
1
2
4
8
DC
0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9
CD
1
2
4
2
y1
y2
y3
y4

Третий подход позволяет значительно повысить регулярность структуры ПК и одновременно существенно расширить его функциональные возможности. В этом случае запрещается использовать специализированные компоненты. Таким образом, ПК должен содержать полный двоичный n-входной дешифратор и 2^n-входной шифратор. В результате получается постоянное запоминающее устройство (ПЗУ). Следовательно, ПЗУ – это n, m-преобразователь с двухкомпонентной регулярной структурой, на выходе которого включен шифратор, формирующий m-разрядные слова.

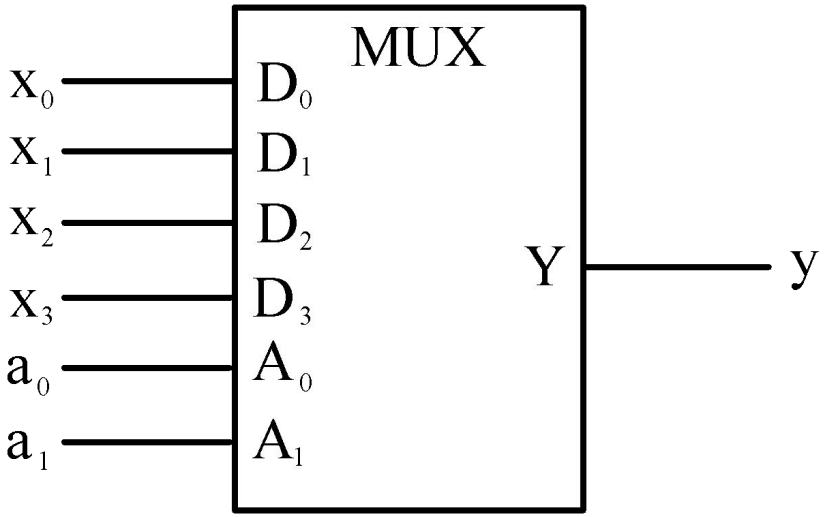
Входное слово ПК в этом случае является адресом ячейки ПЗУ, в которой хранится соответствующее выходное слово. Оно представляет собой прямоугольник с n-входами и m-выходами, во внутреннем поле которого записана аббревиатура ROM (от англ. Read Only Memory).

ПЗУ имеет целый ряд недостатков (невозможность обновления записанной информации, аппаратурная избыточность при реализации тех или иных ПК и др.), однако в цифровой схемотехнике они очень широкого применяются благодаря широким функциональным возможностям (хранение констант, микропрограмм, программ начальной загрузки, кодопреобразование, выполнение арифметических и логических операций), регулярности структуры, а следовательно, высокой технологичности их изготовления.




ROM
x1
x2
x3
x4
y1
y2
y3
y4

Мультиплексоры



- ***Мультиплексором*** называется КЦУ, которое обеспечивает альтернативную (поочередную) передачу данных от нескольких источников одному приемнику. Эта операция коммутации каналов называется ***мультиплексированием.*** Если требование альтернативности отсутствует, то задача мультиплексирования вырождается в случай логического сложения данных. При m источниках информации мультиплексор должен иметь m информационных входов, $k=\log_2 m$ адресных входов и один информационный выход. Разрядности каналов передачи могут быть различными, мультиплексоры для коммутации многоразрядных слов составляются из одноразрядных.
- Принцип построения одноразрядных мультиплексоров рассмотрим на примере синтеза мультиплексора на четыре информационных входа (m=4). УГО такого мультиплексора представляет собой прямоугольник с аббревиатурой MUX (от англ. Multiplexer) во внутреннем поле.
- Входы A_1, A_0 служат для приема адреса источника, от которого подается информация в данный момент.

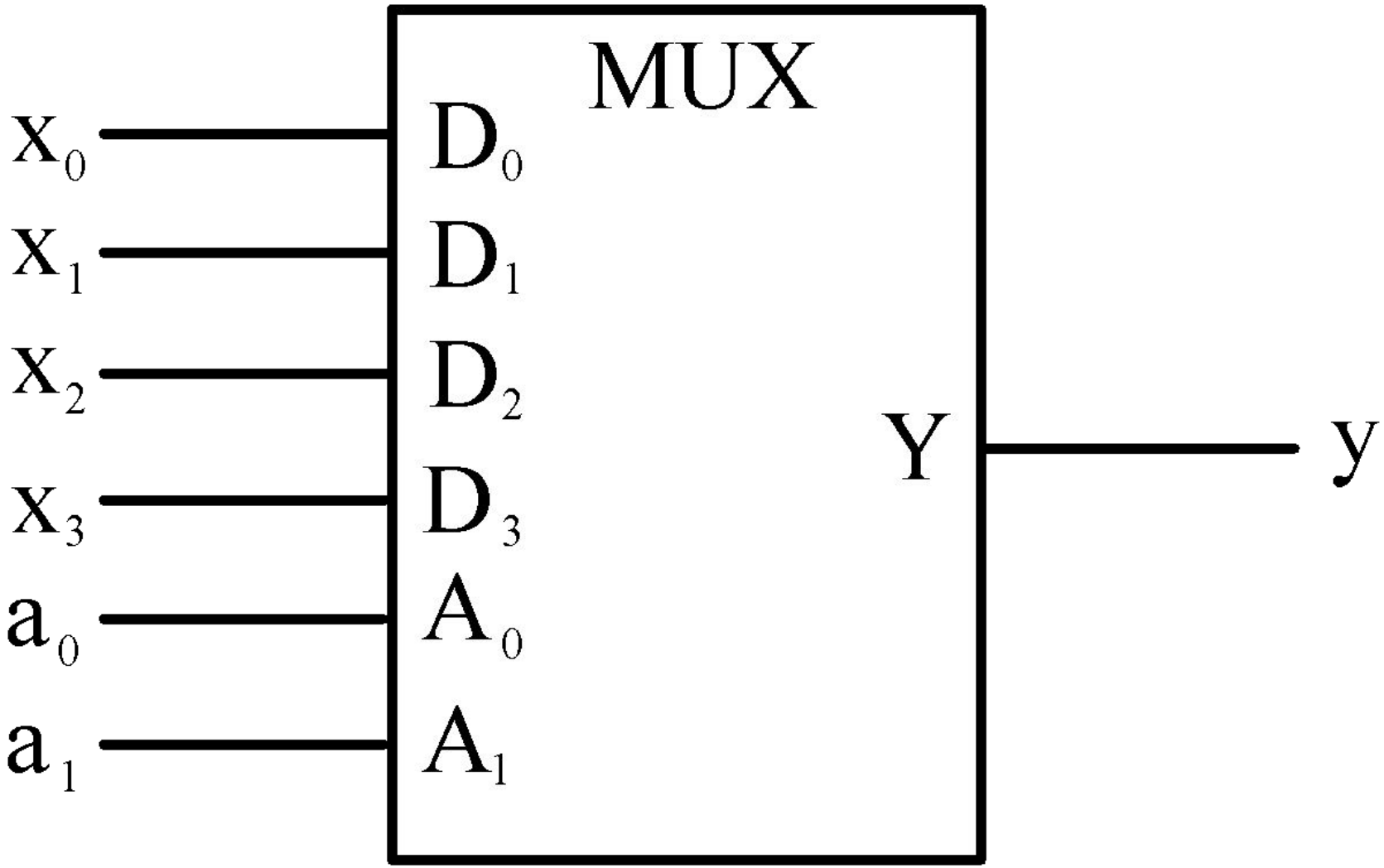
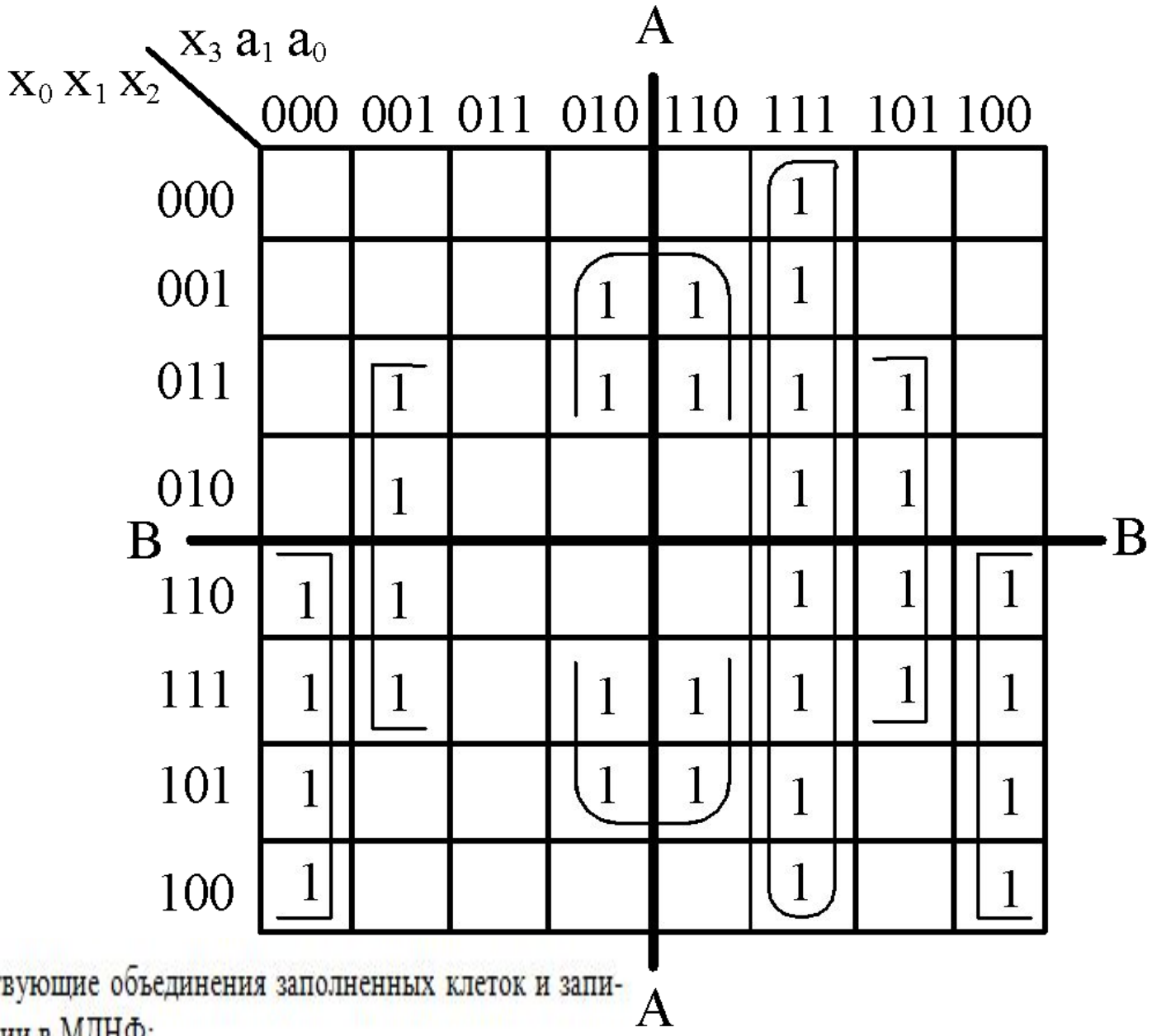

MUX
x_0
x_1
x_2
x_3
a_0
a_1
D_0
D_1
D_2
D_3
A_0
A_1
Y
y

Таблица истинности одноразрядного мультиплексора для m=4

Логические аргументы						Логическая функция
x_0	x_1	x_2	x_3	a_1	a_0	y
0	x	x	x	0	0	0
1	x	x	x	0	0	1
x	0	x	x	0	1	0
x	1	x	x	0	1	1
x	x	0	x	1	0	0
x	x	1	x	1	0	1
x	x	x	0	1	1	0
x	x	x	1	1	1	1

сигнал на выходе *y* является логической функцией шести аргументов, следовательно, в СДНФ эта функция содержит 32 конституенты единицы



Выполним соответствующие объединения заполненных клеток и запишем результат минимизации в МДНФ:

$$y = x_0 \cdot \overline{a_1} \cdot \overline{a_0} \vee x_1 \cdot \overline{a_1} \cdot a_0 \vee x_2 \cdot a_1 \cdot \overline{a_0} \vee x_3 \cdot a_1 \cdot a_0.$$

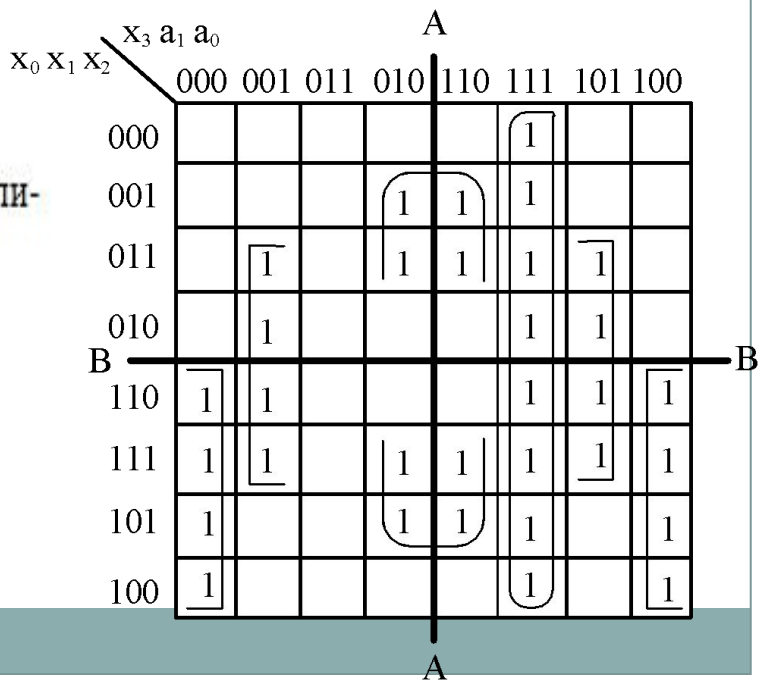
Выполним соответствующие объединения заполненных клеток и запишем результат минимизации в МДНФ:

$$y = x_0 \cdot \overline{a_1} \cdot \overline{a_0} \vee x_1 \cdot \overline{a_1} \cdot a_0 \vee x_2 \cdot a_1 \cdot \overline{a_0} \vee x_3 \cdot a_1 \cdot a_0.$$

Таблица истинности одноразрядного мультиплексора для m=4

Логические аргументы						Логическая функция
x_0	x_1	x_2	x_3	a_1	a_0	y
0	x	x	x	0	0	0
1	x	x	x	0	0	1
x	0	x	x	0	1	0
x	1	x	x	0	1	1
x	x	0	x	1	0	0
x	x	1	x	1	0	1
x	x	x	0	1	1	0
x	x	x	1	1	1	1

сигнал на выходе *y* является логической функцией шести аргументов, следовательно, в СДНФ эта функция содержит 32 конституенты единицы



Выполним соответствующие объединения заполненных клеток и запишем результат минимизации в МДНФ:

$$y = x_0 \cdot \overline{a_1} \cdot \overline{a_0} \vee x_1 \cdot \overline{a_1} \cdot a_0 \vee x_2 \cdot a_1 \cdot \overline{a_0} \vee x_3 \cdot a_1 \cdot a_0.$$

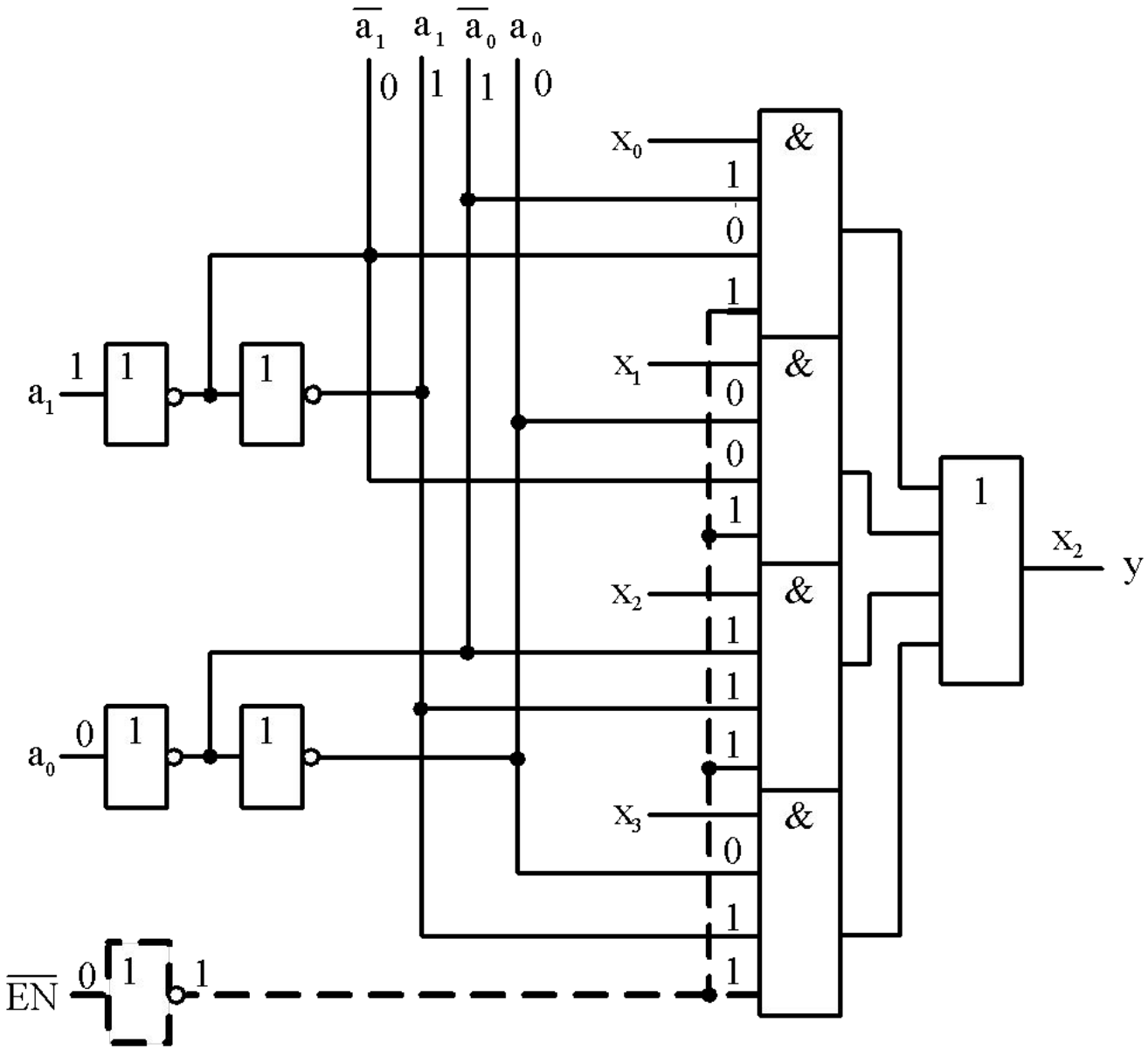
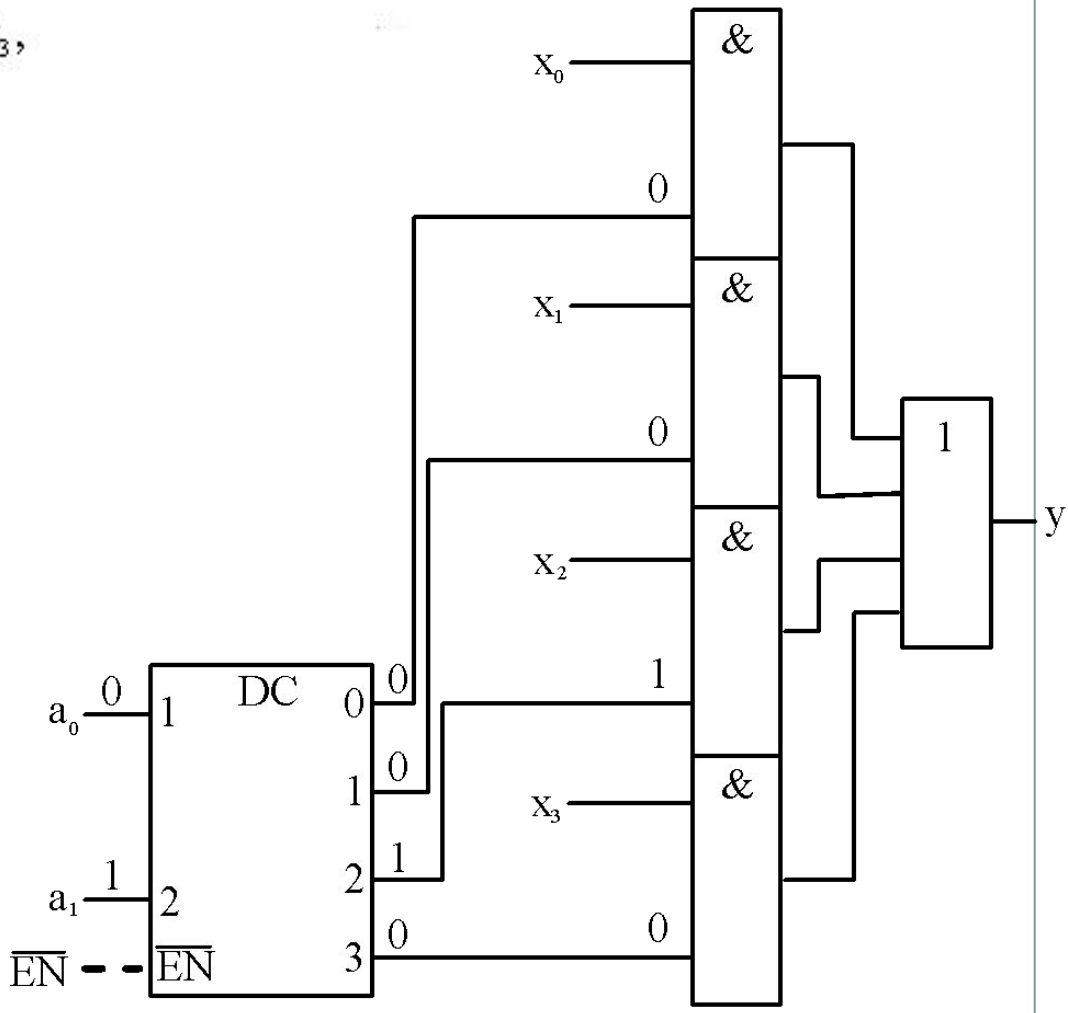

