Аннотация курса «Логическое проектирование»

Курс состоит из двух частей: теория переключательных схем и методы логического синтеза и проектирования. В основном излагается материал, не вошедший в программы аналогичных регулярных курсов.

Первая часть курса содержит систематическое описание теории логических схем. Ее целью является ознакомление студентов с основными принципами проектирования синхронных и асинхронных логических устройств в различных логических базисах и приобретение навыков практического проектирования. В конце первой части курса студенты должны усвоить основные понятия и идеи теории переключательных схем, уметь минимизировать реализации логических функций в различных логических базисах, а также проектировать синхронные и асинхронные комбинационные устройства.

Во второй части курса излагаются основы абстрактной и структурной теории синхронных и асинхроннных конечных автоматов с уделением особого внимания методам декомпозиции, минимизации и размещению состояний, а также дополнительным знаниям, необходимым для качественного проектирования модулей вычислительных и информационных устройств. Вторая половина курса обеспечивает приобретение практического опыта проектирования различных устройств. После изучения второй части курса студенты должны усвоить основные понятия и идеи структурной теории синхронных и асинхронных автоматов и уметь проектировать модули и блоки цифровых устройств, вкючая реализацию интерфейсов.

Ключевые слова: логическая функция, переключательная схема, функциональная полнота, гиперкуб, подкуб перехода, минимизация, декомпозиция, минтерм, импликанта, неизбыточная оболочка, мажоритарная логика, пороговая логика, конечный автомат, регулярное выражение, последовательностная схема, распознающий автомат, минимизация автомата, размещение состояний, асинхронный автомат, самосинхронный автомат, пересчетные схемы.



Логическое Проектирование

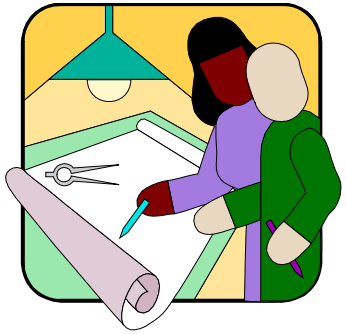
Проф. Вячеслав Борисович Мараховский

Лекция 1

Основная литература

1. Миллер Р., *Теория переключательных схем*. –М.: Наука, т.1, 1970, т.2, 1971.
2. Поспелов Д.А., *Логические методы анализа и синтеза схем*. – М.: Энергия, 1974.
3. Шоломов Л.А., Основы теории дискретных логических и вычислительных устройств. –М.: Наука, 1980.
4. Randy H. Katz, *Contemporary Logic Design*, The Benjamin/ Cummings Publishing Company, Redwood City, California, 1995.
5. Tsutomu Sasao, *Switching Theory for Logic Synthesis, Kluwer Academic Publishers, Feb. 1999/ ISBN: 0-7923-8456-3.*
6. Карпов Ю.Г. Теория автоматов. Учебник: Питер_СПб, 2006.
7. Антонов А.П., Мелехин В.Ф., Митина Т.М., Тарабукин В.И., Филиппов А.С. Структурный синтез цифровых устройств: учебное пособие. - СПб.: СПбГТУ, 2006.

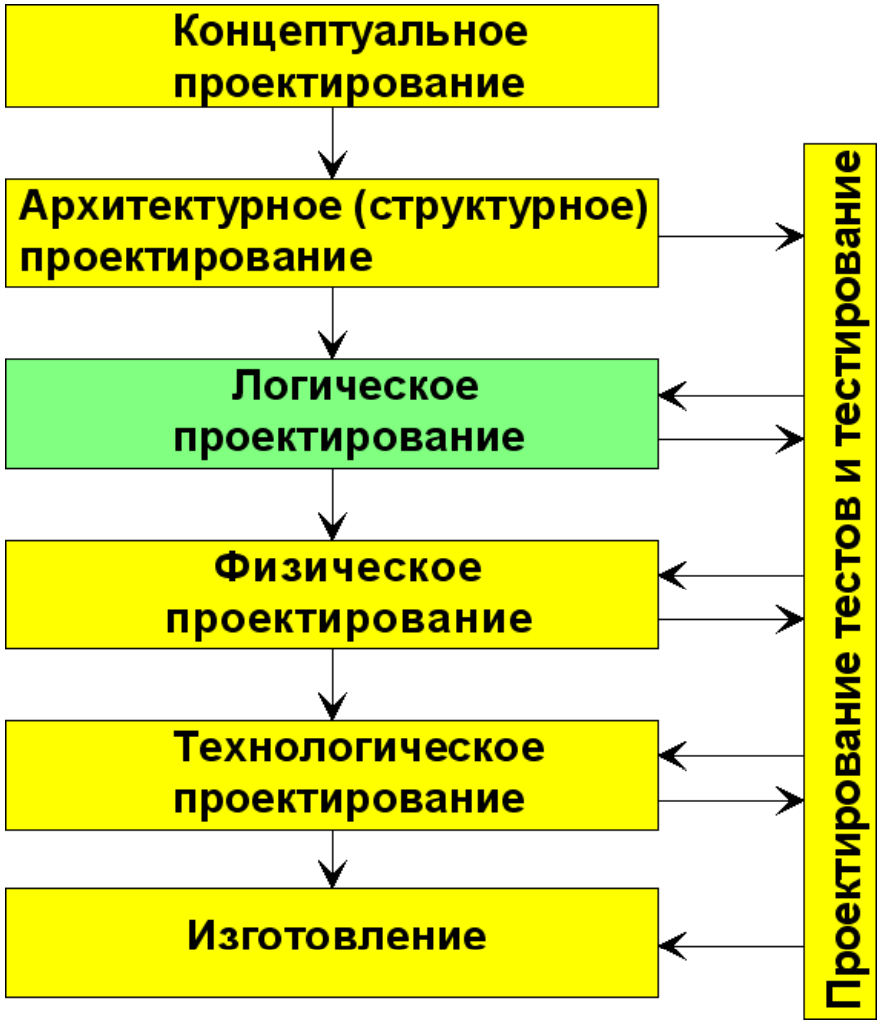
➢ *Современное логическое проектирование*



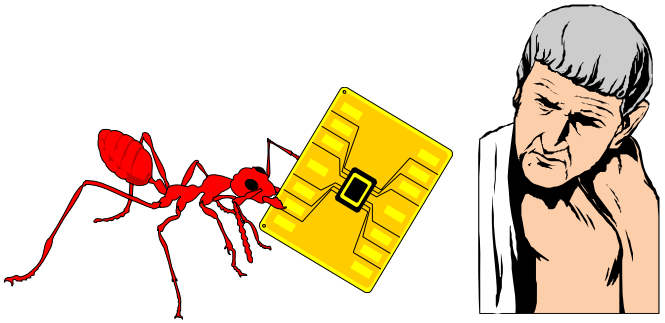
Проектирование – создание описаний объектов, удобное для их производства.

Проектирование предусматривает использование формальных методов и в то же время является искусством (ремеслом).

Ход проектирования аппаратуры



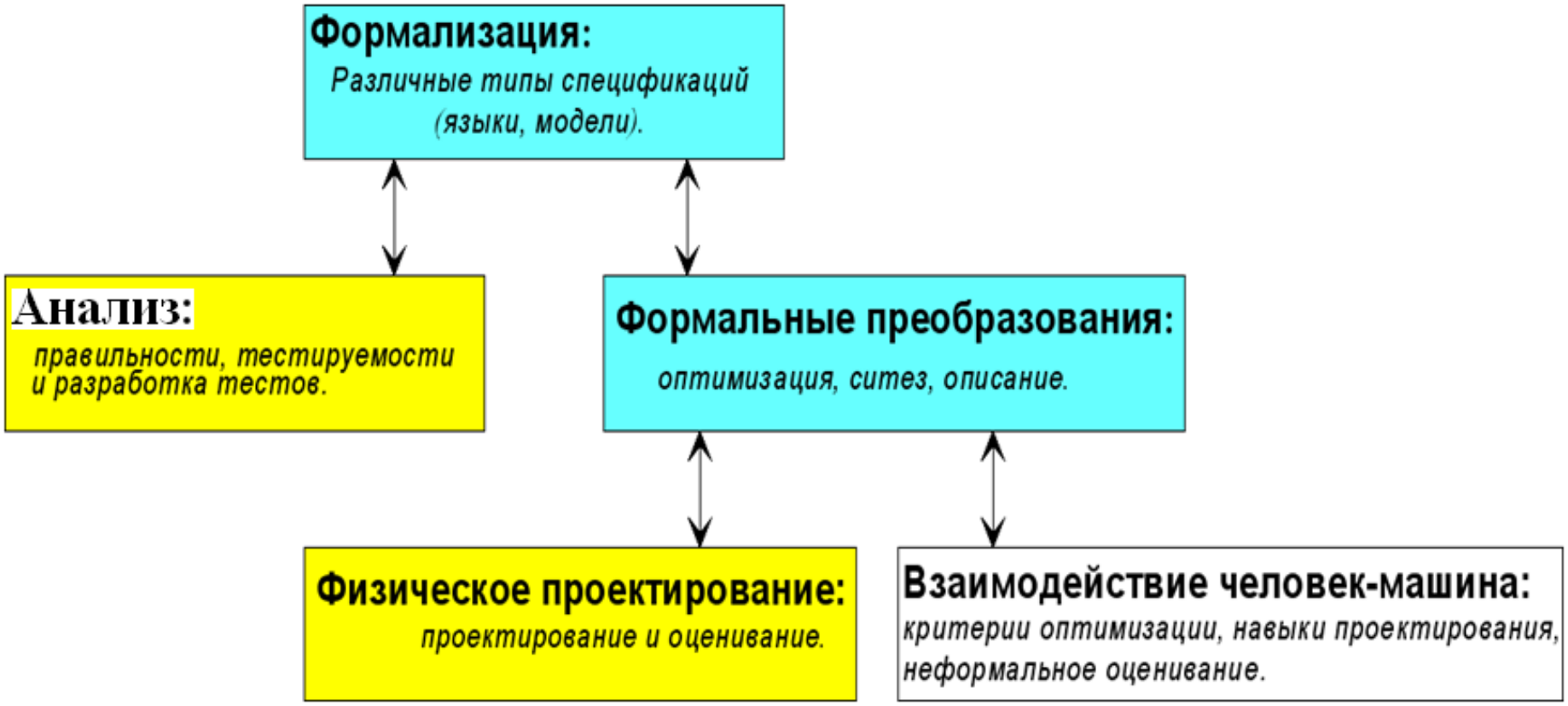
➢ *Современное логическое проектирование*



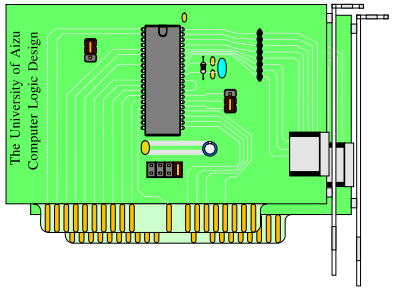
Логика – (математическая, или символическая, логика) является математической дисциплиной, которая, в частности, изучает отношения между высказываниями и правила вывода.

Логическое проектирование – создание описаний объектов в терминах высказываний о их компонентах (состояний компонентов).

Ход логического проектирования



➢*Современное* логическое проектирование



Современное – означает , что мы будем рассматривать различные подходы к логическому проектированию и дополнительные темы, которые не включены в регулярные курсы.

Мы попытаемся научиться, как думать во время пректирования, как освоить это искусство. Умение думать как раз и является искусством, подобным игре на музыкальных инструментах или работе художника.

Обучение искусству включает:

- **обучение формальным методам и правилам;**
- **обучение неформальным методам и мастерству на примерах;**
- **тренировка и натаскивание;**
- **взаимодействие с преподавателем.**

Кроме изучения формальных методов мы будем решать задачи на примерах.

В качестве первого примера подумайте, как решить следующую задачу:

Имеется N-bit регистр.

Спроектируйте устройство, которое отвечает на вопрос: делится ли на три двоичное число, записанное в этот регистр, или нет.

Очень короткое введение в исчисление высказываний и предикатов

Высказывание – это утверждение (выражение), относительно которого может быть сделано заключение о его истинности или ложности.

“6 – число чётное” – истина;
“5 – число чётное” – *ложь;*
“Сергей Иванов – студент магистратуры” – *?*

Предикат – это функция, значение которой является высказыванием о множестве из *N* объектов.
Когда *N=1,* предикат называется *“свойством”*,
Когда N>1, предикат назевается *“отношением”.*

Пусть *M* – множество чётных чисел и *X* – произвольное число.
“X ∈ M” (“X – чётное число*”)* – истина или ложь зависит от аргумента *X.*
“Этот студент учится в магистратуре” является *свойством;*
“Все студенты этой группы учатся в магистратуре” – это *отношение.*

Исчисление предикатов (Операции)

$\neg F$ - отрицание; (Not)

$(F \,\&\, G)$ - конъюнкция; (And)

$(F \vee G)$ - дизъюнкция; (Or)

$(F \rightarrow G)$ - импликация; (Если... То)

$\forall x \in F$ - квантор общности; (Для всех *x* в *F*)

$\exists x \in F$ - квантор существования; (Существует *x* в *F* , такой что…)

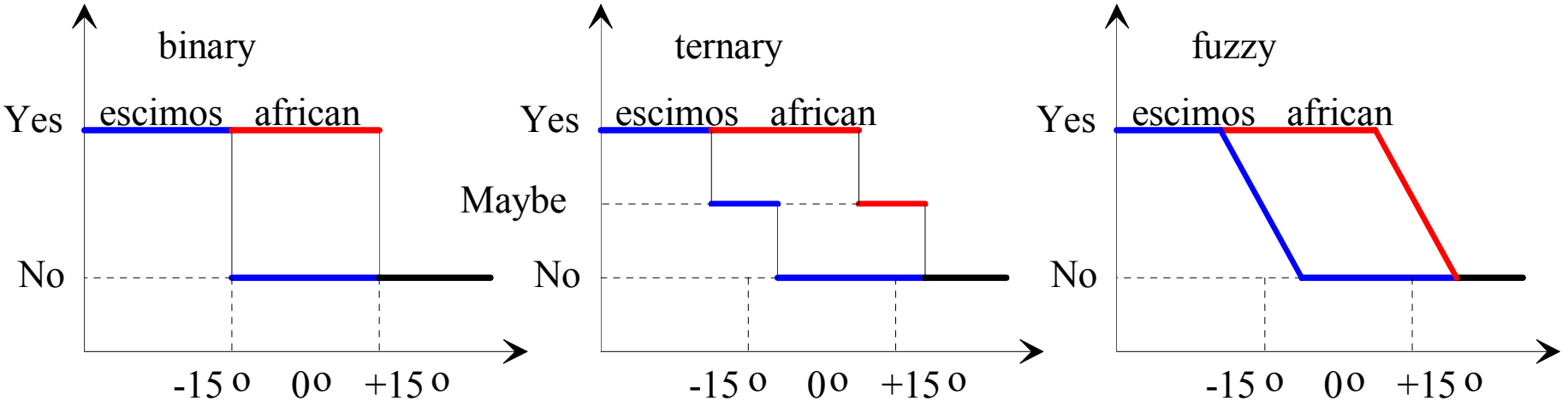
Возможные значения

Двоичная логика оперирует только с двумя значениями: “истина”, “ложь”; “да”, “нет”; “1”, “0”.

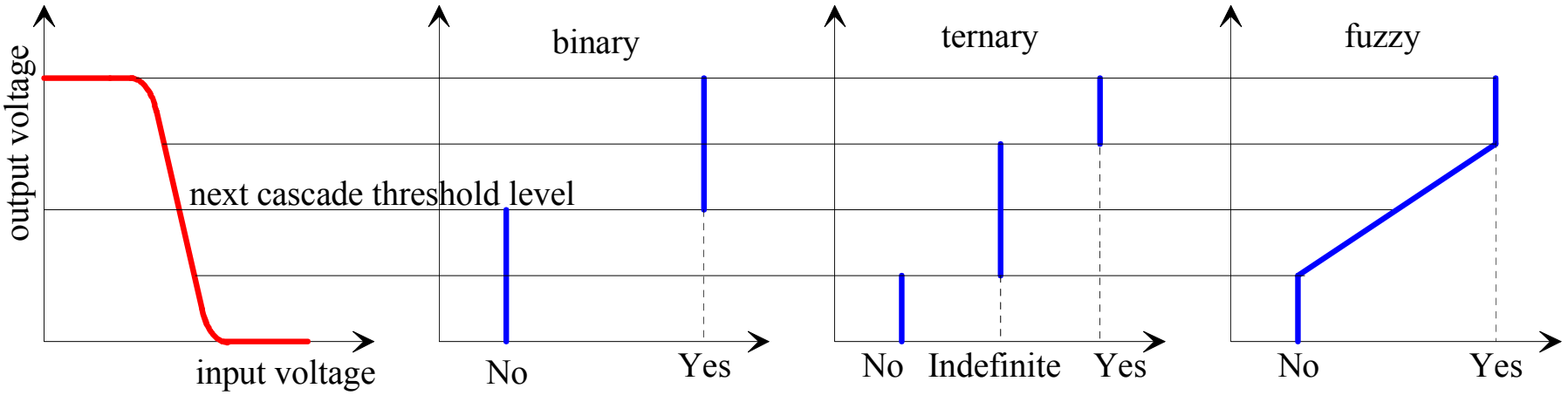
Многозначная логика может иметь больше значений, например: “истина”, “неопределенно”, “ложь”; “да”, “может быть”, “нет”; “1”, “1/2”,“0”.

Размытая логика (Fuzzy Logic) имеет значения, непрерывно распределенные между “истина” и “ложь” (функция принадлежности).

“Сегодня холодно” – результат зависит от точки зрения.



“На выходе элемента высокое напряжение” – результат зависит от порогового напряжения следующего элемента и момента времени.



Алгебраические основы логического проектирования

- Множества
- Отношения
- Частичное упорядочение
- Решетки
- Булева алгебра

ОТНОШЕНИЯ

1. Прямое произведение:

Декартово прямое произведение $D = A \times B$ является множеством, состоящим из всех упорядоченных пар элементов (a_i, b_j)

$$a_i \in A = \{a_1, \ldots, a_n\}, b_j \in B = \{b_1, \ldots, b_m\}:$$

$$D = \{(a_1, b_1), \ldots, (a_1, b_m), \ldots, (a_n, b_1), \ldots (a_n, b_m)\}.$$

Пример:

$$A = \{0,1\}, B = \{1,2\};$$

$$A \times B = \{(0,1), (0,2), (1,1), (1,2)\};$$

$$A^2 = A \times A = \{(0,0), (0,1), (1,0), (1,1)\}.$$

2. Двоичное отношение:

Двоичное отношение R между A в B является подмножеством множества $D = A \times B$. Иными словами, если $(a_i, b_j) \in R \subseteq A \times B$, то a_i и b_j связаны отношением R: $a_i R b_j$.

3. Обратное отношение:

Обратное отношение R^{-1} определяется как $B \times A : R^{-1} = \{(b_j, a_i) | (a_i, b_j) \in R\}$.

Частично упорядоченные множества

Частично упорядоченным множеством называется множество *A, на котором определено бинарное отношение* $a_i \le a_j$ **, удовлетворяющее для всех *i* и *j* следующим условиям:**

- *рефлексивность* ($a_i \le a_i$),
- *антисимметричность* (если $a_i \le a_j$ и $a_j \le a_i$, то $a_i = a_j$),
- *транзитивность* (если $a_i \le a_j$ и $a_j \le a_k$, то $a_i \le a_k$).

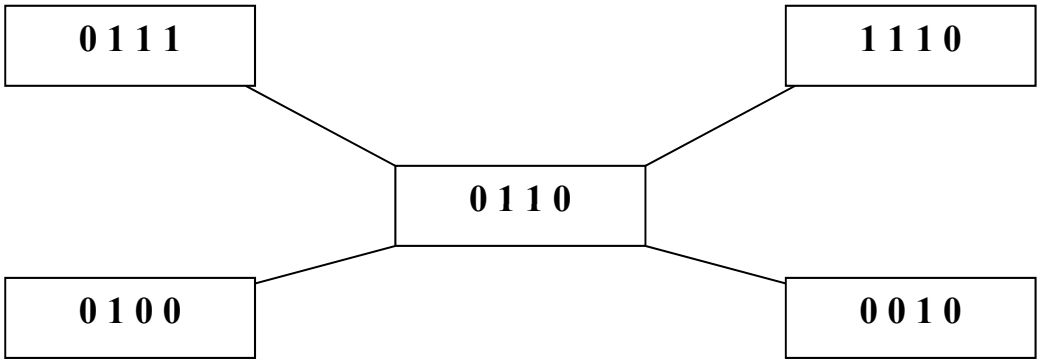
Диаграмма Хасса

Пусть $a, b \in A$, $a \overset{R}{<} b$ и отсутствует элемент $c \in A$, такой что $a \overset{R}{<} c$ и $c \overset{R}{<} b$, тогда на диаграмме b располагается выше a, и соединяется a линией: b | a .

Пример:

Два вектора $A = a_1, a_2, ..., a_k$ and $B = b_1, b_2, ..., b_k$ связаны отношением $A \overset{R}{<} B$ тогда и только тогда, когда $a_i \le b_i$ для всех $i = 1,2,...,k$.

Пусть A = { 0111, 1110, 0110, 0100, 0010}, отношение $\overset{R}{<}$ на диаграмме Хасса представляется как:



РЕШЁТКА

По терминологии Биркгофа *решеткой* называется частично упорядоченное множество A, в котором любые два элемента имеют точную нижнюю грань, или « пересечение», обозначаемое $x \wedge y$, и точную верхнюю грань, или «объединение» , обозначаемое $x \vee y$. Любая непустая решетка содержит наименьший элемент O и наибольший элемент I.

Иными словами, решетка является алгебраической системой $\langle A, \vee, \wedge \rangle$ с двумя бинарными операциями и для любых элементов a, b, c, принадлежащих A, для этих операций должны выполняться следующие законы:

(1) идемпотентность: $a \vee a = a,\ a \wedge a = a$;
(2) коммутативность: $a \vee b = b \vee a,\ a \wedge b = b \wedge a$;
(3) ассоциативность: $a \vee (b \vee a) = (a \vee b) \vee c,\ a \wedge (b \wedge c) = (a \wedge b) \wedge c$;
(4) поглощение: $a \vee (a \wedge b) = a,\ a \wedge (a \vee b) = a.$

Кроме того, неравенство $a \le b$ равносильно каждому из условий $a \wedge b = a$ и $a \vee b = b$ (совместимость).

Булева алгебра

Булева алгебра представляет собой множество элементов *B*, для которых определено отношение эквивалентности (=) и три операции:
объединение (или дизъюнкция, или логическое сложение), обозначается как $\vee$, | или + ;
логическое умножение (или конъюнкция), обозначается как $\wedge$, & или • ;
отрицание (или инверсия), обозначаемая как $\overline{x}$, или x'.
Множество *B* является замкнутым, т.е. оно содержит константы “0” и “1”, переменные и все выражения, получаемые с помощью логических операций из переменных и выражений.

Отношение эквивалентности удовлетворяет следующим свойствам: *рефлексивность* (*a*=*a*), *симметричность* (если *a*=*b*, то *b*=*a*) и *транзитивность* (если *a*=*b* и *b*=*c*, то *a*=*c*).

Аксиомы булевой алгебры:

Тождества: $a \vee 0 = a, \;\; a \cdot 1 = a$;

Идемпотентные законы: $a \vee a = a, \;\; a \cdot a = a$;

Дополнения: $a \vee \overline{a} = 1, \;\; a \cdot \overline{a} = 0$;

Коммутативные законы: $a \vee b = b \vee a, \;\; a \cdot b = b \cdot a$;

Дистрибутивные законы: $a \vee (b \cdot c) = (a \vee b) \cdot (a \vee c), \;\; a \cdot (b \vee c) = (a \cdot b) \vee (a \cdot c)$;

Ассоциативные законы: $a \vee (b \vee c) = (a \vee b) \vee c, \;\; a \cdot (b \cdot c) = (a \cdot b) \cdot c.$

Булевы функции. Основные определения

Переменные: x_i .

Значения переменных: $\omega_i = \{0,1\}$.

Множество переменных (Вектор): $X = \{x_0, x_1, ..., x_{n-1}\}$.

Множество значений (Вектор): $\Omega = \{\omega_0, \omega_1, ..., \omega_{n-1}\}$.

Число множества: $N(\Omega) = \sum_{j=0}^{n-1} 2^j \omega_j .$

Для n переменных $0 \le N(\Omega) \le 2^n - 1$.

Пример:

$n = 3;\ \Omega = \{0,1,1\};$

$N(\Omega) = 2^0 \bullet 0 + 2^1 \bullet 1 + 2^2 \bullet 1 = 6;\ x_0 = 0;\ x_1 = 1;\ x_2 = 1.$

Функции одной переменной

Функции $F_J(X)$	X		Содержание Функций
	0	1	
$F_0(X)$	0	0	Константа 0 $F_0(X)=0$
$F_1(X)$	1	0	Инверсия (отрицание) $F_1(X) = \overline{X}$
$F_2(X)$	0	1	Переменная $F_2(X)=X$
$F_3(X)$	1	1	Константа 1 $F_3(X)=1$

Номер функции: $J = 2^0 \bullet F(0) + 2^1 \bullet F(1)$

Функции двух переменных

x_0	0	1	0	1	Содержание
x_1	0	0	1	1	функций
$F_0(X)$	0	0	0	0	Константа 0
$F_1(X)$	1	0	0	0	Стрелка Пирса (NOR) $x_0 \downarrow x_1$
$F_2(X)$	0	1	0	0	x_0 AND Not x_1
$F_3(X)$	1	1	0	0	Not x_1
$F_4(X)$	0	0	1	0	Not x_0 AND x_1
$F_5(X)$	1	0	1	0	Not x_0
$F_6(X)$	0	1	1	0	Сумма по модулю 2 $x_0 \oplus x_1$
$F_7(X)$	1	1	1	0	Штрих Шеффера (NAND) $x_0 \mid x_1$
$F_8(X)$	0	0	0	1	И x_0 AND x_1, $x_0 \& x_1$, $(x_0 x_1)$
$F_9(X)$	1	0	0	1	Эквивалентность $x_0 \sim x_1$
$F_{10}(X)$	0	1	0	1	x_0
$F_{11}(X)$	1	1	0	1	Импликация $x_1 \rightarrow x_0$
$F_{12}(X)$	0	0	1	1	x_1
$F_{13}(X)$	1	0	1	1	Импликация $x_0 \rightarrow x_1$
$F_{14}(X)$	0	1	1	1	ИЛИ $x_0 + x_1$ $(x_0 \vee x_1)$
$F_{15}(X)$	1	1	1	1	Константа 1

Если x_1, то x_0

Если x_0, то x_1

Диаграмма Хасса для булевых функций двух переменных

4 minterms

3 minterms

2 minterms

1 minterms

0 minterms

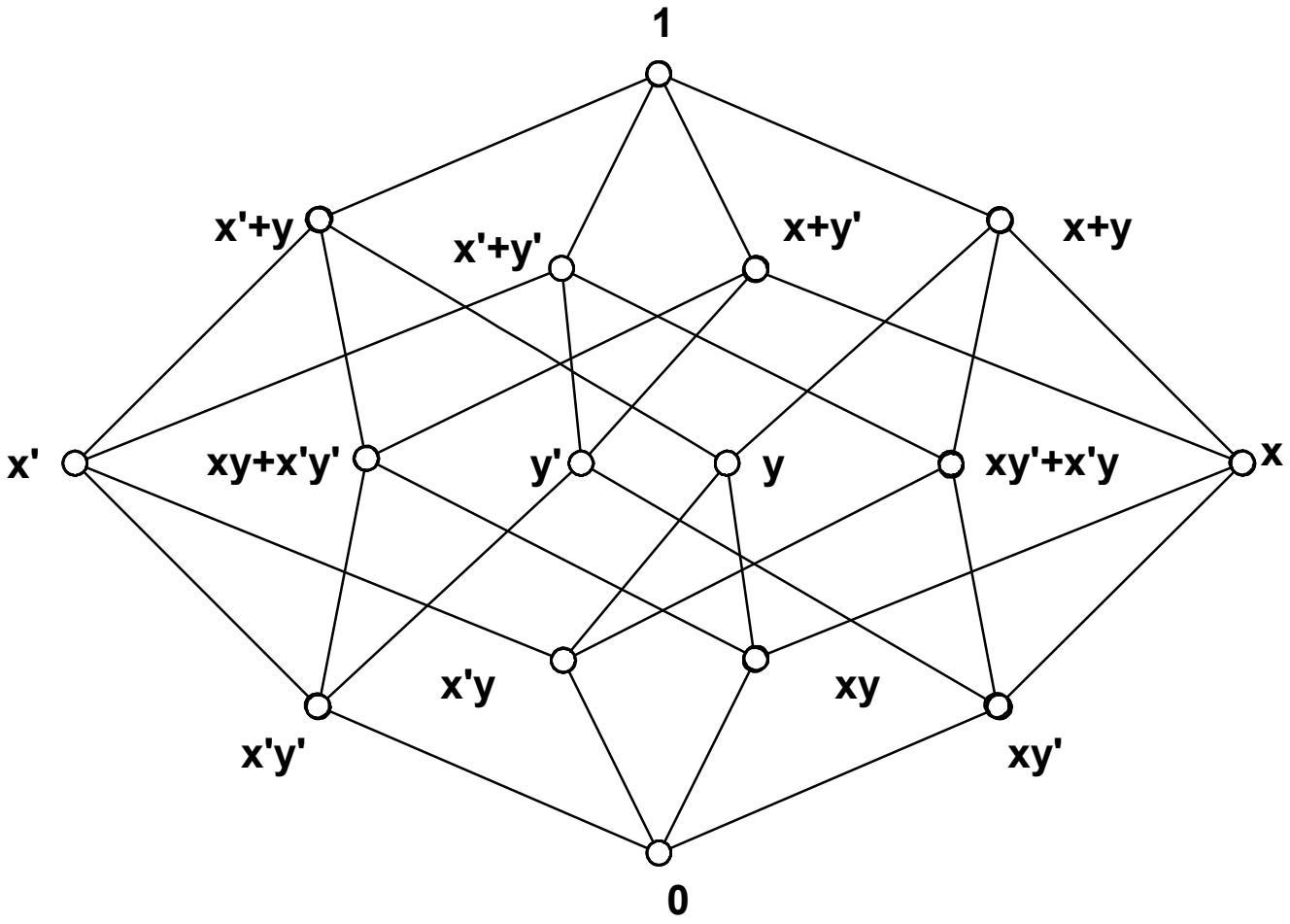


Диаграмма Хасса для булевых функций двух переменных

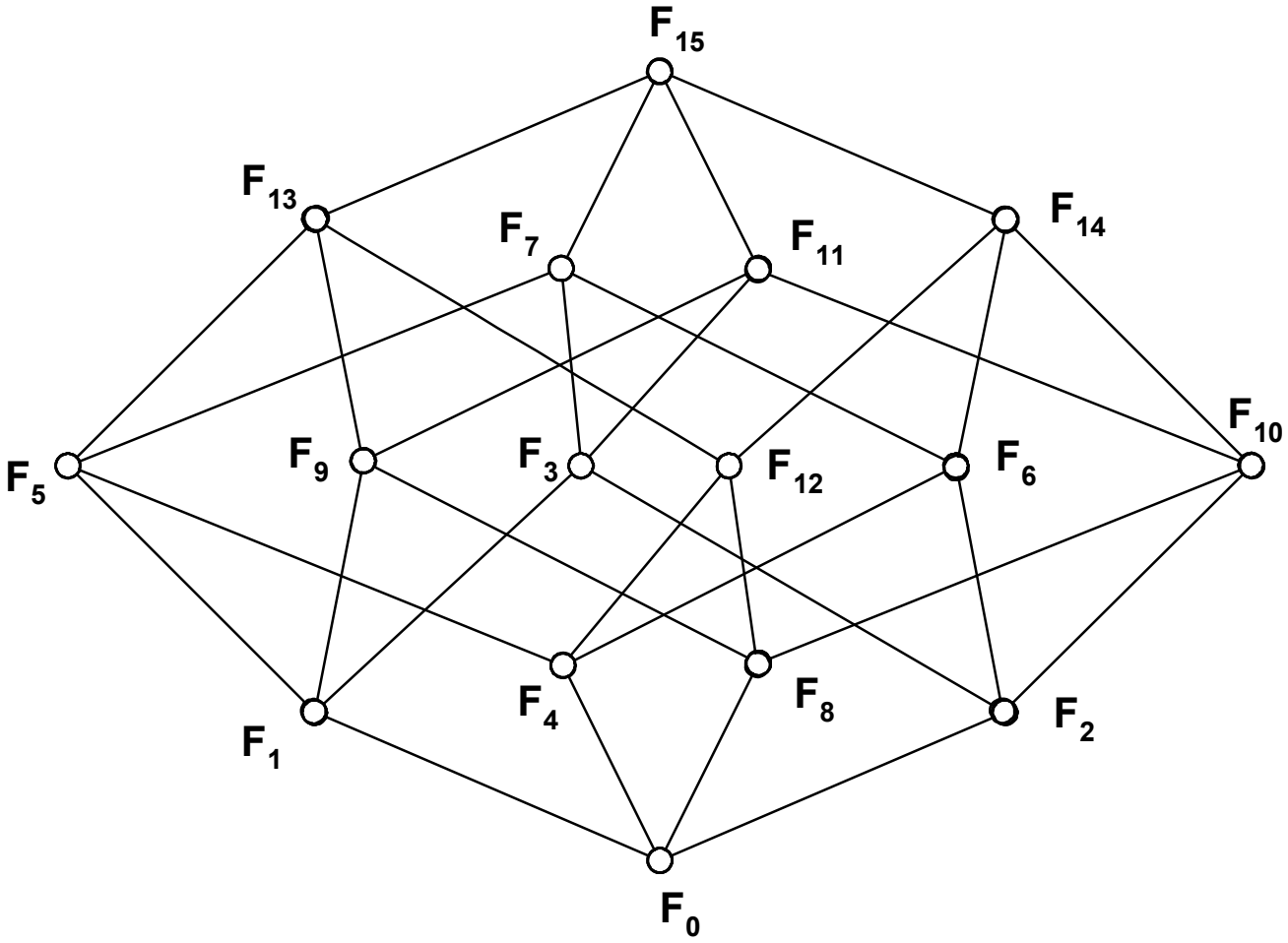
4 minterms

3 minterms

2 minterms

1 minterms

0 minterms



Номер функции вычисляется как : $$j = \sum_{N(\Omega)=0}^{2^n-1} 2^{N(\Omega)} \bullet F(\Omega)$$

Отношения между векторами: пусть $X = \{x_0, x_1, ..., x_{n-1}\}$

$\Omega = \{\omega_0, \omega_1, ..., \omega_{n-1}\}$

тогда $\overline{X} = \{\overline{x}_0, \overline{x}_1, ..., \overline{x}_{n-1}\}$

$\overline{\Omega} = \{\overline{\omega}_0, \overline{\omega}_1, ..., \overline{\omega}_{n-1}\}$

$$\overline{\omega} = 1 - \omega; \quad \overline{\Omega} = \{1-\omega_0, 1-\omega_1, ..., 1-\omega_{n-1}\}$$

$$N(\overline{\Omega}) = \sum_{i=0}^{n-1} 2^i \bullet (1-\omega_i) = \sum_{i=0}^{n-1} 2^i - \sum_{i=0}^{n-1} 2^i \omega_i = 2^n - 1 - N(\Omega)$$

Функция является вектором: $F(X) = \{F(\Omega_0), F(\Omega_1), ...F(\Omega_{2^n-1})\}$

Если *j* – номер $F_j(X)$, то: $\overline{F}_j(X) = F_{2^{2^n}-1-j}(X)$

Пример:

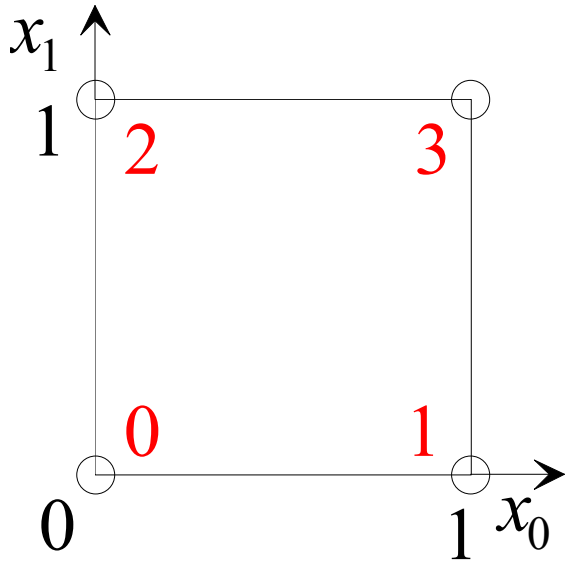
$n = 2;\quad 2^{2^2} - 1 = 15;$

$\overline{F}_0(X) = F_{15}(X);\ \overline{F}_1(X) = F_{14}(X);\ \overline{F}_2(X) = F_{13}(X);\ \overline{F}_3(X) = F_{12}(X);$

$\overline{F}_4(X) = F_{11}(X);\ \overline{F}_5(X) = F_{10}(X);\ \overline{F}_6(X) = F_9(X);\ \overline{F}_7(X) = F_8(X).$

Функция – это: **таблица => вектор => число => формула**

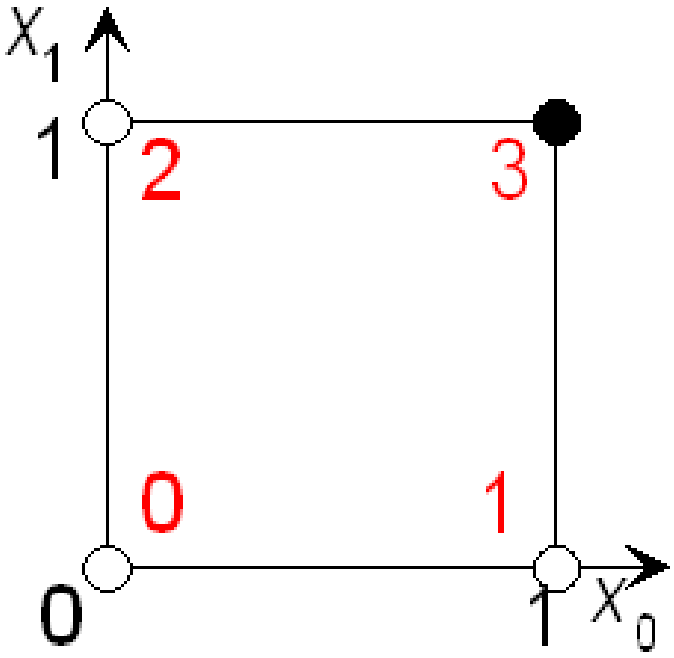
Пусть $x^{\omega} = \begin{cases} 0 & \text{if } x \neq \omega \\ 1 & \text{if } x = \omega \end{cases}$ **, тогда** $x^1 = x;\ x^0 = \overline{x}$ **.**



Графическое представление наборов значений переменных для *n*=2

Свойства функций двух переменных

$F_8(X)$



$$F_8(X) = x_0 x_1 = x_0 \,\&\, x_1 = \min(x_0, x_1);$$

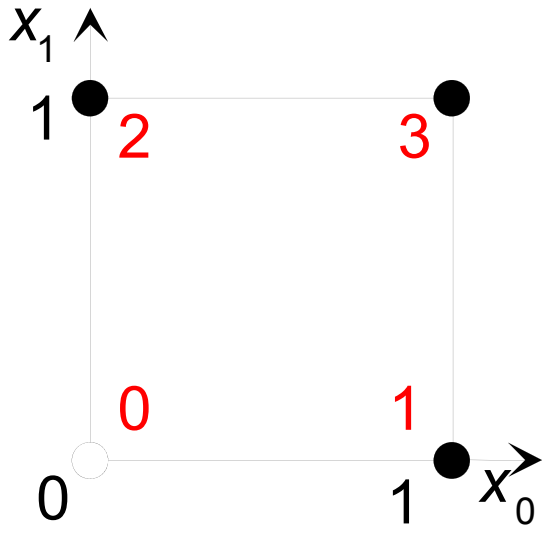
$$x \,\&\, 0 \equiv 0;$$

$$x \,\&\, 1 = x;$$

$$x \,\&\, x = x;$$

$$x \,\&\, \bar{x} = 0.$$

$F_{14}(X)$



$$F_{14}(X) = x_0 + x_1 = x_0 \vee x_1 = \max(x_0, x_1);$$

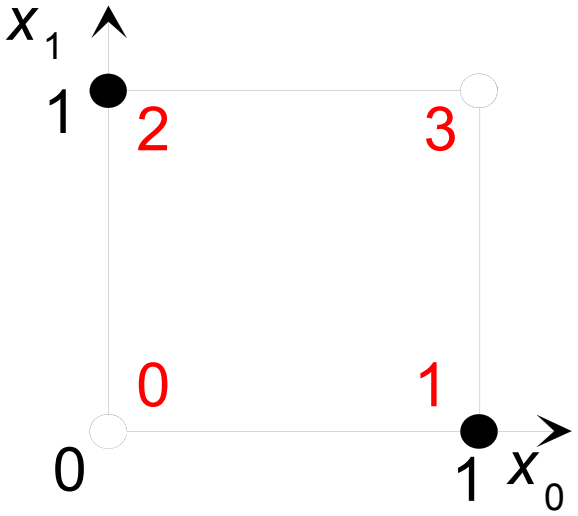
$$x + 0 = x;$$

$$x + 1 \equiv 1;$$

$$x + x = x;$$

$$x + \bar{x} = 1.$$

$F_6(X)$ и $F_9(X)$

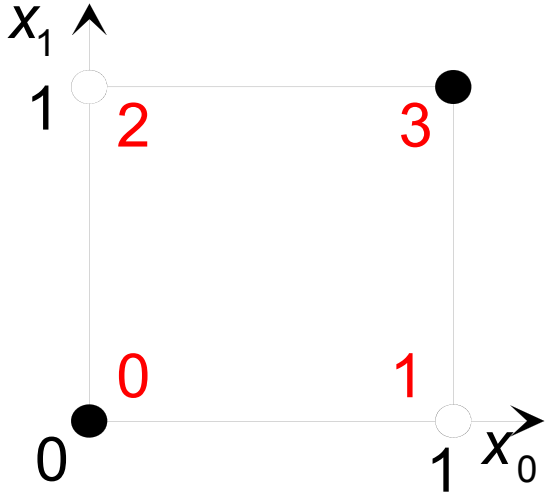


$$F_6(X) = x_0 \oplus x_1 = x_0\bar{x}_1 + \bar{x}_0 x_1;$$

$$x \oplus 0 = x;\ x \oplus 1 = \bar{x};$$

$$x \oplus x = 0;\ x \oplus \bar{x} = 1.$$

$$\bar{x}_0 \oplus x_1 = \overline{x_0 \oplus x_1}$$



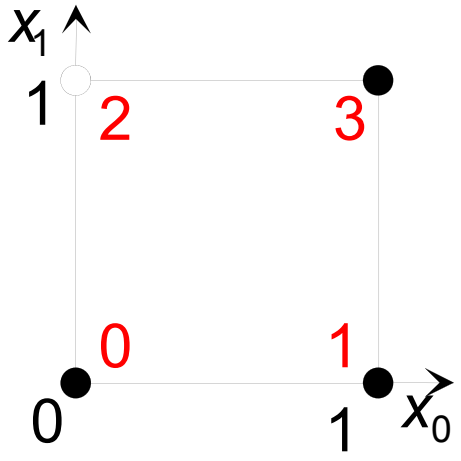
$$F_9(X) = x_0 \sim x_1 = x_0 x_1 + \bar{x}_0 \bar{x}_1 = x_0^{x_1} = x_1^{x_0};$$

$$x \sim 0 = \bar{x};\ x \sim 1 = x;$$

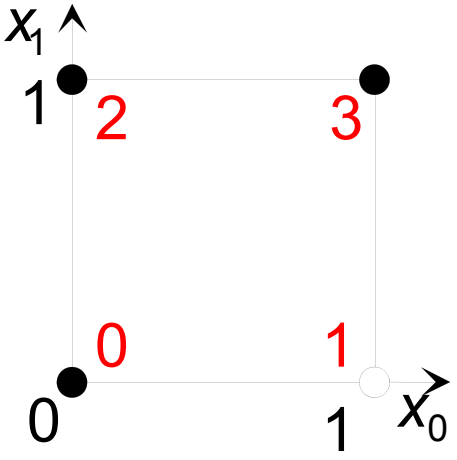
$$x \sim x = 1;\ x \sim \bar{x} = 0;$$

$$\bar{x}_0 \sim x_1 = x_0 \oplus x_1 = \overline{x_0 \sim x_1}.$$

$F_{11}(X)$ и $F_{13}(X)$



$$F_{11}(X) = x_1 \to x_0 = x_0 + \bar{x}_1;$$
$$x \to 0 = \bar{x};\ \ x \to 1 = 1;$$
$$1 \to x = x;\ \ 0 \to x = 1;$$
$$x \to \bar{x} = \bar{x};\ \ x \to x = 1;$$
$$\bar{x} \to x = x.$$

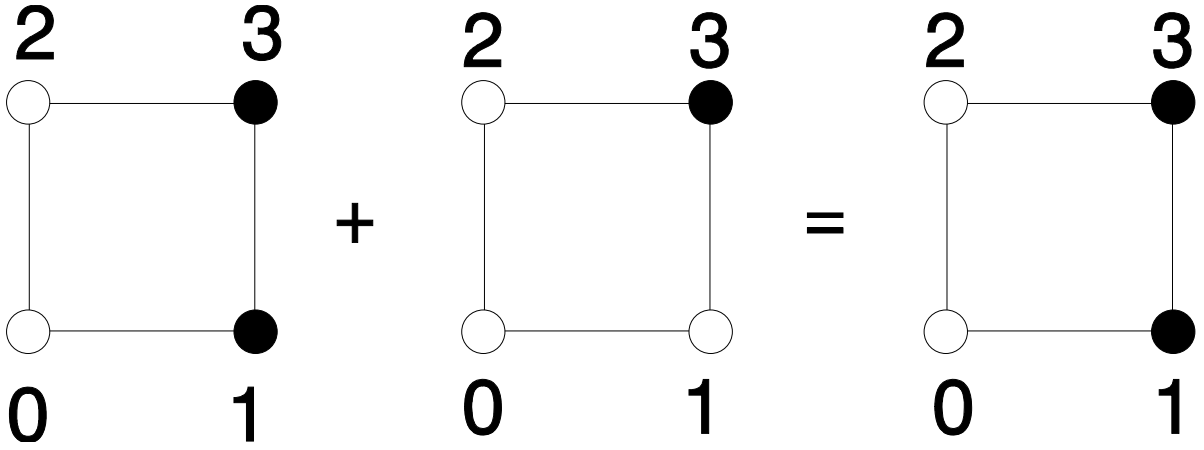


Инвертирование переменных равносильно зеркальному отражению рисунка

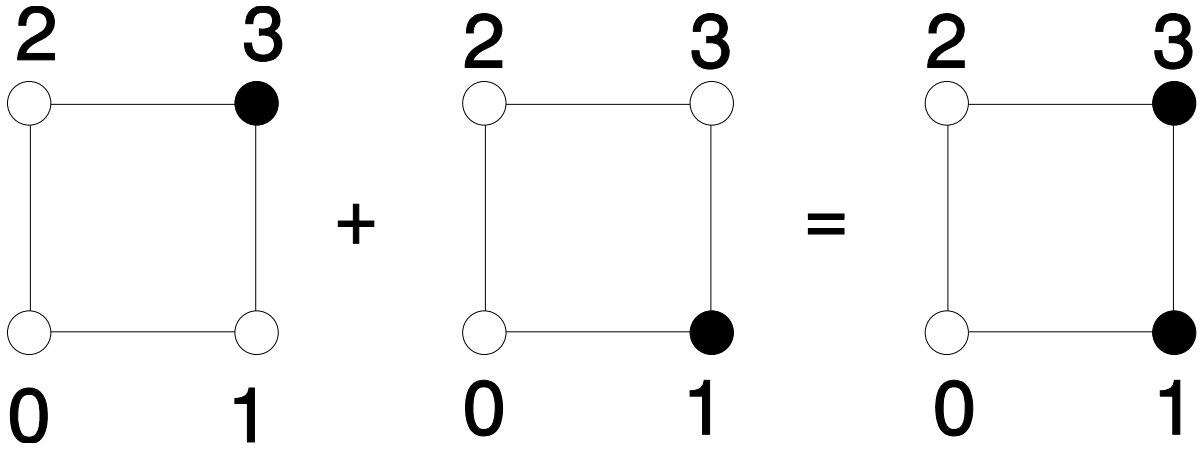
$$F_{13}(X) = x_0 \to x_1 = \bar{x}_0 + x_1;$$
$$\bar{x}_0 \to \bar{x}_1 = x_1 \to x_0.$$

Некоторые свойства функций двух переменных

1. $X_0 + X_0X_1 = X_0(1+X_1) = X_0$ (поглощение)

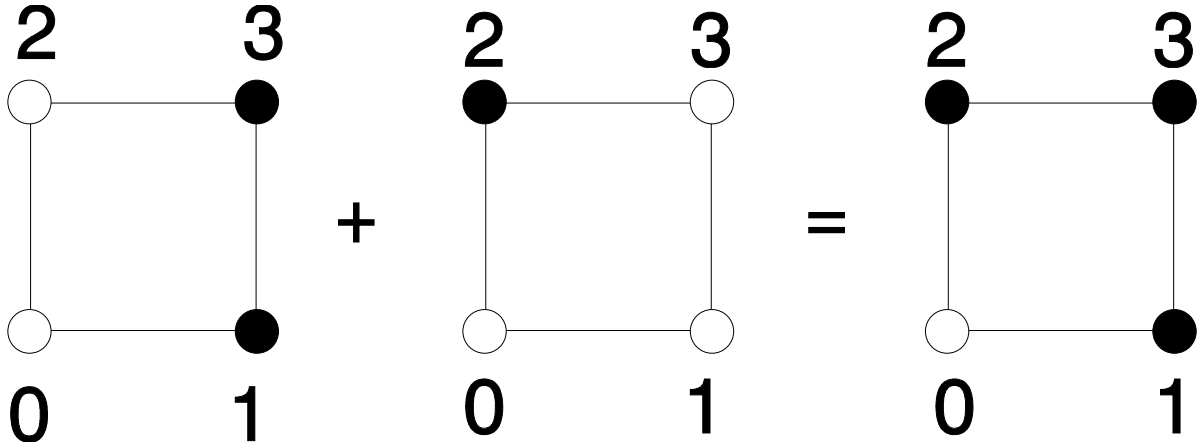


2. $X_0X_1 + X_0\overline{X}_1 = X_0(X_1 + \overline{X}_1) = X_0$ (склеивание)

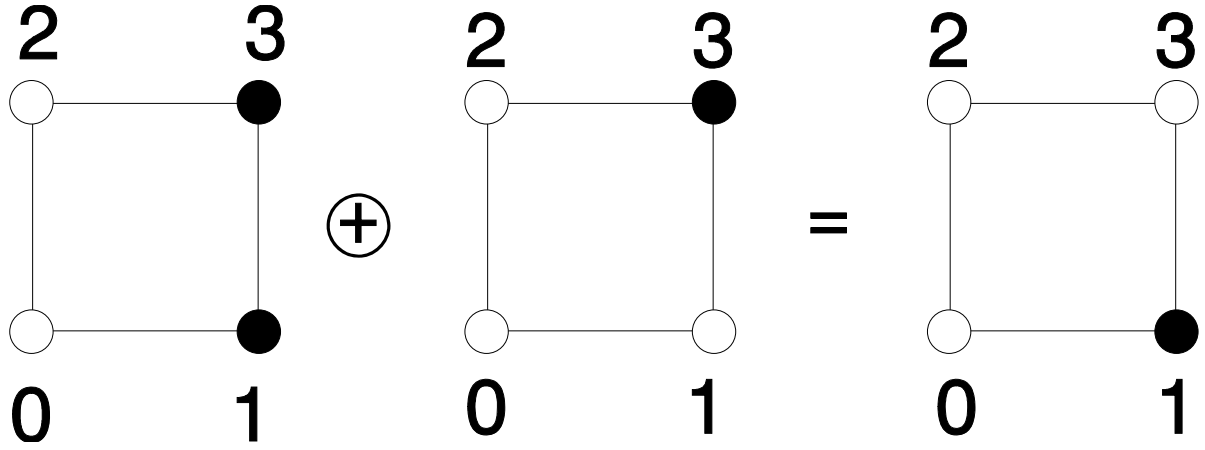


3. $X_0 + \overline{X}_0 X_1 = (X_0 + X_0 X_1) + \overline{X}_0 X_1 =$

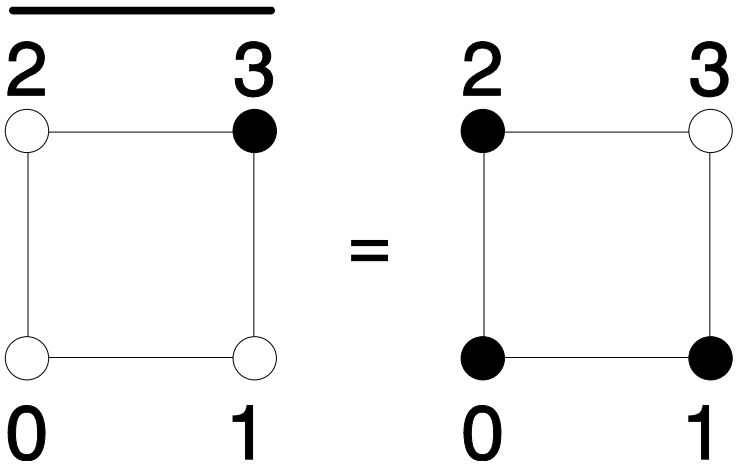
$= X_0 + (X_0 X_1 + \overline{X}_0 X_1) = X_0 + X_1$ (склеивание)



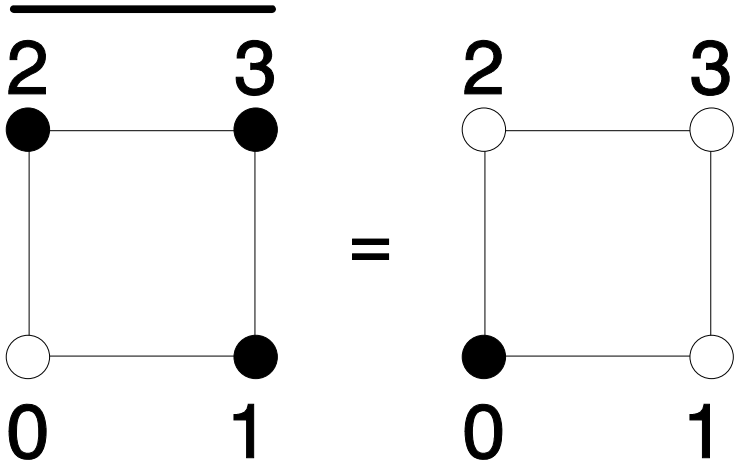
4. $X_0 \oplus X_0 X_1 = X_0 (1 \oplus X_1) = X_0 \overline{X}_1$



5. $\overline{X_0 X_1} = \overline{X}_0 + \overline{X}_1$



6. $\overline{X_0 + X_1} = \overline{X}_0 \overline{X}_1$



$$X_0 \to X_0 X_1 \neq X_0 (1 \to X_1)$$

$$X_0 \to X_0 X_1 = \overline{X}_0 + X_0 X_1 = \overline{X}_0 + X_1 = X_0 \to X_1$$

$$X_0 \to \overline{X}_0 X_1 = \overline{X}_0 + \overline{X}_0 X_1 = \overline{X}_0$$

$$X_0 X_1 \to X_0 = \overline{X_0 X_1} + X_0 = \overline{X}_0 + \overline{X}_1 + X_0 = 1$$

$$\overline{X}_0 X_1 \to X_0 = \overline{\overline{X}_0 X_1} + X_0 = X_0 + \overline{X}_1 + X_0 = X_0 + \overline{X}_1 = X_1 \to X_0$$

$$(X_0 \to X_1) \to X_1 = \overline{(\overline{X}_0 + X_1)} + X_1 = X_0 \overline{X}_1 + X_1 =$$
$$= X_0 + X_1 = \overline{X}_0 \to X_1$$

$$(X_0 \to X_1) \to (X_1 \to X_0) = \quad ?$$

Операции с числами

1. 10 OR **8** = **10**

$$\begin{array}{cccc} 0 & 1 & 0 & 1 \\ + & & & \\ 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & 1 \end{array}$$

2. 8 OR **2** = **10**

$$\begin{array}{cccc} 0 & 0 & 0 & 1 \\ + & & & \\ 0 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 \end{array}$$

3. 10 OR **4** = **14**

$$\begin{array}{cccc} 0 & 1 & 0 & 1 \\ + & & & \\ 0 & 0 & 1 & 0 \\ \hline 0 & 1 & 1 & 1 \end{array}$$

4. 10 XOR **8** = **2**

$$\begin{array}{cccc} 0 & 1 & 0 & 1 \\ \oplus & & & \\ 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 \end{array}$$

Представление функций арифметическими операциями

$$A \bullet B\ ,\ A + B\ ,\ A - B\ .$$

$$X_0 X_1 = X_0 \bullet X_1$$

$$X_0 \overline{X}_1 = X_0 - X_0 \bullet X_1$$

$$X_0 \vee X_1 = X_0 + X_1 - X_0 \bullet X_1$$

$$X_0 \oplus X_1 = X_0 + X_1 - 2 \bullet X_0 \bullet X_1$$

Каждая функция n переменных может быть представлена арифметическим полиномом n-ого порядка

Совершенная нормальная дизъюнктивная форма булевых функций

$$X^{\Omega} = \begin{cases} 1 \text{ if } X = \Omega \\ 0 \text{ if } X \neq \Omega \end{cases}$$

$$X = \Omega \xrightarrow{\forall j} x_j = \omega_j$$

$$X^{\Omega} = x_0^{\omega_0} x_1^{\omega_1} \ldots x_{n-1}^{\omega_{n-1}}$$

Примеры:

$$n = 3; \quad X^5 = x_0^1 x_1^0 x_2^1 = x_0 \bar{x}_1 x_2$$

$$X^6 = x_0^0 x_1^1 x_2^1 = \bar{x}_0 x_1 x_2$$

$$n = 4; \quad X^{10} = x_0^0 x_1^1 x_2^0 x_3^1 = \bar{x}_0 x_1 \bar{x}_2 x_3$$

$$X^7 = x_0^1 x_1^1 x_2^1 x_3^0 = x_1 x_2 x_3 \bar{x}_0$$

X^{Ω} функции для трёх переменных

x_0	x_1	x_2		X^0	X^1	X^2	X^3	X^4	X^5	X^6	X^7	F(X)
0	0	0		1	0	0	0	0	0	0	0	0
1	0	0		0	1	0	0	0	0	0	0	1
0	1	0		0	0	1	0	0	0	0	0	0
1	1	0		0	0	0	1	0	0	0	0	0
0	0	1		0	0	0	0	1	0	0	0	0
1	0	1		0	0	0	0	0	1	0	0	1
0	1	1		0	0	0	0	0	0	1	0	0
1	1	1		0	0	0	0	0	0	0	1	1

Пусть функция трех переменных определена как:

F(1)=F(5)=F(7)=1 и
F(0)=F(2)=F(3)=F(4)=F(6)=0, тогда

$F(X)=X^1+X^5+X^7$

Если функция *n* переменных F(X) определена как множество значений F(Ω), тогда:

$$F(X)=\sum_{\Omega=0}^{2^n-1} F(\Omega)X^{\Omega}$$

Совершенная нормальная конъюнктивная форма булевых функций

$$\overline{X^{\Omega}} = \begin{cases} 0 & \text{if} \quad X = \Omega \\ 1 & \text{if} \quad X \neq \Omega \end{cases}$$

$$X = \Omega \xrightarrow{\forall j} x_j = \omega_j$$

$$\overline{X^{\Omega}} = \bar{x}_0^{\omega_0} + \bar{x}_1^{\omega_1} . + \ldots + \bar{x}_{n-1}^{\omega_{n-1}}$$

примеры:

$$n = 3; \quad \overline{X^5} = \bar{x}_0^1 + \bar{x}_1^0 + \bar{x}_2^1 = \bar{x}_0 + x_1 + \bar{x}_2$$

$$\overline{X^6} = \bar{x}_0^0 + \bar{x}_1^1 + \bar{x}_2^1 = x_0 + \bar{x}_1 + \bar{x}_2$$

$$n = 4; \quad \overline{X^{10}} = \bar{x}_0^0 + \bar{x}_1^1 + \bar{x}_2^0 + \bar{x}_3^1 = x_0 + \bar{x}_1 + x_2 + \bar{x}_3$$

$$\overline{X^7} = \bar{x}_0^1 + \bar{x}_1^1 + \bar{x}_2^1 + \bar{x}_3^0 = \bar{x}_0 + \bar{x}_1 + \bar{x}_2 + x_3$$

$\overline{X^{\Omega}}$ функции для трёх переменных

x_0	x_1	x_2		$\overline{X^0}$	$\overline{X^1}$	$\overline{X^2}$	$\overline{X^3}$	$\overline{X^4}$	$\overline{X^5}$	$\overline{X^6}$	$\overline{X^7}$	F(X)
0	0	0		0	1	1	1	1	1	1	1	0
1	0	0		1	0	1	1	1	1	1	1	1
0	1	0		1	1	0	1	1	1	1	1	0
1	1	0		1	1	1	0	1	1	1	1	0
0	0	1		1	1	1	1	0	1	1	1	0
1	0	1		1	1	1	1	1	0	1	1	1
0	1	1		1	1	1	1	1	1	0	1	0
1	1	1		1	1	1	1	1	1	1	0	1

Пусть функция трех переменных определена как:

F(1)=F(5)=F(7)=1 and F(0)=F(2)=F(3)=F(4)=F(6)=0, тогда

$$F(X) = \overline{X^0} \bullet \overline{X^2} \bullet \overline{X^3} \bullet \overline{X^4} \bullet \overline{X^6}$$

Если функция n переменных F(X) определена как множество значений F(Ω) , то:

$$F(X) = \&_{\Omega=0}^{\Omega=2^n-1} [F(\Omega) + \overline{X^{\Omega}}]$$

Инверсия этой функции имеет вид:

$$\overline{F(X)} = \&_{\Omega=0}^{\Omega=2^n-1} [\overline{F(\Omega)} + \overline{X^{\Omega}}] = \sum_{\Omega=0}^{\Omega=2^n-1} \overline{F(\Omega)} \bullet X^{\Omega}$$

Каждая функция может быть представлена как:

$$F(X) = \Phi_1(X) + \Phi_2(X) \quad \text{or} \quad F(X) = \Psi_1(X) \bullet \Psi_2(X)$$

and

$$\overline{F(X)} = \overline{\Phi_1(X)} \bullet \overline{\Phi_2(X)} = \overline{\Psi_1(X)} + \overline{\Psi_2(X)}$$

Теорема де Моргана:
Для получения инверсии функции необходимо:
инвертировать все переменные;
сделать подстановки: AND вместо OR и
OR вместо AND.



Логическое Проектирование

Лекция 2

Классы эквивалентности логических функций

Классификация логических функций по классам эквивалентности может оказаться очень полезной.

Пример 1: **Если построены минимальные схемы, реализующие логические функции, которые являются представителями классов эквивалентности, то построение минимальной схемы для произвольной функции сводится к поиску в библиотеке минимальной схемы, реализующей ее представителя, и внесению простых изменений в найденную схему.**

Пример 2: **Если построен модуль, реализующий какую либо функцию из класса эквивалентности, то он может быть настроен на реализацию любой функции из этого класса.**

NPN-эквивалентность

Определение: Если логические функции *f* и *g* могут быть получены одна из другой с помощью следующих операций или комбинации этих операций, то функции *f* и *g* являются *NPN-эквивалентными.*

1. **Инвертирование (Negation) некоторых переменных;**
2. **Перестановки (Permutation) некоторых переменных местами;**
3. **Инвертирование (Negation) функции.**

NPN-преобразование

NPN-преобразование – это преобразование одной функции в другую с помощью инвертирования и перестановок переменных, а также инвертирования функции.

NPN-класс функций представляет собой замкнутое множество функций, трансформируемых одна в другую с помощью инвертирования и перестановок переменных и инвертирования функции.

Классификация функций двух переменных

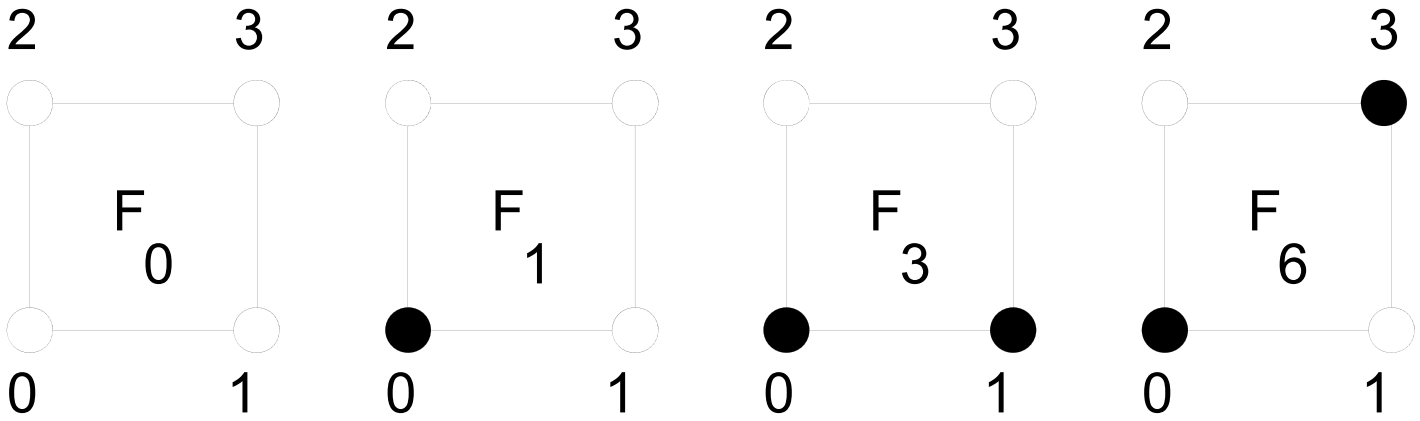
n	Функции	P	NP	NPN
0	0	0	0	0
	1	1	1	
1	x , y	x	x	x
	$\bar{x}$, $\bar{y}$	$\bar{x}$		
	xy	xy	xy	xy
	$\bar{x}y$	$\bar{x}y$		
	$x\bar{y}$			
	$\overline{xy}$	$\overline{xy}$		
2	$x + y$	$x + y$	$x + y$	
	$\bar{x} + y$	$\bar{x} + y$		
	$x + \bar{y}$			
	$\bar{x} + \bar{y}$	$\bar{x} + \bar{y}$		
	$x \oplus y$	$x \oplus y$	$x \oplus y$	$x \oplus y$
	$\bar{x} \oplus y$	$\bar{x} \oplus y$		

Число эквивалентных классов

Число переменных	0	1	2	3	4
Число функций	**2**	**4**	**16**	**256**	**65536**
P-эквивалентность	**2**	**4**	**12**	**80**	**3984**
NP-эквивалентность	**2**	**3**	**6**	**22**	**402**
NPN-эквивалентность	**1**	**2**	**4**	**14**	**222**

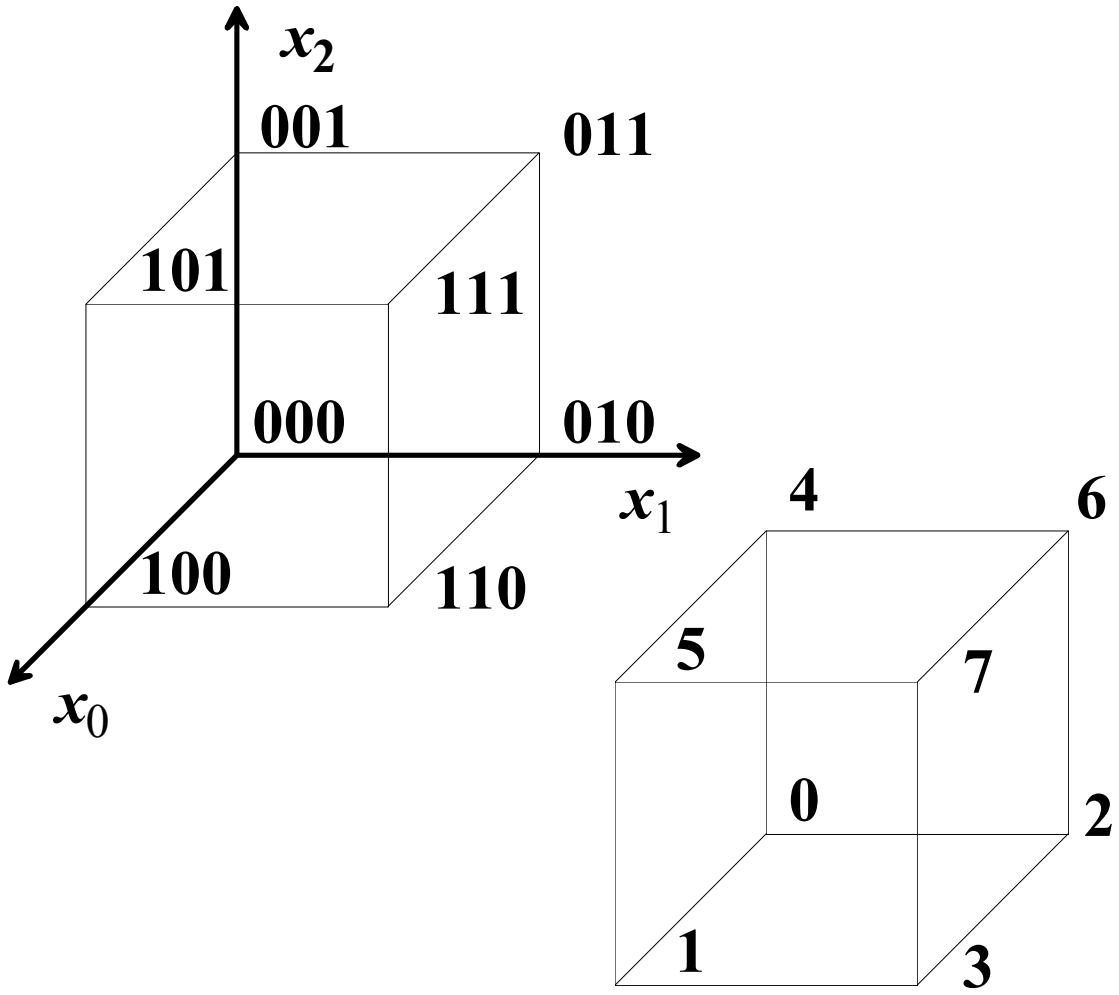
Пример

$$F_1(X_0, X_1) = \overline{F_{14}(X_0, X_1)} = F_2(\overline{X}_0, X_1) = \overline{F_{13}(\overline{X}_0, X_1)} =$$
$$= F_4(X_0, \overline{X}_1) = \overline{F_{11}(X_0, \overline{X}_1)} = F_8(\overline{X}_0, \overline{X}_1) = \overline{F_7(\overline{X}_0, \overline{X}_1)}$$

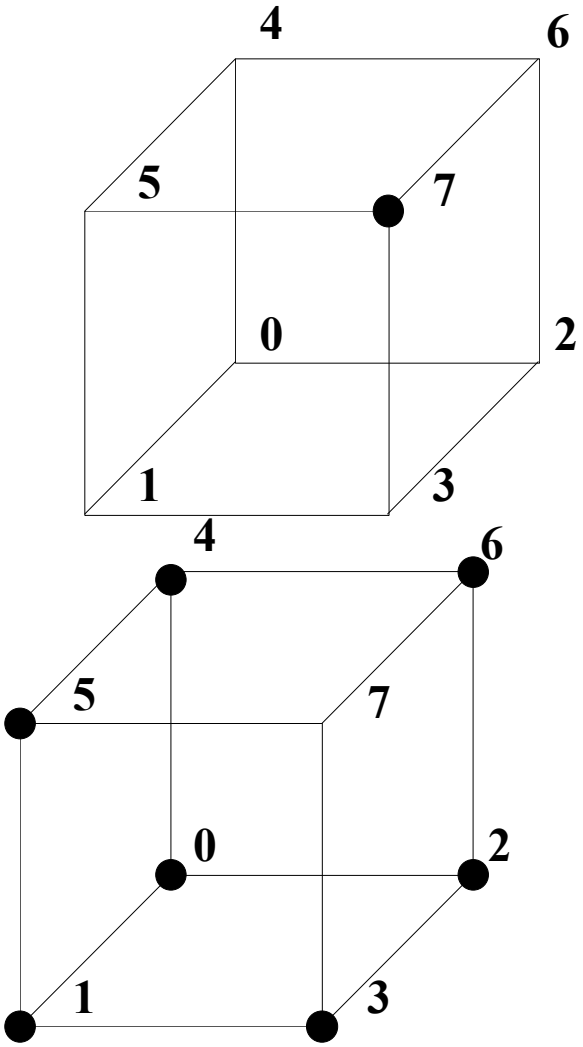


Функции двух переменных имеют
4 NPN класса эквивалентности.

NPN-классы функций трех переменных



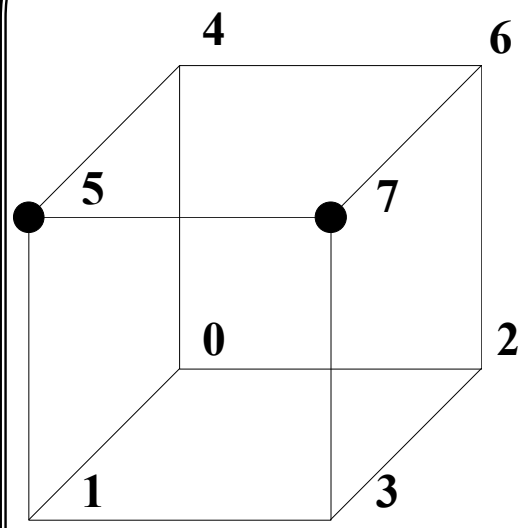
C_1 $N(C_1)=16$



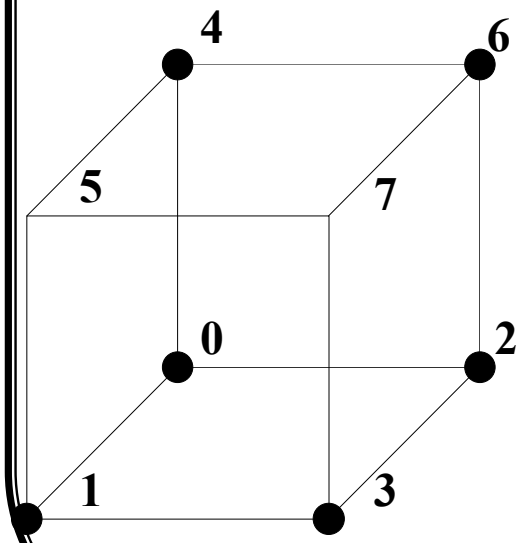
$$F(X) = X^7 = x_0 x_1 x_2$$

$$\overline{F(X)} = \overline{X^7} = \overline{x}_0 + \overline{x}_1 + \overline{x}_2$$

C_2 $N(C_2)=24$

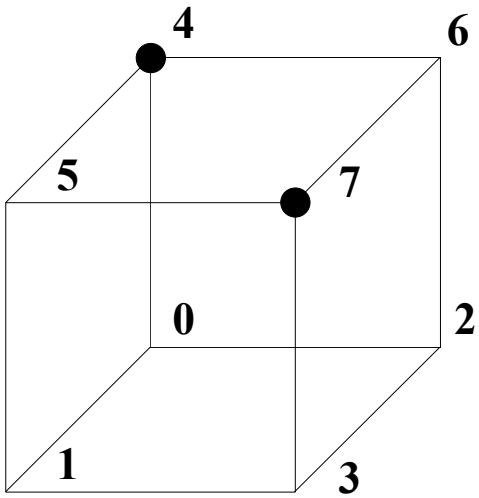


$$F(X) = X^7 + X^5 = x_0 x_1 x_2 + x_0 \overline{x}_1 x_2$$

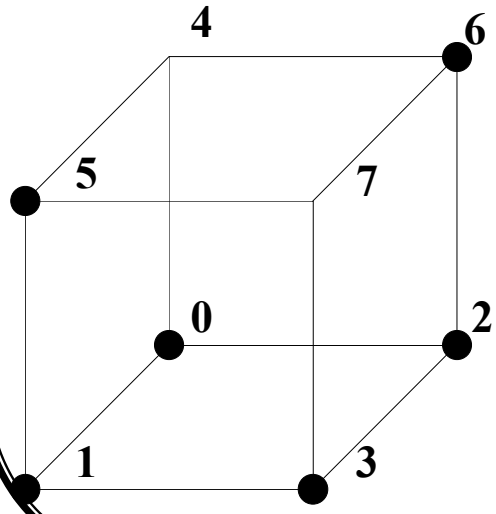


$$\overline{F(X)} = \overline{X^7} \bullet \overline{X^5} = (\overline{x}_0 + \overline{x}_1 + \overline{x}_2)(\overline{x}_0 + x_1 + \overline{x}_2)$$

C_3 $N(C_3)=24$

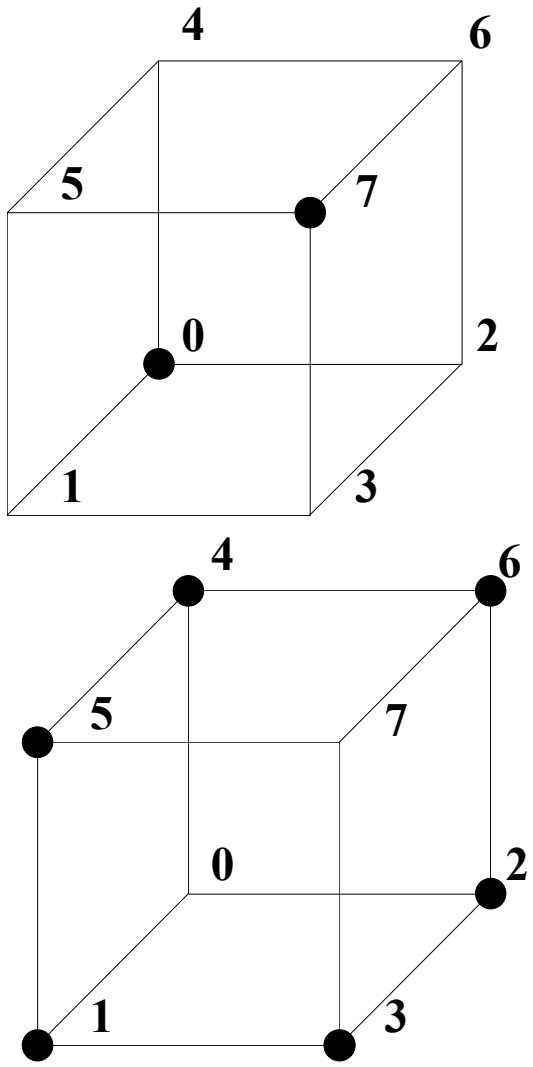


$$F(X) = X^7 + X^4 = x_0 x_1 x_2 + \overline{x}_0 \overline{x}_1 x_2$$



$$\overline{F(X)} = \overline{X^7} \bullet \overline{X^4} = (\overline{x}_0 + \overline{x}_1 + \overline{x}_2)(x_0 + x_1 + \overline{x}_2)$$

C_4 $N(C_4)=8$

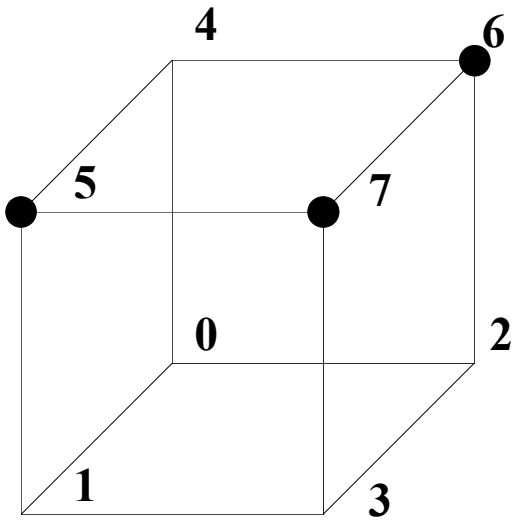


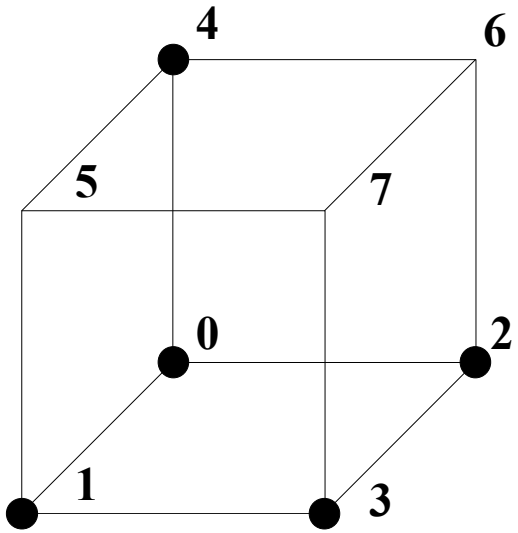
$$F(X) = X^7 + X^0 = x_0x_1x_2 + \overline{x}_0\overline{x}_1\overline{x}_2$$

$$\overline{F(X)} = \overline{X^7} \bullet \overline{X^0} =$$

$$= (\overline{x}_0 + \overline{x}_1 + \overline{x}_2)(x_0 + x_1 + x_2)$$

C_5 $N(C_5)=48$



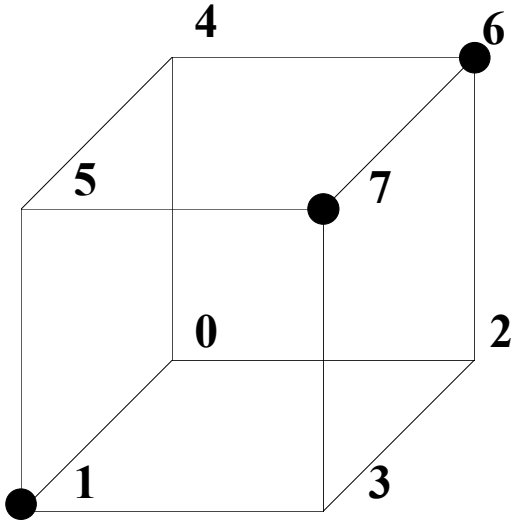


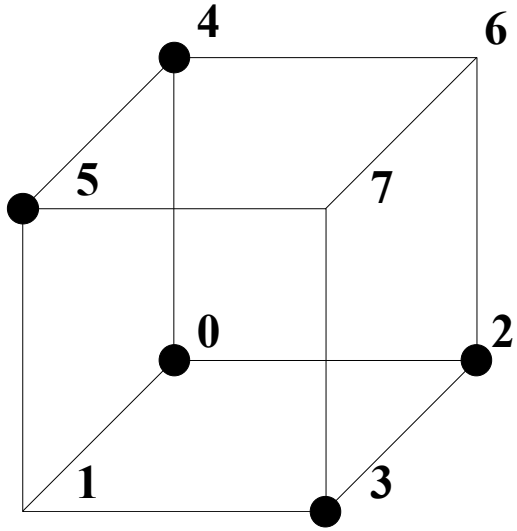
$$F(X) = X^7 + X^6 + X^5 = x_0x_1x_2 + \overline{x}_0x_1x_2 + x_0\overline{x}_1x_2$$

$$\overline{F(X)} = \overline{X^7} \bullet \overline{X^6} \bullet \overline{X^5} =$$

$$= (\overline{x}_0 + \overline{x}_1 + \overline{x}_2)(x_0 + \overline{x}_1 + \overline{x}_2)(\overline{x}_0 + x_1 + \overline{x}_2)$$

C_6 $N(C_6)=48$



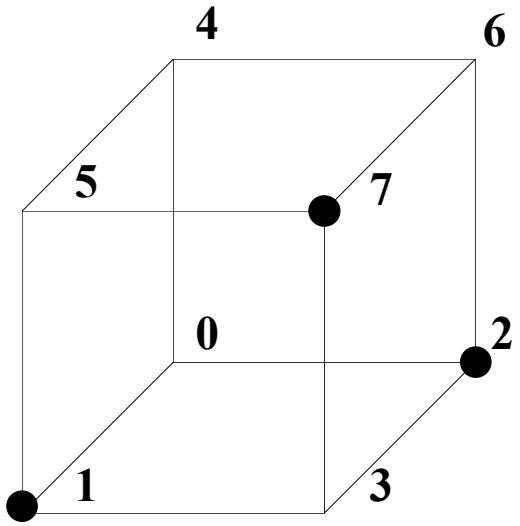


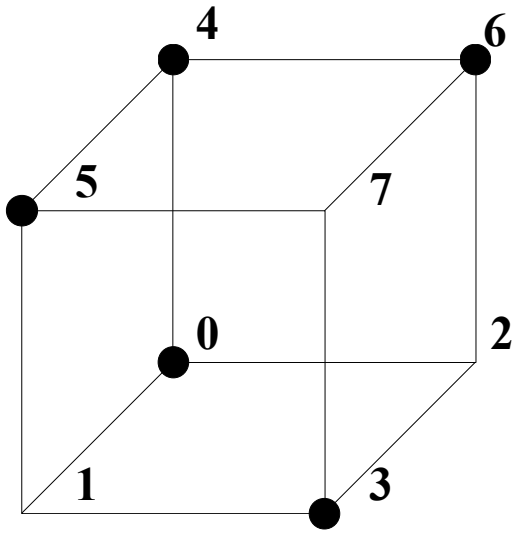
$$F(X) = X^7 + X^6 + X^1 = x_0x_1x_2 + \overline{x}_0x_1x_2 + x_0\overline{x}_1\overline{x}_2$$

$$\overline{F(X)} = \overline{X^7} \bullet \overline{X^6} \bullet \overline{X^1} =$$

$$= (\overline{x}_0 + \overline{x}_1 + \overline{x}_2)(x_0 + \overline{x}_1 + \overline{x}_2)(\overline{x}_0 + x_1 + x_2)$$

C_7 $N(C_7)=16$



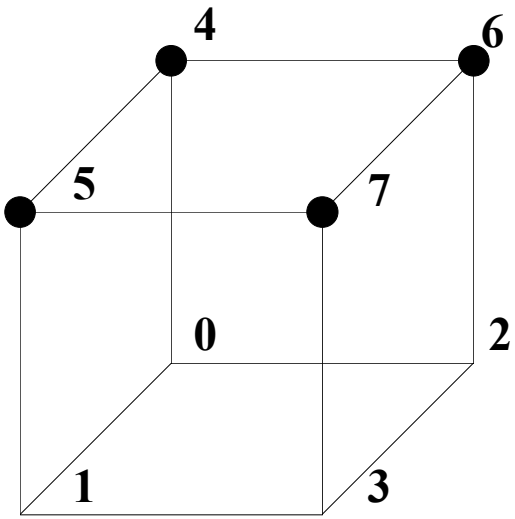


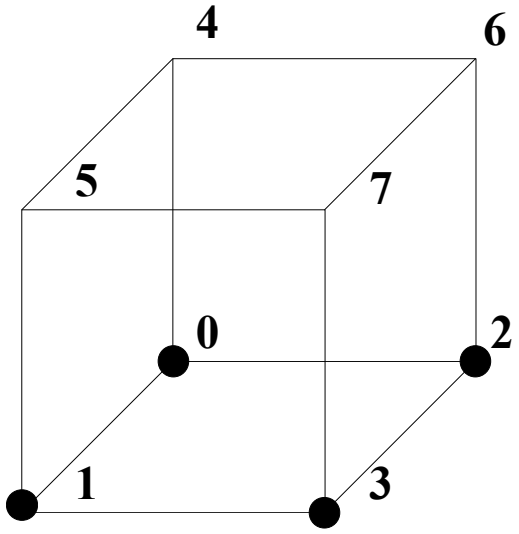
$$F(X) = X^7 + X^2 + X^1 = x_0x_1x_2 + \overline{x}_0x_1\overline{x}_2 + x_0\overline{x}_1\overline{x}_2$$

$$\overline{F(X)} = \overline{X^7} \bullet \overline{X^2} \bullet \overline{X^1} =$$

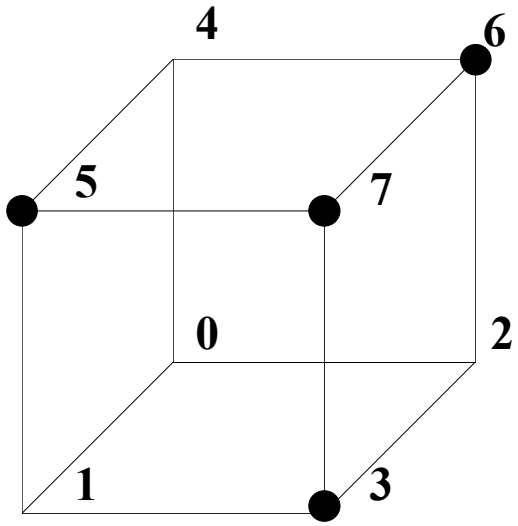
$$= (\overline{x}_0 + \overline{x}_1 + \overline{x}_2)(x_0 + \overline{x}_1 + x_2)(\overline{x}_0 + x_1 + x_2)$$

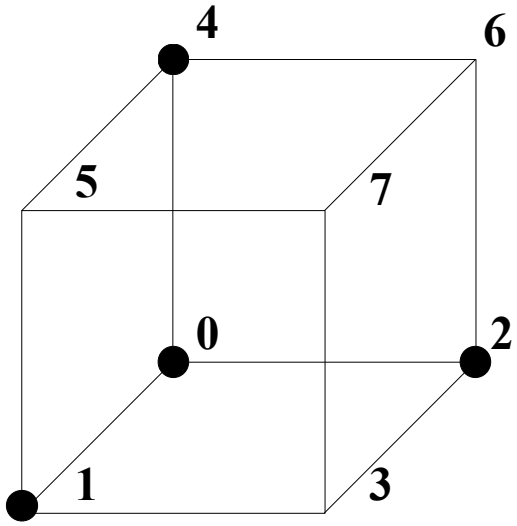
C_8 $N(C_8)=6$





C_9 $N(C_9)=8$

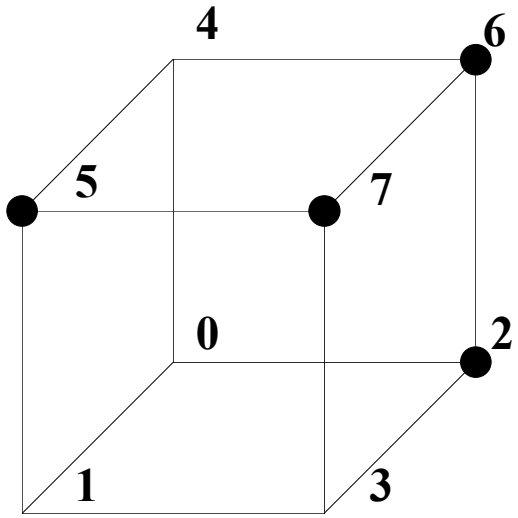


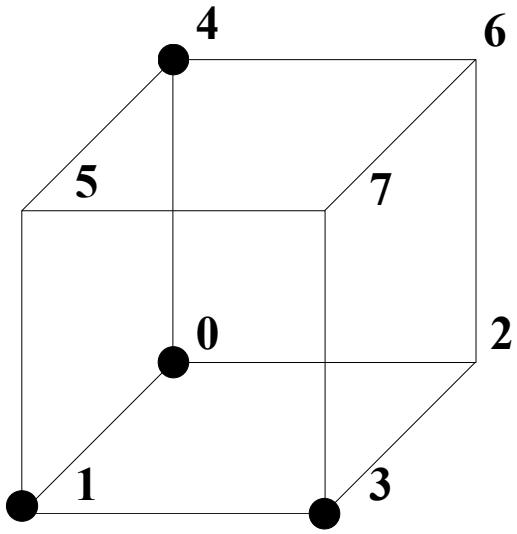


$$F(X) = X^7 + X^6 + X^5 + X^3 =$$
$$= x_0 x_1 x_2 + \overline{x}_0 x_1 x_2 + x_0 \overline{x}_1 x_2 + x_0 x_1 \overline{x}_2$$

$$\overline{F(X)} = \overline{X^7} \cdot \overline{X^6} \cdot \overline{X^5} \cdot \overline{X^3} = X^4 + X^2 + X^1 + X^0$$