$\overline{a_1}$ a_1 $\overline{a_0}$ a_0
0 1 1 0
x_0
x_1
x_2
x_3
&
1
a_1
a_0
$\overline{EN}$
x_2
y
a)

Схема получилась экономичной по аппаратурным затратам ($E_{МUX} = 20$ условных транзисторов), достаточно быстродействующей ($T_{МUX} = 4\ t_{зд.\ лэ}$), но плохо структурированной.

Для структурирования схемы мультиплексора представим функцию в виде:

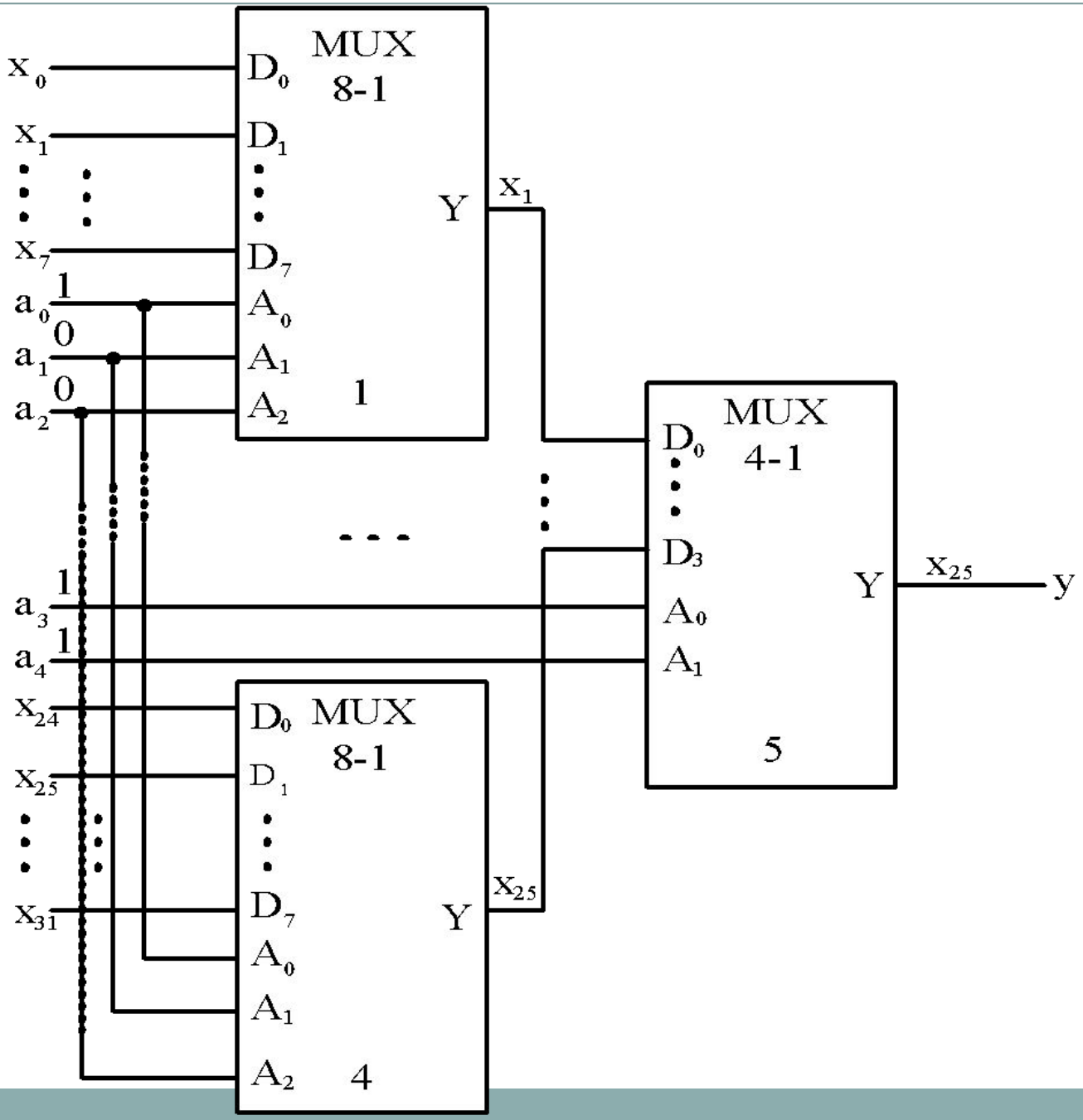
$$y = x_0 \cdot z_0 \vee x_1 \cdot z_1 \vee x_2 \cdot z_2 \vee x_3 \cdot z_3,$$

где $z_0 = \overline{a_1} \cdot \overline{a_0}; z_1 = \overline{a_1} \cdot a_0; z_2 = a_1 \cdot \overline{a_0}; z_3 = a_1 \cdot a_0.$

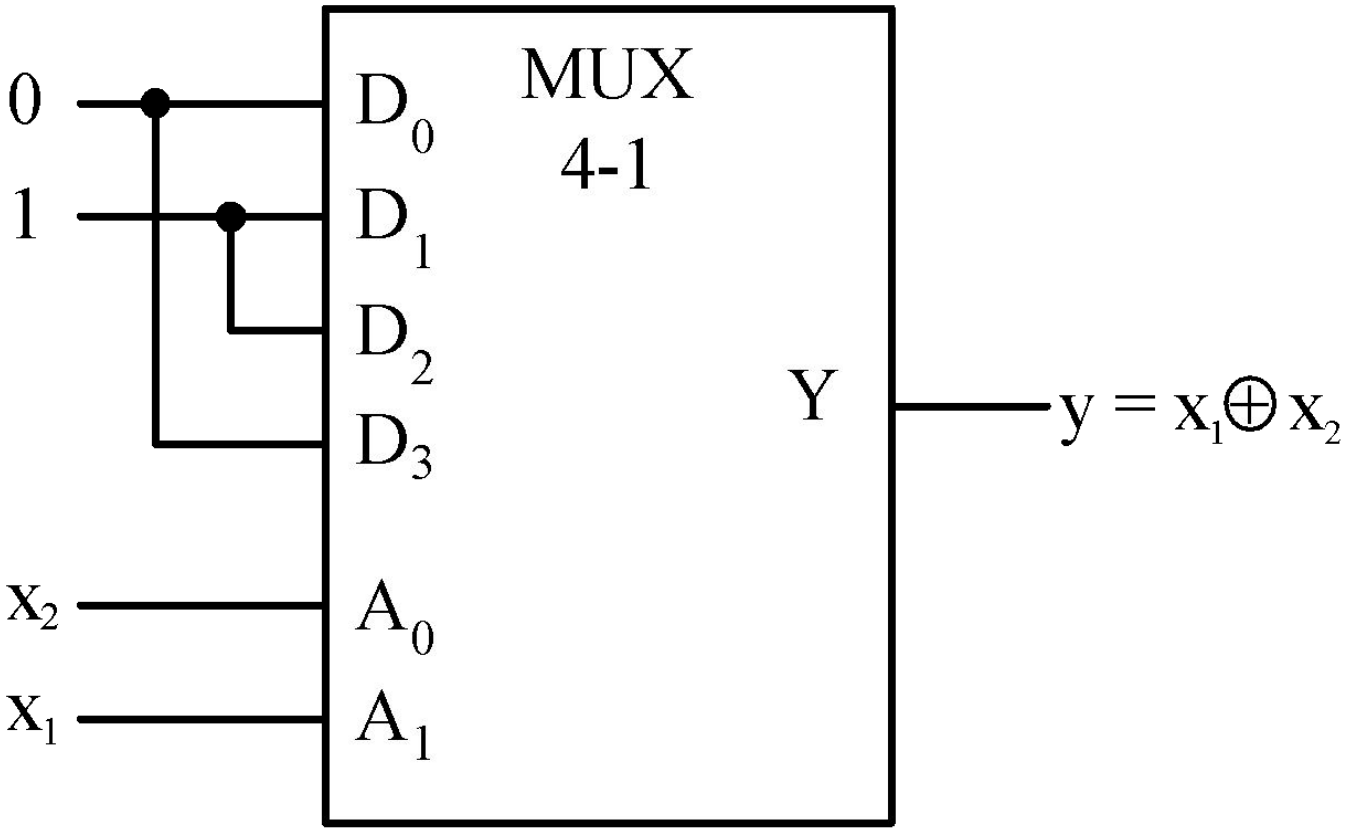


б)

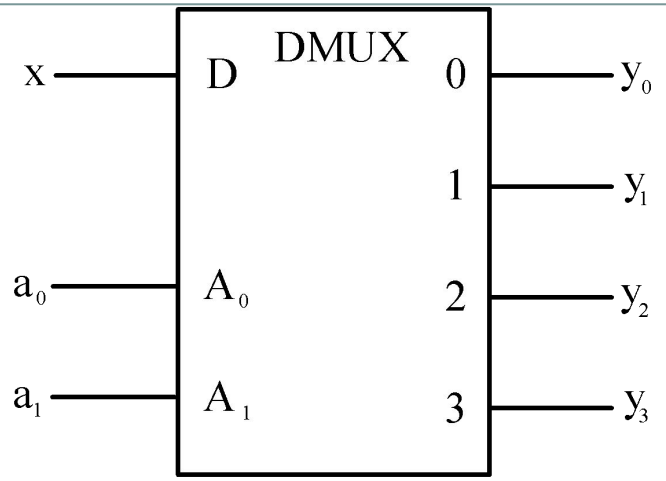
- Аппаратурные затраты на реализацию структурированного варианта мультиплексора составляют 24 условных транзистора, а быстродействие оценивается величиной 5 $t_{зд. ЛЭ}$.
- В стандартных сериях интегральных схем число информационных входов мультиплексоров m не более 16. Для наращивания числа информационных входов строят пирамидальную структуру из нескольких мультиплексоров с меньшим числом информационных входов, называемую мультиплексорным деревом. При этом первый ярус схемы представляет собой столбец, содержащий столько мультиплексоров, сколько необходимо для получения нужного числа информационных входов m. Все мультиплексоры столбца адресуются младшими разрядами k_1 общего адресного кода ($k_1 = \log_2 m_1$, где m_1 – число информационных входов мультиплексоров первого яруса). Старшие разряды адресного кода, число которых равно $k - k_1$ ($k=\log_2 m$, где m - общее число информационных входов мультиплексорного дерева), используются во втором ярусе, мультиплексор которого обеспечивает поочередную работу мультиплексоров первого яруса на общий выходной канал.


MUX
8-1
x_0
x_1
x_7
D_0
D_1
D_7
Y
x_1
a_0
a_1
a_2
1
0
0
A_0
A_1
A_2
1
MUX
4-1
D_0
D_3
A_0
A_1
Y
x_{25}
y
5
a_3
a_4
1
1
x_{24}
x_{25}
x_{31}
MUX
8-1
D_0
D_1
D_7
A_0
A_1
A_2
Y
x_{25}
4

Универсальные логические модули (УЛМ) на основе мультиплексоров можно использовать для схемотехнической реализации различных логических функций. Универсальность их состоит в том, что для заданного числа аргументов можно настроить УЛМ на любую функцию. Для использования мультиплексора в качестве УЛМ следует изменить назначение его входов. На адресные входы следует подавать аргументы функции, а на информационные входы – сигналы настройки. Действительно, каждому набору аргументов соответствует передача на выход одного из сигналов настройки. Если этот сигнал есть значение функции на данном наборе аргументов, то задача решена.


MUX
4-1
0
1
x_2
x_1
D_0
D_1
D_2
D_3
A_0
A_1
Y
$y = x_1 \oplus x_2$

Демультиплексоры



- ***Демультиплексором*** называется КЦУ, которое обеспечивает альтернативную (поочередную) передачу данных от одного источника нескольким адресатам (приемникам). Эта операция коммутации каналов называется ***демультиплексированием***. При m адресатах демультиплексор должен иметь один информационный вход, $k \geq \log_2 m$ адресных входов и m информационных выходов.
- В дальнейшем будем рассматривать одноразрядные демультиплексоры, осуществляющие обработку (коммутацию) одного бита информации. При необходимости демультиплексирования n-разрядных слов надо использовать n демультиплексоров. Если требование альтернативности отсутствует, то задача демультиплексирования вырождается в случай разветвления электрической цепи.
- УГО одноразрядного демультиплексора для m = 4 представляет собой прямоугольник с аббревиатурой DMUX (от англ. Demultiplexer) во внутреннем поле. Входы A_1,A_0 служат для приема адреса абонента, которому предназначена информация в данный момент.