C_{10} $N(C_{10})=24$



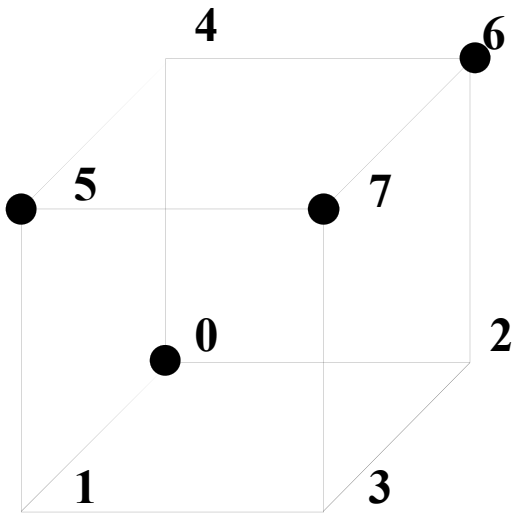


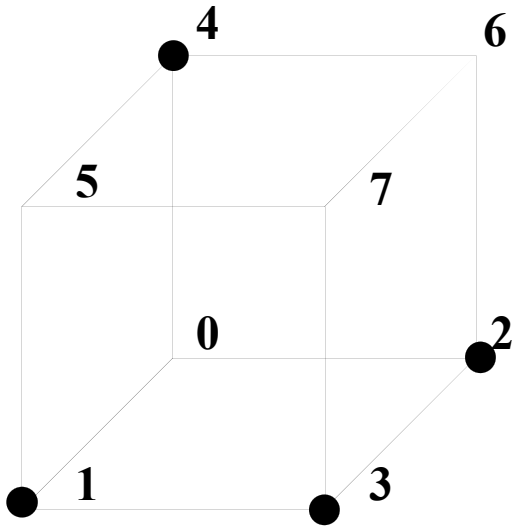
$$F(X) = X^7 + X^6 + X^5 + X^2 =$$

$$= x_0 x_1 x_2 + \overline{x}_0 x_1 x_2 + x_0 \overline{x}_1 x_2 + \overline{x}_0 x_1 \overline{x}_2$$

$$\overline{F(X)} = \overline{X^7} \bullet \overline{X^6} \bullet \overline{X^5} \bullet \overline{X^2} = X^4 + X^3 + X^1 + X^0$$

C_{11} $N(C_{11})=24$



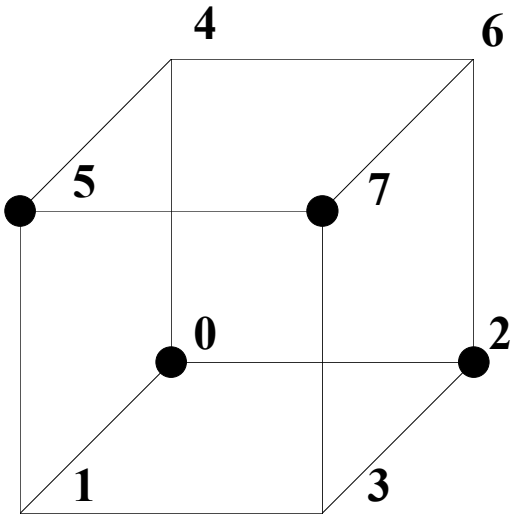


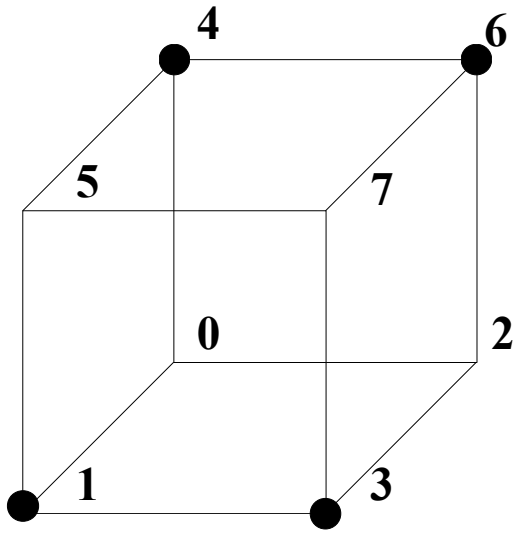
$$F(X) = X^7 + X^6 + X^5 + X^0 =$$

$$= x_0 x_1 x_2 + \overline{x}_0 x_1 x_2 + x_0 \overline{x}_1 x_2 + \overline{x}_0 \overline{x}_1 \overline{x}_2$$

$$\overline{F(X)} = \overline{X^7} \bullet \overline{X^6} \bullet \overline{X^5} \bullet \overline{X^0} = X^4 + X^3 + X^1 + X^2$$

C_{12} $N(C_{12})=6$



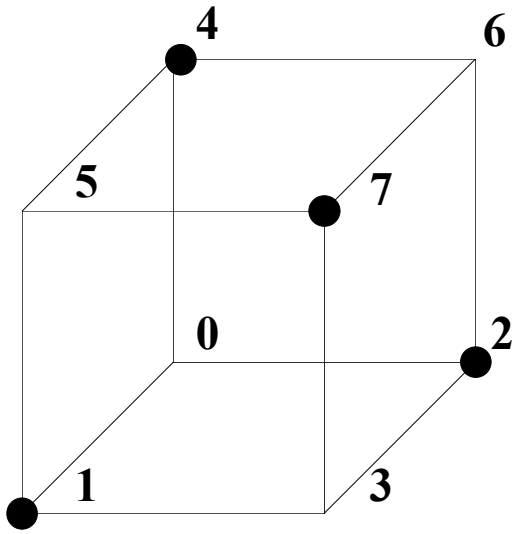


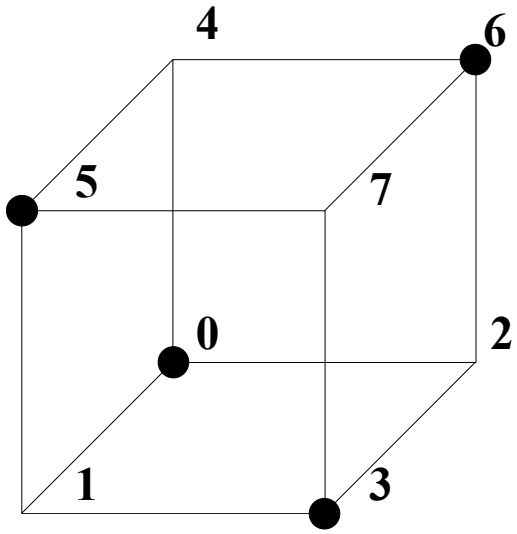
$$F(X) = X^7 + X^5 + X^2 + X^0 =$$

$$= x_0 x_1 x_2 + x_0 \overline{x}_1 x_2 + \overline{x}_0 x_1 \overline{x}_2 + \overline{x}_0 \overline{x}_1 \overline{x}_2$$

$$\overline{F(X)} = \overline{X^7} \bullet \overline{X^5} \bullet \overline{X^2} \bullet \overline{X^0} = X^4 + X^6 + X^1 + X^3$$

C_{13} $N(C_{13})=2$



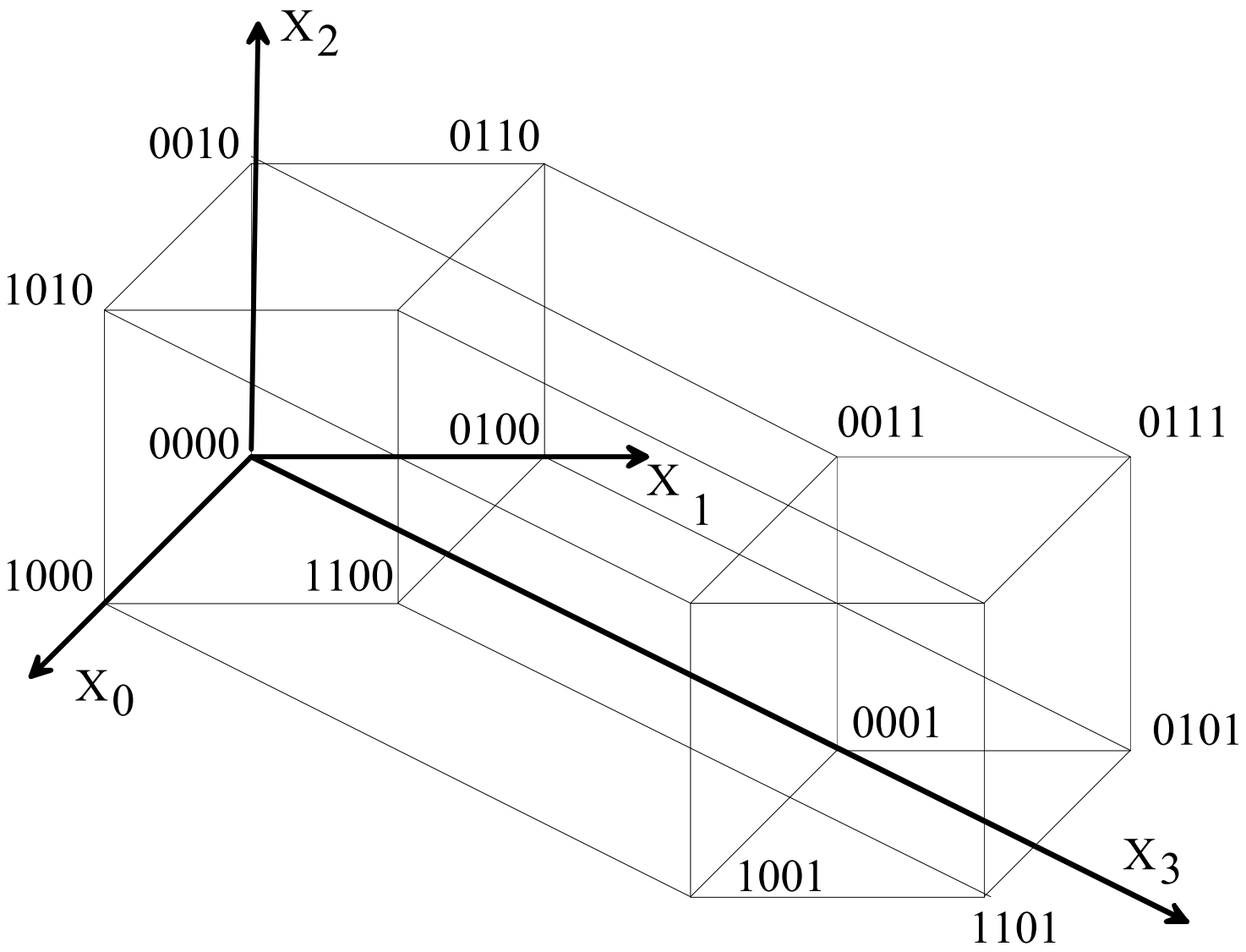


$$F(X) = X^7 + X^4 + X^2 + X^1 =$$

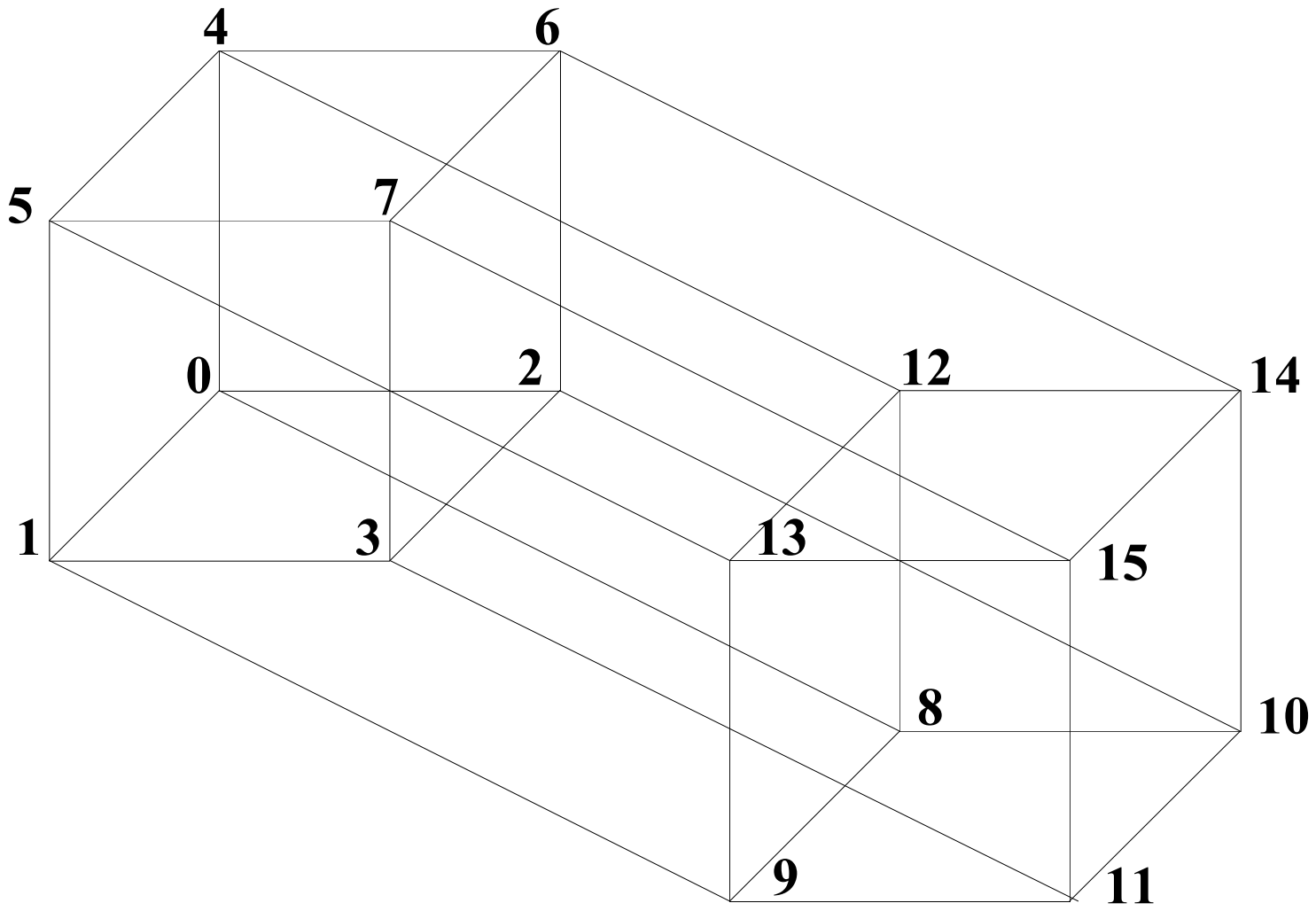
$$= x_0 x_1 x_2 + \overline{x}_0 \overline{x}_1 x_2 + \overline{x}_0 x_1 \overline{x}_2 + x_0 \overline{x}_1 \overline{x}_2$$

$$\overline{F(X)} = \overline{X^7} \bullet \overline{X^4} \bullet \overline{X^2} \bullet \overline{X^1} = X^6 + X^5 + X^3 + X^0$$

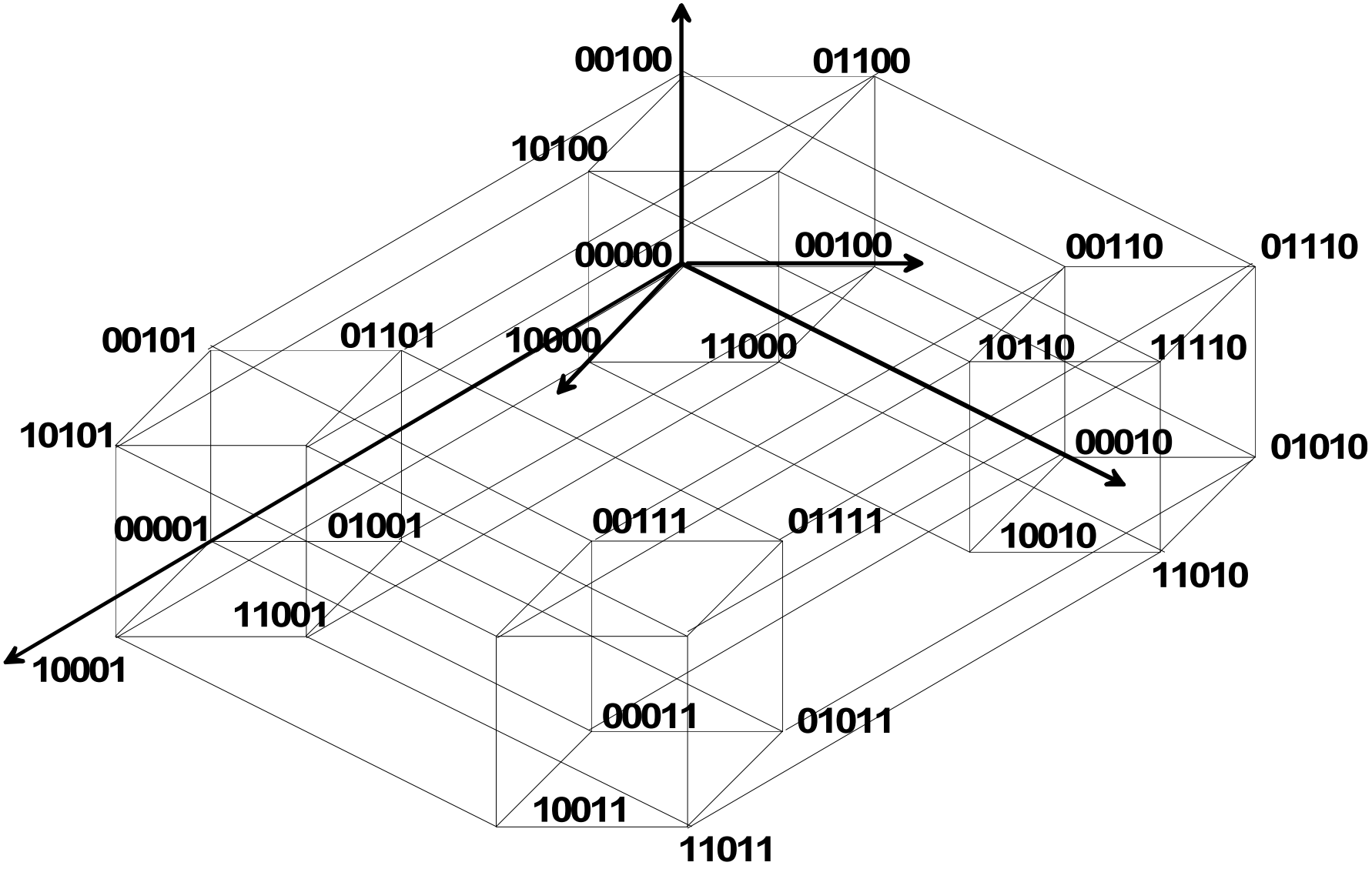
4-мерный гиперкуб



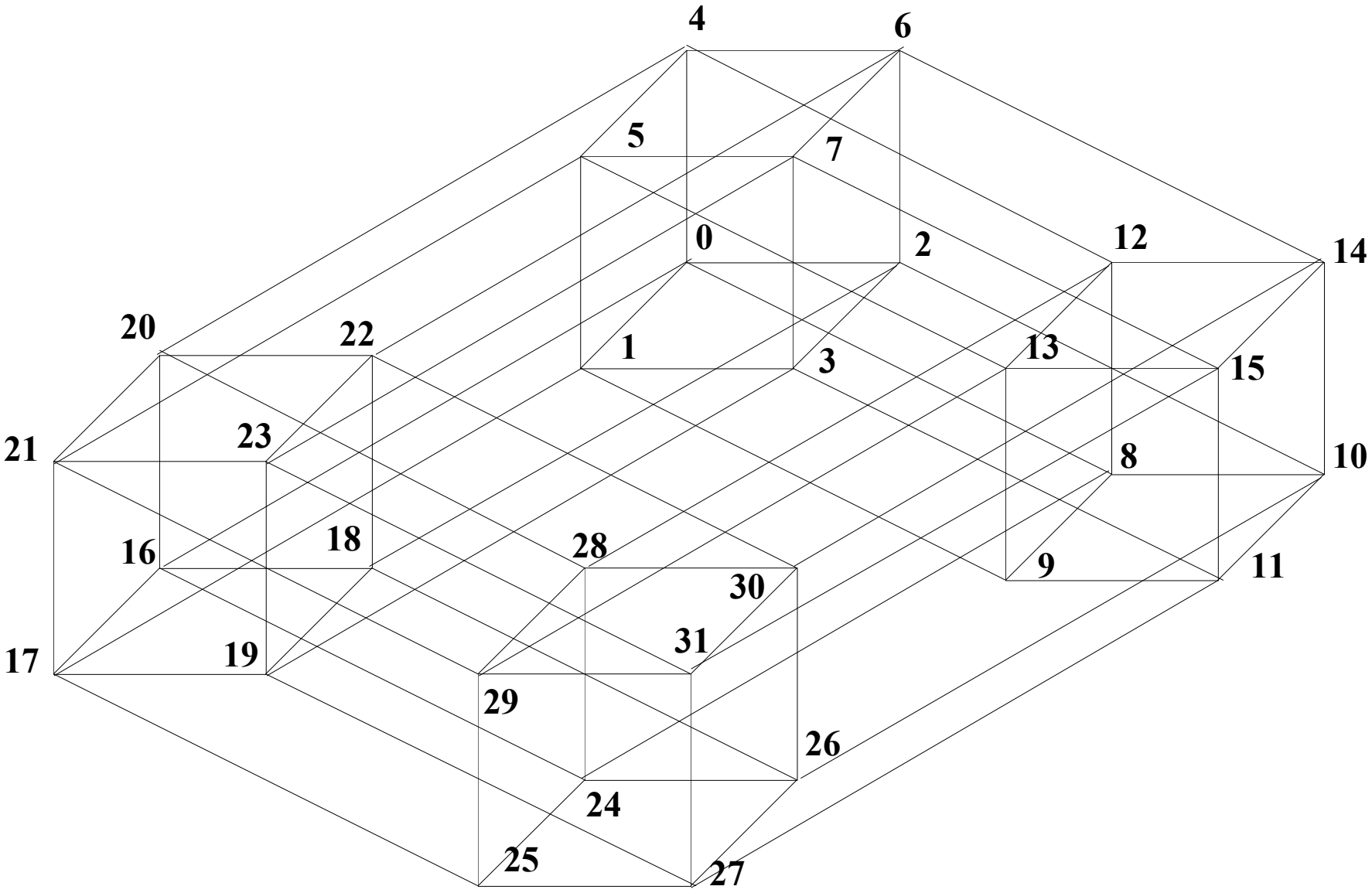
4-мерный гиперкуб



5-мерный гиперкуб



5-мерный гиперкуб



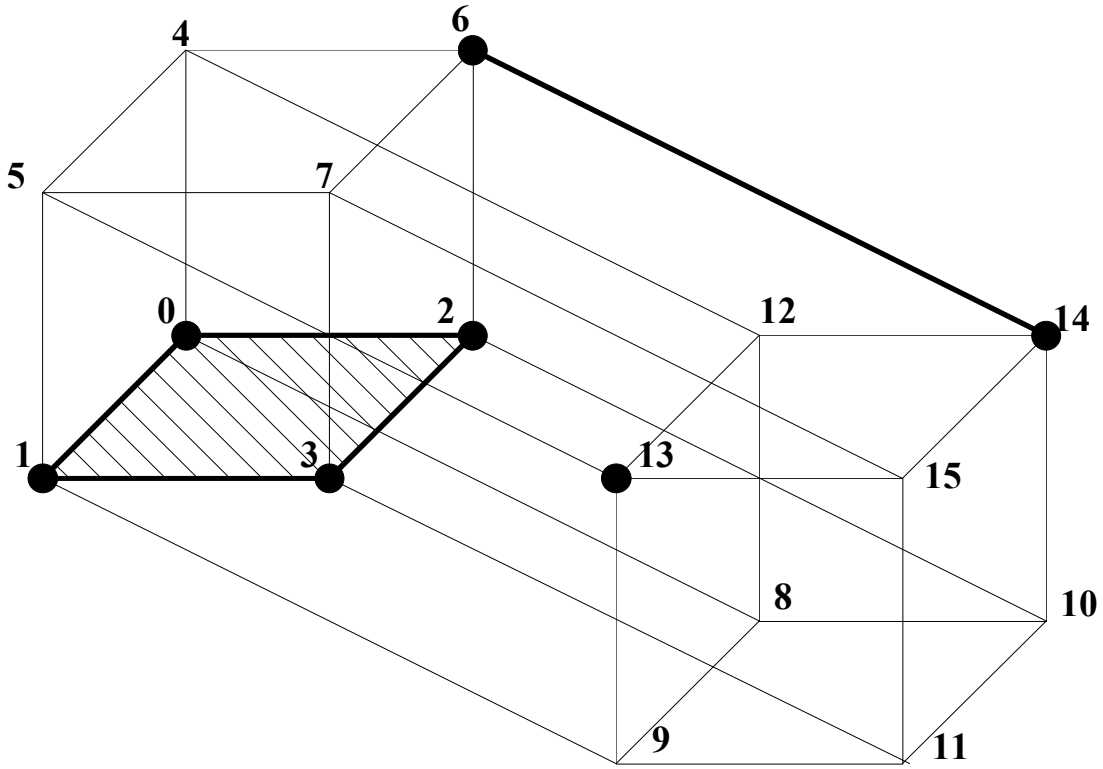
*Элемент размерности **k** **n**-гиперкуба является множеством* $\mathbf{2^k}$ *вершин, имеющих **n-k** общих координат.*

Элемент размерности **0**=> *вершина* (**2^0=1** *вершина*, ***n*** *общих координат*);

Элемент размерности **1** => **2^1=2** *вершины*, **n-1** *общих координат*;

Элемент размерности **2**=> **2^2=4** *вершины*, **n-2** *общих координат*;

Примеры

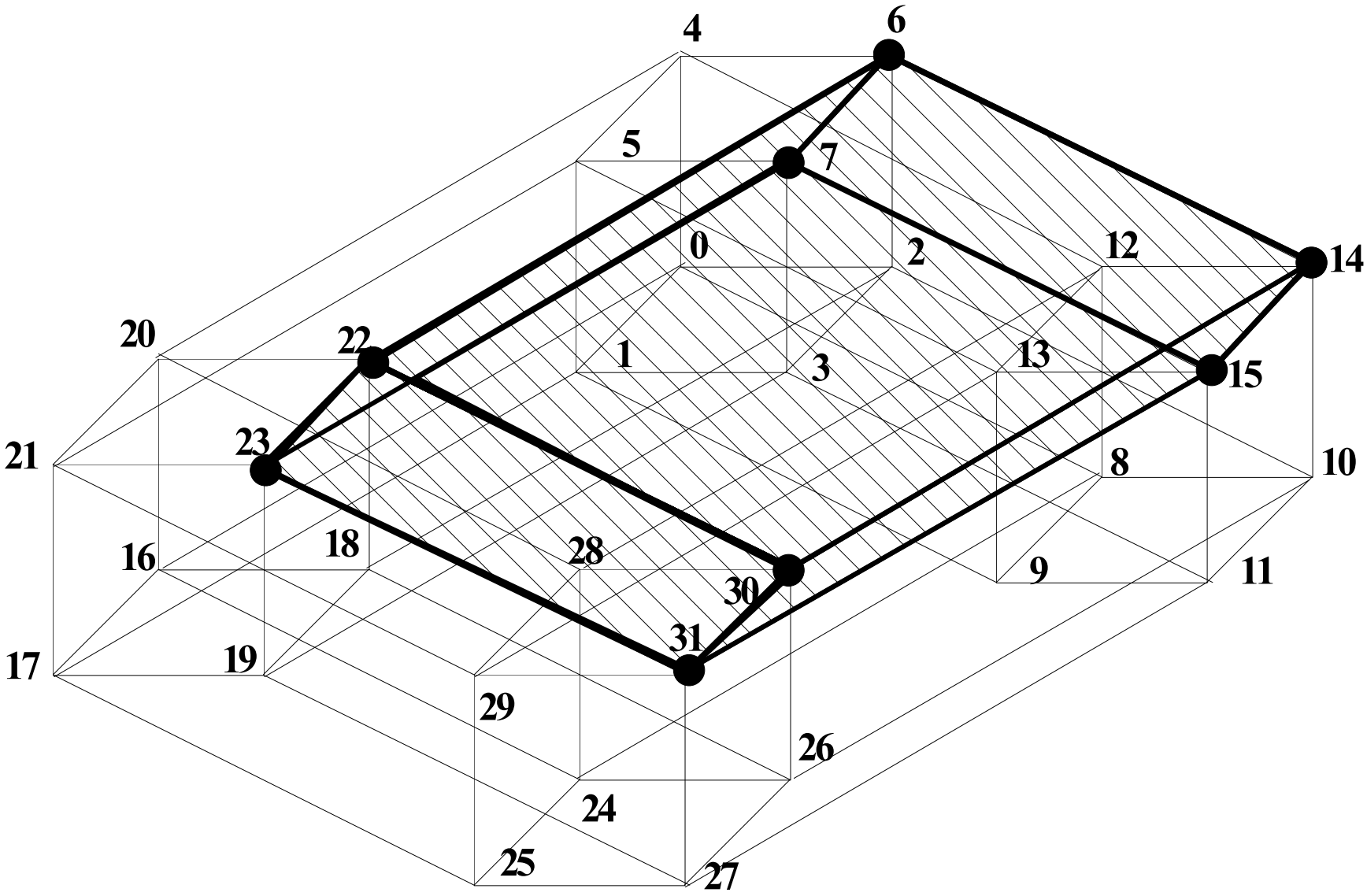


$$C^0 = \{13\} = x_0^1 x_1^0 x_2^1 x_3^1 = x_0 \overline{x}_1 x_2 x_3;$$

$$C^1 = \{6,14\} = \overline{x}_0 x_1 x_2 \overline{x}_3 + \overline{x}_0 x_1 x_2 x_3 = \overline{x}_0 x_1 x_2;$$

$$C^2 = \{0,1,2,3\} = \overline{x}_0 \overline{x}_1 \overline{x}_2 \overline{x}_3 + + x_0 \overline{x}_1 \overline{x}_2 \overline{x}_3 + \overline{x}_0 x_1 \overline{x}_2 \overline{x}_3 +$$

$$+ x_0 x_1 \overline{x}_2 \overline{x}_3 = \overline{x}_2 \overline{x}_3 (\overline{x}_0 \overline{x}_1 + x_0 \overline{x}_1 + \overline{x}_0 x_1 + x_0 x_1) = \overline{x}_2 \overline{x}_3$$

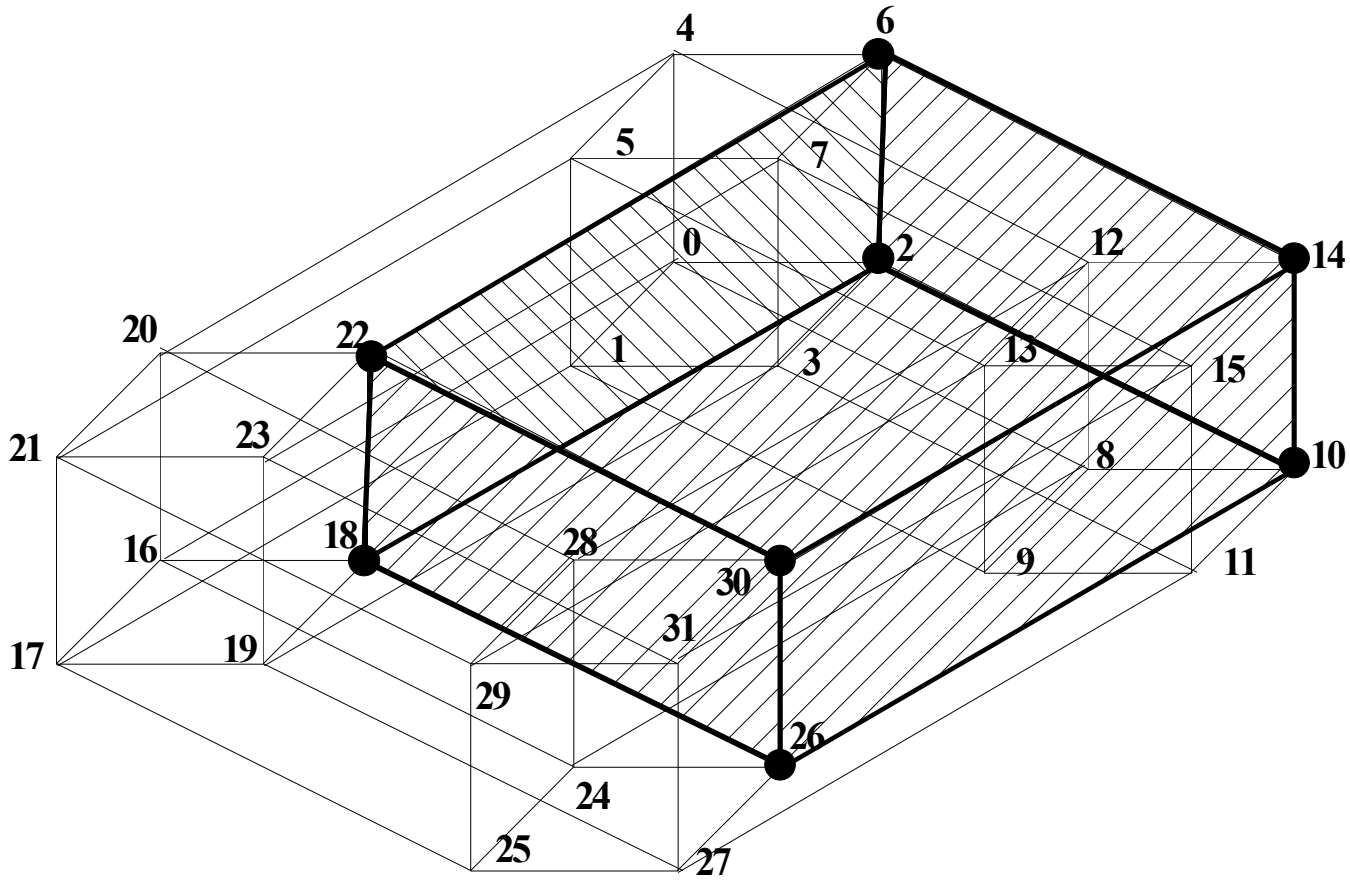

4
6
5
7
0
2
12
14
20
22
1
3
13
15
21
23
8
10
16
18
28
9
11
30
31
17
19
29
26
24
25
27

$C^3 = \{6,7,14,15,22,23,30,31\}$ => x_1=1, x_2=1, x_0=x_3=x_4=*

x_0, x_3, x_4 – внутренние переменные,
x_1, x_2 - внешние переменные.

$$C^3 = x_1^1 x_2^1 = x_1 x_2$$

$$x^{\omega} = \begin{cases} x & \text{if} \quad \omega = 1 \\ \overline{x} & \text{if} \quad \omega = 0 \\ 1 & \text{if} \quad \omega = * \end{cases} \qquad X^C = \&_{j=0}^{n-1} x_j^{c_j}$$



$$C = \{0,1,*,*,*\};$$

$$X^C = x_0^0 x_1^1 x_2^* x_3^* x_4^* = \overline{x}_0 x_1 \bullet 1 \bullet 1 \bullet 1 = \overline{x}_0 x_1$$

Разложение Шеннона

Теорема: ***Произвольная логическая функция*** $f(x_1, x_2, \ldots, x_n)$ ***может быть представлена в виде***

$$f(x_1, x_2, \ldots, x_n) = \bar{x}_1 f(0, x_2, \ldots, x_n) \vee x_1 f(1, x_2, \ldots, x_n).$$

Доказательство:

1. $x_1 = 1$

$$\overline{x_1} f(0, x_2, \ldots, x_n) \vee x_1 f(1, x_2, \ldots, x_n) = 0 f(0, x_2, \ldots, x_n) \vee 1 f(1, x_2, \ldots, x_n) =$$
$$= f(1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_n).$$

2. $x_1 = 0$

$$\overline{x_1} f(0, x_2, \ldots, x_n) \vee x_1 f(1, x_2, \ldots, x_n) = 1 f(0, x_2, \ldots, x_n) \vee 0 f(1, x_2, \ldots, x_n) =$$
$$= f(0, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_n).$$

Пример:

$$x_1 \vee x_2 x_3 = x_1 f(1, x_2, x_3) \vee \overline{x_1} f(0, x_2, x_3) = x_1 \vee \overline{x_1} x_2 x_3$$

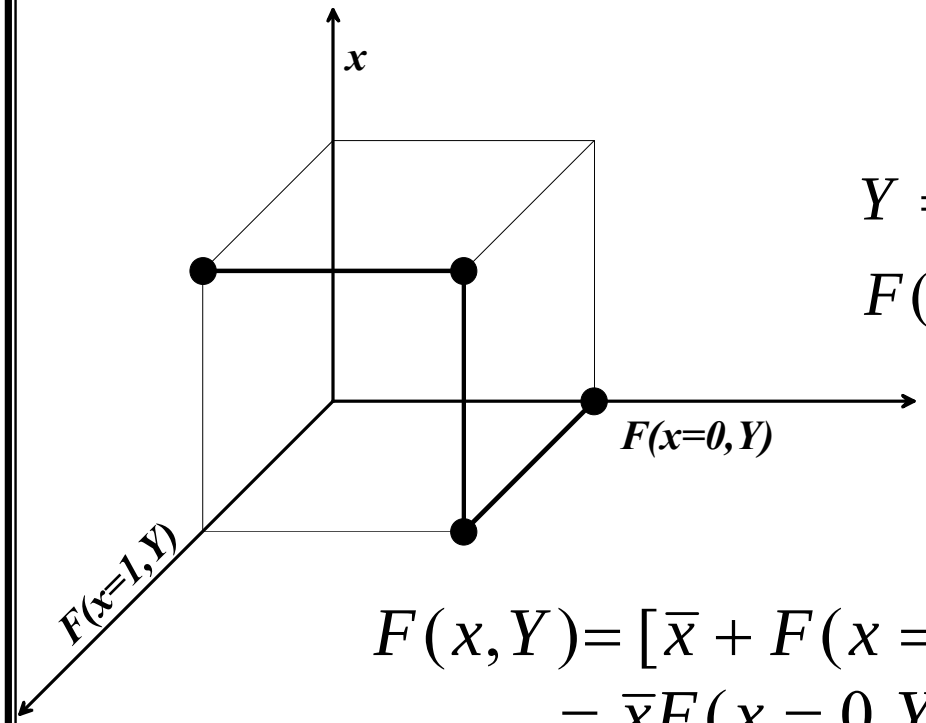
Двойственное разложение Шеннона

Теорема: ***Произвольная логическая функция*** $f(x_1,x_2,..,x_n)$ ***может быть представлена в виде***

$$f(x_1, x_2, \ldots, x_n) = (\bar{x}_1 \vee f(1, x_2, \ldots, x_n))(x_1 \vee f(0, x_2, \ldots, x_n)).$$

Пример:

$$x_1 x_2 \vee x_3 = (\bar{x}_1 \vee f(1, x_2, x_3))(x_1 \vee f(0, x_2, x_3)) = (\bar{x}_1 \vee x_2 \vee x_3)(x_1 \vee x_3).$$

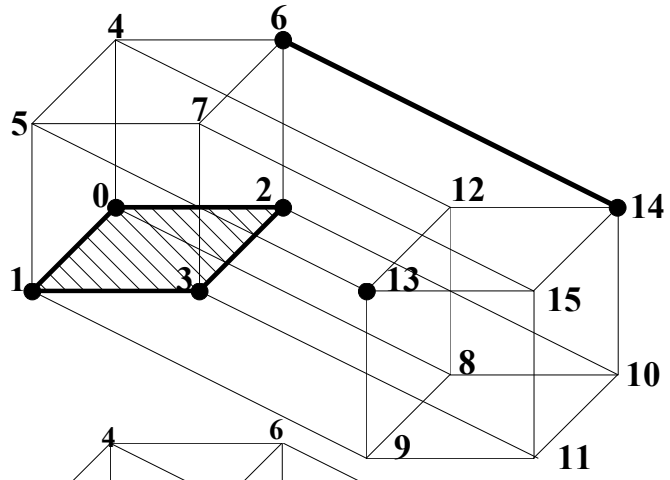


$$Y = \{y_0, y_1, \ldots, y_{n-2}\}$$

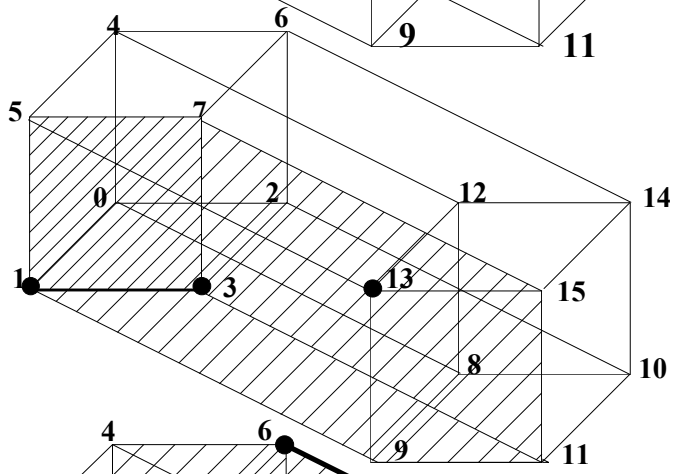
$$F(x,Y) = \bar{x}F(x=0,Y) + xF(x=1,Y)$$

$$F(x,Y) = [\bar{x} + F(x=1,Y)] \bullet [x + F(x=0,Y)] =$$
$$= \bar{x}F(x=0,Y) + xF(x=1,Y) +$$
$$+ \cancel{F(x=0,Y)F(x=1,Y)}$$

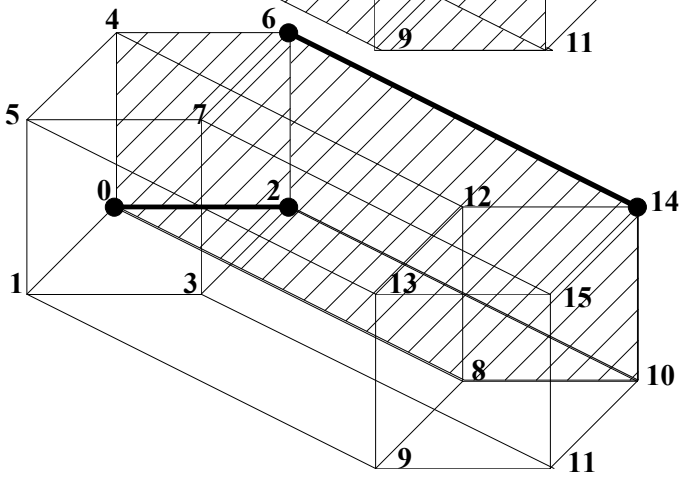
$$F(x_0, x_1, x_2, x_3) =$$



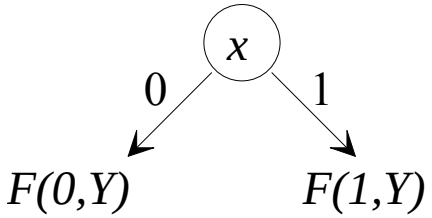
$$x_0 F(x_0 = 1, x_1, x_2, x_3) +$$

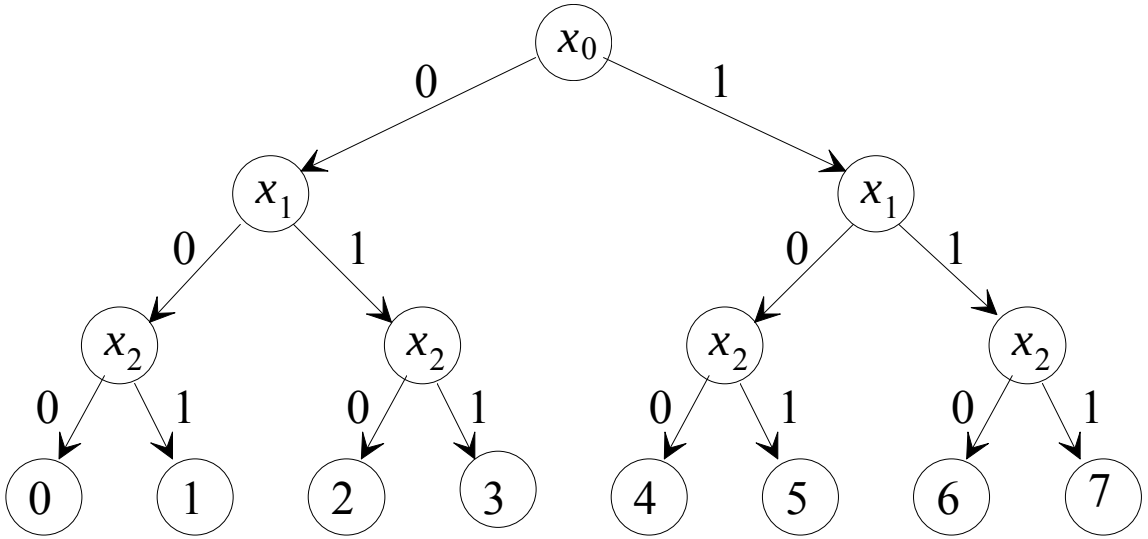


$$\overline{x}_0 F(x_0 = 0, x_1, x_2, x_3)$$



BDD (Binary Decision Diagram) – диаграмма функции, которая строится последовательным применением разложения Шеннона относительно всех входных переменных





Разложение Маллера-Рида логических функций

Для обычных аналитических функций

$$\Delta f = f(x+\Delta x) - f(x) \cong +\Delta x \frac{df(x)}{dx},$$

где производная по x определяется как

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}.$$

Для булевых функций: $\Delta x=\{0,1\}$ и $\Delta F=\{0,1\}$,

$$\Delta_x F(x,Y) = F(x \oplus \Delta x, Y) \oplus F(x,Y)$$

и производная имеет вид

$$\frac{\partial F(x,Y)}{\partial x} = \frac{\Delta_x F(x,Y)}{\Delta x} = F(\overline{x},Y) \oplus F(x,Y) =$$

$$F(0,Y) \oplus F(1,Y)$$

Теорема: ***Любая логическая функция может быть представлена в виде***

$$F(x,Y) = F(x=0,Y) \oplus x\frac{\partial F(x,Y)}{\partial x}.$$

Доказательство:

$$F(0,Y) \oplus x\frac{\partial F(x,Y)}{\partial x} =$$

$$F(0,Y) \bullet \overline{x\frac{\partial F(x,Y)}{\partial x}} + \overline{F(0,Y)} \bullet x\frac{\partial F(x,Y)}{\partial x} =$$

$$F(0,Y)[\overline{x} + \overline{\frac{\partial F(x,Y)}{\partial x}}] + \overline{F(0,Y)} \bullet x\frac{\partial F(x,Y)}{\partial x} =$$

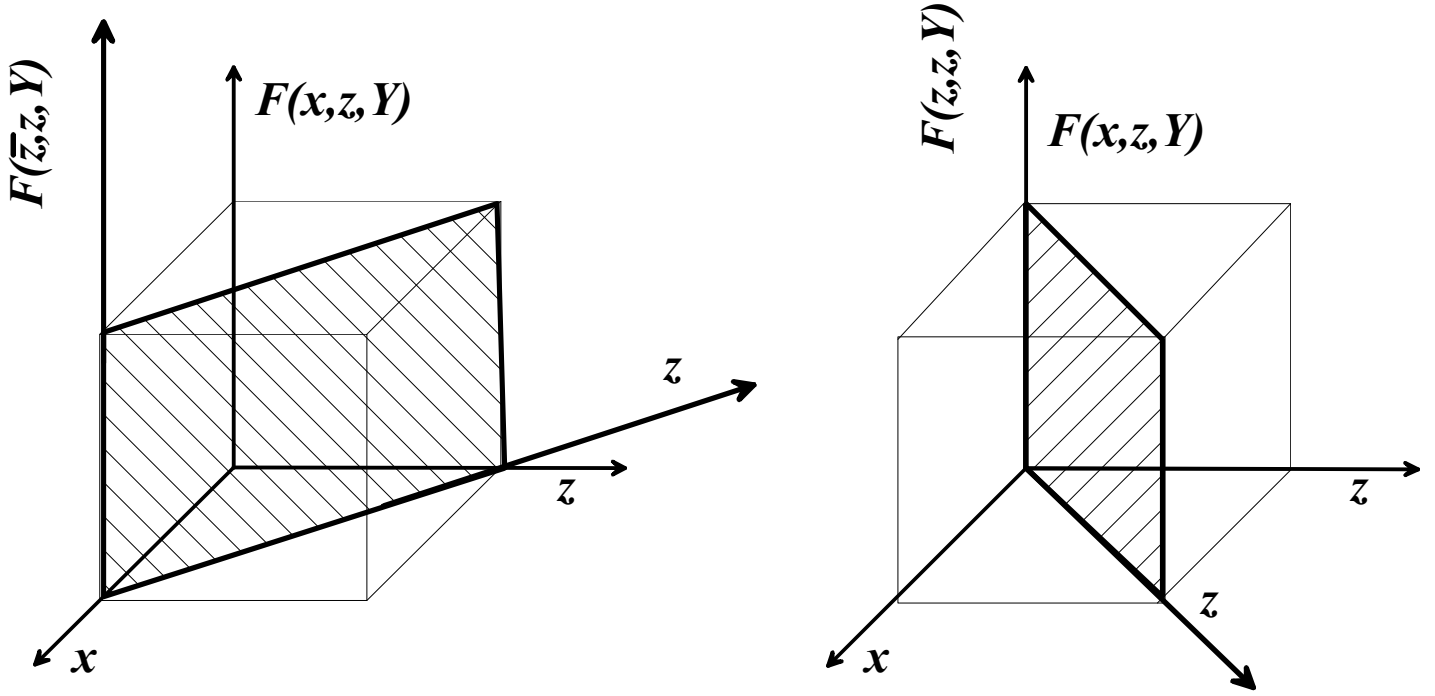
$$\overline{x}F(0,Y) + F(0,Y)\overline{\frac{\partial F(x,Y)}{\partial x}} + \overline{F(0,Y)} \bullet x\frac{\partial F(x,Y)}{\partial x} =$$

$$\overline{x}F(0,Y) + x[F(0,Y)\overline{\frac{\partial F(x,Y)}{\partial x}} + \overline{F(0,Y)}\frac{\partial F(x,Y)}{\partial x}] =$$

$$\overline{x}F(0,Y) + x[F(0,Y) \oplus \frac{\partial F(x,Y)}{\partial x}] = \overline{x}F(0,Y) + xF(1,Y)$$

$$F(0,Y) \oplus \frac{\partial F(x,Y)}{\partial x} = F(0,Y) \oplus F(1,Y) \oplus F(0,Y) = F(1,Y)$$

Диагональная декомпозиция (Варшавского)



$$F(x,z,Y) = (x \oplus z)F(\bar{z},z,Y) + \overline{(x \oplus z)}F(z,z,Y)$$



Логическое Проектирование

Лекция 3

Минимизация Булевых функций

Критерии минимизации

1. ***Критерии минимизации схем:***
   - *1.1. Занимаемая площадь кристалла;*
   - *1.2. Скорость;*
   - *1.3. Потребляемая мощность.*
2. ***Критерии минимизации функций:***
   - *2.1 Количество операций с их весами;*
   - *2.2 Количество вхождений переменных в формулу (количество литералов);*
   - *2.3 Суммарное число входов логических элементов схемы, реализующей функцию;*
   - *2.4 Число транзисторов, необходимых для реализации функции.*

Основные определения

1. ***Функция***:

Два множества вершин: ***T и F***

$$F(\Omega \in T) = 1$$

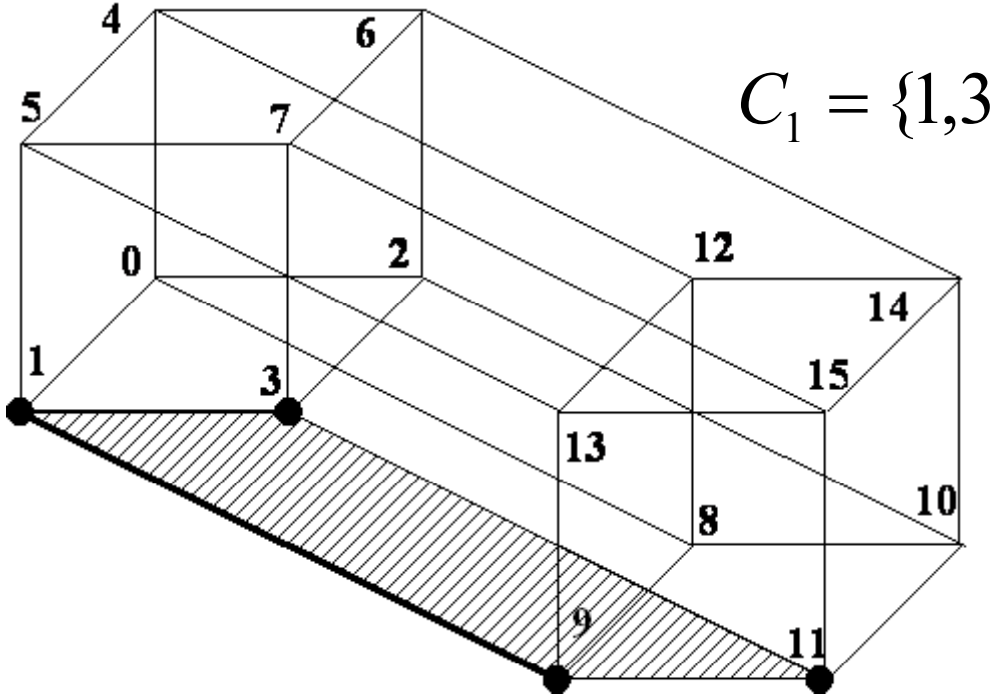
$$F(\Omega \in F) = 0$$

2. ***Элемент функции (терм, импликант)***:

Подмножество 2^k *вершин* $\in T$ *с n-k общими координатами.*

3. ***Простой элемент (простой импликант):***

Элемент функции, который не является подмножеством никакого другого элемента функции.



$$C_1 = \{1,3,9,11\}; C_2 = \{1,9\}; C_2 \in C_1$$

C_1 – простой элемент;
C_2 – непростой элемент.

C_1={1,5}; 5=1010

Обозначение элемента – два числа Ω и B.

Ω – множество координат любой вершины, принадлежащей элементу;

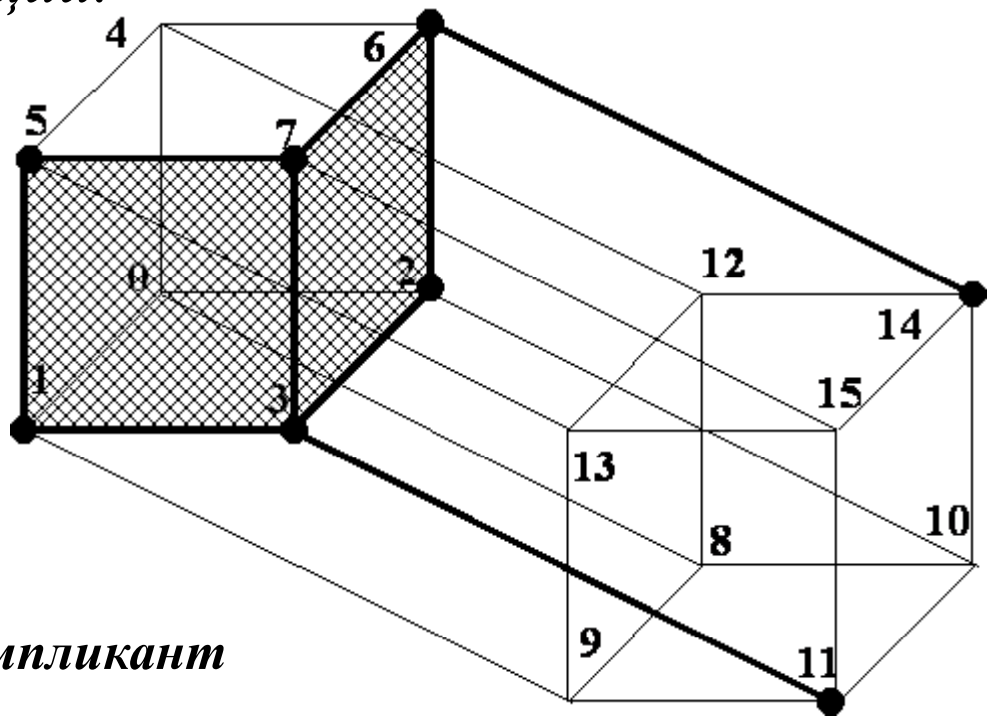
B – код, указывающий внешние координаты.

$$(x_0^{\omega_0} + \bar{b}_0)(x_1^{\omega_1} + \bar{b}_1)(x_2^{\omega_2} + \bar{b}_2)(x_3^{\omega_3} + \bar{b}_3) =$$

$$(x_0^1 + 0)(x_1^0 + 1)(x_2^0 + 0)(x_3^0 + 1) = x_0\bar{x}_2$$

4. Покрытие функции:

Множество простых элементов, покрывающих все единичные вершины функции.



C_1={3,9};C_2={3,10};
C_3={3,7};C_4={6,7}

Простой элемент <=> Простой импликант

$$x_0\overline{x}_3 + x_1\overline{x}_3 + x_0x_1\overline{x}_2 + \overline{x}_0x_1x_2$$

Сокращенная форма булевой функции содержит все простые импликанты.

Простейший способ получения сокращенной формы:

Пример: **T={1,2,3,5,6,7,14,11}**

F={0,4,8,9,10,12,13,15}

1. ***Строим совершенную нормальную конъюнктивную фому функции:***

$$F(X) = (\bar{x}_0^0 + \bar{x}_1^0 + \bar{x}_2^0 + \bar{x}_3^0) \bullet (\bar{x}_0^0 + \bar{x}_1^0 + \bar{x}_2^1 + \bar{x}_3^0) \bullet$$
$$\bullet (\bar{x}_0^0 + \bar{x}_1^0 + \bar{x}_2^0 + \bar{x}_3^1) \bullet (\bar{x}_0^1 + \bar{x}_1^0 + \bar{x}_2^0 + \bar{x}_3^1) \bullet$$
$$\bullet (\bar{x}_0^0 + \bar{x}_1^1 + \bar{x}_2^0 + \bar{x}_3^1) \bullet (\bar{x}_0^0 + \bar{x}_1^0 + \bar{x}_2^1 + \bar{x}_3^1) \bullet$$
$$\bullet (\bar{x}_0^1 + \bar{x}_1^0 + \bar{x}_2^1 + \bar{x}_3^1) \bullet (\bar{x}_0^1 + \bar{x}_1^1 + \bar{x}_2^1 + \bar{x}_3^1) =$$
$$(x_0 + x_1 + x_2 + x_3)(x_0 + x_1 + \bar{x}_2 + x_3) \bullet$$
$$(x_0 + x_1 + x_2 + \bar{x}_3)(\bar{x}_0 + x_1 + x_2 + \bar{x}_3) \bullet$$
$$(x_0 + \bar{x}_1 + x_2 + \bar{x}_3)(x_0 + x_1 + \bar{x}_2 + \bar{x}_3) \bullet$$
$$(\bar{x}_0 + x_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_0 + \bar{x}_1 + \bar{x}_2 + \bar{x}_3) =$$

2. Открываем скобки и удаляем элементарные поглощения

$$(x_0 + x_1 + x_3)(x_1 + x_2 + \bar{x}_3) \bullet$$

$$(x_0 + \bar{x}_1\bar{x}_2 + x_1x_2 + \bar{x}_3)(\bar{x}_0 + \bar{x}_2 + \bar{x}_3) =$$

$$(x_0x_2 + x_0\bar{x}_3 + x_1 + x_2x_3) \bullet$$

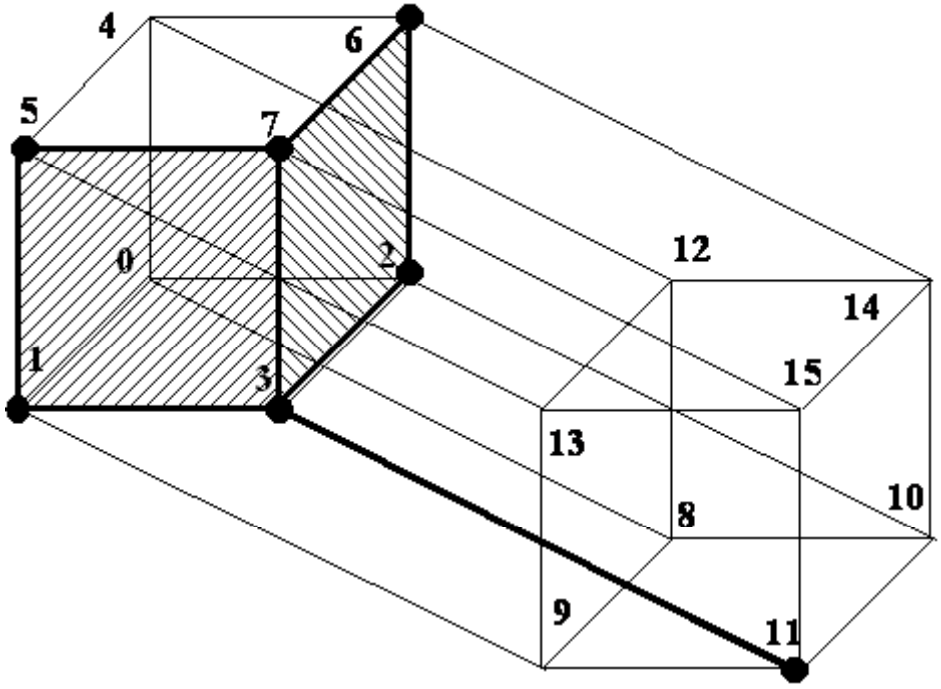
$$(x_0\bar{x}_2 + \bar{x}_1\bar{x}_2 + \bar{x}_3 + \bar{x}_0x_1x_2) =$$

$$x_0\bar{x}_3 + x_1\bar{x}_3 + \bar{x}_0x_1x_2 + x_0x_1\bar{x}_2.$$

Это метод ***Парецкого-Блейка***

5. Звезда из элементов:

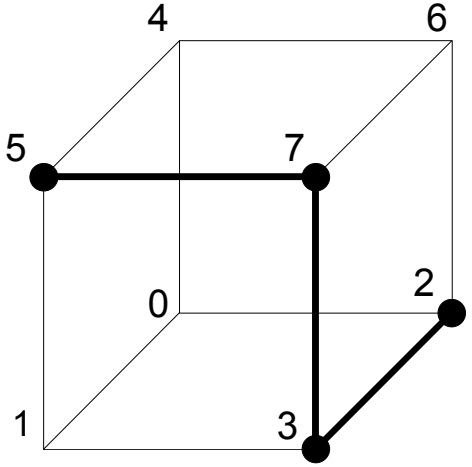
Звезда – это множество элементов,имеющих общую вершину.



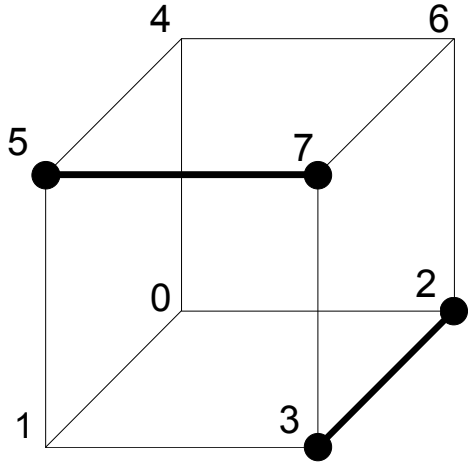
6. Существенная звезда, или существенный элемент

Звезда из одного элемента, который нельзя выбросить из покрытия.

7. Неизбыточное покрытие:

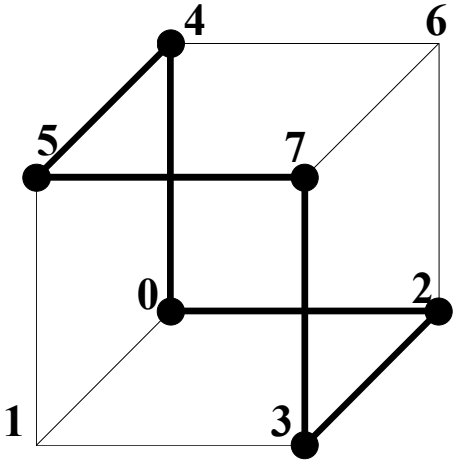


$$x_0x_2 + x_0x_1 + x_1\overline{x}_2$$

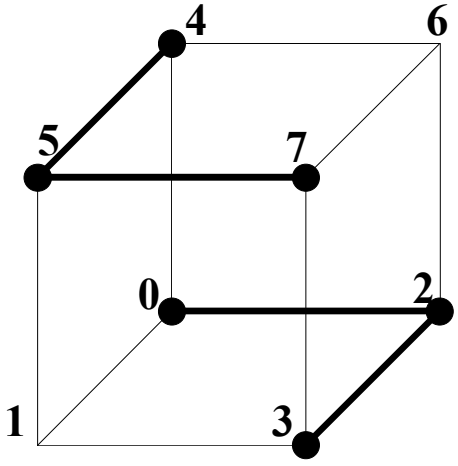


$$x_0x_2 + x_1\overline{x}_2$$

8. Тупиковое покрытие:

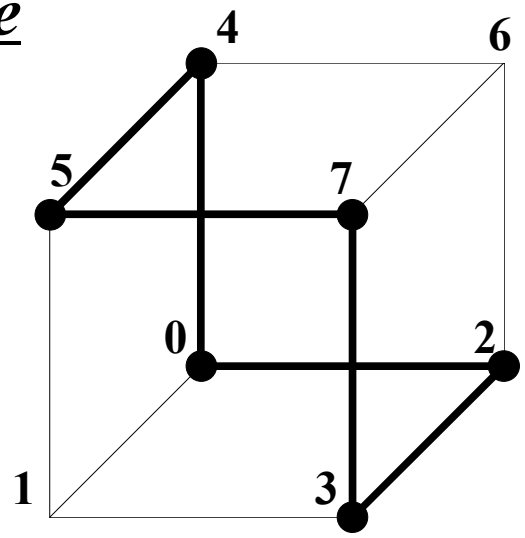


$$\overline{x}_0\overline{x}_1 + \overline{x}_1x_2 + x_0x_2 + x_0x_1 + x_1\overline{x}_2 + \overline{x}_0\overline{x}_2$$

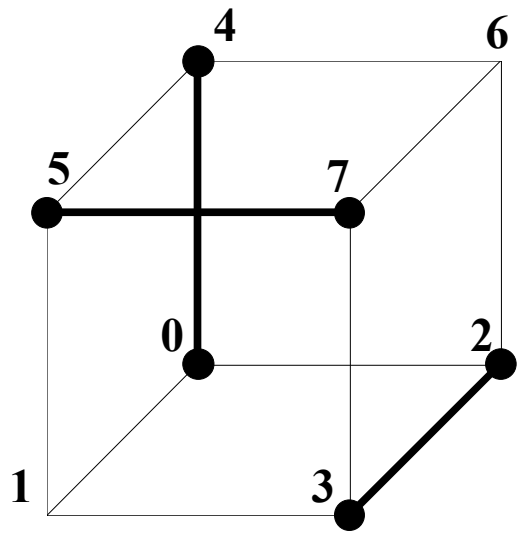


$$\overline{x}_1x_2 + x_0x_2 + x_1\overline{x}_2 + \overline{x}_0\overline{x}_2$$

9. Минимальное покрытие:



$$\bar{x}_0\bar{x}_1 + \bar{x}_1x_2 + x_0x_2 + x_0x_1 + x_1\bar{x}_2 + \bar{x}_0\bar{x}_2$$



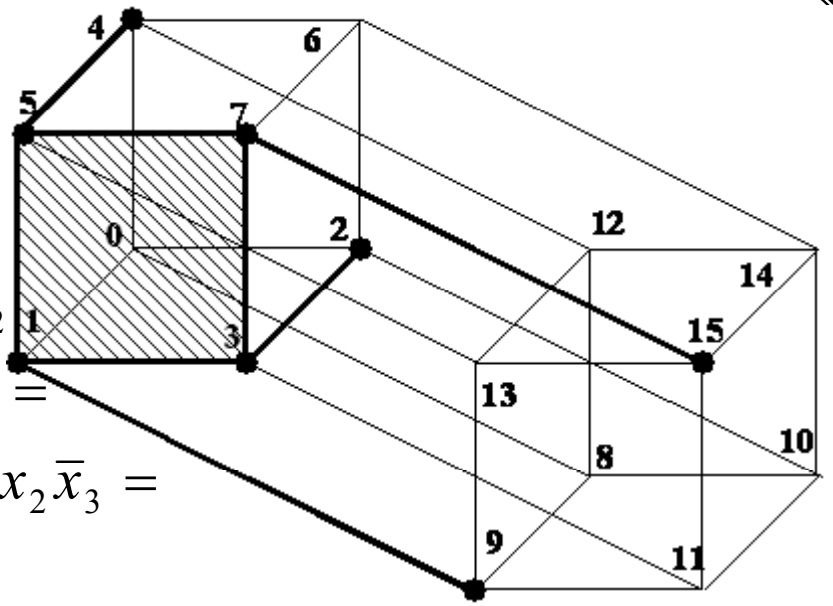
$$\bar{x}_0\bar{x}_1 + x_0x_2 + x_1\bar{x}_2$$

10. Неэлементарное поглощение:

Множество элементов $\Theta = \{C_1,.....,C_k\}$ *поглощает элемент E, если*

$$E \cap \Theta = E \cap \bigcup_{j=1}^{k} C_j = E$$

Пример:

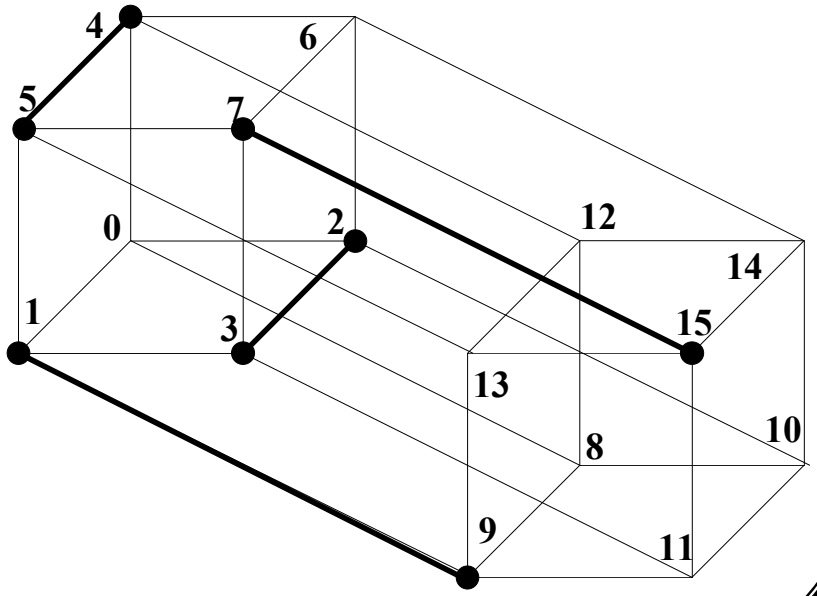


$$x_0\bar{x}_3 + x_0\bar{x}_1\bar{x}_2 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3 + x_0x_1x_2 =$$

$$x_0\bar{x}_3(x_0\bar{x}_1\bar{x}_2 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3 + x_0x_1x_2) =$$

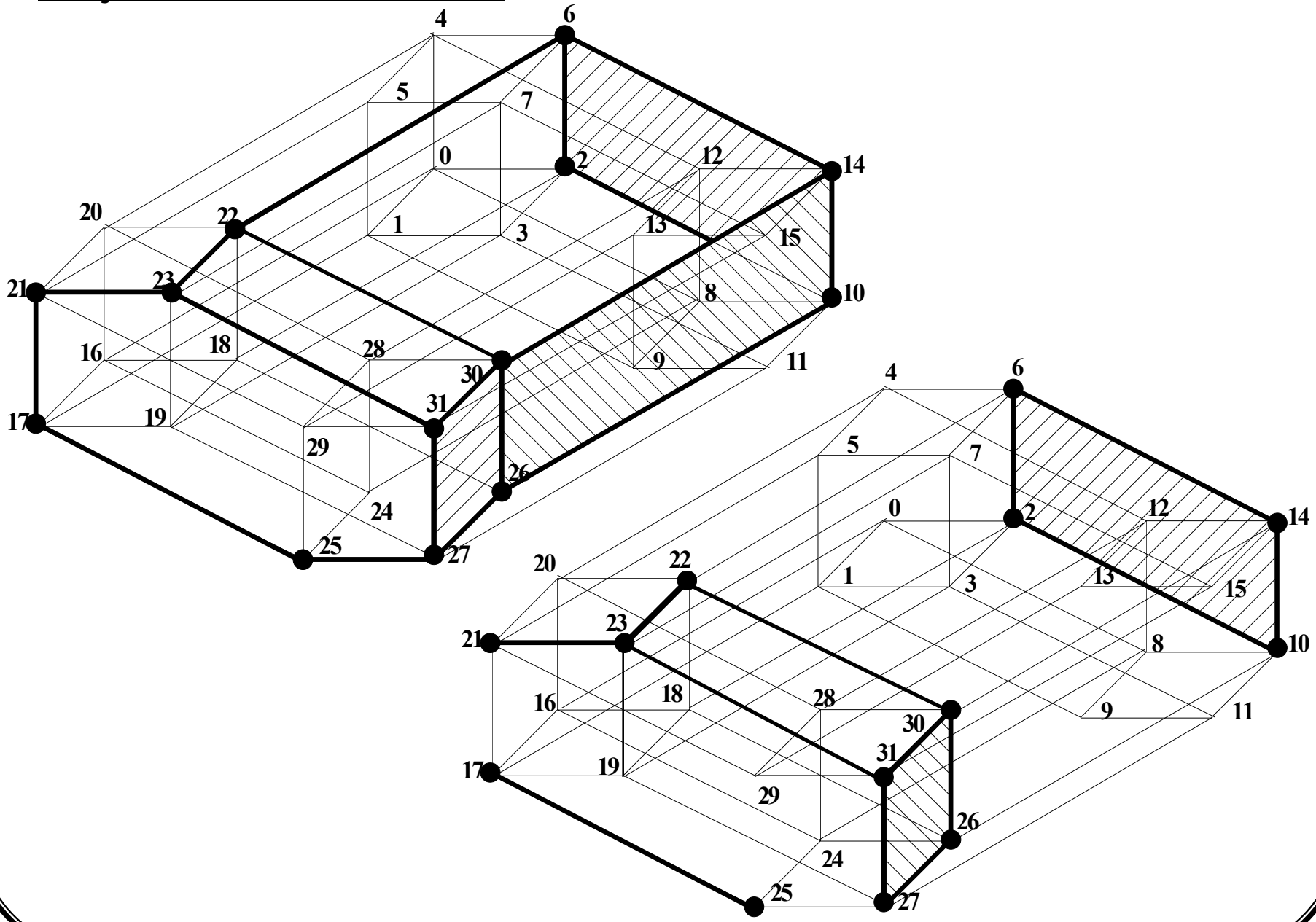
$$= x_0\bar{x}_1\bar{x}_2\bar{x}_3 + x_0\bar{x}_1x_2\bar{x}_3 + x_0x_1\bar{x}_2\bar{x}_3 + x_0x_1x_2\bar{x}_3 =$$

$$= x_0\bar{x}_3(\bar{x}_1\bar{x}_2 + \bar{x}_1x_2 + x_1\bar{x}_2 + x_1x_2) = x_0\bar{x}_3$$



$$x_0\bar{x}_1\bar{x}_2 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3 + x_0x_1x_2$$

Визуальная минимизация:

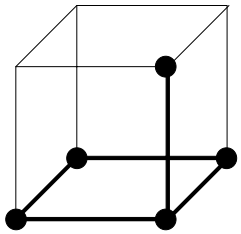


Некоторые основные определения

Пусть $A = \{\Omega_0, \Omega_1, \Omega_2, ..., \Omega_{2^n}\}$ – множество вершин n-мерного гиперкуба ($|A|=2^n$) и A_i - подмножество A ($A_i \subset A$). Тогда $F(A_i)=1$ and $F(A \setminus A_i)=0$ являются *характеристическими функциями* множества A_i. Они определяют принадлежит ли какая-либо вершина множеству A_i или не принадлежит.

Характеристическая функция вершины $F(\Omega_j)$ соответствует *конституенте единицы*. Пример: $F(0101)=\overline{x}_0 x_1 \overline{x}_2 x_3$.

Элемент размерности k – является подмножеством из 2^k вершин, Имеющих одинаковые значения *n-k общих координат.*
Элемент размерности 0 – вершина.
Элемент размерности 1 – ребро.
Элемент размерности 2 – грань..
Элемент размерности *k* - подкуб.



Каждый элемент размерности *k* соответствует *конъюнкции ранга n-k* (*импликант*) .

Оболочка функции – это множество вершин, на которых функция равна 1, и множество всех элементов, образуемых этими вершинами.

Простым элементом функции называется элемент, который не является частью любого другого элемента. Простой элемент представляется *простой импликантой*.

Множество простых элементов функции образует *сокращенную оболочку*, которая может быть представлена *сокращенной дизъюнктивной формой* булевой функции.

Неизбыточной оболочкой называется оболочка, удаление из которой хотя бы одного простого элемента изменяет функцию.

Существенным элементом называется такой элемент, который содержит хотя бы одну вершину, не принадлежащую никакому другому элементу. Некоторые несущественные элементы могут быть удалены из сокращенной оболочки (но не из неизбыточной оболочки).

Все существенные элементы принадлежат неизбыточной оболочке и не могут быть удалены.

Все неизбыточные оболочки порождают множество *тупиковых форм*.

Минимальная оболочка (*форма*) является неизбыточной оболочкой (тупиковой формой) минимальной сложности. Поиск минимальных форм является *универсальной комбинаторной проблемой*.

Вы уже знаете метод Блейка-Порецкого получения сокращенных форм булевых функций и метод их визуальной минимизации.

Теперь рассмотрим возможность вычисления неизбыточных оболочек с помощью арифметических операций с номерами вершин гиперкуба.

Арифметические операции с номерами вершин

Пусть $N(\Omega)$ – десятичный номер вершины and $N_j(\Omega)$ – номер соседней вершины по координате j, тогда

$$N_j(\Omega) = N(\Omega) + (-1)^{\omega_j} 2^j$$

$$N(\Omega) = \sum_{i=0}^{n-1} \omega_i 2^i$$

Example: Ω= 001100;

$N(\Omega) = N(001100) = 12;$

$N_0(\Omega) = N(101100) = 12 + (-1)^0 2^0 = 13$

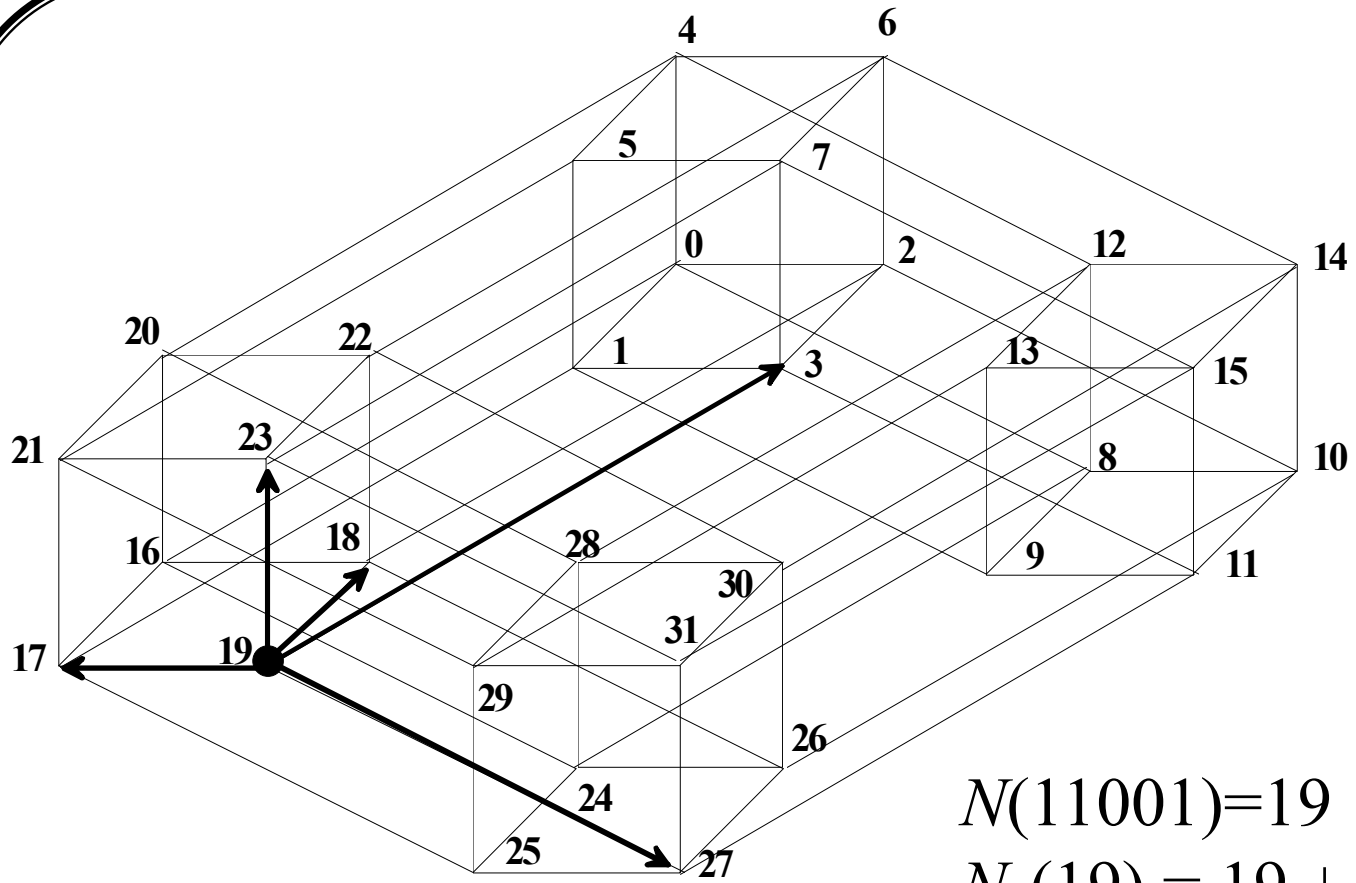
$N_1(\Omega) = N(011100) = 12 + (-1)^0 2^1 = 14$

$N_2(\Omega) = N(000100) = 12 + (-1)^1 2^2 = 8$

$N_3(\Omega) = N(001000) = 12 + (-1)^1 2^3 = 4$

$N_4(\Omega) = N(001110) = 12 + (-1)^0 2^4 = 28$

$N_5(\Omega) = N(001101) = 12 + (-1)^0 2^5 = 44$



$$N(11001)=19$$

$$N_0(19) = 19 + (-1)^1 2^0 = 18$$

$$N_1(19) = 19 + (-1)^1 2^1 = 17$$

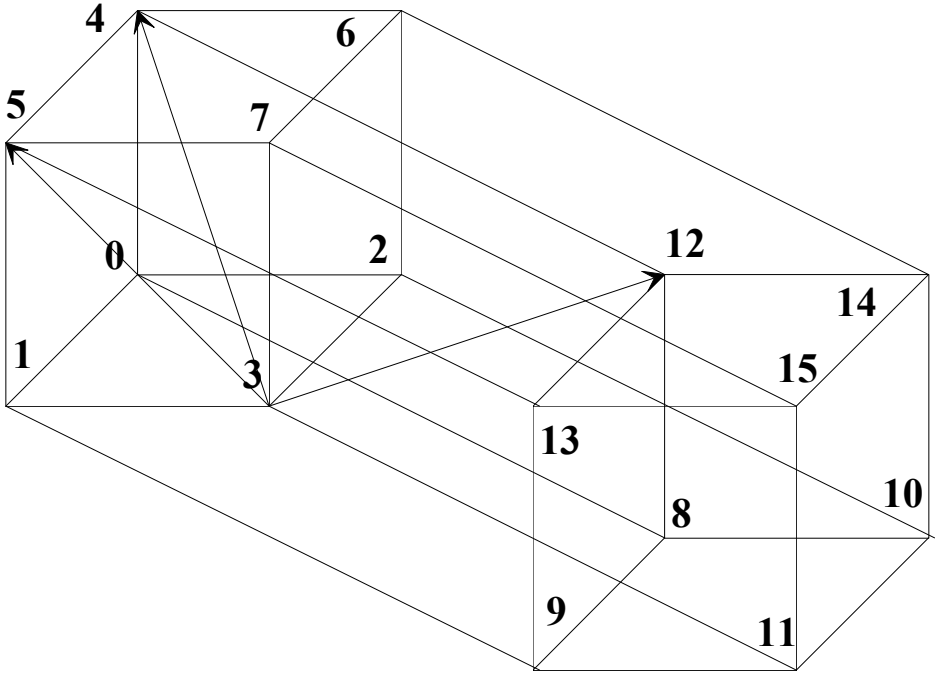
$$N_2(19) = 19 + (-1)^0 2^2 = 23$$

$$N_3(19) = 19 + (-1)^0 2^3 = 27$$

$$N_4(19) = 19 + (-1)^1 2^4 = 3$$

Вычисление номеров диагональных соседей

$N_{j_1, j_2, \ldots, j_k}(\Omega)$ – ***номер вершины, противоположной Ω по диагонали в k-мерном гиперкубе с внутренними переменными $j_1, j_2, \ldots, j_k$.***



$N_{1,2}(3)=5$; $N_{0,1,2}(3)=4$;
$N_{0,1,2,3}(3)=12$.

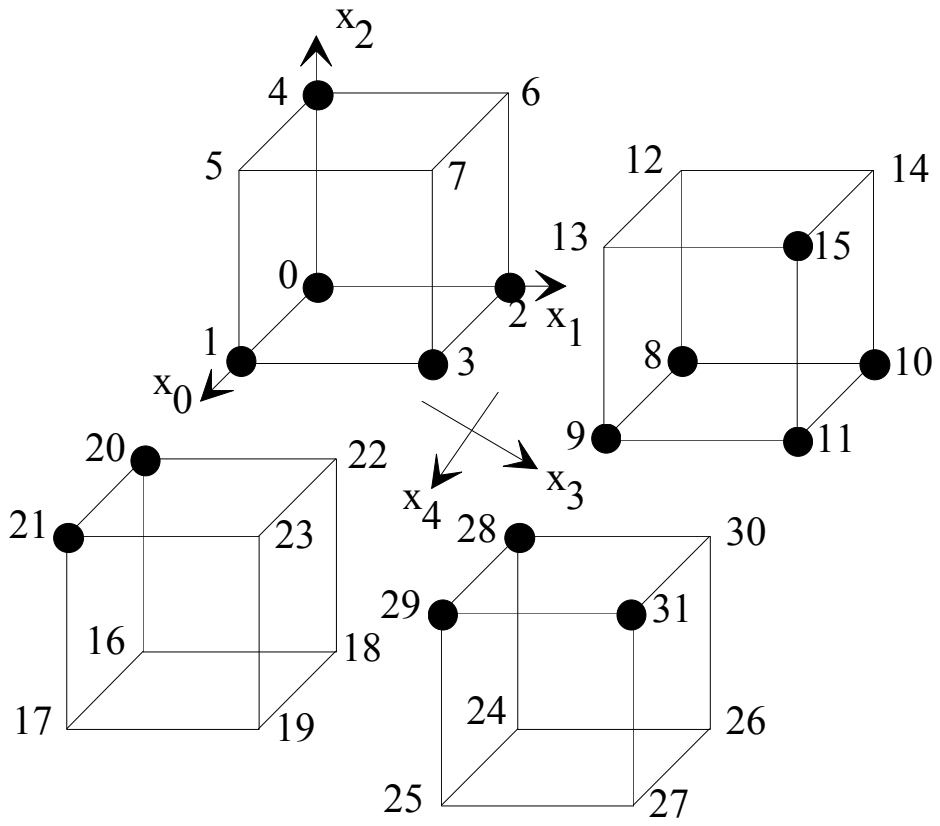
$$N_{j_1, j_2, \ldots, j_k}(\Omega) = \sum_{i=1}^{k} N_{j_i}(\Omega) - (k-1)N(\Omega)$$

Численная процедура вычисления простых элементов булевых функций (Urbana & Muller)

Рассмотрим пример:

Пусть функция задана на вершинах:

T={0, 1, 2, 3, 4, 8, 9, 10, 11, 15, 20, 21, 28, 31}.



Основными арифмерическими операциями с номерами вершин являются:

1. $N_j(\Omega)$ – номер вершины, являющейся соседней вершине $N(\Omega)$ по координате j.

$$N_j(\Omega) = N(\Omega) + (-1)^{\omega_j} 2^j$$

2. $N_{j_1,j_2,\ldots,j_k}(\Omega)$ – номер вершины, которая может быть достигнута из вершины $N(\Omega)$ изменением значений координат $j_1, j_2, \ldots, j_k$.

$$N_{j_1,j_2,\ldots,j_k}(\Omega) = \sum_{i=1}^{k} N_{j_i}(\Omega) - (k-1)N(\Omega)$$

Построим таблицу:

i	0	1	2	3	4		Вершины		
w_i	1	2	4	8	16	T	0	15	20
	0	0	0	0	0	0	~~0~~		
						1	0		
						2	1		
						3	0,1		
						4	[2]		[4]
						8	3		
						9	0,3		
						10	1,3		
						11	[0,1,3]	[2]	
	1	1	1	1	0	15		~~2~~	
	0	0	1	0	1	20			~~4~~
						21			0
						28			3
						29			[0,3]
						31		[4]	

Выбираем вершину $N(\Omega)=0$:

$N_0=1$

$N_1=2$

$N_2=4$

$N_3=8$

~~$N_4=16$~~

$N_{0,1}=N_0+N_1=1+2=3$

~~$N_{0,2}=N_0+N_2=1+4=5$~~

$N_{0,3}=N_0+N_3=1+8=9$

~~$N_{1,2}=N_1+N_2=2+4=6$~~

$N_{1,3}=N_1+N_3=2+8=10$

~~$N_{2,3}=N_2+N_3=4+8=12$~~

~~$N_{0,1,2}=N_0+N_1+N_2=1+2+4=7$~~

$N_{0,1,3}=N_0+N_1+N_3=1+2+8=11$

Выбираем вершину $N(\Omega)=15$:

~~$N_0=15-1=14$~~

~~$N_1=15-2=13$~~

$N_2=15-4=11$

~~$N_3=15-8=7$~~

$N_4=15+16=31$

~~$N_{2,4}=N_2+N_4-N=11+31-15=27$~~

Выбираем вершину $N(\Omega)=20$:

$N_0=20+1=21$

~~$N_1=20+2=22$~~

~~$N_2=20-4=16$~~

$N_3=20+8=28$

$N_4=20-16=4$

$N_{0,3}=N_0+N_3-N=21+28-20=29$

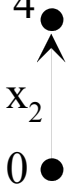
~~$N_{0,4}=N_0+N_4-N=21+4-20=5$~~

~~$N_{3,4}=N_3+N_4-N=28+4-20=12$~~

~~$N_{0,3,4}=N_0+N_3+N_4-2N=21+28+4-40=13$~~

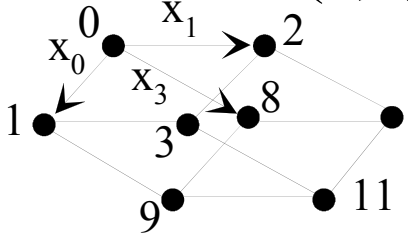
Простые элементы, содержащие вершину $N(\Omega)=0$, $\Omega=00000$:

Элемент (2)



$$x_0^{\omega_0} x_1^{\omega_1} x_3^{\omega_3} x_4^{\omega_4} = \overline{x}_0 \overline{x}_1 \overline{x}_3 \overline{x}_4$$

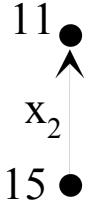
Элемент (0,1,3)



$$x_2^{\omega_2} x_4^{\omega_4} = \overline{x}_2 \overline{x}_4$$

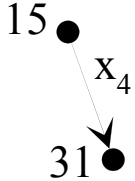
Простые элементы, содержащие вершину $N(\Omega)=15$, $\Omega=11110$:

Элемент (2)



$$x_0^{\omega_0} x_1^{\omega_1} x_3^{\omega_3} x_4^{\omega_4} = x_0 x_1 x_3 \overline{x}_4$$

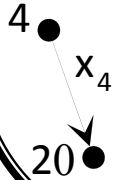
Элемент (4)



$$x_0^{\omega_0} x_1^{\omega_1} x_2^{\omega_2} x_3^{\omega_3} = x_0 x_1 x_2 x_3$$

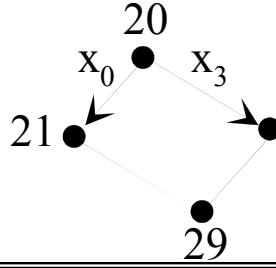
Простые элементы, содержащие вершину $N(\Omega)=20$, $\Omega=00101$:

Элемент (4)



$$x_0^{\omega_0} x_1^{\omega_1} x_2^{\omega_2} x_3^{\omega_3} = \overline{x}_0 \overline{x}_1 x_2 \overline{x}_3$$

Элемент (0,3)



$$x_1^{\omega_1} x_2^{\omega_2} x_4^{\omega_4} = \overline{x}_1 x_2 x_4$$

Решение задачи покрытия

Существенными элементами являются $\bar{x}_2\bar{x}_4$ и $\bar{x}_1x_2x_4$. Они покрывают следующие вершины: 0, 1, 2, 3, 8, 9, 10, 11, 20, 21, 28, 29.

Остались непокрытыми вершины 4, 15, 31.
Элемент $x_0x_1x_2x_3$ покрывает вершины 15 and 31.

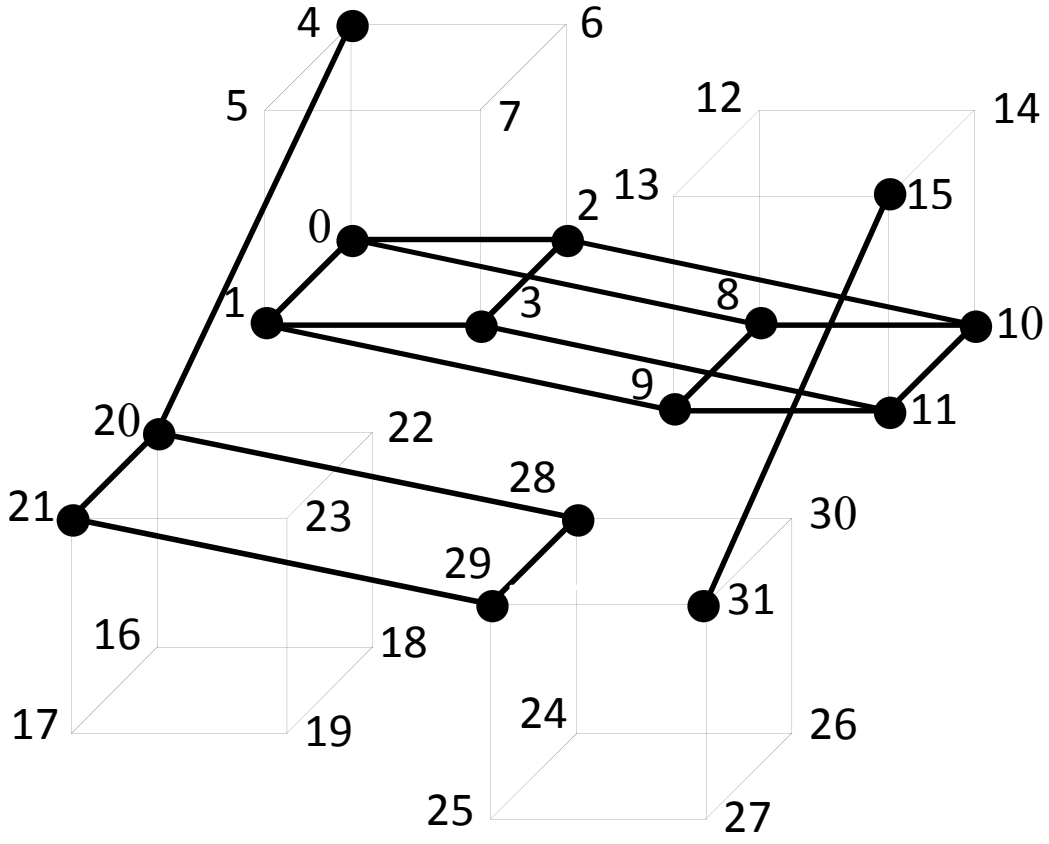
Вершина 4 может быть покрыта одним из двух элементов: $\bar{x}_0\bar{x}_1\bar{x}_3\bar{x}_4$ и $\bar{x}_0\bar{x}_1x_2\bar{x}_3$. Выбираем второй из них.

Одной из возможных неизбыточных оболочек (тупиковой формой) является

$$F = \bar{x}_2\bar{x}_4 + \bar{x}_1x_2x_4 + x_0x_1x_2x_3 + \bar{x}_0\bar{x}_1x_2\bar{x}_3$$

Можем ли мы быть уверены, что эта форма минимальная? Для данного примера это так, но в общем случае эта процедура не может дать минимального решения.

Неизбыточная оболочка



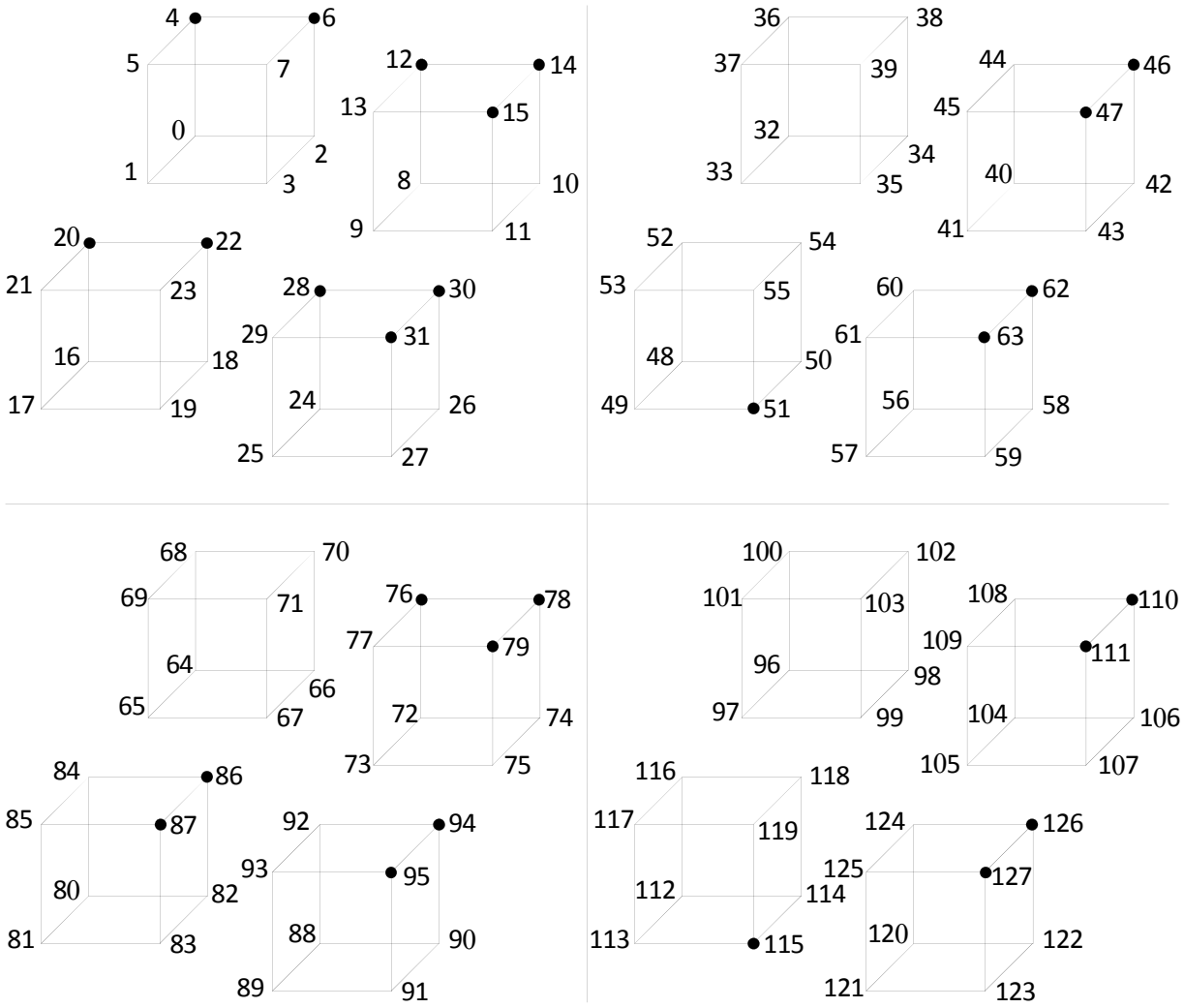
Заключение

- Для получения *сокращенной оболочки* (всех простых элементов) повторить эту процедуру для каждого возможного выбора первой вершины и всех последующих выборов вершин. Только в сокращенной оболочке могут быть найдены все неизбыточные оболочки и выбраны среди них минимальные оболочки. Очень часто нас удовлетворяет нахождение не минимального решения, а достаточно хорошего.

- Сложность процедуры зависит только от числа вершин в множестве T и не зависит от числа переменных.

- Если число вершин в множестве T больше, чем 2^{n-1}, где n – число переменных, то следует применить процедуру к множеству F вершин, после чего инвертировать полученную функцию.

Упражнение



$T = \{4, 6, 12, 14, 15, 20, 22, 28, 30, 31, 46, 47, 51, 62, 63, 76, 78, 79, 86, 87, 94, 95, 110, 111, 115, 126, 127\}$

T	4	15	51	76	86
4	--				
6	1				
12	3			6	
14	1,3	0		1,6	
15		--			
20	4				
22	1,4				6
28	3,4				
30	1,3,4	0,4			3,6
31		4			
46		0,5			
47		5			
51			--		
62		0,4,5			

T	4	15	51	76	86
63		4,5			
76				--	
78		0,6		1	
79		6			
86					--
87					0
94		0,4,6			3
95		4,6			0,3
110		0,5,6			
111		5,6			
115			6		
126		0,4,5,6			
127		4,5,6			

1. $N(\Omega)=4$, $\Omega=0010000$

~~$N_0=4+1=5$~~
$N_1=4+2=6$
~~$N_2=4-4=0$~~
$N_3=4+8=12$
$N_4=4+16=20$
~~$N_5=4+32=36$~~
~~$N_6=4+64=68$~~
$N_{1,3}=6+12-4=14$
$N_{1,4}=6+20-4=22$
$N_{3,4}=12+20-4=28$
- $N_{1,3,4}=14+20-4=30$

$$\overline{x}_0 x_2 \overline{x}_5 \overline{x}_6$$

2. $N(\Omega)=15$, $\Omega=1111000$

$N_0=15-1=14$
~~$N_1=15-2=13$~~
~~$N_2=15-4=11$~~
~~$N_3=15-8=7$~~
$N_4=15+16=31$
$N_5=15+32=47$
$N_6=15+64=79$
$N_{0,4}=14+31-15=30$
$N_{0,5}=14+47-15=46$
$N_{0,6}=14+79-15=78$
$N_{4,5}=31+47-15=63$
$N_{4,6}=31+79-15=95$
$N_{5,6}=47+79-15=111$
$N_{0,4,5}=30+47-15=62$
$N_{0,4,6}=30+79-15=94$
$N_{0,5,6}=46+79-15=110$
$N_{4,5,6}=63+79-15=127$
- $N_{0,4,5,6}=62+79-15=126$

$$x_1 x_2 x_3$$

3. $N(\Omega)=51$, $\Omega=1100110$

~~$N_0=51-1=50$~~
~~$N_1=51-2=49$~~
~~$N_2=51+4=55$~~
~~$N_3=51+8=59$~~
~~$N_4=51-16=35$~~
~~$N_5=51-32=19$~~
- $N_6=51+64=115$

$$x_0 x_1 \overline{x}_2 \overline{x}_3 x_4 x_5$$

4. $N(\Omega)=76, \Omega=0011001$

~~$N_0=76+1=77$~~
$N_1=76+2=78$
~~$N_2=76-4=72$~~
~~$N_3=76-8=68$~~
~~$N_4=76+16=92$~~
~~$N_5=76+32=108$~~
$N_6=76-64=12$

$\bar{x}_0 x_2 x_3 \bar{x}_4 \bar{x}_5$

- $N_{1,6}=78+12-76=14$

5. $N(\Omega)=86, \Omega=0110101$

$N_0=86+1=87$
~~$N_1=86-2=84$~~
~~$N_2=86-4=82$~~
$N_3=86+8=94$
~~$N_4=86-16=70$~~
~~$N_5=86+32=118$~~
$N_6=86-64=22$

$x_1 x_2 x_4 \bar{x}_5 x_6$

$\bar{x}_0 x_1 x_2 x_4 \bar{x}_5$

- $N_{0,3}=87+94-86=95$

~~$N_{0,6}=87+22-86=23$~~

- $N_{3,6}=94+22-86=30$

Результат:

$$F = x_1 x_2 x_3 + \bar{x}_0 x_2 \bar{x}_5 \bar{x}_6 + \bar{x}_0 x_2 x_3 \bar{x}_4 \bar{x}_5 +$$
$$+ x_1 x_2 x_4 \bar{x}_5 x_6 + x_0 x_1 \bar{x}_2 \bar{x}_3 x_4 x_5$$



Логическое Проектирование

Лекция 4

Минимизация слабо определенных булевых функций

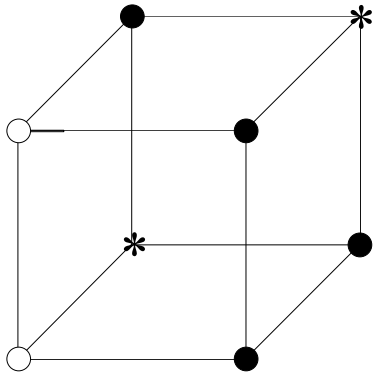
1. Слабо определенные булевы функции

$$A = T \cup F \cup N$$

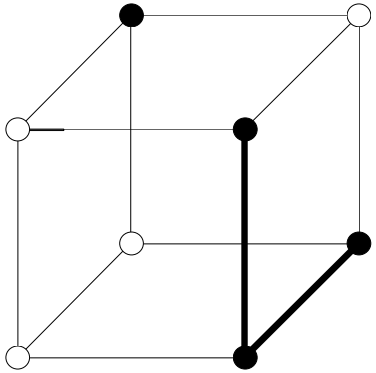
$$f(\Omega \in T) = 1$$

$$f(\Omega \in F) = 0$$

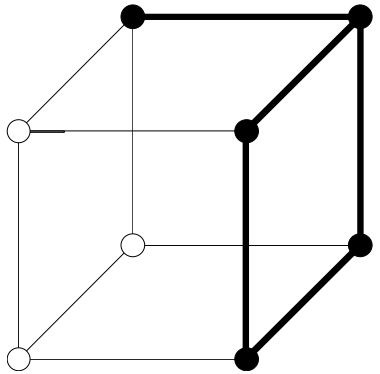
$$f(\Omega \in N) = *$$ (произвольное значение)



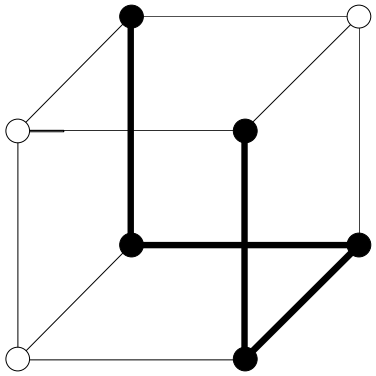
Если $P(N)$ – число вершин типа * в множестве N, тогда $2^{P(N)}$ – число возможных доопределений функции.



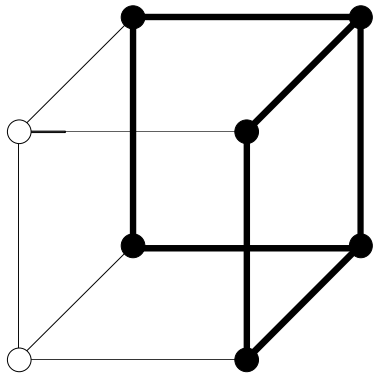
$$\overline{x}_0\overline{x}_1x_2 + x_0x_1 + x_1\overline{x}_2$$



$$\overline{x}_0x_2 + x_1$$



$$\overline{x}_0\overline{x}_1 + x_0x_1 + x_1\overline{x}_2$$



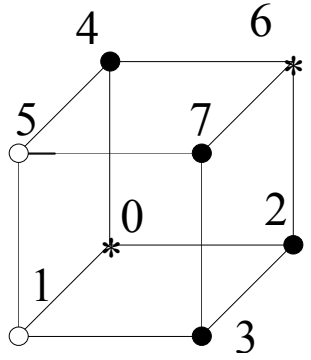
$$\overline{x}_0 + x_1$$

Вопрос 1:

Как отделить вершину $\Omega \in T$ *от множества* F*?*

Переменная x_j отделяет вершину Ω_a от вершины Ω_b, если $\omega_{aj} \neq \omega_{bj}$, или $\omega_{aj} \oplus \omega_{bj} = m_{abj} = 1$.

Вектор $M_{ab} = \{m_{ab0}, m_{ab1}, ..., m_{ab(n-1)}\} = \Omega_a \oplus \Omega_b$ является разделяющим вектором, элементы которого указывают на переменные, разделяющие вершины Ω_a и Ω_b.



$M_{41} = 4 \oplus 1 = 101\ (x_0 \text{ или } x_2)$;

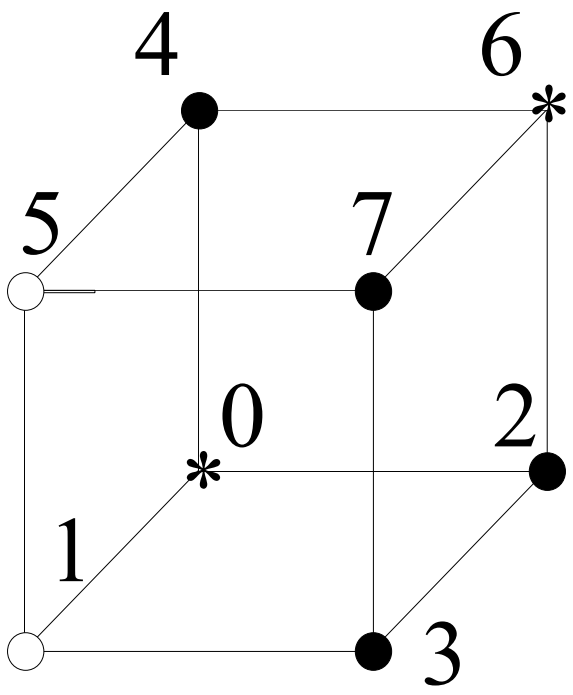
$M_{45} = 4 \oplus 5 = 100\ (x_0)$, тогда если

$\varphi_1(X) = x_0^{\omega_{4,0}} = x_0^0 = \bar{x}_0$,

$\varphi_1(4) = 1;\ \varphi_1(1) = \varphi_1(5) = 0.$

Вопрос 2:

Как отделить множество T от множества F?



$$M_{21} = 2 \oplus 1 = 110\,(x_0, x_1);$$
$$M_{25} = 2 \oplus 5 = 111\,(x_0, x_1, x_2);$$
$$M_{31} = 3 \oplus 1 = 010\,(x_1);$$
$$M_{35} = 3 \oplus 5 = 011\,(x_1, x_2);$$
$$M_{71} = 7 \oplus 1 = 011\,(x_1, x_2);$$
$$M_{75} = 7 \oplus 5 = 010\,(x_1);$$

Если $\varphi_2(X) = x_1$, то

$\varphi_2(2) = \varphi_2(3) = \varphi_2(7) = 1$ и

$\varphi_2(1) = \varphi_2(5) = 0.$

В результате: $\varphi(X) = \varphi_1(X) + \varphi_2(X) = \overline{x}_0 + x_1$

Таблица связей

		F_1	F_2	F_3	F_4
		11110000	**01111000**	**00111100**	**00011110**
T_1	**00000000**	11110000	01111000	00111100	00011110
T_2	**01010100**	10100100	00101100	01101100	01001010
T_3	**11011010**	00101010	10100010	11100110	11000100

Строка 1: $(\bar{x}_0 + \bar{x}_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4)(\bar{x}_2 + \bar{x}_3 + \bar{x}_4 + \bar{x}_5) \bullet$
$(\bar{x}_3 + \bar{x}_4 + \bar{x}_5 + \bar{x}_6) = \bar{x}_3 + \bar{x}_0\bar{x}_4 + \bar{x}_1\bar{x}_4 + \bar{x}_1\bar{x}_5 + \bar{x}_2\bar{x}_4 + \underline{\bar{x}_2\bar{x}_5} + \underline{\underline{x_2\bar{x}_6}};$

Строка 2: $(\bar{x}_0 + \bar{x}_2 + x_5)(\bar{x}_2 + \bar{x}_4 + x_5)(x_1 + \bar{x}_2 + \bar{x}_4 + x_5) \bullet$
$(x_1 + \bar{x}_4 + \bar{x}_6) = \bar{x}_0\bar{x}_4 + x_1\bar{x}_2 + x_1x_5 + \bar{x}_2\bar{x}_4 + \underline{\underline{\bar{x}_2\bar{x}_6}} + \bar{x}_4x_5 + x_5\bar{x}_6;$

Строка 3: $(\bar{x}_2 + x_4 + x_6)(x_0 + \bar{x}_2 + x_6)(x_0 + x_1 + \bar{x}_2 + \bar{x}_5 + x_6) \bullet$
$(x_0 + x_1 + \bar{x}_5) = x_0\bar{x}_2 + x_0x_4 + x_0x_6 + x_1\bar{x}_2 + x_1x_6 + \underline{\bar{x}_2\bar{x}_5} + \bar{x}_5x_6.$

Минимальная форма: $\bar{x}_2\bar{x}_5 + \bar{x}_2\bar{x}_6 = \overline{x_2 + x_5x_6}$

Вычисление существенных элементов

Как вы уже знаете, *каждый элемент C_k (подкуб) n-куба может быть определен с помощью двух векторов:*

Ω_j – любая вершина, принадлежащая подкубу, и

B_k – код-индикатор (b_{ki}=1 для внешних переменных, b_{ki}=0 для внутренних переменных).

Example: $C_k = x_1 \bar{x}_3 x_5$

$T_j = 01001100011$

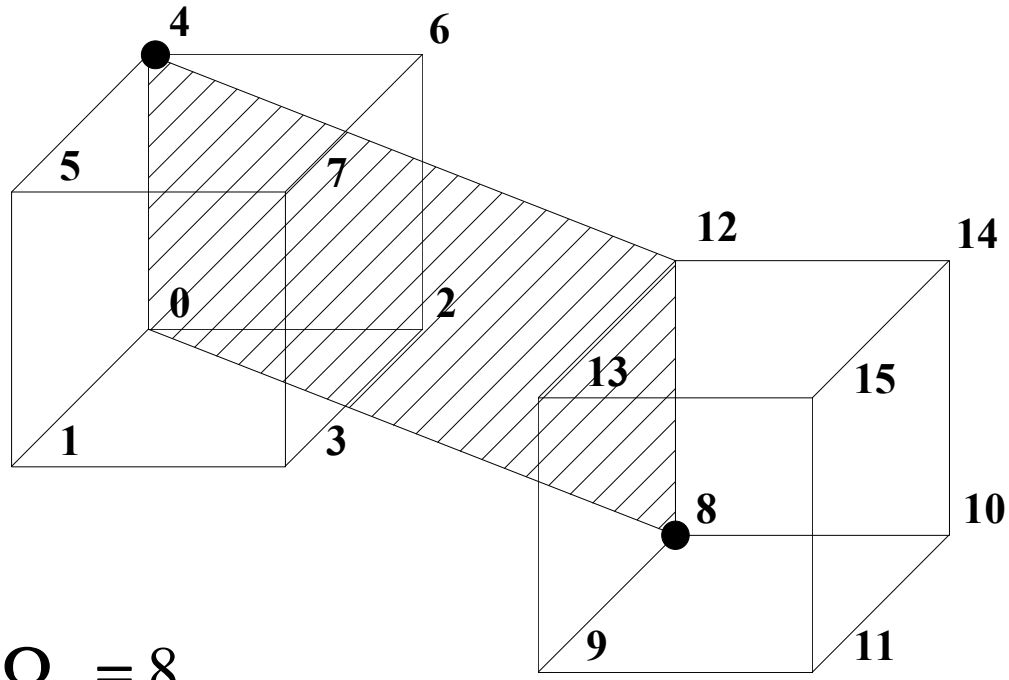
$B_k = 01010100000$

$$C_k = \&_{i=0}^{n-1} (x_i^{\omega_{ji}} + \bar{b}_{ki})$$

Если $\Omega_a \in T$ and $\Omega_b \in T$ – вершины, являющиеся концами главной диагонали в элементе $C_k = \{\Omega_a, B_k\}$, то

$$B_k = \Omega_a \,\overline{\oplus}\, \Omega_b$$

$$b_{ki} = \omega_{ai} \,\overline{\oplus}\, \omega_{bi} = \omega_{ai}\omega_{bi} + \bar{\omega}_{ai}\bar{\omega}_{bi}$$



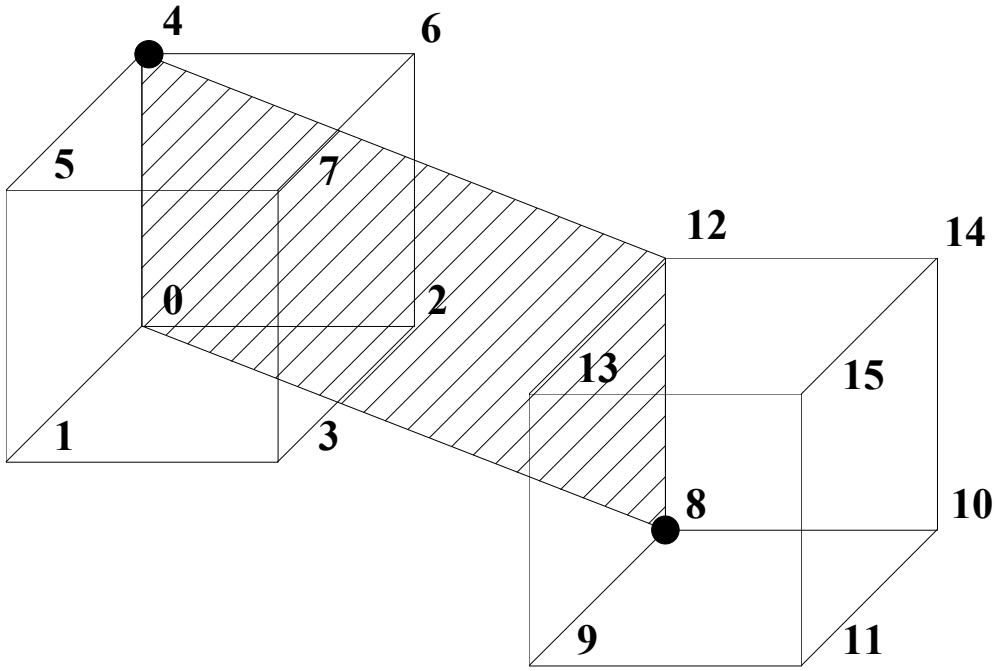
$$\Omega_a = 4, \Omega_b = 8,$$

$$B = 4 \overline{\oplus} 8 = \left|0010\right| \overline{\oplus} \left|0001\right| = \left|1100\right| = 3$$

$$C = (x_0^0 + 0)(x_1^0 + 0)(x_2^1 + 1)(x_3^0 + 1) = \overline{x}_0 \overline{x}_1$$

Вершина Ω_c принадлежит элементу $C = \{\Omega_a, B\}$, если Ω_c имеет те же значения координат, которые являются внешними для элемента C, т.е.

$$\Omega_c \in C, \text{ если } (\Omega_c \oplus \Omega_a) \,\&\, B = 0.$$



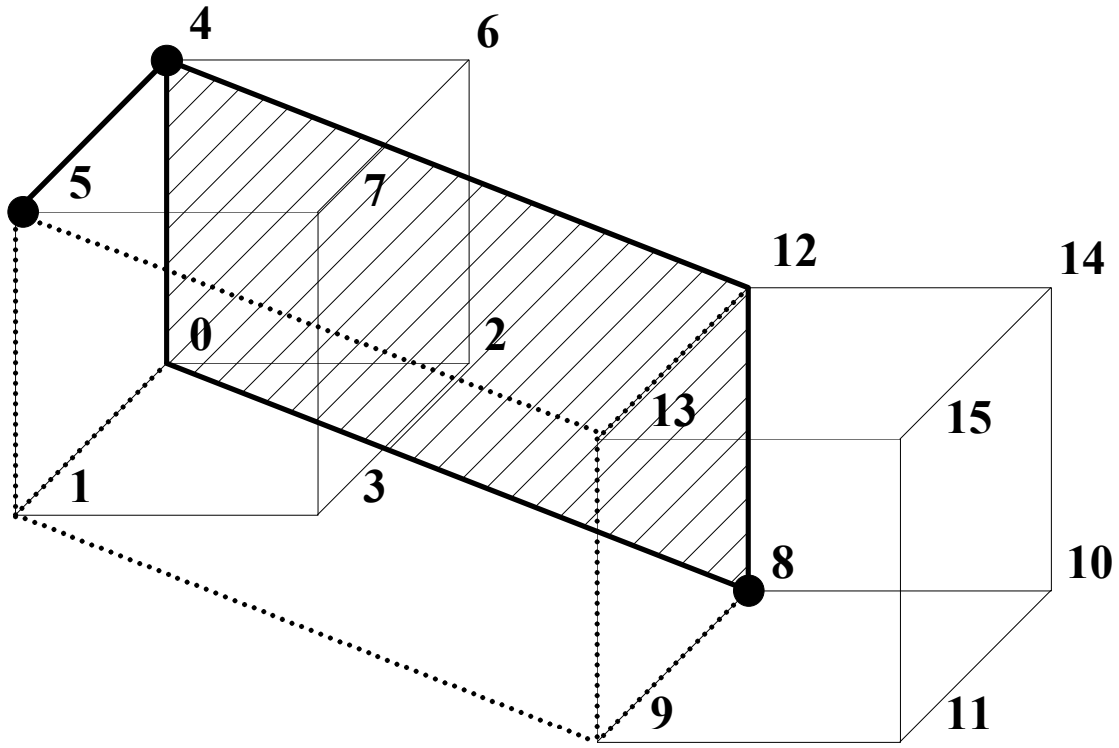
$$(12 \oplus 4)\ \&\ 3 = (|0011| \oplus |0010|)|1100| = |0000| = 0,\ 12 \in C$$

$$(10 \oplus 4)\ \&\ 3 = (|0101| \oplus |0010|)|1100| = |0100| \neq 0,\ 10 \notin C$$

Если $C_1 = \{\Omega_a, B_1\}$ и $C_2 = \{\Omega_a, B_2\}$ – два элемента с общей вершиной Ω_a, то минимальным элементом $C = \{\Omega_a, B\}$, которому принадлежат оба этих элемента ($C_1 \in C,\ C_2 \in C$) является

$$C = \{\Omega_a, B = B_1\ \&\ B_2\}.$$

Пример:

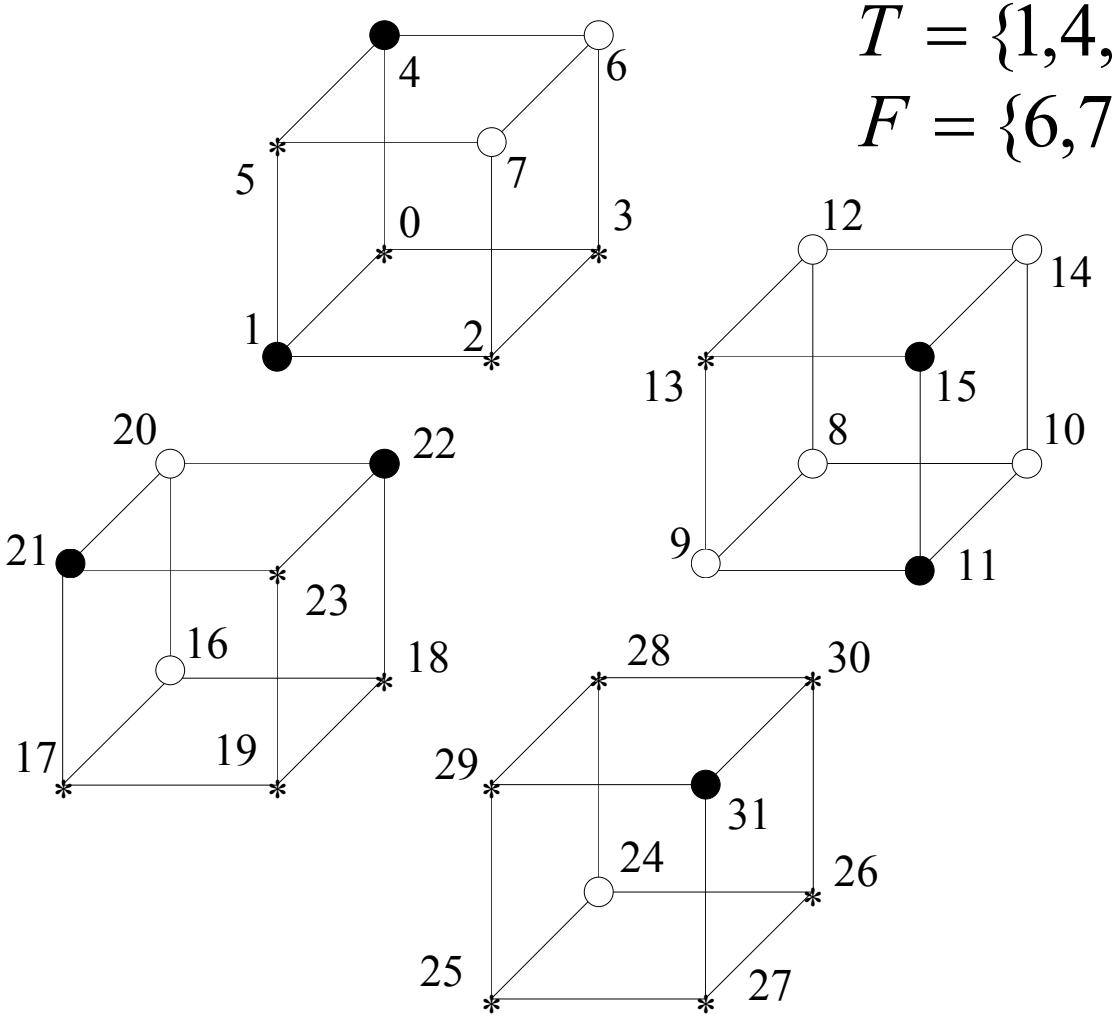


$$C_1 = \{4,3\}; C_2 = \{4,14\}; C = \{4,3 \,\&\, 14 = 2\}$$

$$C = (x_0^0 + 1)(x_1^0 + 0)(x_2^1 + 1)(x_3^0 + 1) = \bar{x}_1$$

Пусть задана следующая функция:

$$T = \{1,4,11,15,21,22,31\};$$
$$F = \{6,7,8,9,10,12,14,16,24\}.$$



Поиск существенных звезд:

$C_1(1) = \{1; 1 \overline{\oplus} 4 = 26\};$

$(1 \oplus 6) \,\&\, 26 \neq 0; 6 \notin C_1(1)$
$(1 \oplus 7) \,\&\, 26 \neq 0; 7 \notin C_1(1)$
$(1 \oplus 8) \,\&\, 26 \neq 0; 8 \notin C_1(1)$
$(1 \oplus 9) \,\&\, 26 \neq 0; 9 \notin C_1(1)$
$(1 \oplus 10) \,\&\, 26 \neq 0; 10 \notin C_1(1)$
$(1 \oplus 12) \,\&\, 26 \neq 0; 12 \notin C_1(1)$
$(1 \oplus 14) \,\&\, 26 \neq 0; 14 \notin C_1(1)$
$(1 \oplus 16) \,\&\, 26 \neq 0; 16 \notin C_1(1)$
$(1 \oplus 20) \,\&\, 26 \neq 0; 20 \notin C_1(1)$
$(1 \oplus 24) \,\&\, 26 \neq 0; 24 \notin C_1(1)$

$C_2(1) = \{1; 1 \overline{\oplus} 11 = 21\};$

$(1 \oplus 6) \,\&\, 21 \neq 0; 6 \notin C_2(1)$
$(1 \oplus 7) \,\&\, 21 \neq 0; 7 \notin C_2(1)$
$(1 \oplus 9) \,\&\, 21 = 0; 9 \in C_2(1)$

$C_3(1) = \{1; 1 \overline{\oplus} 15 = 17\};$

$(1 \oplus 6) \,\&\, 17 \neq 0; 6 \notin C_3(1)$
$(1 \oplus 7) \,\&\, 17 = 0; 7 \in C_3(1)$

$C_4(1) = \{1, 1 \overline{\oplus} 21 = 11\};$

$(1 \oplus 6) \,\&\, 11 \neq 0; 6 \notin C_4(1)$
$(1 \oplus 7) \,\&\, 11 \neq 0; 7 \notin C_4(1)$
$(1 \oplus 8) \,\&\, 11 \neq 0; 8 \notin C_4(1)$
$(1 \oplus 9) \,\&\, 11 \neq 0; 9 \notin C_4(1)$
$(1 \oplus 10) \,\&\, 11 \neq 0; 10 \notin C_4(1)$
$(1 \oplus 12) \,\&\, 11 \neq 0; 12 \notin C_4(1)$
$(1 \oplus 14) \,\&\, 11 \neq 0; 14 \notin C_4(1)$
$(1 \oplus 16) \,\&\, 11 \neq 0; 16 \notin C_4(1)$
$(1 \oplus 20) \,\&\, 11 \neq 0; 20 \notin C_4(1)$
$(1 \oplus 24) \,\&\, 11 \neq 0; 24 \notin C_4(1)$

$C_5(1) = \{1; 1 \overline{\oplus} 22 = 8\};$

$(1 \oplus 6) \,\&\, 8 = 0; 6 \in C_5(1)$

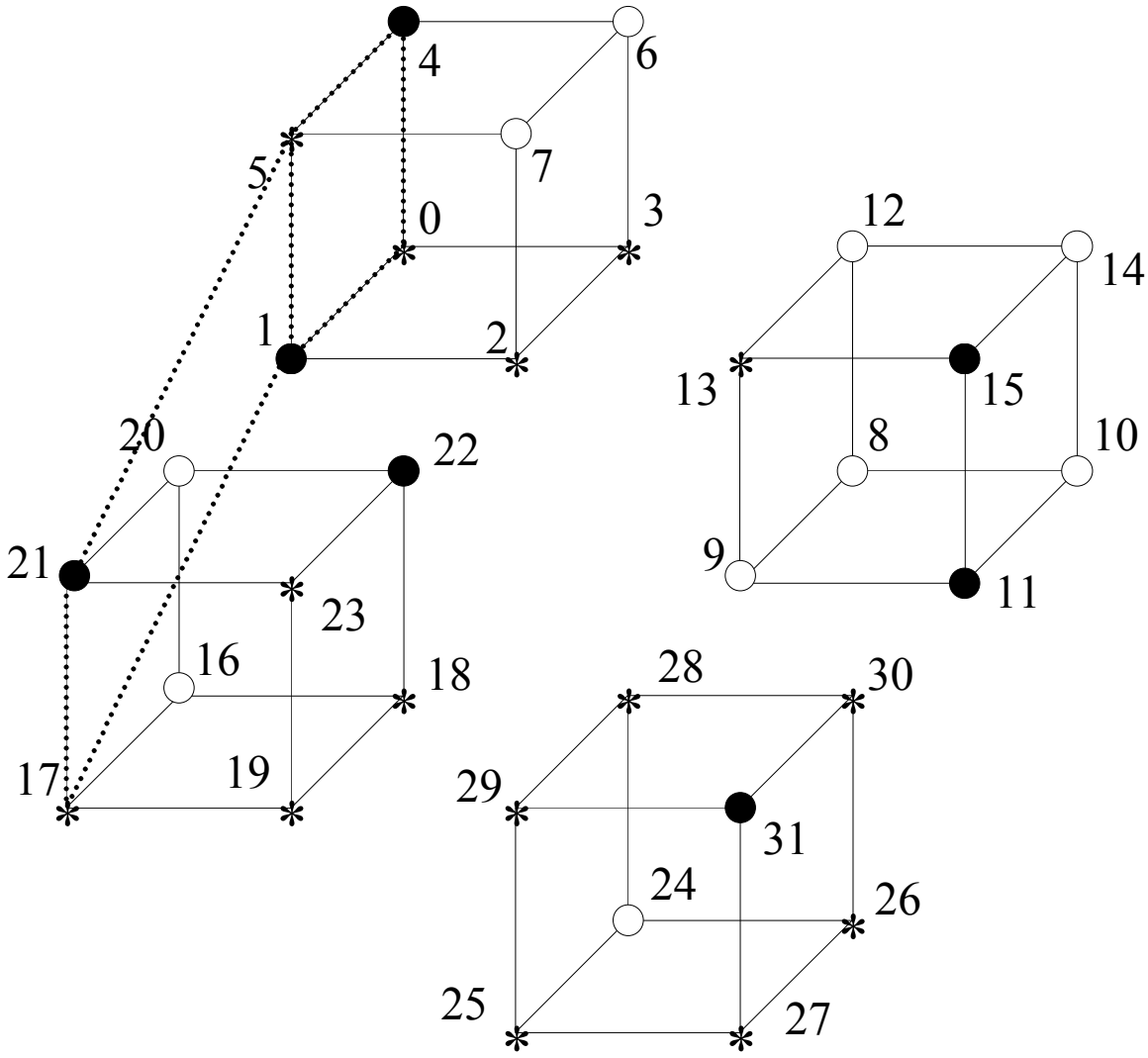
$C_6(1) = \{1; 1 \overline{\oplus} 31 = 1\};$

$(1 \oplus 6) \,\&\, 1 = 0; 6 \in C_5(1)$

Звезда $S(1) = \{C_1(1); C_4(1)\}$ **является несущественной**.

$C_7(1) = \{1; 26 \,\&\, 11 = 10\};$

$(1 \oplus 6) \,\&\, 10 \neq 0; 6 \notin C_7(1)$
$(1 \oplus 7) \,\&\, 10 \neq 0; 7 \notin C_7(1)$
$(1 \oplus 8) \,\&\, 10 \neq 0; 8 \notin C_7(1)$
$(1 \oplus 9) \,\&\, 10 \neq 0; 9 \notin C_7(1)$
$(1 \oplus 10) \,\&\, 10 \neq 0; 10 \notin C_7(1)$
$(1 \oplus 12) \,\&\, 10 \neq 0; 12 \notin C_7(1)$
$(1 \oplus 14) \,\&\, 10 \neq 0; 14 \notin C_7(1)$
$(1 \oplus 16) \,\&\, 10 = 0; 16 \in C_7(1)$


4
6
5
7
0
3
1
2
20
22
21
23
16
18
17
19
12
14
13
15
8
10
9
11
28
30
29
31
24
26
25
27

$C_1(4) = \{4; 4 \overline{\oplus} 1 = 26\};$

$(4 \oplus 6) \& 26 \neq 0; 6 \notin C_1(4)$

$(4 \oplus 7) \& 26 \neq 0; 7 \notin C_1(4)$

$(4 \oplus 8) \& 26 \neq 0; 8 \notin C_1(4)$

$(4 \oplus 9) \& 26 \neq 0; 9 \notin C_1(4)$

$(4 \oplus 10) \& 26 \neq 0; 10 \notin C_1(4)$

$(4 \oplus 12) \& 26 \neq 0; 12 \notin C_1(4)$

$(4 \oplus 14) \& 26 \neq 0; 14 \notin C_1(4)$

$(4 \oplus 16) \& 26 \neq 0; 16 \notin C_1(4)$

$(4 \oplus 20) \& 26 \neq 0; 20 \notin C_1(4)$

$(4 \oplus 24) \& 26 \neq 0; 24 \notin C_1(4)$

$C_2(4) = \{4; 4 \overline{\oplus} 11 = 18\};$

$(4 \oplus 6) \& 18 = 0; 6 \in C_2(4)$

$C_3(4) = \{4; 4 \overline{\oplus} 15 = 20\};$

$(4 \oplus 6) \& 20 = 0; 6 \in C_3(4)$

$C_4(4) = \{4; 4 \overline{\oplus} 21 = 14\};$

$(4 \oplus 16) \& 14 = 0; 16 \in C_2(4)$

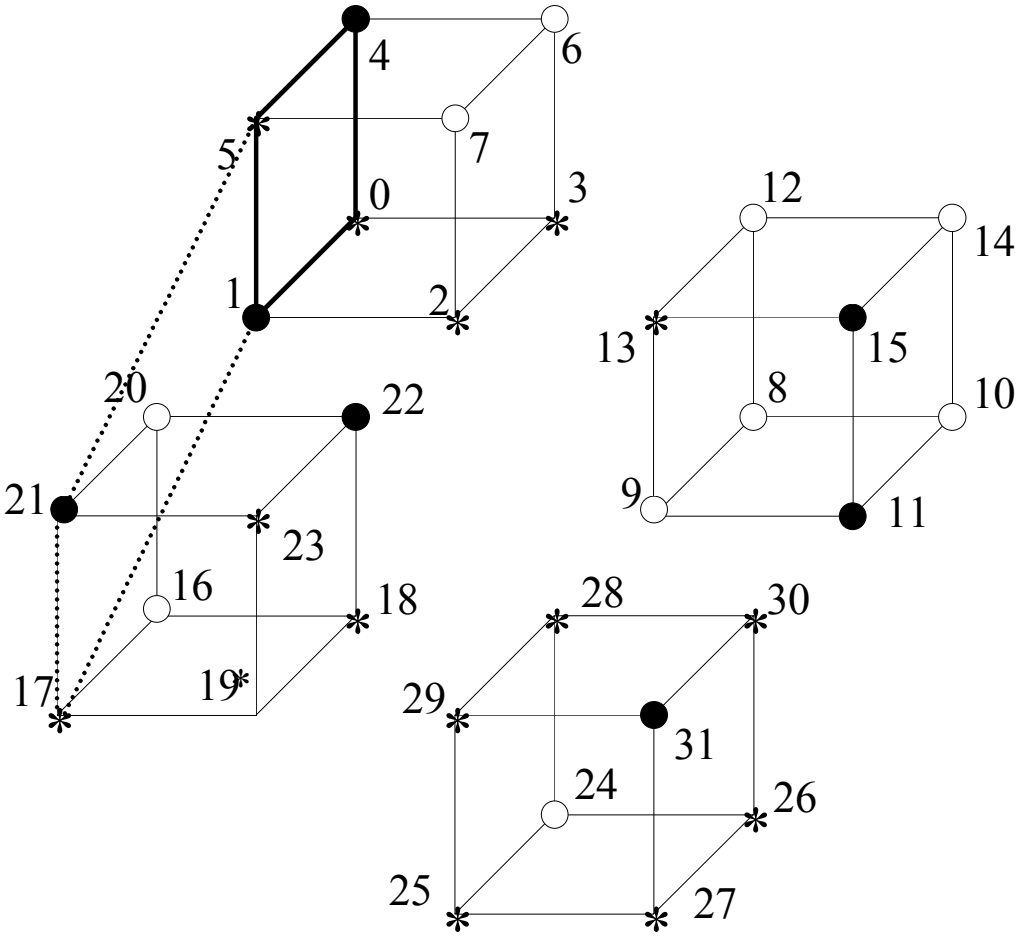
$C_5(4) = \{4; 4 \overline{\oplus} 22 = 13\};$

$(4 \oplus 6) \& 13 = 0; 6 \in C_5(4)$

$C_6(4) = \{4; 4 \overline{\oplus} 31 = 4\};$

$(4 \oplus 31) \& 4 = 0; 6 \in C_2(4)$

$C_1(4)$ –**существенная звезда (существенный элемент)**



$$C_1 = \{11; 11 \overline{\oplus} 15 = 27\};$$

$$(11 \oplus 6) \& 27 \neq 0;\ 6 \notin C_1(11)$$
$$(11 \oplus 7) \& 27 \neq 0;\ 7 \notin C_1(11)$$
$$(11 \oplus 8) \& 27 \neq 0;\ 8 \notin C_1(11)$$
$$(11 \oplus 9) \& 27 \neq 0;\ 9 \notin C_1(11)$$
$$(11 \oplus 10) \& 27 \neq 0;\ 10 \notin C_1(11)$$
$$(11 \oplus 12) \& 27 \neq 0;\ 12 \notin C_1(11)$$
$$(11 \oplus 14) \& 27 \neq 0;\ 14 \notin C_1(11)$$
$$(11 \oplus 16) \& 27 \neq 0;\ 16 \notin C_1(11)$$
$$(11 \oplus 20) \& 27 \neq 0;\ 20 \notin C_1(11)$$
$$(11 \oplus 24) \& 27 \neq 0;\ 24 \notin C_1(11)$$

$C_2(11)=\{11;11\overline{\oplus}31=11\};$

$(11\oplus 6)\&11\neq 0; 6\notin C_1(11)$

$(11\oplus 7)\&11\neq 0; 7\notin C_1(11)$

$(11\oplus 8)\&11\neq 0; 8\notin C_1(11)$

$(11\oplus 9)\&11\neq 0; 9\notin C_1(11)$

$(11\oplus 10)\&11\neq 0; 10\notin C_1(11)$

$(11\oplus 12)\&11\neq 0; 12\notin C_1(11)$

$(11\oplus 14)\&11\neq 0; 14\notin C_1(11)$

$(11\oplus 16)\&11\neq 0; 16\notin C_1(11)$

$(11\oplus 20)\&11\neq 0; 20\notin C_1(11)$

$(11\oplus 24)\&11\neq 0; 24\notin C_1(11)$

$C_3(11)=\{11;11\&27=11\};$

$(11\oplus 6)\&11\neq 0; 6\notin C_1(11)$

$(11\oplus 7)\&11\neq 0; 7\notin C_1(11)$

$(11\oplus 8)\&11\neq 0; 8\notin C_1(11)$

$(11\oplus 9)\&11\neq 0; 9\notin C_1(11)$

$(11\oplus 10)\&11\neq 0; 10\notin C_1(11)$

$(11\oplus 12)\&11\neq 0; 12\notin C_1(11)$

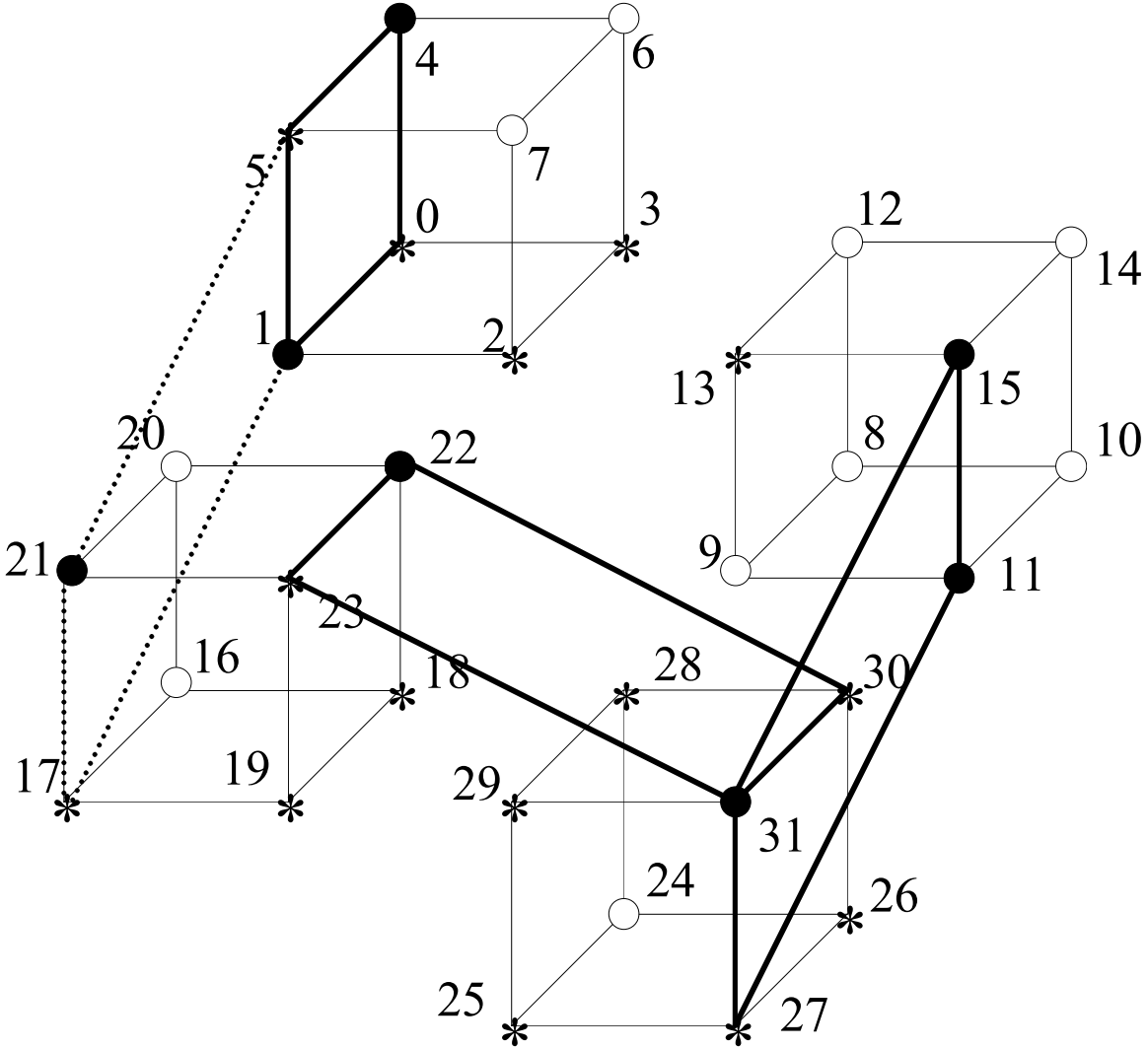
$(11\oplus 14)\&11\neq 0; 14\notin C_1(11)$

$(11\oplus 16)\&11\neq 0; 16\notin C_1(11)$

$(11\oplus 20)\&11\neq 0; 20\notin C_1(11)$

$(11\oplus 24)\&11\neq 0; 24\notin C_1(11)$

$C_3(11)-$ **существенный элемент.**

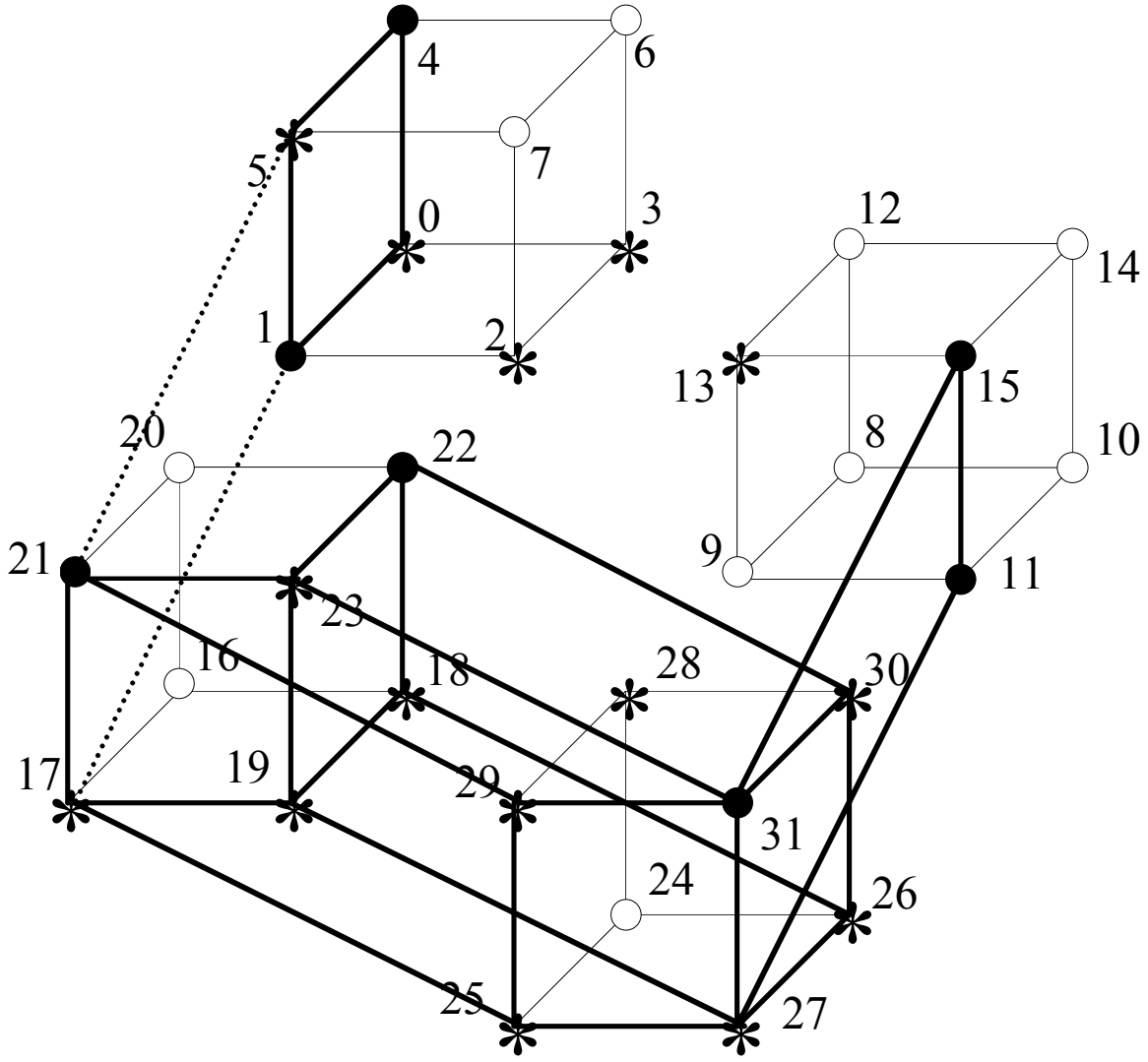


Покрытие (оболочка) функции существенными элементами:

$$C_1(4,26); C_2(11,11); C_3(22,22); 21$$

	T	$C_1(4,26)$	$C_2(11,11)$	$C_3(22,22)$	21
	Ω	**00100**	**11010**	**01101**	**10101**
F	**B**	**01011**	**11010**	**01101**	**-**
6	**01100**	01000	10010	00001	11001
7	**11100**	01000	00010	00001	01001
8	**00010**	00010	11000	01101	10111
9	**10010**	00010	01000	01101	00111
10	**01010**	01010	10000	00101	11111
12	**00110**	00010	11000	00001	10011
14	**01110**	01010	10000	01001	11011
16	**00001**	00001	11010	01100	10100
20	**00101**	00001	11010	01000	10000
24	**00011**	00010	11000	01100	10110
		$C_1(4,26)$	**$C_2(11,11)$**	**$C_3(22,18)$**	**$C_4(21,17)$**

$$\varphi(X) = \overline{x}_1\overline{x}_3\overline{x}_4 + x_0x_1x_3 + x_1x_4 + x_0x_4$$


4
6
5
7
0
3
1
2
12
14
13
15
8
10
9
11
20
22
21
23
16
18
28
30
17
19
29
31
24
26
25
27

Функциональная декомпозиция

- Двухуровневая декомпозиция
- Типы декомпозиций
- Простая разделительная декомпозиция
- Многократная разделительная декомпозиция
- Примеры

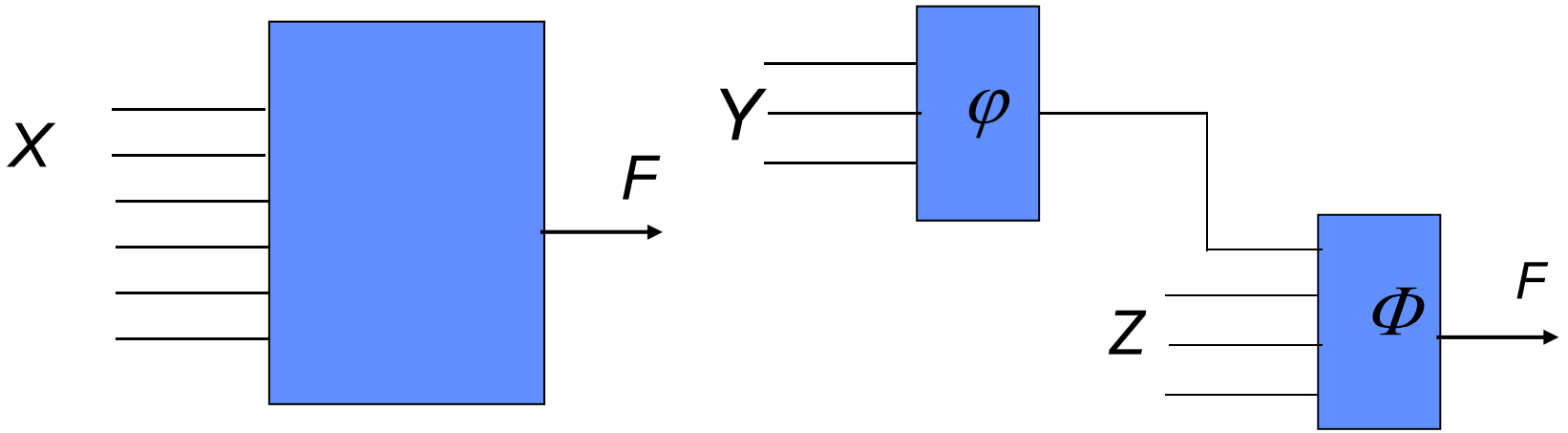
Двухуровневая декомпозиция

$$F(X) = F(Z,Y),\quad Z \cup Y = X,\quad Z \cap Y = \varnothing$$

$$F(Z,Y) = \Phi(Z,\varphi(Y))$$

Z – подмножество свободных переменных,

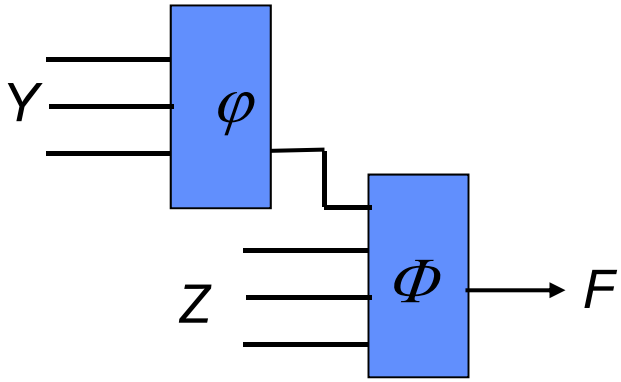
Y – подмножество связанных переменных.



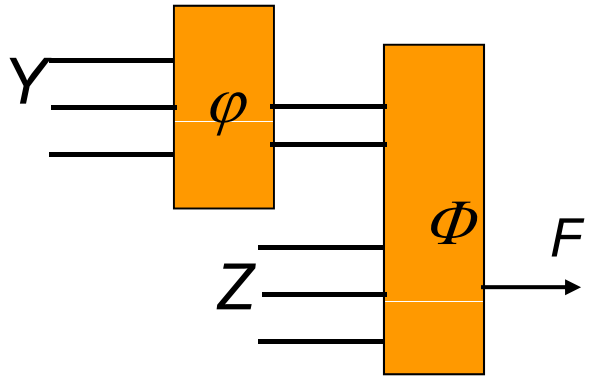
Если $Z \cap Y = \varnothing$, то декомпозиция разделительная.

Если $Z \cap Y \neq \varnothing$, то декомпозиция неразделительная.

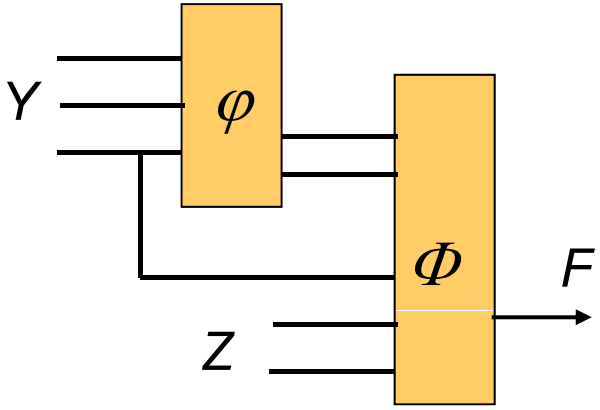
Типы декомпозиций



Простая разделительная **декомпозиция**
(Ашхенхёрст)



Разделительная декомпозиция
(Кёртис)



Неразделительная декомпозиция

Простая разделительная декомпозиция

Если функция

$$F(X) = F(Z,Y), \quad Z \cup Y = X, \quad Z \cap Y = \varnothing$$

может быть представлена в форме

$$F(Z,Y) = \Phi(Z,\varphi(Y)),$$

говорят, что функция $F(X)$ имеет ***простую разделительную декомпозицию относительно разбиения переменных на две группы (Z,Y).***

$\Phi(Z,\varphi)$ *называют* ***образом декомпозиции***, а

$\varphi(Y)$ – ***составляющей*** *(компонентой)* ***декомпозиции***.

Примерами простой разделительной декомпозиции могут служить разложения Шеннона, Рида-Маллера и Варшавского.

Разложение Шеннона:

$$F(x,Y) = [\bar{x} + F(x=1,Y)] \bullet [x + F(x=0,Y)] = \\ = \bar{x}F(x=0,Y) + xF(x=1,Y).$$

Разложение Рида-Маллера:

$$F(x,Y) = F(x=0,Y) \oplus x\frac{\partial F(x,Y)}{\partial x} = \\ F(x=0,Y) \oplus x[F(x=0,Y) \oplus F(x=1,Y)].$$

Разложение Варшавского (диагональное):

$$F(x,z,Y) = (x \oplus z)F(x=\bar{z},z,Y) + \overline{(x \oplus z)}F(x=z,z,Y)$$

(Здесь остаточные функции, или составляющие декомпозиции, вычисляются с помощью отождествления переменных, а в двух предыдущих разложениях – с помощью фиксации значений переменных.)

Если функция $F(X) = F(Z,Y),\ \ Z \cup Y = X,\ \ Z \cap Y = \varnothing$ имеет простую разделительную декомпозицию относительно разбиения (Z,Y), то разложение Шеннона для функции $\Phi(Z,\varphi(Y))$ по $\varphi(Y)$ имеет вид

$$\Phi(Z,\varphi(Y)) = \varphi(Y) f_1(Z) + \overline{\varphi}(Y) f_2(Z),$$

где

$$f_1(Z) = \Phi(Z,\varphi(Y) = 1),\ \ f_2(Z) = \Phi(Z,\varphi(Y) = 0),$$

и прямоугольная таблица для разбиения (Z,Y) имеет только две различные колонки. (Теорема Ашхенхёрста)

		$\varphi(Y)$	
Z		0	1
Ω_0	00....00	$f_2(\Omega_0)$	$f_1(\Omega_0)$
Ω_1	10....00	$f_2(\Omega_1)$	$f_1(\Omega_1)$
.....	.....	.....	.....
$\Omega_{2^{n-k}-1}$	11....11	$f_2(\Omega_{2^{n-k}-1})$	$f_1(\Omega_{2^{n-k}-1})$

Пример простой разделительной декомпозиции:

$$F(X) = F(x_0,x_1,x_2,x_3,x_4) = \Phi(x_0,x_1,\varphi(x_2,x_3,x_4)) =$$

$$= \varphi\,\bar{x}_0\bar{x}_1 + \bar{\varphi}\,x_0x_1$$ *- образ декомпозиции*

$$\varphi = x_2x_3 + x_2x_4 + x_3x_4$$ *- компонента декомпозиции*

$$F(X) = (x_2x_3 + x_2x_4 + x_3x_4)\bar{x}_0\bar{x}_1 +$$

$$+\overline{(x_2x_3 + x_2x_4 + x_3x_4)}x_0x_1 =$$

$$= \bar{x}_0\bar{x}_1x_2x_3 + \bar{x}_0\bar{x}_1x_2x_4 + \bar{x}_0\bar{x}_1x_3x_4 +$$

$$+ x_0x_1\bar{x}_2\bar{x}_3 + x_0x_1\bar{x}_2\bar{x}_4 + x_0x_1\bar{x}_3\bar{x}_4$$

С декомпозицией - 10 двуместных логических операций;

Без декомпозиции - 23 двуместные логические операции.

Таблица функции для заданного разбиения переменных

	Ω_Y	0	1	2	3	4	5	6	7
	$\boldsymbol{Y}$	000	100	010	110	001	101	011	111
Ω_Z	$\boldsymbol{Z}$								
0	00	0	0	0	1	0	1	1	1
1	10	0	0	0	0	0	0	0	0
2	01	0	0	0	0	0	0	0	0
3	11	1	1	1	0	1	0	0	0
		$\varphi(Y) = \varphi(x_2, x_3, x_4)$							
		0	0	0	1	0	1	1	1

Если прямоугольная таблица функции $F(X) = F(Z,Y)$ ***для разбиения (Z,Y)*** ($Z \cup Y = X, Z \cap Y = \varnothing$) ***имеет только две различные колонки, то функция имеет простую разделительную декомпозицию относительно этого разбиения*** $F(Z,Y) = \Phi(Z,\varphi(Y))$. ***Функции образа декомпозиции*** $f_1(Z) = \Phi(Z,\varphi(Y) = 0)$ и $f_2(Z) = \Phi(Z,\varphi(Y) = 1)$ ***определяются этими колонками, а компонента декомпозиции*** $\varphi(Y)$ ***и ее инверсия определяются ненулевыми строками таблицы.***

Общий вид таблицы функции, имеющей простую разделительную декомпозицию

Ω_Y	0	.....	N_Y	....	2^k-1
Ω_Z					
0	$f_1(0)$	.....	$f_2(0)$	.....	$f_1(0)$
1	$f_1(1)$	.....	$f_2(1)$	.....	$f_1(1)$
.....	.....	.....	.....	.....	.....
N_z	$f_1(N_z)$	.....	$f_2(N_z)$	.....	$f_1(N_z)$
.....	.....	.....	.....	.....	.....
2^{n-k}-2	$f_1(2^{n-k}-2)$	.....	$f_2(2^{n-k}-2)$	.....	$f_1(2^{n-k}-2)$
2^{n-k}-1	$f_1(2^{n-k}-1)$	.....	$f_2(2^{n-k}-1)$	.....	$f_1(2^{n-k}-1)$
	$\varphi(Y)$				
	1	.....	***0***	.....	***1***

Пример:

Функция 6 переменных: $Y = \{x_3, x_4, x_5\}$, $Z = \{x_0, x_1, x_2\}$ и

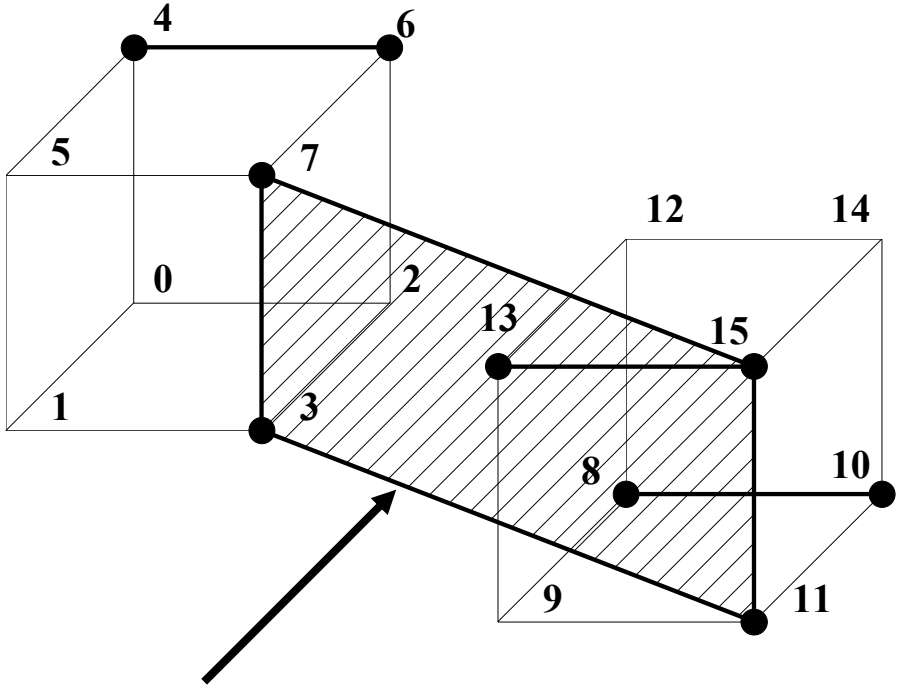
$$\Omega_x = \Omega_z + 2^3 \Omega_y .$$

$T = \{3,4,6,7,8,10,11,13,15,16,18,19,21,23,27,28,30,31,$
$32,34,35,37,39,40,42,43,45,47,48,50.51,53,55,59,60,62,63\}$

	0	1	2	3	4	5	6	7
0	0	1	1	0	1	1	1	0
1	0	0	0	0	0	0	0	0
2	0	1	1	0	1	1	1	0
3	1	1	1	1	1	1	1	1
4	1	0	0	1	0	0	0	1
5	0	1	1	0	1	1	1	0
6	1	0	0	1	0	0	0	1
7	1	1	1	1	1	1	1	1

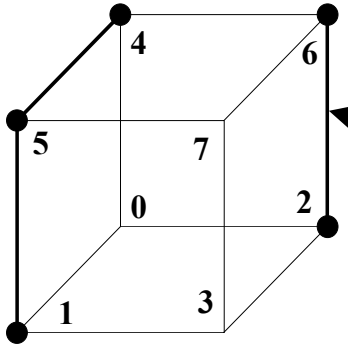
$$T_{\varphi(Y)} = \{1,2,4,5,6\};\ T_{f_1(Z)} = \{0,2,3,5,7\};\ T_{f_2(Z)} = \{3,4,6,7\}.$$

$\varphi(Y)$	0	1
Ω_z		
0	0	1
1	0	0
2	0	1
3	1	1
4	1	0
5	0	1
6	1	0
7	1	1



Образ декомпозиции:

$$\Phi(Z,\varphi(Y)) = \overline{x}_0 x_2 \overline{\varphi} + x_0 x_1 + x_0 x_2 \varphi + \overline{x}_0 \overline{x}_2 \varphi$$



Составляющая декомпозиции:

$$\varphi(Y) = x_3 \overline{x}_4 + \overline{x}_3 x_4 + \overline{x}_4 x_5$$



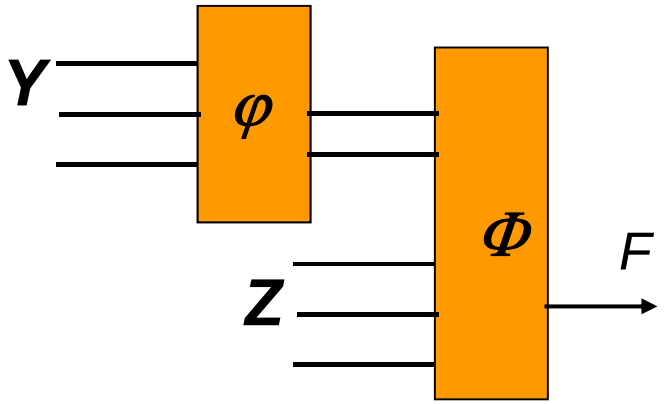
Логическое Проектирование

Лекция 5

Многократная разделительная декомпозиция

Если функция

$$F(X) = F(Z,Y),\ Z \cup Y = X,\ Z \cap Y = \varnothing$$



может быть представлена в виде

$$F(Z,Y) = \Phi(Z, \varphi_0(Y), \ldots, \varphi_{r-1}(Y)),$$

то говорят, что функция $F(X)$ имеет

многократную разделительную декомпозицию относительно разбиения (Z,Y).

Функция $\Phi(\Theta)$ – **образ декомпозиции**;

Функции $\varphi_j(Y)$ – **компоненты декомпозиции.**

Если функция

$$F(X) = F(Z,Y) \quad Z \cup Y = X, Z \cap Y = \varnothing$$

имеет разделительную декомпозицию относительно разбиения (Z,Y), то для функции $F(Z,Y) = \Phi(Z, \varphi_0(Y),..., \varphi_{r-1}(Y))$ может быть построено разложение Шеннона

$$\begin{aligned}
&\Phi(Z, \varphi_0(Y), \varphi_1(Y),..., \varphi_{r-1}(Y)) = \\
&\overline{\varphi}_0(Y)\overline{\varphi}_1(Y)...\overline{\varphi}_{r-1}(Y) f_0(Z) + \\
&\varphi_0(Y)\overline{\varphi}_1(Y)...\overline{\varphi}_{r-1}(Y) f_1(Z) + \\
&..... + \\
&\varphi_0(Y)\varphi_1(Y)...\varphi_{r-1}(Y) f_{2^r-1}(Z)
\end{aligned}$$

, где

$$f_0(Z) = \Phi(Z, \varphi_0(Y) = 0, \varphi_1(Y) = 0,..., \varphi_{r-1}(Y) = 0);$$
$$f_1(Z) = \Phi(Z, \varphi_0(Y) = 1, \varphi_1(Y) = 0,..., \varphi_{r-1}(Y) = 0);$$
$$.....$$
$$f_{2^r-1}(Z) = \Phi(Z, \varphi_0(Y) = 1, \varphi_1(Y) = 1,..., \varphi_{r-1}(Y) = 1);$$

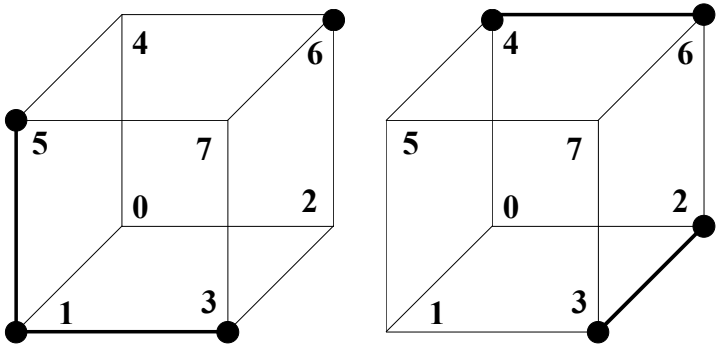
и прямоугольная таблица для разбиения (Z,Y) имеет $\leq 2^r$ различных столбцов.

Если прямоугольная таблица функции для разбиени (Z,Y) имеет $2^{r-1} < N_c \le 2^r$ ***различных столбцов, то функция может быть представлена в виде:***

$$F(Z,Y) = \Phi(Z, \varphi_0(Y), ..., \varphi_{r-1}(Y)).$$

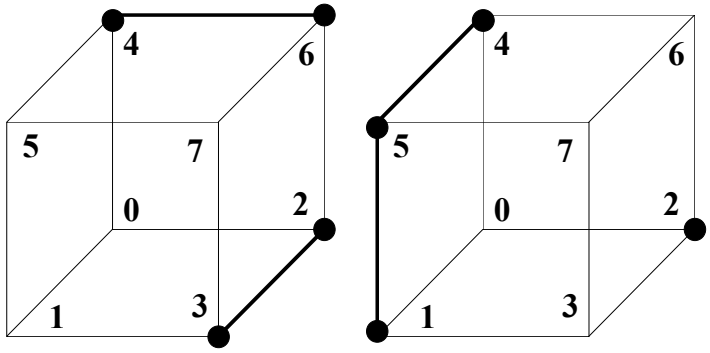
Ω_y	0	1	2	3	4	5	6	7
Ω_z								
0	0	0	0	0	0	0	0	0
1	0	1	1	0	1	1	0	0
2	0	0	0	0	0	0	0	0
3	1	0	0	1	0	0	1	1
4	0	0	0	0	0	0	0	0
5	0	1	1	1	1	1	1	0
6	0	0	0	0	0	0	0	0
7	1	0	1	1	1	0	1	1
Ω_φ^1	0	1	2	3	2	1	3	0
Ω_φ^2	0	2	3	1	3	2	1	0

Ω_y	0	1	2	3	4	5	6	7
φ_0^1, φ_1^1	0,0	1,0	0,1	1,1	0,1	1,0	1,1	0,0



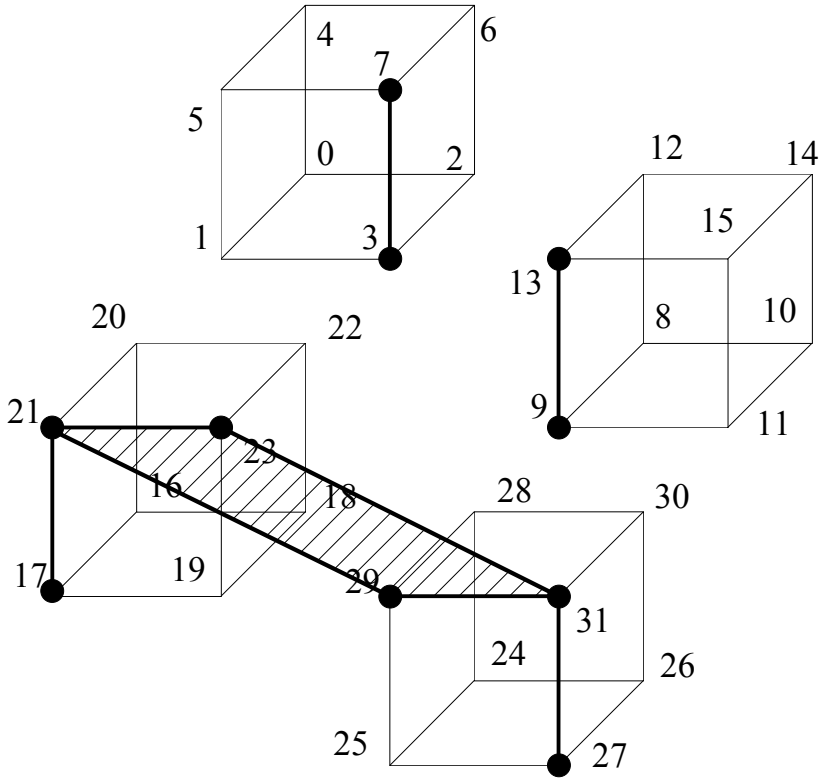
$$\varphi_0^1(Y) = x_3\bar{x}_4 + x_3\bar{x}_5 + \bar{x}_3x_4x_5;$$
$$\varphi_1^1(Y) = \bar{x}_3x_5 + x_4\bar{x}_5.$$

Ω_y	0	1	2	3	4	5	6	7
φ_0^2, φ_1^2	0,0	0,1	1,1	1,0	1,1	0,1	1,0	0,0



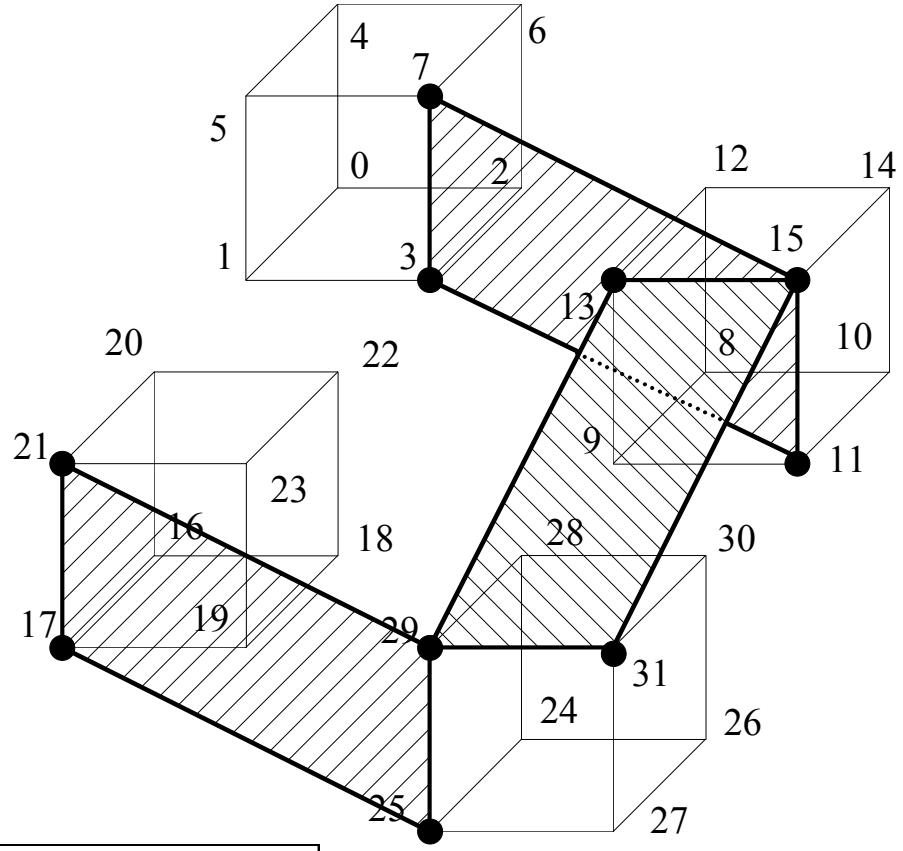
$$\varphi_0^2(Y) = \bar{x}_3x_5 + x_4\bar{x}_5;$$
$$\varphi_1^2(Y) = x_3\bar{x}_4 + \bar{x}_4x_5 + \bar{x}_3x_4\bar{x}_5$$

Ω_{φ}^{1}	0	1	2	3
Ω_{z}				
0	0	0	0	0
1	0	1	1	0
2	0	0	0	0
3	1	0	0	1
4	0	0	0	0
5	0	1	1	1
6	0	0	0	0
7	1	0	1	1



$$\Phi(x_0, x_1, x_2, \varphi_0^1, \varphi_1^1) =$$
$$= x_0 x_1 \overline{\varphi}_0^1 \overline{\varphi}_1^1 + x_0 \overline{x}_1 \varphi_0^1 \overline{\varphi}_1^1 + x_0 \overline{x}_1 \overline{\varphi}_0^1 \varphi_1^1 + x_0 x_1 \varphi_0^1 \varphi_1^1 + x_0 x_2 \varphi_1^1$$

Ω_{φ}^{2}	0	1	2	3
Ω_{z}				
0	0	0	0	0
1	0	0	1	1
2	0	0	0	0
3	1	1	0	0
4	0	0	0	0
5	0	1	1	1
6	0	0	0	0
7	1	1	0	1



$$\Phi(x_0, x_1, x_2, \varphi_0^2, \varphi_1^2) = x_0 x_1 \overline{\varphi}_1^2 + x_0 \overline{x}_1 \varphi_1^2 + x_0 x_2 \varphi_0^2$$

$$\Phi(x_0, x_1, x_2, \varphi_0^1, \varphi_1^1) =$$
$$= x_0 x_1 \overline{\varphi}_0^1 \overline{\varphi}_1^1 + x_0 \overline{x}_1 \varphi_0^1 \overline{\varphi}_1^1 + x_0 \overline{x}_1 \overline{\varphi}_0^1 \varphi_1^1 + x_0 x_1 \varphi_0^1 \varphi_1^1 + x_0 x_2 \varphi_1^1$$

Классификация и свойства булевых функций

- Монотонные функции
- Самодвойственные функции
- Симметрические функции
- Линейные функции
- Функциональная полнота системы логических функций

I. Монотонные функции

Булева функция $F(x,Y)$ называется ***изотонной*** относительно переменной x, если из $x_1 > x_2$ следует $F(x_1,Y) \geq F(x_2,Y)$, или $F(1,Y) \geq F(0,Y)$.

$$F(x,Y) = \bar{x}f_1(Y) + xf_2(Y)$$

Если $f_2(Y) \geq f_1(Y)$, то $f_2(Y) = f_1(Y) + f_3(Y)$.

$$F(x,Y) = \bar{x}f_1(Y) + x\{f_1(Y) + f_3(Y)\} =$$

$$= (x + \bar{x})f_1(Y) + xf_3(Y) =$$

$$= f_1(Y) + xf_3(Y).$$

Изотонная функция относительно переменной x может быть приведена к виду, в котором x встречается только в прямом виде.

Булева функция $F(x, Y)$ называется ***антитонной*** относительно переменной x, если из $x_1 > x_2$ следует $F(x_1, Y) \le F(x_2, Y)$, или $F(1, Y) \le F(0, Y)$.

$$F(x, Y) = \bar{x} f_1(Y) + x f_2(Y)$$

Если $f_1(Y) \ge f_2(Y)$, то $f_1(Y) = f_2(Y) + f_3(Y)$.

$$F(x, Y) = \bar{x}\{f_2(Y) + f_3(Y)\} + x f_2(Y) =$$

$$= (x + \bar{x}) f_2(Y) + \bar{x} f_3(Y) =$$

$$= f_1(Y) + \bar{x} f_3(Y).$$

Антитонная функция относительно переменной x может быть приведена к виду, в котором эта переменная встречается только в инверсном виде.

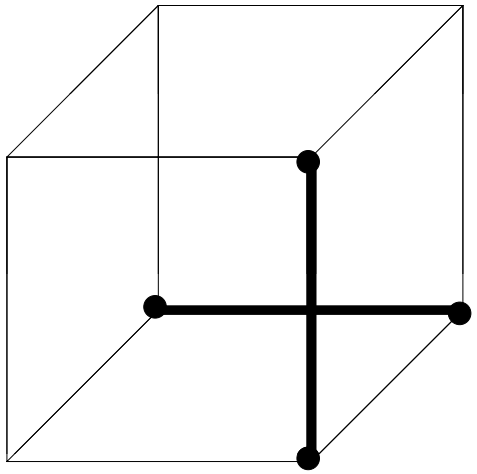
Булева функция $F(x, Y)$ называется ***монотонной*** относительно переменной ***x,*** если эта функция ***изотонна*** или ***антитонна*** относительно этой переменной.

Булева функция называется ***изотонной***, если она *изотонна* относительно всех ее переменных.

Булева функция называется ***антитонной***, если она *антитонна* относительно всех ее переменных.