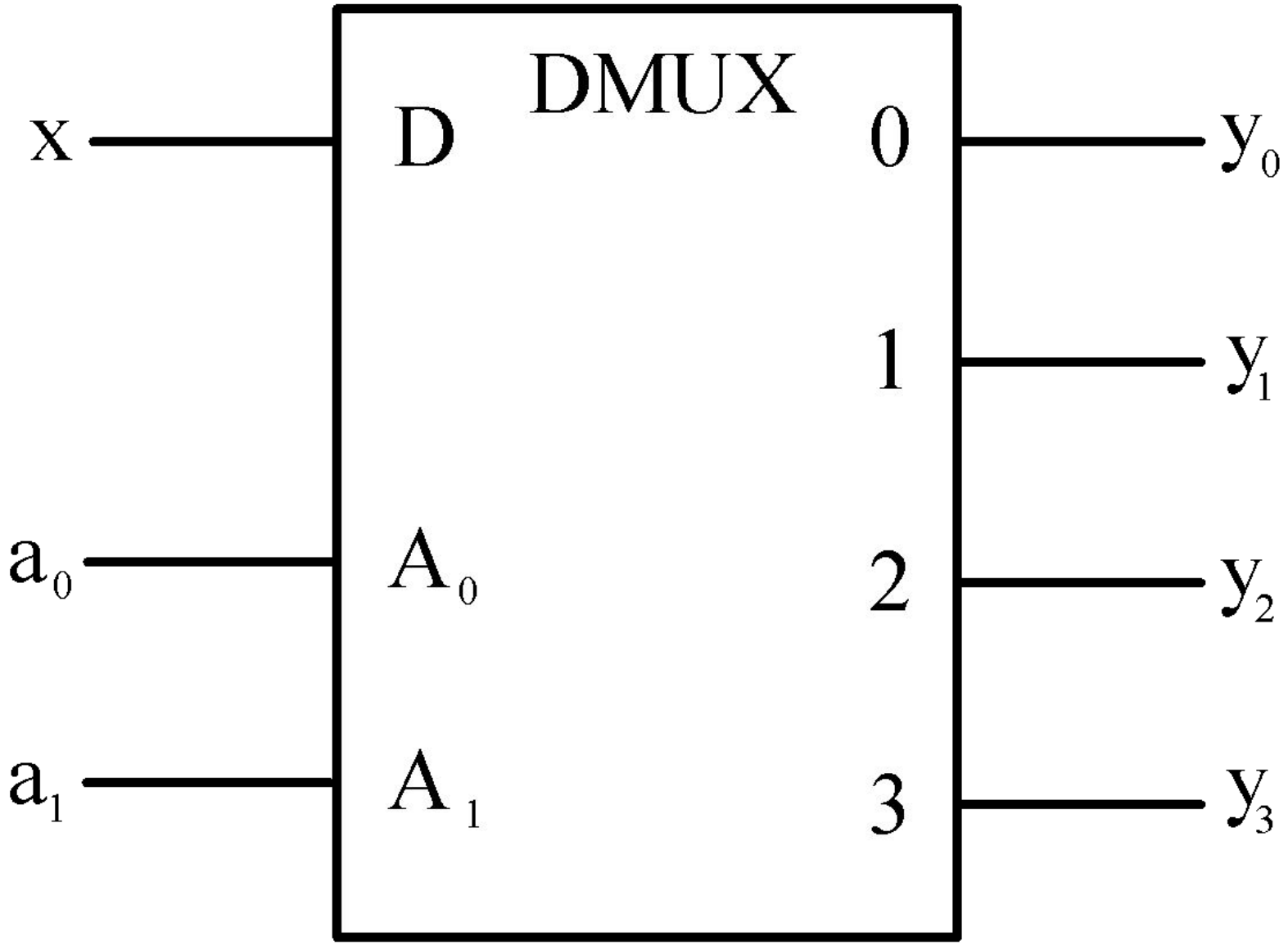

DMUX
x
D
0
y_0
1
y_1
a_0
A_0
2
y_2
a_1
A_1
3
y_3

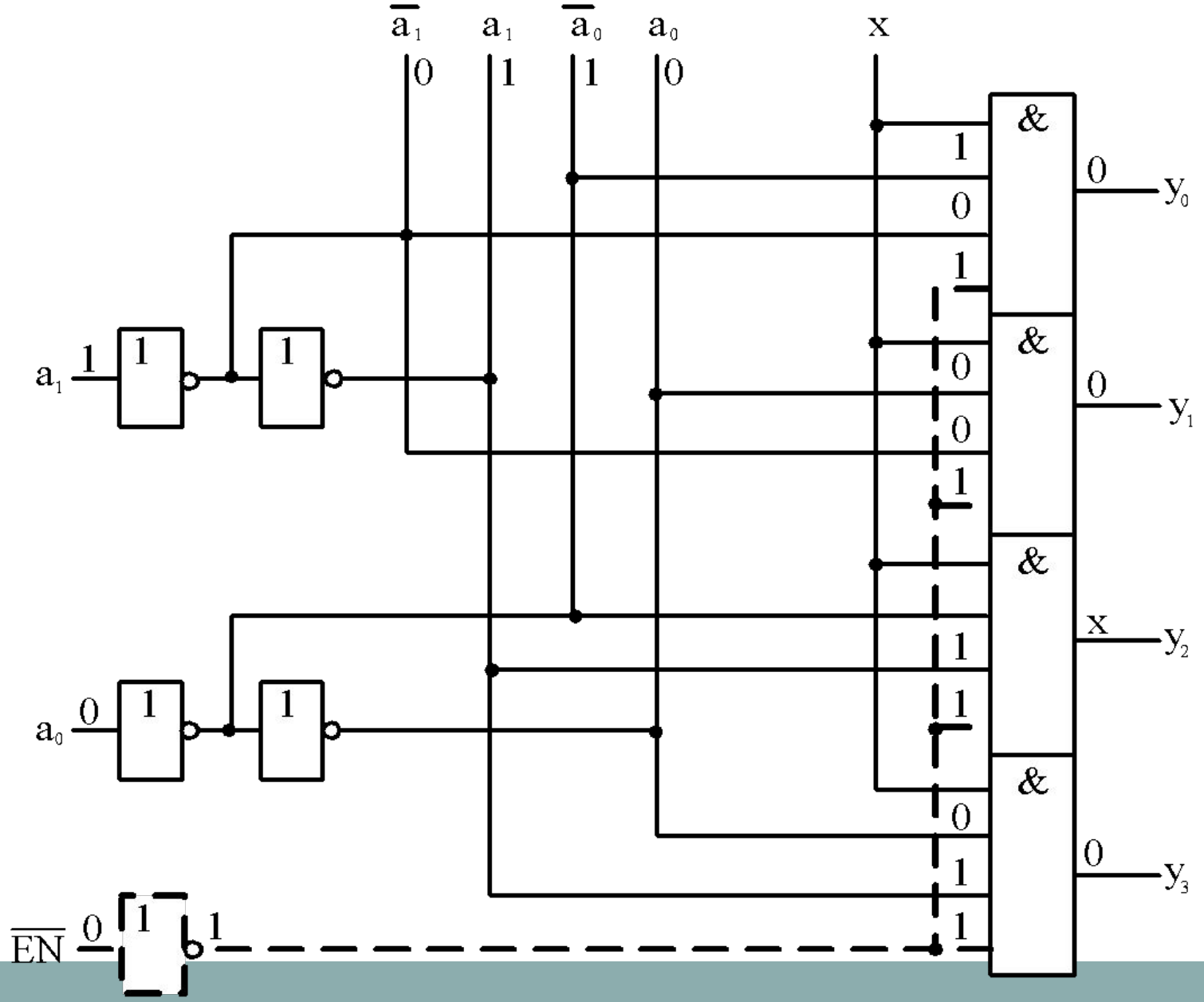
Таблица истинности одноразрядного демультиплексора для m = 4

Логические аргументы			Логические функции			
x	a_1	a_0	y_0	y_1	y_2	y_3
0	0	0	**0**	0	0	0
0	0	1	0	**0**	0	0
0	1	0	0	0	**0**	0
0	1	1	0	0	0	**0**
1	0	0	**1**	0	0	0
1	0	1	0	**1**	0	0
1	1	0	0	0	**1**	0
1	1	1	0	0	0	**1**

$$\left.\begin{aligned} y_0 &= x \cdot \overline{a_1} \cdot \overline{a_0}; \\ y_1 &= x \cdot \overline{a_1} \cdot a_0; \\ y_2 &= x \cdot a_1 \cdot \overline{a_0}; \\ y_3 &= x \cdot a_1 \cdot a_0. \end{aligned}\right\}$$

Функции получились простыми, и минимизация не требуется.

Логическая схема неструктурированная



Анализ схемы показывает, что одноразрядный демультиплексор фактически является двоичным дешифратором (вход X может выполнять функцию входа разрешения). Поэтому в интегральном исполнении обычно выпускаются дешифраторы-демультиплексоры. Схема плохо структурирована, так как в ней нет структурных компонентов промежуточного уровня. Аппаратурные затраты на реализацию такого демультиплексора оцениваются величиной $E_{DMUX} = 16$ условных транзисторов. Быстродействие схемы с учетом инверторов оценивается величиной $T_{DMUX} = 3\ t_{зд.\ лэ}$.

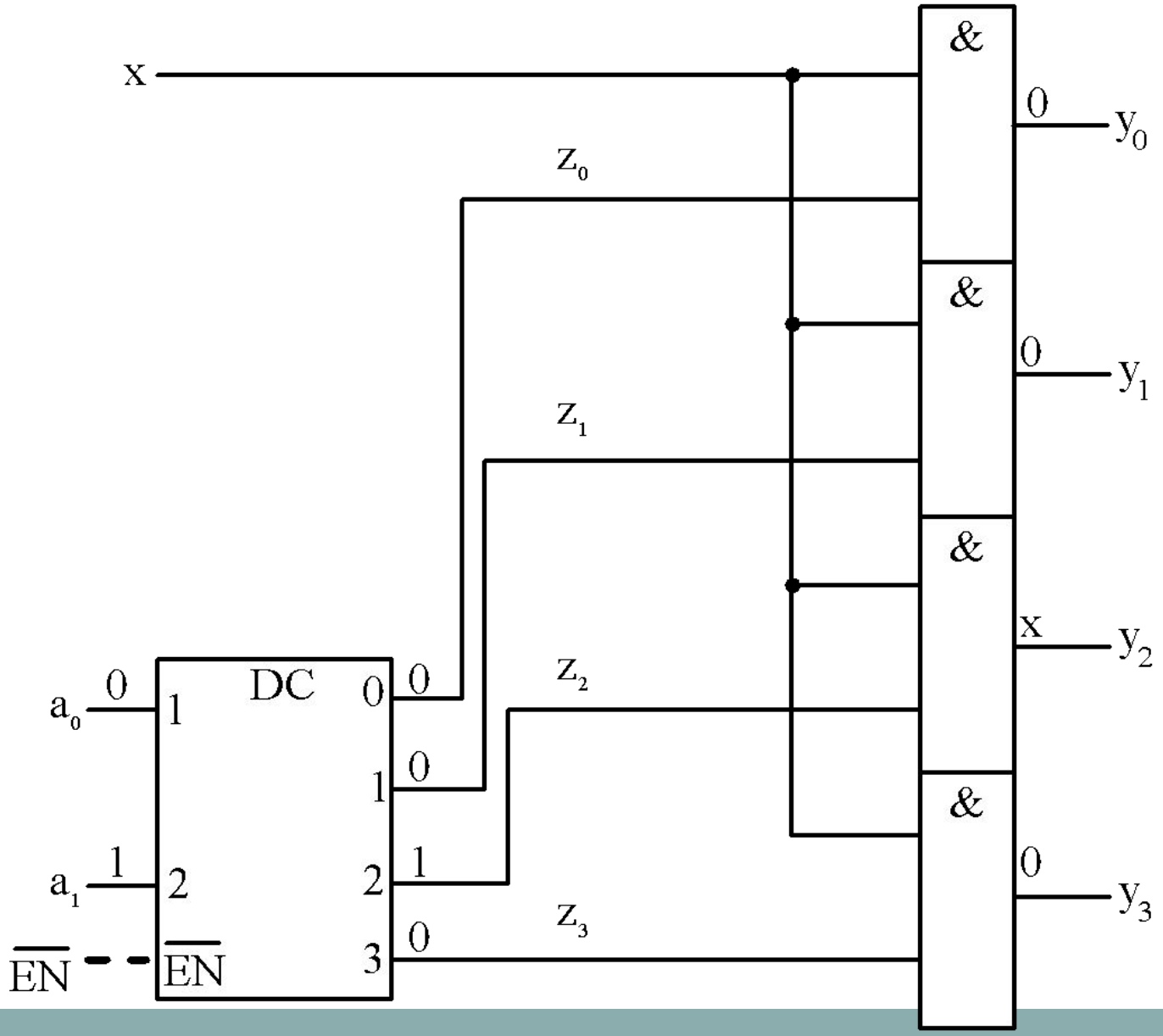
Схему демультиплексора можно структурировать следующим образом:

$$y_0 = x \cdot z_0; y_1 = x \cdot z_1; y_2 = x \cdot z_2; y_3 = x \cdot z_3,$$

где $z_0 = \overline{a_1} \cdot \overline{a_0}; z_1 = \overline{a_1} \cdot a_0; z_2 = a_1 \cdot \overline{a_0}; z_3 = a_1 \cdot a_0.$

В этом случае также выделяются два структурных компонента схемы: управляемый коммутатор и управляющий дешифратор.

Логическая схема структурированная

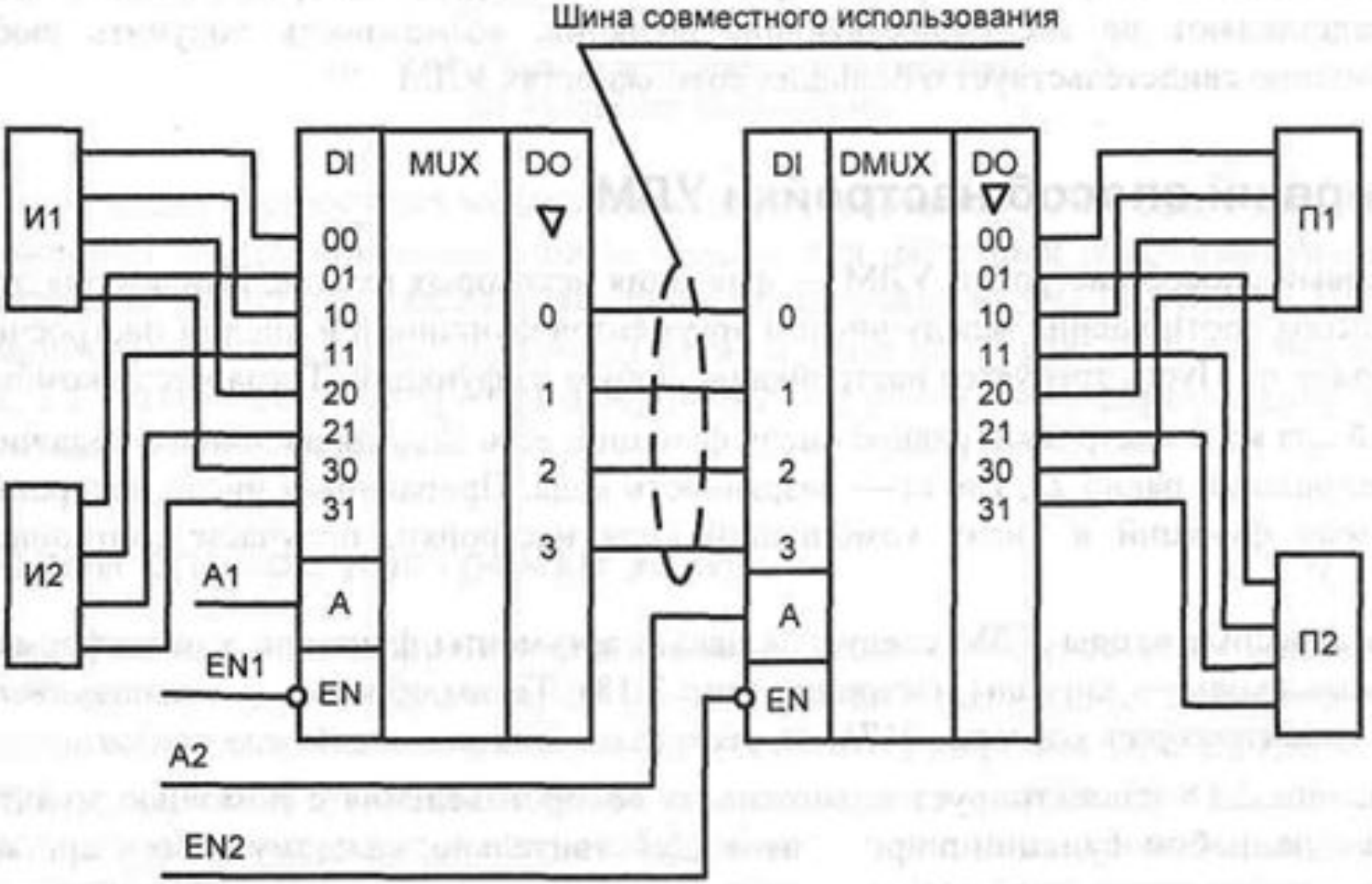


Структурная схема демультиплексора более технологична в изготовлении, более проста при поиске неисправностей. Аппаратурные затраты оцениваются величиной $E_{DMUX} = 20$ условных транзисторов, а быстродействие – $T_{DMUX} = 4t_{зд.\,ЛЭ}$. Но на практике чаще используется неструктурированная схема, поскольку она более быстродействующая и требует меньше аппаратурных затрат.

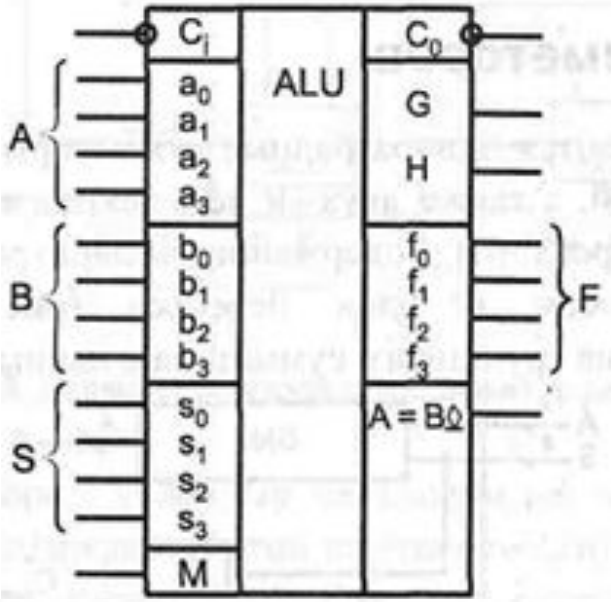
Число выходов демультиплексоров в интегральном исполнении не превышает 16. Для наращивания числа выходов демультиплексора строят демультиплексорное дерево. Входы разрешения работы будут играть роль информационных входов дешифраторов-демультиплексоров. В рассмотренном примере для адреса 11001 поток данных с информационного входа будет передаваться на выход y_{25}.

Мультиплексоры и демультиплексоры широко применяются в микропроцессорной технике, например, для стыковки внутренней шины данных с внешней шиной меньшей разрядности.

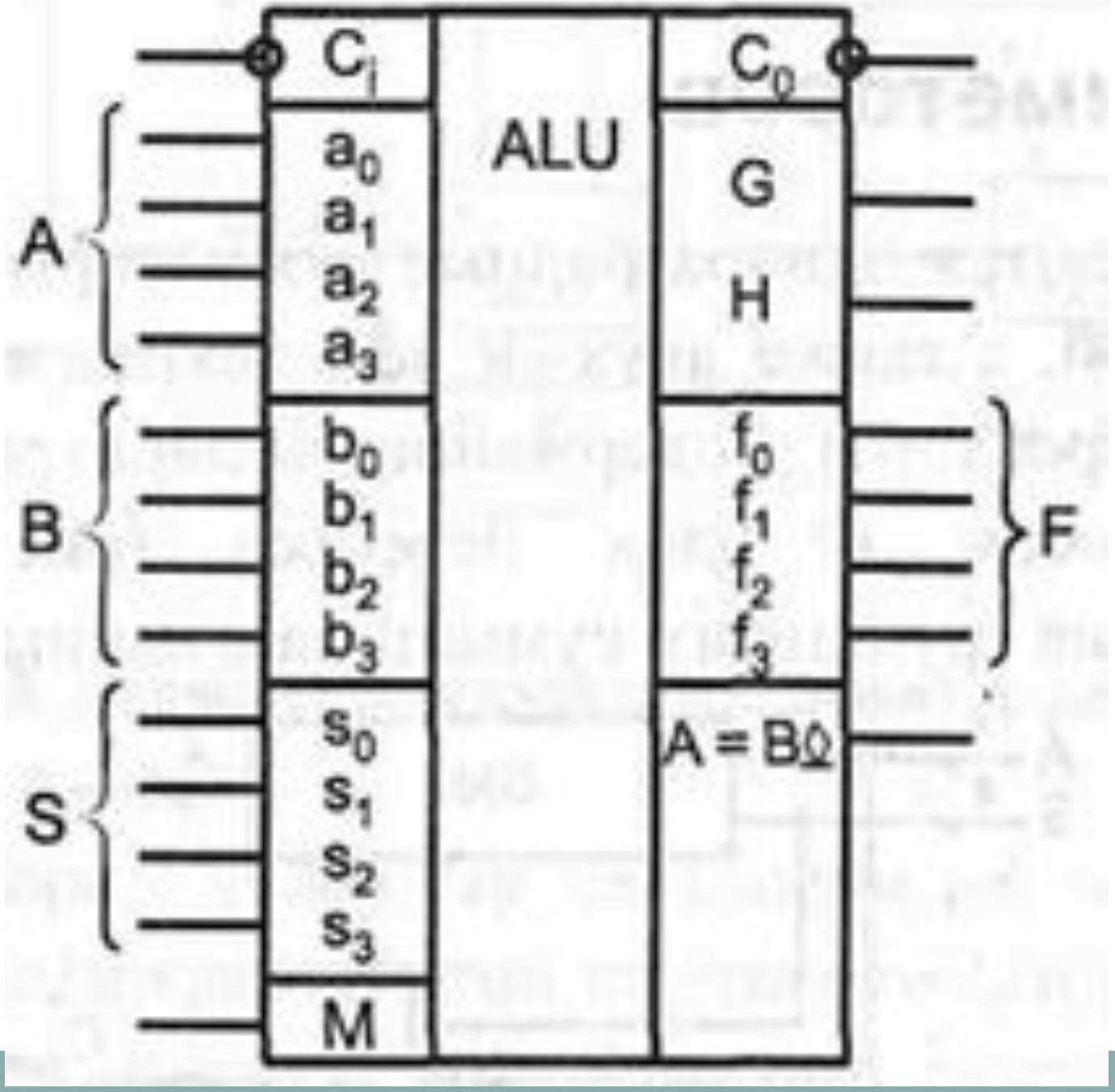
Кроме того, пара *мультиплексор-демультиплексор* представляет собой электронный коммутатор, находящий широкое использование в информационных сетях различного вида, например, в коммутационных полях цифровых коммутационных станций.