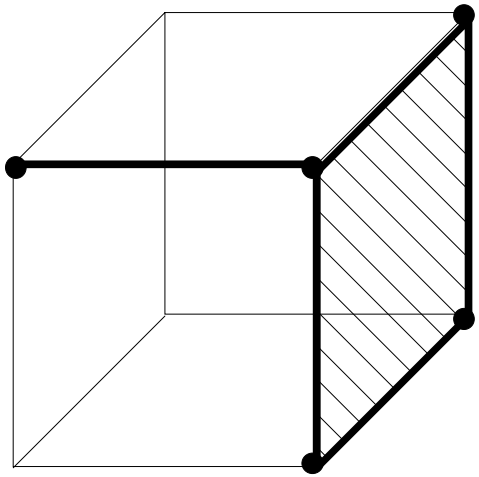
Булева функция называется ***монотонной***, если она *монотонна* относительно всех ее переменных (*изотонна* относительно некоторой части ее переменных и *антитонна* относительно остальных переменных).

Пример:



$$F(x_0, x_1, x_2) = x_0 x_1 + \bar{x}_0 \bar{x}_2$$

Функция является
изотонной относительно x_1 ,
антитонной относительно x_2 ,
немонотонной относительно x_0 .



$$F(x_0, x_1, x_2) = x_0 x_2 + x_1$$

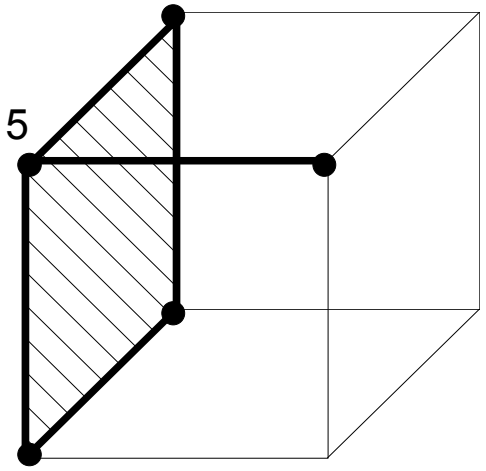
Изотонная функция

Монотонная функция является *“звездой”*.

Монотонная функция монотонна относительно вершины звезды.

Изотонная функция изотонна относительно вершины 2^n-1.

Антитонная функция антитонна относительно вершины 0.



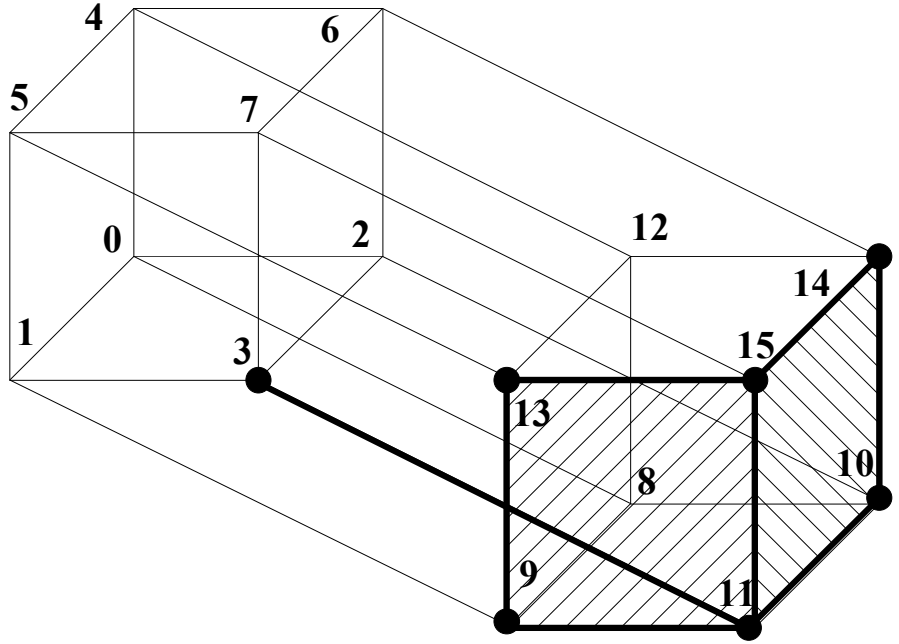
$$F(x_0, x_1, x_2) = x_0 x_2 + \overline{x}_1$$

Функция монотонна относительно вершины 5.

Теорема:

Для монотонных функций сокращенная форма является минимальной.

Доказательство: Предположим, что функция $F(X)$ монотонна и некоторый простой элемент $\boldsymbol{E_m}$ ее покрытия не является существенным.



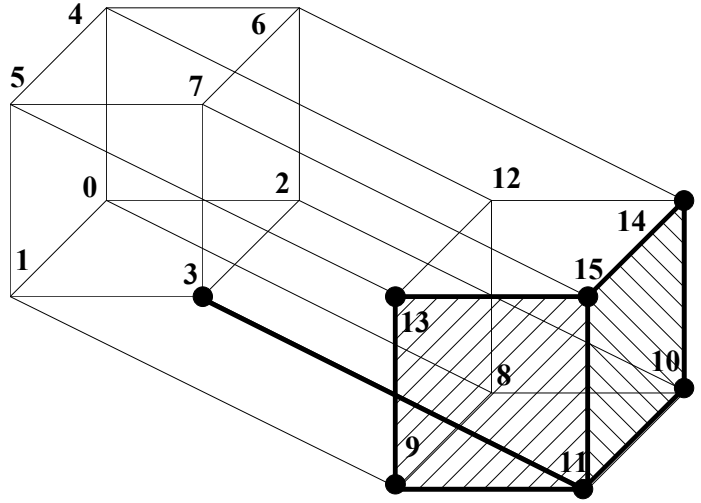
$$F(x_0, Y) =$$

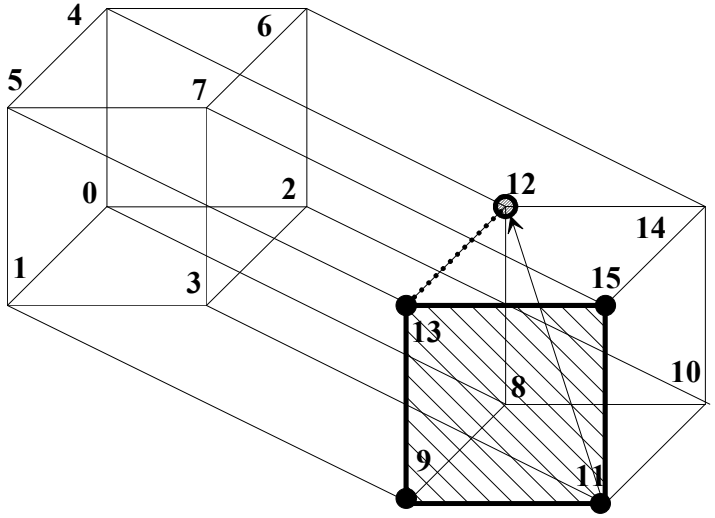
$$= x_0 x_3 + x_1 x_3 + x_0 x_1 \overline{x}_2$$

Тогда для вершины $T_k \in E_m$, принадлежащей максимальной диагонали элемента E_m, противоположной вершиной которой является вершина звезды, существует вершина T_g такая, что:

- T_g является соседней T_k по некоторому направлению j, внешнему для E_m, и
- $F(T_g)=1$.

Однако, как следует из определения монотонности, координата j должна быть внутренней для E_m и в этом случае E_m не является простым элементом. Таким образом, наше предположение не верно.

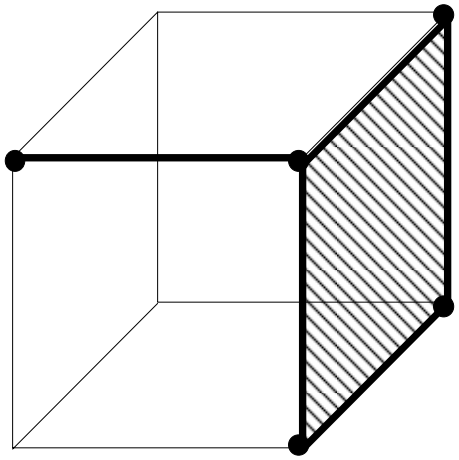




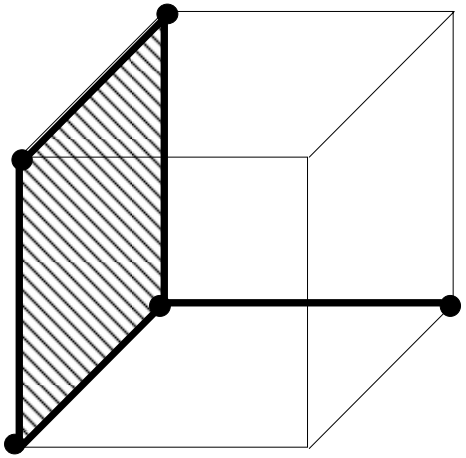
II. Двойственные и самодвойственные функции

Функция $F(X)$ называется ***двойственной*** функции $\Phi(X)$, если

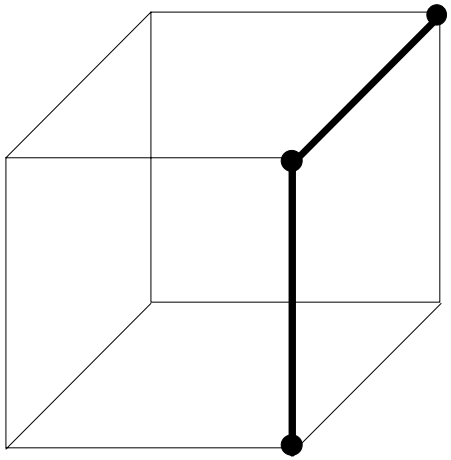
$$F(X) = \overline{\Phi(\overline{X})}.$$



$\Phi(X)$



$\Phi(\overline{X})$



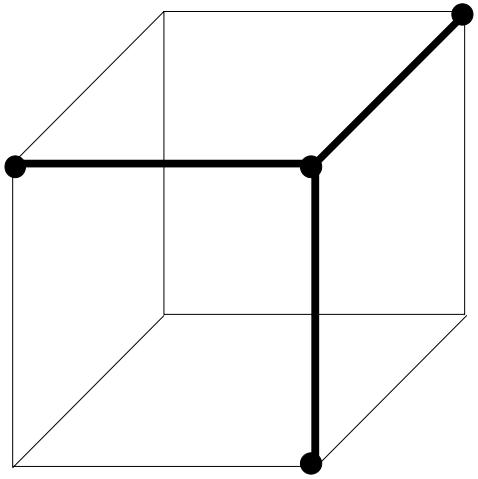
$F(X) = \overline{\Phi(\overline{X})}$

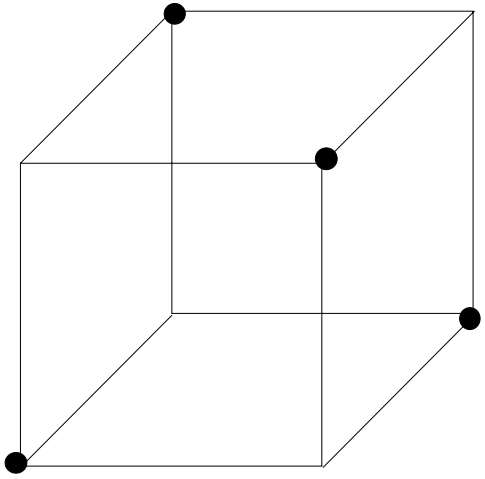
Функция $F(X)$ называется самодвойственной, если

$$F(X) = \overline{F(\overline{X})}.$$

Не существует самодвойственных функций двух переменных. Существуют два NPN класса самодвойственных функций трех переменных:

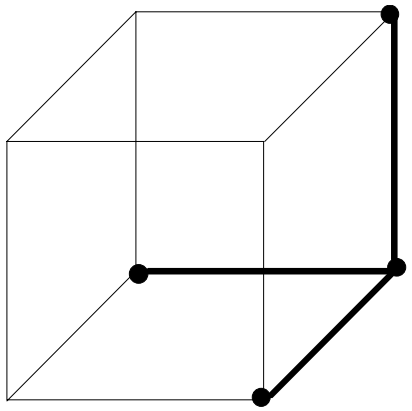
$$F_1(x_0, x_1, x_2) = x_0x_1 + x_0x_2 + x_1x_2;$$
$$F_2(x_0, x_1, x_2) = x_0 \oplus x_1 \oplus x_2.$$

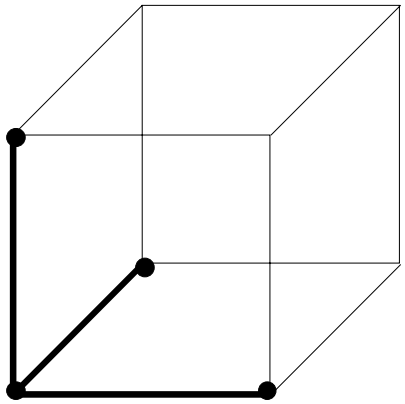


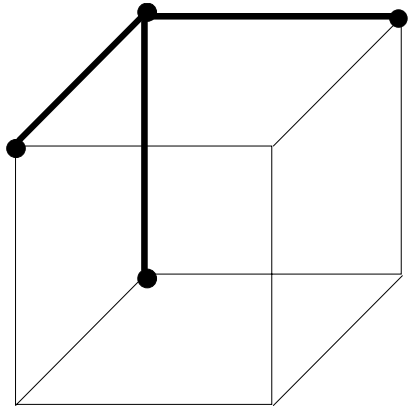


Самодвойственная функция имеет противоположные значения в двух вершинах, принадлежащих противоположным концам каждой наибольшей диагонали n-куба. Поэтому самодвойственная функция имеет значение 1 (или 0) на половине вершин n-куба.

Если некоторая функция является самодвойственной, то все функции, принадлежащие NPN классу также самодвойственны.







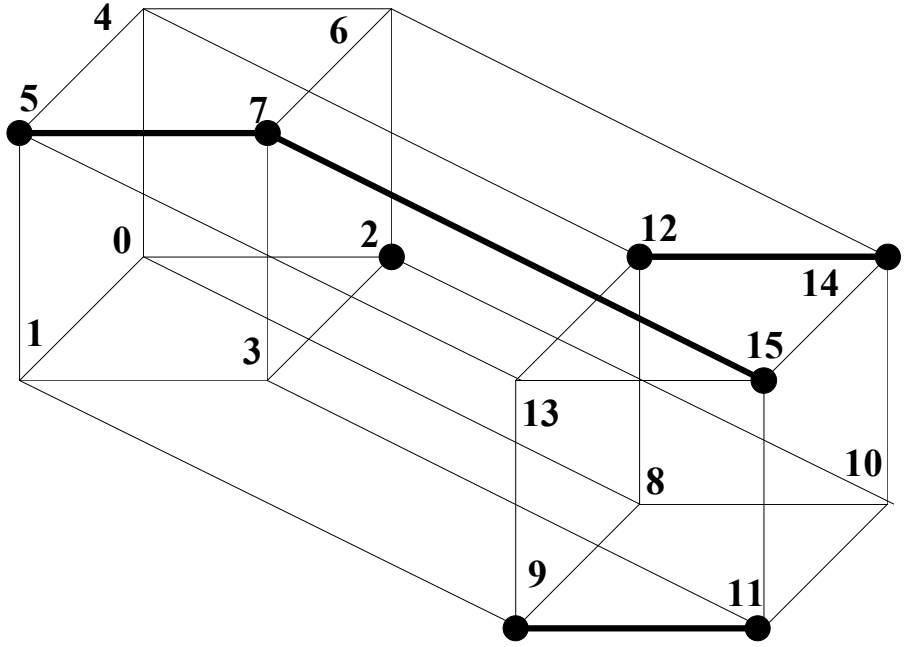
Существует четыре NPN класса самодвойственных функций четырех переменных:

$$F_1(x,y,z,t) = xyz + t(x+y+z);$$

$$F_2(x,y,z,t) = xyz + \bar{x}\bar{y}z + x\bar{y}t + \bar{x}yt;$$

$$F_3(x,y,z,t) = xyzt + \bar{x}\bar{y}\bar{z}t + (x\bar{y} + \bar{x}z)t;$$

$$F_4(x,y,z,t) = xyz + xz\bar{t} + x\bar{z}t + \bar{x}zt + \bar{x}y\bar{z}\bar{t}.$$



F_4

Из каждой несамодвойственной функции n переменных f(Y) может быть построена самодвойственная функция (n+1) переменной.

Пусть функция $F(x,Y) = xf(Y) + \bar{x}\varphi(Y)$ самодвойственная. Тогда

$$F(x,Y) = \overline{F(\bar{x},\bar{Y})} = \overline{\bar{x}f(\bar{Y}) + x\varphi(\bar{Y})} = (x + \overline{f(\bar{Y})})(\bar{x} + \overline{\varphi(\bar{Y})}) =$$
$$= (x + f^d(Y))(\bar{x} + \varphi^d(Y)) = x\varphi^d(Y) + \bar{x}f^d(Y) + f^d(Y)\varphi^d(Y) =$$
$$= x\varphi^d(Y) + \bar{x}f^d(Y) + f^d(Y)\varphi^d(Y)(x + \bar{x}) =$$
$$= x(\varphi^d(Y) + f^d(Y)\varphi^d(Y)) + \bar{x}(f^d(Y) + f^d(Y)\varphi^d(Y)) =$$
$$= x\varphi^d(Y) + \bar{x}f^d(Y).$$

$$xf(Y) + \bar{x}\varphi(Y) = x\varphi^d(Y) + \bar{x}f^d(Y) \text{ and } f(Y) = \varphi^d(Y)$$

Поэтому, если $f(Y)$ – любая несамодвойственная функция, то $F(x,Y) = xf(Y) + \bar{x}\varphi(Y)$ – самодвойственная функция. Если $f(Y)$ – самодвойственная функция, то $F(x,Y) = xf(Y) + \bar{x}f(Y) = f(Y)$ не зависит от переменной x.

Используя подстановки $x_i = 1$ or $x_i = x_j$, из представителей NPN классов всех самодвойственных функций n+1 переменной можно получить представителей всех NPN классов функций n переменных

Например, для самодвойственной функции

$F_4(x, y, z, t) = xyz + xz\bar{t} + x\bar{z}t + \bar{x}zt + \bar{x}y\bar{z}\bar{t}$:

$F_4(1, y, z, t) = yz + z\bar{t} + \bar{z}t;$

$F_4(x, 1, z, t) = xz + xz\bar{t} + x\bar{z}t + \bar{x}zt + \bar{x}\bar{z}\bar{t} = xz + xt + zt + \bar{x}\bar{z}\bar{t};$

$F_4(x, y, 1, t) = xy + x\bar{t} + \bar{x}t;$

$F_4(x, y, z, 1) = xyz + x\bar{z} + \bar{x}z = xy + x\bar{z} + \bar{x}z;$

$F_4(x, x, z, t) = xz + xz\bar{t} + x\bar{z}t + \bar{x}zt = xz + xt + zt;$

III. Симметрические функции

Функция F(X) называется симметрической, если для любых двух переменных x_i *и* x_j

$$F(x_i, x_j, Y) = F(x_j, x_i, Y).$$

Пример:

$xy+xz+yz = yx+yz+xz$ - симметрическая функция;

$xz + y \neq yz + x$ - несимметрическая функция.

Элементарные симметричекие функции

Для элементарных симметрических функций:

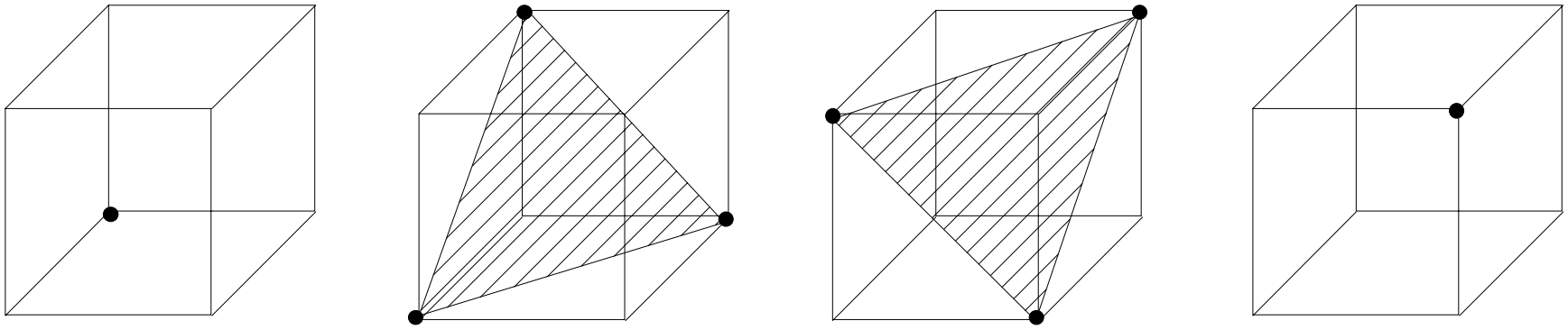
- *любые перестановки переменных не изменяют функцию;*
- *фукция зависит только от числа единиц в векторе значений переменных, на котором функция равна единице, и не зависит от конфигурации единиц и нулей в векторе;*
- *число единиц в векторе значений переменных, на котором функция равна единице, называется* ***рабочим числом симметрической функции.***
- *для симметрических функций n переменных существуют* $n+1$ *различных рабочих чисел* $(0,1,\ldots,n-1,n)$.

Элементарная симметрическая функция принимает значение единицы на одном и только одном рабочем числе.

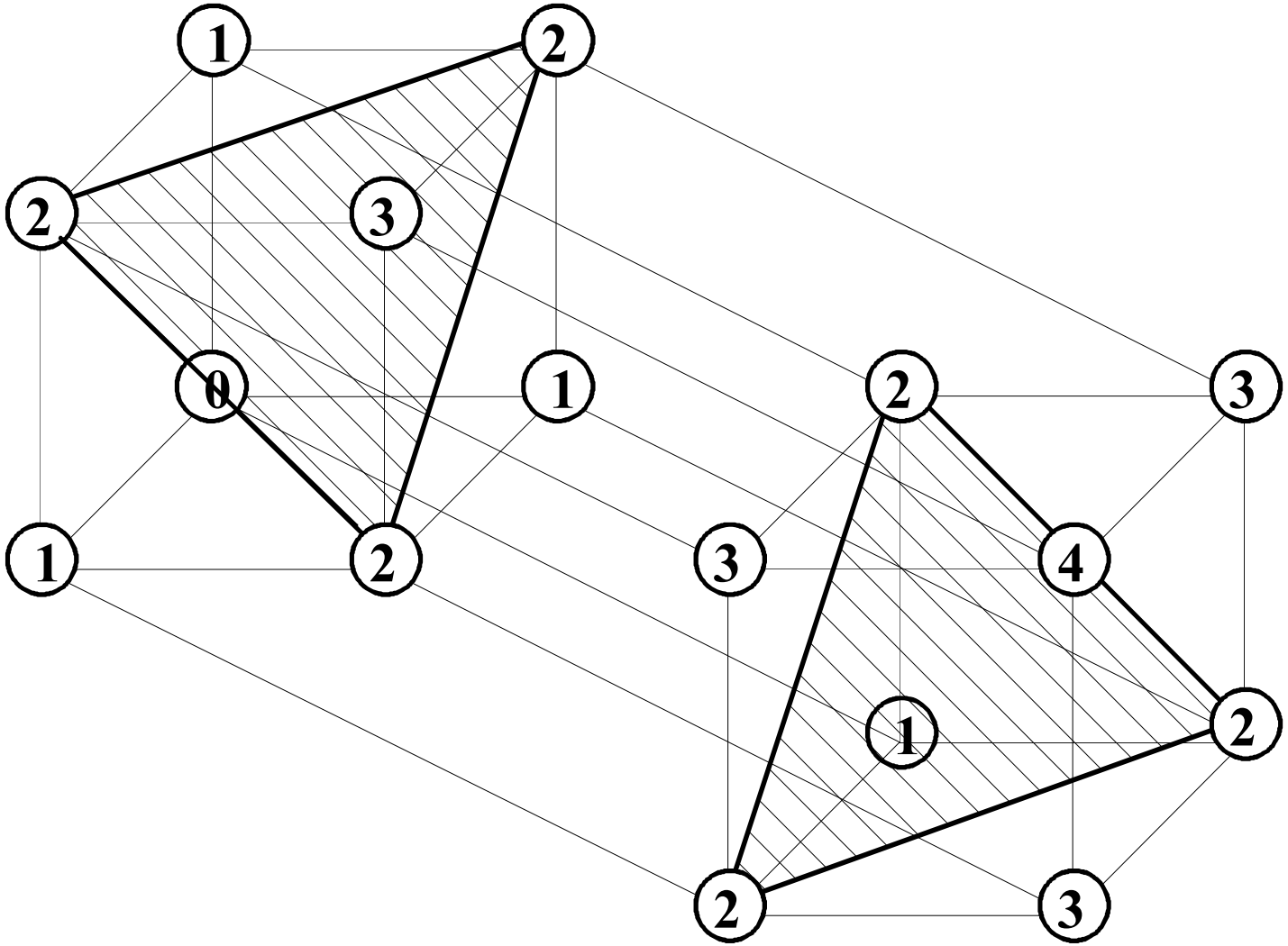
Симметрические функции трех переменных:

$$S_0^3 = \bar{x}\bar{y}\bar{z}; \quad S_1^3 = x\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}z;$$

$$S_2^3 = xy\bar{z} + x\bar{y}z + \bar{x}yz; \quad S_3^3 = xyz;$$



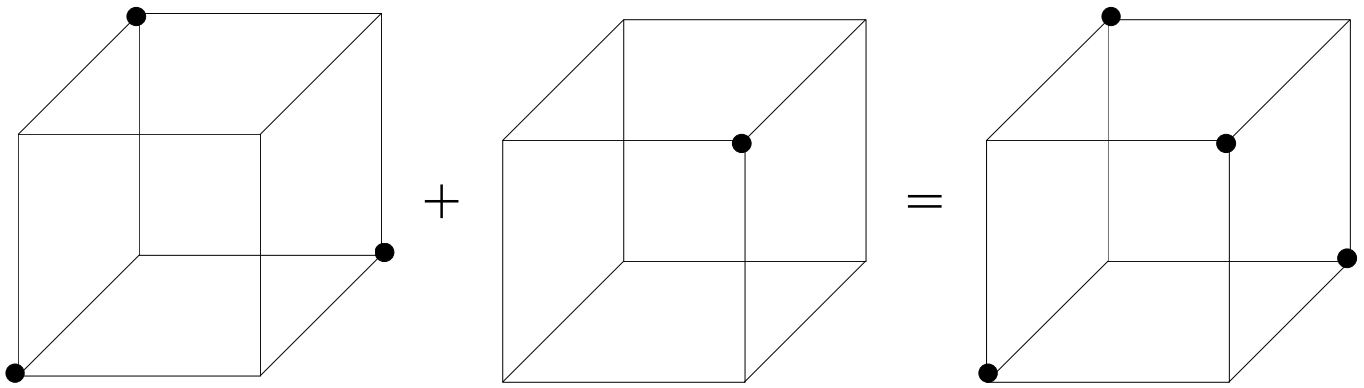
Расстояния по Хеммингу вершин гиперкуба от нулевой вершины



Произвольная симметрическая функция опрелеляется множеством рабочих чисел:

$$S^n_{k_1,k_2,\ldots,k_m} = \sum_{j=1}^{m} S^n_{k_j}.$$

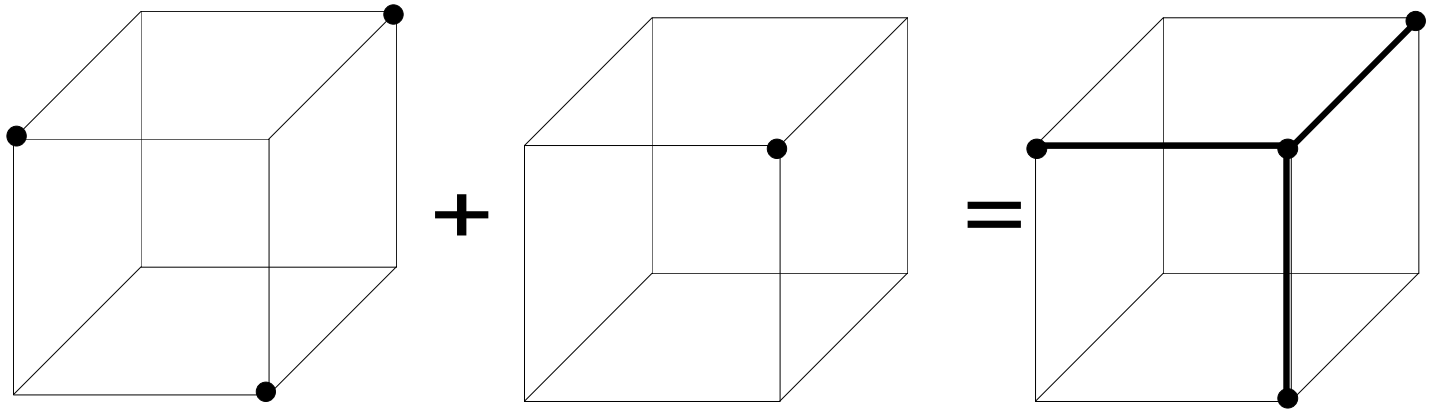
Пример: $S^3_{1,3} = S^3_1 + S^3_3 = x\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}z + xyz$



Монотонная симметрическая функция определяется одним рабочим числом:

$$M_k^n = \sum_{j=k}^{n} S_j^n .$$

Пример: $M_2^3 = S_2^3 + S_3^3 = xy + xz + yz$



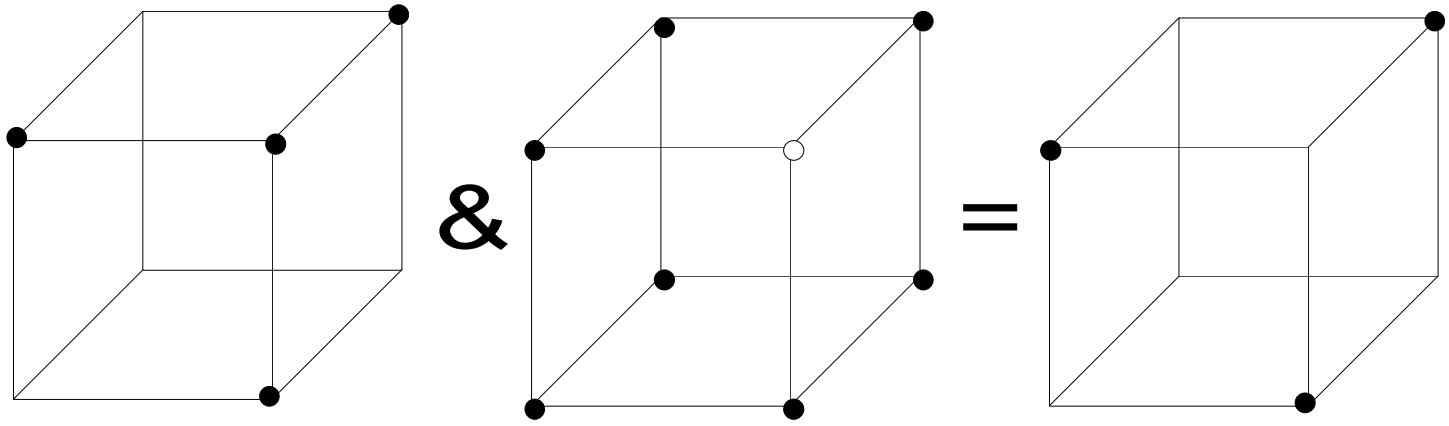
С другой стороны:

$$S_k^n = M_k^n \;\&\; \overline{M_{k+1}^n}.$$

Пример: $S_2^3 = M_2^3 \;\&\; \overline{M_3^3} = (xy + xz + yz)\overline{xyz} =$

$$= (xy + xz + yz)(\bar{x} + \bar{y} + \bar{z}) =$$

$$= xy\bar{z} + x\bar{y}z + \bar{x}yz$$



Каждая элементарная симметрическая функция n переменных может быть представлена в виде композиции элементарных симметрических функций, зависящих от m и n-m переменных:

$$S_k^n = \sum_{j=0}^{k} S_j^m \;\&\; S_{k-j}^{n-m}.$$

Пример:

$$S_2^4 = S_2^2 \,\&\, S_0^2 + S_1^2 \,\&\, S_1^2 + S_0^2 \,\&\, S_2^2 =$$

$$= xy\bar{z}\bar{t} + (x\bar{y} + \bar{x}y)(z\bar{t} + \bar{z}t) + \bar{x}\bar{y}zt$$

Каждая монотонная симметрическая функция n переменных может быть представлена композицией монотонных симметроических функций, зависящих от m и n-m переменных:

$$M_k^n = \sum_{j=0}^{k} M_j^m \;\&\; M_{k-j}^{n-m}.$$

Пример:

$$M_2^4 = M_2^2 \,\&\, M_0^2 + M_1^2 \,\&\, M_1^2 + M_0^2 \,\&\, M_2^2 =$$

$$= xy + (x+y)(z+t) + zt$$

Полиномиальное представление логических функций (полином Жегалкина)

Теорема: ***Произвольная логическая функция n переменных может быть представлена в полиномиальной канонической форме, как***

$$F(x_1, x_2, ..., x_n) = a_0 \oplus (a_1 x_1 \oplus a_2 x_2 \oplus ... \oplus a_n x_n) \oplus (a_{1,2} x_1 x_2 \oplus a_{1,3} x_1 x_3 \oplus ... \oplus a_{n-1,n} x_{n-1} x_n) \oplus ... \oplus a_{1,2,...,n} x_1 x_2 ... x_n, \text{ где } a_i \in \{0,1\}.$$

Алгоритм получения полиномиальной формы функции:

1. Представить функцию в канонической ДНФ и заменить все операции “+” на “⊕”:

$$F(X) = \sum_{\Omega=0}^{2^n-1} F(\Omega) X^{\Omega} = \bigoplus_{\Omega=0}^{2^n-1} F(\Omega) X^{\Omega}.$$

2. Заменить все $\overline{x}_i$ на $x_i \oplus 1$ во всех X^{Ω}.

3. Выполнить логические умножения и провести все поглощения.

Пример получения полиномиального представления функции:

$$F(X) = \underline{x_1} x_2 \vee x_2 x_3 = x_1 x_2 \overline{x}_3 \vee x_1 x_2 x_3 \vee \overline{x}_1 x_2 x_3 =$$
$$= x_1 x_2 \overline{x}_3 \oplus x_1 x_2 x_3 \oplus \overline{x}_1 x_2 x_3 =$$
$$= x_1 x_2 (1 \oplus x_3) \oplus x_1 x_2 x_3 \oplus (1 \oplus x_1) x_2 x_3 =$$
$$= x_1 x_2 \oplus x_1 x_2 x_3 \oplus x_1 x_2 x_3 \oplus x_2 x_3 \oplus x_1 x_2 x_3 =$$
$$= x_1 x_2 \oplus x_2 x_3 \oplus x_1 x_2 x_3$$

$$a_0 = a_1 = a_2 = a_{13} = 0 \qquad a_{12} = a_{23} = a_{123} = 1$$

IV. Линейные функции

Определение : Если логическая функция F может быть представлена в виде:

$$F = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \ldots \oplus a_n x_n,$$

где $a_i = 0$ или 1, то такая функция F называется линейной.

Существует 2^{n+1} линейных функций n переменных. Например, среди функций двух переменных $F(x,y)$ их восемь: $0, 1, x, x \oplus 1 = \bar{x}, y, y \oplus 1 = \bar{y}, x \oplus y, \bar{x} \oplus y = x \oplus \bar{y}$.

Если $a_0 = 0$ и $a_1 = a_2 = \ldots = a_n = 1$, то функция F является функцией четности n переменных: при нечетном числе переменных, принимающих значение 1, она равна 1 и 0 – при четном числе переменных, равных 1.

Линейная функция является либо ***самодвойственной*** функцией, либо ***самоантидвойственной*** функцией, т.е. $F(X) = \overline{F(\bar{X})}$ либо $F(X) = F(\bar{X})$.

Действительно,

$$\begin{aligned} F(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n) &= a_0 \oplus a_1(x_1 \oplus 1) \oplus a_2(x_2 \oplus 1) \oplus \ldots \oplus a_n(x_n \oplus 1) = \\ &= a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \ldots \oplus a_n x_n \oplus (a_1 \oplus a_2 \oplus \ldots \oplus a_n) = \\ &= F(x_1, x_2, \ldots, x_n) \oplus (a_1 \oplus a_2 \oplus \ldots \oplus a_n). \end{aligned}$$

Функциональная полнота (теорема Поста)

Определение: Пусть $F = \{f_1, f_2, ..., f_m\}$ – множество логических функций. Если произвольная логическая функция может быть реализована как комбинационная схема без обратных связей в виде суперпозиции функций этого множества, то такая система функций называется ***функционально полной***.

Определение: Функция, такая что $f(0,0,...,0) = 0$, называется функцией, сохраняющей 0. Функция, такая что $f(1,1,...,1) = 1$, называется функцией, сохраняющей 1.

Теорема: Пусть
M_0 – множество функций сохраняющих 0,
M_1 – множество функций сохраняющих 1,
M_2 – множество самодвойственных функций,
M_3 – множество монотонных функций,
M_4 – множество линейных функций,
тогда множество F является функционально полным, если оно содержит хотя-бы одну функцию, не принадлежащую каждому из этих классов функций.

Примеры функционально полных систем:

$$\{\overline{xy}\}, \{\overline{x \vee y}\}, \{xy, x \vee y, \bar{x}\}, \{\overline{xy \vee xz \vee yz}\} \text{ и др.}$$



Логическое Проектирование

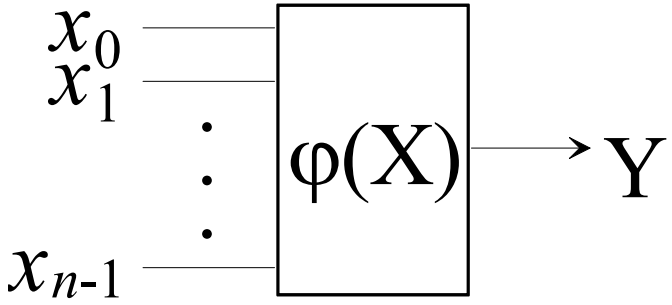
Лекция 6

Состязания в логических схемах

1.Введение

Схемы, реализующие булевы функции, называются ***комбинационными схемами***.

Для построения комбинационных асинхронных схем необходимо иметь в виду, что такие схемы должны взаимодействовать с окружающей средой в соответствии с некоторыми правилами.



Окружающая среда является источником входных сигналов (входных комбинаций) для схемы и приемником ее выходных сигналов. Для простоты будем рассматривать асинхронные комбинационные схемы только с одним выходным сигналом.

Набор значений входных переменных $x_0, x_1, \ldots, x_{n-1}$ будем обозначать n-мерным вектором $A = \{a_0, a_1, \ldots, a_{n-1}\}$.

Значение функции на этом наборе будем обозначать как $\varphi(A)$.

Статическое поведение комбинационной асинхронной схемы полностью определяется функцией $\varphi(X)$. Если среда в некоторый момент времени сформировала набор значений входных переменных $A = \{a_0, a_1, \ldots, a_{n-1}\}$, то после завершения переходных процессов (в стабильном состоянии) на выходе схемы должен появиться выходной сигнал, соответствующий значению $\varphi(A)$.

Изменение входного набора инициирует следующий шаг вычисления значения функции. Смену набора A набором B назовем ***переходом*** *A-B* , а последовательность переходов A-B-C-D-...-Z назовем ***траекторией переходов***.

Каждый переход в траектории происходит только после завершения переходного процесса в схеме, инициированного предыдущим переходом. Это ограничение определяет максимально возможную частоту переходов.

Естественно потребовать, чтобы каждый переход A-B удовлетворял следующим условиям:

- если $\varphi(A) = \varphi(B)$, то выходной сигнал не должен изменяться ни разу во время переходного процесса;
- если $\varphi(A) \neq \varphi(B)$, то выходной сигнал во время переходного процесса должен измениться только один раз.

Эти условия отражают требования к динамическому поведению асинхронной комбинационной схемы. Однако они не всегда удовлетворяются и на переходе A-B выходной сигнал схемы может изменяться несколько раз.

В случае, когда динамическое поведение асинхронной схемы не совпадает с ее статическим поведением, говорят о наличии ***состязаний*** (***hazards***).

Появление состязаний зависит от временных параметров элементов схемы и соединительных проводов, а также от вида функции и порядка следования переходов.

Сначала рассмотрим, какие переходы являются допустимыми (и следовательно, какие ограничения накладываются на внешнюю среду).

Затем рассмотрим модели реализаций асинхронных комбинационных схем, которые, при некоторых допущениях, позволяют строить схемы, свободные от состязаний.

2. Функциональные состязания

2.1 *Статические функциональные состязания*

Рассмотрим некоторые свойства булевых функций.

Определение 2.1. Пусть $A=\{a_0, a_1, \ldots, a_{n-1}\}$ и $B=\{b_0, b_1, \ldots, b_{n-1}\}$. Подкубом (A, B) перехода A-B является множество вершин, имеющих общие координаты с вершинами A и B, которые не изменяются на этом переходе.

Например,

A 1 0 1 0 0 Common coordinates: $x_2=1$, $x_4=0$

B 0 1 1 1 0

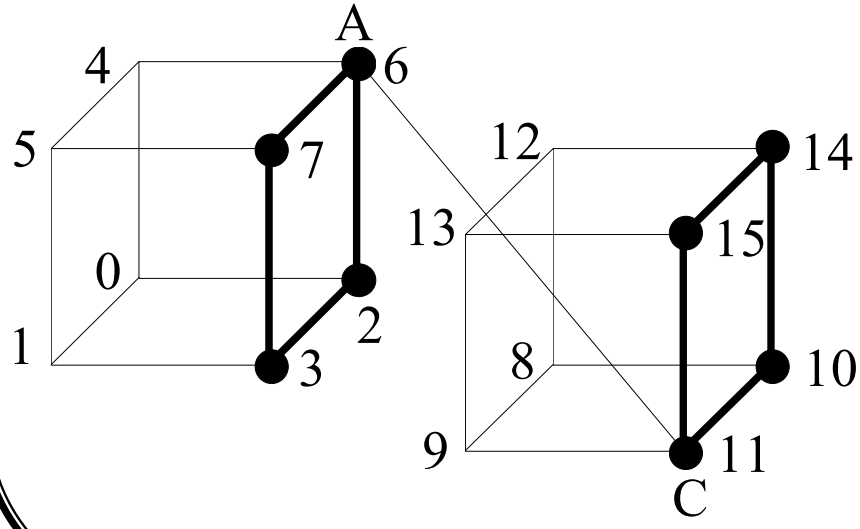
* * 1 * 0

Характеристическая функция подкуба перехода (***терм перехода***) имеет вид: $\omega(A,B) = x_2\bar{x}_4$.

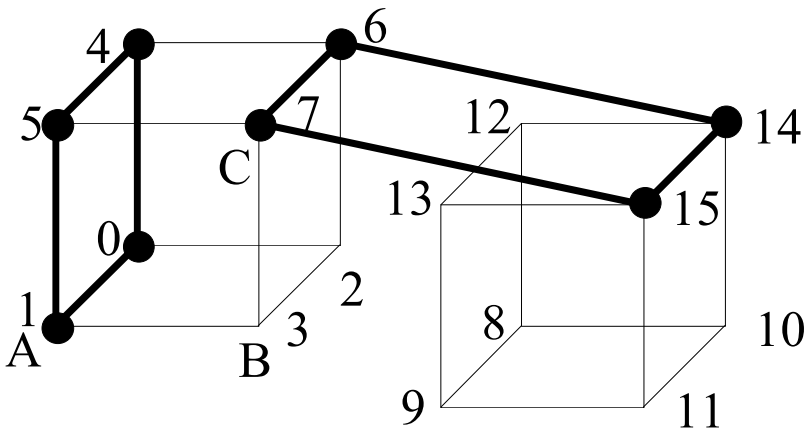
Если вершина C принадлежит подкубу перехода (A,B), то это будем обозначать как $C \in (A,B)$.

Определение 2.2. Говорят, что пара наборов A and C булевой функции φ имеет ***статические функциональные состязания*** (СФС), если выполняются два условия:

1) $\varphi(A) = \varphi(C)$;
2) *существует, по крайней мере, один набор* $B \in (A,C)$, такой что $\varphi(B) \neq \varphi(\mathrm{A})$.



Эта функция не имеет статических состязаний, так как на всех вершинах подкуба $\omega(A,C)$ она имеет постоянное значение $\varphi(\omega(A,C)) = \varphi(A) = \varphi(C) = 1$.



$$\varphi(\mathrm{X}) = \bar{x}_1\bar{x}_3 + x_1x_2$$

$$A = 1000,\ \ C = 1110$$

$$\omega(A, C) = x_0\bar{x}_3$$

То есть x_0=1, x_3=0. Подстановка этих значений в функцию дает

$$\varphi(1, x_1, x_2, 0) = \bar{x}_1 + x_2 \neq Const.$$

Это означает, что существует траектория *A*-*B*-*C* в подкубе (*A*,*C*), $B \in (A,C)$, and $\varphi(B) = 0$.

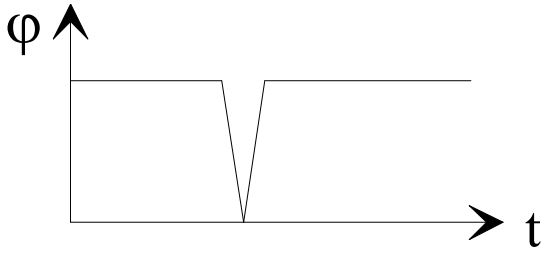
Следовательно, эта функция имеет статические состязания, поскольку их имеет, по крайней мере, одна пара наборов (*A* and *C*).

Теорема 2.1. Пара наборов A и B булевой функции, такой что $\varphi(A) = \varphi(B)$, свободна от статических функциональных состязаний, если выполняется одно из двух условий:
$\varphi(\omega(A,B) =1) \equiv 0$ или $\varphi(\omega(A,B) =1) \equiv 1$.

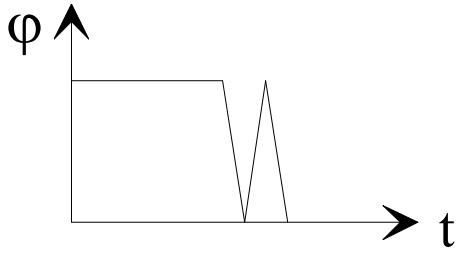
Доказательство непосредственно следует из условия: в подкубе перехода (A,B) значение функции φ не должно изменяться даже один раз , т.е. функция $\varphi(\omega(A,B) =1)$ должна быть константой.

2.2 Динамические функциональные состязания

Статические функциональные состязания



Динамические функциональные состязания



Определение 2.2. Говорят, что пара наборов A and D булевой функции φ имеет ***динамические функциональные состязания*** (ДФС), если выполняются два условия:

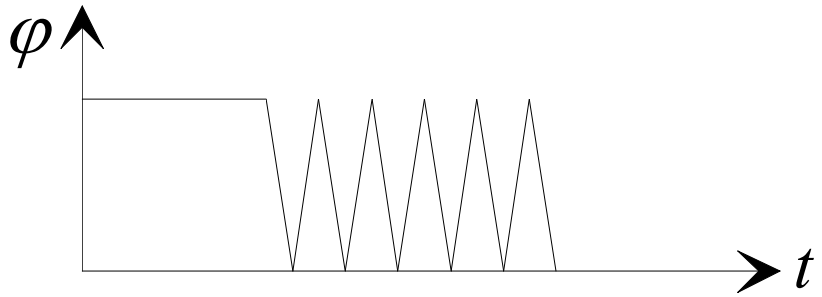
1) $\varphi(A) \neq \varphi(D)$;

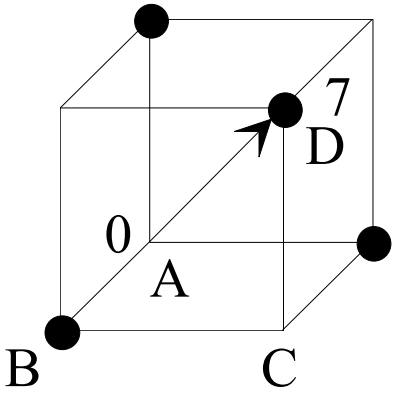
2) существуют, по крайней мере, два набора B и C, таких что

   a) $B \in (A,D)$ и $C \in (A,D)$ и

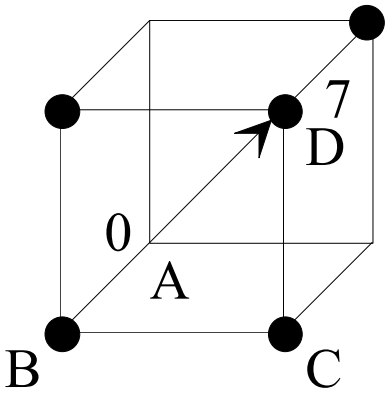
   b) $\varphi(B) = \varphi(D)$ *и* $\varphi(C) = \varphi(A)$.

Например, траектория A-B-C-D дает последовательность 1-0-1-0 значений функции. Вполне можно представить себе последовательность 1-0-1-0-...-1-0 значений функции на некоторой траектории переходов:





Нет ни одной траектории типа *A-B-C-D,* для которой отсутствовали бы функциональные состязания.



Эта функция не содержит динамических функциональных состязаний.
Заметим, что на переходе 000-111 значение функции изменяется монотонно на любой возможной траектории.

Определение 2.4. Говорят, что пара наборов *A* и *B* булевой функции имеет функциональные состязания (ФС), если она имеет либо СФС, либо ДФС.

Переход *A-B* называется ***допустимым***, если наборы *A* и *B* не имеют ФС (свободны от ФС).

Траектория (переходов) *A-B-C-...-Z* называется ***допустимой***, если каждый ее переход является допустимым.

Очевидно (это следует из определений 2.1-2.4), что ФС между наборами заданной булевой функции являются локальным свойством самой функции и не зависят от реализующей ее схемы и характеристик элементов этой схемы.

Тот факт, что пара наборов *A* и *B* имеет ФС, означает невозможность построения асинхронной схемы, реализующей заданную функцию без сбоев на переходе *A-B* (или обратном переходе *B-A*).

Теорема 2.2. В любой булевой функции, существенно зависящей от n переменных, при $n \geq 2$ существует пара наборов A и B, имеющих функциональные состязания.

Доказательство.

1. Если функция $\varphi(X)$ несамодвойственная, т.е.

$$\varphi(X) \neq \varphi^d(X) = \overline{\varphi(\overline{X})},$$

то существует, по крайней мере, одна пара противоположных наборов $A = \{a_0, a_1, \ldots, a_{n-1}\}$ и $B = \{\overline{a}_0, \overline{a}_1, \ldots, \overline{a}_{n-1}\}$, для которых $\varphi(A) = \varphi(B)$. Тогда, так как $n \geq 2$, то существует третий набор C, такой что $C \in (A,B)$ и $\varphi(C) = \overline{\varphi}(A)$. Следовательно, СФС тоже существуют.

2. Если функция φ самодвойственная, то либо n =1 (что противоречит условию теоремы), либо $n \geq 3$ (так как не существует самодвойственных функций двух переменных) и она может быть представлена как $\varphi = x_i\varphi(x_i = 1) + \overline{x}_i\varphi^d(x_i = 1)$, где $\varphi(x_i=1)$ – несамодвойственная функция, существенно зависящая от n-1 ≥ 2 переменных, и $\varphi^d(x_i = 1)$ – ее двойственная функция.

Таким образом, согласно первой части доказательства, каждая из подфункций $\varphi(x_i = 1)$ и $\varphi^d(x_i = 1)$ имеет пару противоположных наборов с ФС. Следовательно, в исходной самодвойственной функции имеется пара наборов $A = \{a_0, a_1, \ldots, a_{n-1}\}$ и $B = \{\overline{a}_0, \overline{a}_1, \ldots, \overline{a}_{n-1}\}$ с функциональными состязаниями.

Из теоремы 2.1 следует, что ***невозможно построить асинхронные схемы без состязаний*** в случае недопустимых переходов.

Следовательно, к внешней среде должны быть предъявлены очень жесткие требования: она должна формировать только допустимые траектории переходов.

2.3. Критерии отсутствия ДФС

Как следует из доказательства теоремы 2.2, когда $\omega(A,B) = 0$ (т.е. подкубом перехода является весь куб), любая пара противоположных наборов A и B произвольной функции n переменных ($n \geq 2$) имеет функциональные состязания (ФС).

Прежде всего, докажем теорему устанавливающую связь между ДФС и СФС.

Теорема 2.3 Наборы A и D булевой функции имеют динамические функциональные состязания тогда и только тогда, когда существуют статические функциональные состязания между наборами B and D, причем $B \in (A,D)$.

Доказательство.

Предположим, что существуют СФС наборов B и D, причем $B \in (A,D)$ и $\varphi(A) \neq \varphi(D)$. Тогда согласно определению 2.2, существует такой набор C, что $C \in (B,D)$ и $\varphi(C) \neq \varphi(D)$. Так как $\varphi(B) = \varphi(D)$ и $\varphi(C) = \varphi(A)$, то выполняются условия существования ДФС для наборов A и D.

Предположим, что существуют ДФС наборов A и D. Тогда по определению 2.3, существует такая пара наборов B и C, что $B \in (A,D)$, $C \in (B,D)$, $\varphi(B) = \varphi(D)$ и $\varphi(C) \neq \varphi(B)$. Следовательно, существуют СФС наборов B и D.

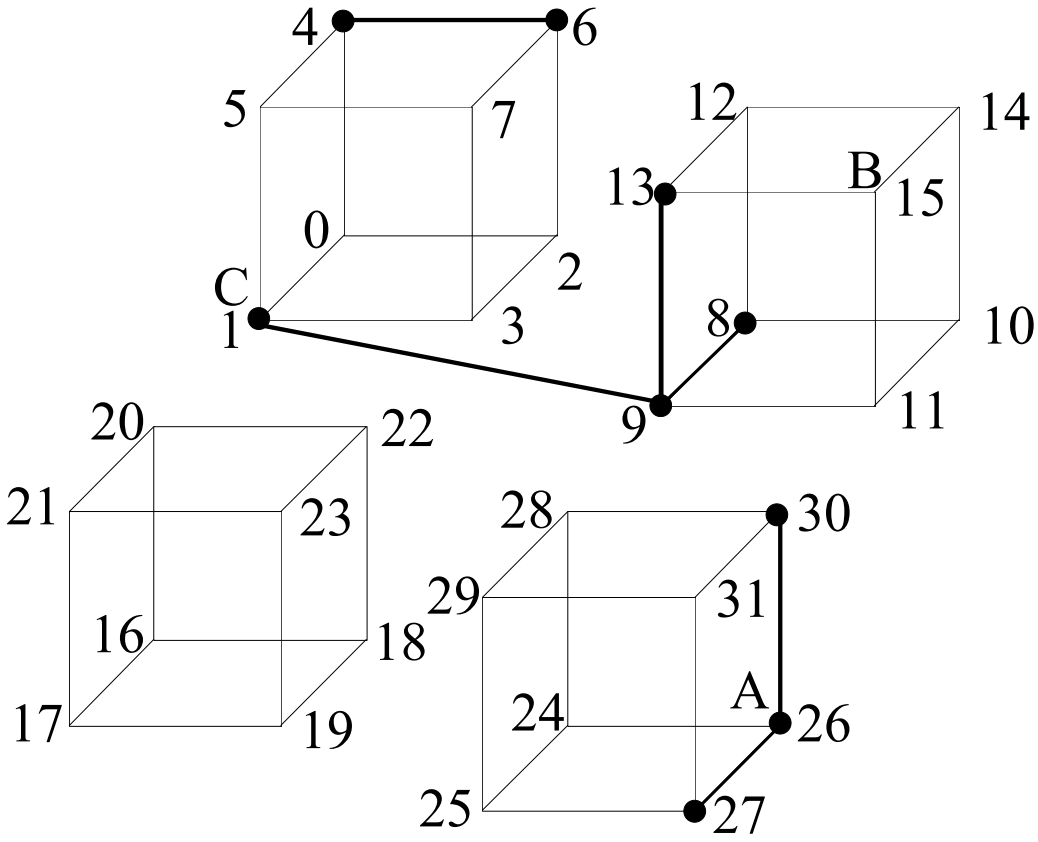
Теперь можно сформулировать критерий отсутствия ДФС.

Теорема 2.4 Пара наборов A и B функции $\varphi(X)$, для которой $\varphi(A) \neq \varphi(B)$ и $\omega(A,B) \neq 0$, свободна от динамических функциональных состязаний тогда и только тогда, когда выполняются два условия:

1) функция $\varphi(\omega(A,B)=1)$ является монотонной;
2) для любой переменной, от которой функция $\varphi(\omega(A,B)=1)$ зависит существенно, $\varphi(\omega(A,B)=1)$ и переход A-B являются одновременно либо изотонными, либо антитонными по этой переменной.

Попытайтесь доказать эту теорему самостоятельно. Подсказка: для доказательства необходимости следует использовать определения изотонных, антитонных и монотонных функций; достаточность доказывается от противного.

Рассмотрим пример.



$$\varphi(X) = x_0\overline{x}_1\overline{x}_2\overline{x}_4 + \overline{x}_0x_2\overline{x}_3\overline{x}_4 + x_0\overline{x}_1x_3\overline{x}_4 + \overline{x}_1\overline{x}_2x_3\overline{x}_4 + x_1\overline{x}_2x_3x_4 + \overline{x}_0x_1x_3x_4$$

Функция является немонотонной по любой из ее переменных, что может быть проверено непосредственно по заданной минимальной ДНФ, которая содержит как x_i, так и $\bar{x}_i$, $1 \le i \le 5$.

Рассмотрим переход (A=01011)-(11110=B) с подкубом перехода $\omega(A,B) = x_1 x_3 \neq 0$. Так как $\varphi(A) = 1$ и $\varphi(B) = 0$, переход A-B является антитонным по x_0 и x_2 и изотонным по x_4. Значение функции φ на этом переходе:

$$\varphi(\omega(A,B) = 1) = \varphi(x_1 = x_3 = 1) = \bar{x}_2 x_4 + \bar{x}_0 x_4 .$$

Эта функция является монотонной, так как она антитонна по x_0 и x_2 и изотонна по x_4. Поскольку и $\varphi(\omega(A,B) = 1)$, и переход A-B одновременно антитонны по x_0 и x_2 и изотонны по x_4, переход является допустимым, т.е. наборы A и B свободны от ДФС.

Теперь рассмотрим переход (B=11110)-(10000=C) с подкубом перехода $\omega(B,C) = x_0\bar{x}_4 \neq 0$. Так как $\varphi(B) = 0,\ \varphi(C) = 1$, переход является антитонным по x_1, x_2, и x_3.

Значение функции φ на переходе B-C:

$$\varphi(\omega(B,C) = 1) = \varphi(x_0 = 1, x_4 = 0) = \bar{x}_1\bar{x}_2 + \bar{x}_1x_3.$$

Эта функция является монотонной. Однако, так как она изотонна по x_3, в то время как переход антитонен по x_3, наборы B и C имеют ДФС. Действительно, переход B-C может, например, проходить по траектории B-13-5-C, на которой функция φ соответственно принимает значения 0-1-0-1, т.е. имеет место ДФС.

2.4 Классы допустимых траекторий переходов

Определение 2.5 Два набора $A=\{a_0, a_1,...,a_{n-1}\}$ и $B=\{b_0, b_1,...,b_{n-1}\}$ значений переменных функции n-переменных называются:

а) соседними, если они отличаются значением только одной переменной;

b) ***сравнимыми*** (обозначается как $A>B$ или $A<B$), если удовлетворяются неравенства только одно из двух типов $a_i>b_i$ или $a_k<b_k$ для любых i и k ($1\le i,k\le n,\ i\ne k$).

Заметим, что соседние набры всегда сравнимы (обратное утверждение в общем случае неверно);

Определение 2.6 Переход A-B называется ***соседним,*** если наборы A and B соседние и ***сравнимым,*** если наборы A и B сравнимы.

Траектория, в которой каждый переход соседний (сравнимый) называется траекторией с соседними (сравнимыми) переходами.

***Примеры**:*

Наборы 1000 и 1001 соседние.
Наборы 1000 и 1011 несоседние, но сравнимые.
Наборы 1000 и 0110 несравнимые.
Переход 1000-1001 соседний.
Переход 1000-1011 сравнимый.
Траектория 1000-1001-1011-1111 с соседними переходами.
Траектория 1001-1011-1111-1000 со сравнимыми переходами.

Теорема 2.5 Любые соседние переходы являются допустимыми.

Доказательство. Пусть наборы A and B произвольной булевой функции являются соседними. Тогда отсутствует набор C, такой что $C \in (A, B)$ и, согласно определениям 2.4 and 2.2, A and B свободны от функциональных состязаний.

Следствие. Класс траекторий с соседними переходами для произвольных булевых функций является допустимым.

Теорема 2.6 Для изотонных (антитонных) булевых функций, класс траекторий со сравнимыми переходами является допустимым.

Доказательство. Пусть A и B – произвольные сравнимые наборы изотонной функции, $\varphi(A) = \varphi(B)$ и $A>B$. Тогда существует набор C, такой что $C \in (A, B)$, $A>C$ и $C>B$. Так как φ – изотонная функция, $\varphi(A) \geq \varphi(C) \geq \varphi(B)$ и, следовательно, $\varphi(C) = \varphi(A)$. По определению 2.3, СФС для наборов A и B отсутствуют. Согласно теореме 2.3, отсутствие СФС для пары сравнимых наборов означает отсутствие ДСФ для любой пары сравнимых наборов. Такое же доказательство может быть проведено для антитонных функций.

Таким образом, если необходимо построить асинхронную схему, которая реализует изотонную (антитонную) булеву функцию, достаточно потребовать, чтобы окружающая среда формировала траектории из сравнимых наборов.

Любая булева функция имеет два набора, сравнимых со всеми другими наборами. Это наборы E_0=00...0 и E_1=11...1. Следовательно, можно было бы использовать подкласс траекторий со сравнимыми переходами, в которых любые наборы перемежаются с набором E_0 или E_1. Однако в этом случае реализуемые изотонные (антитонные) булевы функции должны быть произвольно определены на наборах E_0 или E_1.

Мы пришли к идеи искусственного сведения произвольных функций к изотонным или антитонным. Это можно сделать, представив функцию $\varphi(x_0,x_1,...,x_{n-1})$ как квази-монотонную функцию $2n$ переменных, а именно, $\varphi^*(x_0, \hat{x}_0, x_1, \hat{x}_1,..., x_{n-1}, \hat{x}_{n-1})$, где $\hat{x}_i = \overline{x}_i$.

Квази-монотонная функция имеет следующие свойства:

1) эта функция определена на 2^n наборах из 2^{2n} при условии, что все переменные являются существенными;
2) среди наборов, на которых она определена, нет сравнимых;
3) она является изотонной (антитонной) функцией относительно каждой из $2n$ переменных (согласно определению изотонных и антитонных функций).

Пример. Функция $\varphi(x_0, x_1) = x_0\bar{x}_1 + \bar{x}_0 x_1$ сведена к квази-монотонной функции $\varphi^*(x_0, \hat{x}_0, x_1, \hat{x}_1) = x_0\hat{x}_1 + \hat{x}_0 x_1$, представленной в таблице, в которой только первые 4 набора из 16 определены. В столбце «комментарий» они названы «рабочими».

Так как квази-монотонная функция φ^* не полностью определена, рассмотрим некоторые из ее доопределений в столбце F.

x_0	$\hat{x}_0$	x_1	$\hat{x}_1$	φ^*	F	Комментарий
0	1	0	1	0	0	Рабочие наборы
1	0	0	1	1	1	
0	1	1	0	1	1	
1	0	1	0	0	0	
1	1	1	1	*	1	Спейсер
1	1	1	0	*	1	Промежуточные наборы
1	1	0	1	*	1	
1	0	1	1	*	1	
0	1	1	1	*	1	
0	0	1	0	*	*	Запрещенные наборы
0	0	0	1	*	*	
1	0	0	0	*	*	
0	1	0	0	*	*	
0	0	0	0	*	*	
0	0	1	1	*	*	
1	1	0	0	*	*	

a) Наборы функции F , в которых $x_i \neq \hat{x}_i$ для всех i, $1 \leq i \leq n$, называются *рабочими*.

b) Набор E функции F , в котором $x_i = \hat{x}_i = 1$ для всех i, $1 \leq i \leq n$, называется *спейсером*.

c) Любой набор $C \neq E$, в котором, по крайней мере, для одного i, $1 \leq i \leq n$, $x_i = \hat{x}_i = 1$, называется *промежуточным*.

d) Любой набор, в котором, по крайней мере, для одного i, $1 \leq i \leq n$, $x_i = \hat{x}_i = 0$, называется *запрещенным*.

Нетрудно видеть, что для функции F:
- число рабочих наборов равно 2^n;
- число промежуточных наборов равно $3^n - 2^n - 1$;
- набор E сравним с любым рабочим набором.

Требование отсутствия функциональных состязаний накладывает ограничения на доопределения функции φ^* на промежуточных наборах. Например, если для рабочего набора A $F(A)=1$, для любого промежуточного набора $B>A$ должно быть $F(B)=1$.

Для рассматриваемого примера значение функции на всех промежуточных наборах должно быть равно единице. На запрещенных наборах функция может быть определена произвольно. Таким образом, заданная функция φ порождает семейство функций F.

Определение 2.7. Функция $F(x_1,\hat{x}_1,...,x_n,\hat{x}_n)$, которая является представлением исходной булевой функции $\varphi(x_1,...,x_n)$, заданной минимальной ДНФ, и получена из исходной путем замены $\bar{x}_i$ на $\hat{x}_i$ для всех i, $1\le i \le n$, называется ***супер-монотонной*** функцией.

Любой переход типа «рабочий набор R – спейсер E» (и обратный переход) в супер-монотонной функции является допустимым..

Определение 2.8. Траектория, в которой последовательные переходы являются переходами типа *R-E-R* или *E-R-E* называется траекторией с двухфазными переходами.

Для супер-монотонных функций траектории с двухфазными переходами являются допустимыми.

2.5. Организация внешней среды

Как различные классы допустимых траекторий влияют на организацию внешней среды?

Траектории с соседними переходами

Требование к среде генерировать только траектории с соседними переходами является слишком сильным и вызывает слишком большие трудности для организации среды. В общем случае необходимо избыточное кодирование входных переменных и эта избыточность может быть огромной.

Траектории со сравнимыми переходами

Использование траекторий со сравнимыми переходами формально накладывает менее жесткие ограничения на поведение среды. Однако практически организация среды все еще связана со значительными трудностями.

Каждый недопустимый переход в траекториях должен быть заменен отрезком траектории со сравнимыми переходами. Однако возможно, что некоторая пара рабочих наборов не имеет отрезка допустимой траектории. В этом случае необходимо использовать избыточное кодирование входных переменных.

Траектории с двухфазными переходами

Использование траекторий с двухфазными переходами, вероятно, обеспечивает наиболее простую организацию внешней среды.

Чередование спейсеров и рабочих наборов может быть организовано достаточно просто. Тогда вычисление значения супер-монотонной функции на произвольном переходе будет занимать два цикла.

2.6. Подведение итогов

1. Создание асинхронных схем, реализующих произвольные булевы функции с произвольными траекториями переходов входных наборов принципиально невозможно из-за статических и динамических функциональных состязаний.

2. Критерии отсутствия СФС и ДФС для пары наборов установлены теоремами 2.1 and 2.4 соответственно. Они сводятся к проверке является ли функция в подкубе переходов константой или монотонной относительно перехода.

3. Представлены три класса допустимых траекторий переходов:
- с соседними переходами (для произвольных функций);
- со сравнимыми переходами (для монотонных функций);
- с двухфазными переходами (для супер-монотонных функций).



Логическое Проектирование

Лекция 7

3. Логические состязания

Функциональные состязания – это свойство булевых функций, а ***логические состязания*** – свойство схемных реализаций булевых функций.

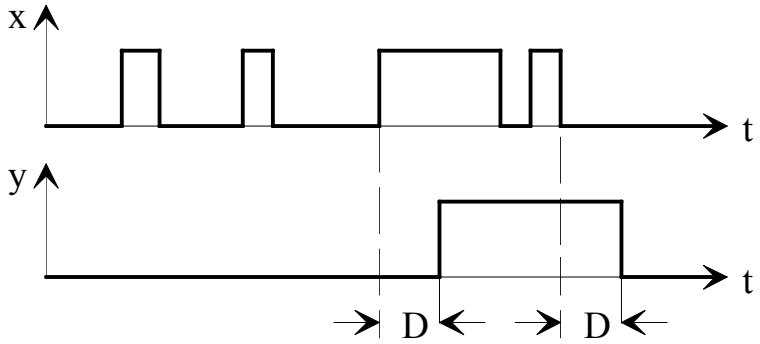
3.1. Задержки в асинхронных схемах

Сигналы появляются на выходах реальных элементов не моментально после смены входных сигналов, а с некоторой задержкой. Наряду с задержками самих элементов схемы существуют задержки сигналов в соединительных проводах.

Задержки элементов и задержки в проводах называются *паразитными,* так как они являются причиной нежелательного поведения схем. Следует отличать паразитные задержки от задержек специальных элементов, используемых для получения временного сдвига сигналов в схемах. Такие элементы называются ***инкорпорированными*** (***встроенными***) . Они могут быть ***чистыми*** и ***инерциальными***.

Чистые встроенные задержки сдвигают входной сигнал на временной интервал *D* без изменения его формы. Такие задержки очень трудно реализовать физически.

Поведение идеальной инерциальной задержки пояснено рисунком:



Из рисунка видно, что сигналы, длительность которых меньше, чем D, не проходят (фильтруются), в то время как сигналы длиннее, чем *D,* появляются на выходе элемента со сдвигом, равным *D*.

Анализ асинхронных схем на обнаружение сбоев (логических состязаний) и синтез схем, свободных от них, требуют различных предположений о соотношении величин задержек в элементах и проводах.

При проектировании схем, свободных от логических состязаний, обычно используют следующие гипотезы о природе задержек.

***Гипотеза* 1.** Паразитные задержки элементов инерциальные; задержки в проводах могут превышать задержки элементов; верхняя граница величин задержек не известна.

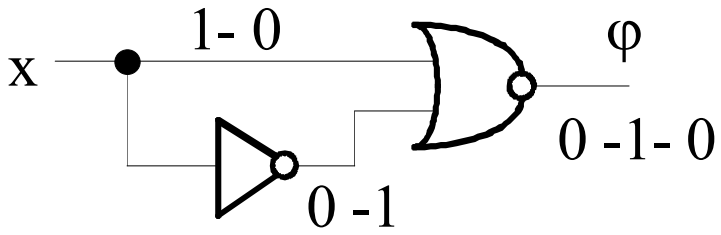
***Гипотеза* 2.** Задержки в проводах равны нулю; задержки элементов инерциальные; верхняя граница их значений не известна.

***Гипотеза* 3.** Возможные минимальные ($\tau_{\min}$) и максимальные ($\tau_{\max}$) значения задержек (чистых или инерциальных) элементов схемы известны.

Эти гипотезы являются основой некоторых структурных методов анализа асинхронных комбинационных схем на отсутствие логических состязаний.

3.2. Анализ логических состязаний

Рассмотрим простейшую схему:



Если предположить, что инвертор и NOR-элемент являются безынерциальными (с нулевыми задержками), то эта схема реализует функцию константы $\varphi = \overline{\overline{x} + x} = 0$, которая не содержит функциональных состязаний.

Если принять во внимание, что задержка в проводе, соединяющем вход x со входом NOR-элемента, меньше собственной задержки инвертора, то короткий сигнал, равный «1», может появиться на выходе схемы при переходе 1-0 переменной x. Этот эффект называется «*состязание в* 1». «*Состязание в* 0» также возможно в некоторых схемах (например, в схеме, двойственной рассмотренной).

Состязания обоих типов опасны, если устройство содержит память (триггер), которая является нагрузкой для комбинационной схемы, с малой постоянной времени на входе.

***Определение* 3.1.** Асинхронная схема имеет статические логические состязания (пики, выбросы) на переходе *A-B,* если
1) $\varphi(A) = \varphi(B)$;
2) наборы A и B не имеют СФС;
3) однократное состязание типа 0-1-0 или 1-0-1 может появиться на выходе схемы.

***Определение* 3.2**. Асинхронная схема имеет динамические логические состязания на переходе *A-B,* если
1) $\varphi(A) \neq \varphi(B)$;
2) наборы A и B не имеют ДФС;
3) многократные сбои типов 0-1-0-1 или 1-0-1-0 могут появиться на выходе схемы.

Как следует из определений, логические состязания могут появиться даже в случае, когда среда формирует допустимые траектории переходов. Попытаемся формализовать пункты 3 определений 3.1 и 3.2.

Булева троичная функция (*BTF*) – это функция троичных переменных. Все переменные и сама функция принимают значения из множества {0, ½, 1}. Значение ½ используется для представления значений переменных и функции в переходных процессах. Если все переменные функции определены на множестве {0,1}, то функция также определена на этом множестве.

BTF используют те же операции, что и булевы функции. Эти операции определяются следующим образом:

- инвертирование, $\bar{x} = 1 - x$;
- сложение, $x+y=max(x,y)$;

x \ y	0	½	1
0	0	½	1
½	½	½	1
1	1	1	1

- умножение, $x\&y=min(x,y)$;

x \ y	0	½	1
0	0	0	0
½	0	½	½
1	0	½	1

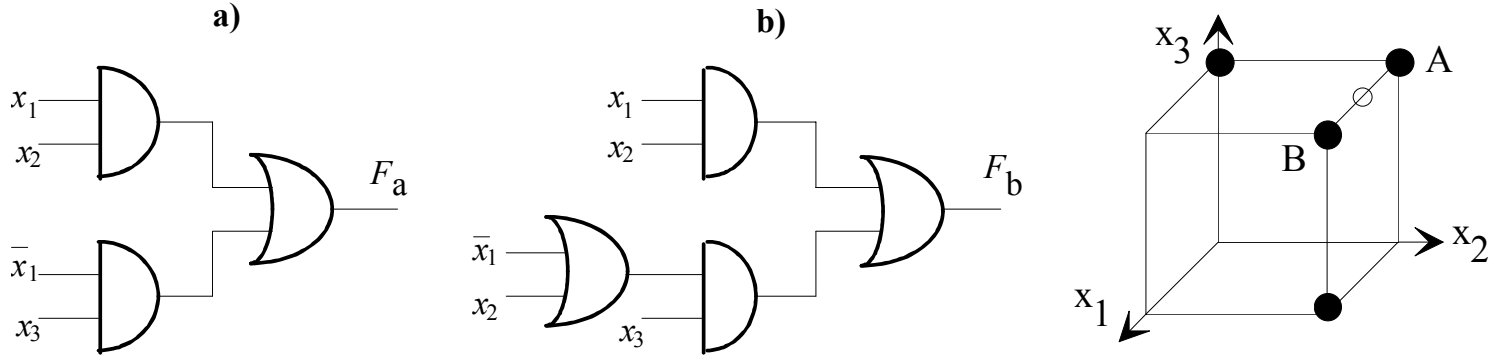
- сложение по модулю 2, $x \oplus y = \bar{x}y + x\bar{y} = \max(\min(\bar{x}, y), \min(x, \bar{y}))$

x \ y	0	½	1
0	0	½	1
½	½	½	½
1	1	½	0

Пусть переменная изменяет свои значения как 0-½-1 и 1-½-0, где ½ – неопределенное значение. В этом случае функционирование двоичной комбинационной схемы, построенной из элементов, может быть описано соответствующей *BT*-функцией. Тогда вместо статических логических состязаний в схеме можно искать статические функциональные состязания в ее *BT*-функции.

Теорема 3.1. Любая схема, реализующая логическую функцию, имеет статические логические состязания на переходе $A \to B$, если $\varphi(A) = \varphi(B)$, функция на этом переходе не имеет СФС, но соответствующая *BTF* на подкубе перехода может принимать значение ½ .

Пример. Рассмотрим две схемы, реализующую одну и ту же логическую функцию.



В стабильных состояниях поведение обеих схем одинаково. Однако их *BTF* различны, т.е.

$F_a = x_1x_2 + \bar{x}_1x_3 = \max(\min(x_1,x_2),\min(\bar{x}_1,x_3))$ в то время, как

$F_b = x_1x_2 + (\bar{x}_1 + x_2)x_3 = \max(\min(x_1,x_2),\min(\max(\bar{x}_1,x_2),x_3))$.

Заметим, что для наборов (011) и (111) выходы обеих схем равны «1», в то время как $F_a(\,½\ 1\ 1) = ½$, а $F_b(\,½\ 1\ 1) = 1$. Первая схема Содержит СЛС на переходе (011)→(111), а вторая схема не содержит СЛС на этом переходе.

Очевиден первый метод проектирования асинхронных схем, свободных от СЛС, который может быть применен в случае, если среда генерирует траектории с соседними переходами.

Любая пара соседних наборов $A=(a_1...a_{i-1}a_ia_{i+1}...a_n)$ и $B=(a_1...a_{i-1}\overline{a}_ia_{i+1}...a_n)$ соответствует переходному набору $A/B=(a_1...a_{i-1}$ ½ $a_{i+1}...a_n)$.

Статические логические состязания могут быть обнаружены по таблице истинности *BT*-функции для переходных наборов. Для функции, существенно зависящей от *n* переменных, достаточно вычислить ее значения на 3^n-2^n переходных наборах. *СЛС* появляются на переходах A→B, если $F(A/B)$ = ½.

Для рассматриваемого примера n=3, число переходных наборов не более, чем 27-8=19. Начнем строить таблицу истинности для *BT*-функций F_a и F_b.

x_1	x_2	x_3	F_a	F_b
½	0	0	0	0
½	1	0	½	½
½	0	1	½	½
½	1	1	½	1
…….			…….	

Нетрудно видеть, что обе функции имеют *СЛС,* несмотря на соседние переходы, которые обеспечивают отсутствие функциональных состязаний для любой функции.

Эта процедура обнаружения *СЛС* не имеет особого практического смысла при большом числе переменных. Существует более эффективная процедура, также основанная на *BTF* , но мы не будем здесь ее рассматривать.

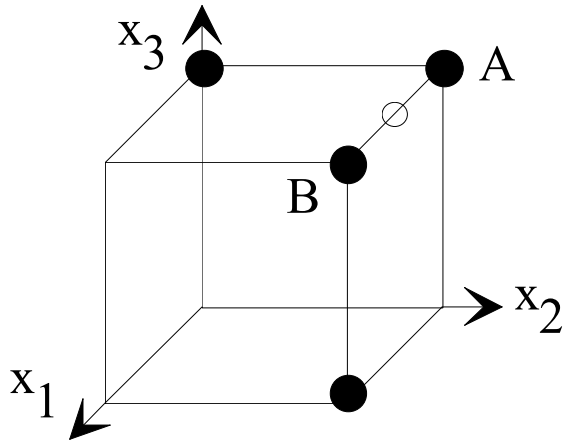
Формальная процедура, которая бы позволяла сформулировать условия отсутствия динамических логических состязаний, не известна. Для выяснения отсутствия *ДЛС* необходимо делать различные предположения об относительных скоростях переключения элементов схемы. Отметим лишь, что даже в случае соседних переходов в схеме, реализующей функцию, могут появиться ДЛС. Нетрудно построить соответствующих пример.

Теорема 3.2. Реализация любой переключательной функции, представленной в виде дизъюнкции всех простых импликант (в сокращенной форме) свободна от статических и динамических логических состязаний на соседних переходах.

Схема доказательства. Отсутствие *ДЛС* доказывается тривиально. Рассмотрим соседний переход $A \rightarrow B$ для наборов, на которых $\varphi(A)=1$ and $\varphi(B)=0$. Сигнал на выходе OR элемента не может измениться из 1 в 0 до тех пор, пока выходы всех AND элементов не станут равными 0. Но так как ни один из AND элементов не стремится переключить свой выходной сигнал в 1, не может возникнуть динамических логических состязаний. Подобным же образом можно доказать, что нет *ДЛС* в случае, когда $\varphi(A)=0$, $\varphi(B)=1$.

Доказательство отсутствия *СЛС* более сложно. Идея заключается в том, что если функция не содержит *СФС,* простые элементы, связанные с переходом, включают, по крайней мере, одну общую вершину. Поэтому, когда один простой импликант изменяет свое значение в 0, другой продолжает поддерживать значение функции, равное 1 (подхватывает).

Пример. Реализация функции

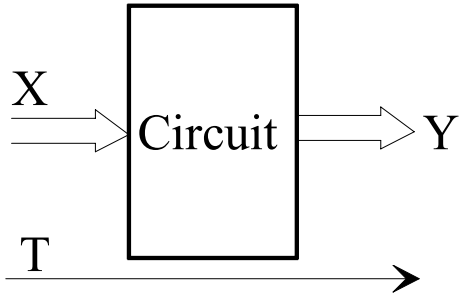


В форме $x_1x_2 + \bar{x}_1x_3 + x_2x_3$ свободна от логических состязаний на соседних переходах.

Заметим, что реализация функции в соответствии с совершенной дизъюнктивной нормальной формой (сумма произведений) имеет СЛС на соседних переходах.

3.3 Универсальные методы борьбы с состязаниями

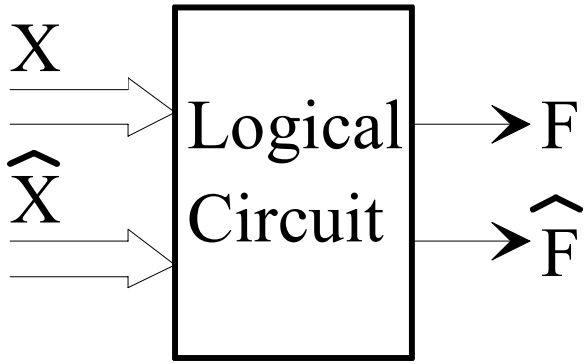
1. Введение синхронизации



Сигнал синхронизации T=1 разрешает запись значения выходного сигнала комбинационной схемы в элемент памяти. Во время интервала между двумя соседними сигналами T=1 все переходные процессы в комбинационной схеме должны завершиться. В этом случае все состязания будут маскированы.

2. Специальная дисциплина сметы входных наборов

Функция n переменных F и ее инверсия $\overline{F}$ представляются как квазимонотонные функции $2n$ переменных (супер-монотонные функции). Эти функции реализуются логической схемой с двумя выходами.



Пусть внешняя среда генерирует траектории с двухфазными переходами: $\ldots \to E \to R \to E \to R \to E \to R \to \ldots$, где E=00…00 является спейсером, а R – рабочим набором. Тогда все входные переменные принимают значение $x_i = \hat{x}_i = 0$ в фазе спейсера и значения $x_i \neq \hat{x}_i$ в рабочей фазе.

Переход от E к R и обратно является монотонным, следовательно состязания не могут появиться. Очевидно, что в конце фазы спейсера $F = \hat{F}$, а в конце рабочей фазы $F \neq \hat{F}$. Этот факт дает возможность фиксировать момент завершения переходных процессов в схеме.

Пороговая и мажоритарная логика

Функция $F(x_0, x_1, ..., x_{n-1})$ называется ***пороговой,*** если она может быть представлена в виде:

$$F(x_0, x_1, ..., x_{n-1}) = Sign(\alpha_0 x_0 + \alpha_1 x_1 + ... + \alpha_{n-1} x_{n-1} - \eta) = Sign(\sum_{j=0}^{n-1} \alpha_j x_j - \eta),$$

где α_j – вес входной переменной x_j, η – порог, $Sign(A) = \begin{cases} 1, & \text{если } A \geq 0 \\ 0, & \text{если } A < 0 \end{cases}$.

Пример: Трехвходовая мажоритарная функция (функция голосования)

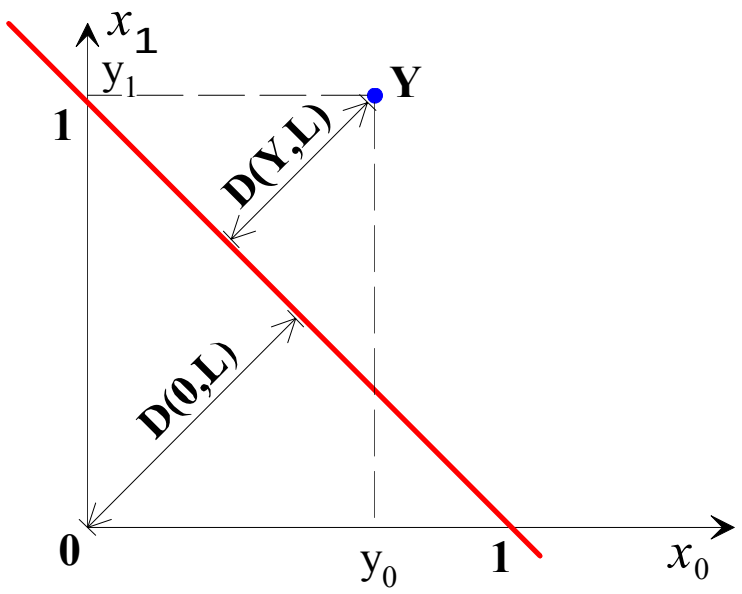
x_0	x_1	x_2	$x_0+x_1+x_2-2$	$Sign(x_0+x_1+x_2-2)$
0	0	0	-2	0
1	0	0	-1	0
0	1	0	-1	0
1	1	0	0	1
0	0	1	-1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

Гипергеометрическое представление пороговой функции

Уравнение $\sum_{j=0}^{n-1} \alpha_j x_j - \eta = 0$ является уравнением гиперплоскости в n-мерном пространстве.

Двумерное пространство:

$\alpha_0 x_0 + \alpha_1 x_1 - \eta = 0$ – уравнение прямой.



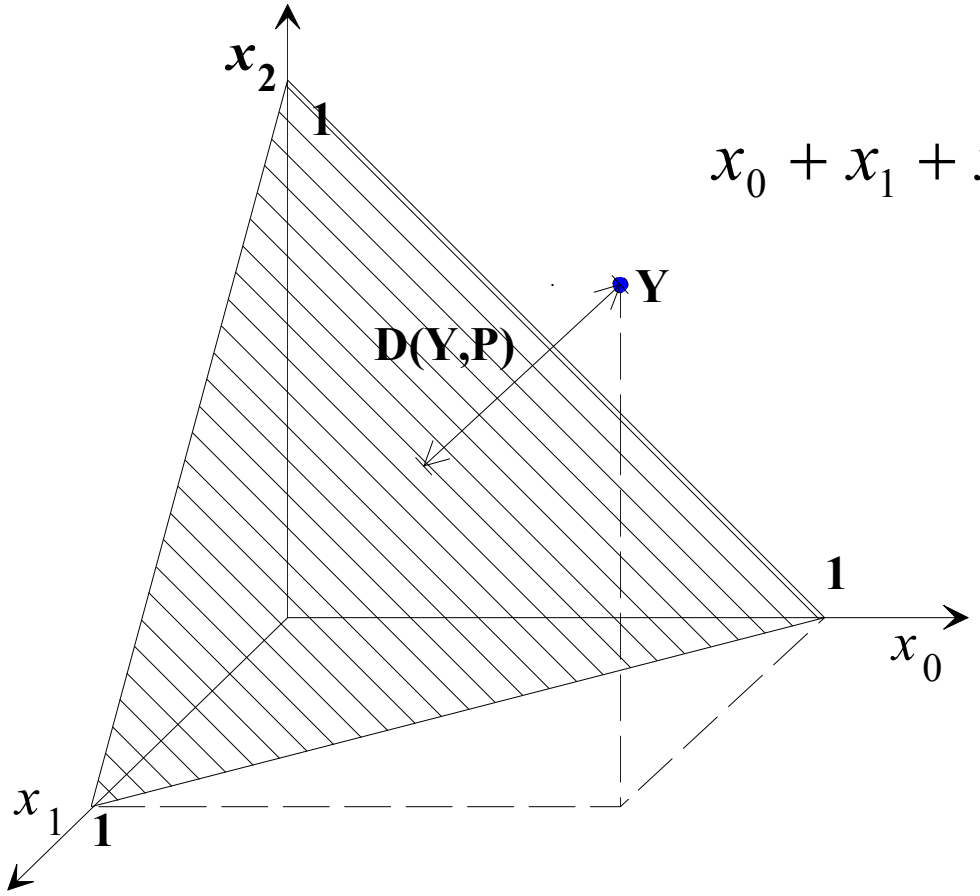
$$x_0 + x_1 - 1 = 0;$$

$$D(Y, L) = \frac{\alpha_0 y_0 + \alpha_1 y_1 - \eta}{\sqrt{\alpha_0^2 + \alpha_1^2}} = \frac{y_0 + y_1 - 1}{\sqrt{2}};$$

$$D(0, L) = \frac{-\eta}{\sqrt{\alpha_0^2 + \alpha_1^2}} = \frac{1}{\sqrt{2}};$$

Трехмерное пространство:

$\alpha_0 x_0 + \alpha_1 x_1 + \alpha_2 x_2 - \eta = 0$ – уравнение плоскости



$$x_0 + x_1 + x_2 - 1 = 0$$

$$D(Y,P) = \frac{\alpha_0 y_0 + \alpha_1 y_1 + \alpha_2 y_2 - \eta}{\sqrt{\alpha_0^2 + \alpha_1^2 + \alpha_2^2}} = \frac{y_0 + y_1 + y_2 - 1}{\sqrt{3}}$$

n-мерное пространство:

$\sum_{j=0}^{n-1} \alpha_j x_j - \eta = 0$ – уравнение гиперплоскости ***в n-***мерном пространстве

$$D(Y, P^n) = \frac{\sum_{j=0}^{n-1} \alpha_j y_j - \eta}{\sqrt{\sum_{j=0}^{n-1} \alpha_j^2}};$$

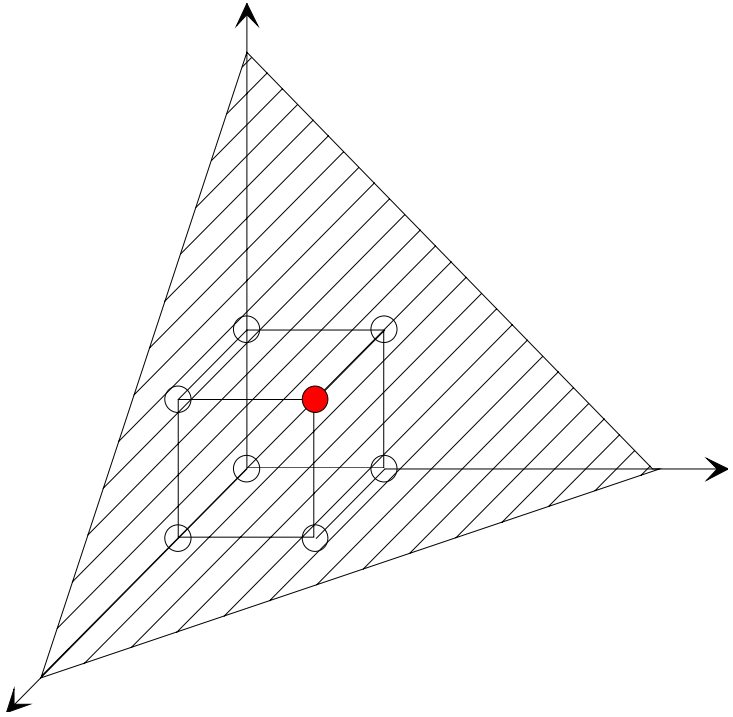
Так как $\sqrt{\sum_{j=0}^{n-1} \alpha_j^2} > 0,$ то $Sign\{D(Y, P^n)\} = Sign(\sum_{j=0}^{n-1} \alpha_j y_j - \eta)$

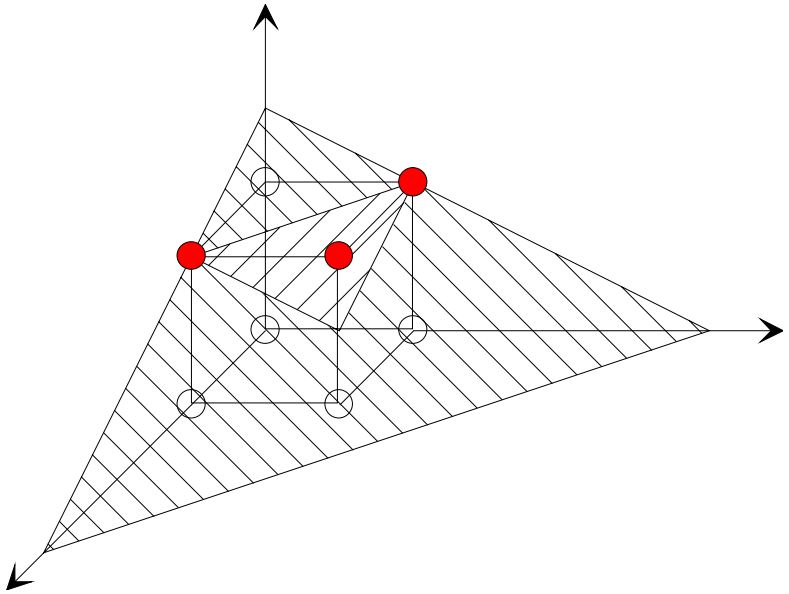
и значение пороговой функции в точке Y (вершине гиперкуба) определяется знаком расстояния от этой точки до гиперплоскости, заданной весами переменных и порогом.

Во всех вершинах гиперкуба, лежащих в гиперплоскости и расположенных выше нее, значение пороговой функции равно 1.

Во всех вершинах гиперкуба, расположенных ниже гиперплоскости, значение пороговой функции равно 0.

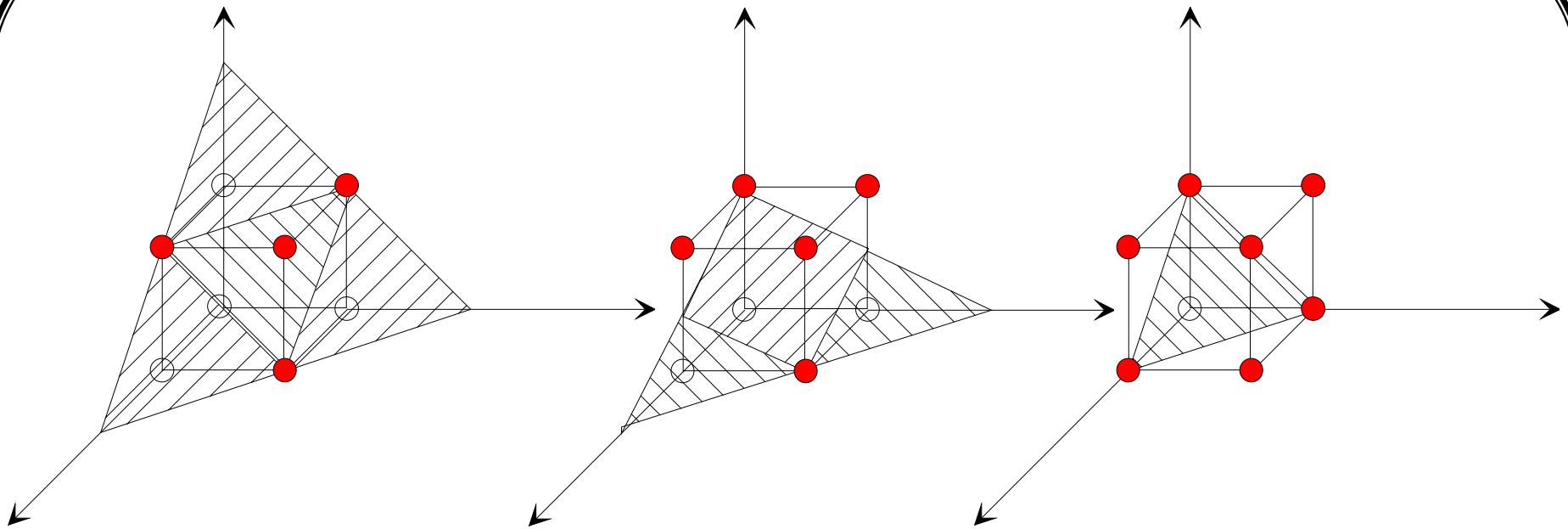
Примеры пороговых функций трех переменных





$$F(x_0,x_1,x_2) = x_0x_1x_2 = \\ = Sign(x_0 + x_1 + x_2 - 3);$$

$$F(x_0,x_1,x_2) = x_0x_2 + x_1x_2 = \\ = Sign(x_0 + x_1 + 2x_2 - 3)$$



$$F(x_0,x_1,x_2) = x_0x_1 + x_0x_2 + x_1x_2 = Sign(x_0 + x_1 + x_2 - 2);$$

$$F(x_0,x_1,x_2) = x_0x_1 + x_2 = Sign(x_0 + x_1 + 2x_2 - 2);$$

$$F(x_0,x_1,x_2) = x_0 + x_1 + x_2 = Sign(x_0 + x_1 + x_2 - 1);$$

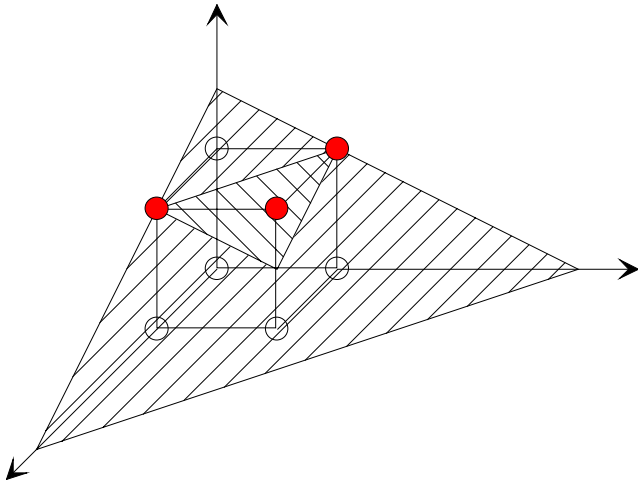
Инвертирование переменной и вращение гиперплоскости

Пусть булева функция $F(x_1,x_2,...,x_k,...,x_{n-1})$ является пороговой, т.е.

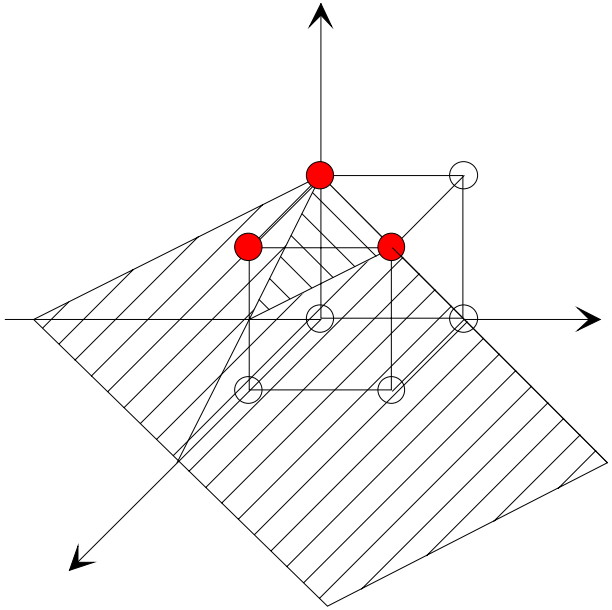
$$F(x_1,x_2,...,x_k,...,x_{n-1}) = Sign(a_0x_0 + \alpha_1x_1 + ... + \alpha_kx_k + ... + \alpha_{n-1}x_{n-1} - \eta),$$

тогда $F(x_1,x_2,...,\bar{x}_k,...,x_{n-1}) =$

$$= Sign(\alpha_0x_0 + \alpha_1x_1 + ... + \alpha_k(1 - x_k) + ... + \alpha_{n-1}x_{n-1} - \eta) =$$
$$= Sign(\alpha_0x_0 + \alpha_1x_1 + ... - \alpha_kx_k + ... + \alpha_{n-1}x_{n-1} - \eta + \alpha_k).$$



$$F(x_0,x_1,x_2) = x_0x_2 + x_1x_2 =$$
$$= Sign(x_0 + x_1 + 2x_2 - 3);$$



$$F(x_0,x_1,x_2) = x_0x_2 + \bar{x}_1x_2 =$$
$$= Sign(x_0 - x_1 + 2x_2 - 2);$$

Теорема: ***Пороговая функция является монотонной относительно всех ее переменных.***

Напоминание: Функция $F(X) = F(x_0, x_1, ..., x_k, ..., x_{n-1})$ является монотонной относительно переменной x_k, если для двух значений a и b этой переменной из $a>b$ следует, что $F(x_k = a) \geq F(x_k = b)$, или из $a<b$ следует, что $F(x_k = a) \geq F(x_k = b)$.

Доказательство: Для любых $\alpha_k > 0$ и $a > b$

$$\alpha_0 x_0 + \alpha_1 x_1 + ... + \alpha_k a + ... + \alpha_{n-1} x_{n-1} > \alpha_0 x_0 + \alpha_1 x_1 + ... + \alpha_k b + ... + \alpha_{n-1} x_{n-1},$$

следовательно,

$$Sign(\alpha_0 x_0 + \alpha_1 x_1 + ... + \alpha_k a + ... + \alpha_{n-1} x_{n-1} - \eta) \geq$$

$$\geq Sign(\alpha_0 x_0 + \alpha_1 x_1 + ... + \alpha_k b + ... + \alpha_{n-1} x_{n-1} - \eta)$$

или для любых $\alpha_k < 0$ и $a > b$

$$\alpha_0 x_0 + \alpha_1 x_1 + ... - \alpha_k a + ... + \alpha_{n-1} x_{n-1} < \alpha_0 x_0 + \alpha_1 x_1 + ... - \alpha_k b + ... + \alpha_{n-1} x_{n-1},$$

следовательно,

$$Sign(\alpha_0 x_0 + \alpha_1 x_1 + ... - \alpha_k a + ... + \alpha_{n-1} x_{n-1} - \eta) \leq$$

$$\leq Sign(\alpha_0 x_0 + \alpha_1 x_1 + ... - \alpha_k b + ... + \alpha_{n-1} x_{n-1} - \eta)$$

Все пороговые функции монотонны, но обратное неверно – не все монотонные функции пороговые. Например, функция $x_0x_1 + x_2x_3$ монотонная, но не пороговая.

Для булевой пороговой функции, заданной двумя множествами вершин T and F, $F(X \in T) = 1$, $F(X \in F) = 0$, веса переменных и значение порога могут быть найдены посредством решения системы неравенств:

$$\min(\sum_{j=0}^{n-1} |\alpha_j| + |\eta|);$$

$$\sum_{j=0}^{n-1} \alpha_j x_j - \eta \geq 0 \qquad \forall X \in T;$$

$$\sum_{j=0}^{n-1} \alpha_j x_j - \eta < 0 \qquad \forall X \in F;$$

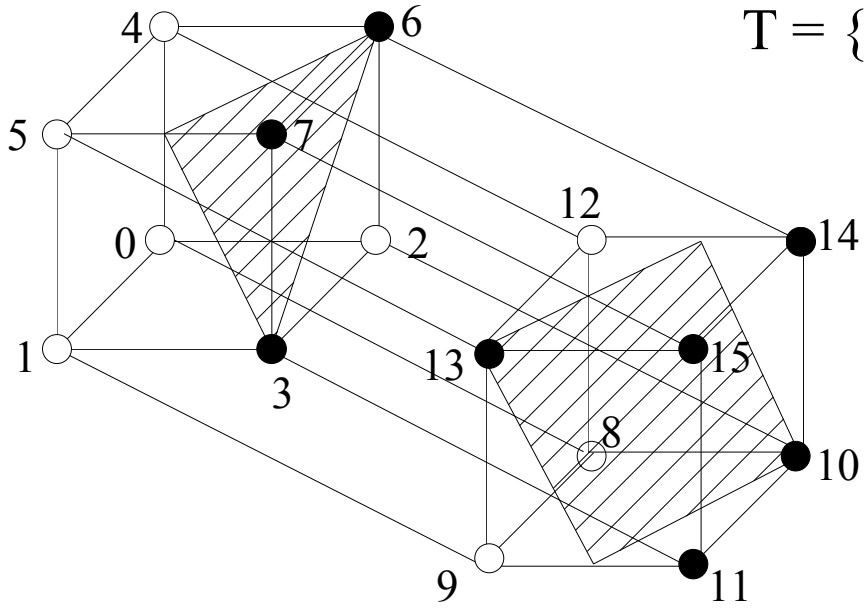
которая является стандартной задачей ***линейного программирования***.

Доказано, что если решение этой задачи существует, то для $x_j = \{0,1\}$ оно существует в целых числах. Если решения не существует, то заданная булева функция не является пороговой.

Так как пороговая функция является монотонной, она представляется звездой из существенных элементов.

Опорные множества T_0 и F_0 пороговой функции – множества вершин, которые лежат на максимальных диагоналях элементов, исходящих из вершины звезды как самой функции, так и ее инверсии.

Example:



Вершина звезды функции – вершина 15.

T = {3,6,7,10,11,13,14,15} ; T_0={3,6,10,13}.

Вершиной звезды инверсной функции является вершина 0.

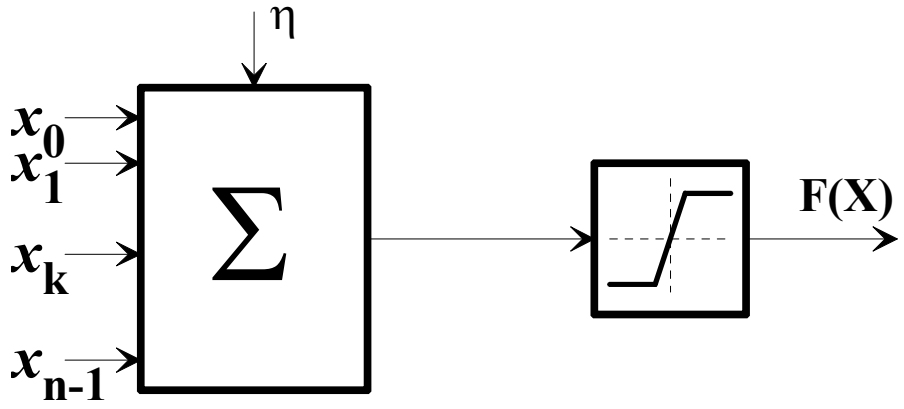
F = {0,1,2,5,8,9,12}; F_0={2,5,9,12}.

$$F(x_0, x_1, x_2, x_3) =$$
$$= x_0x_1 + x_1x_2 + x_1x_3 + x_0x_2x_3 =$$
$$= Sign(x_0 + 2x_1 + x_2 + x_3 - 3)$$

При синтезе порогового элемента достаточно решить задачу линейного программирования только для опорных множеств , так как если для них неравенства выполняются, то они определенно выполняются для всех оставшихся вершин.

Реализуемость пороговых элементов

Пороговый элемент по определению имеет следующую структуру:



Сумматор и нелинейный элемент – физические объекты, функционирующие с некоторой точностью. Сумматор имеет ошибку суммирования $\Delta\Sigma$, которая зависит от ошибок представления входных сигналов Δx_k и разброса физических параметров Δp_i схемы. Нелинейный элемент имеет зону нечувствительности с некоторым разбросом Δs положения порога в этой зоне.

Правильность работы порогового элемента определяется правильностью его поведения в зоне переключения.

$d = 1 / \sum_{j=0}^{n-1} \alpha_j$ – минимально возможное изменение суммы (минимальный скачок). Когда сумма равна наибольшему значению перед порогом, выполняется следующее условие: $(\eta - 1)d + \Delta\Sigma < S - \Delta s$.

Если сумма превышает пороговое значение, то должно выполняться условие $\eta d - \Delta\Sigma > S + \Delta s$.

Откуда следует, что условием правильного функционирования порогового элемента является $\Delta\Sigma + \Delta s < d/2$.

Из стандартной теории ошибок следует, что:

$$\Delta\Sigma = \sum_{j=0}^{n-1}\left|\Delta x_j \frac{\partial\Sigma}{\partial x_j}\right| + \left|\Delta\eta\right| + \sum_{i=0}^{m}\left|\Delta p_i \frac{\partial\Sigma}{\partial p_i}\right| = \sum_{j=0}^{n-1}\left|\Delta x_j \alpha_j\right| + \left|\Delta\eta\right| + \sum_{i=0}^{m}\left|\Delta p_i\right|\sum_{j=0}^{n-1}\left|\frac{\partial\Sigma}{\partial\alpha_j}\frac{\partial\alpha_j}{\partial p_i}\right| =$$

$$= \sum_{j=0}^{n-1}\left|\Delta x_j \alpha_j\right| + \left|\Delta\eta\right| + \sum_{i=0}^{m}\left|\Delta p_i\right|\sum_{j=0}^{n-1}\left|x_j \frac{\partial\alpha_j}{\partial p_i}\right| = \sum_{j=0}^{n-1}\left|\Delta x_j \alpha_j\right| + \left|\Delta\eta\right| + \sum_{j=0}^{n-1} x_j \sum_{i=0}^{m}\left|\Delta p_i \frac{\partial\alpha_j}{\partial p_i}\right|.$$

Здесь $\sum_{j=0}^{n-1}\left|\Delta x_j \alpha_j\right| + \left|\Delta\eta\right|$ – значение функциональной ошибки,

$\sum_{j=0}^{n-1} x_j \sum_{i=0}^{m}\left|\Delta p_i \frac{\partial\alpha_j}{\partial p_i}\right|$ – величина параметрической ошибки.



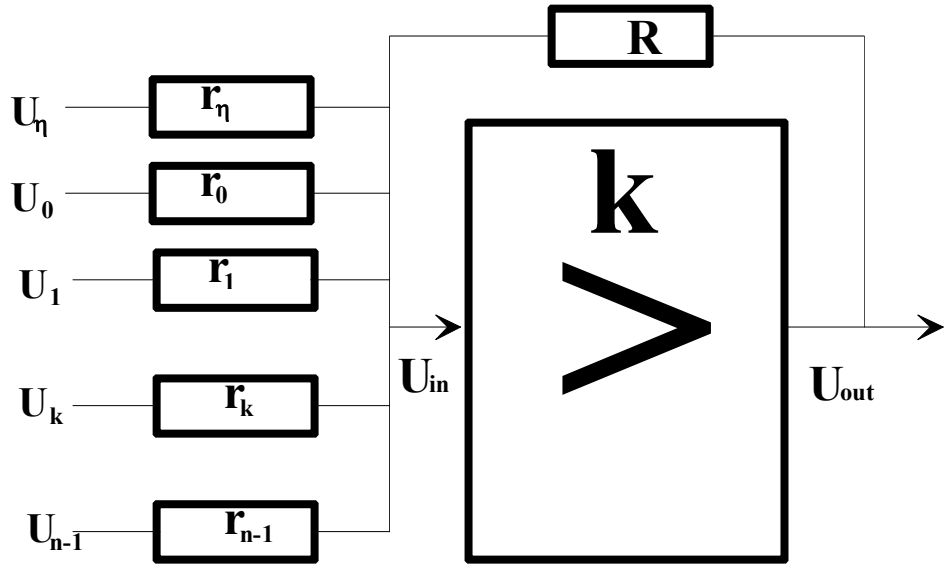
Логическое Проектирование

Лекция 8

Функциональная ошибка зависит от весов входов и точности установки порога.

Параметрическая ошибка определяется точностью физических параметров и зависимостью весов входов от этих параметров.

Пример 1. ***Суммирующий усилитель***



$$\sum_{j=0}^{n-1}(U_j - U_{in})/r_j + (U_\eta - U_{in})/r_\eta + (U_{out} - U_{in})/R = 0$$

$$U_{out} = -kU_{in}; \qquad k >> 1;$$

$$\sum_{j=0}^{n-1} U_j/r_j + U_\eta/r_\eta = \frac{U_{out}}{k}\{\sum_{j=0}^{n-1} 1/r_j + 1/r_\eta + 1/R\} - U_{out}/R$$

$$U_{out} = |\sum_{j=0}^{n-1} U_j \frac{R}{r_j} + U_\eta \frac{R}{r_\eta}|; \qquad \alpha_j = \frac{R}{r_j}; \qquad \eta = \frac{R}{r_\eta}$$

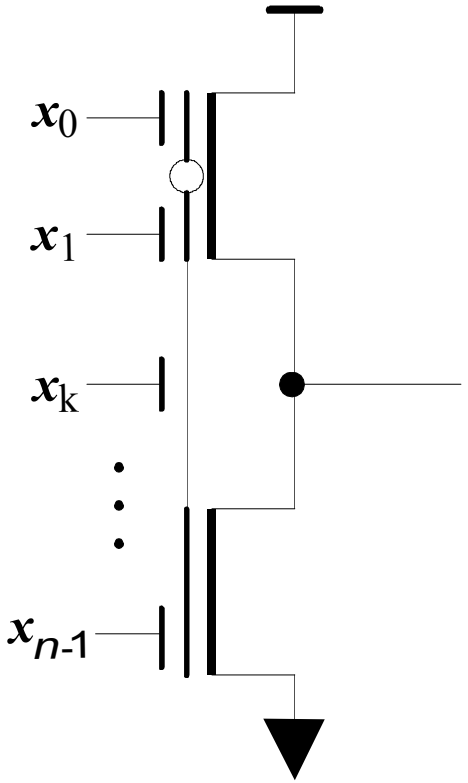
$$\sum_{i=0}^{m}\left|\Delta p_i \frac{\partial \alpha_j}{\partial p_i}\right| = \left|\Delta R \frac{\partial}{\partial R}(\frac{R}{r_j})\right| + \left|\Delta r_j \frac{\partial}{\partial r_j}(\frac{R}{r_j})\right| = \frac{\Delta R}{r_j} + \frac{\Delta r_j R}{r_j^2}$$

если $r_j = R/\alpha_j$ и $\Delta r_j = \Delta R/\alpha_j$

тогда $$\sum_{j=0}^{n-1} x_j \sum_{i=0}^{m}\left|\Delta p_i \frac{\partial \alpha_j}{\partial p_i}\right| = \frac{2\Delta R}{R}\sum_{j=0}^{n-1} x_j \alpha_j.$$

Следовательно, параметрическая ошибка, зависящая от относительной точности изготовления резисторов, линейно возрастает с ростом *n* (числа входов).

Пример 2. *КМОП пара с плавающими затворами*



$$\alpha_j = \frac{C_j}{C_{pair} + \sum_{i=0}^{n-1} C_i}$$

$$\frac{\partial \alpha_j}{\partial C_j} = \frac{C_{pair} + \sum_{i=0}^{n-1} C_i - C_j}{(C_{pair} + \sum_{i=0}^{n-1} C_i)^2}; \quad \left|\frac{\partial \alpha_j}{\partial C_i}\right| = \frac{C_j}{(C_{pair} + \sum_{i=0}^{n-1} C_i)^2};$$

$$\left|\frac{\partial \alpha_j}{\partial C_{pair}}\right| = \frac{C_j}{(C_{pair} + \sum_{i=0}^{n-1} C_i)^2};$$

$$\sum_{i=0}^{m} \left|\Delta p_i \frac{\partial \alpha_j}{\partial p_i}\right| = \frac{C_j(\Delta C_{pair} + \sum_{i=0}^{n-1} \Delta C_i)}{(C_{pair} + \sum_{i=0}^{n-1} C_i)^2} + \Delta C_j \frac{C_{pair} + \sum_{i=0}^{n-1} C_i - 2C_j}{(C_{pair} + \sum_{i=0}^{n-1} C_i)^2}.$$

если $C_j = c\dfrac{\alpha_j}{\beta + \sum_{i=0}^{n-1}\alpha_i}$; $C_{pair} = c\dfrac{\beta}{\beta + \sum_{i=0}^{n-1}\alpha_i}$; $\Delta C_j = \Delta c\dfrac{\alpha_j}{\beta + \sum_{i=0}^{n-1}\alpha_i}$;

тогда $$\sum_{j=0}^{n-1} x_j \sum_{i=1}^{m}\left|\Delta p_i \frac{\partial \alpha_j}{\partial p_i}\right| = \frac{2\Delta c}{c}\left\{\frac{\sum_{j=0}^{n-1} x_j\alpha_j}{\beta + \sum_{j=0}^{n-1}\alpha_j}\left(1 - \frac{1}{(\beta + \sum_{j=0}^{n-1}\alpha_j)}\right)\right\}.$$

Сложение на конденсаторах по сравнению со сложением на резисторах имеет замечательное свойство: ***величина параметрической ошибки не зависит от числа переменных или от суммы весов входов и имеет порядок относительной точности изготовления конденсаторов.***
Это заключение кажется очевидным, так как умножение всех емкостей на один и тот же коэффициент не изменяет распределения напряжений. Величина параметрической ошибки уменьшается с увеличением паразитной емкости, однако при этом уменьшается и величина единичного скачка. Условие правильного функционирования выглядит как :

$$\Delta\Sigma + \Delta s < \frac{1}{2(\beta + \sum_{j=0}^{n-1}\alpha_j)}.$$

Основные области применения пороговых и мажоритарных функций

Пороговая функция называется мажоритарной (*функцией голосования*), *если ее значение совпадает с более, чем половиной входных сигналов*

$$(\alpha_i = 1, \eta =]\frac{n}{2}[+1).$$

1. ***Логический синтез дискретных устройств***
(аппаратные средства)

2. *Распознавание образов. Диагностика*
(аппаратные средства и программное обеспечение).

Пример. На основании множества симптомов (*результатов анализов, температуры, кардиограммы и т.д.*) *для каждой возможной болезни вычисляется взвешенная сумма и сравнивается с порогом.*

3. *Нейронные сети*
(аппаратные средства и программное обеспечение)

Входные веса и порог формируются во время обучения.

Для пунктов 2 и 3 точность определения значения функции в зоне порога и значения взвешенной суммы не является критичной. Более того, по самой природе задач должна существовать зона нечувствительности:

$$\eta - \delta < \sum \alpha_j x_j < \eta + \delta.$$

4. *Кодирование*

(аппаратные средства и программное обеспечение)

Пример 1: Оптимальный равновесный код (*ОРК*)

Рабочий набор *n*-мерного ОРК содержит $]\frac{n+1}{2}[$ единиц.

n=4, множество рабочих наборов: {1100, 1010, 1001, 0110, 0101, 0011}

Проверочное условие, принадлежит ли код множеству рабочих наборов, имеет вид:

$$Sign(\sum_{j=0}^{n-1} x_j -]\frac{n+1}{2}[)(1 - Sign(\sum_{j=0}^{n-1} x_j -]\frac{n+1}{2}[-1)).$$

Пример 2: Код Бергера (*код с идентификатором*)

Код Бергера состоит из двух частей: n-битовый двоичный информационный код и (]$\log_2$n[+1)-битовый код-идентификатор, значение которого равно числу нулей в информационном коде.
Для n=3, k=2 и множество рабочих наборов: {000.11, 100.01, 110.10, 111.00, ...}
Принадлежность кода рабочему множеству определяется истинностью выражения:

$$Sign(\sum_{j=0}^{n-1} x_j + \sum_{i=0}^{k-1} 2^i y_i - n)(1 - Sign(\sum_{j=0}^{n-1} x_j + \sum_{i=0}^{k-1} 2^i y_i - n - 1)).$$

5. *Отказоустойчивые устройства*

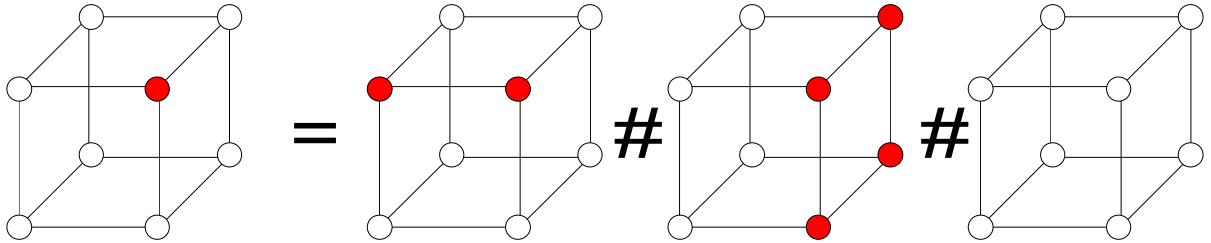
Использование мажоритарных элементов в качестве восстанавливающих органов.

Логический синтез из трехвходовых мажоритарных элементов

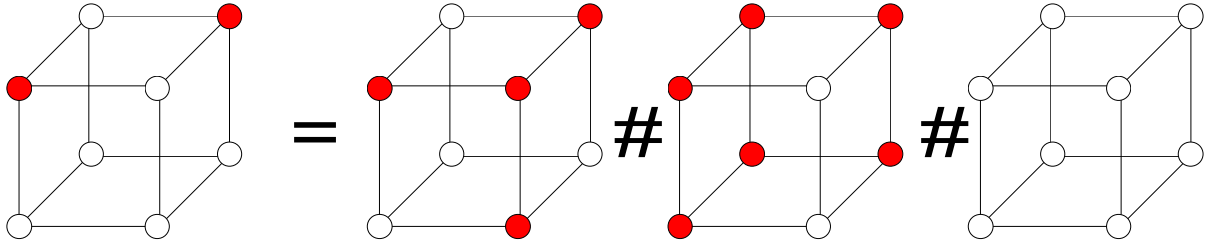
$$\mathrm{Maj}(x_0,x_1,x_2) = \mathrm{Sign}(x_0+x_1+x_2-2) = x_0\#x_1\#x_2 = x_0x_1+x_0x_2+x_1x_2$$

$$x_0x_1 = x_0\#x_1\#0; \quad x_0+x_1 = x_0\#x_1\#1;$$

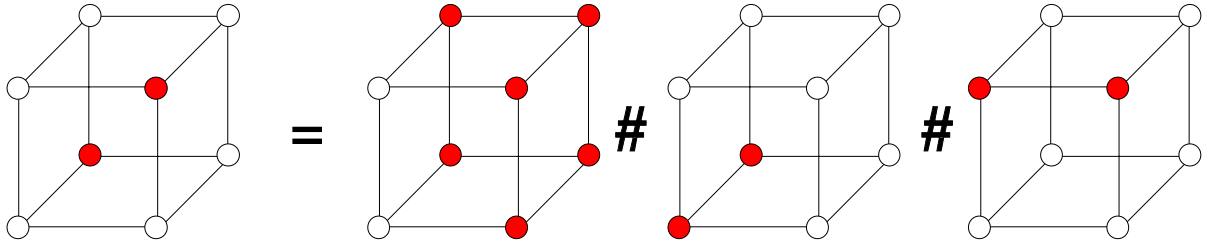
Любая логическая функция трех переменных может быть реализована двухъярусной схемой из четырех мажоритарных элементов с использованием инверсий входов первого яруса.



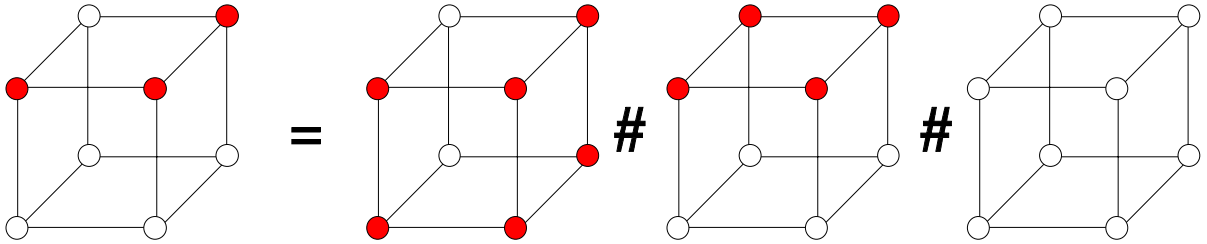
$$x_0x_1x_2 = (x_0\#x_2\#0)\#x_1\#0$$



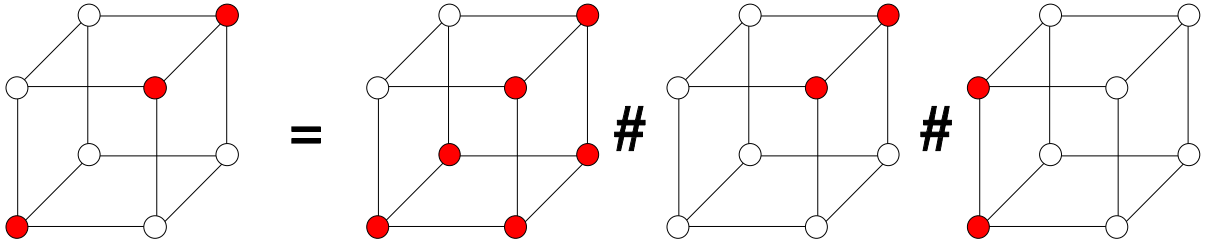
$$x_0\bar{x}_1x_2 + \bar{x}_0x_1x_2 = (x_0\#x_1\#x_2)\#(\bar{x}_0\#\bar{x}_1\#1)\#0$$



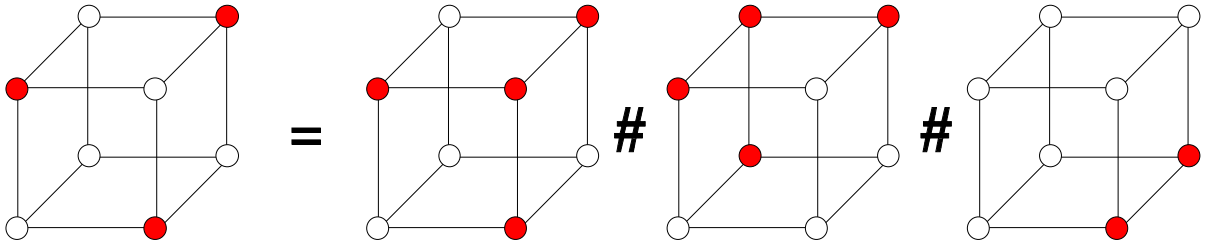
$$x_0x_1x_2 + \bar{x}_0\bar{x}_1\bar{x}_2 = (\bar{x}_0 \# x_1 \# 1) \# (\bar{x}_1 \# \bar{x}_2 \# 0) \# (x_1 \# x_2 \# 0)$$



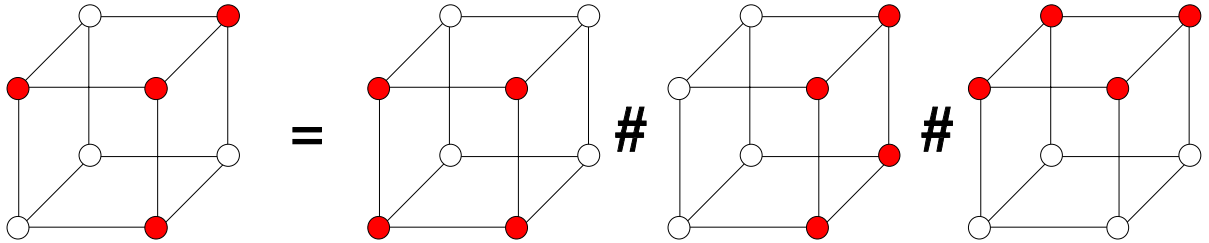
$$(x_0 + x_1)x_2 = (x_0 \# x_1 \# 1) \# x_2 \# 0$$



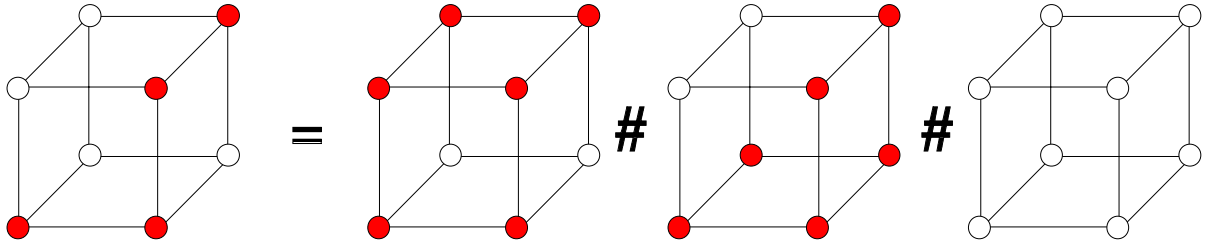
$$x_1x_2 + x_0\bar{x}_1\bar{x}_2 = (x_1 \# \bar{x}_2 \# 1) \# (x_1 \# x_2 \# 0) \# (x_0 \# \bar{x}_1 \# 0)$$



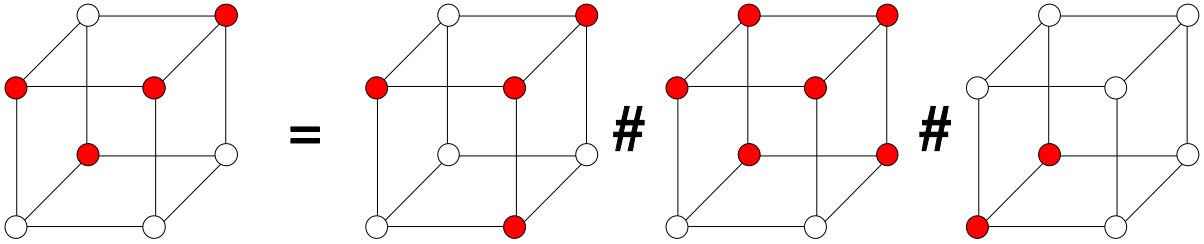
$$x_0x_1\bar{x}_2 + x_0\bar{x}_1x_2 + \bar{x}_0x_1x_2 = (x_0 \# x_1 \# x_2) \# (\bar{x}_0 \# \bar{x}_1 \# x_2) \# (x_1 \# \bar{x}_2 \# 0)$$



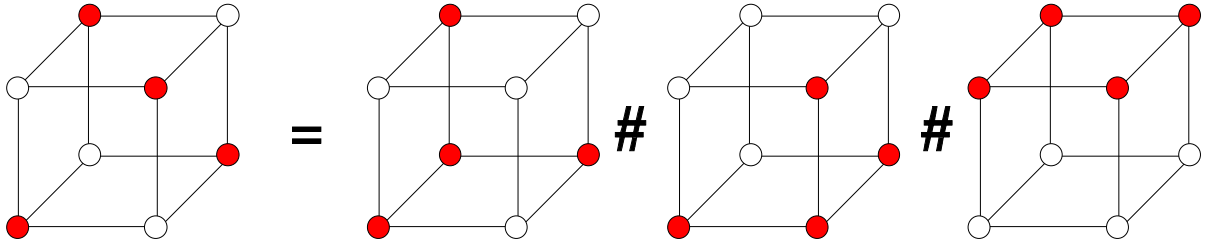
$$x_0x_1 + x_0x_2 + x_1x_2 = x_0 \# x_1 \# x_2$$



$$x_0\bar{x}_2 + x_1x_2 = (x_0 \# x_2 \# 1) \# (x_1 \# \bar{x}_2 \# 1) \# 0$$



$$x_0x_2 + x_1x_2 + \bar{x}_0\bar{x}_1\bar{x}_2 = (x_0 \# x_1 \# x_2) \# (\bar{x}_0 \# x_2 \# 1) \# (\bar{x}_1 \# \bar{x}_2 \# 0)$$



$$x_0 \oplus x_1 \oplus x_2 = (\bar{x}_0 \# \bar{x}_1 \# \bar{x}_2) \# (x_0 \# x_1 \# \bar{x}_2) \# x_2$$

Мажоритарная декомпозиция

Любая схема является композицией простых (базовых) элементов (функций). Следовательно, проблема синтеза схемы, реализующую функцию, является проблемой представления заданной функции как суперпозиции функций базовых элементов (декомпозиция, отображение) .

Синтез может выполняться **«сверху-вниз»** (от выхода) или **«снизу-вверх»** (от входов).

Для процедуры сверху-вниз

1. Фиксируют выходной элемент функции. В данном случае мажоритарный элемент: $F(X) = Y_1(X)\#Y_2(X)\#Y_3(X)$.

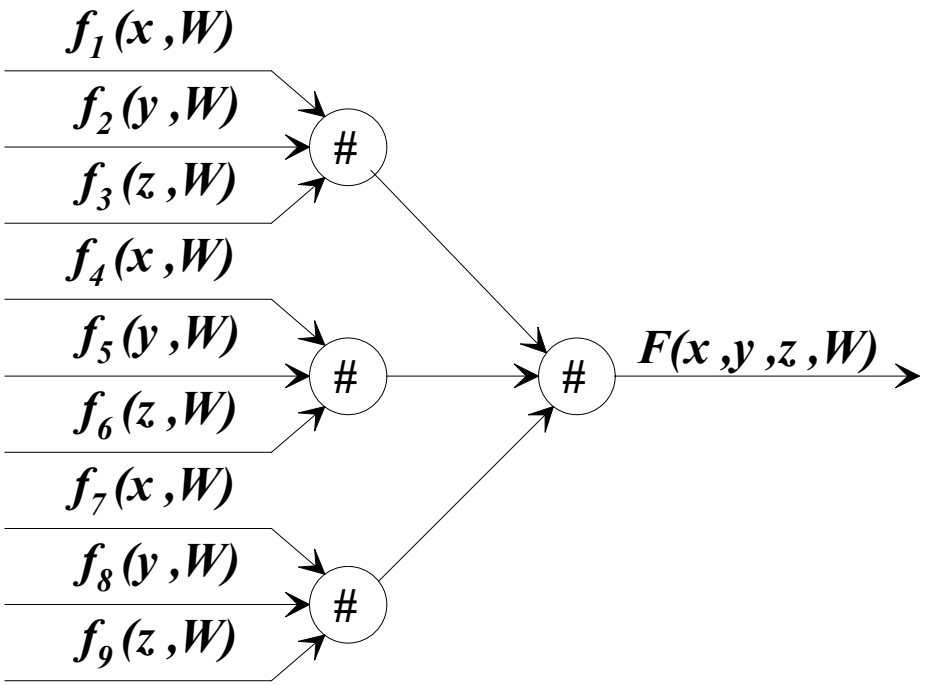
2. Ищут функции $Y_j(X)$, реализация каждой из которых гарантированно проще по некоторому критерию, чем реализация функции $F(X)$.

Известны, по крайней мере два таких критерия:

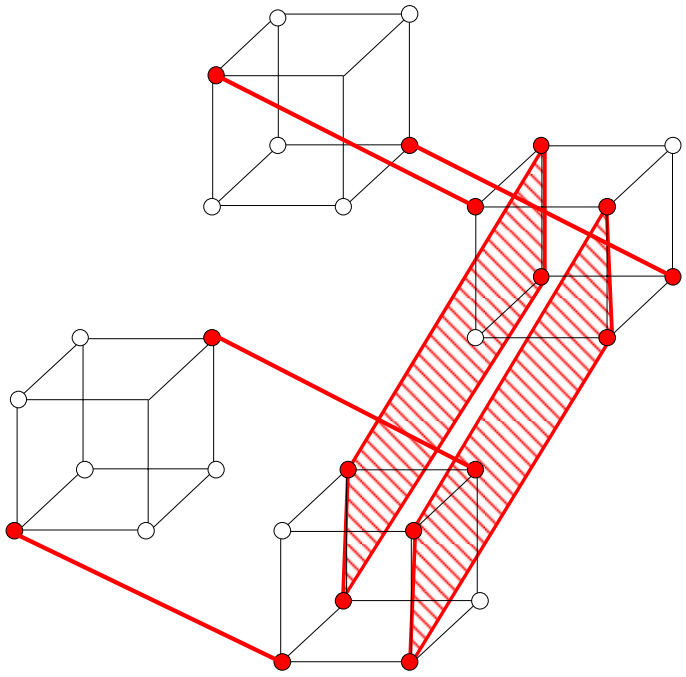
- сокращение числа переменных, от которых зависят функции $Y_j(X)$;
- сокращение области определения функций $Y_j(X)$ (сокращение числа наборов переменых, на которых функция определена).

Сокращение числа переменных, от которых зависят функции $Y_j(X)$

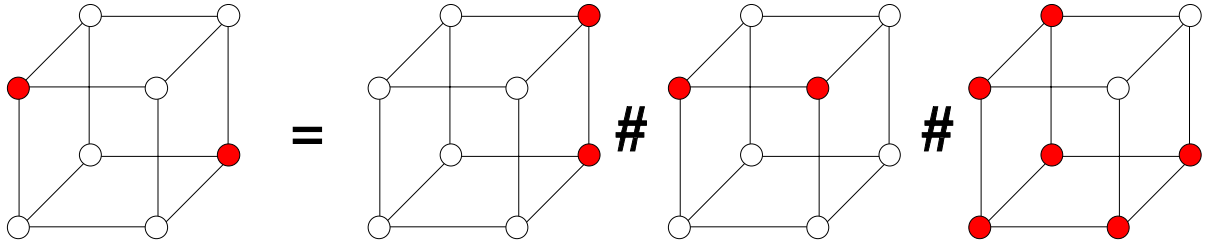
Из рассмотренных ранее реализаций функций трех переменных следует, что любая функция n переменных может быть реализована следующим образом:

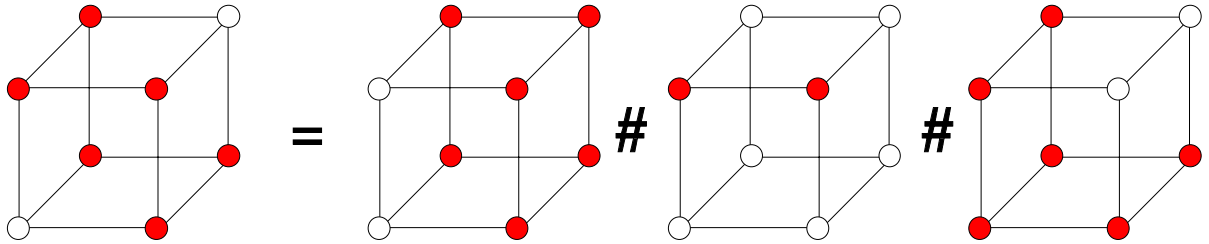


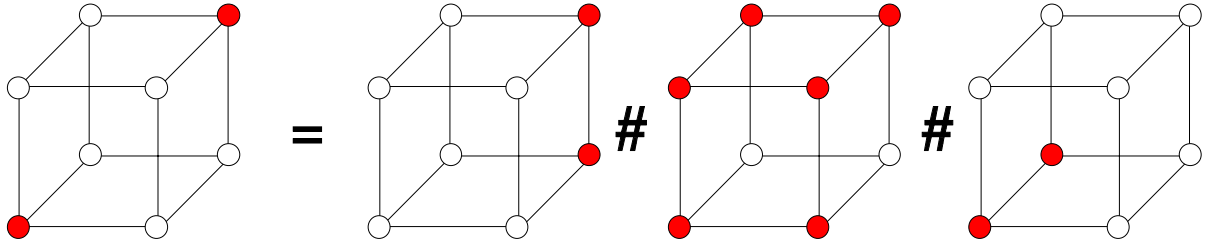
Пример

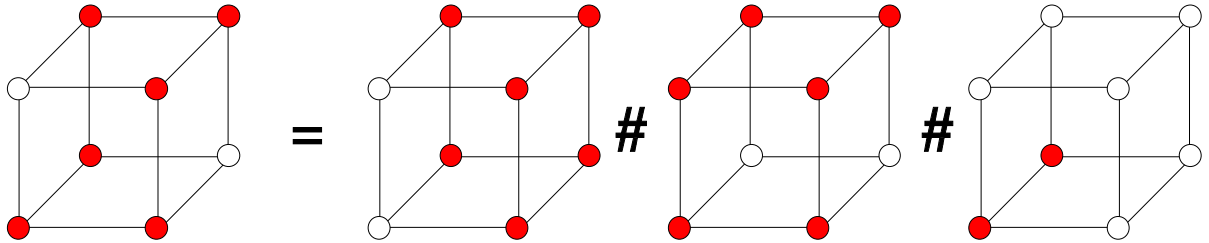


$$F(x, y, z, W) = x\bar{y}z\overline{w}_2 + \bar{x}y\bar{z}\overline{w}_2 + \overline{x\bar{y}}w_1 + xyw_1 + x\bar{y}\bar{z}w_2 + \bar{x}yzw_2$$

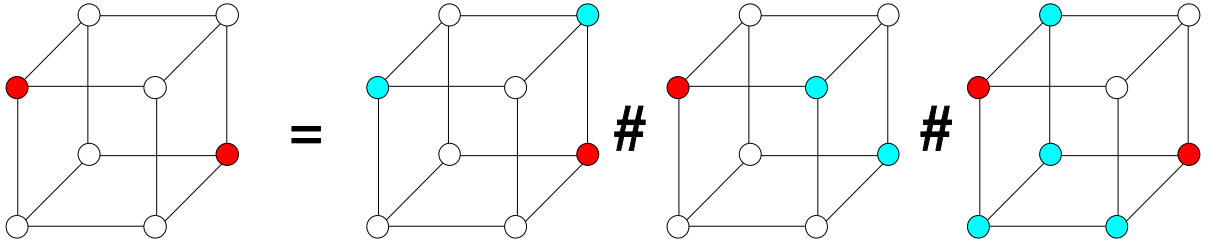
W_1=0; W_2=0; 

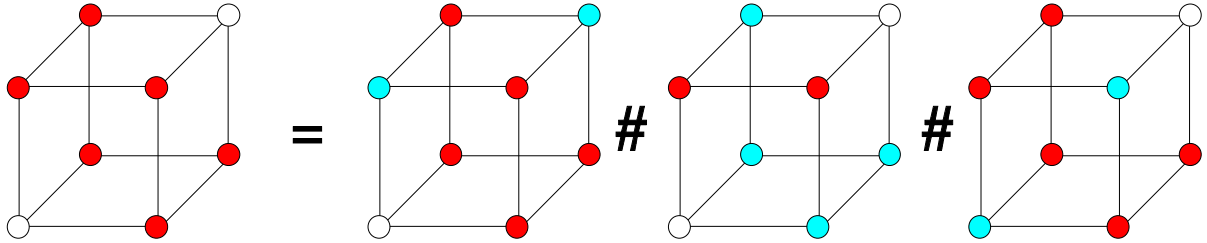
W_1=1; W_2=0; 

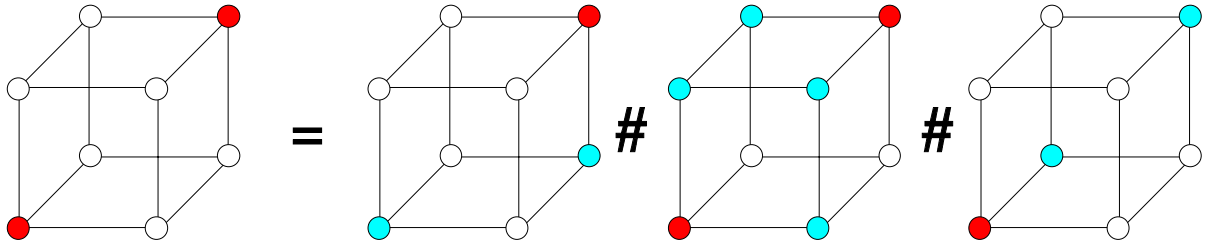
W_1=0; W_2=1; 

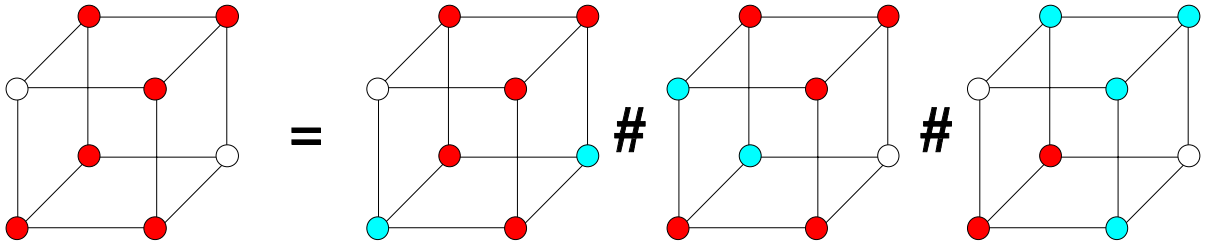
W_1=1; W_2=1; 

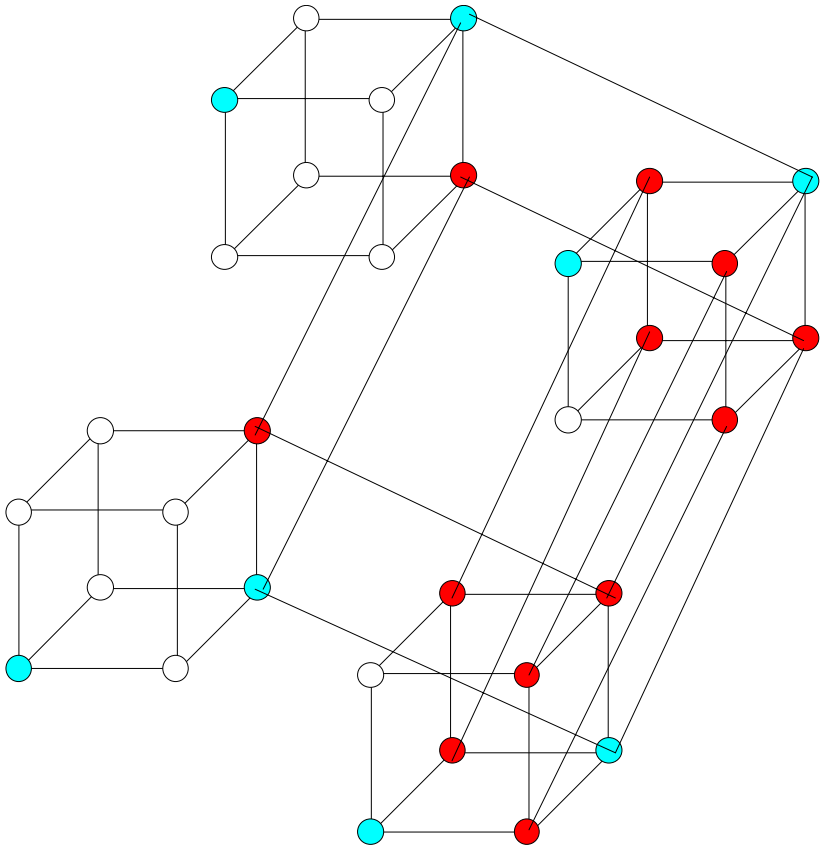
Заметим, что на некоторых вершинах функции могут быть определены произвольно.

$W_1=0;\ W_2=0;$ 

$W_1=1;\ W_2=0;$ 

$W_1=0;\ W_2=1;$ 

$W_1=1;\ W_2=1;$ 



$$Y_1(x, y, z, W) = \overline{x} \# y \# w_1$$

Можно проверить, что:

$$F(x, y, z, W) = (\overline{x} \# y \# w_1) \# (x \# z \# w_2) \# (\overline{x} \# \overline{z} \# \overline{w}_2)$$

Сокращение области определения $Y_j(X)$

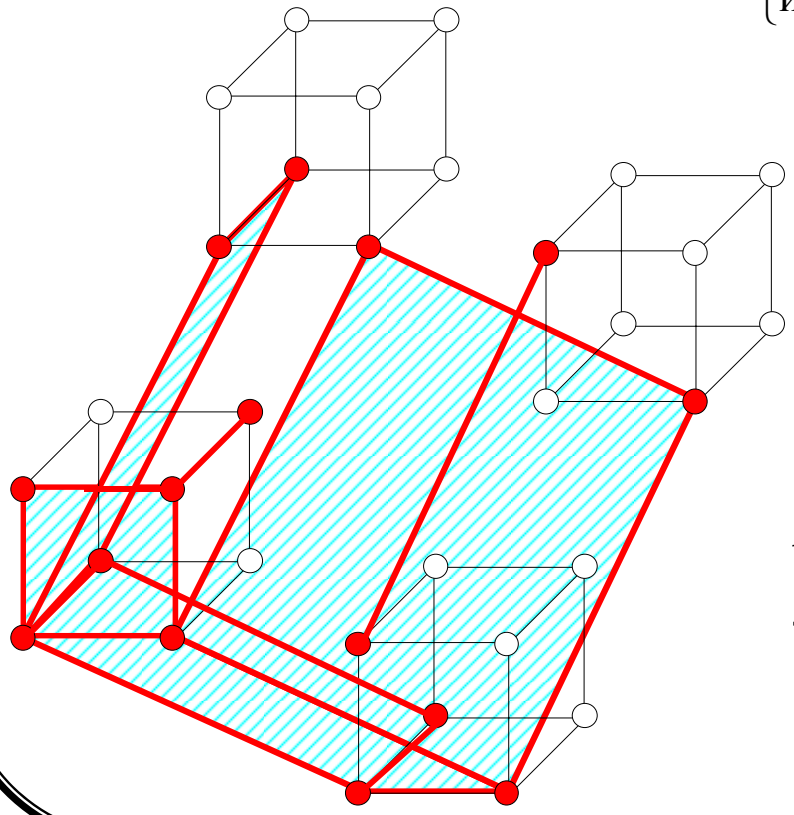
Если $F(X)=1$, то

$$\begin{cases} Y_1(X)=1;\ Y_2(X)=1;\ Y_3(X)=* \\ \text{или}\quad Y_1(X)=1;\ Y_2(X)=*;\ Y_3(X)=1 \\ \text{или}\quad Y_1(X)=*;\ Y_2(X)=1;\ Y_3(X)=1 \end{cases}$$

Если $F(X)=0$, то

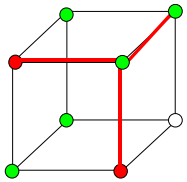
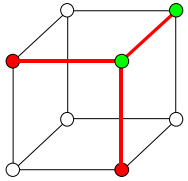
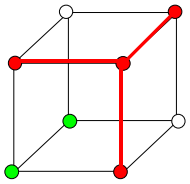
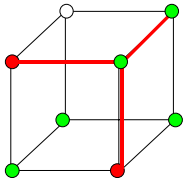
$$\begin{cases} Y_1(X)=0;\ Y_2(X)=0;\ Y_3(X)=* \\ \text{или}\quad Y_1(X)=0;\ Y_2(X)=*;\ Y_3(X)=0 \\ \text{или}\quad Y_1(X)=*;\ Y_2(X)=0;\ Y_3(X)=0 \end{cases}$$

Пример

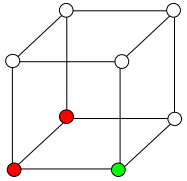
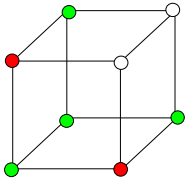
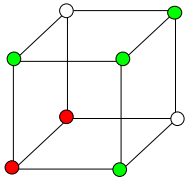
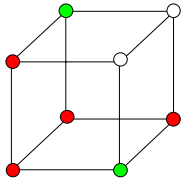


$$F(x,y,z,v,w)=xy\bar{z}+x\bar{z}w+$$
$$+x\bar{v}w+x\bar{y}zv+yz\bar{v}w+\bar{y}\bar{z}\bar{v}$$

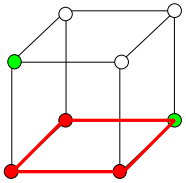
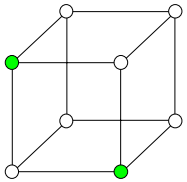
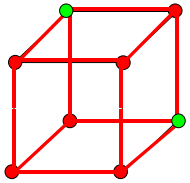
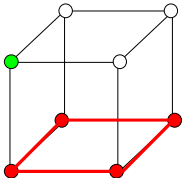
Входы							*Компоненты*		
x	*y*	*z*	*v*	*w*	*No*	*F(X)*	$Y_1(X)$	$Y_2(X)$	$Y_3(X)$
0	0	0	0	0	0	1	*	1	1
1	0	0	0	0	1	1	*	1	1
0	1	0	0	0	2	0	0	0	*
1	1	0	0	0	3	1	1	*	1
0	0	1	0	0	4	0	*	0	0
1	0	1	0	0	5	0	1	0	*
0	1	1	0	0	6	0	*	0	0
1	1	1	0	0	7	0	*	0	0
0	0	0	1	0	8	0	0	*	0
1	0	0	1	0	9	0	0	*	0
0	1	0	1	0	10	0	0	*	0
1	1	0	1	0	11	1	1	1	*
0	0	1	1	0	12	0	0	*	0
1	0	1	1	0	13	1	1	1	*
0	1	1	1	0	14	0	*	0	0
1	1	1	1	0	15	0	*	0	0
0	0	0	0	1	16	1	*	1	1
1	0	0	0	1	17	1	*	1	1
0	1	0	0	1	18	0	0	0	*
1	1	0	0	1	19	1	1	*	1
0	0	1	0	1	20	0	0	0	*
1	0	1	0	1	21	1	1	*	1
0	1	1	0	1	22	1	1	*	1
1	1	1	0	1	23	1	1	*	1
0	0	0	1	1	24	1	*	1	1
1	0	0	1	1	25	1	*	1	1
0	1	0	1	1	26	1	*	1	1
1	1	0	1	1	27	1	1	*	1
0	0	1	1	1	28	0	0	*	0
1	0	1	1	1	29	1	1	1	*
0	1	1	1	1	30	0	*	0	0
1	1	1	1	1	31	0	*	0	0

$$Y_1(X) = x\#\, y\#\, z$$

$$Y_2(X) = \overline{y}\#\, \overline{z}\#\, v$$

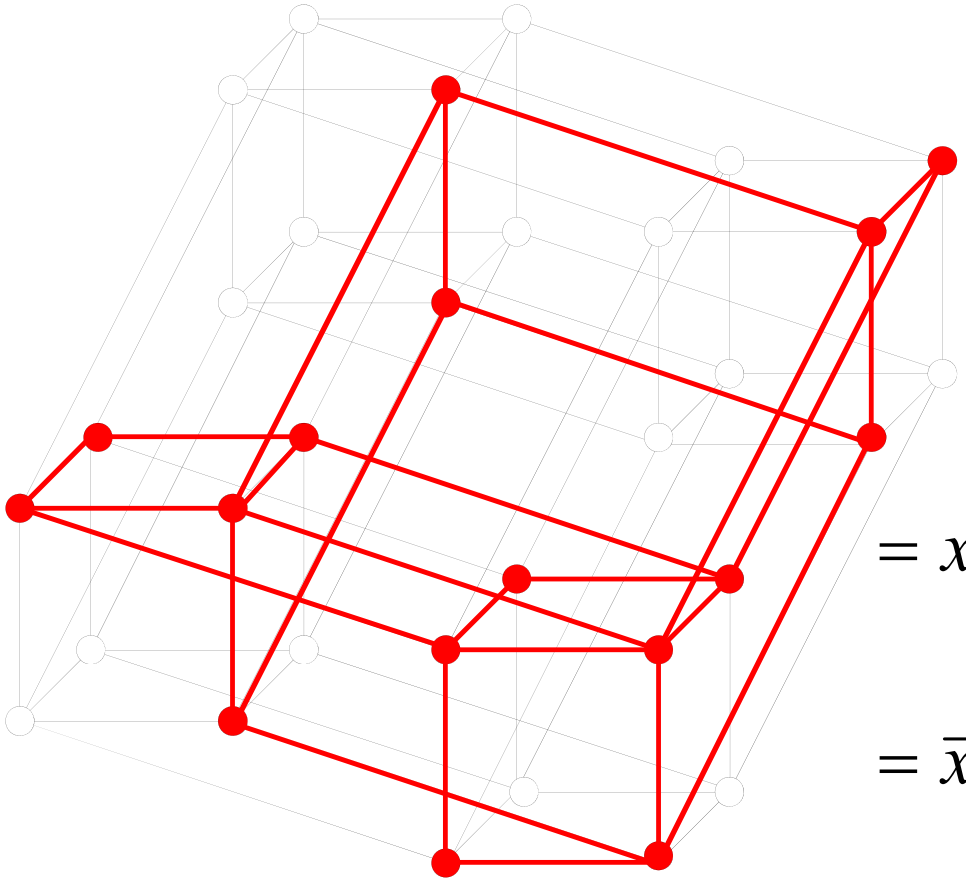





$$Y_3(X) = \overline{z}\#\, v\#\, \overline{w}$$

Мостиковые схемы

Пример: Пусть задана булева функция 5 переменных:

T={3,7,11,12,13,14,15,19,22,23,25,27,28,29,30,31}

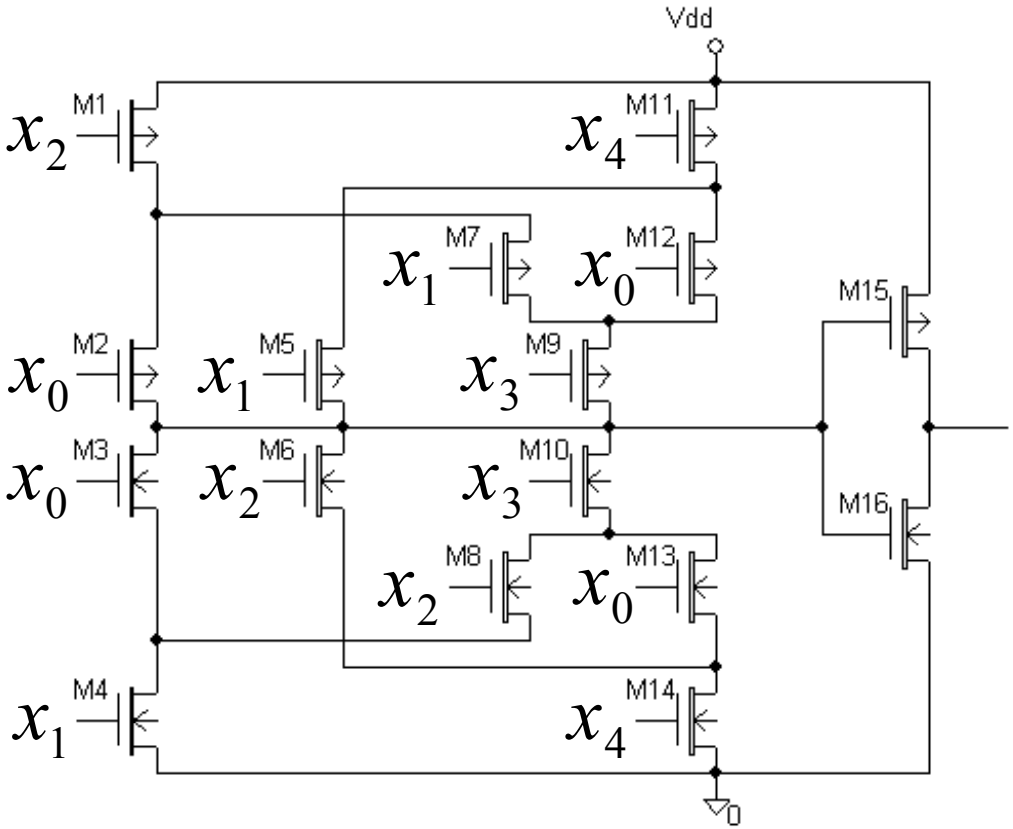


$$F(x_0, x_1, x_2, x_3, x_4) =$$
$$= x_0x_1 + x_2x_4 + x_0x_3x_4 + x_1x_2x_3$$

$$\overline{F}(x_0, x_1, x_2, x_3, x_4) =$$
$$= \overline{x}_0\overline{x}_2 + \overline{x}_1\overline{x}_4 + \overline{x}_0\overline{x}_3\overline{x}_4 + \overline{x}_1\overline{x}_2\overline{x}_3$$

КМОП реализация функции:

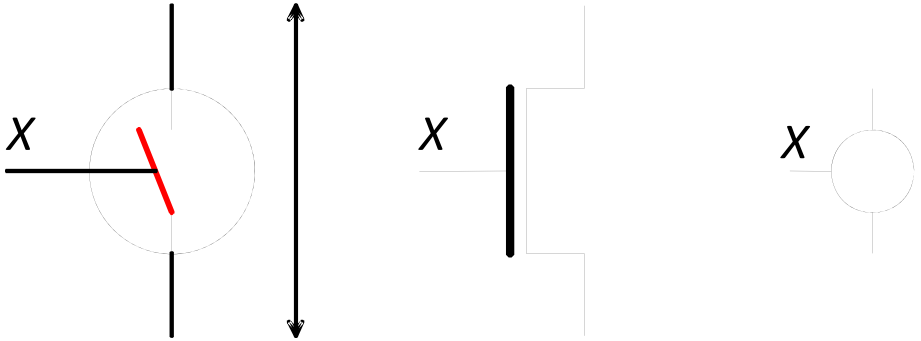
$$\overline{F}(x_0, x_1, x_2, x_3, x_4) =$$
$$= \overline{x}_0\overline{x}_2 + \overline{x}_1\overline{x}_4 + \overline{x}_0\overline{x}_3\overline{x}_4 + \overline{x}_1\overline{x}_2\overline{x}_3$$



$$F(x_0, x_1, x_2, x_3, x_4)$$

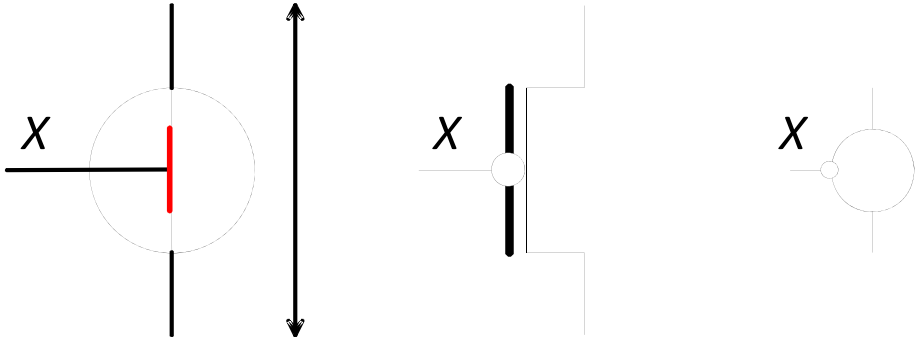
$$F(x_0, x_1, x_2, x_3, x_4) =$$
$$= x_0x_1 + x_2x_4 + x_0x_3x_4 + x_1x_2x_3$$

Элементы с двухсторонней проводимостью



Нормально открытый контакт

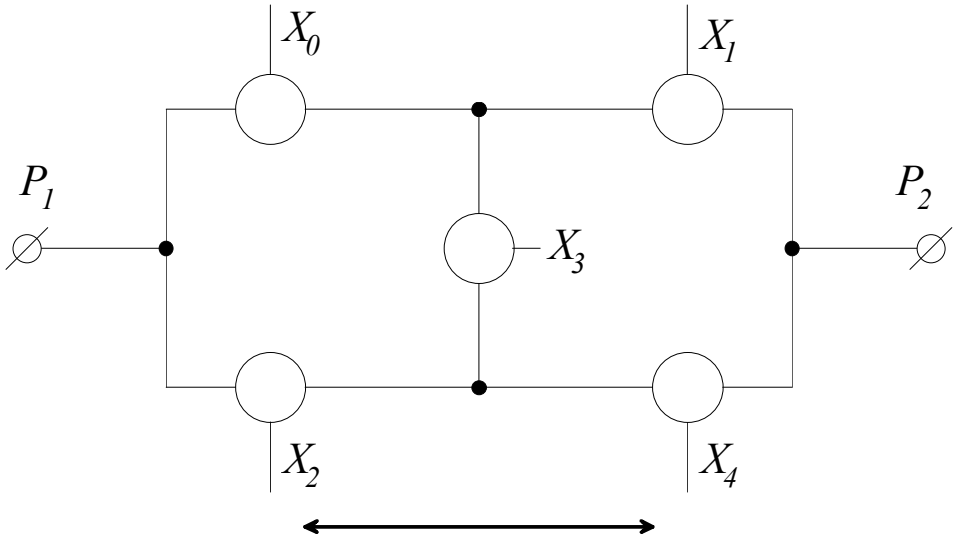
*n*МОП транзистор

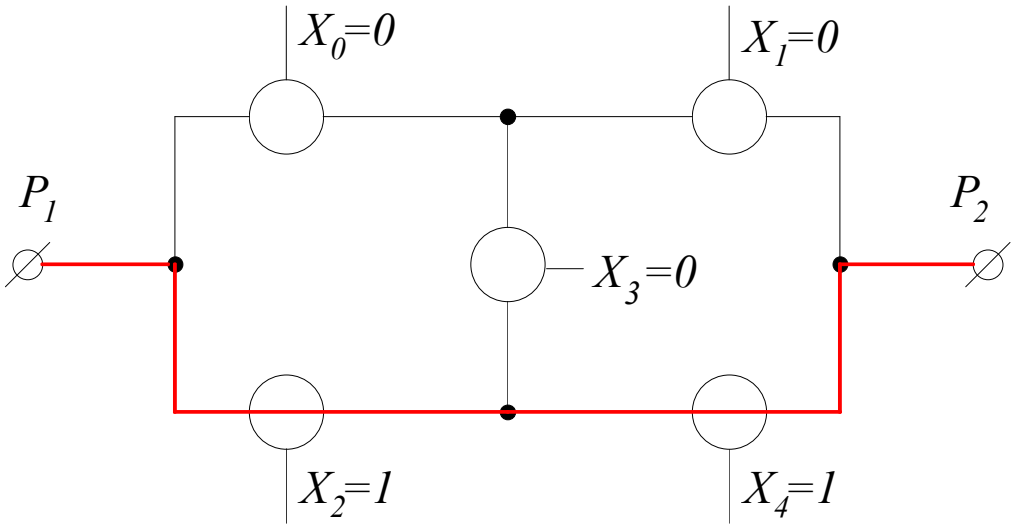


Нормально закрытый контакт

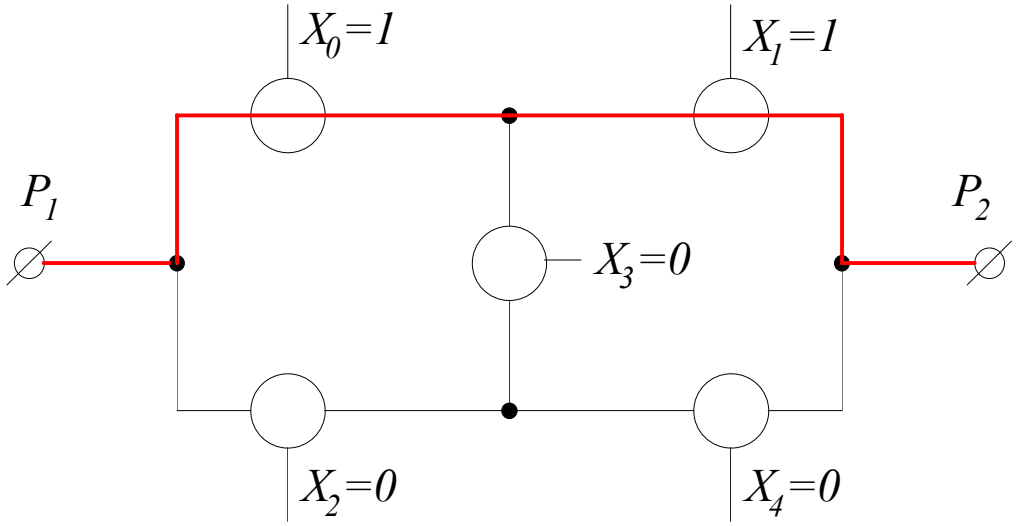
*p*МОП транзистор

Мостиковая схема

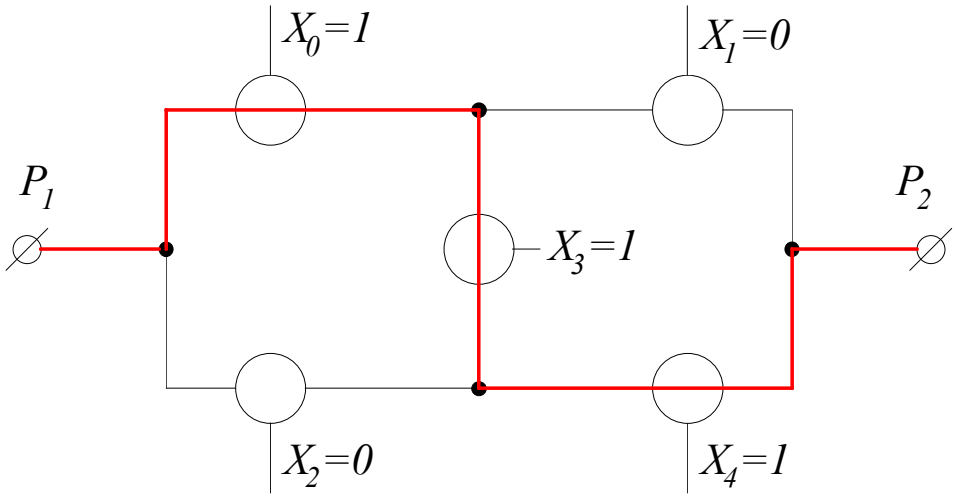




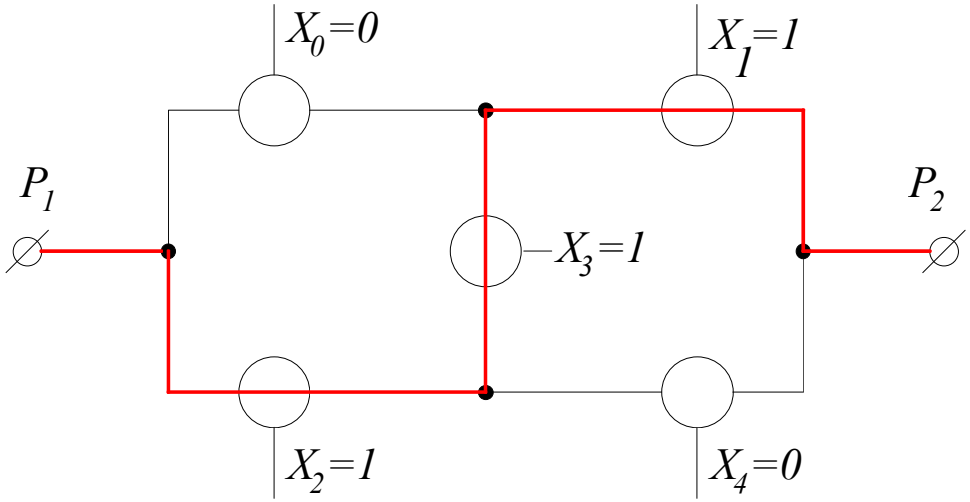
Проводимость = $X_2 X_4$



Проводимость = $X_0 X_1$



Проводимость = $X_0 X_3 X_4$

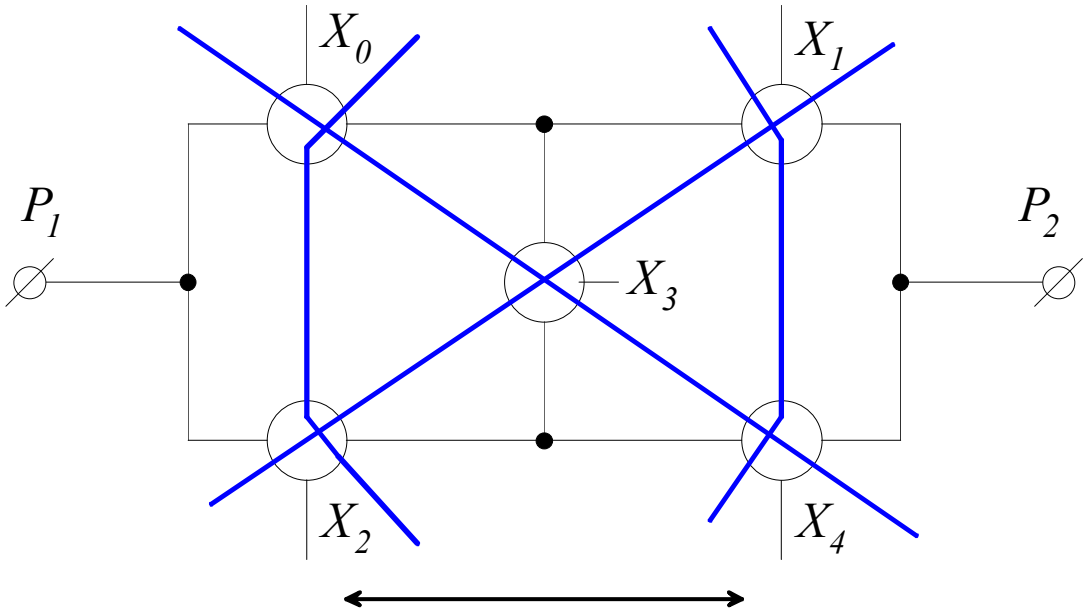


Проводимость = $X_1 X_2 X_3$

Множество соединений (цепей):

$$(P_1P_2) - \text{проводимость} = X_0X_1 + X_2X_4 + X_0X_3X_4 + X_1X_2X_3$$

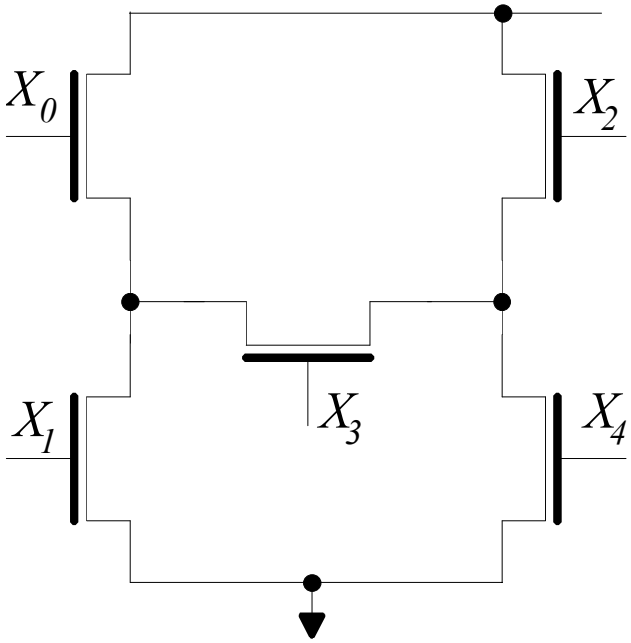
Множества разъединений:



$$(P_1P_2) - \text{проводимость} = (X_0 + X_2)(X_1 + X_4)(X_0 + X_3 + X_4)(X_1 + X_2 + X_3)$$

Мостиковая реализация функции

$$F(x_0, x_1, x_2, x_3, x_4) =$$
$$= x_0x_1 + x_2x_4 + x_0x_3x_4 + x_1x_2x_3$$

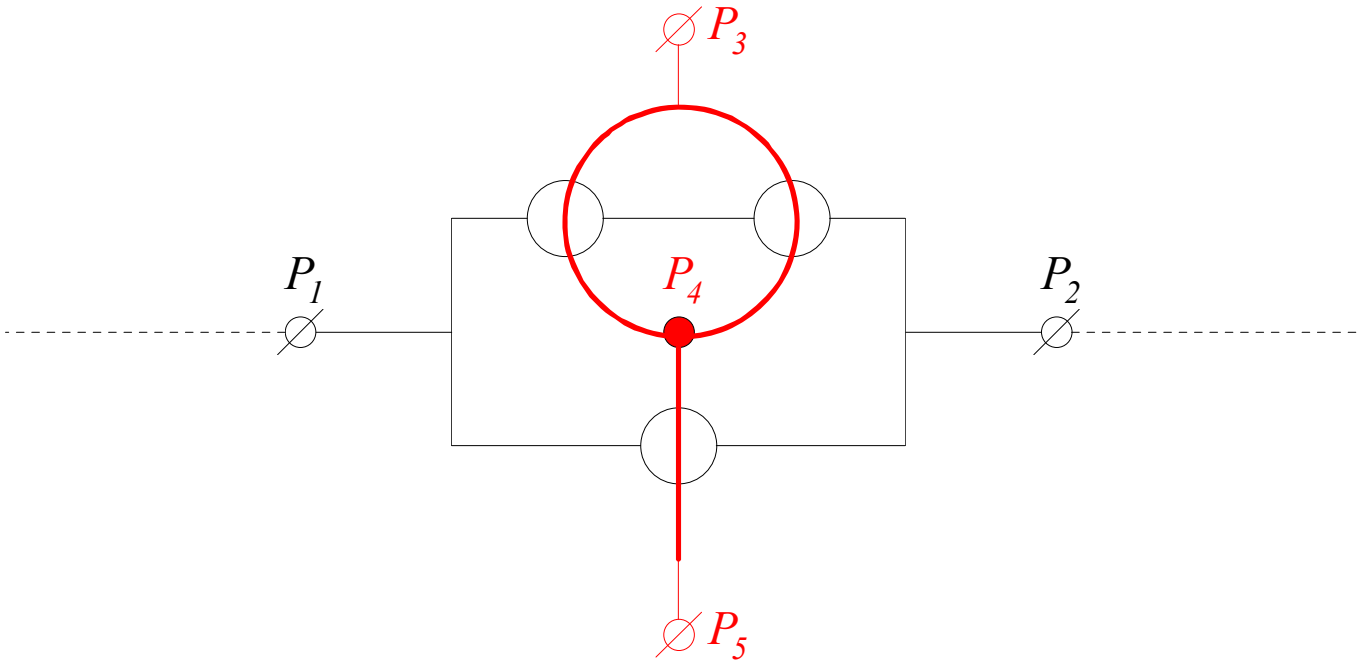


5 транзисторов

Инвертирование мостиковых схем

I

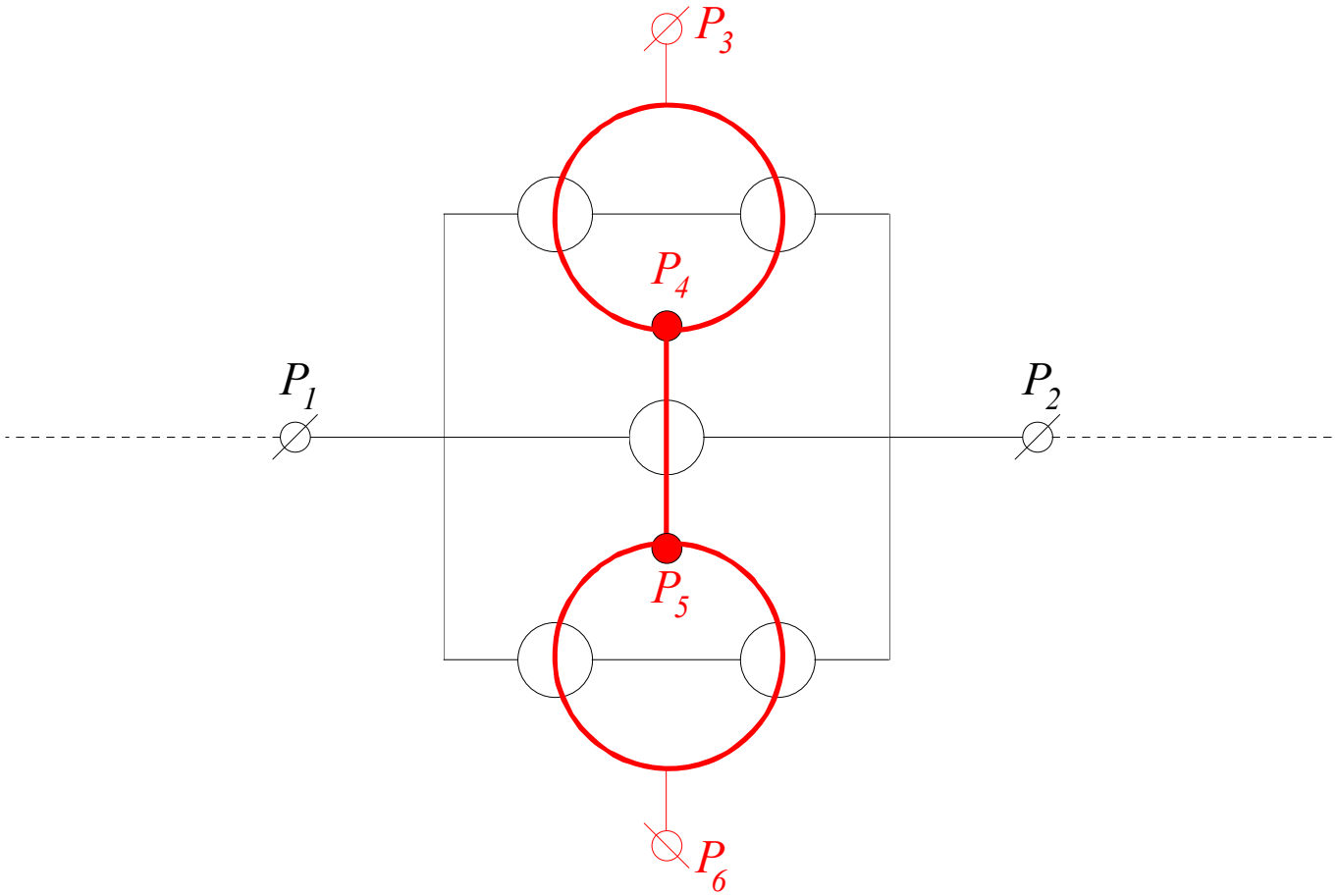
Инвертирование проводимости одного контура



Инвертирование мостиковых схем

II

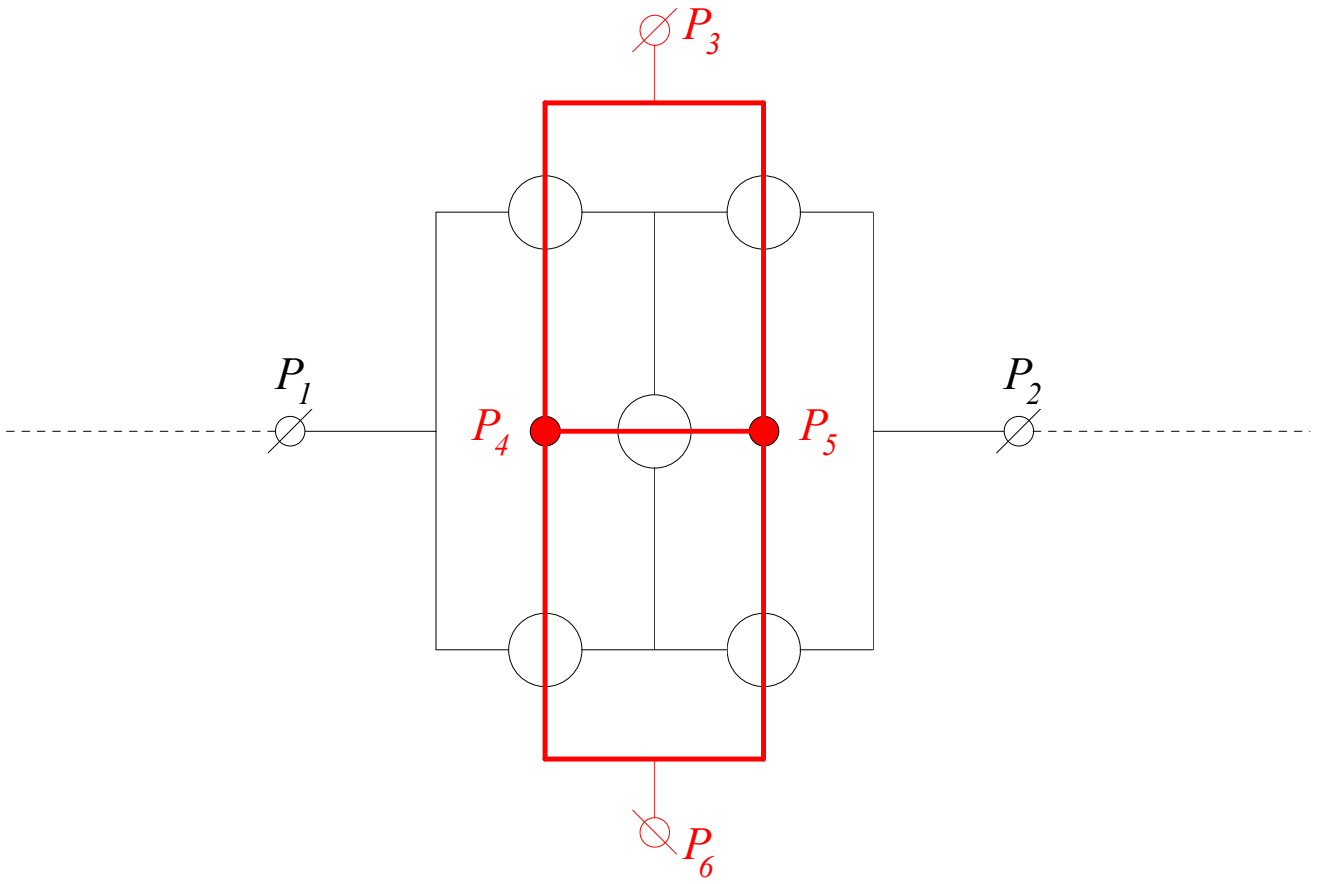
Инвертирование проводимости двух контуров