Шина совместного использования
И1
И2
DI
MUX
DO
00
01
10
11
20
21
30
31
0
1
2
3
A
EN
A1
EN1
A2
EN2
DI
DMUX
DO
0
1
2
3
A
EN
00
01
10
11
20
21
30
31
П1
П2

Арифметико-логические устройства, или сокращенно АЛУ (ALU, Arithmetic-Logic Unit), выполняют над словами ряд действий. Основой АЛУ служит сумматор, схема которого дополнена логикой, расширяющей функциональные возможности АЛУ и обеспечивающей его перестройку с одной операции на другую.



АЛУ имеет входы операндов A и B, входы выбора операций S, вход переноса $\overline{C}_i$ и вход M (Mode), сигнал которого задает тип выполняемых операций: логические (M = 1) или арифметико-логические (M = 0). Результат операции вырабатывается на выходах F, выходы G и H дают функции генерации и прозрачности, используемые для организаций параллельных переносов при наращивании размерности АЛУ. Сигнал $\overline{C}_0$ — выходной перенос, а выход A = B есть выход сравнения на равенство с открытым коллектором.


C_i
C_0
ALU
a_0
a_1
a_2
a_3
A
G
H
b_0
b_1
b_2
b_3
B
f_0
f_1
f_2
f_3
F
s_0
s_1
s_2
s_3
S
A = BΩ
M

S	Логические функции (M = 1)	Арифметико-логические функции (M = 0)
0	$\overline{A}$	$A + C_i$
1	$\overline{A \vee B}$	$A \vee B + C_i$
2	$\overline{A}B$	$A \vee \overline{B} + C_i$
3	0	$1 + C_i$
4	$\overline{A}\overline{B}$	$A + A\overline{B} + C_i$
5	$\overline{B}$	$A \vee B + A\overline{B} + C_i$
6	$A \oplus B$	$A + \overline{B} + C_i$
7	$A\overline{B}$	$A\overline{B} + 1 + C_i$
8	$\overline{A} \vee B$	$A + AB + C_i$
9	$\overline{A \oplus B}$	$A + B + C_i$
10	B	$A \vee \overline{B} + AB + C_i$
11	AB	$AB + 1 + C_i$
12	1	$A + A + C_i$
13	$A \vee \overline{B}$	$A \vee B + A + C_i$
14	$A \vee B$	$A \vee \overline{B} + A + C_i$
15	A	$A + 1 + C_i$

Двоичные сумматоры

- ***Одноразрядным двоичным сумматором*** *(ОДС)* называется КЦУ, которое предназначено для сложения двух одноразрядных двоичных чисел с учетом переноса из соседнего младшего разряда.
- ОДС имеет три входа для подачи разрядов слагаемых a_i, b_i и переноса из соседнего младшего разряда c_i. На выходах ОДС формируется сумма s_i и перенос в соседний старший разряд c_{i+1}.

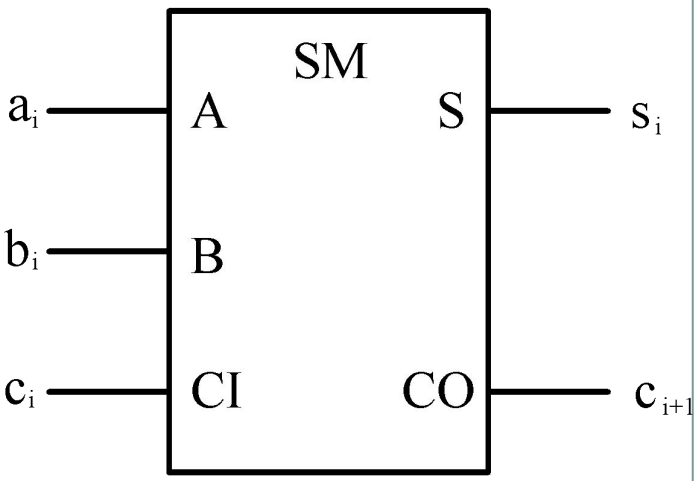
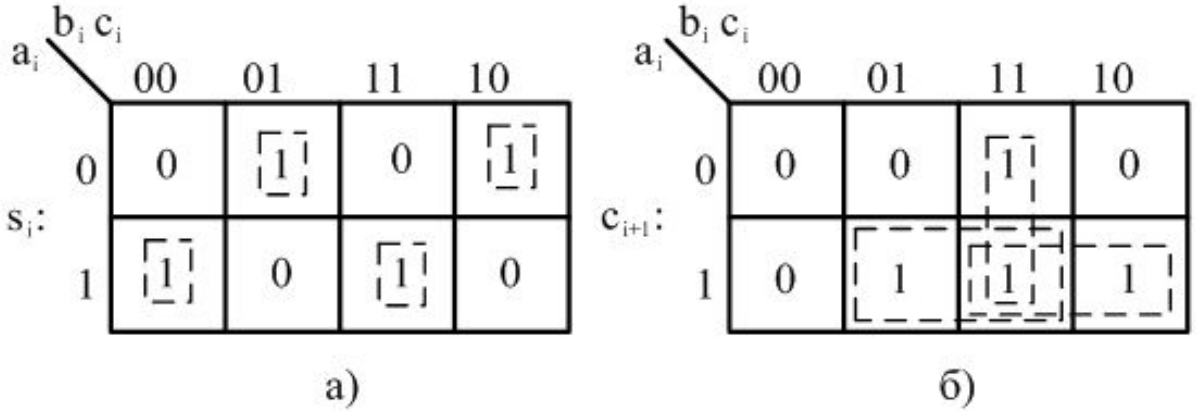


Таблица истинности ОДС

Логические аргументы			Логические функции	
a_i	b_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

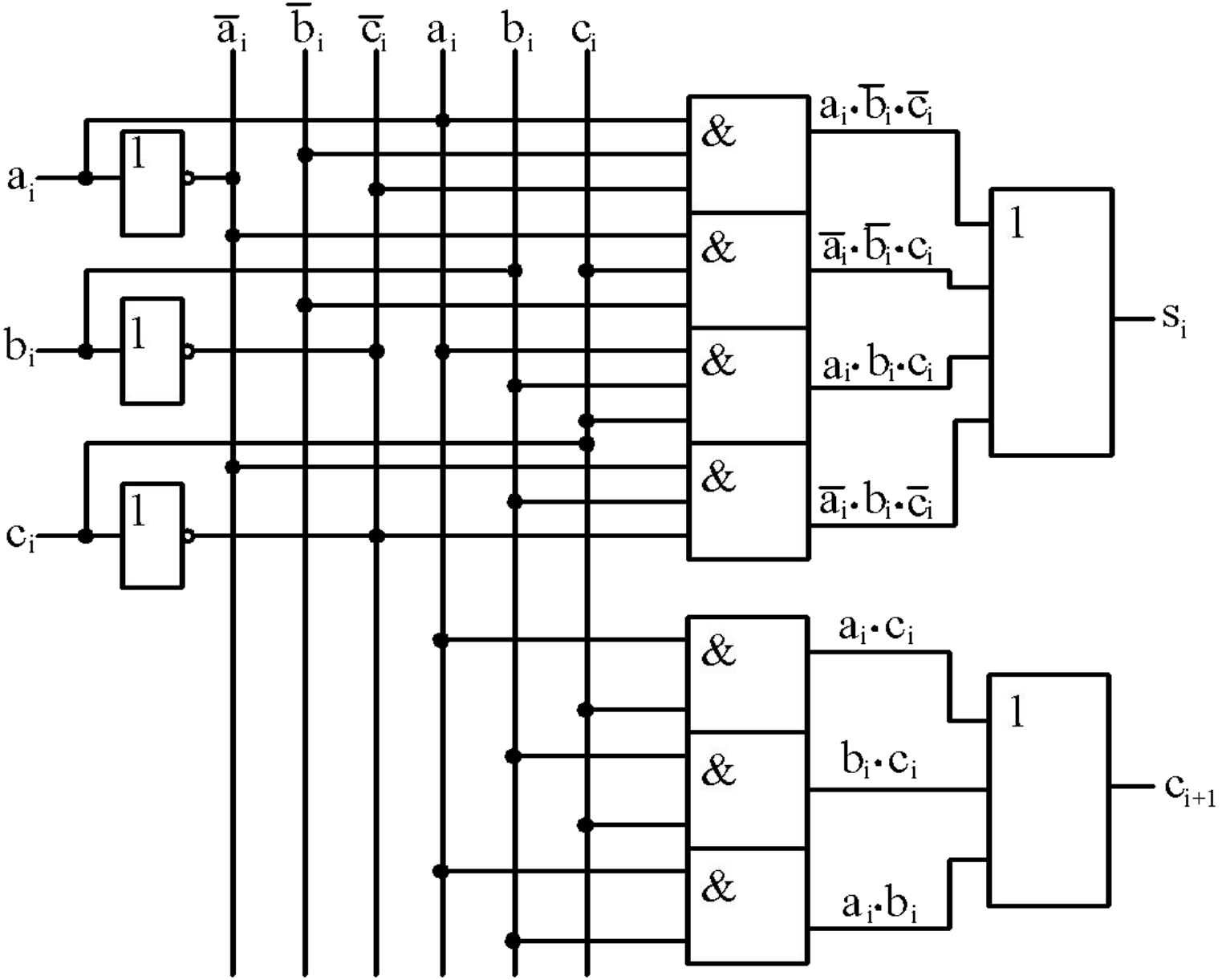


а) б)

Выполним необходимые объединения и запишем результат минимизации в МДНФ:

$$s_i = a_i \cdot \overline{b_i} \cdot \overline{c_i} \vee \overline{a_i} \cdot \overline{b_i} \cdot c_i \vee a_i \cdot b_i \cdot c_i \vee \overline{a_i} \cdot b_i \cdot \overline{c_i}.$$

$$c_{i+1} = a_i \cdot c_i \vee b_i \cdot c_i \vee a_i \cdot b_i.$$


$\overline{a}_i$ $\overline{b}_i$ $\overline{c}_i$ a_i b_i c_i
a_i
b_i
c_i
1
1
1
&
&
&
&
$a_i \cdot \overline{b}_i \cdot \overline{c}_i$
$\overline{a}_i \cdot \overline{b}_i \cdot c_i$
$a_i \cdot b_i \cdot c_i$
$\overline{a}_i \cdot b_i \cdot \overline{c}_i$
1
s_i
&
&
&
$a_i \cdot c_i$
$b_i \cdot c_i$
$a_i \cdot b_i$
1
c_{i+1}

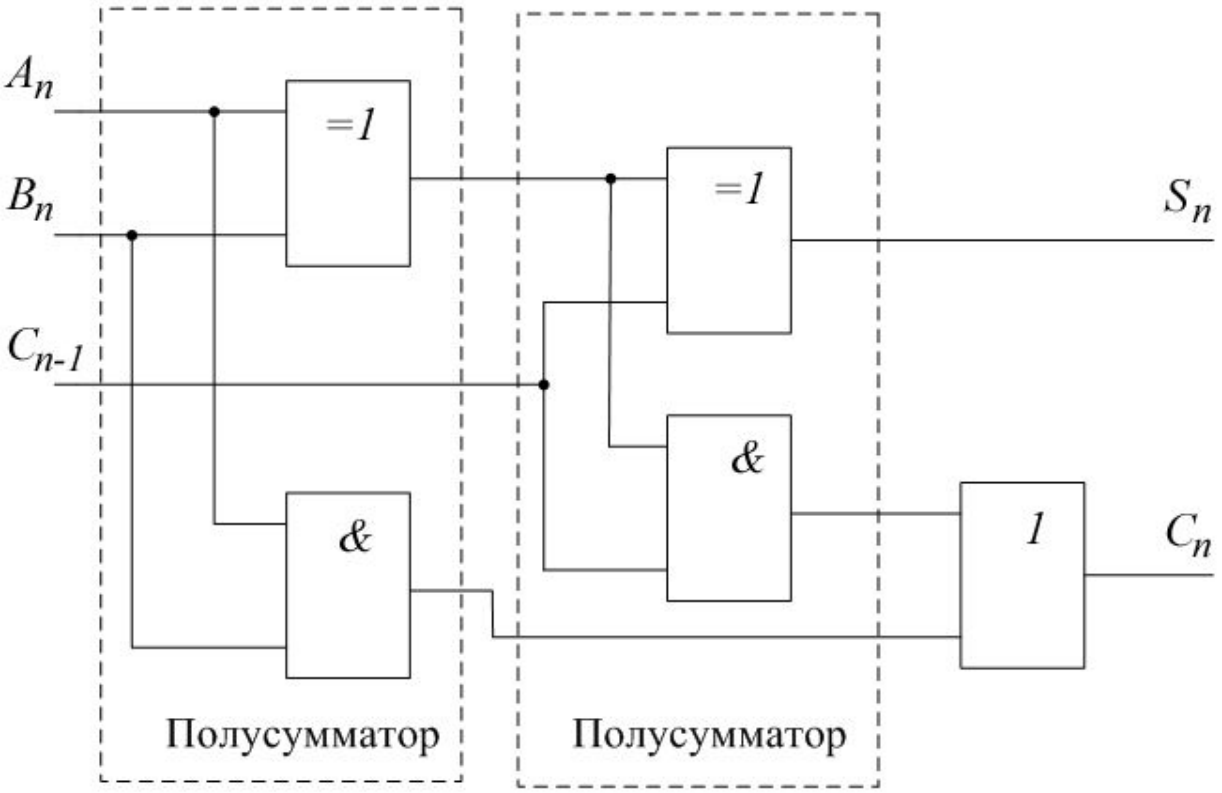
Логические выражения для S_n и C_n могут быть упрощены с помощью карт Карно.

Упрощение выражений для S_n и C_n полного сумматора

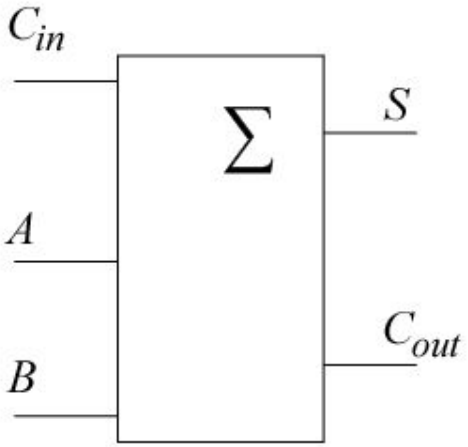
Логическое выражение для S_n может быть упрощено в базисе функций ИСКЛЮЧАЮЩЕЕ ИЛИ.

$$S_n = (\overline{A}_n \overline{B}_n C_{n-1} + \overline{A}_n B_n \overline{C_{n-1}}) + (A_n \overline{B}_n \overline{C}_{n-1} + A_n B_n C_{n-1}) =$$
$$= \overline{A}_n (B_n \oplus C_{n-1}) + A_n (\overline{B_n \oplus C_{n-1}}) = A_n \oplus B_n \oplus C_{n-1}$$

ФАЛ для C_n может быть покрыта с помощью обычного логического соседства, а также диагонального соседства.

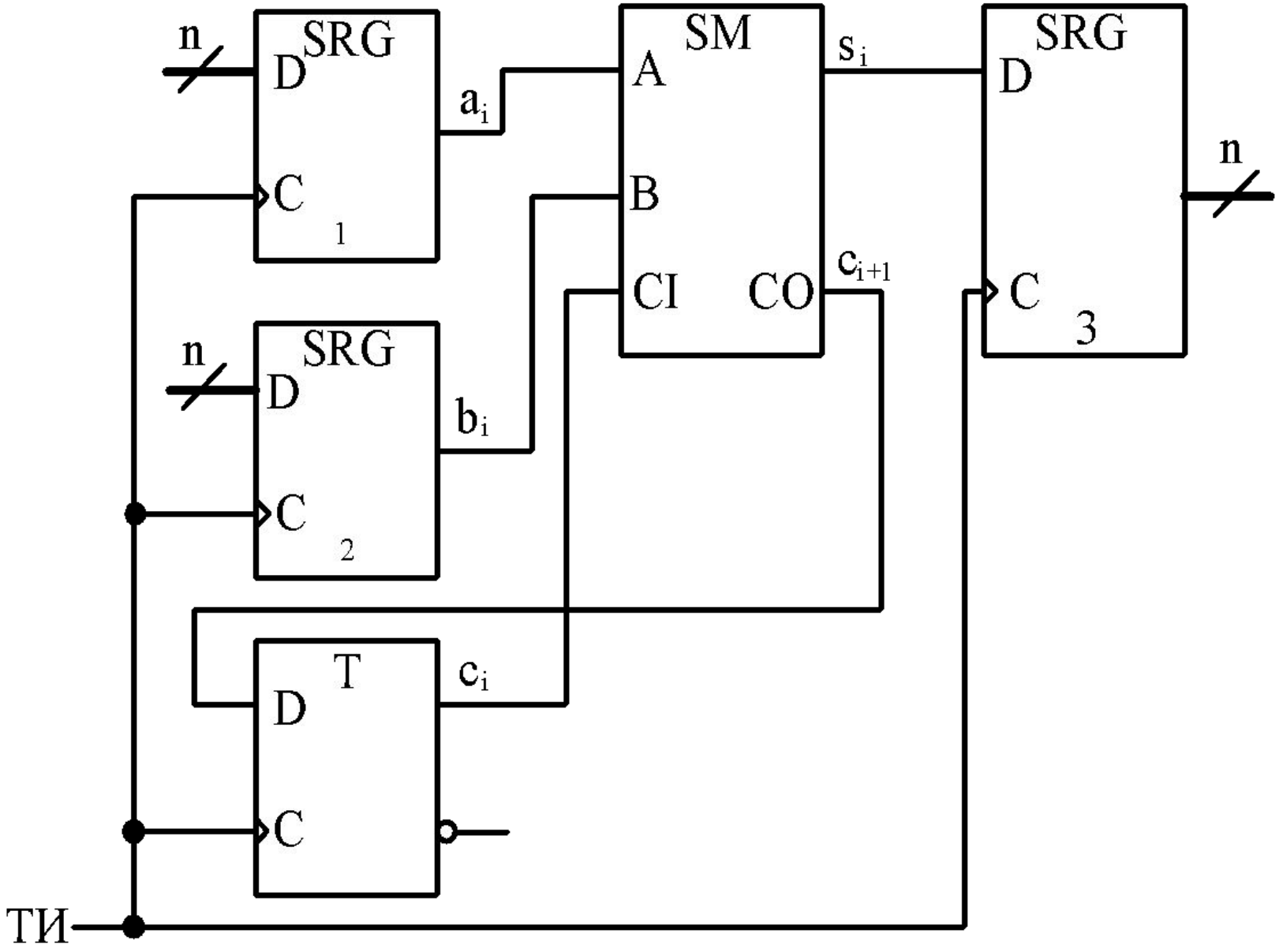

A_n
B_n
C_{n-1}
=1
&
=1
&
1
S_n
C_n
Полусумматор
Полусумматор

а)


C_{in}
A
B
$\sum$
S
C_{out}

б)

- *Многоразрядные двоичные сумматоры (МДС)* в зависимости от способа ввода кодов слагаемых делятся на два типа: последовательного действия и параллельного действия. В МДС последовательного действия коды чисел вводятся в последовательной форме, т.е. разряд за разрядом, начиная с младшего. В МДС параллельного действия каждое слагаемое подается в параллельной форме, т.е. одновременно всеми разрядами.
- Логическая схема МДС последовательного действия состоит из одноразрядного двоичного сумматора (ОДС), выход CO (от англ. Carry Output) которого соединен со входом CI (от англ. Carry Input) через D-триггер. Сдвиговые регистры 1 и 2 служат для подачи на входы сумматора разрядов слагаемых, а регистр 3 – для приема результата суммирования.
- Операция суммирования во всех разрядах слагаемых осуществляется с помощью одного и того же ОДС.

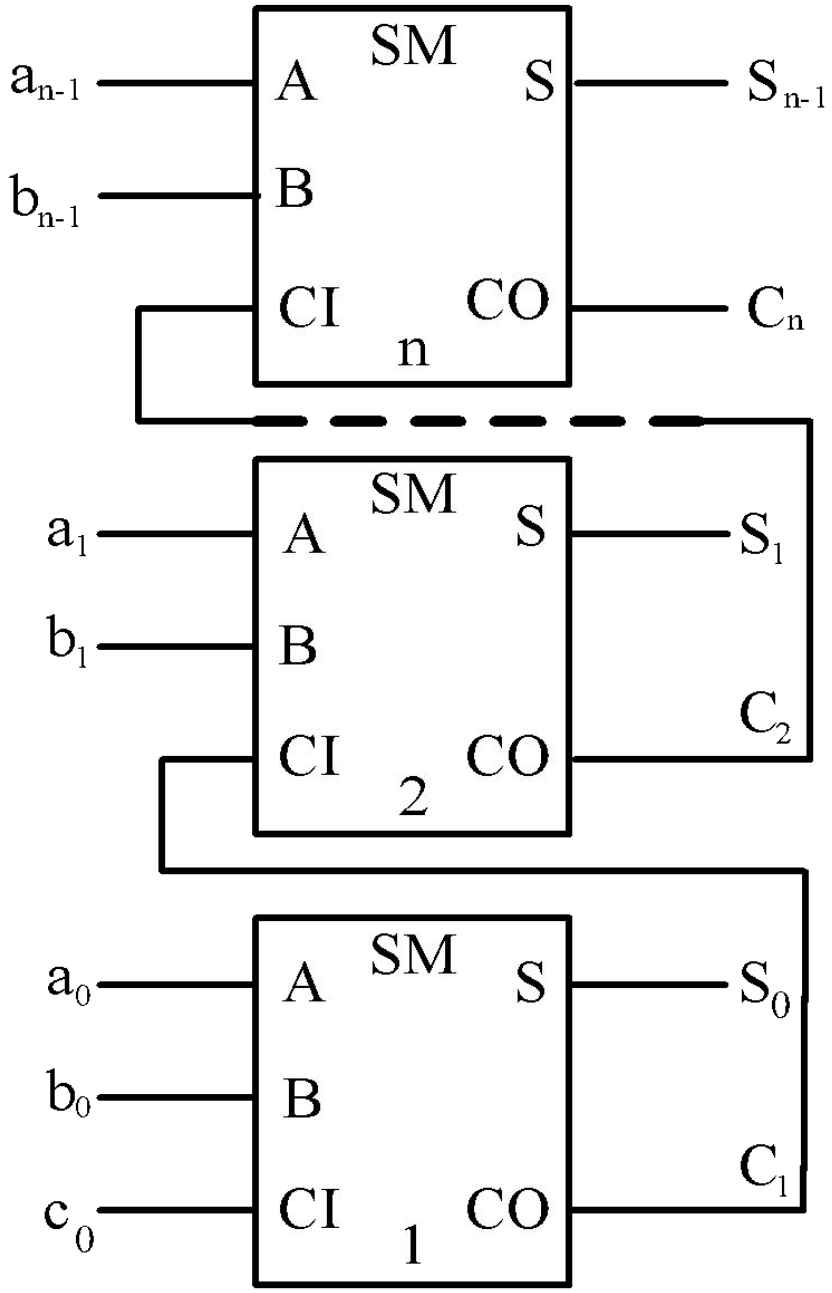

n
SRG
D
C
1
a_i
SM
A
B
CI
CO
s_i
c_{i+1}
SRG
D
C
3
n
n
SRG
D
C
2
b_i
T
D
C
c_i
ТИ

- С первым тактовым импульсом (ТИ) на входы ОДС поступают из регистров 1 и 2 цифры первого разряда слагаемых a_0 и b_0, а из D-триггера на вход CI подается нулевой сигнал. Суммируя поданные на входы цифры, ОДС формирует первый разряд суммы s_0, выдаваемый на вход регистра 3, и перенос c_1, принимаемый в D-триггер. Второй ТИ осуществляет в регистрах сдвиг на один разряд вправо, при этом на входы ОДС подаются цифры второго разряда слагаемых a_1, b_1 и c_1. Получающаяся цифра второго разряда суммы s_1 вдвигается в регистр 3, перенос c_2 принимается в D-триггер и т.д.

- Достоинством МДС последовательного действия является малый объем оборудования, требуемый для его построения, а недостатком – низкое быстродействие, так как время суммирования T_{SM} пропорционально разрядности слагаемых.

МДС параллельного действия в зависимости от способа передачи переносов от младших разрядов в старшие могут быть двух типов:

- с последовательным переносом;
- с параллельным (ускоренным) переносом.



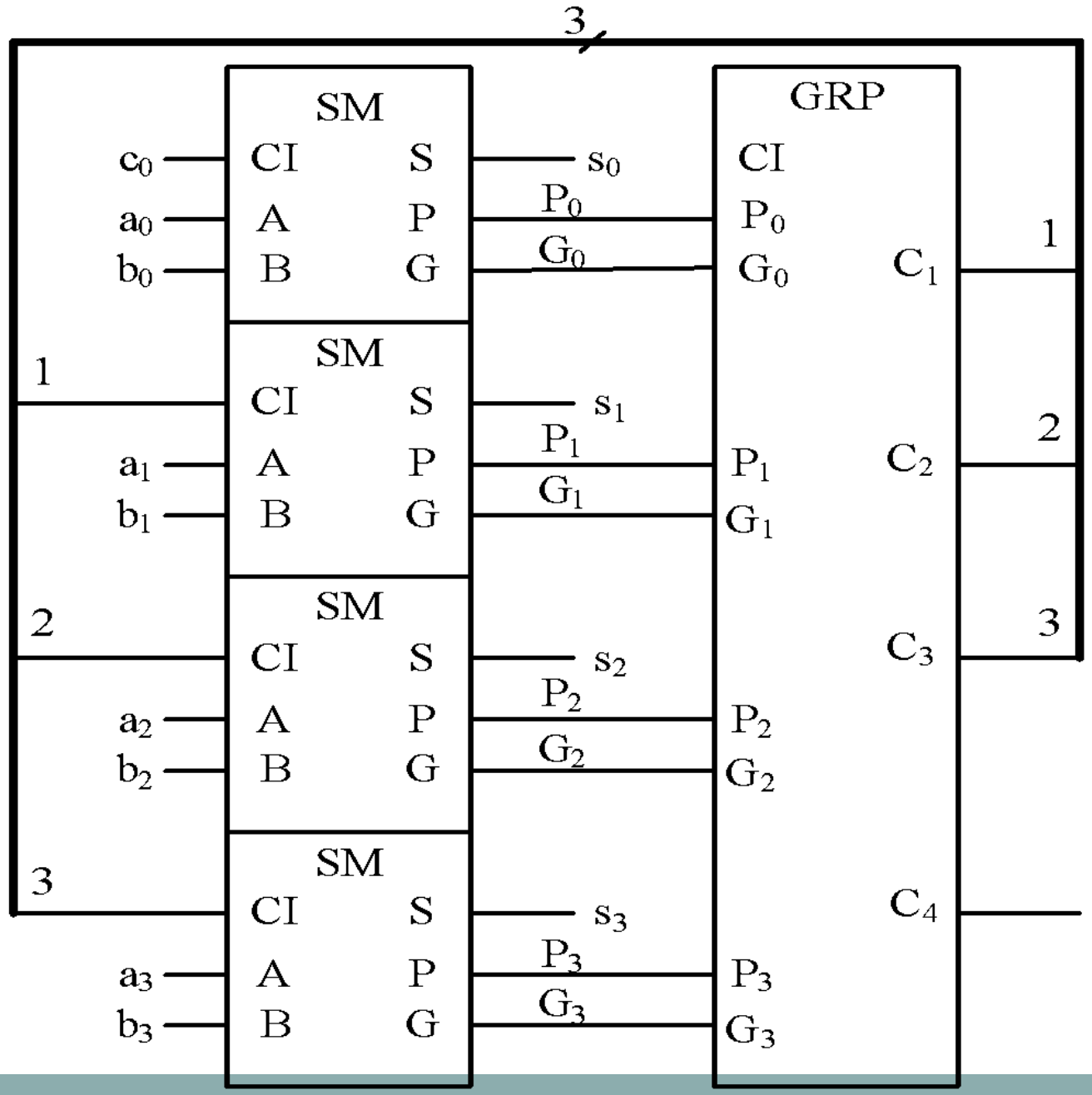
Логическая схема МДС параллельного действия с последовательным переносом состоит из отдельных разрядов, каждый из которых содержит ОДС.

- При подаче слагаемых цифры их разрядов поступают на соответствующие ОДС. Каждый из ОДС формирует на своих выходах цифру соответствующего разряда суммы и перенос в соседний старший разряд. Сигнал переноса в каждом разряде формируется после того, как будет сформирован и передан сигнал переноса из предыдущего разряда. В худшем случае, возникший в младшем разряде перенос может последовательно вызывать переносы во всех остальных разрядах. При этом время передачи переносов $T_C = n\ t_{C\ ОДС}$, где $t_{C\ ОДС}$ – задержка распространения в одном разряде. Таким образом, последовательный перенос в МДС параллельного действия не обеспечивает высокое быстродействие.

Для обеспечения высокого быстродействия в МДС параллельного действия сигналы переносов формируются одновременно для всех разрядов с помощью **блока ускоренного переноса**. При этом разрядные сумматоры не содержат цепей формирования переносов, они формируют только сумму s_i и функции G_i, P_i, для получения которых переносы не требуются. Эти вспомогательные функции генерации переноса $G_i = a_i \cdot b_i$ и распространения переноса $P_i = a_i \vee b_i$ необходимы для формирования переносов в блоке ускоренного переноса GRP. Исходя из этого, выражение (16) можно представить в следующем виде:

$$c_{i+1} = G_i \vee P_i \cdot c_i .$$

Из выражения следует, что сигнал переноса на выходе i-го разряда генерируется самим разрядом ($G_i = 1$) при $a_i = b_i = 1$ независимо от результата переноса из соседнего младшего разряда. Следовательно, можно передавать сигнал переноса для обработки старших разрядов, не дожидаясь окончания формирования переносов из младших разрядов. Однако, если только один из сигналов a_i, b_i равен единице, то перенос в следующий разряд будет иметь место только при наличии переноса из предыдущего разряда ($P_i = 1$, $c_i = 1$). Таким образом, сигналы переноса в каждом разряде формируются одновременно.


3
SM
CI
S
A
P
B
G
GRP
CI
P0
G0
C1
C2
C3
C4
c0
a0
b0
s0
P0
G0
1
1
a1
b1
s1
P1
G1
P1
G1
2
2
a2
b2
s2
P2
G2
P2
G2
3
3
a3
b3
s3
P3
G3
P3
G3

Вычитатели

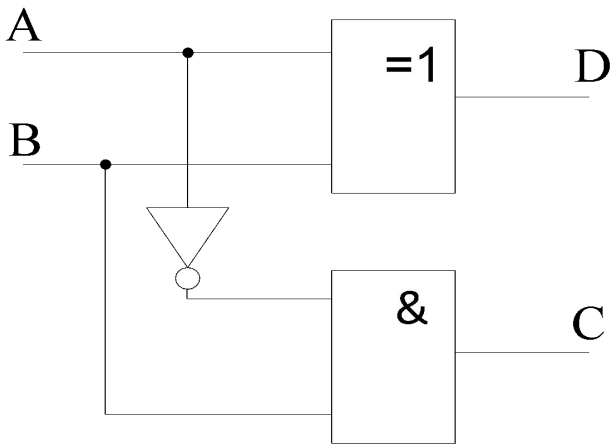
- **Полувычитатель.** Логическая схема, которая осуществляет вычитание B(вычитаемое) из A(уменьшаемое), где A и B это однобитовые числа, называется полувычитателем. Процесс вычитания может быть представлен с помощью таблицы истинности.

Таблица истинности полувычитателя

Входы		Выходы	
A	B	D	C
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Здесь A и B это входы, а выходы D разность и C - заем.
Из таблицы истинности

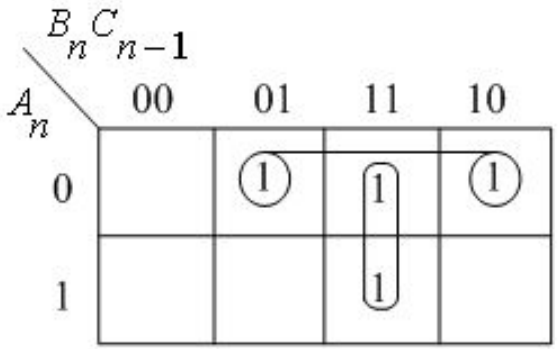
$$D = \bar{A}B + A\bar{B} = A \oplus B;$$
$$C = \bar{A}B$$



- **Полный вычитатель.** Подобно полному сумматору, мы нуждаемся в полном вычитателе, для выполнения многобитового вычитания, где заем из предыдущего разряда присутствует. Таким образом полный вычитатель имеет три входа: (уменьшаемое), (вычитаемое) и (заем от предыдущего разряда) и два выхода (разность) и (заем).

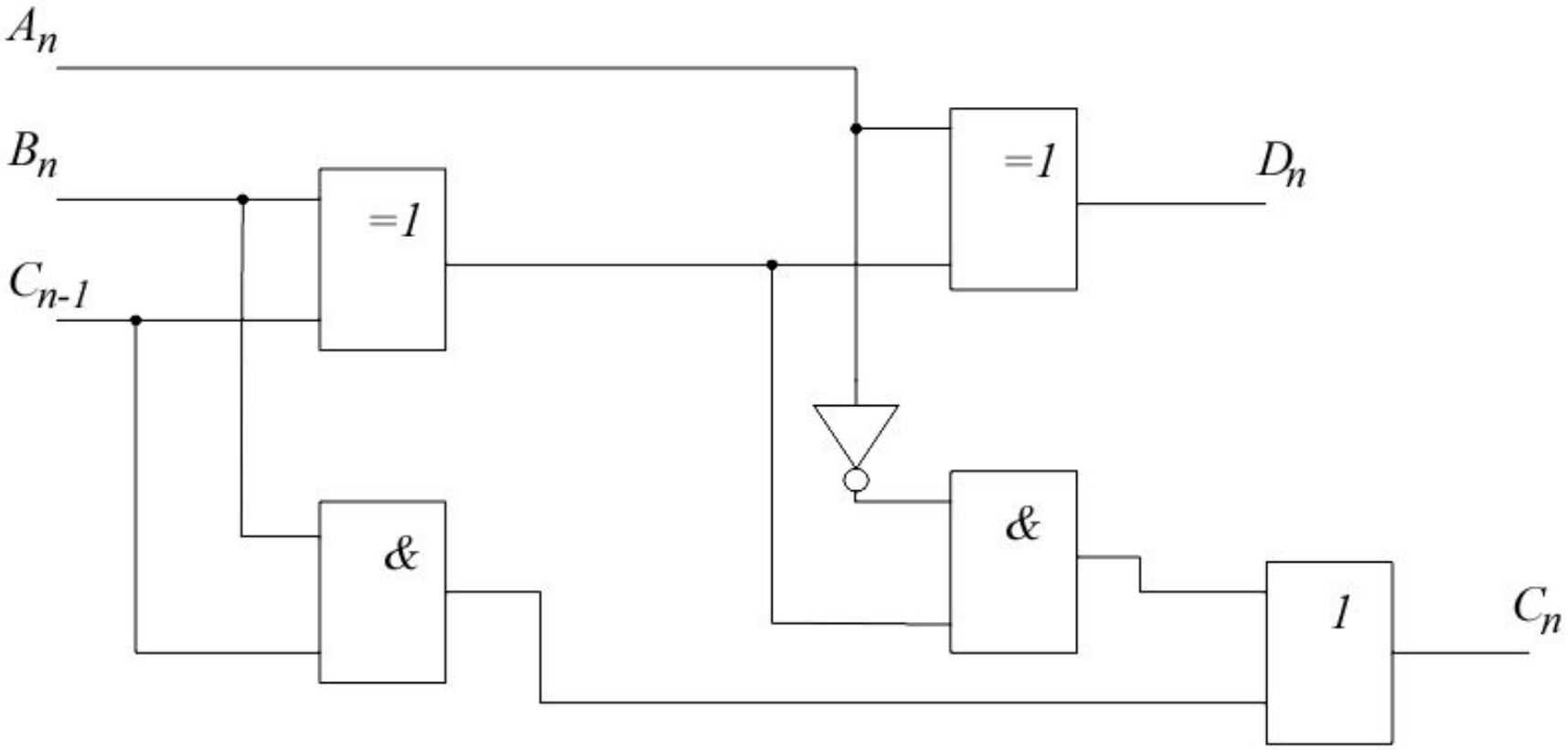
Входы			Выходы	
A_n	B_n	C_{n-1}	D_n	C_n
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$D_n = A_n \oplus B_n \oplus C_{n-1}$$