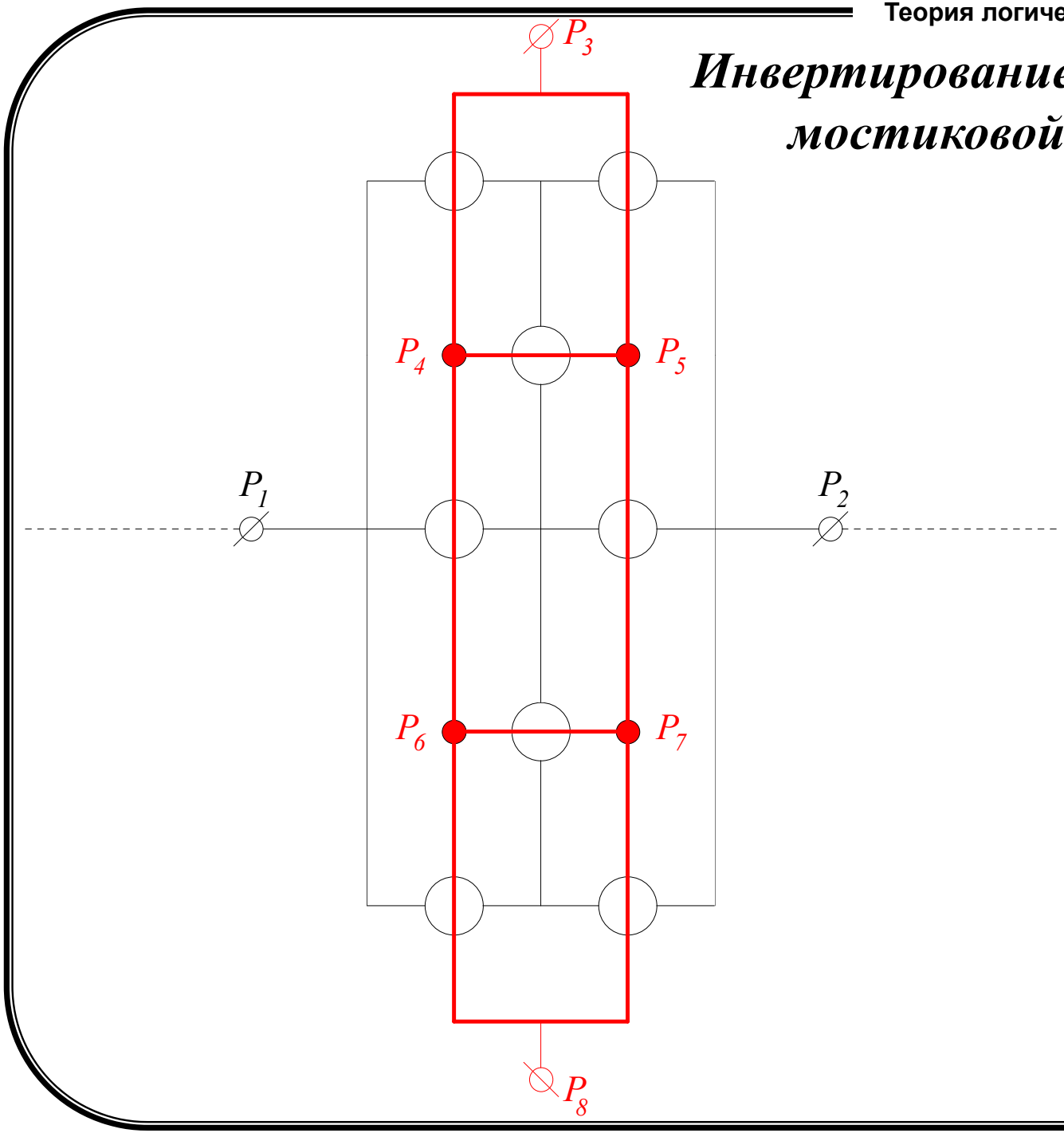
Инвертирование мостиковых схем

III

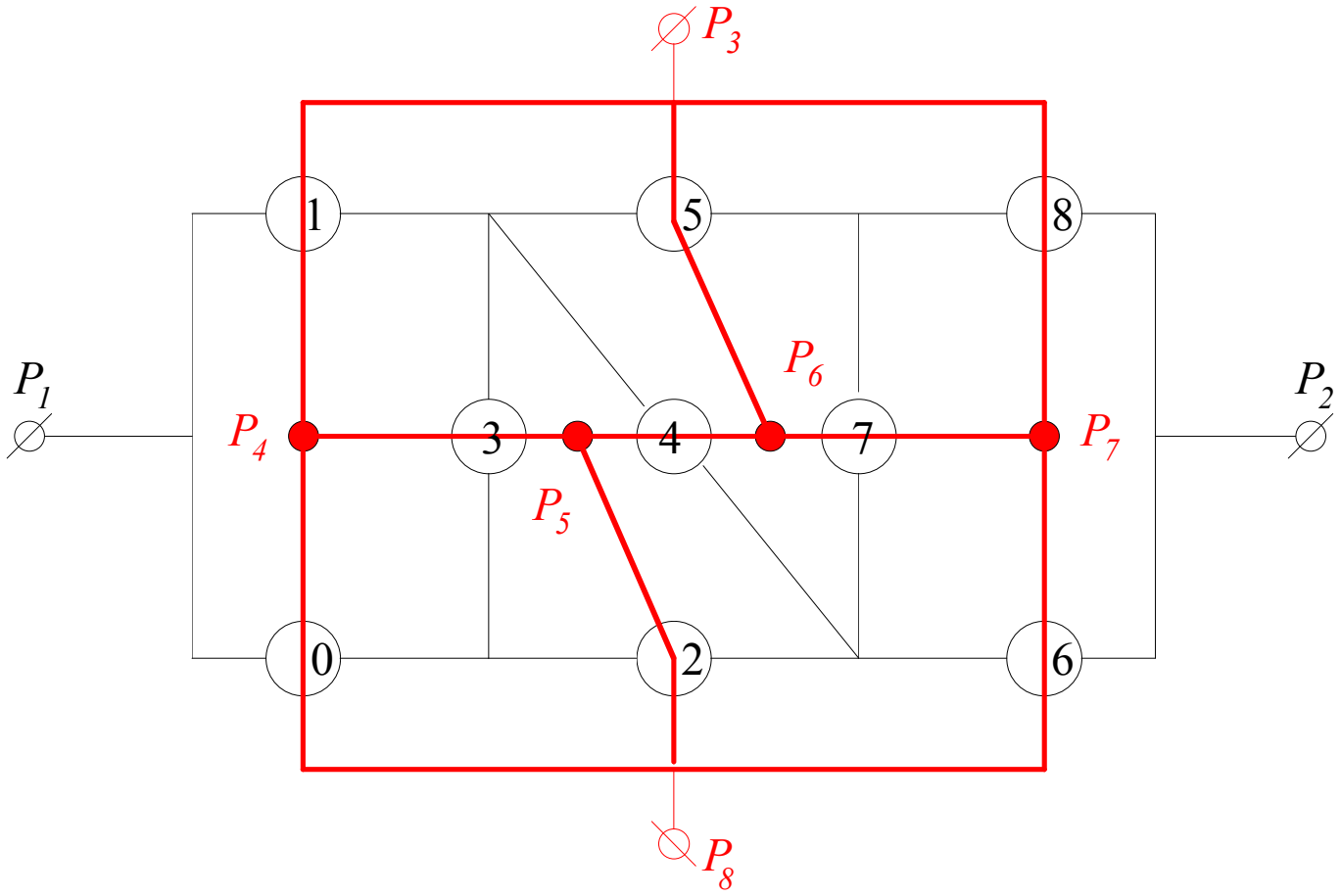
Инвертирование проводимости мостиковой структуры

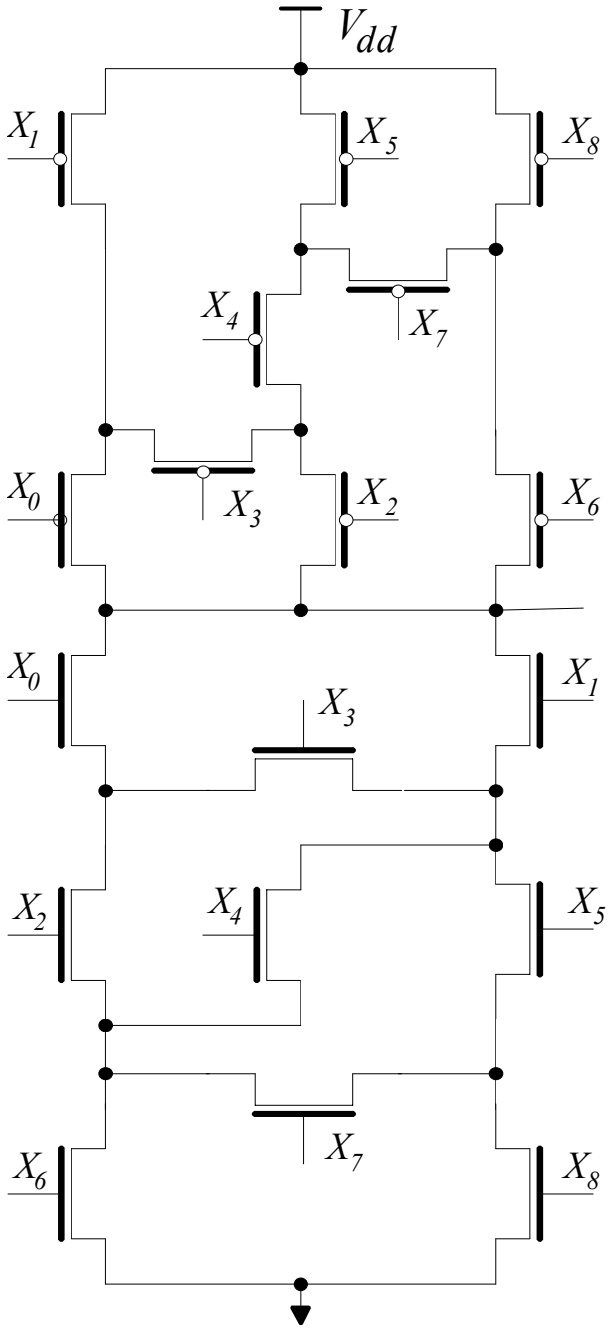


Инвертирование проводимости мостиковой структуры



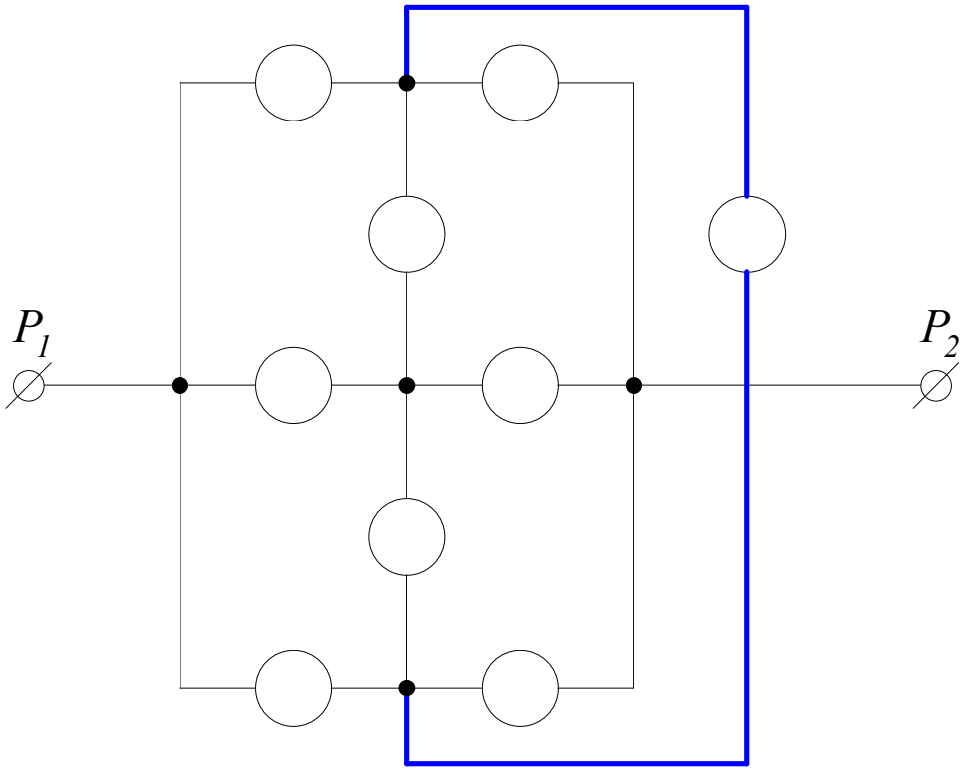
Инвертирование проводимости мостиковой структуры





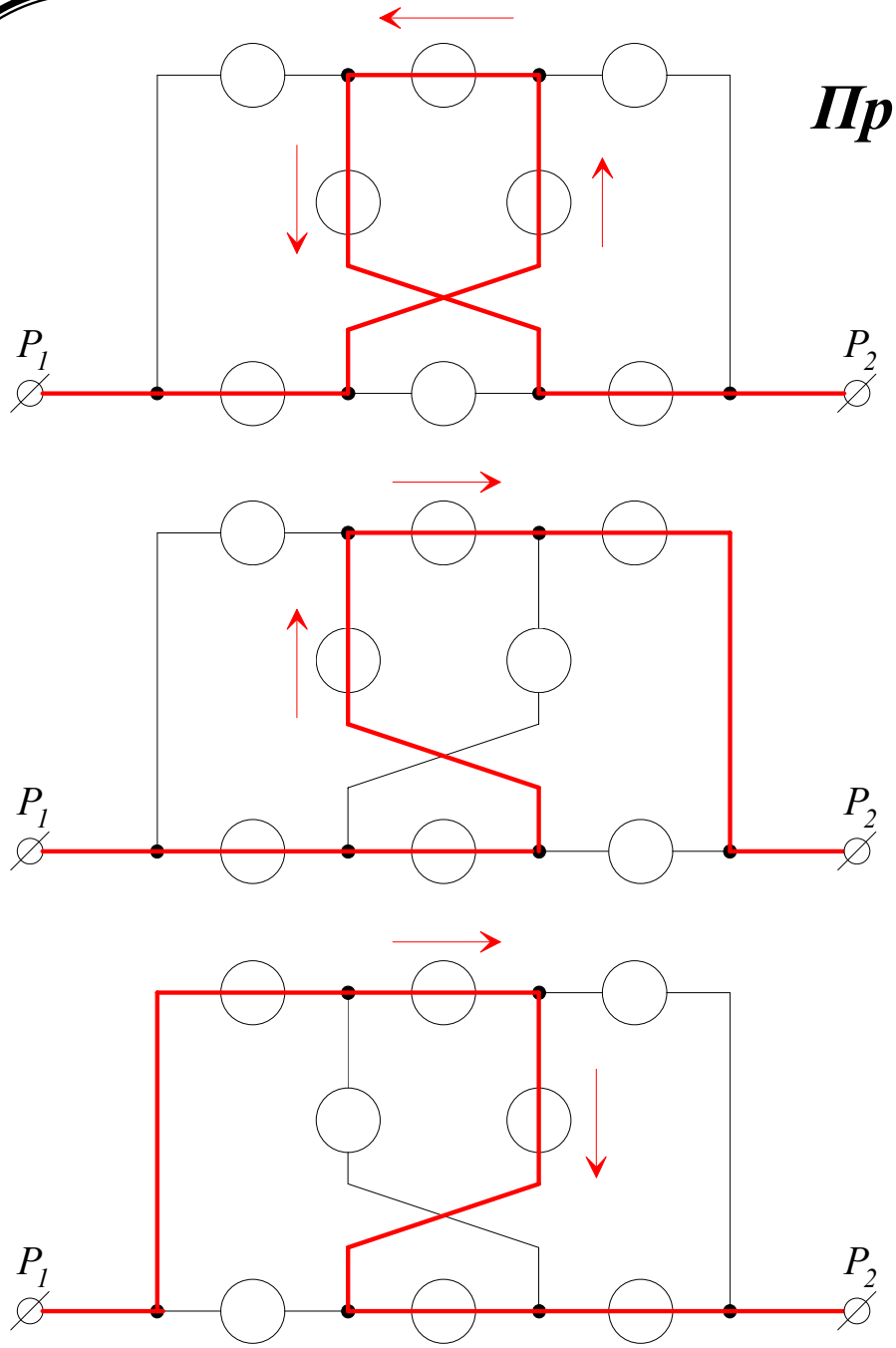
КМОП реализация мостиковой структуры, приведенной на слайде 32

Непланарные мостиковые структуры



Метод инвертирования непланарных мостиковых структур не известен

Пример возможной мостиковой структуры





Логическое Проектирование

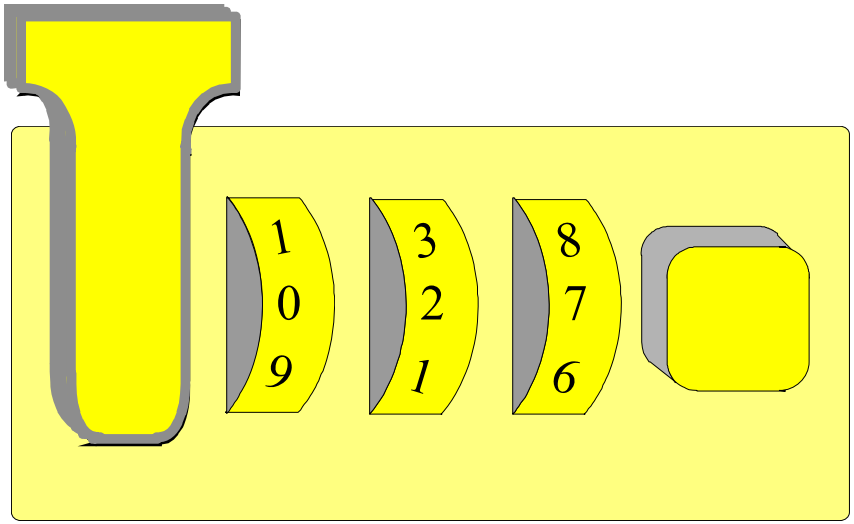
Лекция 9

Что такое комбинационные схемы?

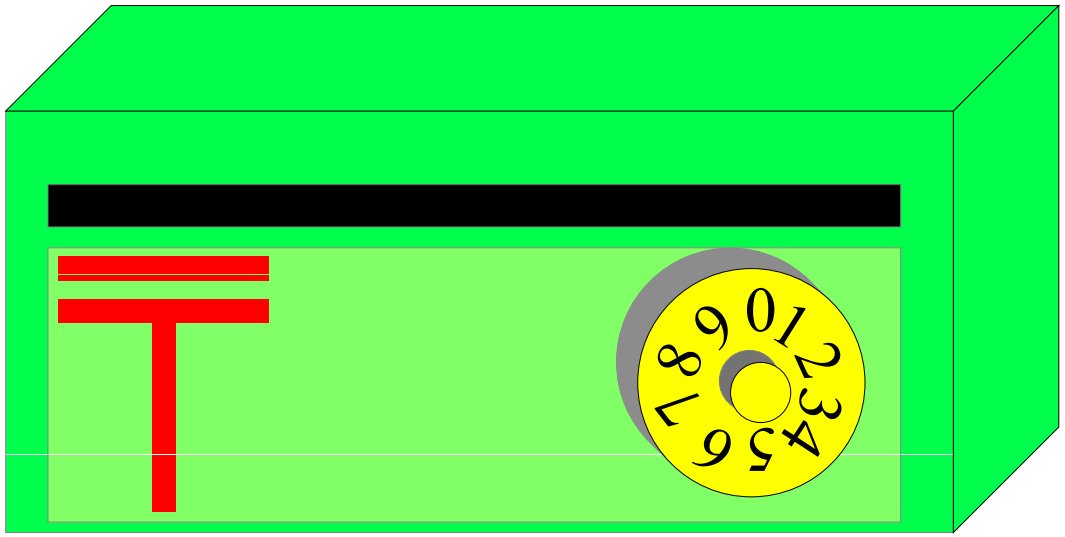
Определение: Схема называется комбинационной, если она реализует функцию, зависящую только от входных переменных; для каждого набора значений входных переменных существует единственное значение выхода схемы (устройство без памяти, или примитивный автомат).

- Схемы из логических элементов, топология которых не содержит циклов, являются комбинационными.
- Существуют комбинационные схемы с топологией, содержащей циклы [Kautz 1970].
- Разработаны алгоритмы проверки принадлежности логических схем классу комбинационных [Malik'94, Shiple'95], использующие многозначное моделирование.

Замок чемодана – комбинационное устройство



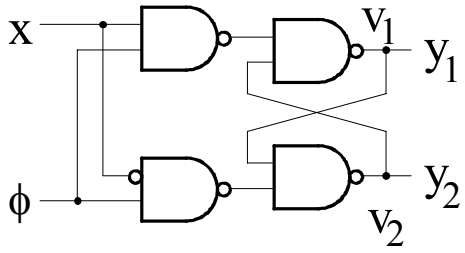
Замок почтового ящика – последовательностное устройство

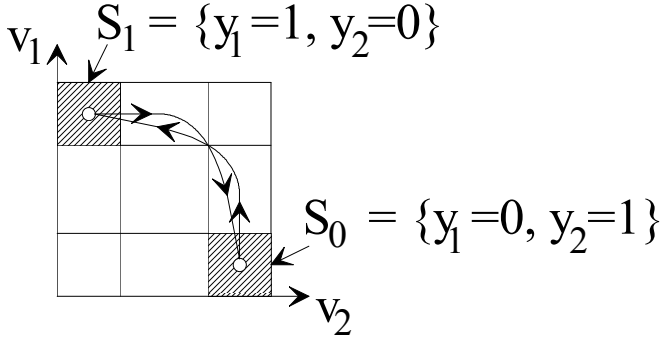


Что такое последовательностные схемы?

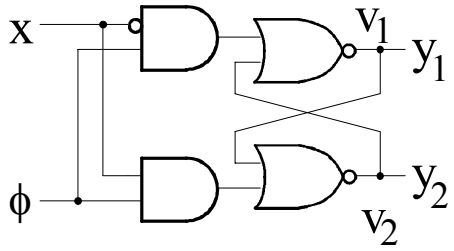
Элементарные устройства памяти (защелки) на базе простейших триггеров

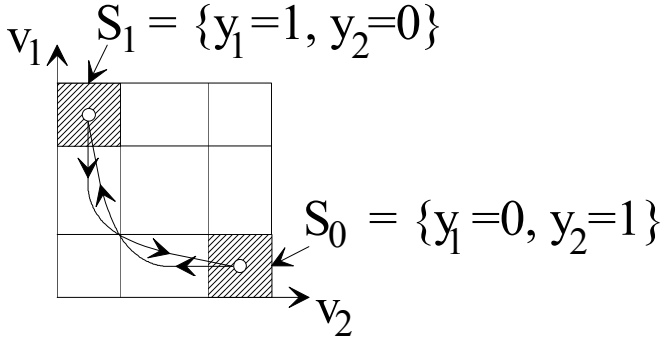
a)



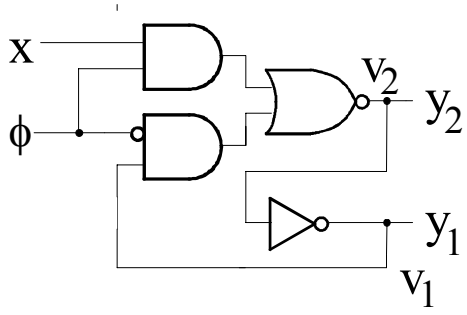


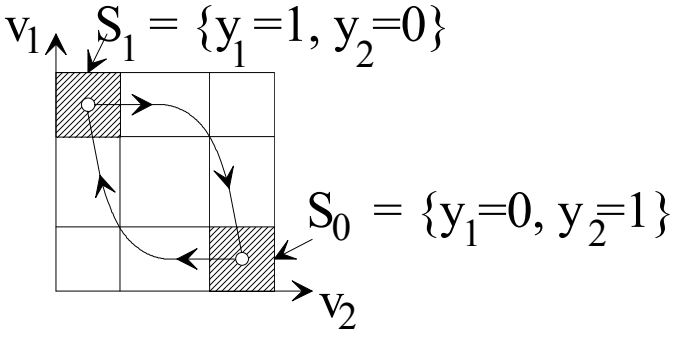
b)





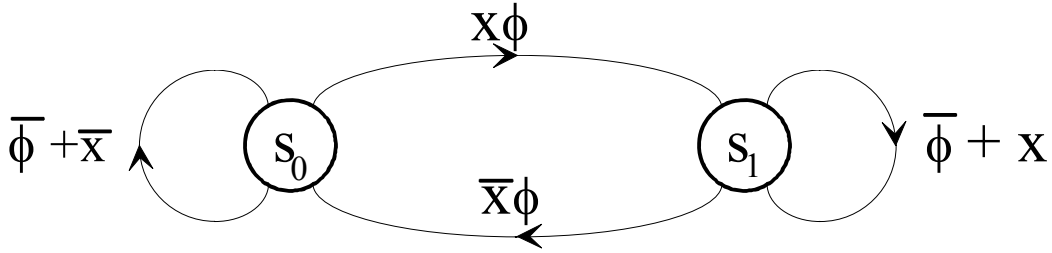
c)





Что такое последовательностные схемы?

Представление поведения защелки с помощью графа



Множество состояний входов:

$$I = \{i_1, i_2, i_3, i_4\}$$

$$i_1 = \{x = 0, \phi = 0\},\quad i_2 = \{x = 1, \phi = 0\},$$

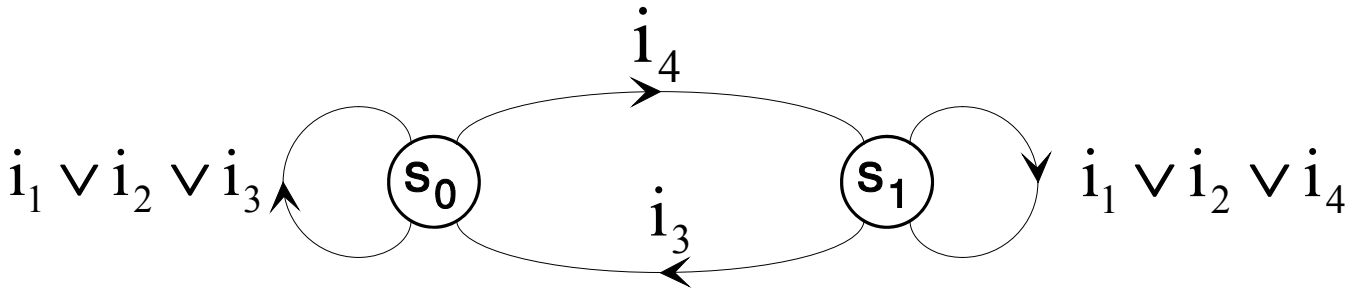
$$i_3 = \{x = 0, \phi = 1\},\quad i_4 = \{x = 1, \phi = 1\}$$

Множество внутренних состояний:

$$S = \{s_0, s_1\}$$

$$s_0 = \{y_1 = 0, y_2 = 1\},$$

$$s_1 = \{y_1 = 1, y_2 = 0\}$$



Что такое последовательностные схемы?

- В общем случае устройства (схемы) для обработки последовательностей сигналов содержат элементы памяти. Такие схемы с внутренней памятью называются *последовательностными*.
- Наличие обратных связей (циклов) является необходимым, но не достаточным условием того, чтобы схема была последовательностной.
- Двоичный последовательный сумматор является примером такой схемы:



Эта схема имеет два внутренних состояния (значения переноса). Ее выход зависит от входа и состояния сумматора.

- Математической моделью последовательностной схемы является ***конечный автомат***.

Конечный автомат является моделью

- ***физического устройства с внутренней конечной памятью;***
- ***алгоритма с конечной памятью, выполняющего операции со словами.***

Конечные автоматы (формальное определение)

Конечный автомат – это математическая модель дискретного устройства, представленная пятеркой:

$$A = < I, O, S, F, \Phi >$$

$I=\{i_1, i_2, \ldots, i_n\}$ – множество из n входных символов (входной алфавит);

$O=\{o_1, o_2, \ldots, o_m\}$ – множество из m выходных символов (выходной алфавит);

$S=\{s_0, s_1, \ldots, s_{r-1}\}$ – множество из r внутренних состояний (алфавит состояний);

F – отношение вида $F: I \times S \xrightarrow{F} O$ (F – ***функция выходов***);

Φ – отношение вида $\Phi : I \times S \xrightarrow{\Phi} S$ (Φ – ***функция переходов***).

Это определение *детерминированного автомата.* Автомат является *конечным,* так как числа n, m, and r конечны (конечные алфавиты входа, выхода и внутренних состояний).

Автомат называется *недетерминированным*, если *отношение* Φ не функционально: $\Phi : I \times S \to \mathrm{P}(S)$, где $P(S)$ означает множество подмножеств в алфавите S.

Конечные автоматы

Конечный автомат функционирует в дискретные моменты времени
t = 0, 1, 2, … .
Если $i(t)$, $o(t)$ и $s(t)$ – значения входа i, выхода o и состояния s в момент времени t, то поведение автомата может быть описано следующими *каноническими уравнениями*:

$$\begin{cases} o(t) = F(i(t), s(t)), \\ s(t) = \Phi(i(t), s(t-1)). \end{cases}$$

Здесь F – функция выхода и Φ – функция переходов. Дискретные моменты времени понимаются как шаги алгоритма, или такты, и никак не связаны с моментами реального времени. Эти уравнения определяют модель, которая называется ***автоматом Мили***.

Другая модель, называемая ***автоматом Мура***, определяется следующими уравнениями:

$$\begin{cases} o(t) = F(s(t)), \\ s(t) = \Phi(i(t), s(t-1)). \end{cases}$$

В этой модели значение функции выхода F в момент t зависит только от состояния автомата в тот же самый момент времени и не зависит от входа.

Конечные автоматы

Подстановка второго уравнения в первое уравнение в модели автомата Мура дает:

$$o(t) = F(s(t)) = F(\Phi(i(t), s(t-1))) = F''(i(t), s(t-1)).$$

Таким образом, автомат Мура может рассматриваться как частный случай автомата Мили с функцией выхода специального вида: $F'' = F(\Phi)$.

Для каждого автомата Мура существует эквивалентный ему автомат Мили и наоборот.

Два автомата называются эквивалентными, если для одной и той же входной последовательности $i(1)\ i(2)\ i(3)\ \ldots$ они производят одинаковые выходные последовательности $o(1)\ o(2)\ o(3)\ \ldots$. Понятие эквивалентности автоматов будет рассмотрено более детально позднее.

Обратное преобразование автомата Мили в автомат Мура ведет к увеличению числа состояний , однако условия переходов и функции выходов становятся более простыми.

Конечные автоматы

Конечный автомат может быть *инициальным* или *неинициальным*. Инициальный автомат всегда начинает работать из начального состояния, которое фиксировано. В неинициальном автомате начальное состояние не фиксировано и автомат может стартовать из любого состояния.

Другая интерпретация модели конечного автомата: Конечный автомат является последовательностной машиной, преобразующей входные последовательности символов (слов) из алфавита I

$$i_{a1} i_{a2} \ldots i_{at} \ldots$$

в выходные последовательности символов (слов) из алфавита O

$$o_{b1} o_{b2} \ldots o_{bt} \ldots$$

и создающей последовательности внутренних состояний из алфавита S

$$s_{c0} s_{c1} \ldots s_{c(t-1)} \ldots \ .$$

Если ввести обозначения $i_{at} \to i(t)$, $o_{bt} \to o(t)$ и $s_{c(t-1)} \to s(t-1)$, то в канонических автоматных уравнениях t можно трактовать как номер позиции символа в последовательности.

Спецификации конечного автомата

1. Функции Φ и F могут быть представлены ***таблицей переходов*** (состояний) и ***таблицей выходов.***

Пример: $I = \{i_1, i_2, i_3\}$, $O = \{o_1, o_2\}$, $S = \{s_0, s_1, s_2, s_3\}$ и функции F и Φ заданы таблицами:

Таблица переходов (Φ)

i \ S	s_0	s_1	s_2	s_3
i_1	s_1	s_2	s_2	s_1
i_2	s_0	s_0	s_2	s_0
i_3	s_0	s_0	s_2	s_3

Таблица выходов (F)

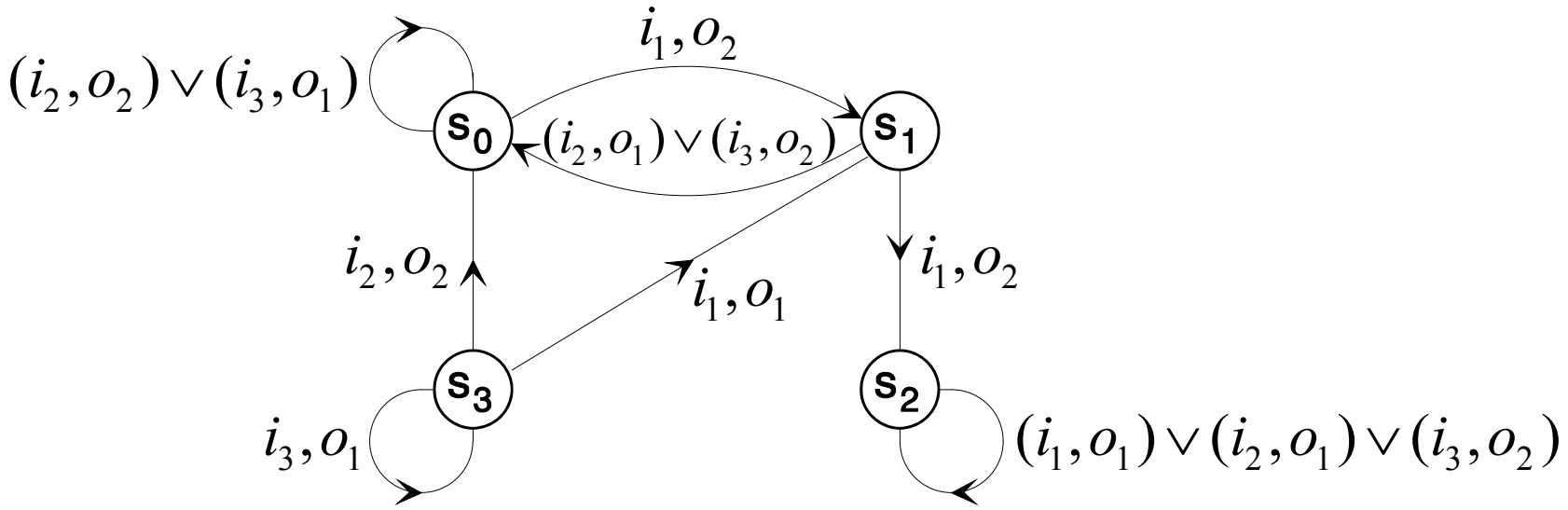
i \ S	s_0	s_1	s_2	s_3
i_1	o_2	o_2	o_1	o_1
i_2	o_2	o_1	o_1	o_2
i_3	o_1	o_2	o_2	o_1

Объединенная таблица переходов и выходов

i \ S	s_0	s_1	s_2	s_3
i_1	s_1, o_2	s_2, o_2	s_2, o_1	s_1, o_1
i_2	s_0, o_2	s_0, o_1	s_2, o_1	s_0, o_2
i_3	s_0, o_1	s_0, o_2	s_2, o_2	s_3, o_1

Спецификации конечного автомата

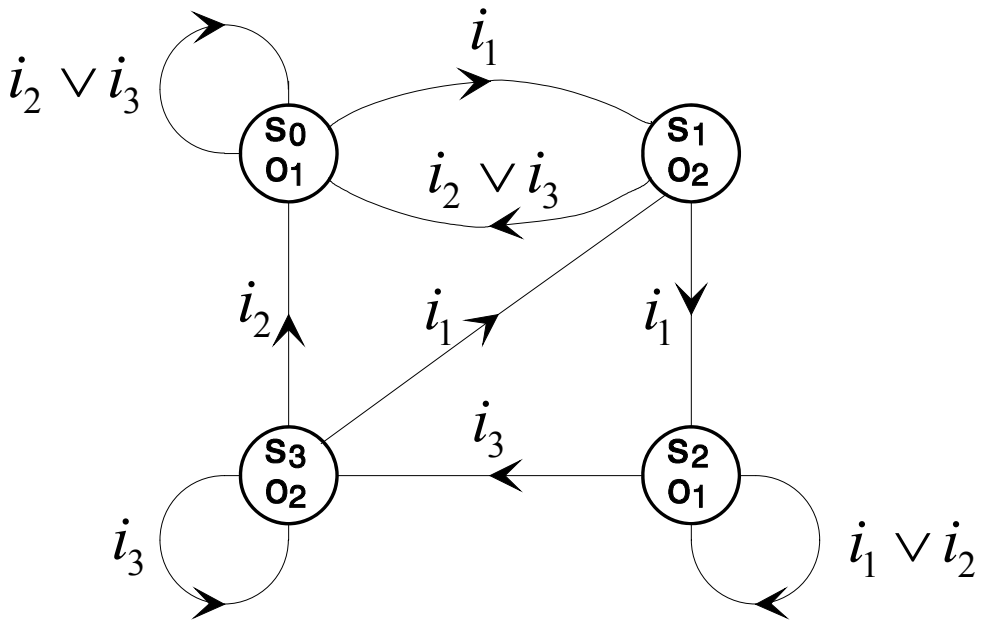
2. Функции Φ и F могут быть заданы ***графом переходов состояний*** (***S**tate **T**ransition **G**raph*, или ***STG***)



- Этот STG определяет автомат Мили. Дуги (переходы) помечены выражениями , определяющими условия переходов состояний и выходные символы.
- Графы переходов отличаются от таблиц переходов только визуальным представлением. STG более нагляден, чем таблица. Например, из графа легко видеть, что состояние S_2 является заключительным (дедлоком), а состояние S_3 является начальным.

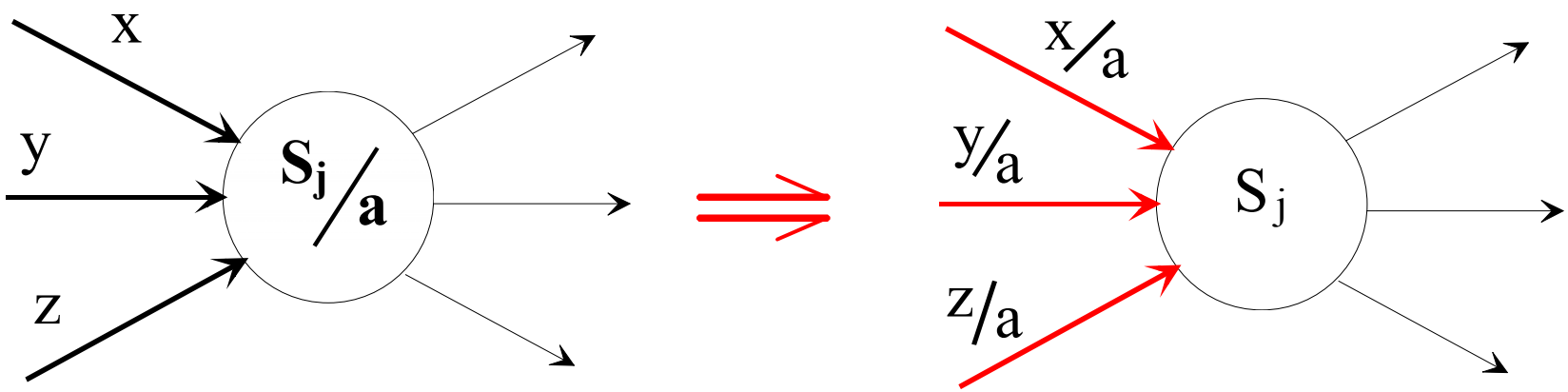
Спецификации конечного автомата

Граф переходов для автомата Мура



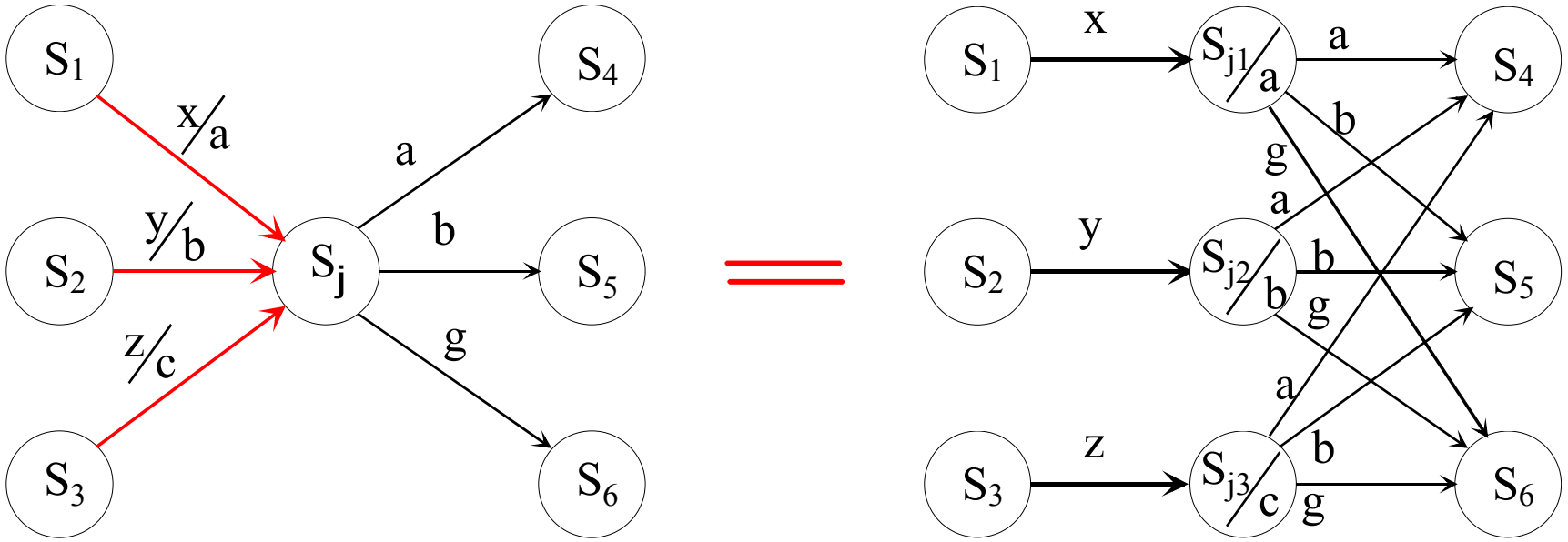
- Вершины обозначены символами состояний и выходных сигналов.
- Дуги (переходы) обозначены логическими функциями (предикатами) входных символов.
- Домашнее задание: построить автомат Мура, эквивалентный автомату Мили, представленному на слайде 11-13.

Преобразование автомата Мура в автомат Мили



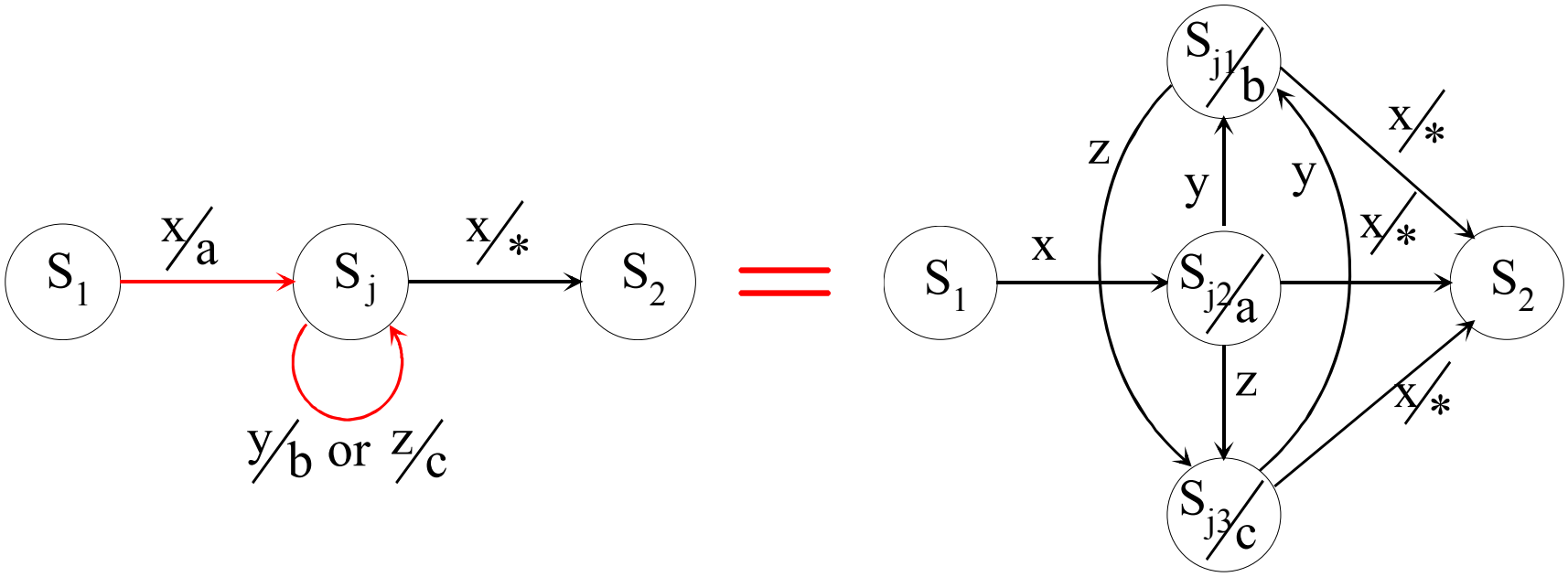
Выходные символы переносятся из вершин на входные дуги

Преобразование автомата Мили в автомат Мура



Скопировать вершину для каждого выходного символа на входных дугах и переместить выходные символы в соответствующие копии вершины.

Преобразование автомата Мили в автомат Мура



Спецификации конечного автомата

Матрица переходов (состояний)

Матрица переходов состояний является математическим аналогом STG. Для автомата с n состояниями матрица переходов имеет вид:

$$A = \left| a_{ij} \right|,\ i, j = 0,1,2,\ldots, n-1$$

$$a_{ij} = \begin{cases} b_{ij}, \text{если } s_i \xrightarrow{b_{ij}} s_j, \\ 0, \text{ если нет дуги, ведущей из } s_i \text{ в } s_j. \end{cases}$$

Example: Матрица переходов для автомата Мура на слайде №11-14:

$$A = \begin{vmatrix} b_{00} & b_{01} & 0 & 0 \\ b_{10} & 0 & b_{12} & 0 \\ 0 & 0 & b_{22} & b_{23} \\ b_{30} & b_{31} & 0 & b_{33} \end{vmatrix} = \begin{vmatrix} i_2 \vee i_3 & i_1 & 0 & 0 \\ i_2 \vee i_3 & 0 & i_1 & 0 \\ 0 & 0 & i_1 \vee i_2 & i_3 \\ i_2 & i_1 & 0 & i_3 \end{vmatrix}$$

i-ая строка содержит обозначения выходных дуг *i*-го состояния.

j-ый столбец содержит обозначения входных дуг j-го состояния.

Диагональные элементы содержат обозначения циклических дуг состояний.

Спецификации конечного автомата

Последовательность из ***k*** дуг, ведущая в STG из одного состояния в другое называется ***путем*** и ***k*** – ***длина пути***. Матрица переходов порядка ***k*** определяется как $A^k = \left|b_{ij}\right|^k = \left|b_{ij}\right|^{k-1} \times \left|b_{ij}\right| = \left|c_{ij}\right|,$

где element c_{ij} содержит все пути длины k, ведущие из состояния i в состояние j.

Пример:

$$A^2 = \begin{vmatrix} b_{00} & b_{01} & 0 & 0 \\ b_{10} & 0 & b_{12} & 0 \\ 0 & 0 & b_{22} & b_{23} \\ b_{30} & b_{31} & 0 & b_{33} \end{vmatrix}^2 = \begin{vmatrix} b_{00} & b_{01} & 0 & 0 \\ b_{10} & 0 & b_{12} & 0 \\ 0 & 0 & b_{22} & b_{23} \\ b_{30} & b_{31} & 0 & b_{33} \end{vmatrix} \times \begin{vmatrix} b_{00} & b_{01} & 0 & 0 \\ b_{10} & 0 & b_{12} & 0 \\ 0 & 0 & b_{22} & b_{23} \\ b_{30} & b_{31} & 0 & b_{33} \end{vmatrix} =$$

$$= \begin{vmatrix} b_{00}b_{00} + b_{10}b_{10} & b_{00}b_{01} & b_{01}b_{12} & 0 \\ b_{10}b_{00} & b_{10}b_{01} & b_{12}b_{22} & b_{12}b_{23} \\ b_{23}b_{30} & b_{23}b_{31} & b_{22}b_{22} & b_{23}b_{33} \\ b_{30}b_{00} + b_{31}b_{10} + b_{33}b_{30} & b_{30}b_{01} + b_{33}b_{31} & b_{31}b_{12} & b_{33}b_{33} \end{vmatrix}$$

Спецификации конечного автомата

Если желательно определить только количество путей длины k, ведущих из одного состояния автомата в другое, то можно использовать порождающую матрицу переходов (матрицу инцидентности):

$$E = \left| e_{ij} \right|, \; i, j = 0,1,2,\ldots, n-1$$

$$e_{ij} = \begin{cases} 1, \text{если } s_i \xrightarrow{b_{ij}} s_j, \\ 0, \text{если отсутствует дуга, ведущая из } s_i \text{ в } s_j. \end{cases}$$

Для того, чтобы найти все числа путей длины k, необходимо вычислить k-ю степень этой матрицы.

Пример: Вторая степень порождающей матрицы для автомата Мура, граф которого приведен на слайде №11-14, имеет вид:

$$E^2 = \begin{vmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{vmatrix}^2 = \begin{vmatrix} 2 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 3 & 2 & 1 & 1 \end{vmatrix}$$

Спецификации конечного автомата

Представление в виде функций

Для автомата Мура:

$$S(k) = F[S(k-1), I(k)]$$
$$O(k) = \Phi[S(k)]$$

Для автомата Мили:

$$S(k) = F[S(k-1), I(k)]$$
$$O(k) = \Phi[S(k), I(k)]$$

Свойство марковости:

Вся предыстория (входов) автомата связана с одним и только одним внутренним состоянием

$$S(k) = F_1[S(0), I(1), I(2), \ldots, I(k)],$$

$$S(k) = F_2[S(k-1), I(k)].$$

Пусть Ξ – вектор текущего состояния

$$\Xi = \left| \sigma_0 \quad \sigma_1 \quad \ldots \quad \sigma_{n-1} \right|$$

$$\sigma_j = \begin{cases} \mathbf{1} \ \ \textbf{if} \ \ \boldsymbol{S(k) = s_j} \\ \mathbf{0} \ \ \textbf{if} \ \ \boldsymbol{S(k) \neq s_j} \end{cases}$$

тогда

$$\Xi(k) = \Xi(k-1) \times M(I)$$

где

$$\sigma_j(k) = \sum_i \sigma_i(k-1) m_{ij}$$

$$\begin{vmatrix} 0 & 1 & 0 & 0 \end{vmatrix} \times \begin{vmatrix} 0 & x_0 & 0 & x_1 \\ 0 & x_1 & 0 & x_0 \\ x_0 & 0 & x_1 & 0 \\ x_1 & 0 & x_0 & 0 \end{vmatrix} = \begin{vmatrix} 0 & x_1 & 0 & x_0 \end{vmatrix}$$

$$\begin{vmatrix} \sigma_0 & \sigma_1 & \sigma_2 & \sigma_3 \end{vmatrix} \times \begin{vmatrix} 0 & x_0 & 0 & x_1 \\ 0 & x_1 & 0 & x_0 \\ x_0 & 0 & x_1 & 0 \\ x_1 & 0 & x_0 & 0 \end{vmatrix} =$$

$$\begin{vmatrix} \sigma_2 x_0 + \sigma_3 x_1 & \sigma_0 x_0 + \sigma_1 x_1 & \sigma_3 x_0 + \sigma_2 x_1 & \sigma_1 x_0 + \sigma_0 x_1 \end{vmatrix}$$

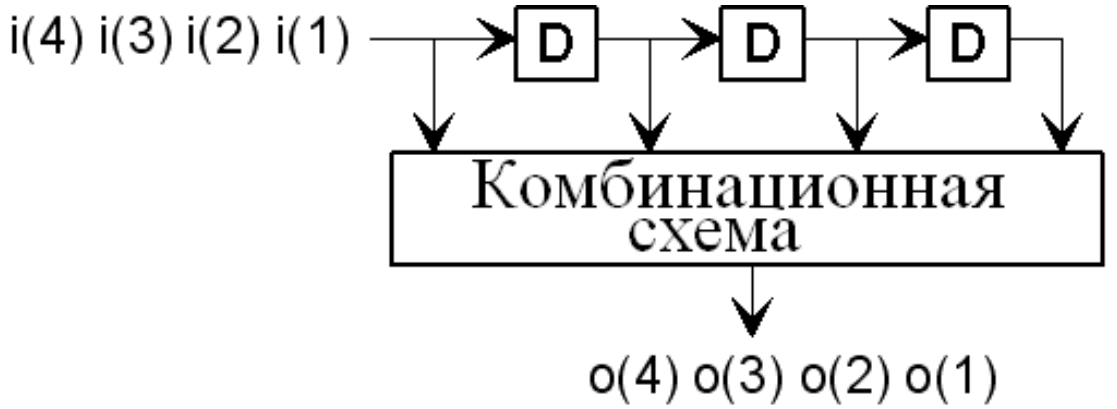
Распознавание автоматом свойств конечных последовательностей

Конечный автомат может быть использован для распознавания некоторых свойств конечных фрагментов во входной последовательности.

Пример: Пусть имеется автомат с начальным состоянием S_0 и бинарным входом. Автомат должен информировать нас, если число, представленное фрагментом входной последовательности из 4 битов, делится на 3.

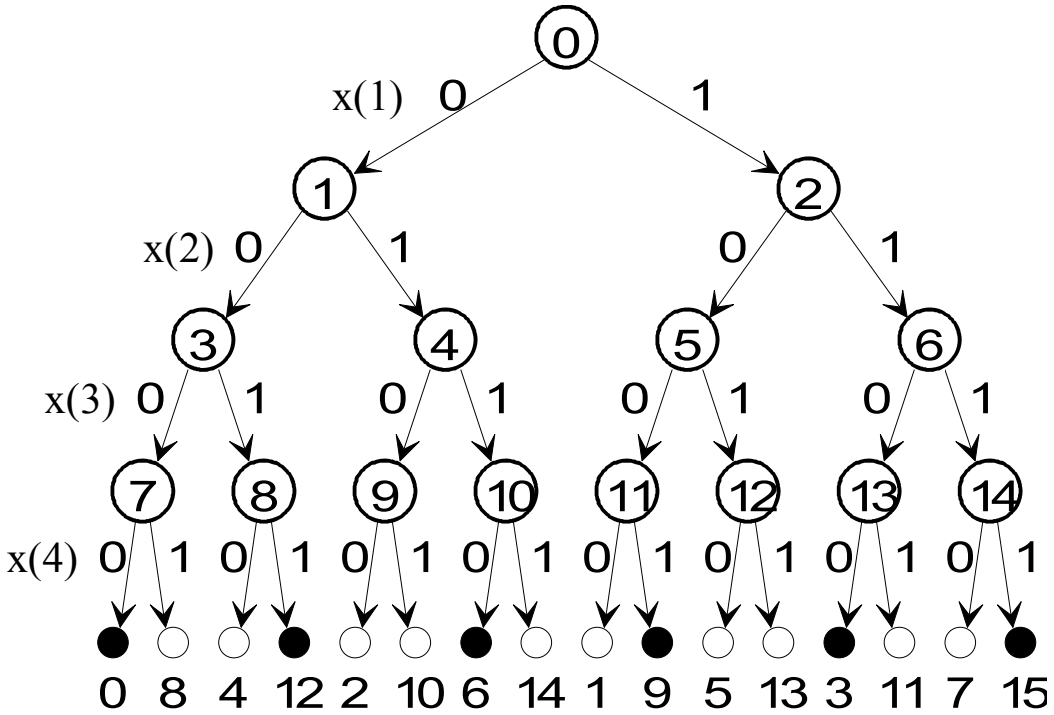
Фрагменты	Свойство	Фрагменты	Свойство
(0) 0000	1	(13) 1011	0
(1) 1000	0	(14) 0111	0
(2) 0100	0	(15) 1111	1
(3) 1100	1		
(4) 0010	0		
(5) 1010	0		
(6) 0110	1		
(7) 1110	0		
(8) 0001	0		
(9) 1001	1		
(10) 0101	0		
(11) 1101	0		
(12) 0011	1		

Возможная структура автомата:

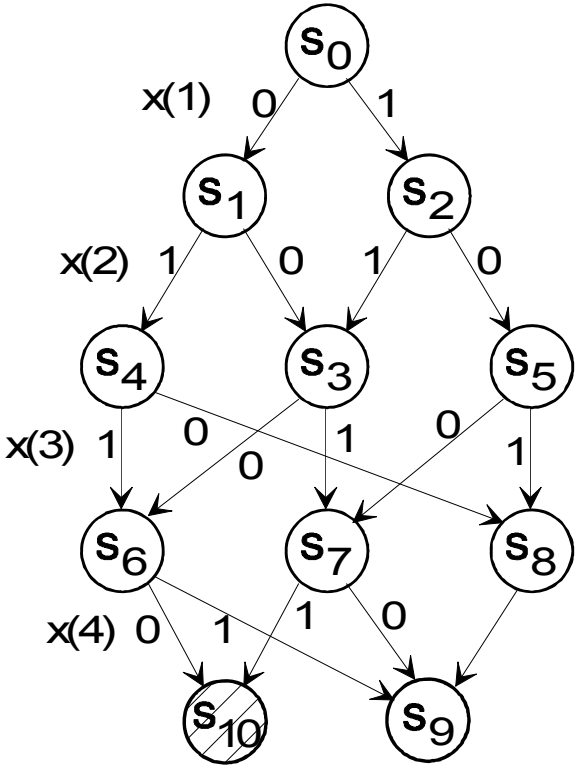


Распознавание автоматом свойств конечных последовательностей

Представление автомата деревом



Граф переходов автомата



$$s_0 = 0,\ s_1 = 1,\ s_2 = 2,\ s_3 = 3 \vee 6,\ s_4 = 4,\ s_5 = 5,$$

$$s_6 = 7 \vee 10 \vee 13;\ s_7 = 8 \vee 11 \vee 14,\ s_8 = 9 \vee 12.$$

Распознавание автоматом свойств неограниченных последовательностей

Язык регулярных выражений

Язык регулярных выражений является инструментом для представления событий на входе конечного автомата.



Событие E – это некоторая последовательность значений входа автомата (входное слово) или некоторое множество таких последовательностей (слов).

Алфавит регулярных выражений $A = \{a_1, a_2, ..., a_n\}$ – множество символов, совпадающих с символами значений входа автомата $I = \{i_0, i_1, ..., i_{n-1}\}$.

Слово: Слово – это символ $w = a_j$.

Если $\boldsymbol{w}$ – слово, то $\boldsymbol{wa_i}$ тоже является словом (элементарная конкатенация).

b, bo, boy, boyc, boyco, boycot, boycott

$\varnothing$ – пустое слово, $\varnothing \boldsymbol{w} = \boldsymbol{w} \varnothing = \boldsymbol{w}$.

Множество слов – объединение, или интеграция:

$$W = \{w_1, w_2, ..., w_k\} = w_1 \vee w_2 \vee ... \vee w_k$$

Событие – это множество слов $\boldsymbol{E = W}$.

Λ - пустое событие (пустое множество слов),

$$\Lambda \vee E = E \vee \Lambda = E.$$

Конкатенация слов:

$$w_1(w_2 \vee w_3) = w_1 w_2 \vee w_1 w_3$$

$$(w_2 \vee w_3) w_1 = w_2 w_1 \vee w_3 w_1$$

$$w_i w_j \neq w_j w_i \ !$$

$$\Lambda w = w \Lambda = \Lambda$$

$$(\text{cow} \vee \text{play})\text{boy} = \text{cowboy} \vee \text{playboy}$$

$$\text{boy}(\text{cott} \vee \text{scout}) = \text{boycott} \vee \text{boyscout}$$

$$\text{boy}(\text{cot} \vee \text{scou})\text{t} = \text{boycott} \vee \text{boyscout}$$

Итерация

$$a^* = \varnothing \vee a \vee aa \vee aaa \vee aaaa \vee .... \vee aa..aaa \vee ....$$

$$(ab)^* = \varnothing \vee ab \vee abab \vee ababab \vee ....$$

$$(w)^* = \varnothing \vee w \vee ww \vee www \vee wwww \vee ....$$

$$(a \vee b)^* = \varnothing \vee a \vee b \vee aa \vee ab \vee ba \vee bb \vee aaa \vee ...$$

$(A)^* =$ множество всех возможных слов в алфавите A

$(\Lambda)^* = \varnothing$ (по определению)

$$(\varnothing)^* = \varnothing$$

$$(a^* \vee b^*)^* = (a^* b^*)^* = (a \vee b)^*$$

Итак, имеется ***алфавит*** $A=\{a_1,a_2,...,a_n\}$.

Множество всех слов в алфавите A обозначается как A^*.
Множество A^* включает ***пустое слово*** $\varnothing$.

Любое подмножество слов E множества A^* $(E \subseteq A^*)$ называется ***событием***.

Необходимо ввести пустое событие Λ.

Пусть E_1 и E_2 – события в алфавите A. Определены три операции над событиями:

1. ***Интеграция*** $E_1 \vee E_2$ (нормальная интеграция множеств);
2. ***Конкатенация*** (умножение) $E_1 \cdot E_2$: $E = E_1E_2$ – множество всех слов следующего вида:

$$\alpha_1\alpha_2, \alpha_1 \in E_1, \alpha_2 \in E_2.$$

3. ***Итерация***

$$E^* = \Lambda \vee E \vee EE \vee EEE \vee ... \vee E^n \vee ... = \bigcup_{i=0}^{\infty} E^i.$$

Алфавит символов $a_i \in A$ и пустое слово $\varnothing$ называются ***элементарными событиями***.

Событие называется ***регулярным,*** если оно может быть получено из элементарных событий с помощью этих трех операций. Таким образом, любое регулярное событие может быть описано формулой, которая называется ***регулярным выражением***.

Формальное определение регулярных выражений может быть дано рекурсивно:

1. Элементарные события – регулярные выражения;

2. Если E_1 и E_2 – регулярные выражения, то $E_1 \vee E_2$, $E_1 E_2$, E_1^*, E_2^* – регулярные выражения;

3. Выражение является регулярным, если и только если оно получено посредством конечного применения правил 1 и 2.

Одно и то же регулярное событие может быть представлено различными регулярными выражениями.

Свойства операций $\vee, \cdot, \{\ \}^*$:

$$E \vee \Lambda = \Lambda \vee E = E, \qquad E \vee E = E,$$

$$E \cdot \Lambda = \Lambda \cdot E = \Lambda, \qquad E^* = \varnothing \vee E \vee E^*,$$

$$E \cdot \varnothing = \varnothing \cdot E = E, \qquad E^* \vee E = E^*,$$

$$\{\Lambda\}^* = \varnothing, \qquad E^* \cdot E = E \cdot E^*,$$

$$\{\varnothing\}^* = \varnothing, \qquad E^* \cdot E^* = E^*, \quad \{E^*\}^* = E^*,$$

$$\{E_1^* \vee E_2^*\}^* = \{E_1^* E_2^*\}^* = \{E_1 \vee E_2\}^*.$$

Примеры регулярных выражений в алфавите {a,b,c}:

1. $E = (a \vee b \vee c) \cdot (a \vee b \vee c) \cdot (a \vee b \vee c)$
2. $E = ab^*c = \{ac \vee abc \vee abbc \vee \ldots \vee abb\ldots bc \vee \ldots\}$
3. $E = \{a \vee b \vee c\}^*$
4. $E = (a \vee c) \cdot \{a \vee b \vee c\}^* \cdot ab$

- Все конечные события регулярны.

- Использование итерации порождает бесконечные регулярные события. Большинство бесконечных событий нерегулярны. Множество всех регулярных событий в непустом алфавите счетно, а множество всех событий в том же алфавите имеет мощность континуума.

- Будем говорить, что автомат представляет регулярное событие, если он производит специальный выходной сигнал (автомат Мили) или попадает в специальное состояние (автомат Мура) каждый раз, когда он получает входную последовательность, принадлежащую регулярному событию.

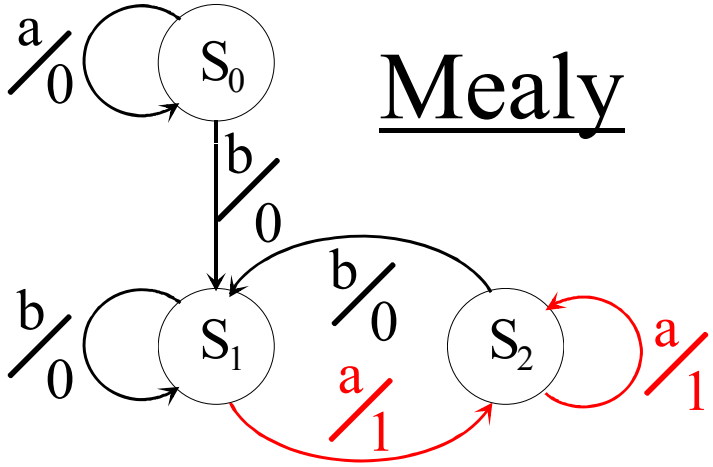
Теорема (Клини): ***Класс событий, представимых конечным автоматом совпадает с классом регулярных событий.***

Как построить граф переходов автомата, представляющего заданное регулярное событие?

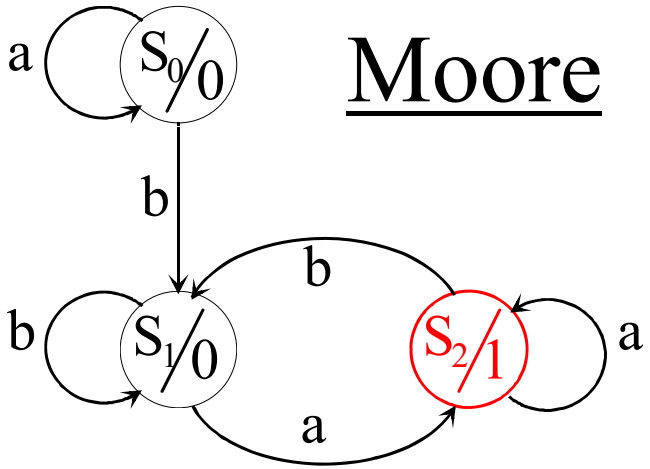
Прежде, чем ответить на этот вопрос, рассмотрим пример.

$$E = baa^* = ba \vee baa \vee baaa \vee \ldots.$$

$$aabbaaabaabbbaa\ldots\ldots$$
$$\downarrow\downarrow\downarrow\ \downarrow\ \ \downarrow\downarrow$$
$$1\,1\,1\ \ 1\ \ \ 1\,1$$



Mealy



Moore

Существуют два подхода к пониманию распознавания автоматом регулярных выражений

I. Непрерывное распознавание: *автомат вырабатывает распознающий сигнал каждый раз, как только встречается регулярное событие.*

$$E = bob \vee bill$$

$$\begin{array}{l} bbiblbo\,b\,o\,b\,il\,l\,iioobo\,b \\ \qquad\qquad\quad\ \downarrow\ \downarrow\ \downarrow\qquad\quad\ \downarrow \\ \qquad\qquad\quad\ 1\ \ 1\ \ 1\qquad\quad\ 1 \end{array}$$

II. Распознавание с начального состояния: *автомат вырабатывает распознающие сигналы, если и только если входная последовательность принадлежит регулярному событию.*

$$E = ta(-ta)^* = ta \vee ta - ta \vee ta - ta - ta \vee \ldots$$

$$\begin{array}{l} t\,a - t\,a - t\,a - t\,a\,tta - ta - ta\ldots \\ \ \downarrow\ \ \downarrow\ \ \downarrow\ \ \downarrow \\ \ 1\ \ 1\ \ 1\ \ 1 \end{array}$$



Логическое Проектирование

Лекция 10

Нахождение графа переходов состояний автомата, соответствующего регулярному выражению

(прямой алгоритм Мак Нотона-Ямады)

Поясним работу алгоритма на примере построения графа переходов автомата Мура, распознающего регулярное выражение:

$$E = \{a \vee b\}^* c \vee bc^* a$$

Первый шаг – разметка формулы

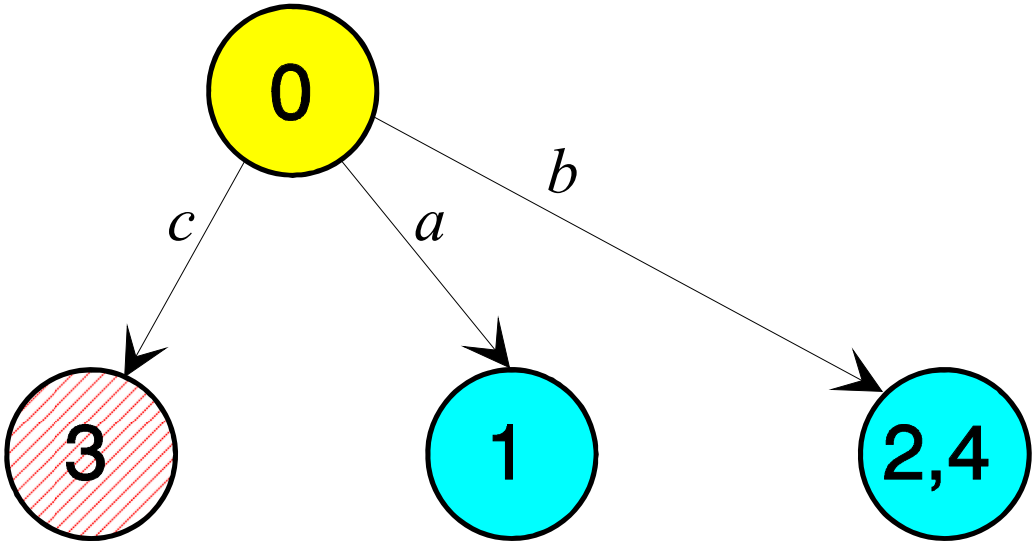
$$\underset{0}{|}(\{\boldsymbol{a}\underset{1}{|}\vee \boldsymbol{b}\underset{2}{|}\}^{*}\boldsymbol{c}\underset{3}{|}\vee \boldsymbol{b}\underset{4}{|}\boldsymbol{c}\underset{5}{|}^{*}\boldsymbol{a}\underset{6}{|}).$$

Каждому вхождению символа в выражение поставим в соответствие место непосредственно справа от символа. Все места пронумеруем (1, 2, 3, ...) слева направо. Место «0» соответствует начальному состоянию автомата $\boldsymbol{S}_0$.

Разворачивая выражение в слова, можно наблюдать смену мест (переходы). Место, которое соответствует первому символу слова, принадлежащему *E*, назовем начальным. (В данном случае места 1, 2, 3 и 4 являются начальными.) Место, которое соответствует последнему символу слова, принадлежащему *E*, назовем конечным. (Для рассматриваемого примера места 3 и 6 являются конечными.)

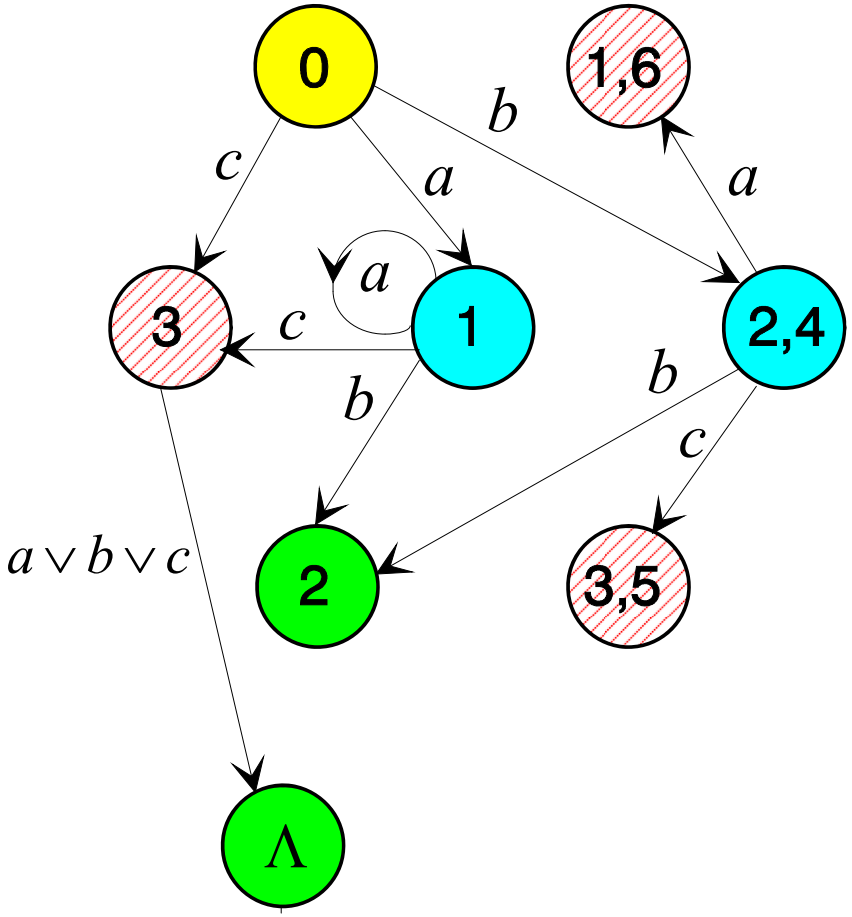
Второй шаг — построение переходов из начального состояния в состояния второго уровня.

$$\underset{0}{|}\underset{1}{(}\underset{2}{\{}\boldsymbol{a}\underset{3}{|}\vee\boldsymbol{b}\underset{4}{|}\}^{*}\boldsymbol{c}\underset{5}{|}\vee\boldsymbol{b}\underset{6}{|}\boldsymbol{c}|^{*}\boldsymbol{a}|)$$



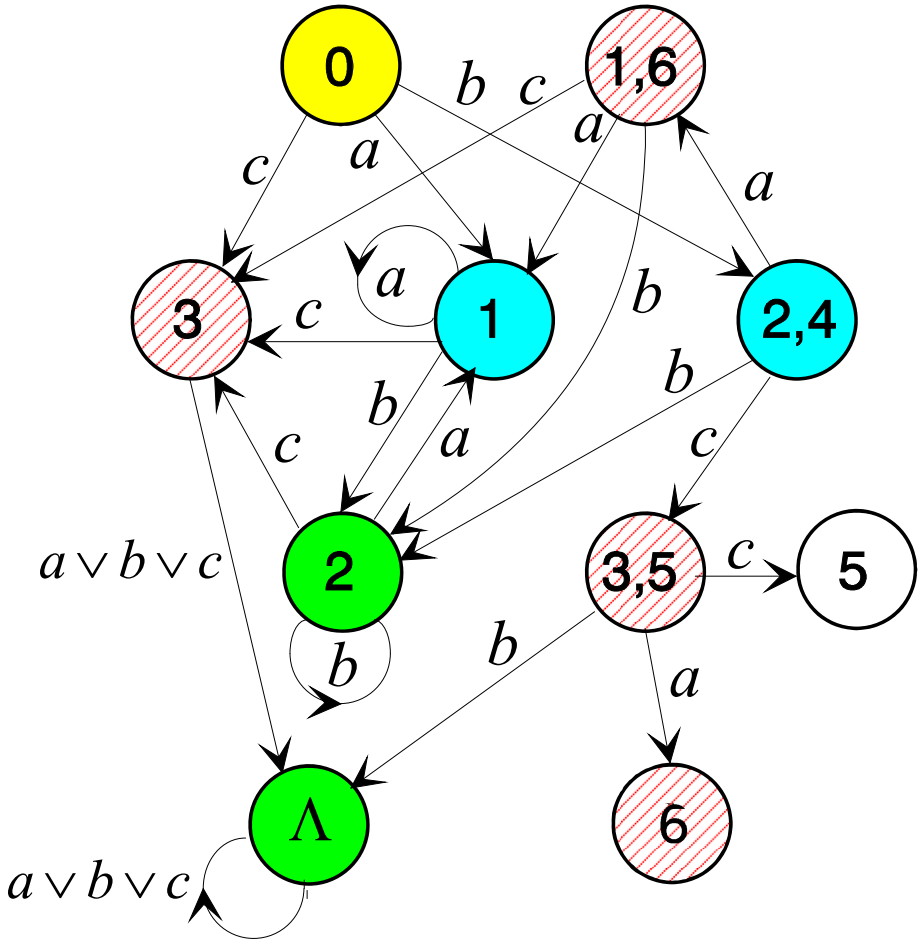
3 шаг – построение переходов следующего уровня.

$$\underset{0}{|}\underset{1}{(}\underset{2}{\{}\boldsymbol{a}\underset{3}{|}\vee\boldsymbol{b}\underset{4}{|}\}^{*}\boldsymbol{c}\underset{5}{|}\vee\boldsymbol{b}\underset{6}{|}\boldsymbol{c}|^{*}\boldsymbol{a}|)$$



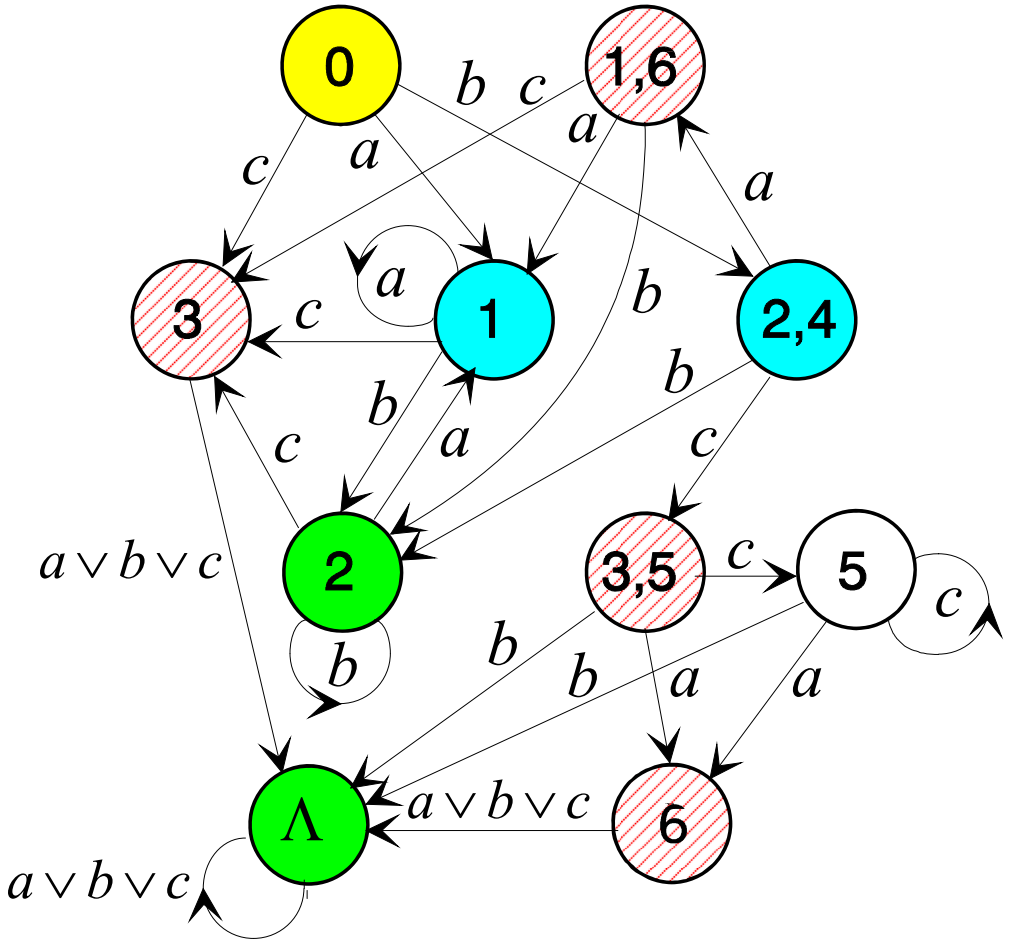
4 шаг – следующие переходы.

$$\underset{0}{|}(\{\boldsymbol{a}\underset{1}{|}\vee \boldsymbol{b}\underset{2}{|}\}^{*}\boldsymbol{c}\underset{3}{|}\vee \boldsymbol{b}\underset{4}{|}\boldsymbol{c}\underset{5}{|}^{*}\boldsymbol{a}\underset{6}{|})$$

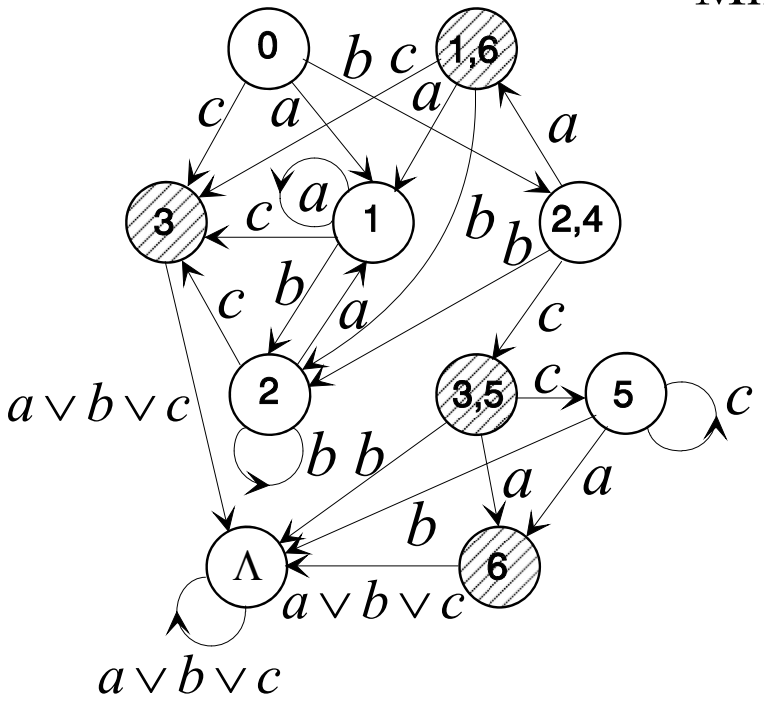


5 шаг – следующие переходы (заключительный).

$$\underset{0}{|}\underset{1}{(}\underset{2}{\{}\underset{3}{a}\underset{4}{|}\vee\underset{5}{b}\underset{6}{|}\}^{*}c|\vee b|c|^{*}a|)$$



Для заданного регулярного выражения граф переходов автомата Мура имеет вид:



Множества мест → Состояние/Выход

$$\{0\} \to s_0 / 0$$
$$\{3\} \to s_1 / 1$$
$$\{3,5\} \to s_2 / 1$$
$$\{1\} \to s_3 / 0$$
$$\{1,6\} \to s_4 / 1$$
$$\{2,4\} \to s_5 / 0$$
$$\{2\} \to s_6 / 0$$
$$\{5\} \to s_7 / 0$$
$$\{6\} \to s_8 / 1$$
$$\Lambda \to s_9 / 0$$

Верхний предел для числа состояний автомата, распознающего регулярное выражение с p местами, равен $2^p + 1$.

Домашнее задание: Пожалуйста, постройте граф переходов автомата Мура, распознающего регулярные выражения:

$$E_1 = ab^* c \vee bc^* a, \quad E_2 = \{a \vee b\}^* b.$$

Последовательность мест называется *допустимой,* если она соответствует слову, принадлежащему событию, описываемому регулярным выражением.

Граф переходов автомата Мура может быть построен следующим образом:

1. Алгоритм начинается из начального состояния автомата, соответствующего месту 0.
2. Если мы находимся в месте *i* регулярного выражения, найти для каждого символа входного алфавита подмножество мест, в которое символ ведет из места *i*.
3. Поставить в соответствие подмножеству мест состояние автомата; различные подмножества обозначаются различными состояниями. Пустое множество мест обозначается специальным состоянием.
4. Включить в граф переход, ведущий из состояния, содержащего место *i* , в это состояние и обозначить переход входным символом.
5. Если состояние содержит хотя бы одно конечное место, оно порождает выходной сигнал, равный 1, в противном случае, сигнал, равный 0.
6. Любой входной символ переводит пустое состояние в себя.

Обратный алгоритм Мак Нотона-Ямады

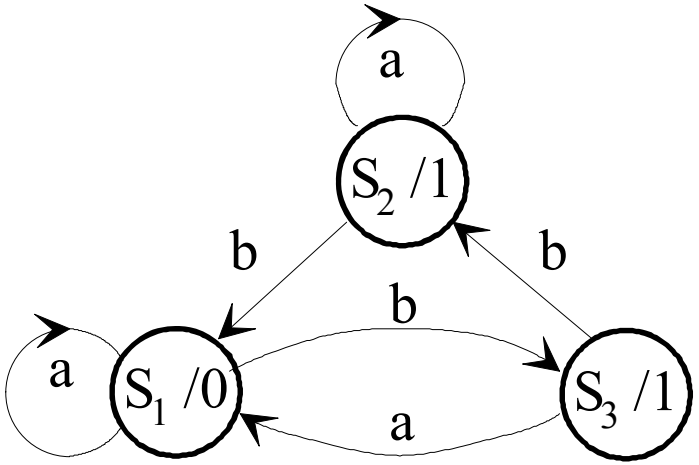
(для нахождения регулярного выражения, распознаваемого конечным автоматом)

Пусть имеется STG (State Transition Graph) автомата Мура. Автомат является инициальным и имеет n состояний . Среди этих состояний p состояний с номерами $m_1, m_2, \ldots, m_l, \ldots, m_p$ $(m_l \in \{1,2,\ldots,n\})$ помечены выходным сигналом, равным 1.

Алгоритм подразумевает получение вспомогательных регулярных выражений, которые строятся методом индукции следующим образом.

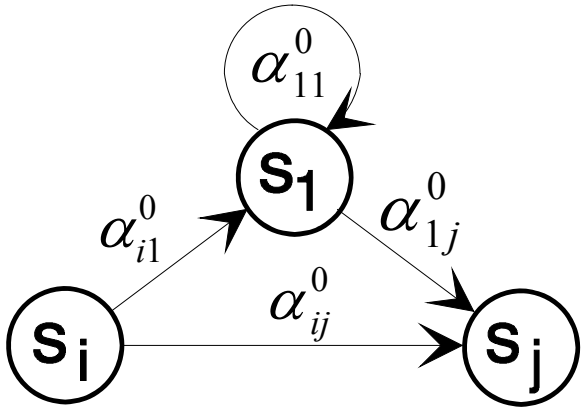
1. Пусть α_{ij}^{0} – объединение входных символов (регулярное выражение), которые непосредственно переводят состояние S_i в состояние S_j. Если нет перехода из S_i в S_j, то $\alpha_{ij}^{0} = \Lambda$, α_{ij}^{0} – элемент матрицы связей $\left|\alpha_{ij}^{0}\right|$.