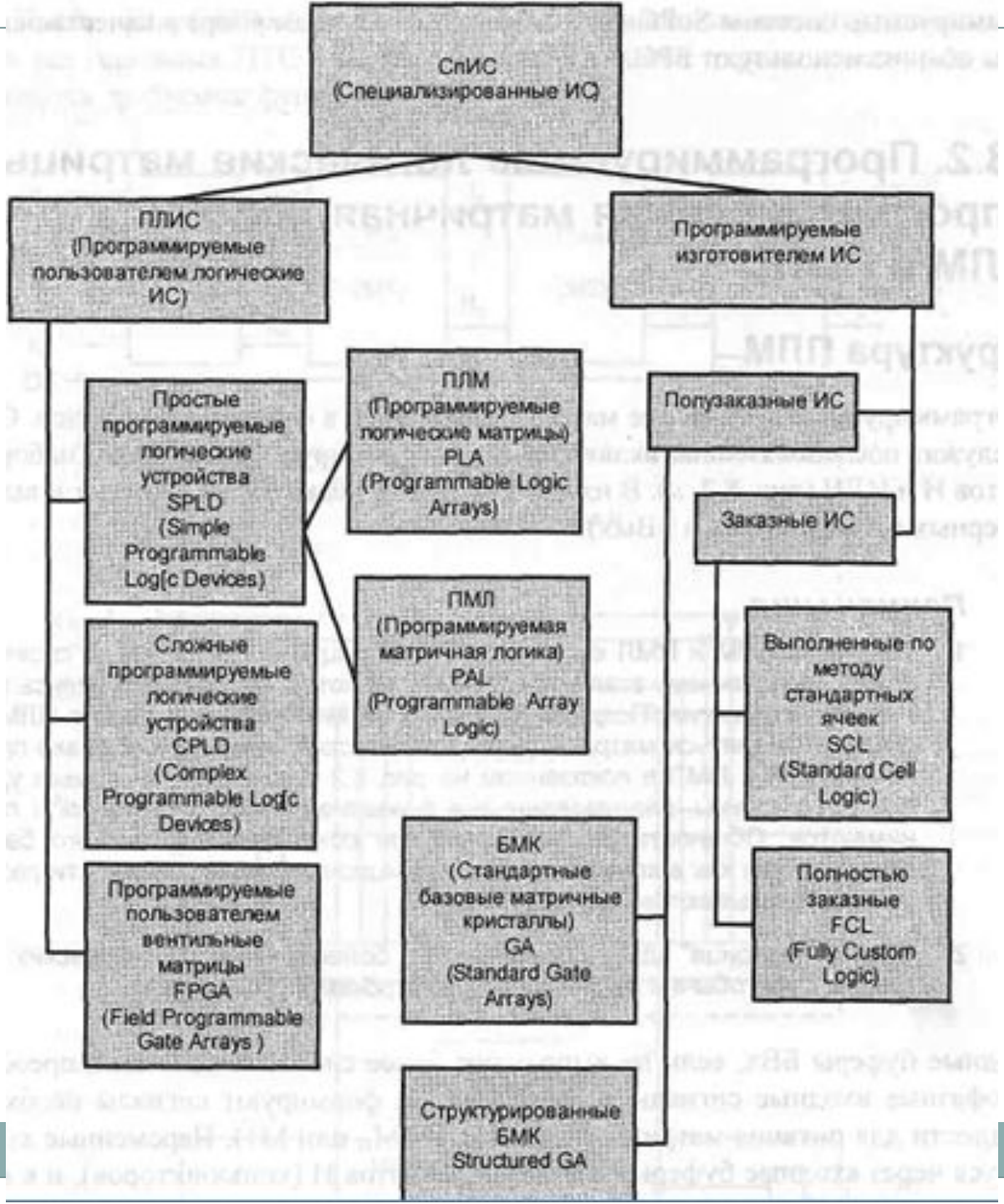
$$C_n = \overline{A}_n (B_n \oplus C_{n-1}) + B_n C_{n-1}$$

$$C_n = (\overline{A}_n \overline{B}_n C_{n-1} + \overline{A}_n B_n \overline{C}_{n-1}) + (\overline{A}_n B_n C_{n-1} + A_n B_n C_{n-1}) =$$
$$= \overline{A}_n (B_n \oplus C_{n-1}) + B_n C_{n-1}$$

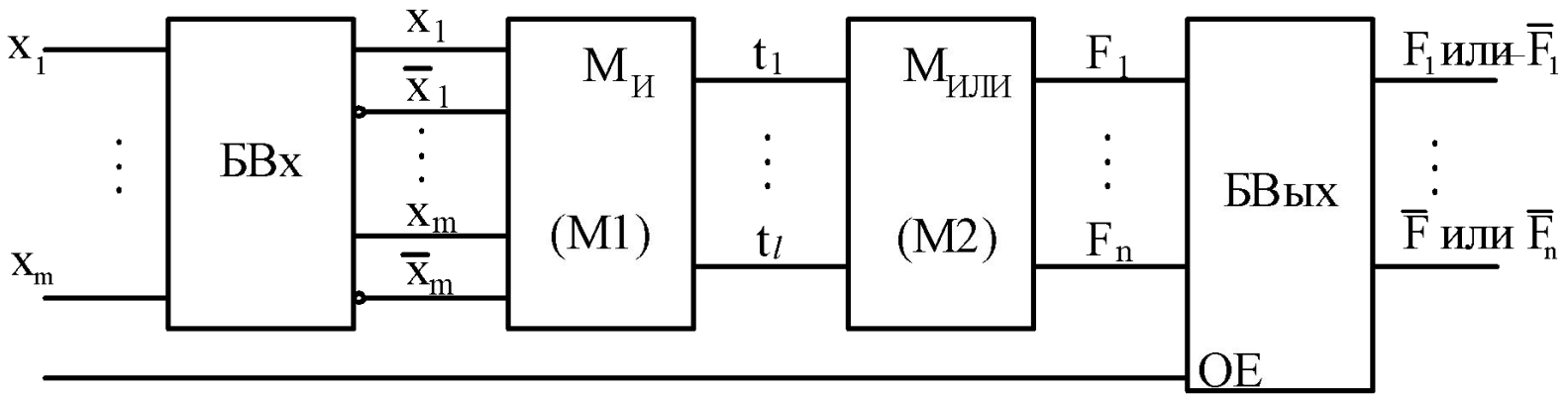


Программируемые логические структуры

- В последнее время все более широкое распространение получают различные программируемые логические структуры, которые можно разделить на программируемые логические матрицы (ПЛМ), программируемые матрицы логики (ПМЛ) и базовые матричные кристаллы (БМК). Причем ПЛМ и ПМЛ являются наиболее простыми схемами с программируемой структурой. Дальнейшее развитие этого направления привело к разработке БМК, уровень интеграции которых достиг миллионов вентилей на кристалле. Кроме того, в последние годы появился новый тип логических микросхем – перепрограммируемые логические интегральные схемы (ПЛИС). Эти микросхемы обеспечивают разработчику цифровых устройств все преимущества использования стандартного БМК, добавляя при этом гибкость и значительное сокращение времени проектирования. Особенностями ПЛИС являются: значительный объем ресурсов (до 10 млн. вентилей на кристалл); высокая производительность (до 420 МГц); высокая гибкость архитектуры с множеством системных особенностей (внутреннее ОЗУ, логика ускоренного переноса, встроенные блоки умножителей, наличие порядка ста тысяч триггеров и сдвиговых регистров); низкое энергопотребление; возможность использования развитых и недорогих средств проектирования и др.


СпИС
(Специализированные ИС)
ПЛИС
(Программируемые пользователем логические ИС)
Программируемые изготовителем ИС
Простые программируемые логические устройства
SPLD
(Simple Programmable Log[c Devices)
ПЛМ
(Программируемые логические матрицы)
PLA
(Programmable Logic Arrays)
ПМЛ
(Программируемая матричная логика)
PAL
(Programmable Array Logic)
Сложные программируемые логические устройства
CPLD
(Complex Programmable Log[c Devices)
Программируемые пользователем вентильные матрицы
FPGA
(Field Programmable Gate Arrays)
Полузаказные ИС
Заказные ИС
Выполненные по методу стандартных ячеек
SCL
(Standard Cell Logic)
Полностью заказные
FCL
(Fully Custom Logic)
БМК
(Стандартные базовые матричные кристаллы)
GA
(Standard Gate Arrays)
Структурированные БМК
Structured GA

- *Программируемая логическая матрица* характеризуется простотой получения необходимых функций. Основой ПЛМ служат последовательно включенные программируемые матрицы элементов И и ИЛИ

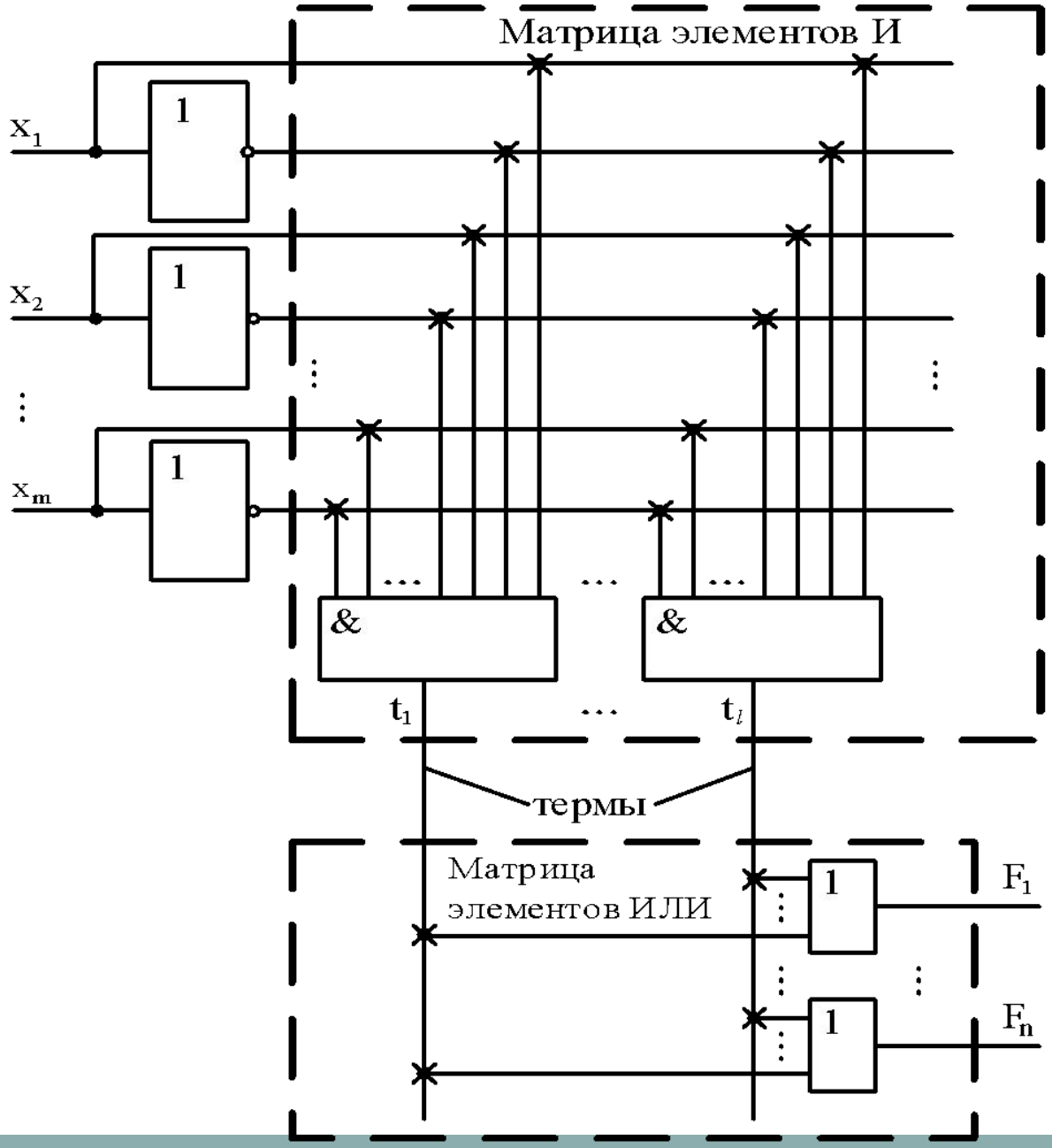


Базовая структура ПЛМ

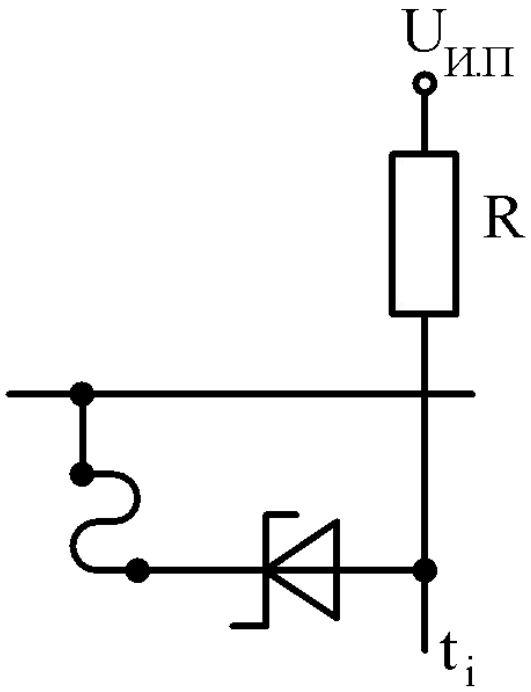
В ПЛМ также входят блоки входных и выходных буферных каскадов (БВх и БВых). Входные буферы преобразуют однофазные входные сигналы в парафазные и формируют сигналы необходимой мощности для матрицы элементов И. Выходные буферы обеспечивают необходимую нагрузочную способность выходов, разрешают или запрещают выход ПЛМ на внешние шины с помощью сигнала OE, а нередко выполняют и более сложные действия. Выпускаются ПЛМ на основе как биполярной, так и МОП-технологии.

- Основными параметрами ПЛМ являются число входов m, число термов *l* и число выходов n. *Под термом понимается конъюнкция, связывающая входные аргументы, представленные в прямой или инверсной формах.*

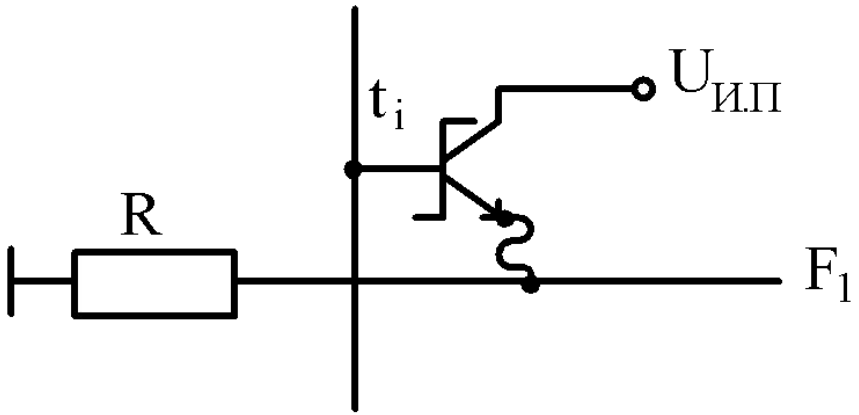
- Схема ПЛМ на вентильном уровне. Крестики в пересечениях горизонтальных и вертикальных линий обозначают программируемые точки связей (ПТС).
- **В первой ситуации** незапрограммированная ПЛМ имеет соединения во всех пересечениях, а при ее программировании часть соединений удаляется. Как видно из схемы, в этом случае в исходном состоянии все термы и функции независимо от входных переменных имеют нулевые значения, так как на входы схем *И* подаются одновременно прямые и инверсные значения аргументов, а . Элементами связей в матрице *И* служат диоды, соединяющие горизонтальные и вертикальные шины, изображающем цепи выработки терма t_i.


Матрица элементов И
x1
x2
xm
1
1
1
&
&
t1
tl
термы
Матрица
элементов ИЛИ
1
1
F1
Fn

ПЛМ схемотехники ТТЛШ. Элементы связей в матрицах И (а) и ИЛИ (б)



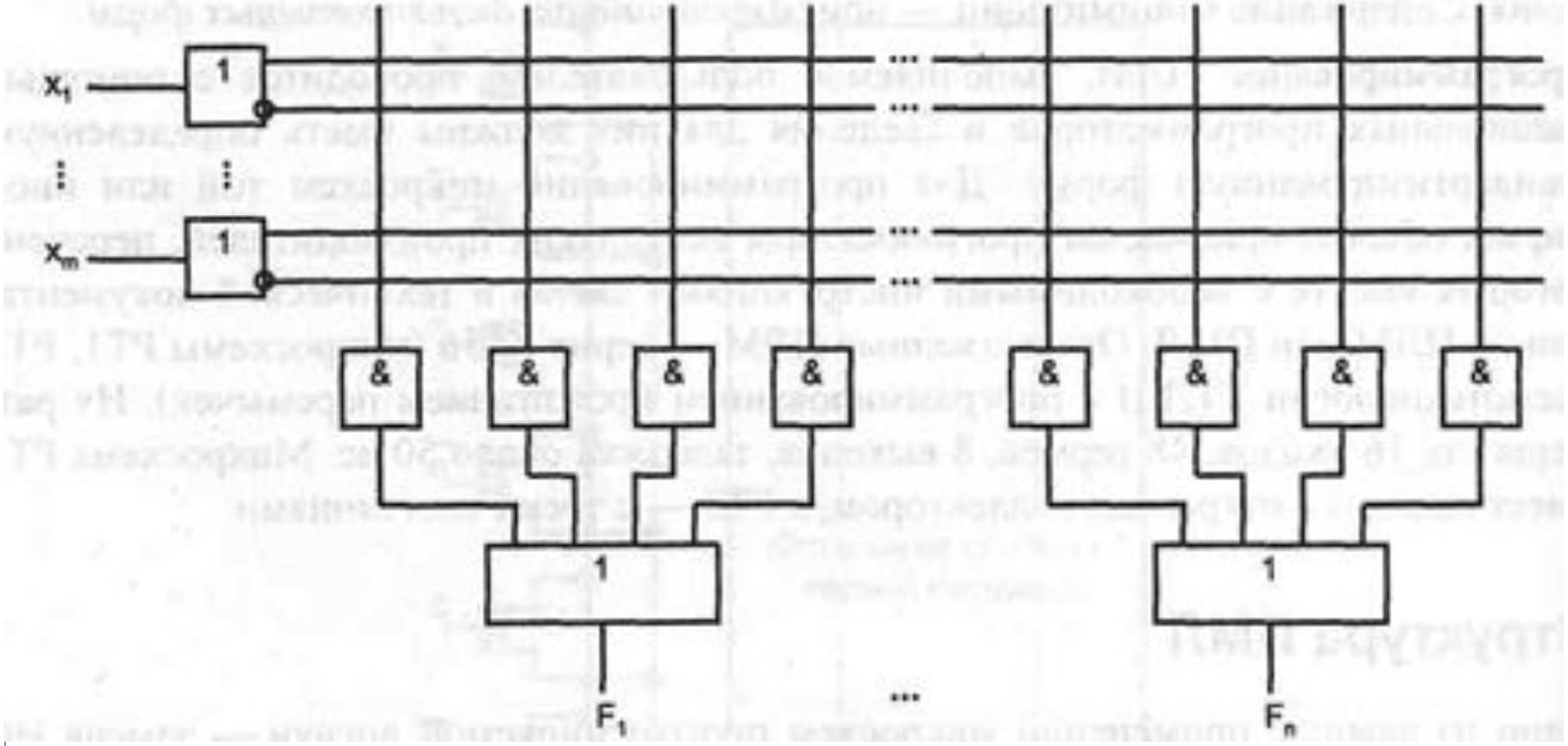
а)