Пример:



$$\left|\alpha_{ij}^{0}\right| = \begin{vmatrix} a & \Lambda & b \\ b & a & \Lambda \\ a & b & \Lambda \end{vmatrix}$$

2. Пусть α_{ij}^{1} – регулярное выражение, переводящее автомат из состояния S_i в состояние S_j и разрешающее прохождение только через состояние S_1.



$$\alpha_{ij}^{1} = \alpha_{ij}^{0} \vee \alpha_{i1}^{0} \{\alpha_{11}^{0}\}^{*} \alpha_{1j}^{0}$$

Пример:

$$\alpha_{ij}^{1} = \alpha_{ij}^{0} \vee \alpha_{i1}^{0} \{\alpha_{11}^{0}\}^{*} \alpha_{1j}^{0}$$

$$\left|\alpha_{ij}^{0}\right| = \begin{vmatrix} a & \Lambda & b \\ b & a & \Lambda \\ a & b & \Lambda \end{vmatrix}$$

$$\alpha_{11}^{1} = a \vee a\{a\}^{*} a = a^{*} a$$

$$\alpha_{12}^{1} = \Lambda \vee a\{a\}^{*} \Lambda = \Lambda$$

$$\alpha_{13}^{1} = b \vee a\{a\}^{*} b = a^{*} b$$

$$\alpha_{21}^{1} = b \vee b\{a\}^{*} a = ba^{*}$$

$$\alpha_{22}^{1} = a \vee b\{a\}^{*} \Lambda = a$$

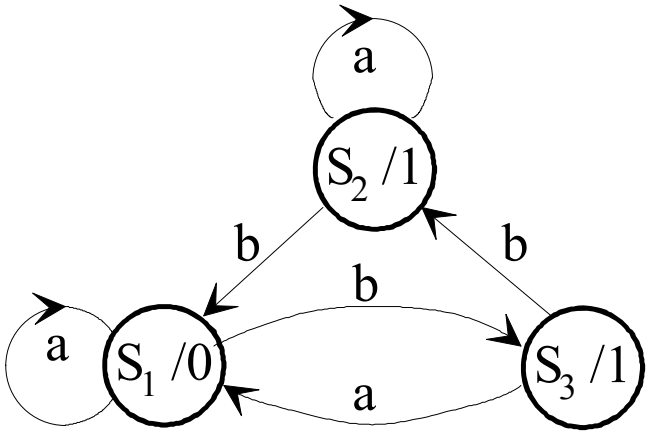
$$\alpha_{23}^{1} = \Lambda \vee b\{a\}^{*} b = ba^{*} b$$

$$\alpha_{31}^{1} = a \vee a\{a\}^{*} a = a^{*} a$$

$$\alpha_{32}^{1} = b \vee a\{a\}^{*} \Lambda = b$$

$$\alpha_{33}^{1} = \Lambda \vee a\{a\}^{*} b = aa^{*} b$$

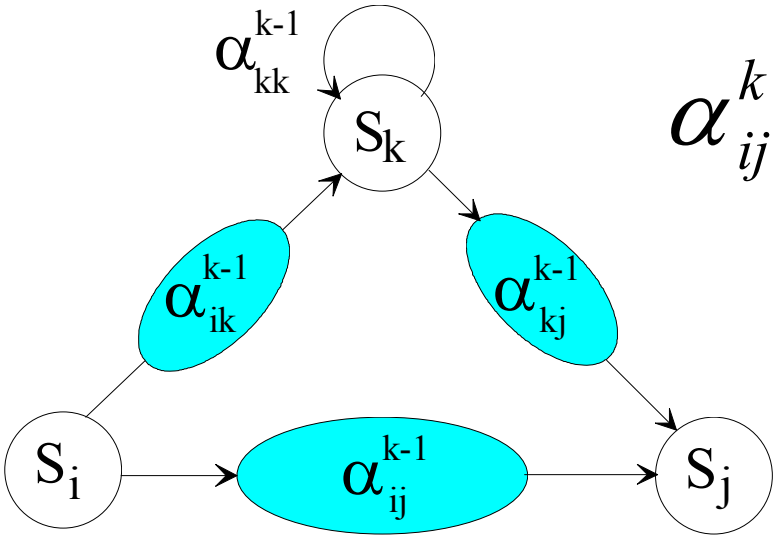
Пример:



$$\left|\alpha_{ij}^{0}\right| = \begin{vmatrix} a & \Lambda & b \\ b & a & \Lambda \\ a & b & \Lambda \end{vmatrix}$$

$$\left|\alpha_{ij}^{1}\right| = \begin{vmatrix} a^*a & \Lambda & a^*b \\ ba^* & a & ba^*b \\ a^*a & b & aa^*b \end{vmatrix}$$

3. Пусть α_{ij}^{k} – регулярное выражение, переводящее автомат из состояния S_i в состояние S_j и разрешающее прохождение только через состояния $S_1, S_2, \ldots, S_k$ $(k \le n)$.



$$\alpha_{ij}^{k} = \alpha_{ij}^{k-1} \vee \alpha_{ik}^{k-1} \{\alpha_{kk}^{k-1}\}^{*} \alpha_{kj}^{k-1}$$

Пример:

$$\alpha_{11}^2 = \alpha_{11}^1 \vee \alpha_{12}^1 \{\alpha_{22}^1\}^* \alpha_{21}^1 = a^* a \vee \Lambda = a^* a,$$

$$\alpha_{12}^2 = \alpha_{12}^1 \vee \alpha_{12}^1 \{\alpha_{22}^1\}^* \alpha_{22}^1 = \Lambda,$$

$$\alpha_{13}^2 = \alpha_{13}^1 \vee \alpha_{12}^1 \{\alpha_{22}^1\}^* \alpha_{23}^1 = a^* b \vee \Lambda = a^* b,$$

$$\alpha_{21}^2 = \alpha_{21}^1 \vee \alpha_{22}^1 \{\alpha_{22}^1\}^* \alpha_{21}^1 = ba^* \vee aa^* ba^* = a^* ba^*,$$

$$\alpha_{22}^2 = \alpha_{22}^1 \vee \alpha_{22}^1 \{\alpha_{22}^1\}^* \alpha_{22}^1 = a \vee aa^* a = a^* a,$$

$$\alpha_{23}^2 = \alpha_{23}^1 \vee \alpha_{22}^1 \{\alpha_{22}^1\}^* \alpha_{23}^1 = ba^* b \vee aa^* ba^* b = a^* ba^* b,$$

$$\alpha_{31}^2 = \alpha_{31}^1 \vee \alpha_{32}^1 \{\alpha_{22}^1\}^* \alpha_{21}^1 = (a \vee ba^* b)a^*,$$

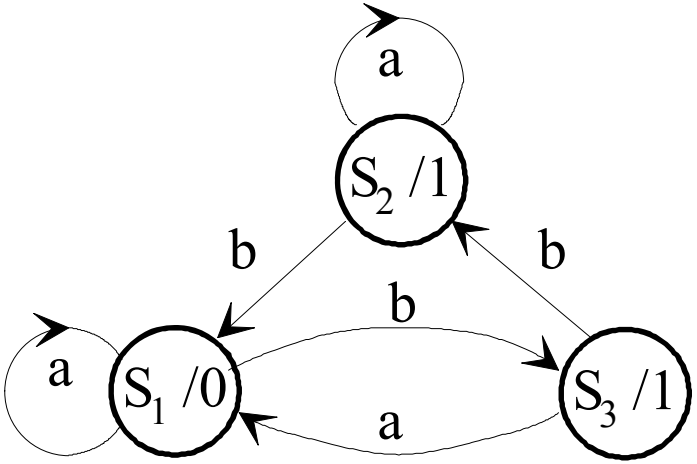
$$\alpha_{32}^2 = \alpha_{32}^1 \vee \alpha_{32}^1 \{\alpha_{22}^1\}^* \alpha_{22}^1 = b \vee ba^* a = ba^*,$$

$$\alpha_{33}^2 = \alpha_{33}^1 \vee \alpha_{32}^1 \{\alpha_{22}^1\}^* \alpha_{23}^1 = (a \vee ba^* b)a^* b.$$

4. Из правил 1-3 следует, что регулярное выражение для входных последовательностей, представленное автоматом с начальным состоянием S_r, имеет вид:

$$\alpha^n_{rm_1} \vee \alpha^n_{rm_2} \vee \ldots \vee \alpha^n_{rm_p}.$$

Пример: В нашем случае автомат Мура имеет три состояния и входной алфавит состоит из двух символов.



Начальное состояние – S_1.

Выходной сигнал равен 1 в состояниях S_2 и S_3.

Для события, представленного автоматом, необходимо найти регулярное выражение в форме: $\alpha^3_{13} \vee \alpha^3_{12}$.

$$\alpha_{12}^3 = \alpha_{12}^2 \vee \alpha_{13}^2 \{\alpha_{33}^2\}^* \alpha_{32}^2,$$

$$\alpha_{13}^3 = \alpha_{13}^2 \vee \alpha_{13}^2 \{\alpha_{33}^2\}^* \alpha_{33}^2,$$

$$\alpha_{12}^3 \vee \alpha_{13}^3 = \Lambda \vee \alpha_{13}^2 \{\alpha_{33}^2\}^* \alpha_{32}^2 \vee$$

$$\alpha_{13}^2 \vee \alpha_{13}^2 \{\alpha_{33}^2\}^* \alpha_{33}^2 =$$

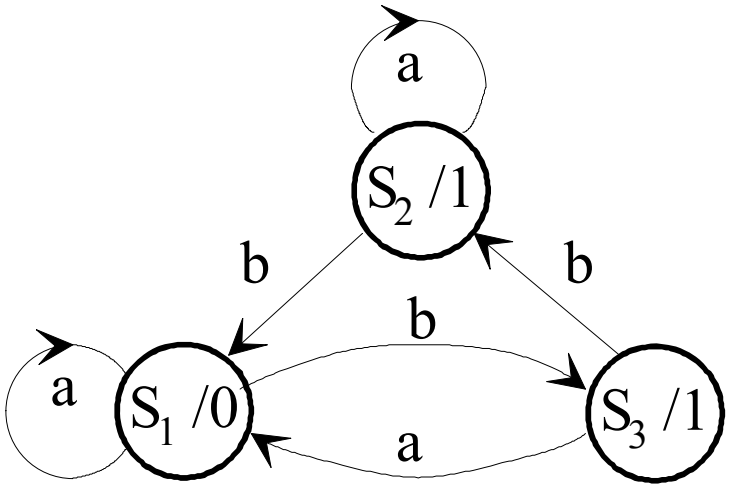
$$\alpha_{13}^2 \{\alpha_{33}^2\}^* \alpha_{32}^2 \vee \alpha_{13}^2 \{\alpha_{33}^2\}^* =$$

$$\alpha_{13}^2 \{\alpha_{33}^2\}^* (\alpha_{32}^2 \vee \phi) =$$

$$a^* b \{a a^* b \vee b a^* b a^* b\}^* (b a^* \vee \phi).$$

Начиная с инициального состояния S_1 графа переходов автомата, нетрудно проверить входную последовательность, порождаемую регулярным выражением:

$$\alpha_{12}^3 \vee \alpha_{13}^3 = a^* b\{aa^* b \vee ba^* ba^* b\}^* (\phi \vee ba^*).$$

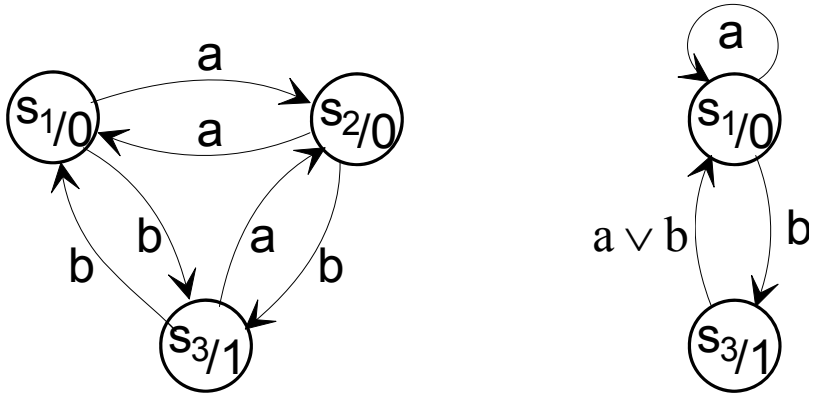


Сложность регулярного выражения (его форма) зависит от нумерации состояний. Пожалуйста, проверьте это сами.

Минимизация состояний полностью определенного автомата

Графы переходов могут содержать избыточные состояния, т.е. состояния, функция которых может быть выполнена другими состояниями.

Минимизация числа состояний – это преобразование данного автомата в другой, не имеющий избыточных состояний, такое, что оба автомата имеют одно и тоже поведение (для одинаковых входных последовательностей они производят одинаковые выходные последовательности).



Состояния s_1 и s_2 могут быть объединены в одно состояние.

Минимизация состояний полностью определенного автомата

Два состояния s_i и s_j автомата A ***различимы,*** если и только если существует конечная входная последовательность, которая вызывает различные выходные последовательности в зависимости от того, из какого состояния стартует автомат A из s_i или s_j.

Такая последовательность называется ***различающей последовательностью*** для (s_i , s_j). Если существует различающая последовательность длины k для (s_i , s_j), говорят, что эти состояния ***k-различимы.***

Говорят, что состояния s_i и s_j ***эквивалентны*** $(s_i \Leftrightarrow s_j)$, если и только если не существует различающей последовательности для (s_i , s_j).

Если s_i эквивалентно s_j и s_j эквивалентно s_k, то s_i эквивалентно s_k. Таким образом, эквивалентность состояний является ***отношением эквивалентности*** (т.е. оно рефлексивно, симметрично и транзитивно).

Отношение эквивалентности разбивает состояния автомата на множество ***эквивалентных классов***.

Свойство. Если s_j и s_j – эквивалентные состояния, то состояния, являющиеся их непосредственными последователями, также эквивалентны.

Минимизация состояний полностью определенного автомата

Для будущего полезно ввести понятие ***k-эквивалентных*** состояний.

Два состояния s_i и s_j автомата A ***k-эквивалентны*** $(s_i \overset{k}{\Leftrightarrow} s_j)$, если и только если не существует различающей последовательности длины k или меньше для этих состояний.

Все состояния автомата могут быть разбиты на k-эквивалентные классы:

1. Все состояния, принадлежащие одному классу, должны быть k-эквиваленты ;
2. Все состояния, принадлежащие различным классам , должны быть k-различимы.

Пусть k-эквивалентное разбиение состояний дает P_k k-эквивалентных класса и $|P_k|$ – число таких классов, тогда очевидно, что

$$n \geq |P_1| \geq 2, \ |P_{k+1}| \geq |P_k|.$$

Минимизация состояний полностью определенного автомата

Если $|P_k| = |P_{k+1}|$, то $P_k = P_{k+u} = P$, $u = 1,2,...,$ и P является разбиением на полностью эквивалентные классы.

По индукции можно доказать, что если $|P_k| \neq |P_{k-1}|$, то $|P_k| \geq k+1$.

Для автомата с n состояниями $P_n = P_{n-1}$.

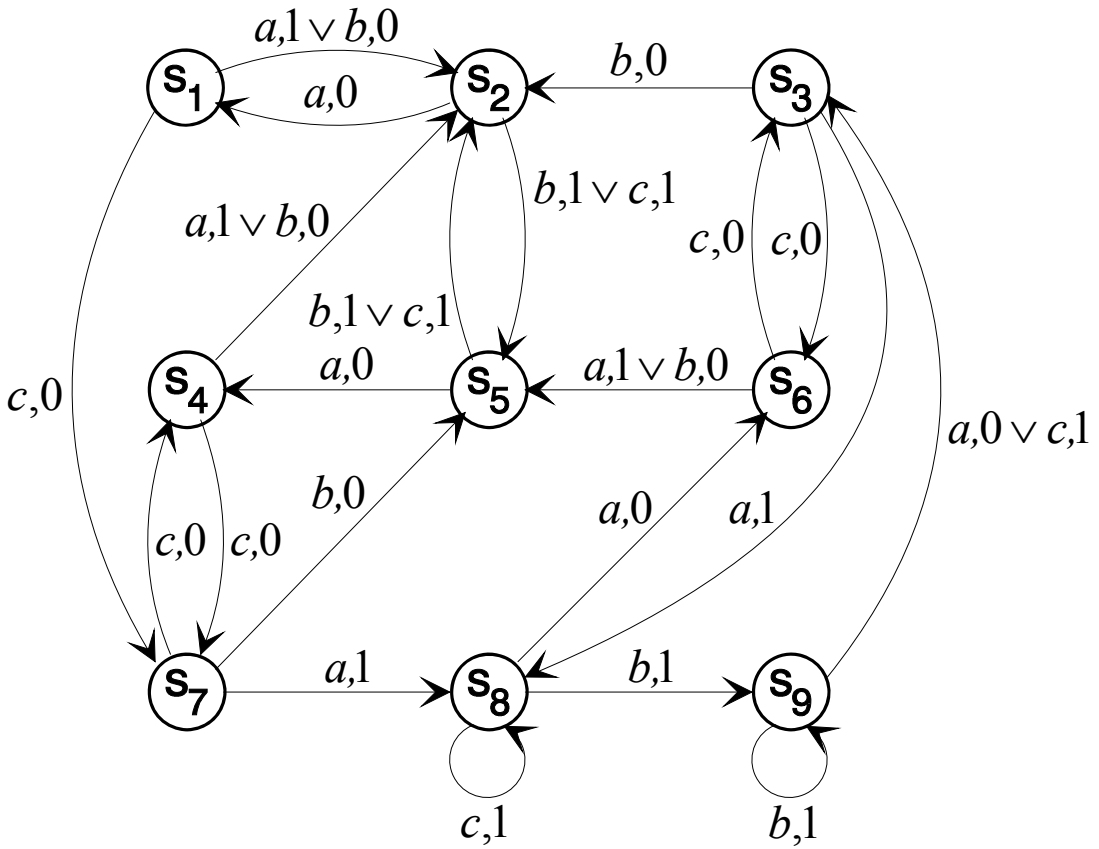
Предположим, что $P_n \neq P_{n-1}$, тогда $|P_n| \geq n+1$, что невозможно.

Теорема. Для автомата с n состояниями $P_{n-1} = P$.

Поэтому процедура нахождения классов эквивалентных состояний требует не более, чем $k \leq n-1$ шагов нахождения k-эквивалентных классов состояний.

Рассмотрим процедуру минимизации числа состояний автомата Мили на примере.

Минимизация состояний полностью определенного автомата



Минимизация состояний полностью определенного автомата

Разбиение с помощью таблиц P_k:

Таблица переходов состояний автомата

s_i \ i_j	a	b	c
1	2/1	2/0	7/0
2	1/0	5/1	5/1
3	8/1	2/0	6/0
4	2/1	2/0	7/0
5	4/0	2/1	2/1
6	5/1	5/0	3/0
7	8/1	5/0	4/0
8	6/0	9/1	8/1
9	3/0	9/1	3/1

Строки с различными выходами объединяются в различные группы

P_1	s_i \ i_j	a	b	c
α	1	2/1	2/0	7/0
	3	8/1	2/0	6/0
	4	2/1	2/0	7/0
	6	5/1	5/0	3/0
	7	8/1	5/0	4/0
β	2	1/0	5/1	5/1
	5	4/0	2/1	2/1
	8	6/0	9/1	8/1
	9	3/0	9/1	3/1

Минимизация состояний полностью определенного автомата

Таблица P_1

P_1	s_i \ i_j	a	b	c
α	1	2/β	2/β	7/α
	3	8/β	2/β	6/α
	4	2/β	2/β	7/α
	6	5/β	5/β	3/α
	7	8/β	5/β	4/α
β	2	1/α	5/β	5/β
	5	4/α	2/β	2/β
	8	6/α	9/β	8/β
	9	3/α	9/β	3/α

Таблица P_2

P_2	s_i \ i_j	a	b	c
α	1	2/β	2/β	7/α
	3	8/β	2/β	6/α
	4	2/β	2/β	7/α
	6	5/β	5/β	3/α
	7	8/β	5/β	4/α
β	2	1/α	5/β	5/β
	5	4/α	2/β	2/β
	8	6/α	9/γ	8/β
γ	9	3/α	9/γ	3/α

Минимизация состояний полностью определенного автомата

Таблица P_3

P_3	s_i \ i_j	a	b	c
α	1	2/β	2/β	7/α
	3	8/δ	2/β	6/α
	4	2/β	2/β	7/α
	6	5/β	5/β	3/α
	7	8/δ	5/β	4/α
β	2	1/α	5/β	5/β
	5	4/α	2/β	2/β
γ	9	3/α	9/γ	3/α
δ	8	6/α	9/γ	8/δ

Таблица P_4

P_4	s_i \ i_j	a	b	c
α	1	2/β	2/β	7/ε
	4	2/β	2/β	7/ε
	6	5/β	5/β	3/ε
β	2	1/α	5/β	5/β
	5	4/α	2/β	2/β
γ	9	3/ε	9/γ	3/ε
δ	8	6/α	9/γ	8/δ
ε	3	8/δ	2/β	6/α
	7	8/δ	5/β	4/α

Минимизация состояний полностью определенного автомата

Разбиение состояний автомата на классы эквивалентности:

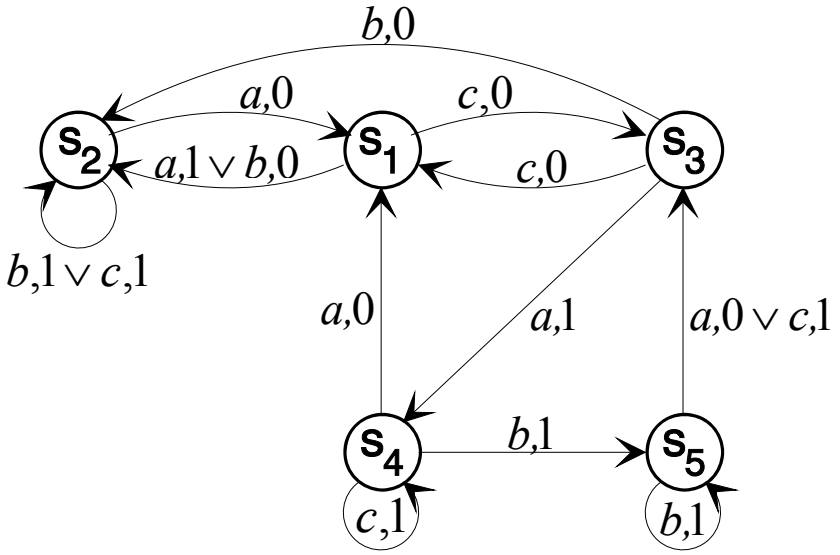
$$P = \{(1,4,6),(2,5),(3,7),8,9\}.$$

Обозначим эквивалентные классы состояний новыми состояниями:

$$s_1 = \{1,4,6\},\quad s_2 = \{2,5\},$$
$$s_3 = \{3,7\},\ s_4 = \{8\},\ s_5 = \{9\}.$$

Строим таблицу переходов и граф переходов минимального автомата:

s_i \ i_j	a	b	c
1	2/1	2/0	3/0
2	1/0	2/1	2/1
3	4/1	2/0	1/0
4	1/0	5/1	4/1
5	3/0	5/1	3/1



Минимизация состояний полностью определенного автомата

Разбиение с помощью таблицы пар состояний:

Таблица переходов состояний автомата

s_i \ i_j	a	b	c
1	2/1	2/0	7/0
2	1/0	5/1	5/1
3	8/1	2/0	6/0
4	2/1	2/0	7/0
5	4/0	2/1	2/1
6	5/1	5/0	3/0
7	8/1	5/0	4/0
8	6/0	9/1	8/1
9	3/0	9/1	3/1

Разбиение состояний автомата на эквивалентные классы такое же, как и в предыдущем случае:

$$P = \{(1,4,6),(2,5),(3,7),8,9\}.$$

Пары	a	b	c	
~~1,3~~	~~2,8~~	~~2~~	~~7,6~~	3
1,4	2	2	7	
1,6	2,5	2,5	7,3	
~~1,7~~	~~2,8~~	~~2,5~~	~~7,4~~	3
2,5	1,4	5,2	5,2	
~~2,8~~	~~1,6~~	~~5,9~~	~~5,8~~	2
~~2,9~~	~~1,3~~	~~5,9~~	~~5,3~~	1
~~3,4~~	~~8,2~~	~~2,2~~	~~6,7~~	3
~~3,6~~	~~8,5~~	~~2,5~~	~~6,3~~	2
3,7	8	2,5	6,4	
4,6	2,5	2,5	7,3	
~~4,7~~	~~2,8~~	~~2,5~~	~~7,4~~	3
~~6,7~~	~~5,8~~	~~5~~	~~3,4~~	2
~~5,8~~	~~4,6~~	~~2,9~~	~~2,8~~	1
~~5,9~~	~~4,3~~	~~2,9~~	~~2,3~~	1
~~8,9~~	~~6,3~~	~~9~~	~~8,3~~	1

Минимизация состояний полностью определенного автомата

Первый столбец таблицы пар содержит все неупорядоченные пары 1-эквивалентных состояний $\{s_i, s_j\}, i \neq j$, которые можно найти из таблицы переходов.

Клетки таблицы, лежащие на пересечениях строки $\{s_i, s_j\}$ и столбцов, обозначенных входными символами, содержат состояния, которые являются непосредственными последователями состояний $\{s_i, s_j\}$ по каждому входному символу.

Несколько раз просматриваем таблицу пар состояний и каждый раз вычеркиваем строки неэквивалентных состояний в соответствии со следующим правилом: строка вычеркивается, если, по крайней мере, одна из клеток содержит пару состояний, которая отсутствует в первой колонке таблицы. Процесс завершается, когда ни одна строка не может быть удалена.

Пары состояний, обозначающие оставшиеся строки в таблице, образуют классы эквивалентности. Для нашего примера получим три класса: {1,4,6}, {2,5} and {3,7}. Они не содержат состояний 8 and 9. Поэтому добавим два класса: {8} and {9}.

Минимизация состояний полностью определенного автомата

Заключение о процедуре минимизации числа состояний:

- Все состояния из эквивалентного класса попарно эквивалентны. Класс эквивалентных состояний может быть заменен одним состоянием в сокращенном автомате.
- Эквивалентные классы образуют разбиение множества состояний.
- Начинаем с разбиения, когда все состояния образуют один блок. Итеративно улучшаем разбиение, выделяя 1-различимые состояния, 2-различимые состояния и т.д.
- При получении P_{k+1} из P_k размещаем в одном и том же блоке P_{k+1} состояния, которые (k+1)-эквивалентны, и в различные блоки состояния, которые (k+1)-различимы.

Минимизация состояний полностью определенного автомата

Теорема. Разбиение на эквивалентные классы состояний единственно для любого автомата.

Определение. Говорят, что два автомата A_1 и A_2 эквивалентны, если и только если для каждого состояния в A_1 существует соответствующее ему эквивалентное состояние в A_2 и наоборот.

Теорема. Для каждого автомата A существует минимальный автомат A_{min} , эквивалентный A, который является единственным с точностью до изоморфизма (с точностью до переобозначения состояний).

Сложность проблемы минимизации числа состояний автомата:

- Непосредственная реализация процедур минимизации имеет сложность Cn^2, где n – число состояний исходного автомата.
- Лучшая известная реализация имеет сложность $Cn\log(n)$ [Hopcroft 1971].



Логическое Проектирование

Лекция 11

Неполностью определенная спецификация и недетерминизм

Часто при работе реальных устройств некоторые наборы, состоящие из входного символа и внутреннего состояния (i_j, s_k), никогда не встречаются и в этом случае можно не заботиться о поведении устройства на этих наборах. Иногда на некотором наборе (паре) может быть несущественными только выходной сигнал или только внутреннее состояние.

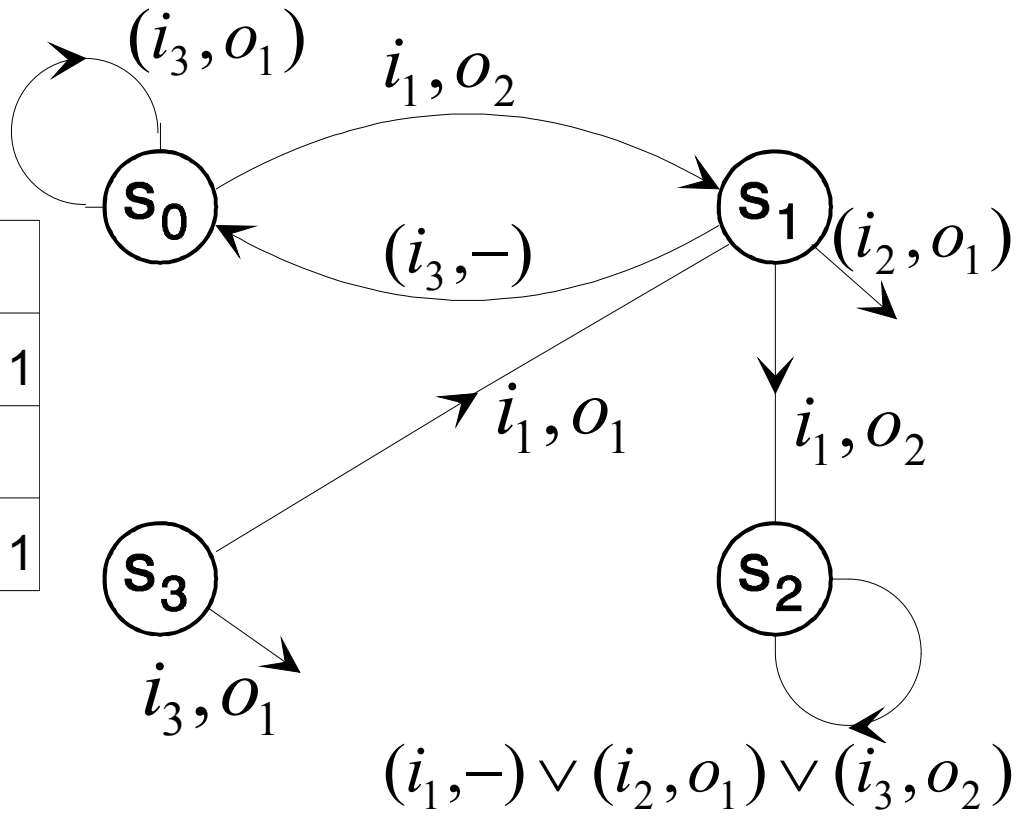
- ***Частичный конечный автомат*** является моделью такого типа дискретных устройств с конечной памятью. Для такого автомата значения функции переходов $\Phi(i_j, s_k)$ и функции выходов $F(i_j, s_k)$ определены не на всех возможных парах (i_j, s_k).

Доопределением частичного автомата называется всюду определенный автомат, для которого функции переходов и выходов совпадают с соответствующими функциями частичного автомата везде, где они определены.

Частичный автомат может быть задан таблицей или графом (диаграммой) переходов состояний. Некоторые клетки таблицы содержат вместо пар (s_k, o_j) или элементов пар ***произвольные задания*** (прочерки, или *don't cares*). В графе переходов состояний выходные дуги некоторых состояний могут не вести ни в какое состояние, вообще отсутствовать или не содержать значений выходных символов.

Таблица и граф переходов состояний частичного автомата Мили

i \ S	s_0	s_1	s_2	s_3
i_1	s_1, o_2	s_2, o_2	$s_2, -$	s_1, o_1
i_2	–	$-, o_1$	s_2, o_1	–
i_3	s_0, o_1	$s_0, -$	s_2, o_2	$-, o_1$



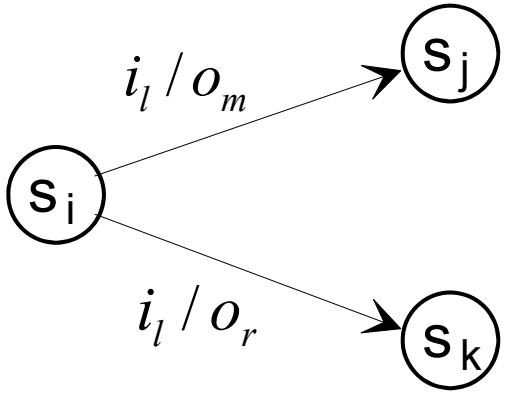
Этот автомат не является детерминированным, так как в нем не определены некоторые переходы.
Такой вид ***недетерминизма*** называется "***неполной структурой переходов***".
Последовательности входных символов, ведущие в неопределенные состояния запрещены.

- Последовательность входных символов называется ***применимой к состоянию*** S_k, если эта последовательность не ведет в неопределенное состояние. Например, входная последовательность $i_1 i_1 i_2 i_3$ является применимой к состоянию S_0, а последовательности $i_1 i_2$ и $i_1 i_3 i_2$ являются неприменимыми.

Если не определен только выходной символ, это означает, что с целью доопределения можно выбрать произвольный символ.

- Неполная структура переходов автомата и неопрененные выходные символы могут быть использованы в процедуре синтеза автоматных реализаций с целью их оптимизации.

Другой вид ***недетерминизма*** в теории автоматов связан с возможностью многих переходов: из данного текущего состояния под воздействием одного и того же входного условия автомат может переходить в различные состояния и производить различные выходы.



В таблице переходов состояний такого автомата могут быть клетки, содержащие подмножества состояний. Обозначая эти подмножества новыми состояниями, можно свести недетерминированный автомат к детерминированному с помощью известного алгоритма.

Вход	a	b	c
Состояние	Следующее состояние и выход		
S_1	$S_1/0$	$S_2/0$	$S_4/1$
S_2	$S_2/0$	$S_1/1$	$S_1/0$
S_3	$S_2/0$	$\{S_1,S_4\}/-$	$S_2/1$
S_4	$\{S_3,S_4\}/-$	$S_2/1$	$S_3/1$

Очевидно, что этот алгоритм связан с увеличением числа состояний. Если недетерминированный автомат содержит n состояний, множество всех возможных подмножеств состояний содержит 2^n подмножеств. Это верхняя граница увеличения числа состояний при переходе к детерминированному автомату.

Сведение недетерминированного автомата к детерминированному необходимо для того, чтобы сделать отношение смены состояний функциональным. Это требование аппаратной реализации.

- Таким образом, этот вид недетерминизма обеспечивает только ***компактный*** способ описания множества допустимых переходов и не дает возможностей для оптимизации функции переходов. (Неопределенные выходные символы и в этом случае оставляют некоторую свободу в упрощении функции выходов.)

Минимизация состояний частичного автомата

- Две ***последовательности*** $\tilde{v}$ и $\tilde{w}$, составленные из символов некоторого алфавита и символа неопределенности называются ***совместимыми,*** если существует общая для них покрывающая последовательность. Иными словами, если в некоторой позиции обеих последовательностей стоят значимые символы, то эти символы должны совпадать. Пример совместимых последовательностей:

$$\tilde{v} = \gamma - \alpha\ \beta - - \alpha\ \beta\ \gamma$$

$$\tilde{w} = -\beta\ \alpha - - \beta\ \alpha - -$$

$$\text{Покрывающая последовательность} = - - \alpha - - - \alpha - -$$

- ***Состояния*** S_i и S_j частичного автомата M называются ***совместимыми***, если при любой входной последовательности автомат, стартуя из состояний S_i и S_j, производит совместимые выходные последовательности.

Минимизация состояний частичного автомата

- Подмножество состояний автомата образует ***группу совместимости,*** если все его состояния попарно совместимы.

- ***Группа совместимости*** называется ***максимальной,*** если после добавления в нее любого состояния она теряет это свойство.

- Множество групп совместимости образует ***группировку***, если любое состояние автомата входит, по крайней мере, в одну из этих групп.

- ***Группировка***, составленная из всех максимальных групп совместимости называется ***максимальной***.

Свойство совместимости отличается от свойства эквивалентности, так как оно нетранзитивно. Нахождение группировки отличается от разбиения состояний на эквивалентные классы хотя бы тем, что различные группы совместимости могут пересекаться.

Минимизация состояний частичного автомата

Нахождение максимальной группировки является первым шагом минимизации состояний частичного автомата.

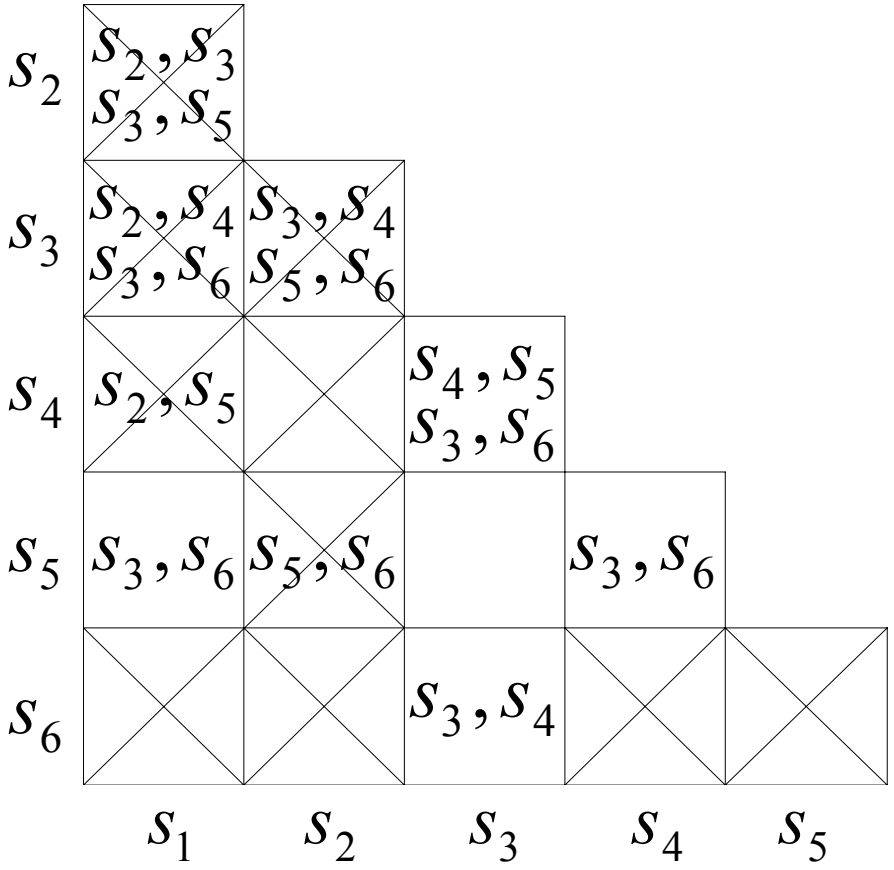
Все определения, введенные для эквивалентных состояний, практически те же для совместимых состояний. Если два состояний (n-1)-совместимы, то они полностью совместимы.

Рассмотрим пример минимизации числа состояний частичного автомата, таблица переходов которого показана на следующем слайде.

Необходимо найти все пары совместимых состояний. Для этого строится треугольная таблица, клетки которой соответствуют парам совместимых состояний.

Минимизация состояний частичного автомата

$s \backslash i$	a	b	c	d
s_1	$s_2,-$	$s_3,0$	$-$	$s_4,-$
s_2	$s_3,0$	$s_5,0$	$-$	$-$
s_3	$s_4,-$	$s_6,-$	$s_3,-$	$-$
s_4	$s_5,1$	$s_3,0$	$-$	$s_1,-$
s_5	$-$	$s_6,0$	$-$	$-$
s_6	$-$	$-,1$	$s_4,-$	$s_2,-$



Минимизация состояний частичного автомата

Первый шаг – вычеркивание клеток, представляющих пары состояний, для которых существует входной символ, приводящий к различным значениям выхода. В остальные клетки вписываются возможные пары следующих состояний для их проверки на совместимость в следующих шагах.

Далее вычеркиваются все клетки, содержащие удаленные пары, и этот процесс повторяется до тех пор, пока это возможно. Невычеркнутые клетки соответствуют всем парам совместимых состояний.

Пары совместимых состояний:

$$(s_1, s_5), (s_3, s_4), (s_3, s_5), (s_3, s_6) \text{ и } (s_4, s_5).$$

Максимальная группировка может быть построена с использованием полученной треугольной таблицы. Просматривая колонки таблицы слева направо, последовательно строят системы множеств состояний с помощью следующего алгоритма. Число шагов алгоритма равно числу столбцов таблицы.

Минимизация состояний частичного автомата

- Множество всех состояний автомата выбирается в качестве исходной системы множеств состояний (шаг 0). Предположим, что уже выполнено *i-1* шагов и построена система множеств

$$S_1, S_2, ..., S_p.$$

- На *i*-м шаге находим все состояния, несовместимые с состоянием s_i (вычеркнутые клетки *i*-го столбца таблицы соответствуют этим состояниям). Если множество S_j не содержит s_i или состояний , несовместимых с ним, это множество не изменяется. В противном случае из S_j формируем два множества: одно из них – посредством удаления состояния s_i, а другое – посредством удаления всех состояний, несовместимых с s_i. Выполняем это для всех S_j и удаляем немаксимальные множества (те, которые содержатся в других) . Преобразованная система множеств является результатом *i*-го шага.

Минимизация состояний частичного автомата

- Система подмножеств, образованная на последнем шаге, является максимальной группировкой .

Результат применения алгоритма иллюстрируется следующей таблицей.

Шаг No	Система множеств
0	$\{s_1, s_2, s_3, s_4, s_5, s_6\}$
1	$\{s_2, s_3, s_4, s_5, s_6\}, \{s_1, s_5\}$
2	$\{s_3, s_4, s_5, s_6\}, \{s_2\}, \{s_1, s_5\}$
3	$\{s_3, s_4, s_5, s_6\}, \{s_2\}, \{s_1, s_5\}$
4	$\{s_3, s_5, s_6\}, \{s_3, s_4, s_5\}, \{s_2\}, \{s_1, s_5\}$
5	$\{s_3, s_6\}, \{s_3, s_4, s_5\}, \{s_2\}, \{s_1, s_5\}$

Минимизация состояний частичного автомата

Построение минимального частичного автомата

Пусть имеется некоторая группировка $S_1, S_2, ..., S_r$ состояний частичного автомата.

- Группировка называется ***замкнутой*** , если для любого множества S_v и любого входного символа α множество всех состояний, в которые переходят состояния из S_v под воздействием α, полностью содержатся в некотором множестве S_w, зависящим от S_v и α. (Это определение имеет в виду только те состояния из S_v, для которых определены следующие состояния.)

Имея замкнутую группировку (анклав) можно построить покрывающий автомат M' подобно тому, как это делается для полностью определенного автомата на основе эквивалентных классов состояний: каждая группа S_i заменяется состоянием s_i' автомата M'.

Минимизация состояний частичного автомата

Таким образом, построение покрывающего автомата сводится к нахождению замкнутой группировки. Построение минимального покрывающего автомата сводится к нахождению замкнутой группировки из минимального числа групп совместимости.

- Проблема нахождения минимального покрывающего автомата может быть решена только на основе полного перебора. Действительно, любая группа совместимости – это подмножество максимальной группы совместимости. Перебирая все такие подмножества и составляя различные группировки из них, можно найти минимальную замкнутую группировку. Конечно, этот подход неэффективен, но другого не существует.
- Важной проблемой является нахождение приблизительного решения, т.е. нахождение покрывающего автомата “близкого к минимальному”. Для этого необходимо иметь систему оценок:

Минимизация состояний частичного автомата

– ***нижняя оценка*** – число состояний минимального автомата не меньше, чем мощность максимального множества попарно несравнимых состояний исходного автомата;
– ***верхняя оценка*** – число множеств в максимальной группировке.

- Можно показать, что максимальная группировка является замкнутой и на ее основе может быть построен покрывающий автомат .

Вернемся к примеру. Покрывающий автомат M' содержит четыре состояния, обозначенных как:

$$q_1 - \{s_3, s_6\},\ q_2 - \{s_3, s_4, s_5\},\ q_3 - \{s_2\},\ q_4 - \{s_1, s_5\}.$$

Минимизация состояний частичного автомата

Нетрудно построить таблицу переходов автомата M':

$q \backslash i$	a	b	c	d
q_1	$q_2,-$	$q_1,1$	$q_2,-$	$q_3,-$
q_2	$q_2,1$	$q_1,0$	$q_2,-$	$q_4,-$
q_3	$q_2,0$	$q_4,0$	$-$	$-$
q_4	$q_3,-$	$q_1,0$	$-$	$q_2,-$

В данном случае автомат M' является минимальным, так как верхняя оценка числа его состояний, равная четырем, совпадает с нижней оценкой: четыре состояния попарно несовместимы $(s_1,\ s_2,\ s_4$ и $s_6)$.

Время и синхронизация

В абстрактной теории конечных автоматов автомат является *математическим объектом*, представляющим собой *алгоритм с конечной памятью для преобразования слов.*

Эта модель содержит дискретную переменную ***t*** , которая трактуется как последовательный номер шага алгоритма или последовательный номер символа в последовательностях.

С другой стороны, конечный автомат можно понимать как машину, или физическое устройство, которое функционирует в реальном физическом времени.

Что такое реальное физическое время? Какова его природа?

В чем заключается различие между реальным физическим временем и искусственным автоматным (логическим) временем?

Как ввести реальное время в модель автомата?

Два определения времени:

Физическое время — непрерывная независимая физическая переменная;

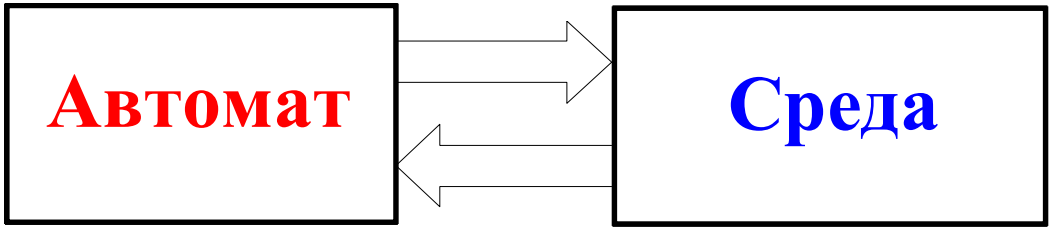
Логическое время — дискретная логическая переменная, отражающая причинно-следственные отношения между событиями, или частичный порядок на множестве событий.

“Если ничего не происходит, то нет и времени.”

Аристотель

Время и синхронизация во времени

определяются через взаимодействие автомата со внешней средой.



Находясь внутри автомата, можно наблюдать внутренние состояния и состояния его входов и выходов только в дискретные моменты времени ***t*** (такты), когда достигнуты их устойчивы значения.

Какова природа механизма квантования времени?
Чтобы ответить на этот вопрос необходимо также рассмотреть систему «Автомат – Среда».

Входы, выходы и внутренние состояния должны быть устойчивы в реальный момент времени изменения значения переменной ***t*** как для автомата, так и для внешней среды. Поэтому механизм квантования времени для автомата и внешней среды должен быть общим или, по крайней мере, скоординированным.

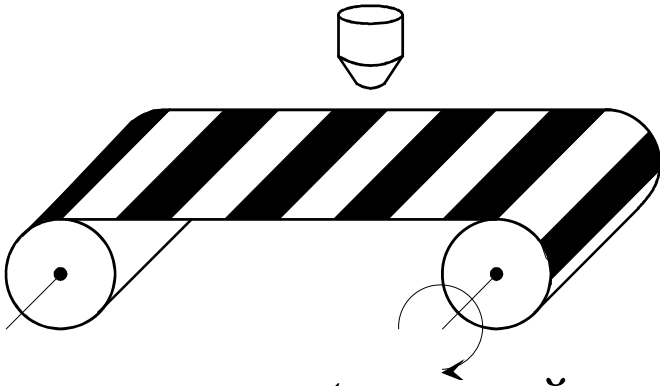
Эта координация функционирования автомата и внешней среды во времени называется ***синхронизацией***.

В частном случае поведение автомата может не зависеть от внешней среды (автомат без входов). Такой автомат называется ***автономным***.

В некоторых случаях система «Автомат – Среда» может сама трактоваться как автономный автомат.

Какие существуют способы введения механизма квантования времени? Будем рассматривать их на примерах.

Пример 1: Пусть необходимо построить автомат, который считает по модулю ***n*** число черных полосок на ленте с нанесенными черными и белыми полосками. Полоски различных цветов перемежаются и имеют одинаковую ширину.

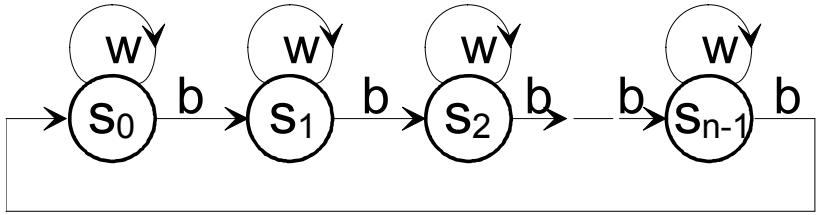


Автоматное уравнение:

$$s(t) = (s(t-1) + x(t))_{\mathrm{mod}\, n},$$

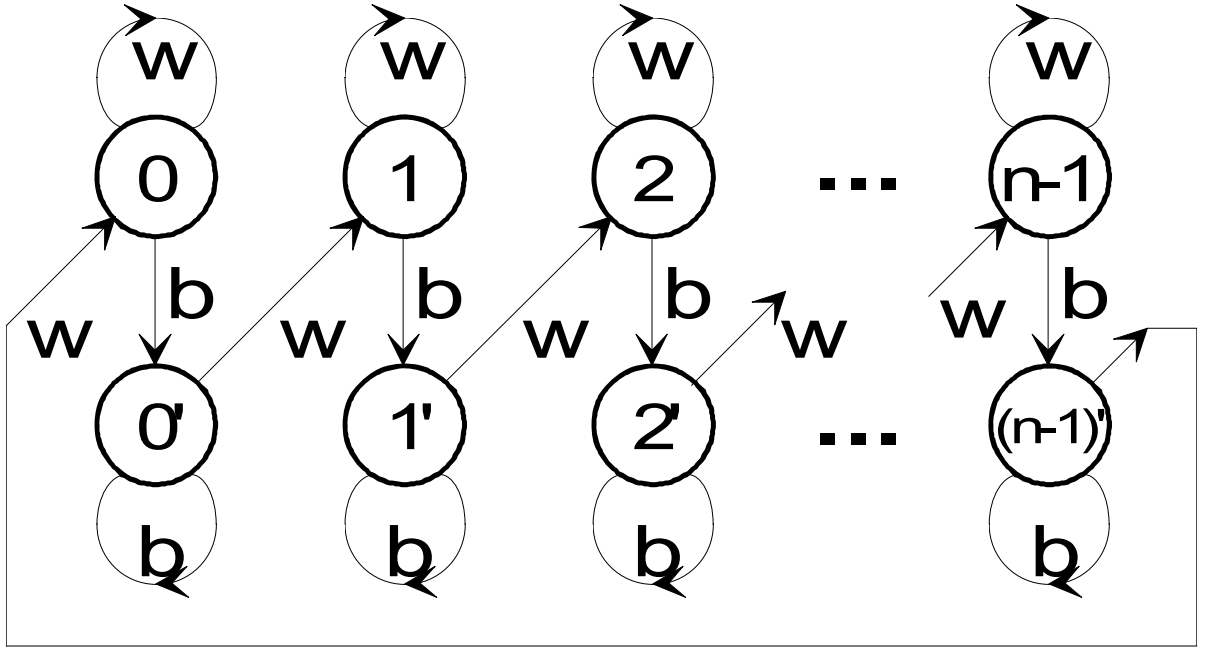
$$x(t) = \begin{cases} 1, \text{если полоска } t \text{ черная} \\ 0, \text{если полоска } t \text{ белая} \end{cases}$$

В то же время, ***t*** – новый момент времени, который устанавливается каждый раз, когда новая полоска появляется под читающей головкой. Граф переходов, соответствующий автоматному уравнению, имеет вид:



Этот граф является моделью абстрактного автомата. Но если рассматривать автомат как устройство, можно видеть, что в случае, когда черная полоска находится под головкой достаточно долгое время, состояния начинают циклически изменяться.

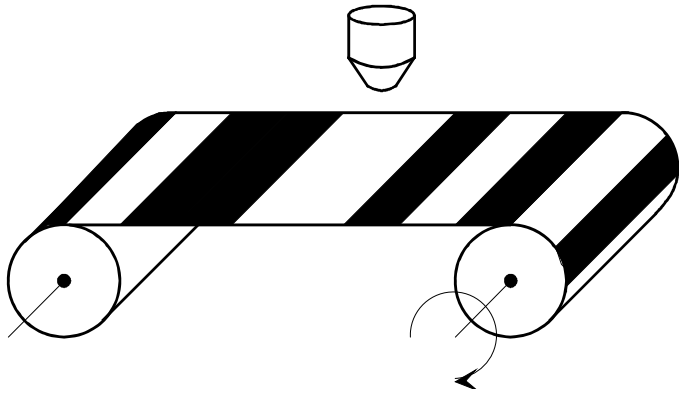
В этом случае функция переходов должна быть изменена. Можно увеличить число состояний в два раза и построить следующий граф переходов:



Теперь только осталось выбрать правильную скорость ленты: время прохождения каждой полоски ленты под читающей головкой должно превышать время достижения автоматом нового устойчивого состояния.

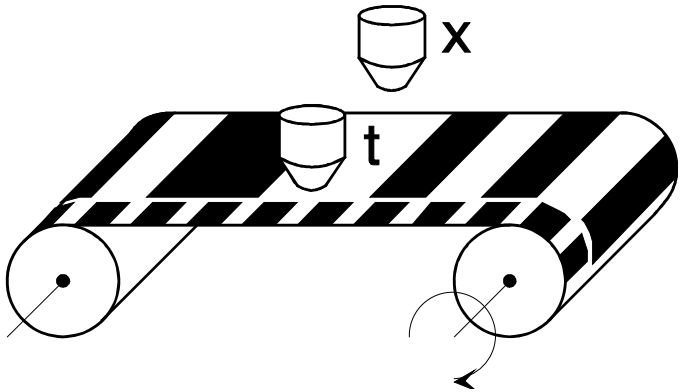
- В случае, когда среда порождает время для автомата и длительность такта определяется временем сохранения стабильными его входных сигналов, автомат называется ***асинхронным***.
- Каждое устойчивое состояние графа переходов, описывающего асинхронный автомат, должно иметь циклическую дугу, обозначенную объединением тех входных символов, которые обозначают дуги, ведущие в это состояние из других состояний.
- Последовательности входных символов для асинхронных автоматов не должны содержать одинаковых соседних символов.

Пример 2: Расширим пример 1: автомат должен различать на ленте соседние полоски одного и того же цвета (черные или белые). Одна широкая полоса содержит целое число полосок нормальной ширины.



Существует, по крайней мере, два способа решения этой задачи.

Первый способ. Для того, чтобы различать соседние полоски одинакового цвета, можно нанести маркеры на ленту и установить дополнительную читающую головку. Если лента движется с постоянной скоростью, головка, читающая маркеры, производит периодическую последовательность сигналов t (мы получили встроенные часы).

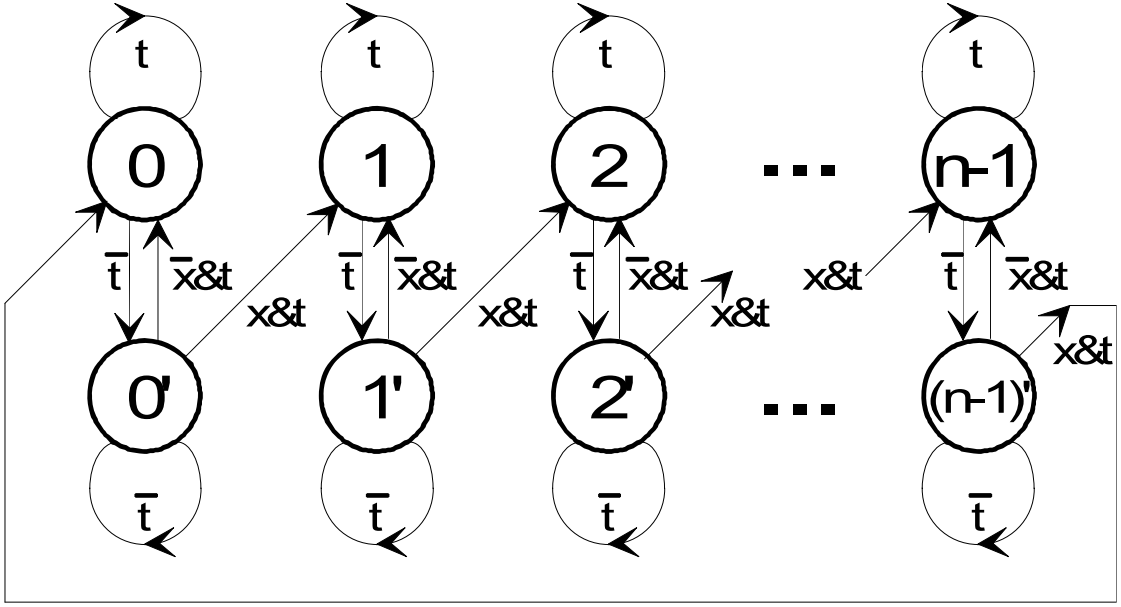


Сигналы x вместе с сигналами t можно рассматривать как переменные, кодирующие входные символы для автомата:

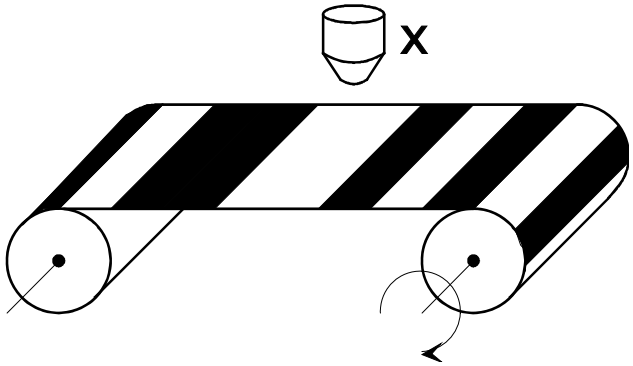
$i_1 = \{x = 0, t = 0\},\ i_2 = \{x = 0, t = 1\},\ i_3 = \{x = 1, t = 0\},\ i_4 = \{x = 1, t = 1\}.$

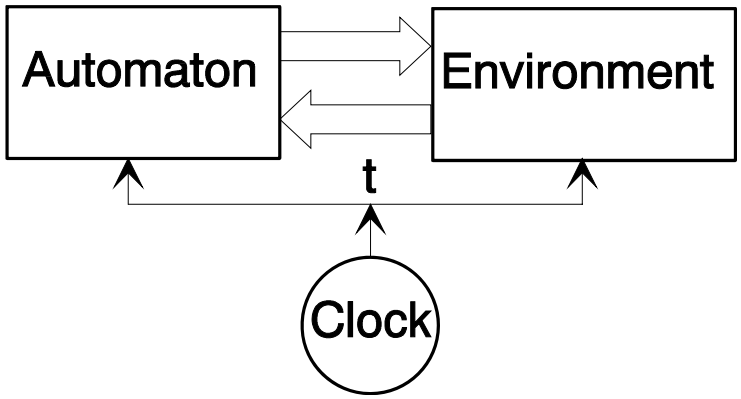
Этот способ сохраняет рамки асинхронной модели автомата: смена входного символа инициирует переход в автомате; входная последовательность не содержит одинаковых смежных символов.

Заметим, что в этой модели последовательность ***x*** может быть произвольной. Граф переходов автомата, считающего черные полоски нормальной ширины, имеет вид:



Второй способ. Может быть использована модель ***синхронного*** автомата, которая является частным случаем асинхронной модели. В этой модели сигнал ***t*** не включен во входные символы и рассматривается как дополнительный входной сигнал, используемый для синхронизации. Таким образом, данный сигнальный граф специфицирует также и поведение синхронного автомата.

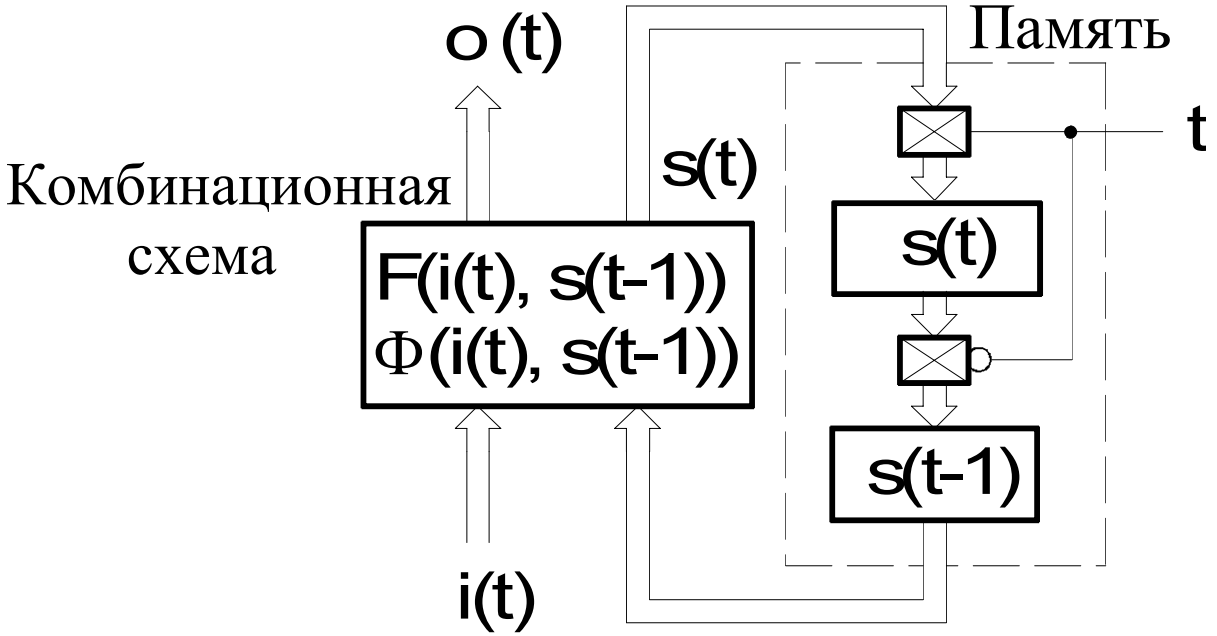




Внешние сигналы ***t*** от тактового генератора (часов) синхронизируют функционирование автомата и внешней среды. Каждый тактовый сигнал сдвигает ленту на одну полоску нормальной ширины и автомат выполняет шаг алгоритма. Длительность цикла тактовых сигналов должна превышать худший случай длительностей переходных процессов в автомате и внешней среде.

Сигнал ***t*** изменяется дважды за один цикл часов, разбивая функционирование устройства на две фазы: ***рабочую фазу*** (изменение состояния) и ***подготовительную фазу*** (запоминание нового состояния).

Когда говорят о синхронной модели автомата, обычно подразумевают следующую структуру для реализации автомата (устройства).



Условия корректной работы автомата:

Если τ_a – длительность переходного процесса в автомате,

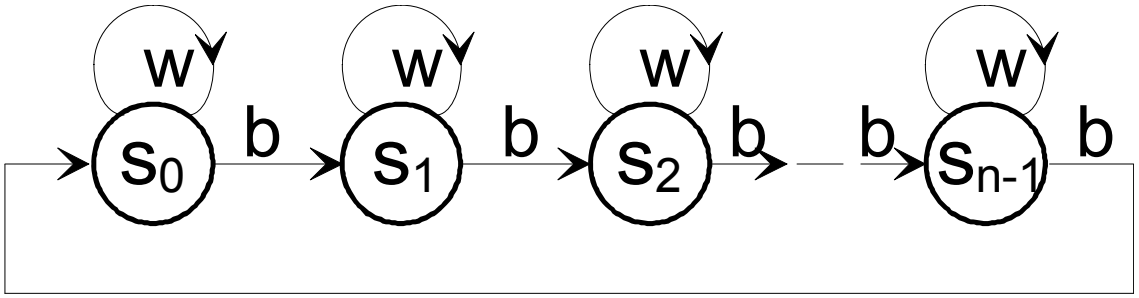
τ_e – длительность переходного процесса в окружающей среде,

T – длительность такта синхронизации,

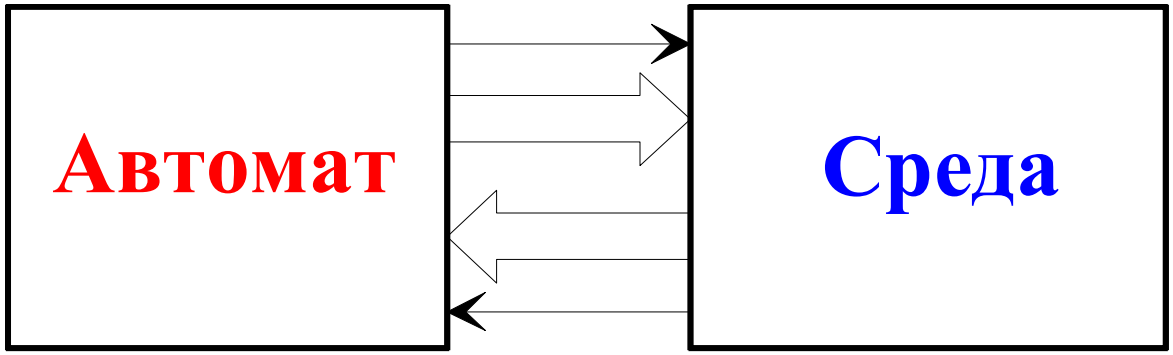
то необходимо, чтобы выполнялось условие: $T > \tau_a$, $T > \tau_e$

Используется способ разрыва обратной связи для предотвращения гонок. Когда *t=1,* новое состояние записывается в основной регистр; когда *t=0,* это состояние переписывается во вспомогательный регистр.

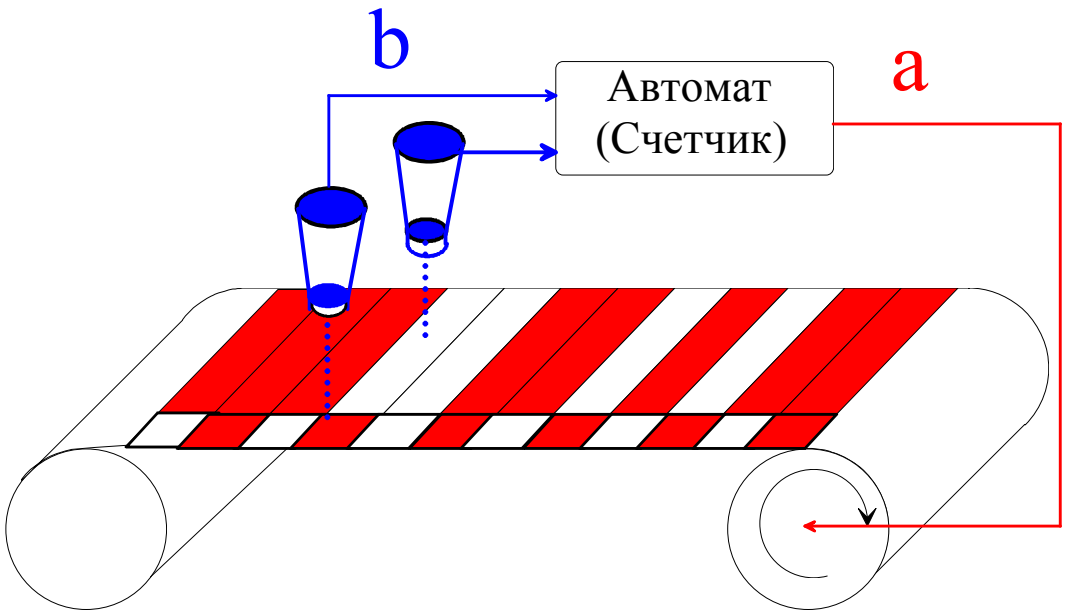
Держа в уме эту структуру, для синхронного автомата обычно, с целью упрощения, используют спецификацию абстрактного автомата. Например, в данном случае автомат может быть специфицирован графом переходов:



Структура системы «автомат-Среда», в которой внешняя среда порождает время для автомата и автомат порождает время для окружающей среды, называется ***согласованной моделью***.
Автомат в этой модели называется ***самосинхронным***.



Пример самосинхронной реализации автомата



a – сигнал продвижения ленты
(переходный процесс в автомате завершен);

b – сигнал, инициирующий смену состояния автомата
(продвижение ленты на одну полоску завершено)

Когда окружающая среда достигает устойчивого состояния, она вырабатывает сигнал окончания переходных процессов, который используется в качестве сигнала тактирования для автомата.

Когда автомат достигает устойчивого состояния, он вырабатывает сигнал завершения в нем переходных процессов, который используется как сигнал тактирования для среды.

Теория самосинхронных автоматов изучает способы построения самосинхронных устройств и способы выработки сигналов завершения переходных процессов.

Процедура проектирования простых дискретных устройств

Общепринятая процедура синтеза достаточно простых дискретных устройств состоит из следующих шагов.

1. ***Этап формального синтеза***:

- построение для устройства модели конечного автомата (построение графа или таблицы переходов состояний);
- минимизация числа состояний.

2. ***Этап структурного синтеза***:

- выбор типа реализации (синхронная, асинхронная, самосинхронная) и выполнение размещения состояний для минимального автомата;
- нахождение переключательных функций для переменных, кодирующих состояния автомата;

Процедура проектирования простых дискретных устройств

- нахождение функций возбуждения для триггеров каждого типа (D, SR, T and JK), соответствующих внутренним переменным автомата, и выбор триггеров с наиболее простыми функциями возбуждения;
- нахождение переключательных функций для выходных сигналов автомата.

3. ***Этап логической реализации***:

- выбор логического базиса (системы логических элементов) для реализации комбинационной схемы и элементов памяти (триггеров);
- построение логической схемы устройства;
- разработка тестов для проверки логической схемы устройства и ее проверка.

4. ***Физическое проектирование устройства для выбранной технологии и его изготовление.***



Логическое Проектирование

Лекция 12

Процедура проектирования простых дискретных устройств

Наличие стандартной процедуры проектирования простых синхронных автоматов не означает, что ее следует очень строго придерживаться. Иногда различные вариации могут дать хорошие результаты.

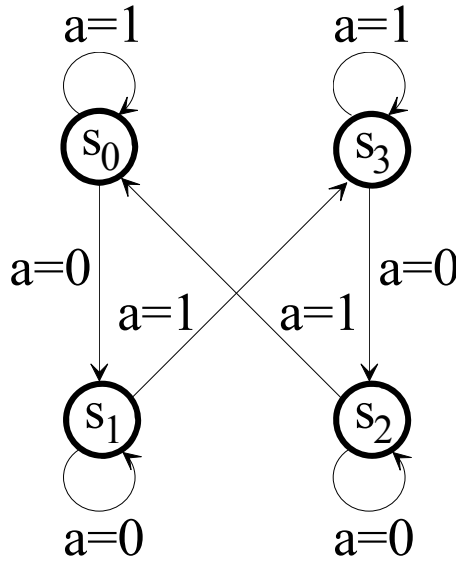
В качестве примера рассмотрим процедуру синтеза простейшего автомата – Т-триггера (счетного триггера). Эта процедура отличается от только что рассмотренной. В ней не ставятся в соответствие внутренним переменным элементы памяти, а непосредственно реализуются их переключательные функции.

Результатом этой процедуры является простейшая схема счетного триггера.

Процедура проектирования простых дискретных устройств

Счетный триггер – это счетчик по модулю 2.

Диаграмма переходов



Матрица переходов

	s_0	s_1	s_2	s_3
s_0	a	$\bar{a}$	–	–
s_1	–	$\bar{a}$	–	a
s_2	a	–	$\bar{a}$	–
s_3	–	–	$\bar{a}$	a

Таблица размещения состояний

Состояния	z_1 z_2
s_0	0 0
s_1	1 0
s_2	0 1
s_3	1 1

Автоматные уравнения:

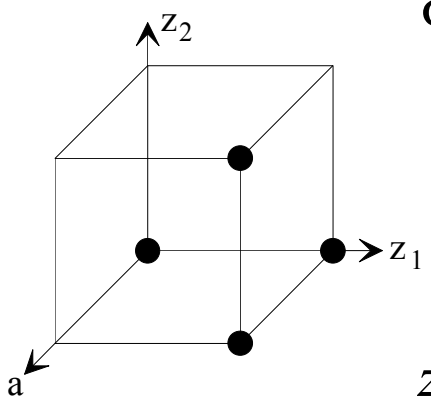
$$\begin{cases} z_1' = f_1(z_1, z_2, a), \\ z_2' = f_2(z_1, z_2, a). \end{cases}$$

Процедура проектирования простых дискретных устройств

Таблица истинности

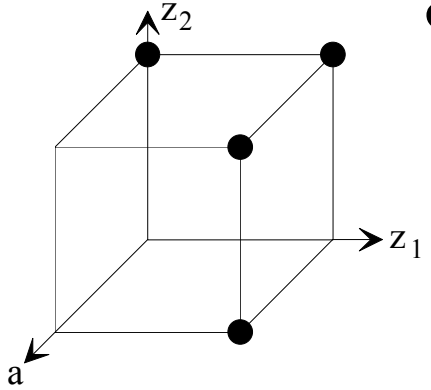
a	z_1	z_2	z_1'	z_2'
0	0	0	1	0
1	0	0	0	0
0	1	0	1	0
1	1	0	1	1
0	0	1	0	1
1	0	1	0	0
0	1	1	0	1
1	1	1	1	1

Функция z_1'



$$z_1' = az_1 \vee \bar{a}\bar{z}_2 = \overline{\overline{az_1}(a \vee z_2)}$$

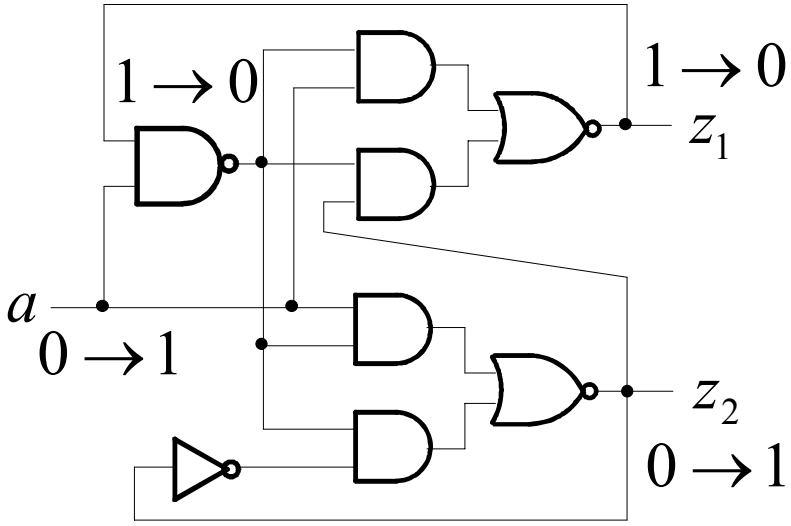
Функция z_2'



$$z_2' = az_1 \vee \bar{a}z_2 = \overline{\overline{az_1}(a \vee \bar{z}_2)}$$

Принимая во внимание, что z_1' и z_2' – это переменные z_1 и z_2, но только в следующий момент времени, можно построить логическую схему.

Процедура проектирования простых дискретных устройств



К сожалению, эта схема неработоспособна. Она содержит логические состязания: когда автомат находится в состоянии 10 и приходит сигнал $a = 1$, только выход z_2 должен переключить свое значение, но в этом случае выход z_1 также может переключиться в 0, что недопустимо.

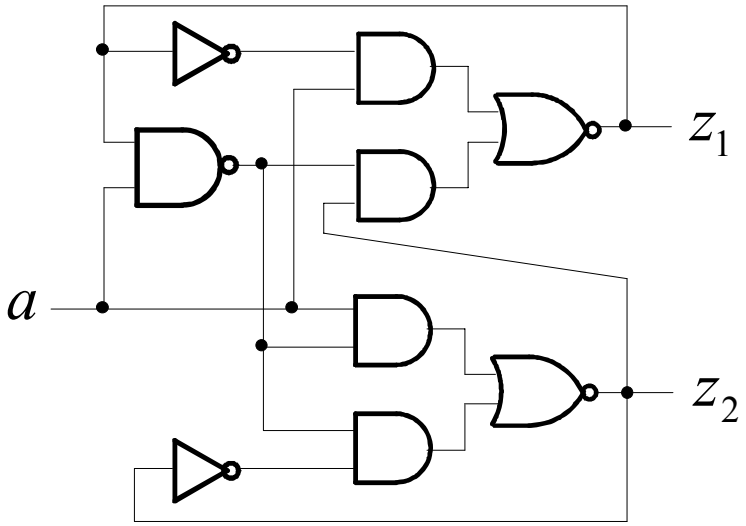
Необходимо изменить реализацию функций. Пусть они будут:

$$z_1' = az_1 \vee \bar{a}\bar{z}_2 = \overline{a\bar{z}_1 \vee az_1z_2},$$

$$z_2' = az_1 \vee \bar{a}z_2 = \overline{\overline{az_1}(a \vee \bar{z}_2)}.$$

Процедура проектирования простых дискретных устройств

Схема, соответствующая реализациям этих функций, является правильной.



В КМОП-технологии эта схема может быть построена с использованием только 20 транзисторов. Известный и широко распространенный Гарвардский Т-триггер содержит 26 транзисторов.

Таким образом, несмотря на то, что общепринятая процедура синтеза автоматов дает хорошие результаты, если есть желание изобрести схему, обладающую неординарными свойствами, следует подумать, как внести изменения в эту процедуру.

Декомпозиция автоматов

Проблема декомпозиции автоматов

- Обычно под ***декомпозицией*** автоматов понимают представление сложного автомата в виде структуры, состоящей из нескольких более простых автоматов.

Часто возникает проблема ***оптимальной декомпозиции***, связанная с нахождением минимального числа элементарных автоматов, образующих сложный автомат.

- Функционирование декомпозированного сложного автомата обычно представлено ***параллельной***, ***последовательной*** или ***смешанной*** работой составляющих автоматов.

Сложный автомат декомпозируется по операциям ***умножение***, ***сложение***, ***суперпозиция*** или по нескольким операциям. Следовательно, возможно рассматривать несколько типов декомпозиции: ***параллельная***, ***последовательная*** и ***смешанная***.

- ***Параллельная декомпозиция*** – это представление автомата в виде:

– произведения двух или более автоматов (автоматы работают ***параллельно***),

– суммы двух или более автоматов (автоматы работают ***альтернативно***, или по очереди).

- ***Последовательная декомпозиция*** – декомпозиция автомата по операции суперпозиции.

- ***Смешанная декомпозиция*** – декомпозиция автомата одновременно по двум операциям (например, умножения и суперпозиции).

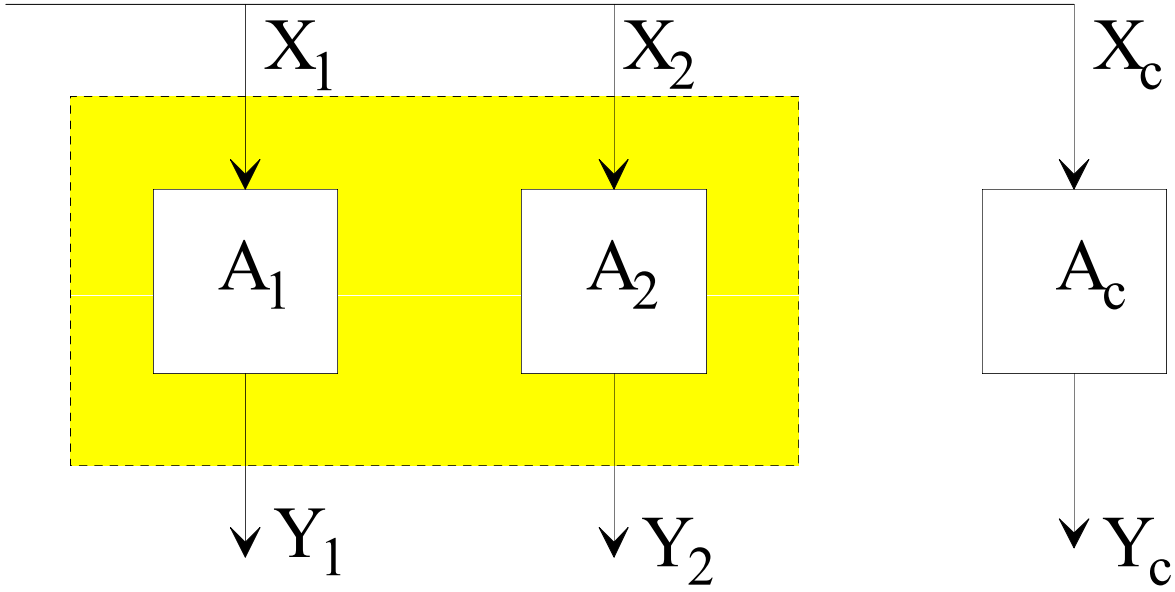
Доля автоматов, имеющих параллельную, последовательную или смешанную декомпозиции, очень мала. Поэтому существует понятие ***общей декомпозиции***, которую понимают как произвольную композицию двух или более автоматов. Параллельная, последовательная и смешанная декомпозиция являются частными случаями общей декомпозиции.

Декомпозиция автоматов

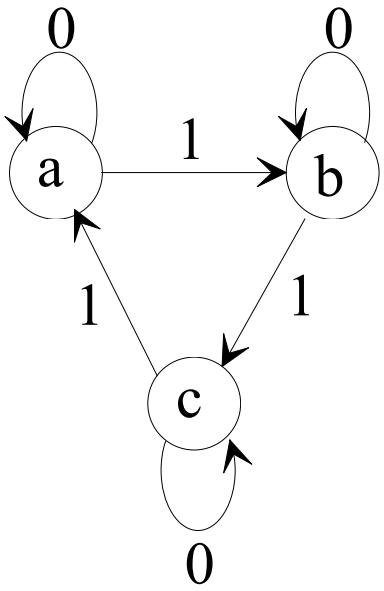
Рассмотрим сначала параллельную композицию

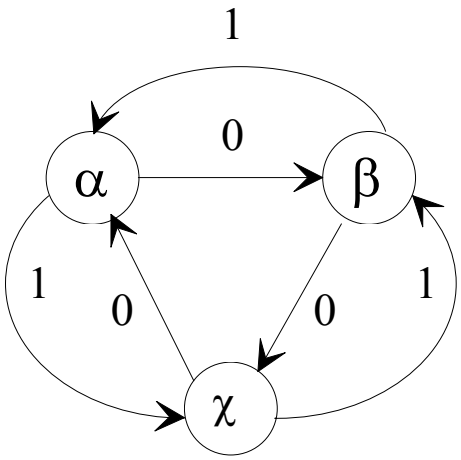
Умножение:

$$X_1 = X_2 = X_c;\ \ Y_c = Y_1 \times Y_2;\ \ S_c = S_1 \times S_2$$



Декомпозиция автоматов





	a	b	c
a	$\bar{x}$	x	0
b	0	$\bar{x}$	x
c	x	0	$\bar{x}$

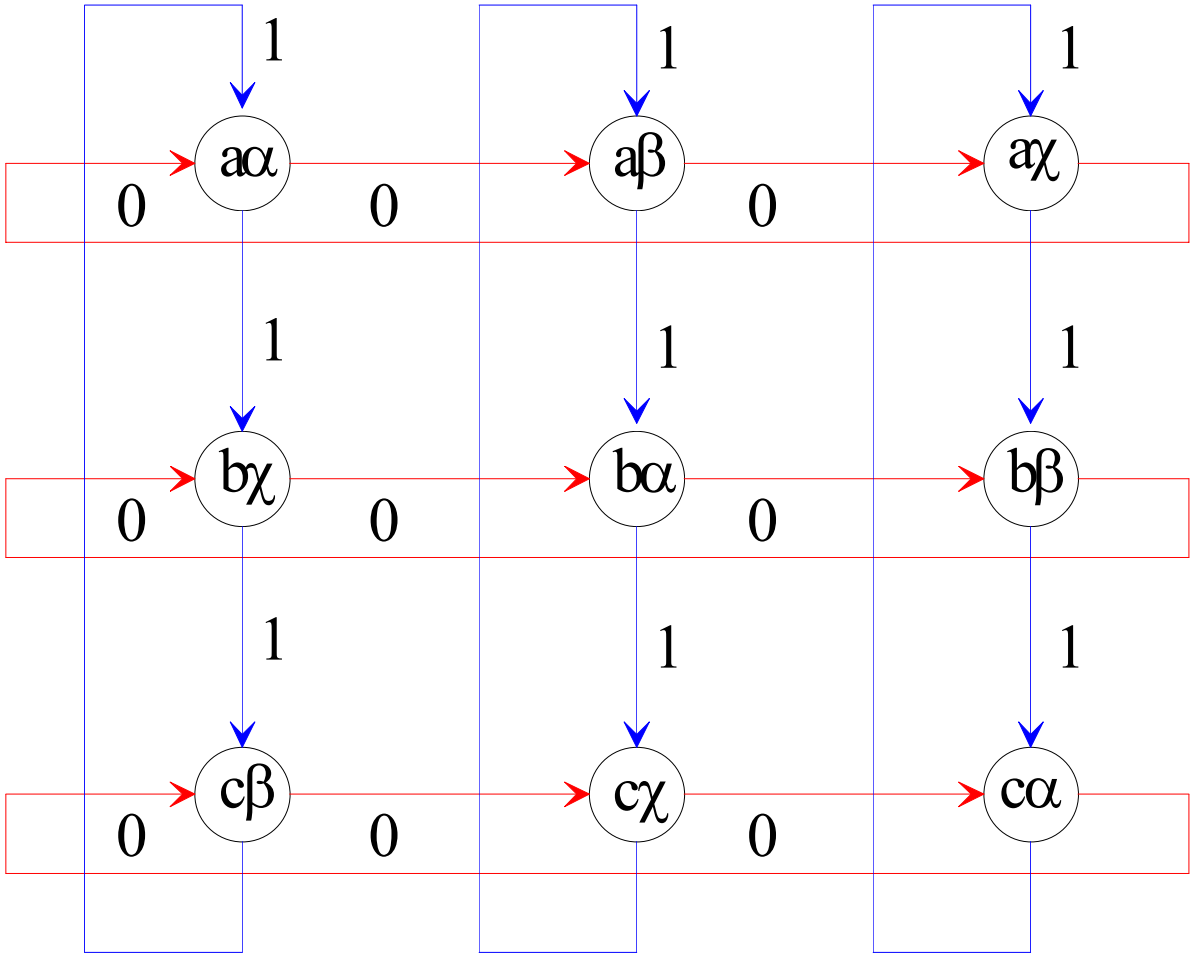
	α	β	χ
α	0	$\bar{x}$	x
β	x	0	$\bar{x}$
χ	$\bar{x}$	x	0

Декомпозиция автоматов

$$X_c = X_1 = X_2$$

	$a\alpha$	$b\alpha$	$c\alpha$	$a\beta$	$b\beta$	$c\beta$	$a\chi$	$b\chi$	$c\chi$
$a\alpha$	0	0	0	$\overline{x}$	0	0	0	x	0
$b\alpha$	0	0	0	0	$\overline{x}$	0	0	0	x
$c\alpha$	0	0	0	0	0	$\overline{x}$	x	0	0
$a\beta$	0	x	0	0	0	0	$\overline{x}$	0	0
$b\beta$	0	0	x	0	0	0	0	$\overline{x}$	0
$c\beta$	x	0	0	0	0	0	0	0	$\overline{x}$
$a\chi$	$\overline{x}$	0	0	0	x	0	0	0	0
$b\chi$	0	$\overline{x}$	0	0	0	x	0	0	0
$c\chi$	0	0	$\overline{x}$	x	0	0	0	0	0

Декомпозиция автоматов

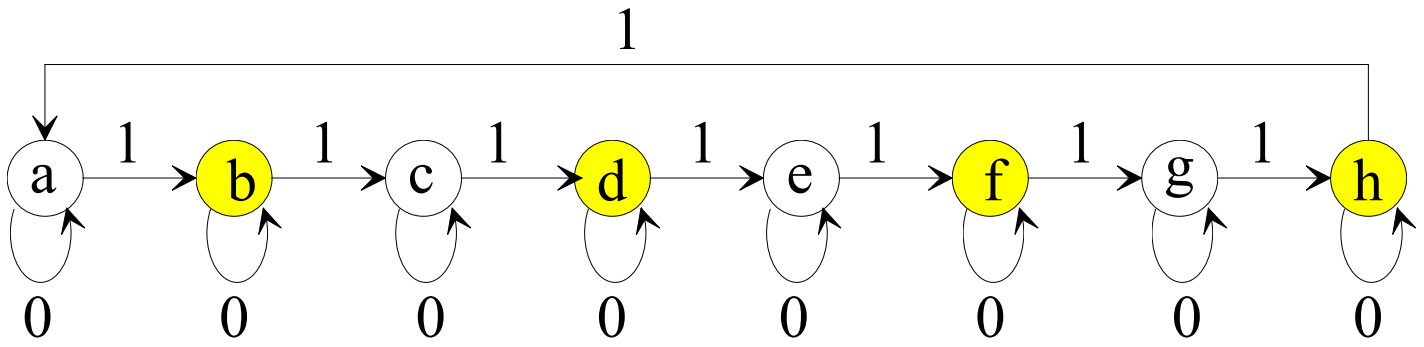


Декомпозиция автоматов

$$X_c = X_1 \times X_2$$

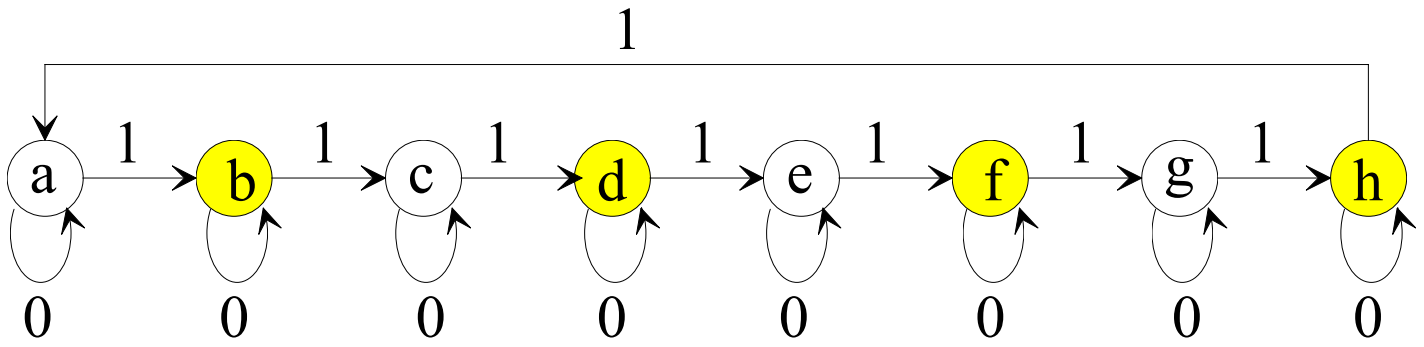
	aα	bα	cα	aβ	bβ	cβ	aχ	bχ	cχ
aα	0	0	0	$\bar{x}_1\bar{x}_2$	$x_1\bar{x}_2$	0	$\bar{x}_1 x_2$	$x_1 x_2$	0
bα	0	0	0	0	$\bar{x}_1\bar{x}_2$	$x_1\bar{x}_2$	0	$\bar{x}_1 x_2$	$x_1 x_2$
cα	0	0	0	$x_1\bar{x}_2$	0	$\bar{x}_1\bar{x}_2$	$x_1 x_2$	0	$\bar{x}_1 x_2$
aβ	$\bar{x}_1 x_2$	$x_1 x_2$	0	0	0	0	$\bar{x}_1\bar{x}_2$	$x_1\bar{x}_2$	0
bβ	0	$\bar{x}_1 x_2$	$x_1 x_2$	0	0	0	0	$\bar{x}_1\bar{x}_2$	$x_1\bar{x}_2$
cβ	$x_1 x_2$	0	$\bar{x}_1 x_2$	0	0	0	$x_1\bar{x}_2$	0	$\bar{x}_1\bar{x}_2$
aχ	$\bar{x}_1\bar{x}_2$	$x_1\bar{x}_2$	0	$\bar{x}_1 x_2$	$x_1 x_2$	0	0	0	0
bχ	0	$\bar{x}_1\bar{x}_2$	$x_1\bar{x}_2$	0	$\bar{x}_1 x_2$	$x_1 x_2$	0	0	0
cχ	$x_1\bar{x}_2$	0	$\bar{x}_1\bar{x}_2$	$x_1 x_2$	0	$\bar{x}_1 x_2$	0	0	0

Декомпозиция автоматов



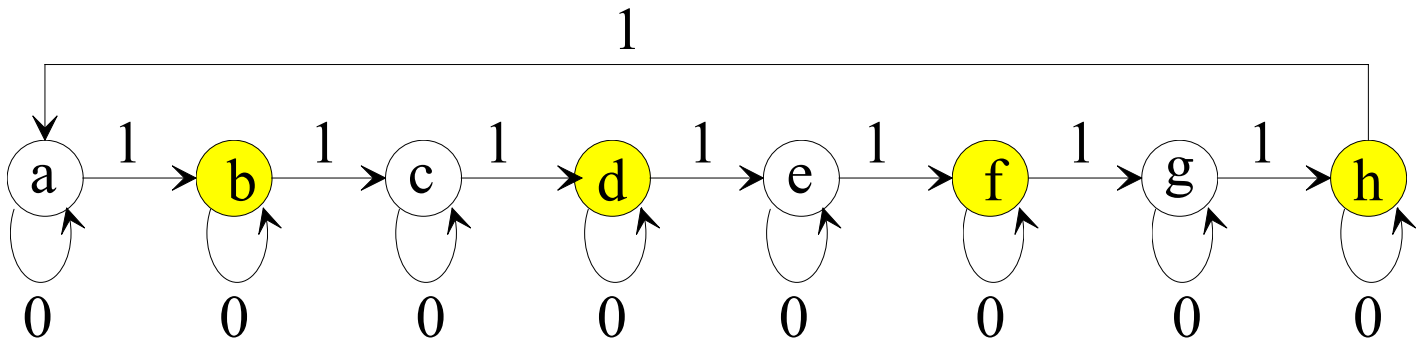
	a	b	c	d	e	f	g	h
a	$\overline{x}$	x	0	0	0	0	0	0
b	0	$\overline{x}$	x	0	0	0	0	0
c	0	0	$\overline{x}$	x	0	0	0	0
d	0	0	0	$\overline{x}$	x	0	0	0
e	0	0	0	0	$\overline{x}$	x	0	0
f	0	0	0	0	0	$\overline{x}$	x	0
g	0	0	0	0	0	0	$\overline{x}$	x
h	x	0	0	0	0	0	0	$\overline{x}$

Декомпозиция автоматов



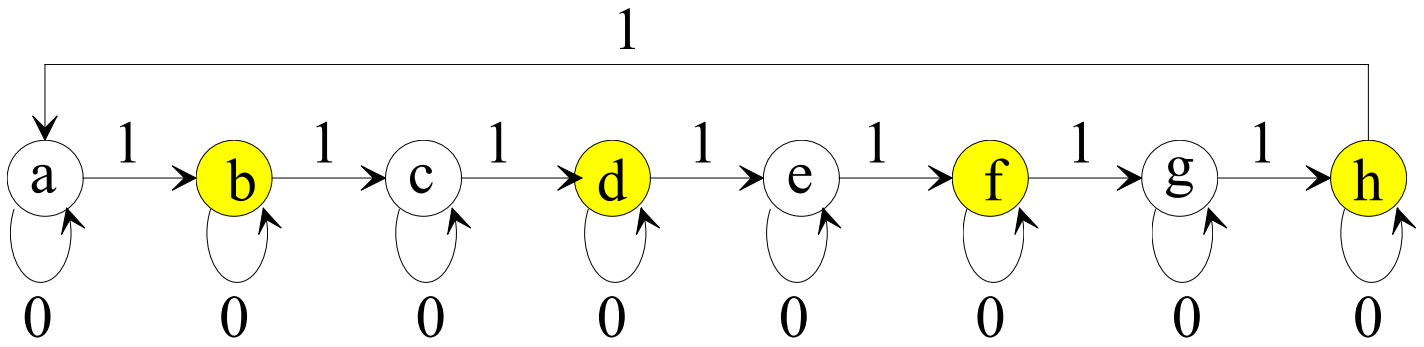
	a	c	e	g	b	d	f	h
a	$\overline{x}$	0	0	0	x	0	0	0
c	0	$\overline{x}$	0	0	0	x	0	0
e	0	0	$\overline{x}$	0	0	0	x	0
g	0	0	0	$\overline{x}$	0	0	0	x
b	0	x	0	0	$\overline{x}$	0	0	0
d	0	0	x	0	0	$\overline{x}$	0	0
f	0	0	0	x	0	0	$\overline{x}$	0
h	x	0	0	0	0	0	0	$\overline{x}$

Декомпозиция автоматов



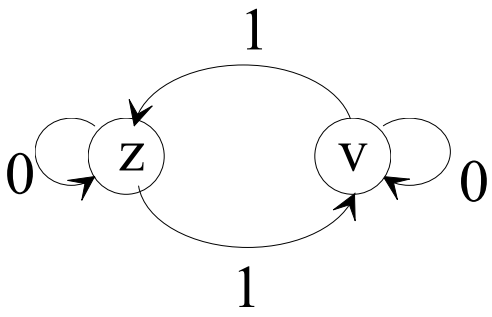
	αz	βz	γz	δz	αv	βv	γv	δv
αz	$\overline{x}$	0	0	0	x	0	0	0
βz	0	$\overline{x}$	0	0	0	x	0	0
γz	0	0	$\overline{x}$	0	0	0	x	0
δz	0	0	0	$\overline{x}$	0	0	0	x
αv	0	x	0	0	$\overline{x}$	0	0	0
βv	0	0	x	0	0	$\overline{x}$	0	0
γv	0	0	0	x	0	0	$\overline{x}$	0
δv	x	0	0	0	0	0	0	$\overline{x}$

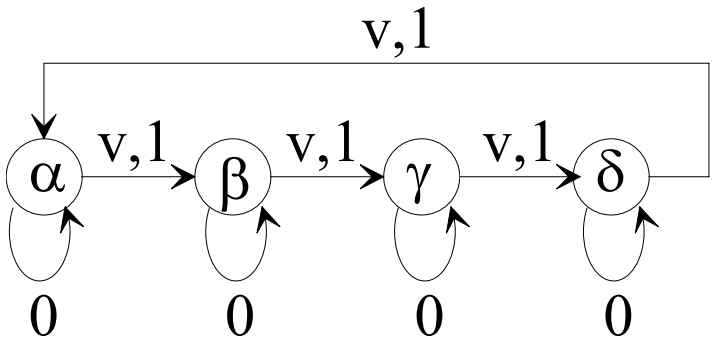
Декомпозиция автоматов



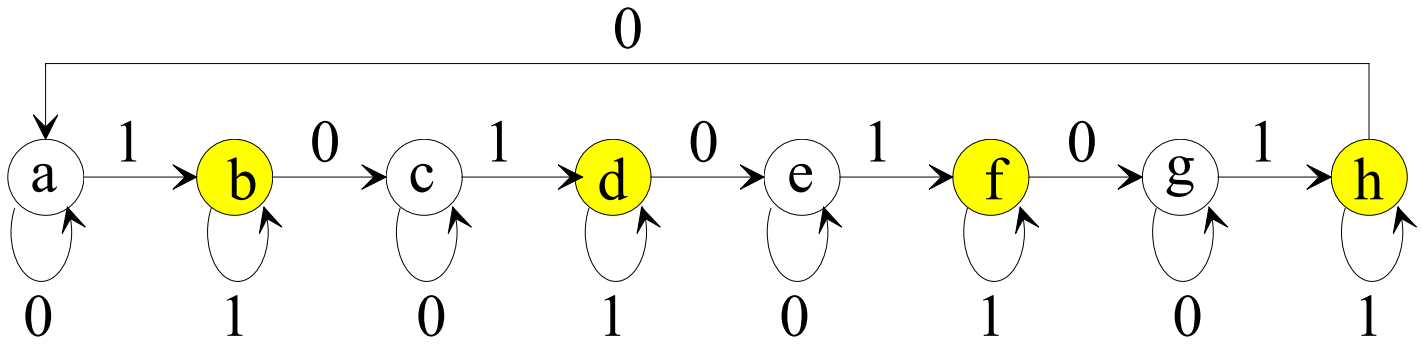
$$a = \alpha z; \quad c = \beta z; \quad e = \gamma z; \quad g = \delta z;$$

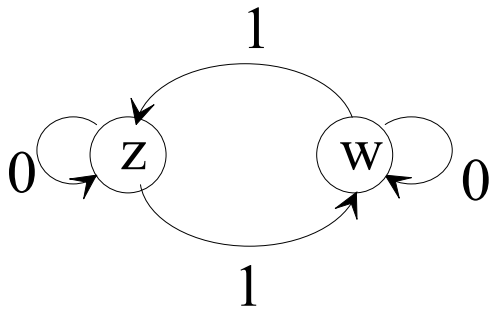
$$b = \alpha v; \quad d = \beta v; \quad f = \gamma v; \quad h = \delta v;$$

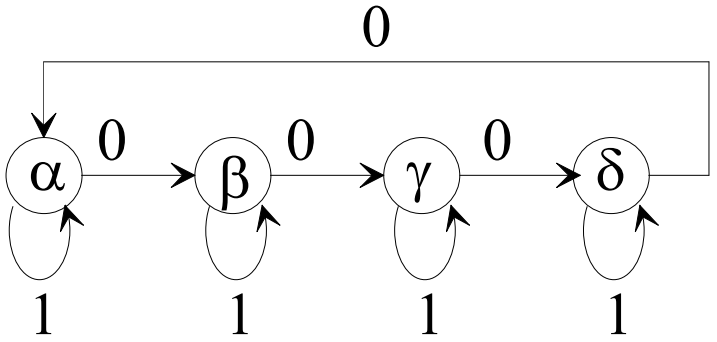




Декомпозиция автоматов







Размещение состояний автомата

Введение

Для автомата $M = < I, O, S, F, \Phi >$ поставим в соответствие:

– каждому входному символу $i \in I$ набор значений входных двоичных переменных $x_1, x_2, ..., x_n$, который является единственным для данного символа;

– каждому выходному символу $o \in O$ единственный для данного символа набор значений выходных двоичных переменных $y_1, y_2, ..., y_m$;

– каждому символу внутреннего состояния $s \in S$ единственный для него набор значений двоичных переменных $z_1, z_2, ..., z_k$.