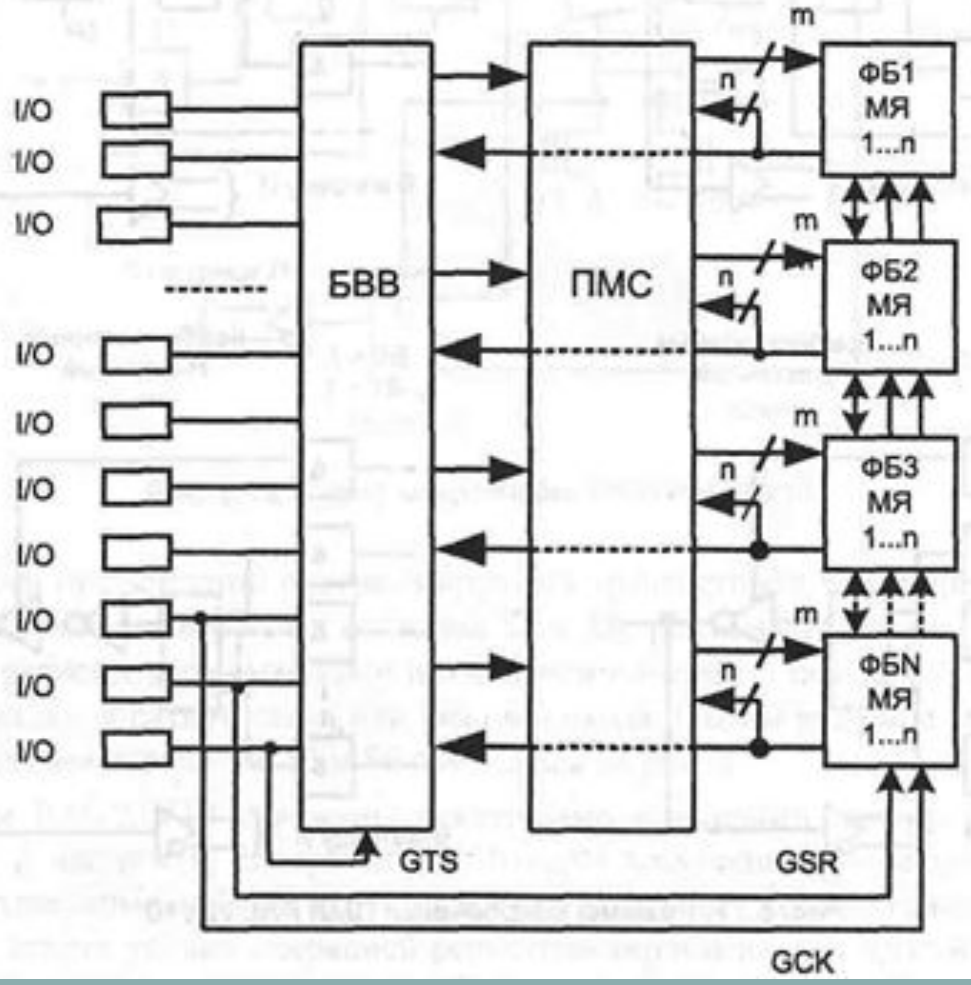
б)

- До программирования все перемычки целы и диоды размещены во всех узлах матрицы И. При программировании в схеме оставляют только необходимые элементы связи, а ненужные устраняются пережиганием перемычек. Высокий уровень на выходе конъюнктора появится при наличии высоких напряжений на всех входах (все диоды заперты). Если же хотя бы на одном входе низкий уровень напряжения, то фиксируется низкий уровень напряжения (диод открыт).
- Элементами связи в матрице ИЛИ служат транзисторы включенные по схеме эмиттерного повторителя относительно линий термов и образующие схему ИЛИ относительно горизонтальной линии выхода ПЛМ. В данном случае схема ИЛИ реализована за счет параллельного соединения эмиттерных повторителей.

  При изображении запрограммированных матриц наличие элементов связей (целые перемычки) отмечается точкой в соответствующем узле.
- Во второй ситуации все соединения отсутствуют, входные сигналы в схему не поступают. Значения термов и функций определяются внутренними цепями ПЛМ, как правило, они единичны. При программировании формируются необходимые термы, из которых и составляются требуемые функции.
- Переменные x1, x2, …, xm подаются через БВх на входы элементов И. В матрице И формируются термы, число которых равно числу конъюнкторов, т.е. числу выходов матрицы И. Далее термы подаются на входы матрицы ИЛИ, т.е. на входы дизъюнкторов, формирующих выходные функции. Число дизъюнкторов равно числу вырабатываемых функций n.
- Таким образом, ПЛМ реализует ДНФ воспроизводимых функций. ПЛМ способна реализовать систему n логических функций от m аргументов, содержащую не более l термов, т.е. представляет собой усеченное ПЗУ.
- ***В программируемых матрицах логики*** по сравнению с ПЛМ программируются только термы, т.е. конъюнкции переменных для СДНФ. Элементы ИЛИ зафиксированы и имеют, как правило, семь-восемь входов.


x1
xm
1
1
&
&
&
&
&
&
&
&
1
1
F1
Fn

CPLD (Complex Programmable Logic Device) представляет собой несколько функциональных блоков — ФБ, объединенных в единую структуру программируемой матрицей соединений — ПМС (*PIA*, Programmable Interconnect Array)
Каждый ФБ подобен ПМЛ (PAL) и содержит несколько *макроячеек* — МЯ. С внешней средой CPLD связывают блоки ввода/вывода — БВВ. Кроме основных блоков CPLD может содержать контроллеры интерфейсов JTAG и ISP, используемые для конфигурирования и тестирования создаваемых структур, и другие блоки.



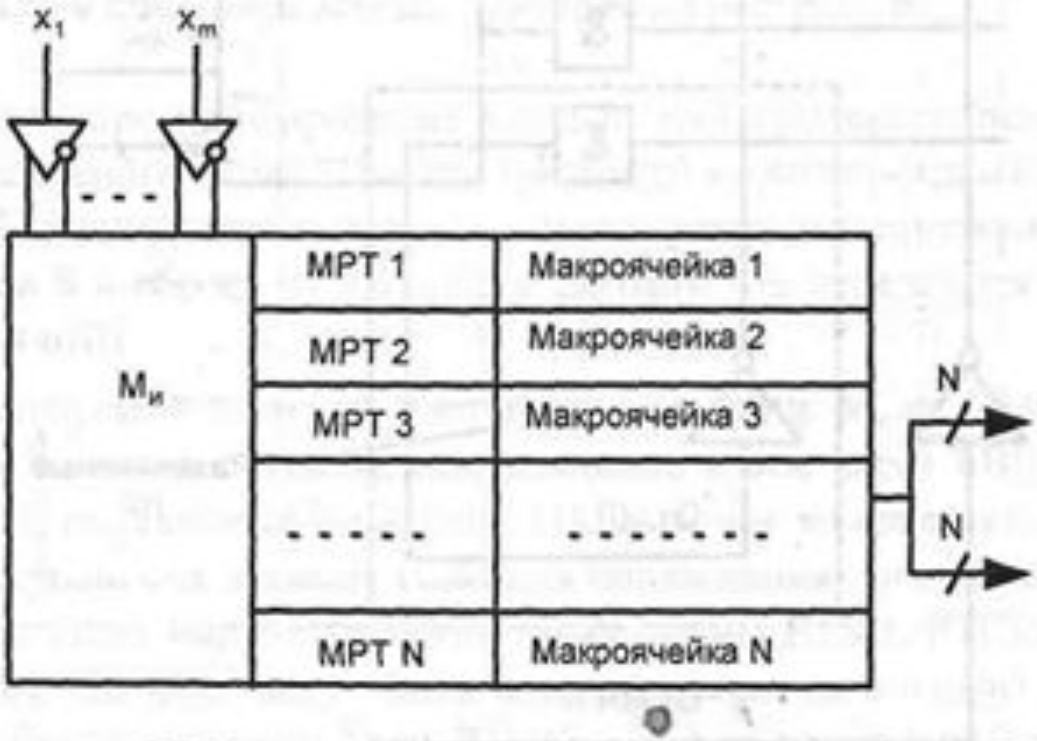
Число ФБ в составе CPLD изменяется в зависимости от ее сложности. Каждый ФБ получает *m* сигналов от ПМС, а *n* его выходов подключены как к ПМС, так и к блокам ввода/вывода БВВ (Input/Output Block, IOB), связанным с внешними двунаправленными выводами. Три вывода специализированы и предназначены для глобальных сигналов тактирования GCK (Global Clock), сброса/установки GSR (Global Set/Reset), управления третьим состоянием GTS (Global Tri-State). Возможно и иное использование специализированных выводов, если они не применяются по назначению.

Число контактов ввода/вывода может быть меньше числа выводов всех ФБ. В этом случае часть макроячеек может быть использована только для выработки внутренних сигналов (сигналов обратных связей), потребность в которых типична для многих видов устройств.

Функциональные блоки CPLD

Функциональные блоки CPLD подобны ПМЛ (PAL) и содержат:

- многовходовую программируемую матрицу элементов И ($M_И$), вырабатывающую конъюнктивные термы из поступающих на ее входы переменных $x_1 \ldots x_m$;
- матрицы распределения термов МРТ;
- группу из N макроячеек, между которыми с помощью МРТ распределяются выработанные матрицей $M_И$ термы.



Обобщенная структура функционального блока CPLD

Функциональные блоки реализуют двухуровневую логику типа ДНФ с вариантами формируемых выходных сигналов (прямой или инверсный, комбинационный или регистровый).

В классических ПМЛ термы жестко распределяются между макроячейками, формирующими выходные функции. Матрицы распределения термов МРТ позволяют варьировать число термов в вырабатываемой макроячейкой функции F_i. При этом термы заимствуются у каналов выработки других функций или отдаются им. Если термы используются не только дизъюнкторами формирования выходных функций, но и другими элементами (например, для управления триггерами, входящими в состав макроячеек), то и для них МРТ играет роль "раздатчика термов".

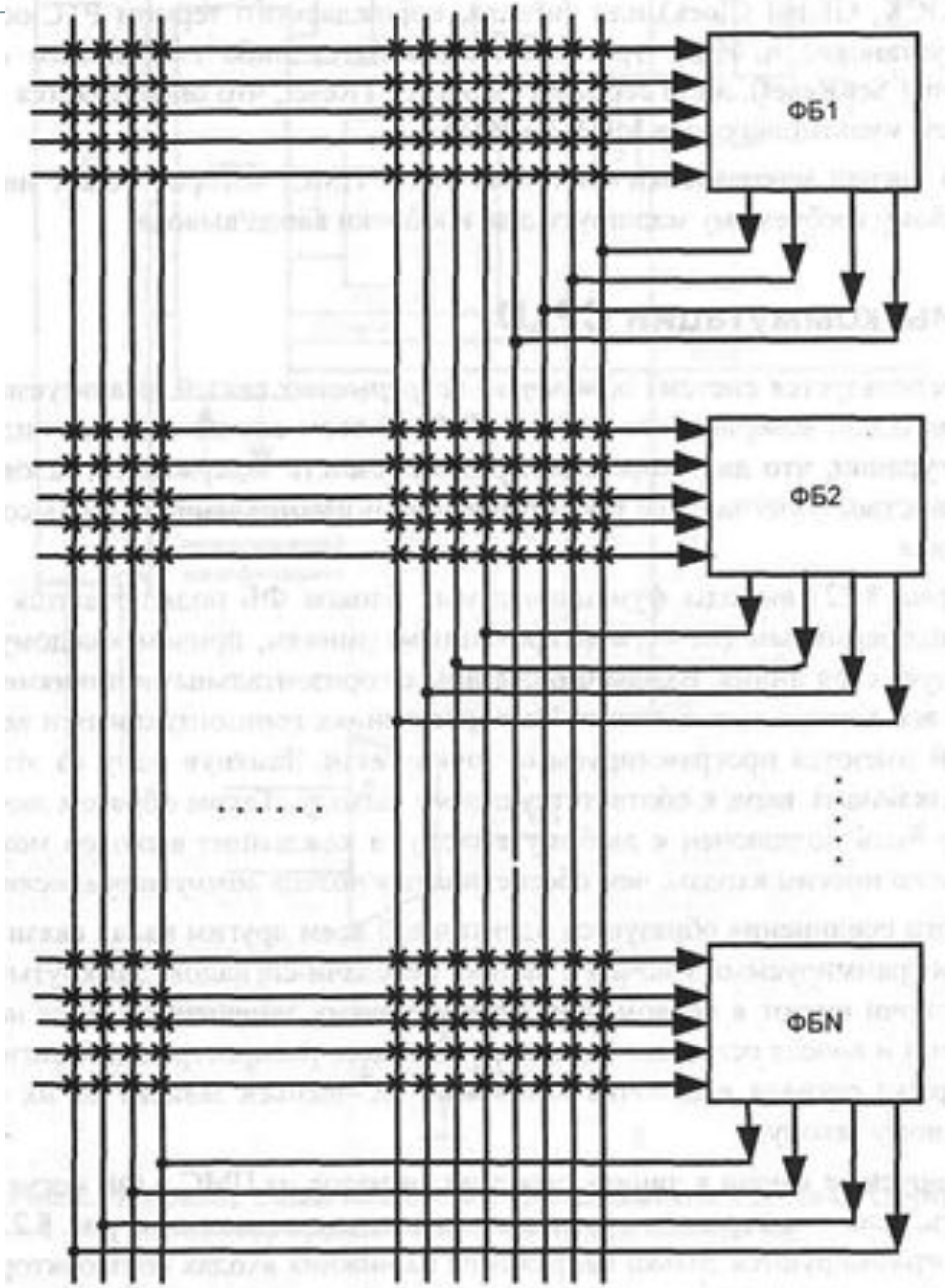


Схема программируемой матрицы соединений

FPGA (Field Programmable Gate Arrays) — программируемые пользователем вентильные матрицы — наиболее обширный класс программируемых схем, обладающих максимальными функциональными возможностями. На их основе созданы системы на программируемом кристалле СнПК (в английском оригинале SoPC, Systems on Programmable Chip).

С учетом архитектурных особенностей и областей применения выделим следующие подклассы FPGA и систем на их основе:

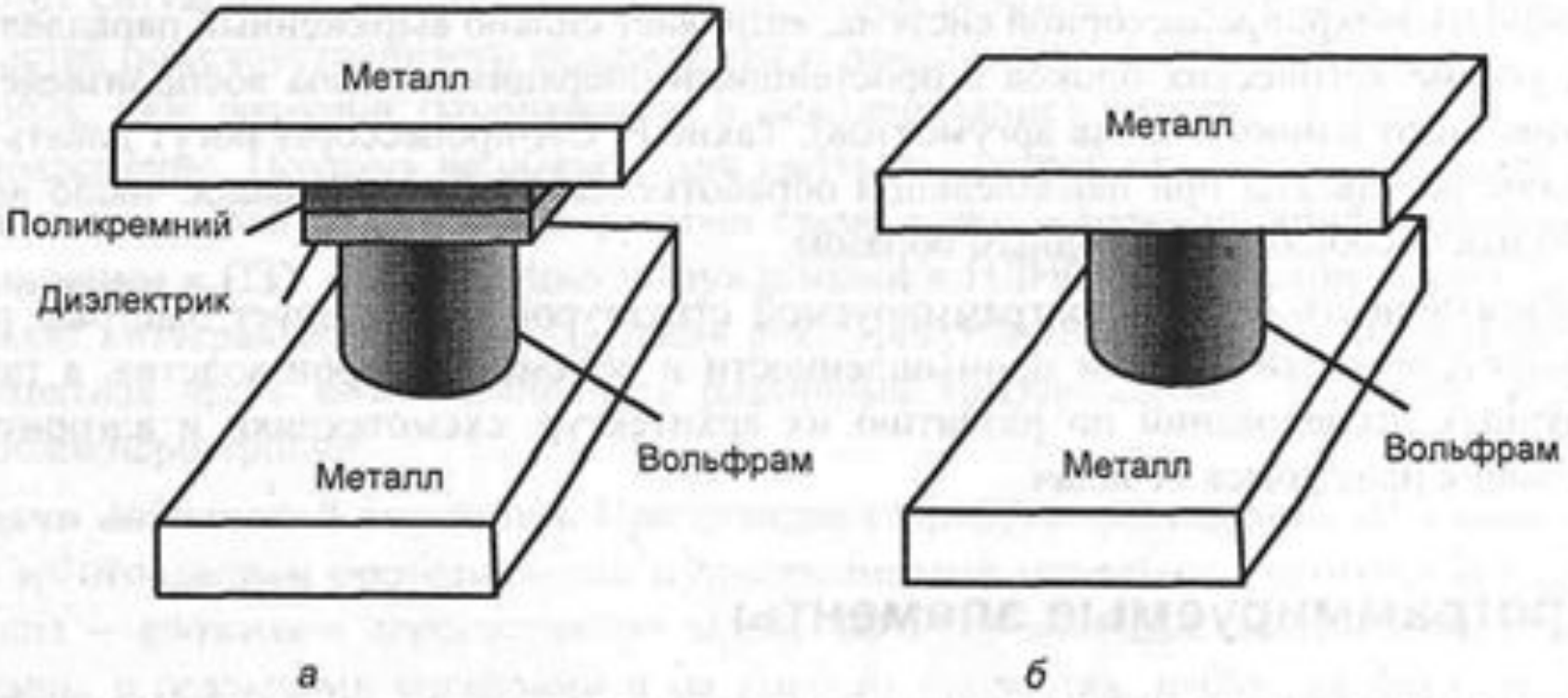
- FPGA невысокой и средней сложности;
- FPGA высокой сложности и системы на кристалле;
- микроконтроллерные программируемые системы.

В разработке FPGA участвуют десятки фирм, ведущие среди них — Xilinx (пионер в создании FPGA), Altera, Actel, Atmel, Lattice Semiconductor, Cypress Semiconductor (все USA) и др. Этими фирмами выпускаются семейства FPGA, которые по мере освоения новых технологических процессов (с интервалом в год-два) подвергаются модификациям и образуют *серии*, состоящие из родственных *семейств*.

Программируемые элементы

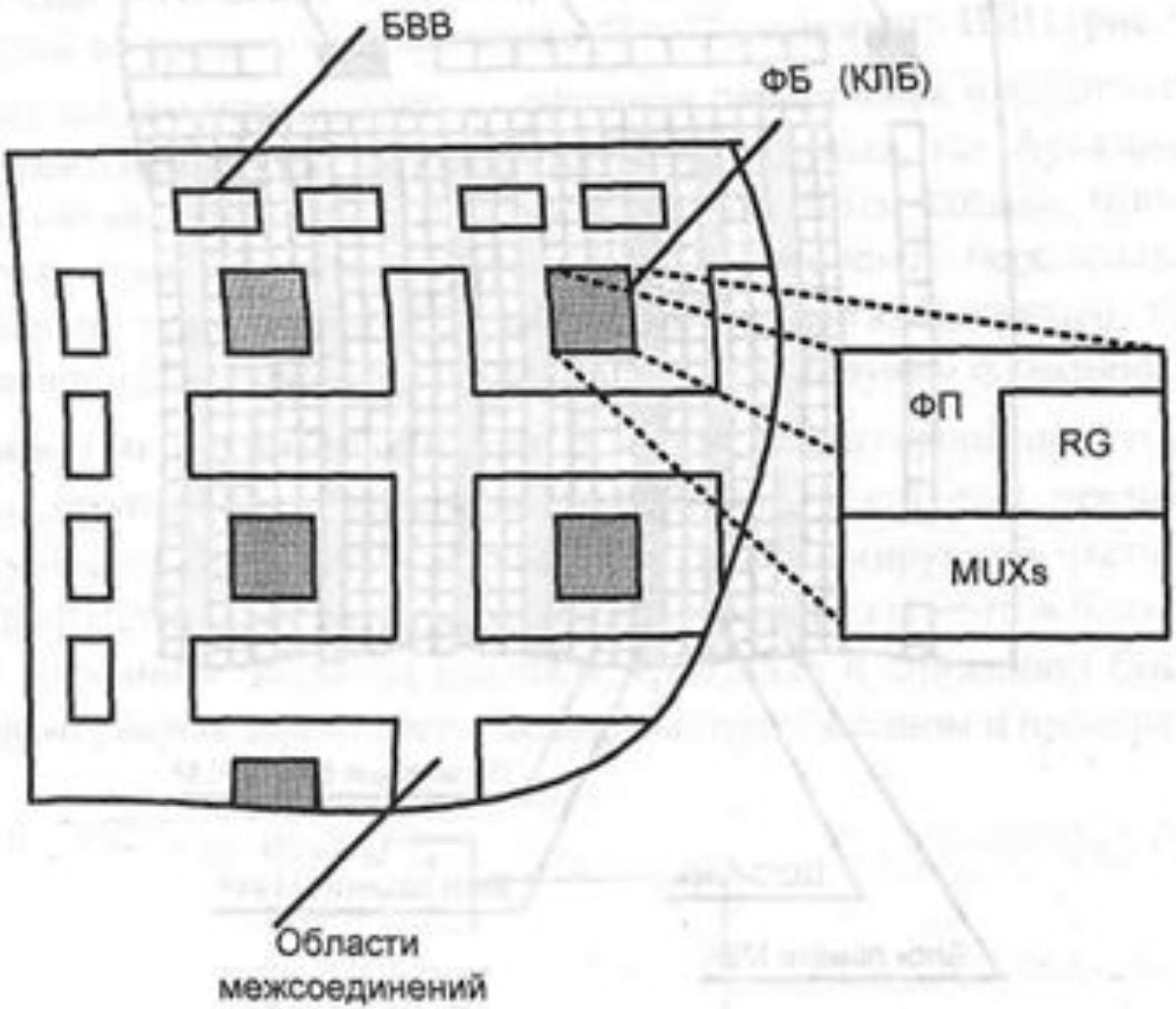
Программируемость пользователем, т. е. реализуемость индивидуального проекта на основе стандартной микросхемы, обеспечивается наличием в схеме множества двухполюсников, проводимость которых может быть задана либо очень малой (это соответствует разомкнутому ключу), либо достаточно большой (это соответствует замкнутому ключу). *Состояния ключей задают конфигурацию схеме, формируемой на кристалле*. Число программируемых ключей (программируемых точек связи) в схеме зависит от ее сложности и может доходить до сотен миллионов и более. Для FPGA характерны следующие виды программируемых ключей:

- перемычки типа antifuse (общепринятый русский термин отсутствует);
- ключевые транзисторы, управляемые триггерами;
- флэш-ключи.



Программируемые перемычки antifuse (*а*)
и обычные перемычки "металл-металл" (*б*)

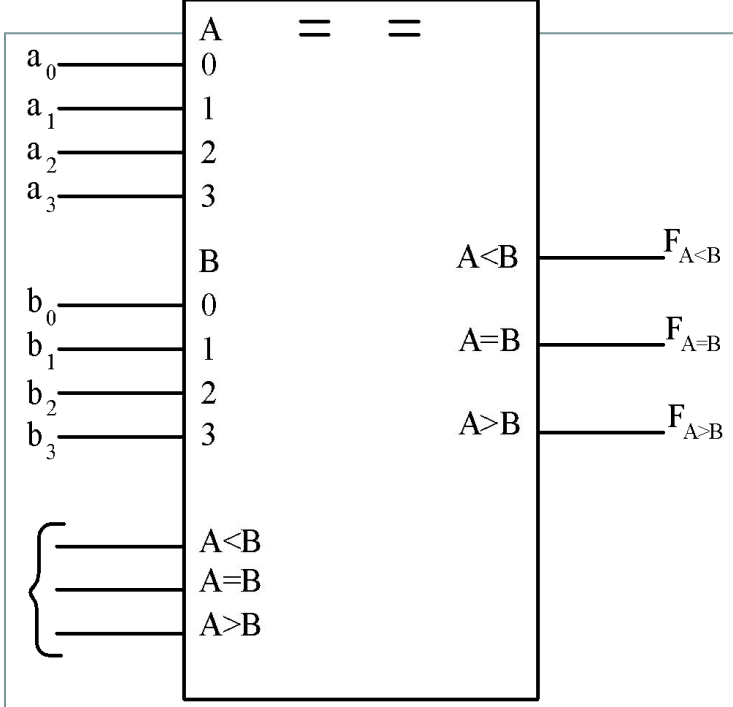
Блоки ввода/вывода связывают FPGA с внешней средой, и их, как можно программировать на выполнение ряда стандартов передачи данных.



Фрагмент FPGA базовой архитектуры и ее функциональный блок

Компараторы

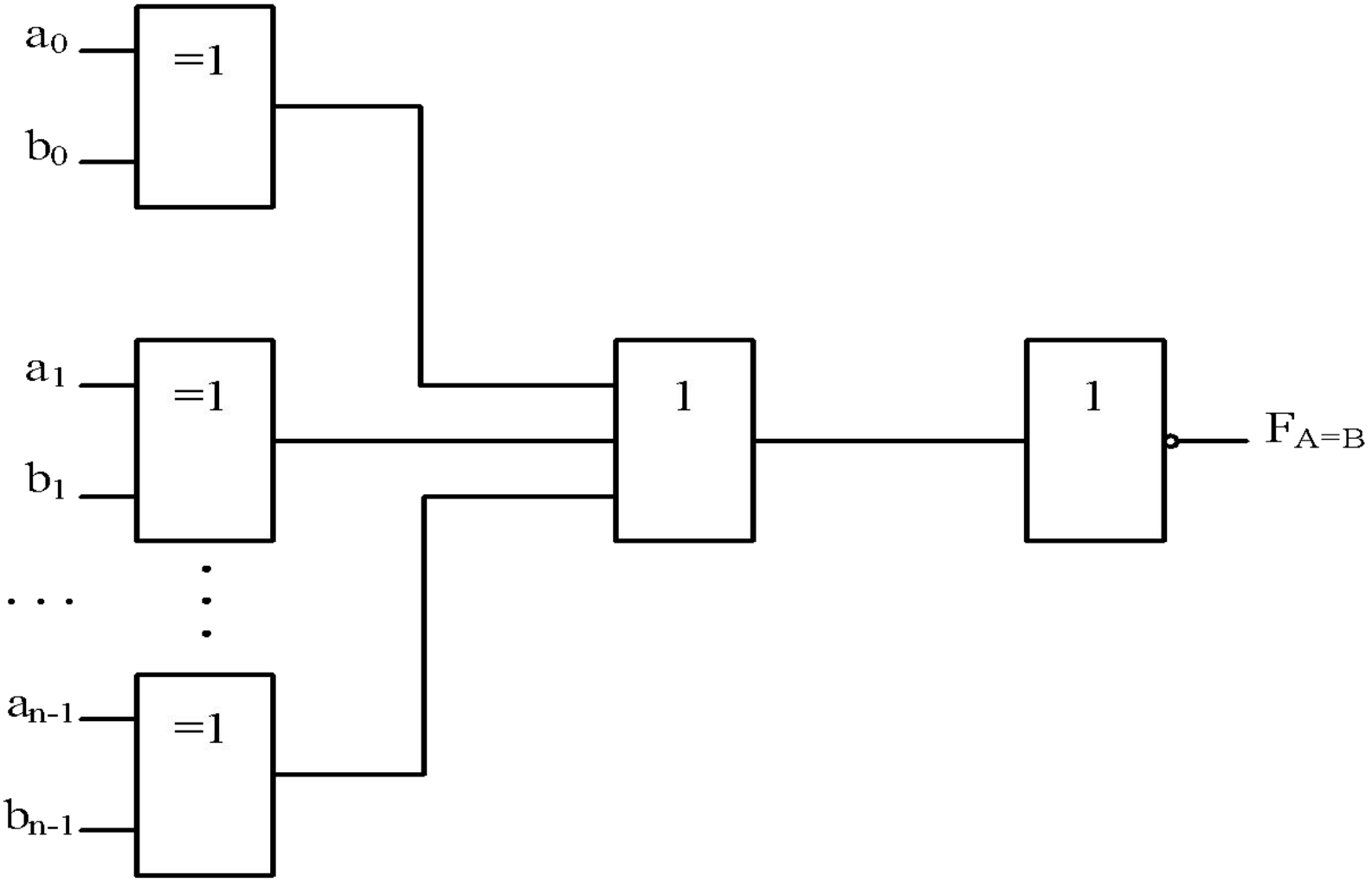
- *Компаратором (устройством сравнения)* называется КЦУ, которое предназначено для сравнения двух двоичных чисел.
- Компаратор имеет две группы входов. На одну из них поступают разряды числа А, на другую группу – разряды числа В.
- Появление одиночного сигнала на одном из трех выходов компаратора фиксирует результат сравнения. Эти соотношения используются как логические условия (признаки) в микропрограммах, в устройствах автоматического контроля и диагностики и т.д.



Логические аргументы		Логические функции		
a_i	b_i	$F_{A<B}$	$F_{A=B}$	$F_{A>B}$
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

$$F_{A<B} = \bar{a}_i \cdot b_i; F_{A=B} = \bar{a}_i \cdot \bar{b}_i \vee a_i \cdot b_i; F_{A>B} = a_i \cdot \bar{b}_i.$$

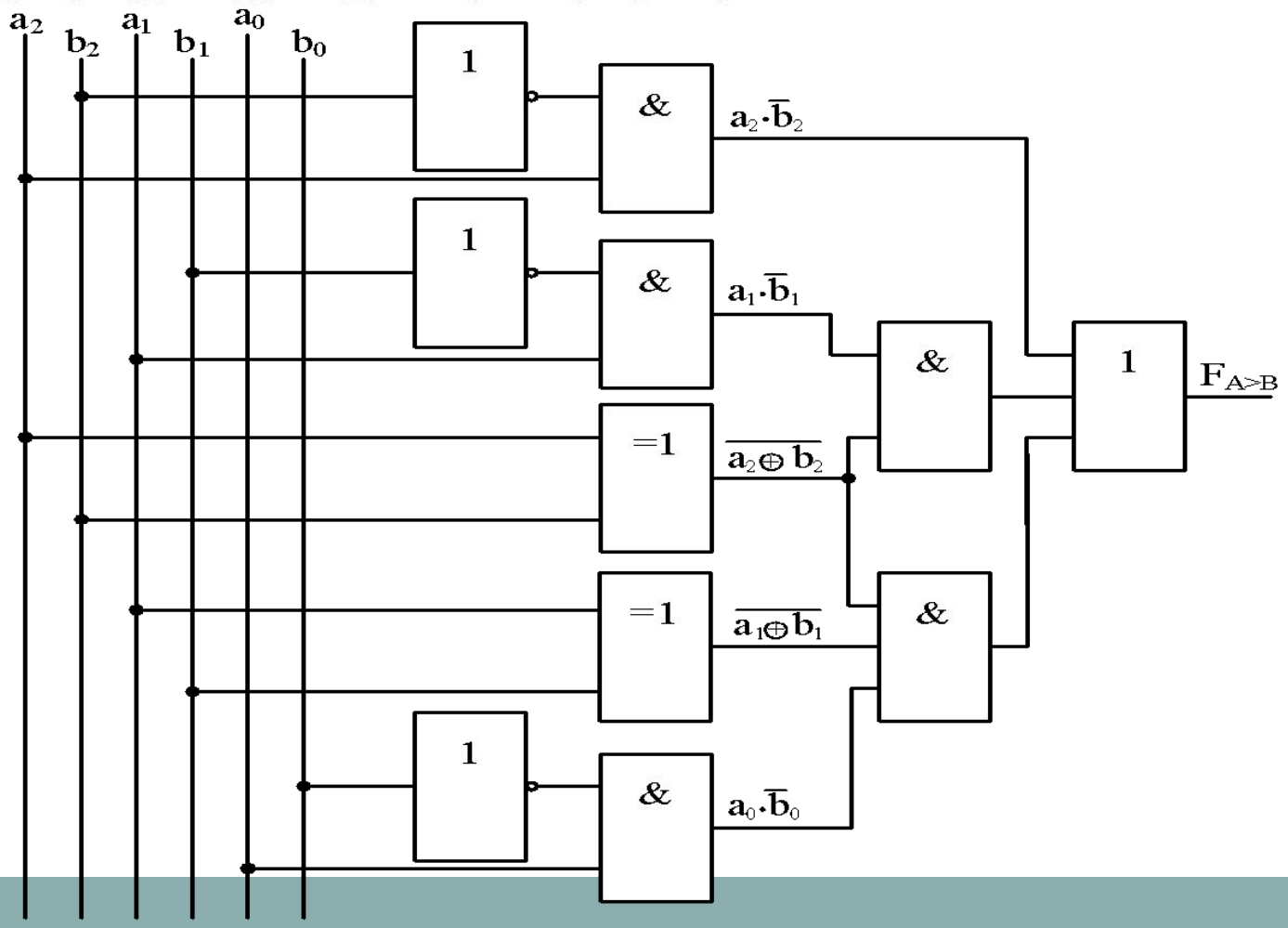
Если значения a_i и b_i таковы, что правые части функций принимают единичные значения, то соотношения, указанные в индексах левых частей, выполняются. Если правые части функций принимают нулевые значения, то соотношения между a_i и b_i противоположны указанным.

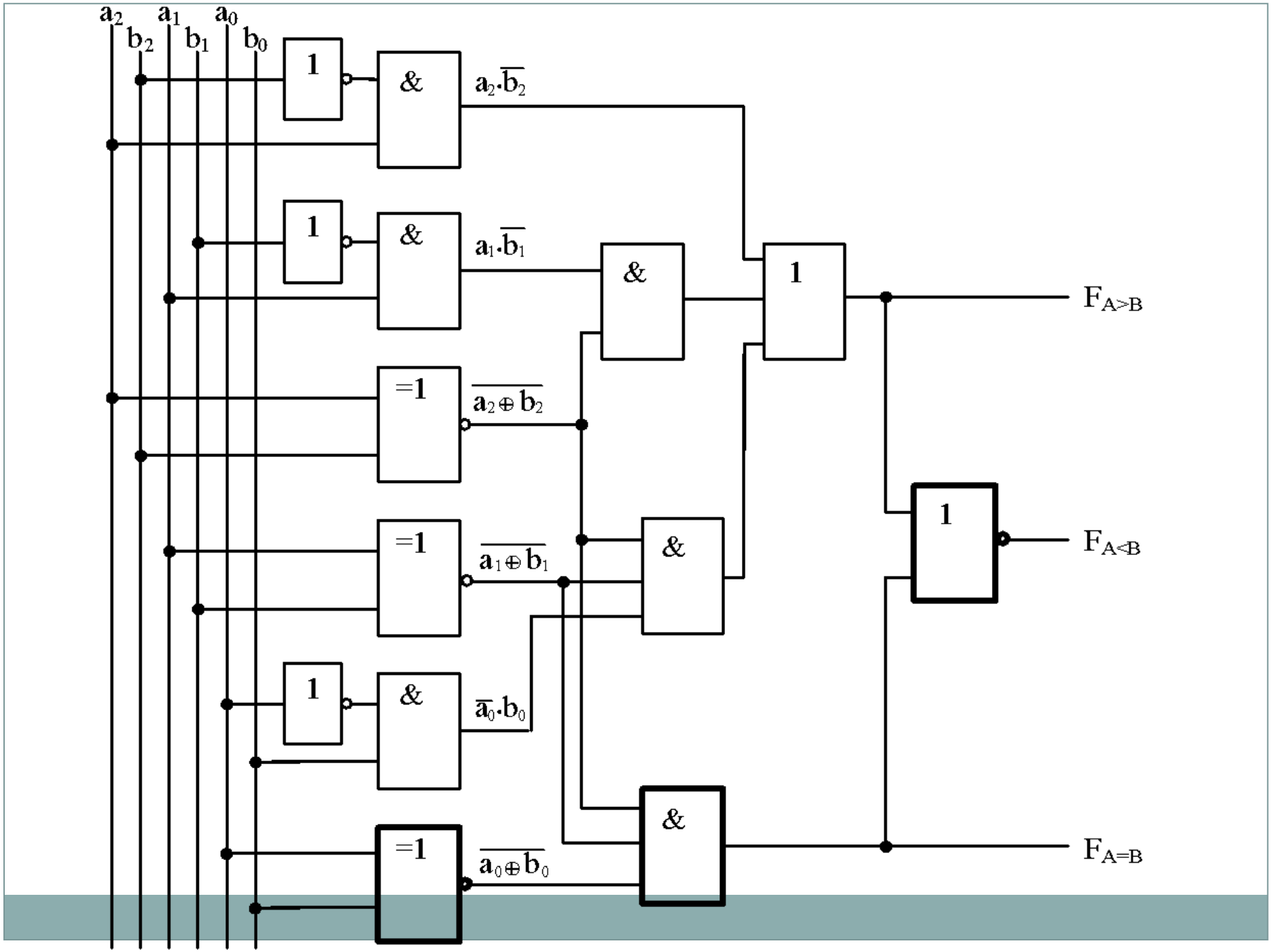

a0
b0
=1
a1
b1
=1
an-1
bn-1
=1
1
1
FA=B

$$F_{A>B} = a_2 \cdot \overline{b_2} \vee \left(\overline{a_2} \cdot \overline{b_2} \vee a_2 \cdot b_2\right) \cdot a_1 \cdot \overline{b_1} \vee$$

$$\vee \left(\overline{a_2} \cdot \overline{b_2} \vee a_2 \cdot b_2\right) \cdot \left(\overline{a_1} \cdot \overline{b_1} \vee a_1 \cdot b_1\right) \cdot a_0 \cdot \overline{b_0} =$$

$$= a_2 \cdot \overline{b_2} \vee \overline{a_2 \oplus b_2} \cdot a_1 \cdot \overline{b_1} \vee \overline{a_2 \oplus b_2} \cdot \vee \overline{a_1 \oplus b_1} \cdot a_0 \cdot \overline{b_0} = 1.$$




a2
a1
a0
b2
b1
b0
1
&
a2.b2
1
&
a1.b1
&
1
FA>B
=1
a2 ⊕ b2
=1
a1 ⊕ b1
&
1
FA<B
1
&
a0.b0
=1
a0 ⊕ b0
&
FA=B