Такое обозначение символов наборами из нулей и единиц называется ***кодированием***.

Размещение состояний автомата

Тогда функция выходов автомата $F = \{f_1, f_2, ..., f_m\}$ представляет собой множество из m выходных функций

$$y_i = f_i(x_1, x_2, ..., x_n, z_1, z_2, ..., z_k),\ i = 1,2,...,m,$$

И функция переходов состояний автомата $\Phi = \{\varphi_1, \varphi_2, ..., \varphi_k\}$ представляет собой множество из k функций переходов

$$z'_j = \varphi_j(x_1, x_2, ..., x_n, z_1, z_2, ..., z_k),\ j = 1,2,...,k.$$

Обычно, когда говорят о кодировании автомата, имеют в виду кодирование его состояний. Для этого имеются две причины.

Во-первых, методы, разработанные для кодирования состояний, также пригодны для кодирования входов и выходов автомата.

Во-вторых, во многих практических случаях коды для входных и выходных символов уже заданы окружающей средой.

Размещение состояний автомата

Другие термины для кодирования состояний: ***размещение состояний,*** или ***назначения состояний***.

- Целью размещения состояний является нахождение решения, упрощающего комбинационную часть автомата. Общая проблема может быть сформулирована следующим образом:

> ***Для заданного автомата M и множества логических элементов с известной стоимостью найти автомат*** $M' \supseteq M$ ***и способ размещения его состояний, ведущий к схеме устройства с возможно меньшей общей стоимостью.***

Эта проблема очень тяжела даже для автомата с малым числом состояний.

Размещение состояний автомата

Если автомат содержит p состояний, для их кодирования необходимо иметь $k =]\log_2 p[$ кодирующих переменных.

Число возможных кодовых наборов равно 2^k. Из них $2^k - p$ кодовых наборов не используются.

Полное число возможных вариантов кодирования равно

$$(2^k)!/(2^k - p)!.$$

Так как существует $k!$ перестановок кодирующих переменных, которые не изменяют сложности комбинационной схемы, можно сократить эту оценку до

$$\frac{2^k!}{(2^k - p)!k!}.$$

Размещение состояний автомата

Если выбросить варианты кодирования, отличающиеся инвертированием переменных (инвертирование переменных дает 2^k наборов), то окончательно получим

$$\frac{2^k!}{(2^k-p)!k!2^k}=\frac{(2^k-1)!}{(2^k-p)!k!}$$

вариантов представителей кодирования состояний.

Эта функция растет очень быстро с ростом числа состояний p .

- ***Эффективных методов решения общей проблемы кодирования состояний не существует, однако некоторые методы, ведущие к получению удовлетворительных вариантов размещения состояний, были найдены . Эти методы основаны на использовании некоторых свойств последовательностных машин.***

Размещение состояний автомата

Рекомендации для размещения состояний

Сформулируем несколько правил, приводящих к кодированию, которое сокращает сложность уравнений для состояний-последо-вателей.

Рассмотрим простой пример автомата с 4 состояниями:

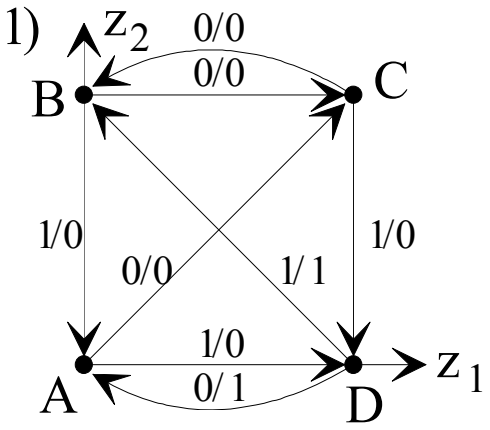
	x = 0	x = 1
A	C/0	D/0
B	C/0	A/0
C	B/0	D/0
D	A/1	B/1

Состояния	Варианты кодов 1 z_1z_2	2 z_1z_2	3 z_1z_2
A	00	00	00
B	01	11	10
C	11	01	01
D	10	10	11

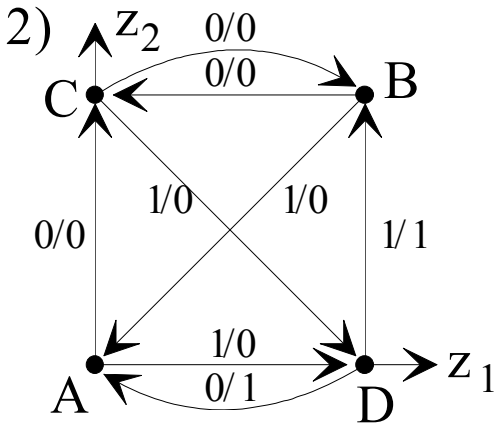
Размещение состояний автомата

Три варианта размещения состояний:

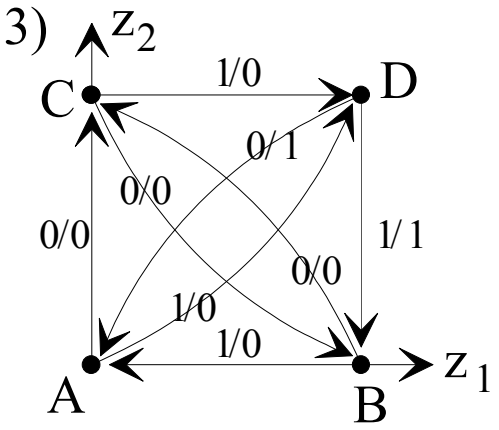
1)



2)



3)



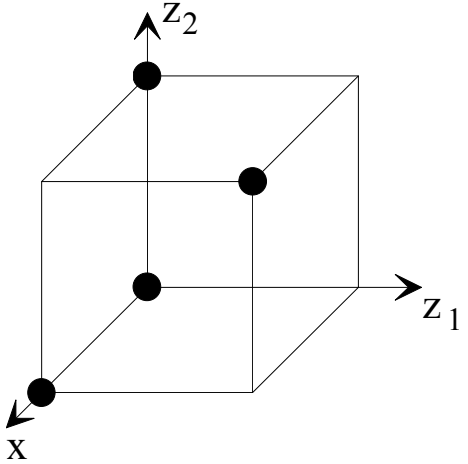
Размещение состояний автомата

Результаты для первого варианта кодирования:

Таблица переходов

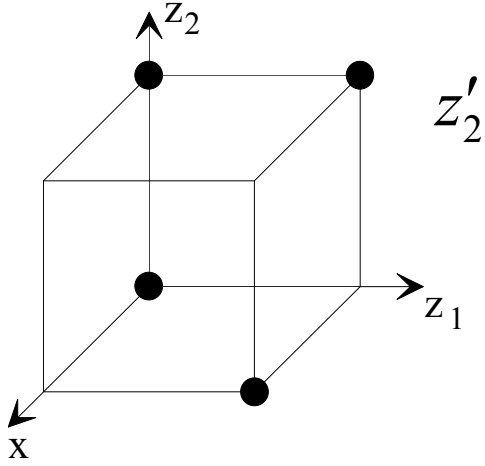
s	$z_1 z_2$	$x=0$	$x=1$
A	00	11/0	10/0
D	10	00/1	01/1
B	01	11/0	00/0
C	11	01/0	10/0

Переключательная функция переменной z_1:



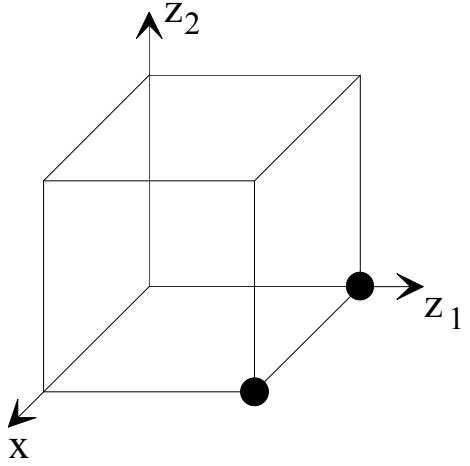
$$z_1' = \bar{z}_1\bar{z}_2 \vee \bar{x}\bar{z}_1 \vee x z_1 z_2$$

Переключательная функция переменной z_2:



$$z_2' = \bar{x}\bar{z}_1 \vee \bar{x} z_2 \vee x z_1 \bar{z}_2$$

Функция выхода y:



$$y = z_1\bar{z}_2$$

$$O(z_1', z_2', y) = 16$$

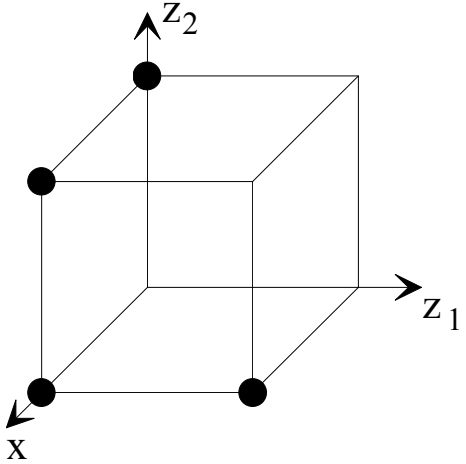
Размещение состояний автомата

Результаты для второго варианта кодирования:

Таблица переходов

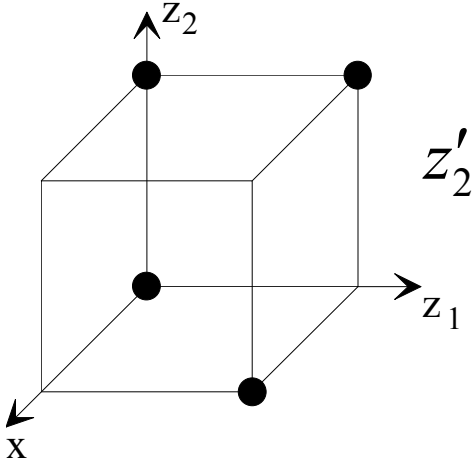
s	$z_1 z_2$	$x=0$	$x=1$
A	00	01/0	10/0
D	10	00/1	11/1
C	01	11/0	10/0
B	11	01/0	00/0

Переключательная функция переменной z_1:



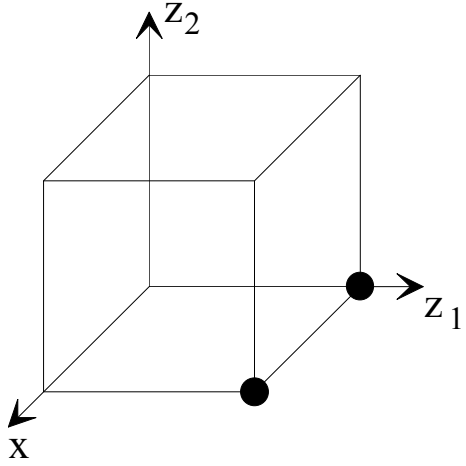
$$z_1' = \bar{z}_1 z_2 \vee x\bar{z}_2$$

Переключательная функция переменной z_2:



$$z_2' = \bar{x}\bar{z}_1 \vee \bar{x} z_2 \vee x z_1 \bar{z}_2$$

Функция выхода y:



$$y = z_1 \bar{z}_2$$

$$O(z_1', z_2', y) = 13$$

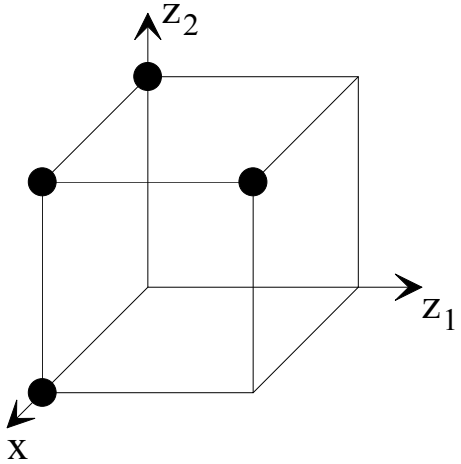
Размещение состояний автомата

Результаты для третьего варианта кодирования:

Таблица переходов

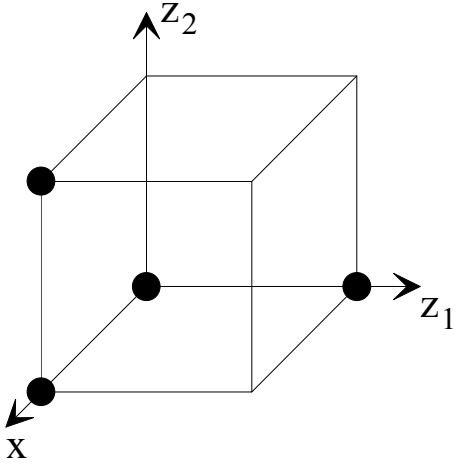
s	$z_1 z_2$	$x=0$	$x=1$
A	00	01/0	11/0
B	10	01/0	00/0
C	01	10/0	11/0
D	11	00/1	10/1

Переключательная функция переменной z_1:



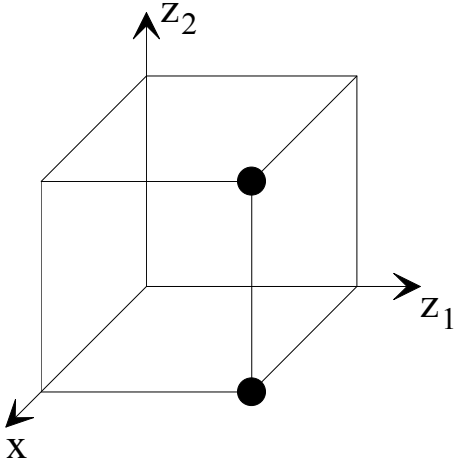
$$z_1' = \bar{z}_1 z_2 \vee x\bar{z}_1 \vee x z_2$$

Переключательная функция переменной z_2:



$$z_2' = x\bar{z}_1 \vee \bar{x}\bar{z}_2$$

Функция выхода y:



$$y = z_1 z_2$$

$$O(z_1', z_2', y) = 12$$

Размещение состояний автомата

Третий вариант кодирования дал лучшие результаты, но почему? По-видимому, потому, что дает лучшую конфигурацию единиц и нулей в графическом представлении логических функций.

Оптимальная конфигурация единиц может быть получена, если принять следующие общие правила размещения состояний.

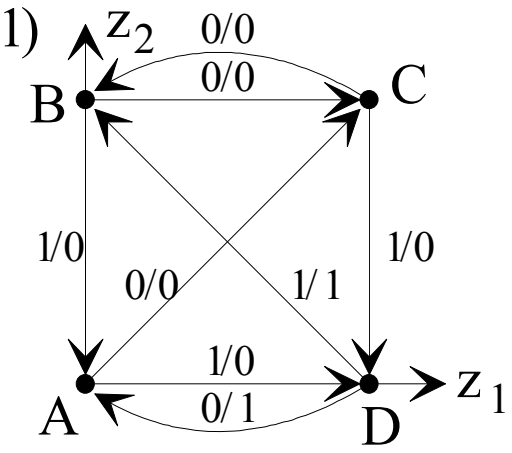
Правило 1. ***Состояния, имеющие одно и то же следующее состояние для данного входа, следует кодировать соседними кодами.***

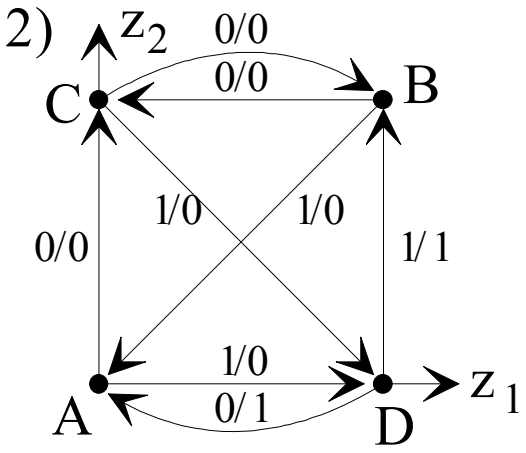
Правило 2. ***Состояния, которые являются непосредственными последователями одного состояния по логически соседним входам, следует кодировать логически соседними кодами.***

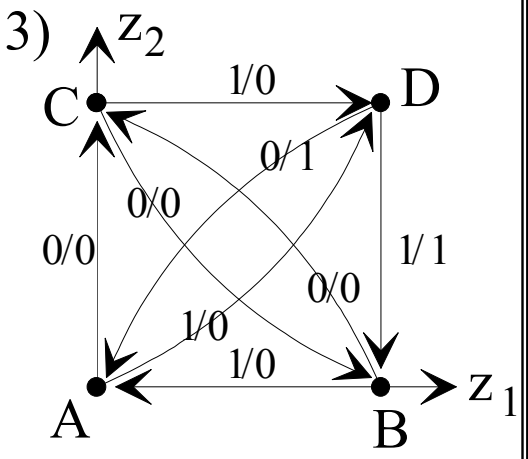
Размещение состояний автомата

Для рассматриваемого примера правило 1 дает, что состояния *A* и *B* должны быть соседними (*A* adj *B*), так как они оба переходят в состояние *C* под воздействием входа, равного 0. То же самое относится и к состояниям A и C (*A* adj *C*).

Правило 2 дает: *A* adj. *B*, *A* adj. *C*, *B* adj. *D* и *C* adj. *D*.







Размещение состояний автомата

• В общем случае не всегда можно удовлетворить все требования соседства кодирующих наборов, предлагаемые правилами 1 и 2. Однако следует попытаться удовлетворить их в наибольшей степени. В случае возникновения конфликтов предпочтение следует отдавать правилу1, что обеспечивает лучшую конфигурацию единиц в гиперкубе.

Другим средством, обеспечивающим хорошее размещение состояний, является ***граф импликаций***.

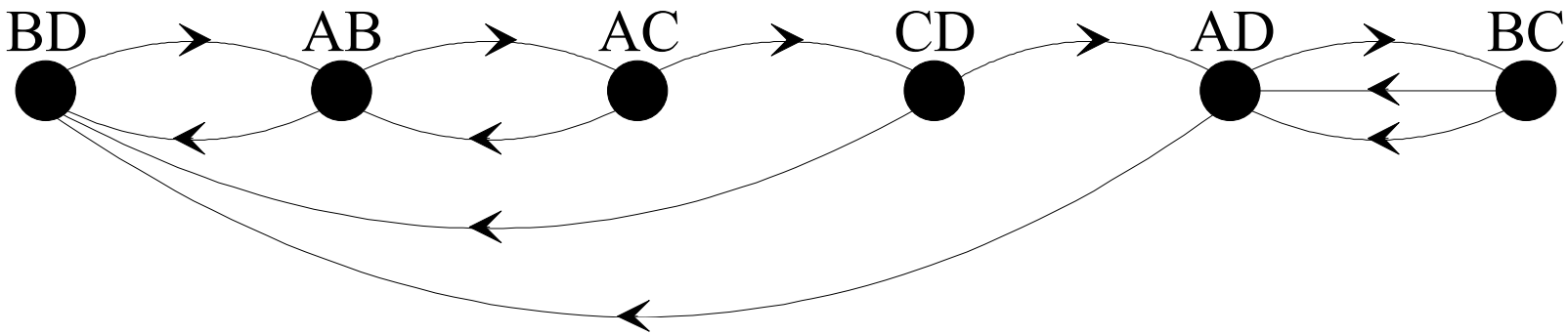
Вершины графа импликаций представляют пары состояний. Вершины соединяются направленными дугами, каждая из которых представляет переходы из одной пары состояний в другую под воздействием данного входа в соответствии с таблицей переходов состояний.

Размещение состояний автомата

Пример:

	x = 0	x = 1
A	B/0	C/0
B	D/0	A/1
C	A/1	D/0
D	D/1	B/1

Автомат с 4 состояниями



Граф импликаций

В графе импликаций опущены циклические дуги.

Размещение состояний автомата

Как граф импликаций может быть использован при размещении состояний, покажем на примере.

Применение правил 1 и 2 к таблице состояний дает:

Правило 1: *B* adj. *D*.

Правило 2: *B* adj. *C*, *A* adj. *D*, *D* adj. *B*.

Некоторые из этих требований могут быть учтены при размещении состояний.

Наиболее важным соседством является *B* adj. *D*. Если выбрать это соседство, согласно графу импликаций желательно сделать *A* adj. *B* (это соседство не предлагается ни правилом 1, ни правилом 2). Граф импликаций также советует *A* adj. *C* и *C* adj. *D*.

Размещение состояний автомата

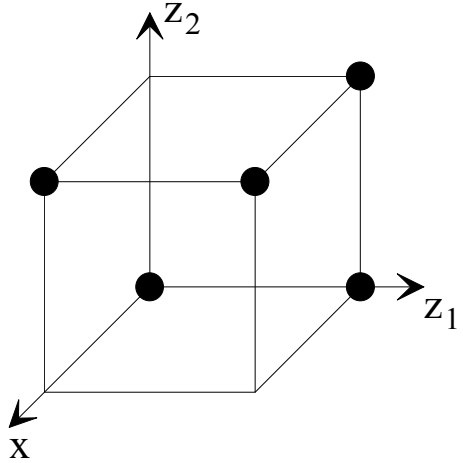
Эти соседства соответствуют единственно возможному кодированию, которое дает наиболее простое представление переключательных функций для внутренних переменных автомата. Любое другое размещение состояний (включая соседства, предлагаемые правилом 2) дает более сложные функции.

Найдем переключательные функции кодирующих переменных.

Таблица переходов

s	$z_1 z_2$	$x=0$	$x=1$
A	00	10/0	01/0
B	10	11/0	00/1
C	01	00/1	11/0
D	11	11/1	10/1

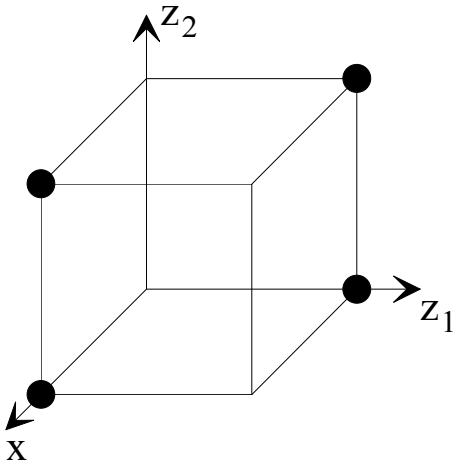
Переключательная функция переменной z_1



$$z_1' = \bar{x} z_1 \vee \bar{x} \bar{z}_2 \vee x z_2$$

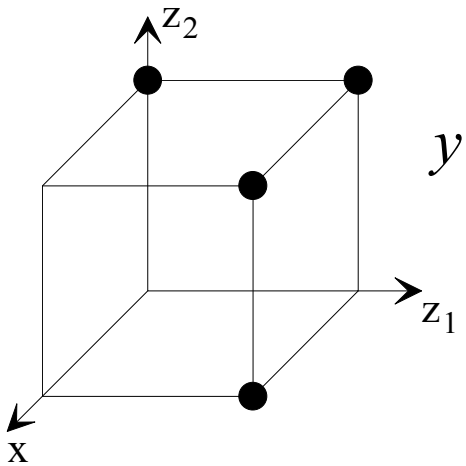
Размещение состояний автомата

Переключательная функция z_2



$$z_2' = x\bar{z}_1 \vee \bar{x}z_1$$

Функция выхода y



$$y = xz_1 \vee \bar{x}z_2$$

$$O(z_1', z_2', y) = 14$$

Этот пример размещения состояний иллюстрирует концепцию установления ***цепи*** пар соседств (пар смежности). Цепь строится в соответствии с дугами переходов графа импликаций.

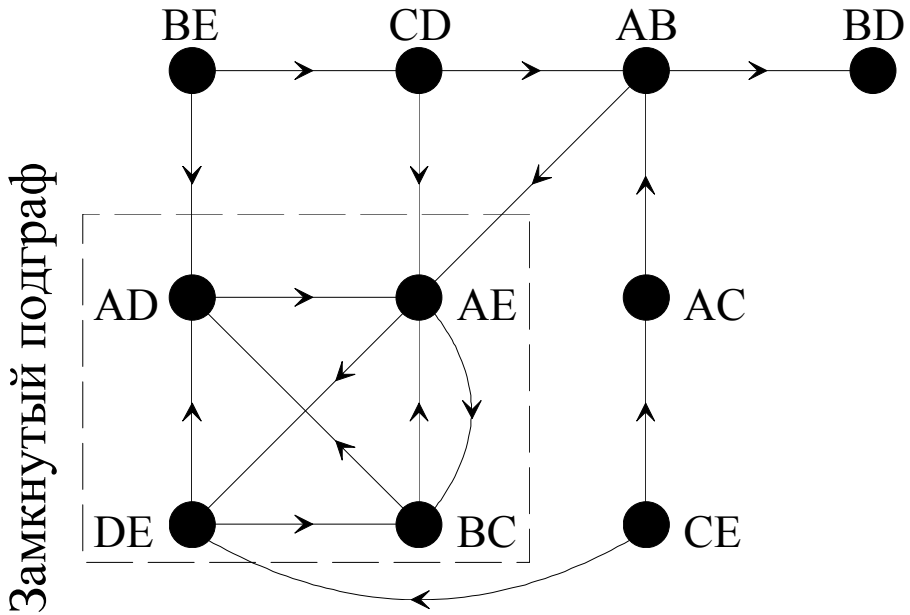
Говорят, что граф импликация является ***полным***, если он содержит все возможные пары состояний. Обычно имеют дело ***частично полным*** (или неполным) графом импликаций.

Размещение состояний автомата

Подграф графа импликаций называется ***замкнутым***, если, во-первых, его вершины содержат все состояния автомата и, во-вторых, все выходные дуги каждой вершины подграфа направлены только к вершинам самого подграфа.

	x = 0	x = 1
A	E/0	B/0
B	A/1	D/1
C	E/0	A/0
D	A/0	B/1
E	D/0	C/0

Автомат с 5 состояниями



Граф импликаций

Размещение состояний автомата

Идеальные цепи для кодирования образуются только замкнутыми подграфами. Если невозможно найти замкнутый подграф, то следует выделить связный подграф, содержащий большое количество дуг переходов.

***Правило 3*: Используя соседства , предлагаемые правилами 1 и 2, построить частично полный граф импликаций. Затем попытаться выделить цепь пар смежности в замкнутом или связном подграфе.**

Заметим, что правила 1 и 3 более важны, чем правило 2.

Как в действительности выбрать кодирование? В основном пытаются минимизировать число логических единиц на кубах функций и максимизировать число произвольных заданий.

Размещение состояний автомата

Следующее правило помогает достичь этой цели:

***Правило 4*: Найти в таблице переходов наиболее часто используемое состояние. Поместить его в нулевую вершину кодирующего куба. Размещать другие состояния в кодирующем кубе согласно правилам 1, 3 и 2, сохраняя вершины с двоичными номерами, имеющими большее число единиц для размещения следующих состояний. Попытаться удовлетворить как можно больше требований к соседству выбираемых назначений.**

На этом можно закончить рекомендации по размещению состояний.



Логическое Проектирование

Лекция 13

Декомпозиция и кодирование автоматов

Можно показать, что декомпозиция автоматов и кодирование состояний являются ***существенно связанными проблемами***.

Один из подходов к кодированию состояний имеет целью сокращение взаимозависимости кодирующих переменных

$$z'_j = \varphi_j(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_k), \quad j = 1,2,\ldots,k,$$

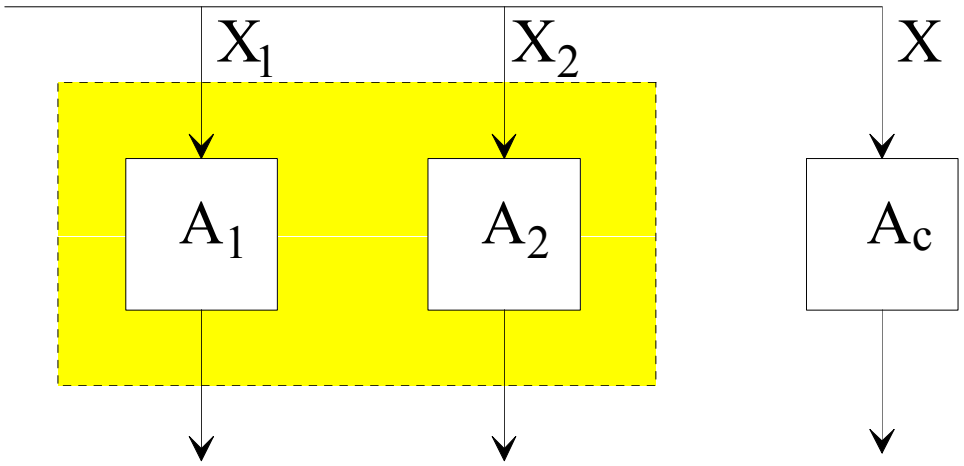
а не минимизацию числа литералов в их переключательных функциях. В результате достигаются обе цели: сокращается число литералов и упрощаются взаимосвязи.

Например, благодаря тщательному кодированию автомата с четырьмя внутренними переменными получилась следующая система функций переходов:

Декомпозиция и кодирование

$$z_1' = \varphi_j(x_1, x_2, \ldots, x_n, z_1, z_2),$$
$$z_2' = \varphi_j(x_1, x_2, \ldots, x_n, z_1, z_2),$$
$$z_3' = \varphi_j(x_1, x_2, \ldots, x_n, z_3, z_4),$$
$$z_4' = \varphi_j(x_1, x_2, \ldots, x_n, z_3, z_4).$$

Если не обращать внимания на функцию выхода, эта система описывает два независимых автомата с общими входными переменными. Эти автоматы работают параллельно.



$$X_1 = X_2 = X;$$
$$S_c = S_1 \times S_2$$

Умножение автоматов

Декомпозиция и кодирование

Разбиения

Каждая кодирующая переменная разбивает множество состояний автомата на два разделительных подмножества, пара переменных – на четыре подмножества, три переменных – на восемь и т.д.

s	z_1	z_2	z_3
A	0	0	0
E	1	0	0
D	0	1	0
C	0	0	1
B	0	1	1

$$z_1 \to \{\overset{z_1=0}{\overline{A, D, C, B}}; \overset{z_1=1}{\overline{E}}\}, \quad z_2 \to \{\overset{z_2=0}{\overline{A, E, C}}; \overset{z_2=1}{\overline{D, B}}\},$$

$$z_3 \to \{\overset{z_3=0}{\overline{A, E, D}}; \overset{z_3=1}{\overline{C, B}}\},$$

$$z_1z_2 \to \{\overset{z_1z_2=00}{\overline{A, C}}; \overset{z_1z_2=10}{\overline{E}}; \overset{z_1z_2=01}{\overline{B, D}}\}, \quad z_1z_3 \to \{\overset{z_1z_3=00}{\overline{A, D}}; \overset{z_1z_3=10}{\overline{E}}; \overset{z_1z_3=01}{\overline{B, C}}\},$$

$$z_2z_3 \to \{\overset{z_2z_3=00}{\overline{A, E}}; \overset{z_2z_3=10}{\overline{D}}; \overset{z_2z_3=01}{\overline{C}}; \overset{z_2z_3=11}{\overline{B}}\},$$

$$z_1z_2z_3 \to \{\overline{A}; \overline{B}; \overline{C}; \overline{D}; \overline{E}\}.$$

Декомпозиция и кодирование

Разбиением π множества *S* называется множество разделительных подмножеств (***блоков***) множества *S*:

$$\pi = \{B_i\}, \quad B_i \cap B_j = \phi \text{ for } i \neq j, \bigcup_i B_i = S.$$

Например:

$$S = \{1,2,3,4,5,6,7,8\},$$

$$\pi = \{B_1, B_2, B_3\} = \{\overline{1,2,4,5}; \overline{3,6}; \overline{7,8}\}.$$

Введем несколько определений.

Обозначим как **0** и **1** элементы тривиальных разбиений,

например, если S={0,1,2,3,4} то

$$0 = \{\overline{1};\overline{2};\overline{3};\overline{4}\} \text{ и } 1 = \{\overline{1,2,3,4}\}.$$

Декомпозиция и кодирование

Можно ввести ***отношение частичного порядка*** для пар разбиений на множестве S. Для π_1 и π_2, говорят, что π_2 больше или равно π_1 $(\pi_2 \geq \pi_1)$, если и только если каждый блок в π_1 полностью принадлежит одному из блоков в π_2. Например,

$$\{\overline{1,4};\overline{2,3}\} \geq \{\overline{1};\overline{2,3};\overline{4}\}.$$

Введем две операции на разбиениях:

- $\pi_r = \pi_k \cdot \pi_l$ – умножение. Блоки π_r являются попарными пересечениями блоков, принадлежащих π_k и π_l;
- $\pi_p = \pi_k + \pi_l$ – сложение. Каждый блок π_p включает блоки π_k и π_l, имеющие, по крайней мере, одно общее состояние.

Например, если $S=\{1,2,3,4,5,6,7,8,9\}$ и

$\pi_1 = \{\overline{1,2};\overline{3,4};\overline{5,6};\overline{7,8,9}\}$, $\pi_2 = \{\overline{1,6};\overline{2,3};\overline{4,5};\overline{7,8};\overline{9}\}$ то

$\pi_1 \cdot \pi_2 = \{\overline{1};\overline{2};\overline{3};\overline{4};\overline{5};\overline{6};\overline{7,8};\overline{9}\}$, $\pi_1 + \pi_2 = \{\overline{1,2,3,4,5,6};\overline{7,8,9}\}$.

Декомпозиция и кодирование

Легко видеть, что разбиения, полученные с помощью этих операций связаны с исходными разбиениями отношением частичного порядка.

Все разбиения на множестве состояний могут быть представлены графом следующим образом.

Вершинами графа являются разбиения. Вершина **1** – верхняя вершина графа, первый слой вершин состоит из двухблочных разбиений, второй слой состоит из трех- и четырехблочных разбиений, которые могут быть получены из разбиений первого слоя с помощью операции умножения и т.д. Единственная вершина **0** находится в самом низу графа. Вершины графа соседних слоев связаны направленными дугами в соответствии с отношением частичного порядка.

Декомпозиция и кодирование

Для построения графа снизу вверх используется операция сложения.

- Граф такого типа называется ***алгебраической структурой***, или ***решеткой***.
- Каждый путь из вершины **1** в вершину **0** в решетке определяет вариант кодирования состояний автомата.

Разбиение со свойством подстановки

Рассмотрим автомат M, составленный из двух компонентных автоматов M_1 and M_2. Состояние автомата M задано парой состояний: одно из M_1 и другое из M_2. Объединяя все состояния M с различной первой компонентой в различные блоки, получим первое разбиение. Объединяя все состояния M с различной второй компонентой в различные блоки, получим второе разбиение. Говорят, что эти два разбиения обладают ***свойством подстановки***.

Декомпозиция и кодирование

Если для некоторого автомата можно найти такие два разбиения, то это означает, что он может быть декомпозирован на два компонентных автомата.

- Говорят, что разбиение π множества состояний автомата обладает ***свойством подстановки*** (СП), если и только если под воздействием одного набора значений входных переменных все переходы из состояний любого блока завершаются состояниями, принадлежащими одному блоку.

- ***Теорема***. ***Автомат M имеет нетривиальную параллельную декомпозицию, если и только если существуют два нетривиальных СП-разбиения состояний автомата*** π_1 ***и*** π_2 ***таких, что*** $\pi_1 \cdot \pi_2 = 0$.

Декомпозиция и кодирование

	x = 0	x = 1
0	3/0	2/0
1	5/0	2/0
2	4/0	1/0
3	1/1	4/1
4	0/0	3/0
5	2/0	3/0

Автомат, имеющий параллельную декомпозицию

Пример:

$$\pi_1 = \{\overline{0,1,2};\overline{3,4,5}\}, \quad \pi_2 = \{\overline{0,5};\overline{1,4};\overline{2,3}\}$$

Оба разбиения обладают свойством подстановки и

$$\pi_1 \cdot \pi_2 = \{\overline{0};\overline{1};\overline{2};\overline{3};\overline{4};\overline{5}\} = 0.$$

A={0,1,2}, B={3,4,5}

C={0,5}, D={1,4}, E={2,3}

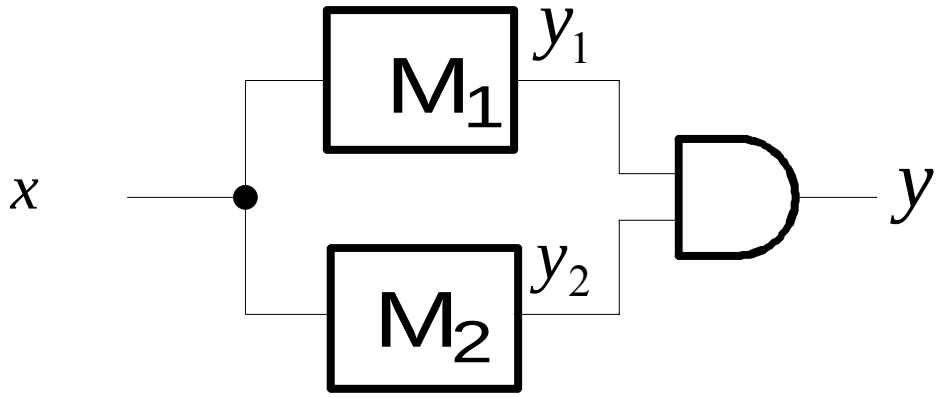
	0	1
A	B/0	A/0
B	A/1	B/1

Автомат M_1

	0	1
C	E/0	E/0
D	C/0	E/0
E	D/1	D/1

Автомат M_2

Декомпозиция и кодирование



$$y_1 = f_1(B)$$
$$y_2 = f_2(E)$$
$$B \bigcap E = 3$$
$$y = f(3) = y_1 y_2$$

Структура параллельной декомпозиции

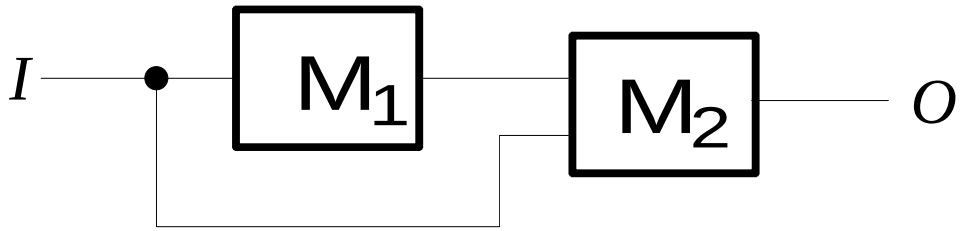
Рассмотрим последовательную декомпозицию автомата.

- ***Теорема*. *Если автомат имеет одно нетривиальное СП-разбиение, то он имеет нетривиальную последовательную декомпозицию*.**

Последовательная декомпозиция состоит из двух компонентных подавтоматов: один подавтомат называется ***независимой компонентой***, а другой – ***зависимой компонентой***.

Декомпозиция и кодирование

Структура последовательной декомпозиции



Зависимая компонента соответствует такому разбиению состояний, что его произведение с СП-разбиением дает тривиальное разбиение **0**.

Пример:

	0	1
1	1/1	4/0
2	3/1	5/0
3	2/0	4/0
4	3/1	1/1
5	2/0	2/1

Автомат, имеющий последовательную декомпозицию

$\pi_1 = \{\overline{1,2,3}; \overline{4,5}\}$ – единственное СП-разбиение. Выберем $\pi_2 = \{\overline{1,4}; \overline{2,5}; \overline{3}\}$ в качестве разбиения для зависимой компоненты. Очевидно, что

$$\pi_1 \cdot \pi_2 = \{\overline{0}; \overline{1}; \overline{2}; \overline{3}; \overline{4}; \overline{5}\} = 0.$$

Декомпозиция и кодирование

Из π_1: A={1,2,3}, B={4,5}

	0	1
A	A	B
B	A	A

Независимая компонента M_1

Из π_2 : C={1,4}, D={2,5}, E={3}

		0	1
A	C	C/1	C/0
	D	E/1	D/0
	E	D/0	C/0
B	C	E/1	C/1
	D	D/0	D/1

Первый шаг в построении зависимой компоненты M_2

	A,0	A,1	B,0	B,1
C	C/1	C/0	E/1	C/1
D	E/1	D/0	D/0	D/1
E	D/0	C/0	–	–

Второй шаг в построении зависимой компоненты M_2

Декомпозиция и кодирование

Вычисление СП-разбиений

СП-разбиения образуют подрешетку решетки всех возможных разбиений.

Сначала необходимо найти ***минимальные СП-разбиения*** и затем суммировать их до тех пор, пока не будут получены все возможные суммы.

Минимальные СП-разбиения – это нетривиальные разбиения, которые имеют не более, чем два состояния в каждом блоке.

Рассмотрим пример:

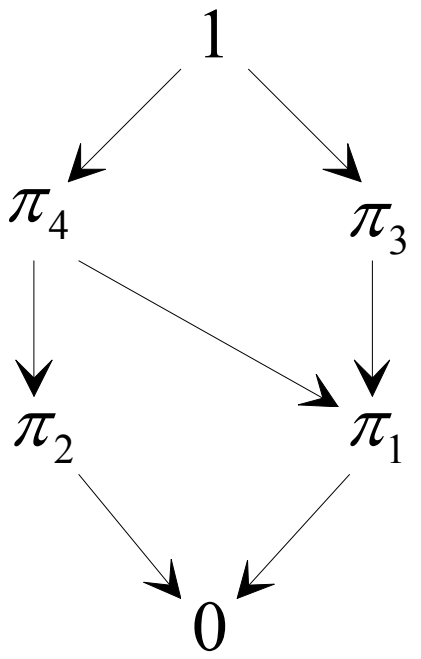
Декомпозиция и кодирование

	0	1
1	1	2
2	4	3
3	5	2
4	1	2
5	1	3

Таблица для нахождения СП-разбиений

Объединяя состояния в пары, находим:

$$\pi_1 = \{\overline{1,4};\overline{2};\overline{3};\overline{5}\}$$
$$\pi_2 = \{\overline{1};\overline{2,3};\overline{4,5}\}$$
$$\pi_3 = \{\overline{1,4};\overline{2,5};\overline{3}\}$$
$$\pi_4 = \pi_1 + \pi_2 = \{\overline{1,4,5};\overline{2,3}\}$$
$$\pi_5 = \pi_3 + \pi_4 = \{\overline{1,2,3,4,5}\} = 1$$



Решетка СП-разбиений

Декомпозиция и кодирование

Общая декомпозиция и кодирование состояний

СП-разбиения позволяют найти последовательную или параллельную декомпозиции. Повторно применяя эти декомпозиции к компонентным автоматам до тех пор, пока каждый компонентный автомат не будет иметь только двух состояний, можно решить проблему размещения состояний.

К сожалению, только очень небольшое число автоматов может быть декомпозировано таким образом. Тем не менее, эта проблема может быть решена с помощью техники, использующей ***пары разбиений***.

- Пара разбиений (π, π') – это упорядоченная пара разбиений множества состояний автомата, которая обеспечивает переходы из состояний любого блока, принадлежащего одному разбиению, в состояния, полностью принадлежащие одному из блоков другого разбиения, под воздействием одного и того же набора значений входных переменных.

Декомпозиция и кодирование

Очевидно, что если (π, π) – пара разбиений, то π обладает свойством подстановки. Следовательно, пары разбиений являются обобщением СП-разбиений.

Рассмотрим пример:

	00	01	10	11
1	1/1	2/1	3/1	4/1
2	3/1	4/1	1/1	2/1
3	2/0	1/0	4/0	3/0
4	4/0	3/0	2/0	1/0

Пример автомата

Легко проверить, что

$$(\pi_1, \pi_2) = (\{\overline{1,2}; \overline{3,4}\}, \{\overline{1,3}; \overline{2,4}\})$$

представляет пару разбиений и

$$\pi_1 \cdot \pi_2 = 0.$$

Для кодирования состояний необходимы две переменные: z_1 и z_2. Пусть z_1 порождает разбиение π_1, а z_2 – разбиение π_2.

Декомпозиция и кодирование

	$z_1 z_2$
$1 \rightarrow$	0 0
$2 \rightarrow$	0 1
$3 \rightarrow$	1 0
$4 \rightarrow$	1 1

Для этого кодирования уравнения автомата имеют вид:

$$z_1' = \bar{x}_1 z_2 \vee x_1 \bar{z}_2,$$

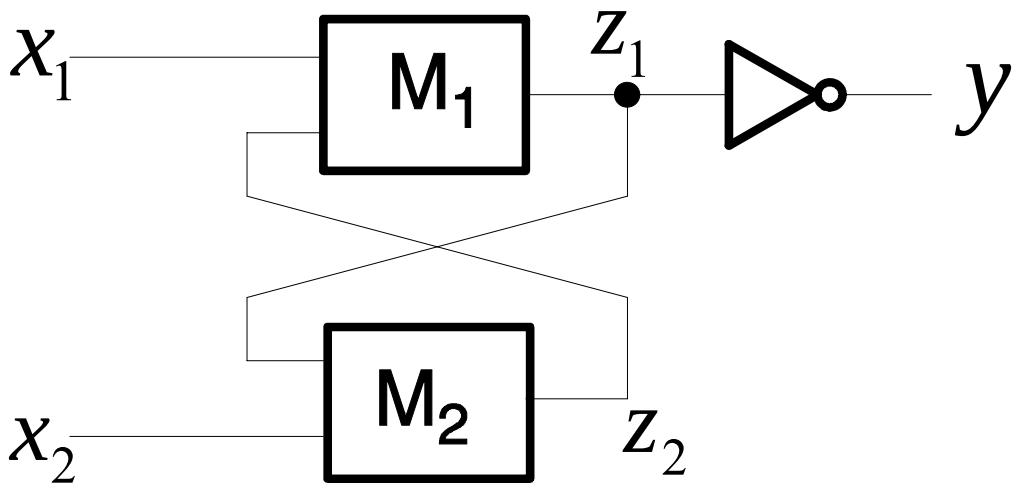
$$z_2' = \bar{x}_2 z_1 \vee x_2 \bar{z}_1,$$

$$y = \bar{z}_1.$$

Этот пример иллюстрирует общий принцип подхода, при котором пары разбиений могут быть использованы для уменьшения зависимости между внутренними переменными. Возможно таким же образом уменьшать зависимость и от входов.

Декомпозиция и кодирование

Блок-схема кодированного таким образом автомата иллюстрирует тип параллельной декомпозиции с перекрестными свозами.



Автоматы, не зависящие от скорости

(Speed-Independent Automata)

Логический элемент (Вентиль)

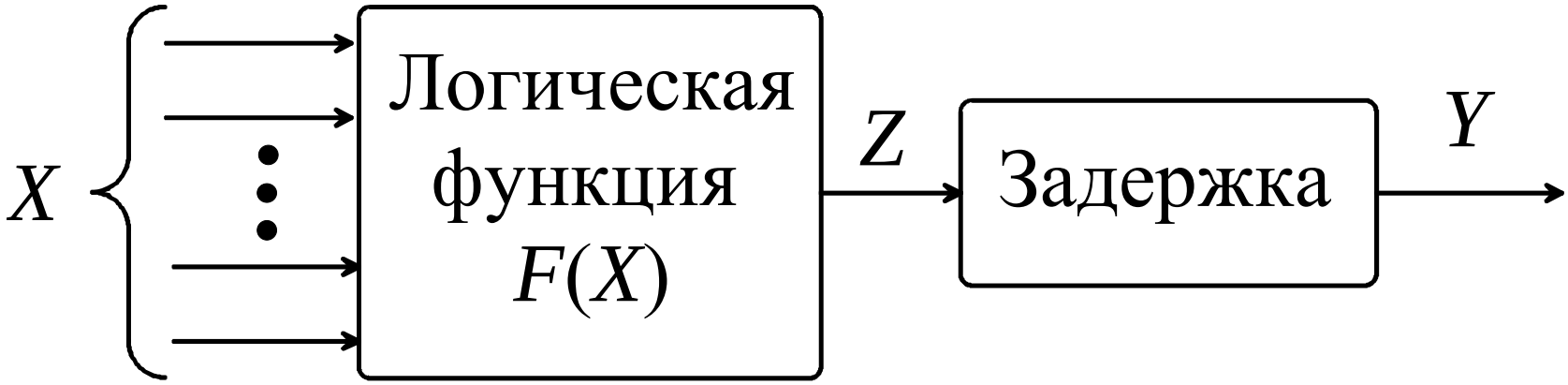
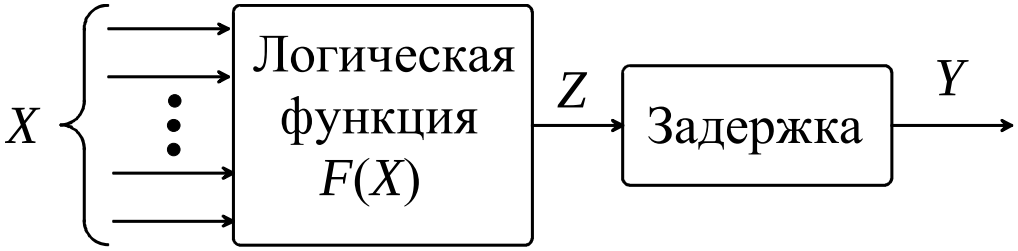


Схема – множество логических элементов (вентилей) вместе с их взаимосоединениями.

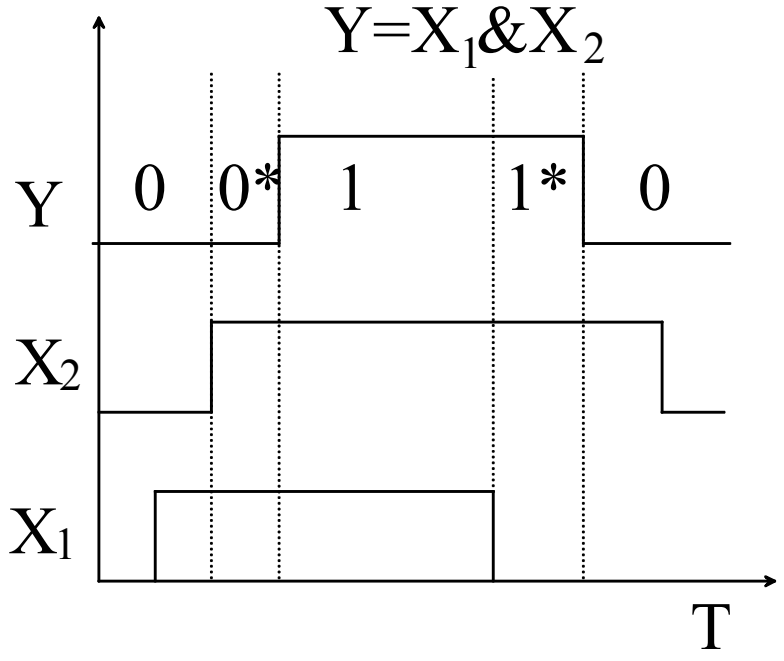
$$y_j = F_j(\boldsymbol{X})$$

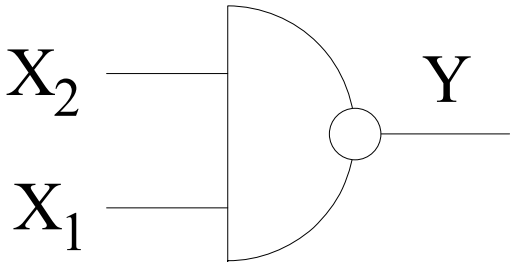


$Y=\{0,1,0^*,1^*\}$

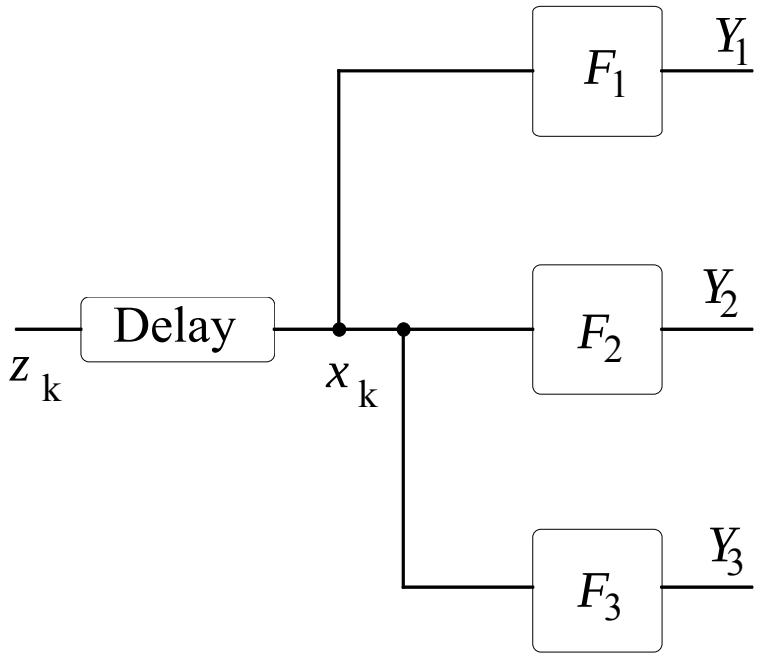
$Y=\{0,1\}$ - устойчивые значения, $Y=F(\boldsymbol{X})=Z$;

$Y=\{0^*,1^*\}$ - неустойчивые значения $Y\neq F(\boldsymbol{X})=Z$





Допущения Маллера



$$F(x_1,\ldots,x_{k-1},1,x_{k+1},\ldots,x_n) =$$

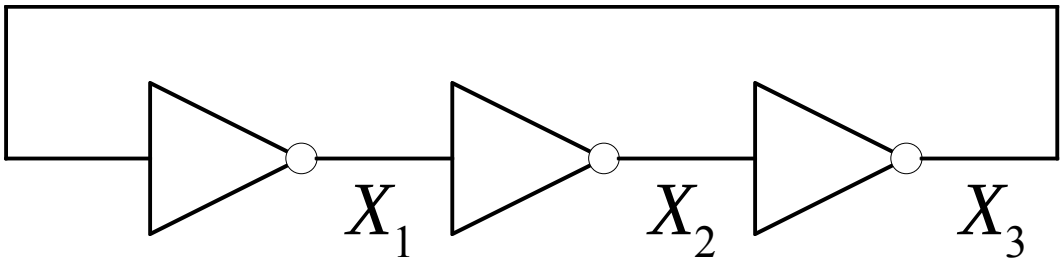
$$= F(x_1,\ldots,x_{k-1},1^*,x_{k+1},\ldots,x_n);$$

$$F(x_1,\ldots,x_{k-1},0,x_{k+1},\ldots,x_n) =$$

$$= F(x_1,\ldots,x_{k-1},0^*,x_{k+1},\ldots,x_n).$$

Диаграмма Маллера

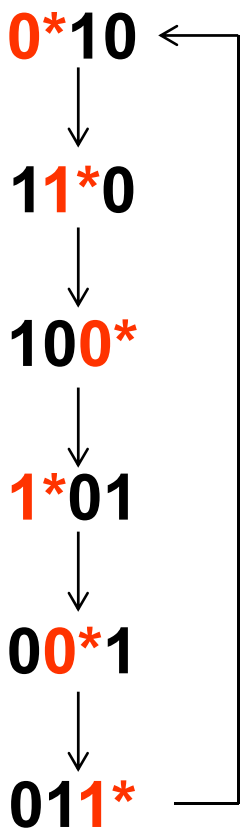
(Свойство полумодулярности)

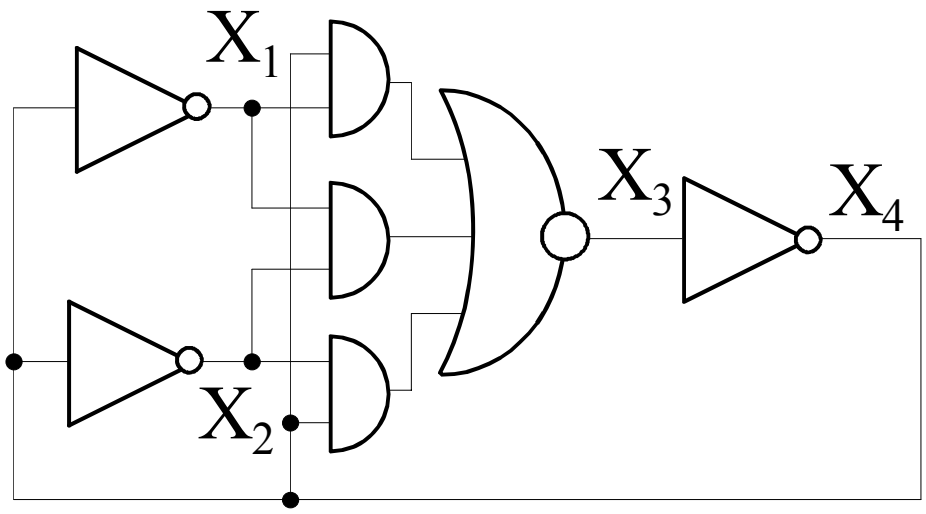


$$X_2 = \overline{X}_1;$$

$$X_3 = \overline{X}_2;$$

$$X_1 = \overline{X}_3$$



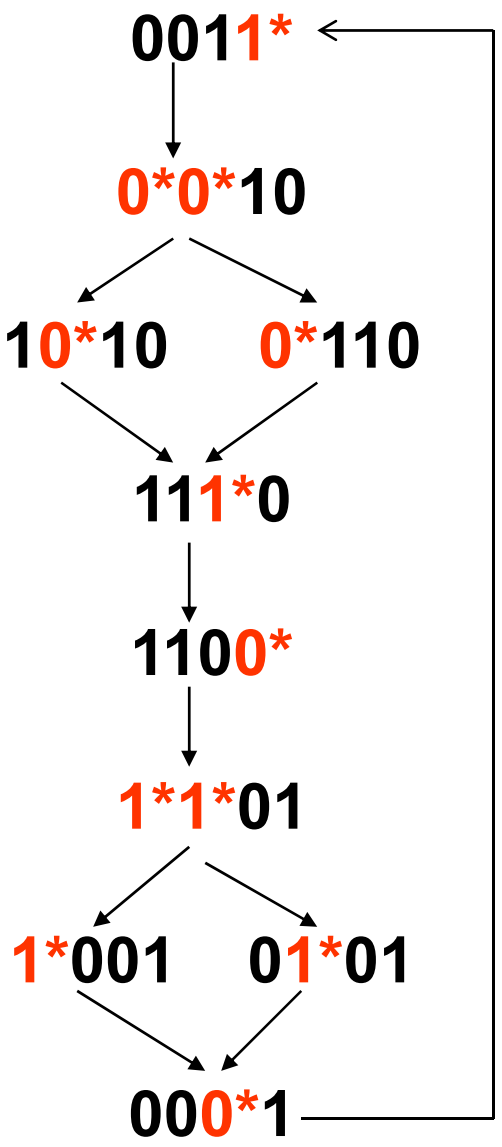


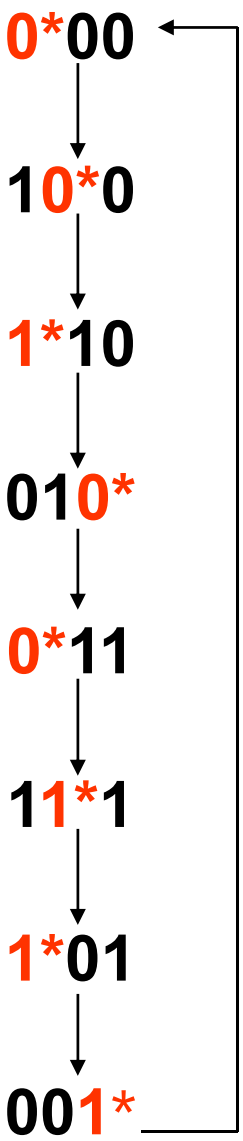
$$X_3 = \overline{X_1 X_2 + X_4 (X_1 + X_2)};$$

$$X_1 = \overline{X_4};$$

$$X_2 = \overline{X_4};$$

$$X_4 = \overline{X_3}.$$





X_1	X_2	X_3	F_1	F_2	F_3
0*	0	0	1	0	0
1	0*	0	1	1	0
0	1	0*	0	1	1
1*	1	0	0	1	0
0	0	1*	0	0	0
1*	0	1	0	0	1
0*	1	1	1	1	1
1	1*	1	1	0	1

$$X_1 = \overline{X_2 \oplus X_3};$$

$$X_2 = \overline{X}_1 X_2 \vee X_1 \overline{X}_3;$$

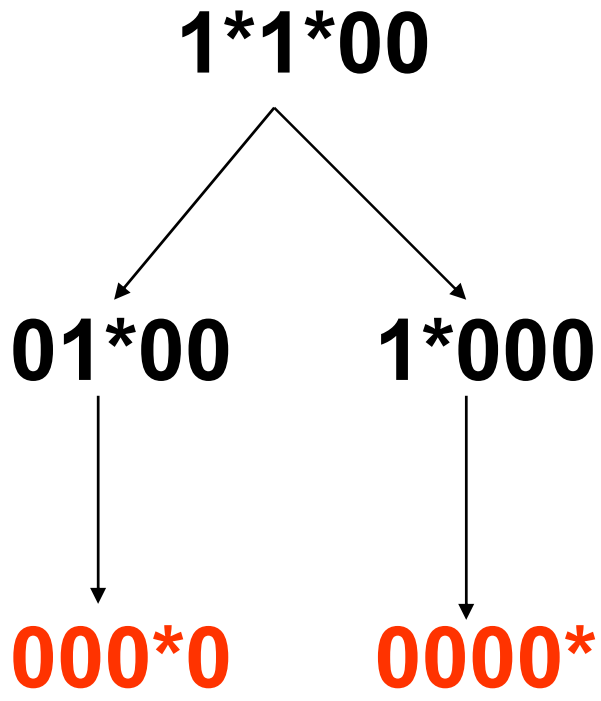
$$X_3 = X_1 X_2 \vee X_1 \overline{X}_3.$$

Корректные диаграмм Маллера

1. Конфликтные состояния

$$x_1,\ldots,x_i^*,\ldots,x_j,\ldots,x_n$$

$$x_1,\ldots,x_i,\ldots,x_j^*,\ldots,x_n$$



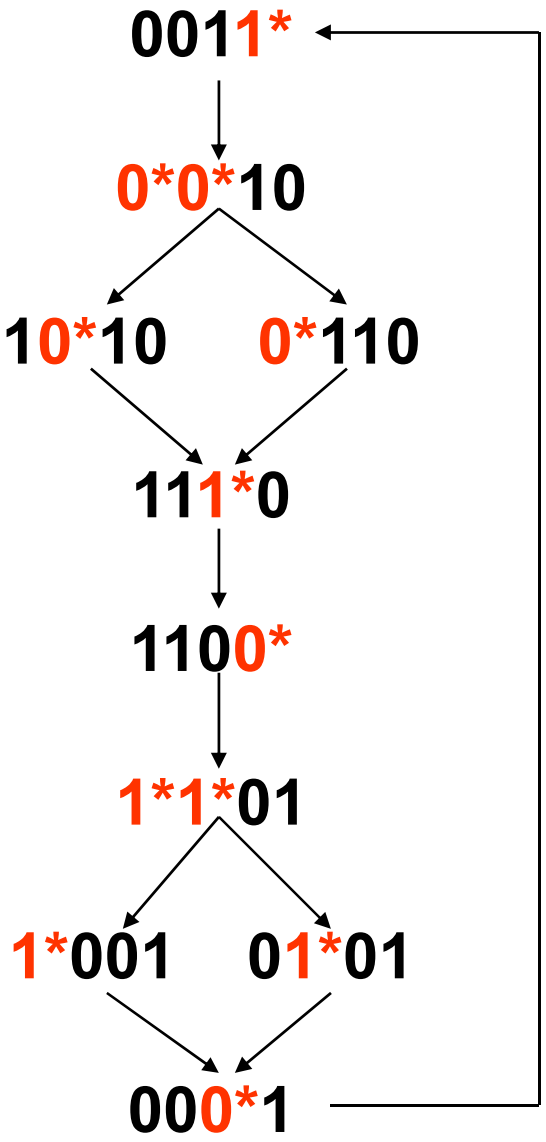
$0 \longrightarrow 0$
$0^* \longrightarrow 1$
$1 \longrightarrow 1$
$1^* \longrightarrow 0$

Правильные переключения значений переменной

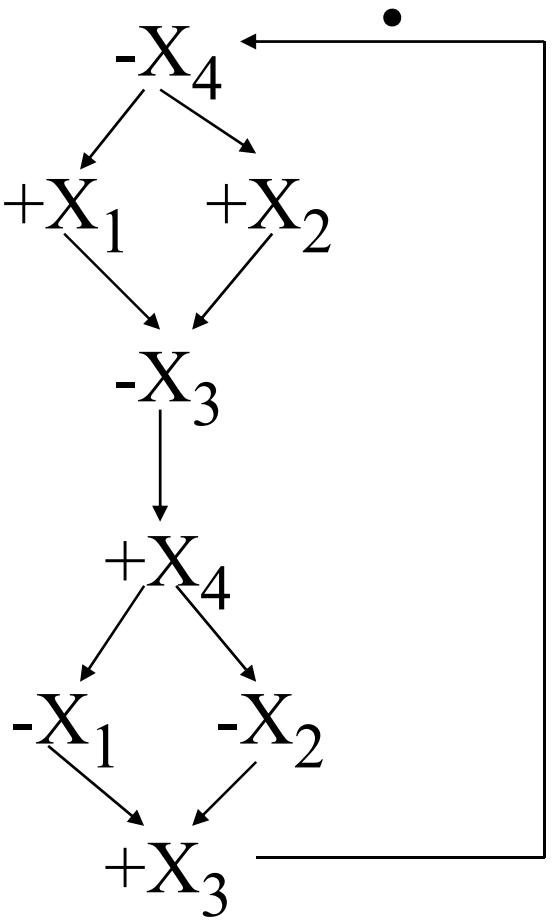
$0^* \longrightarrow 0$
$1^* \longrightarrow 1$

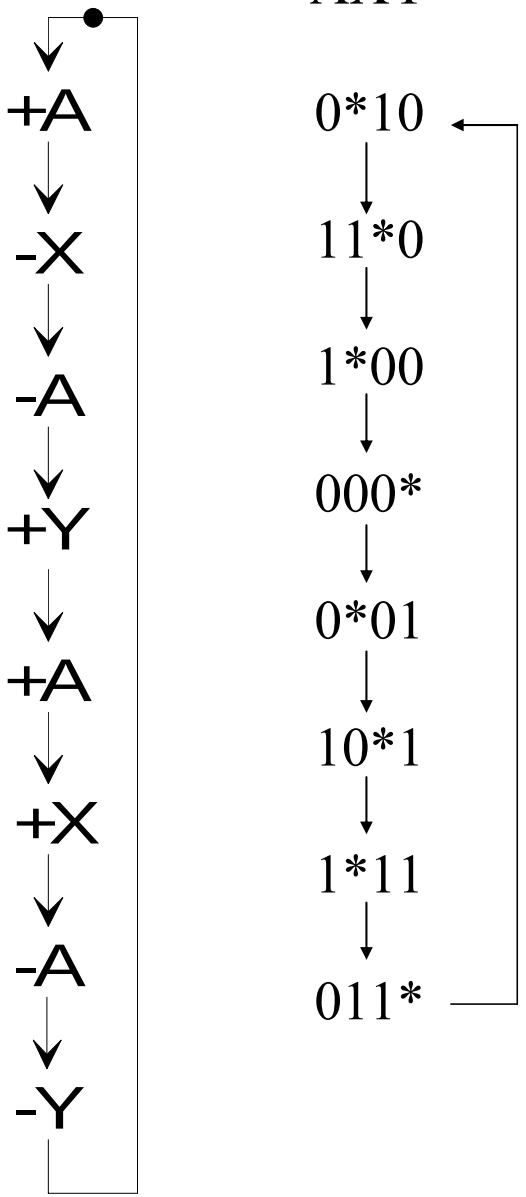
Неправильные переключения Значений переменной

Сигнальный граф (STG – Signal Transition Graph)



События - $+x_j$ and $-x_j$





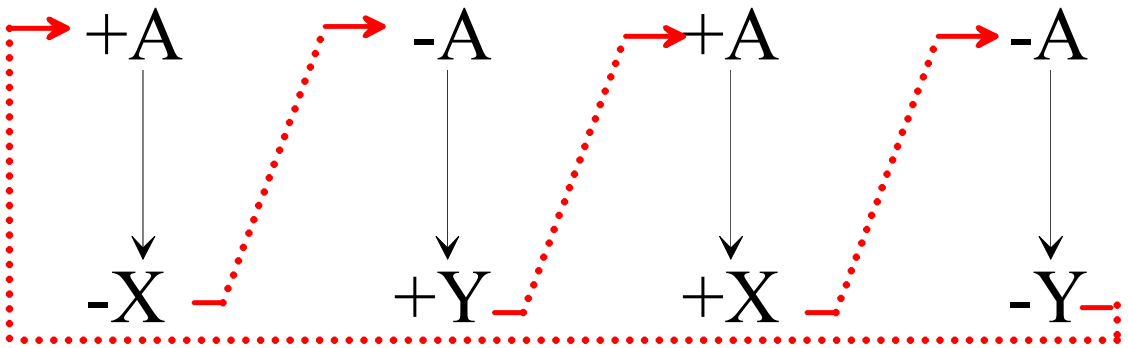
A	X	Y	F_1	F_2	F_3
0	0	**0***	0	0	**1**
1*	0	0	**0**	0	0
0*	1	0	**1**	1	0
1	**1***	0	1	**0**	0
0*	0	1	**1**	0	1
1	**0***	1	1	**1**	1
0	1	**1***	0	1	**0**
1*	1	1	**0**	1	1

A	X	Y	F_1	F_2	F_3
0	0	**0***	0	0	**1**
1*	0	0	**0**	0	0
0*	1	0	**1**	1	0
1	**1***	0	1	**0**	0
0*	0	1	**1**	0	1
1	**0***	1	1	**1**	1
0	1	**1***	0	1	**0**
1*	1	1	**0**	1	1

$$A = X \oplus Y;$$

$$X = \overline{A}X \vee AY;$$

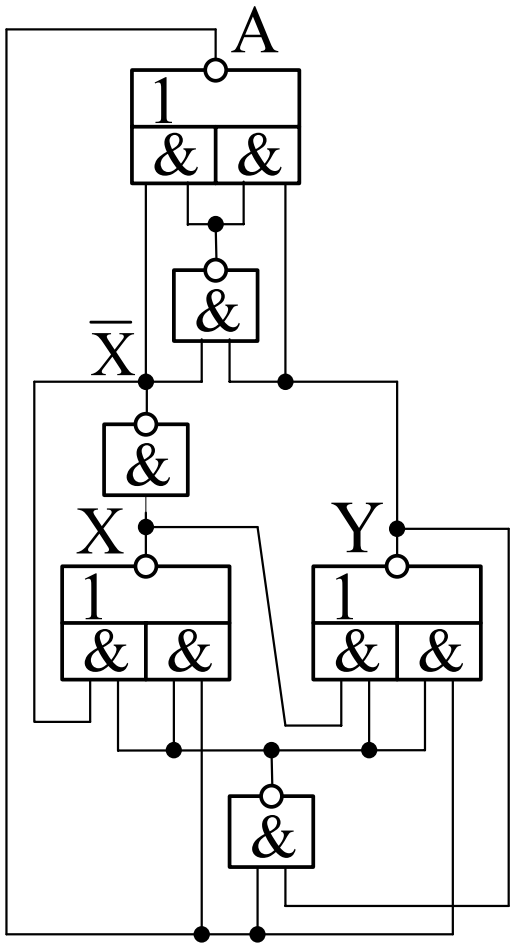
$$Y = \overline{A}\overline{X} \vee AY.$$



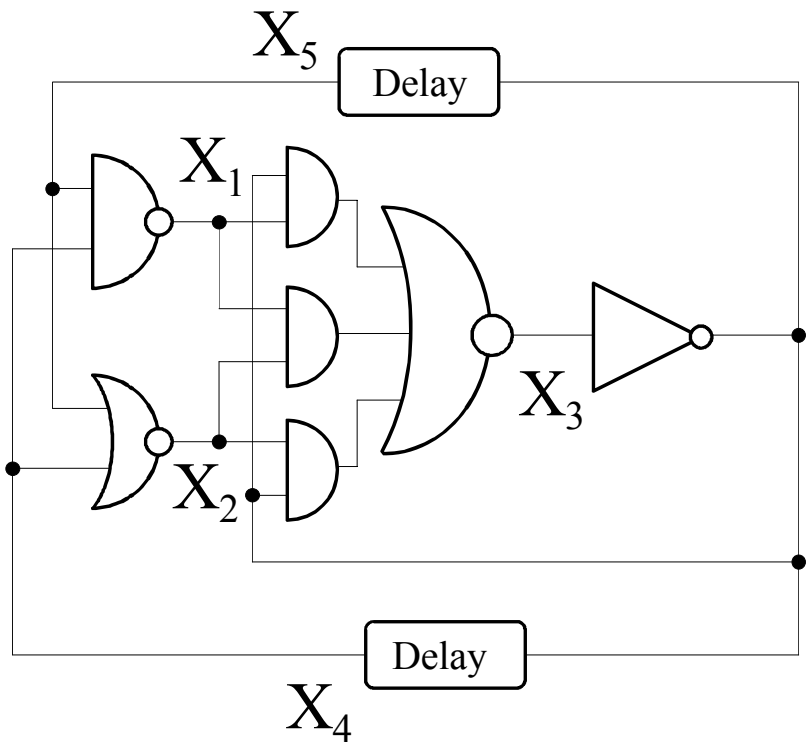
$$A = X \oplus Y = \overline{\overline{X\overline{Y}} + \overline{\overline{X}Y}} = \overline{(\overline{X} + Y)\overline{(\overline{X}Y)}};$$

$$X = \overline{\overline{AX} + AY} = \overline{(AY)\overline{(A + \overline{X})}};$$

$$Y = \overline{\overline{A}\overline{X} + AY} = \overline{(AY)\overline{(A + X)}}.$$



Дистрибутивные и недистрибутивные диаграммы Маллера



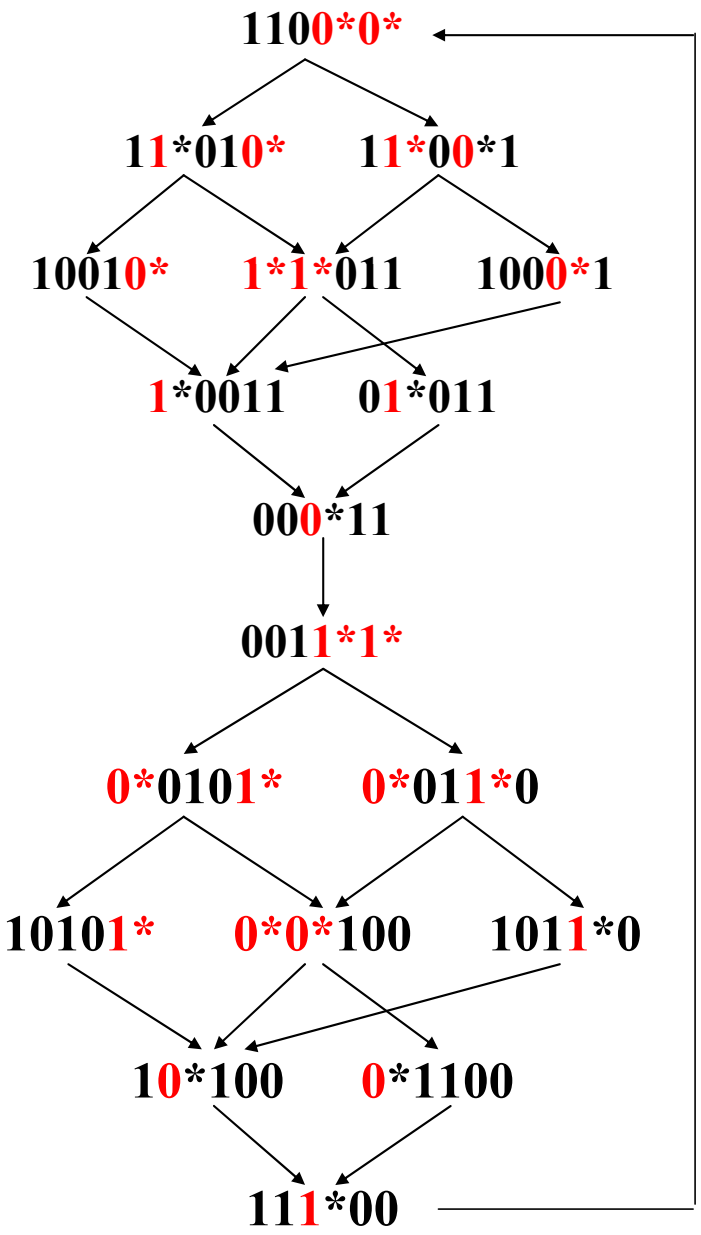
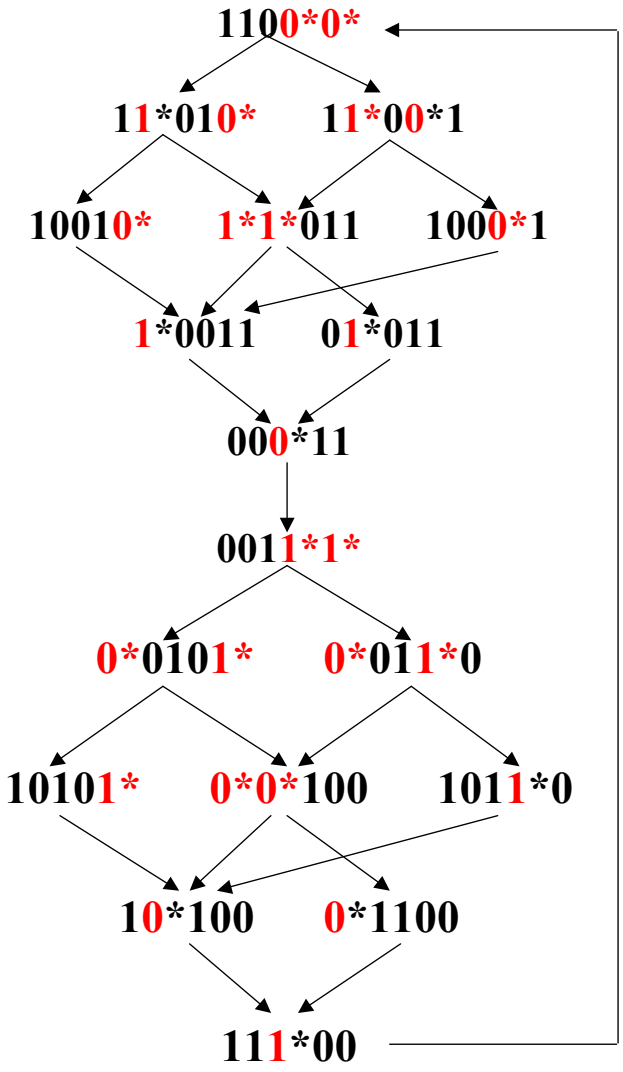
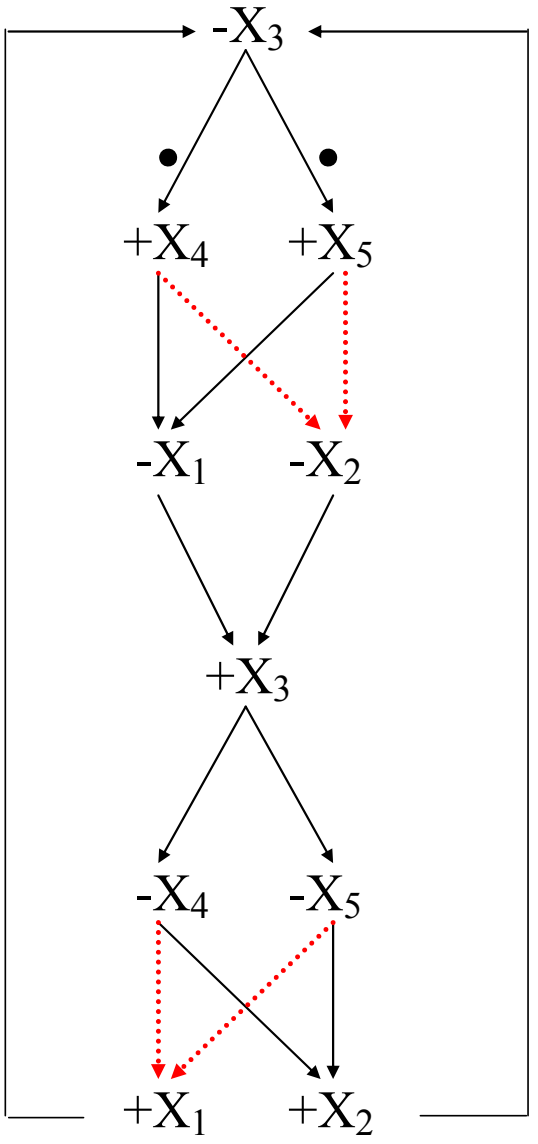


Диаграмма изменений





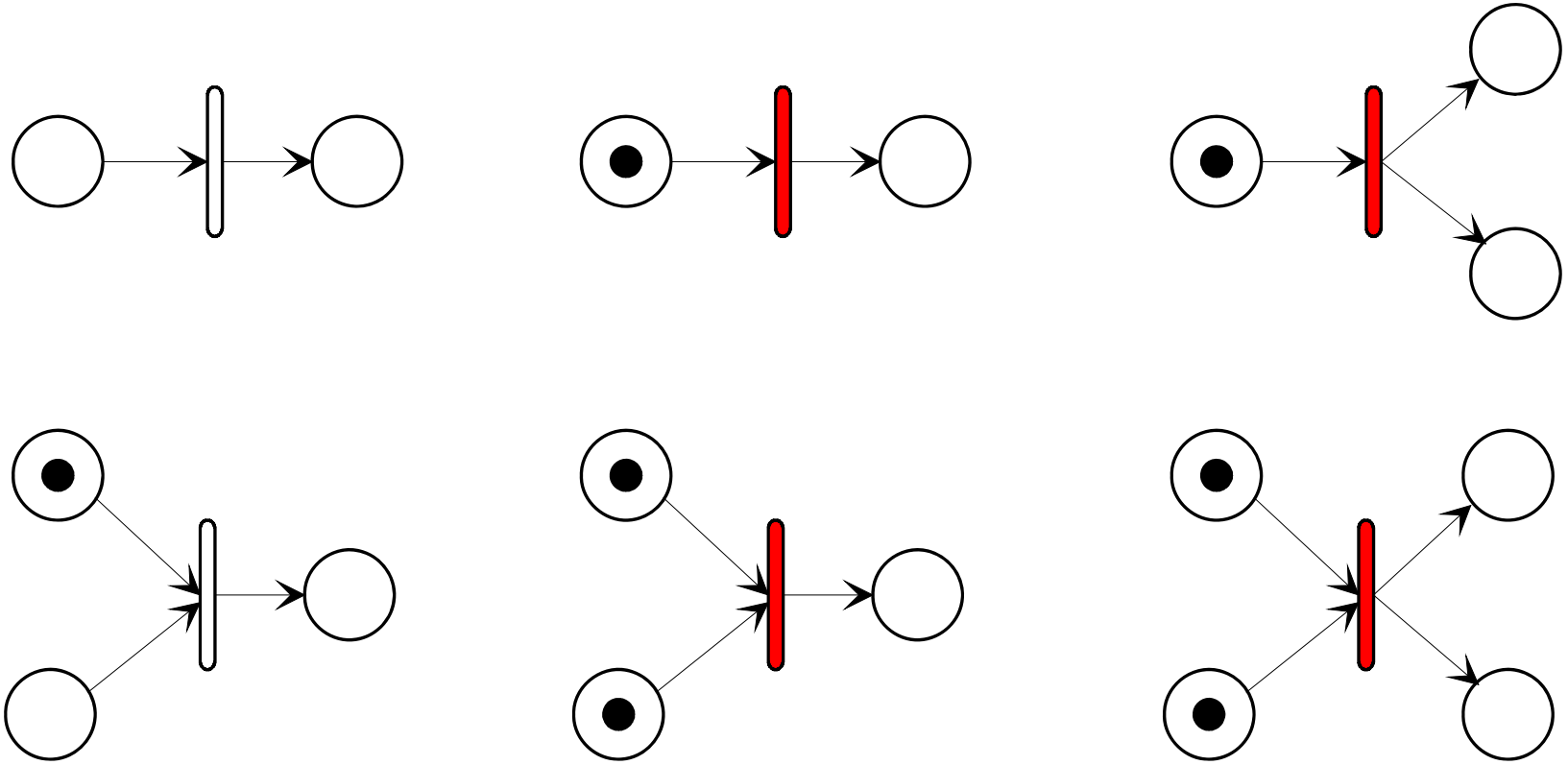


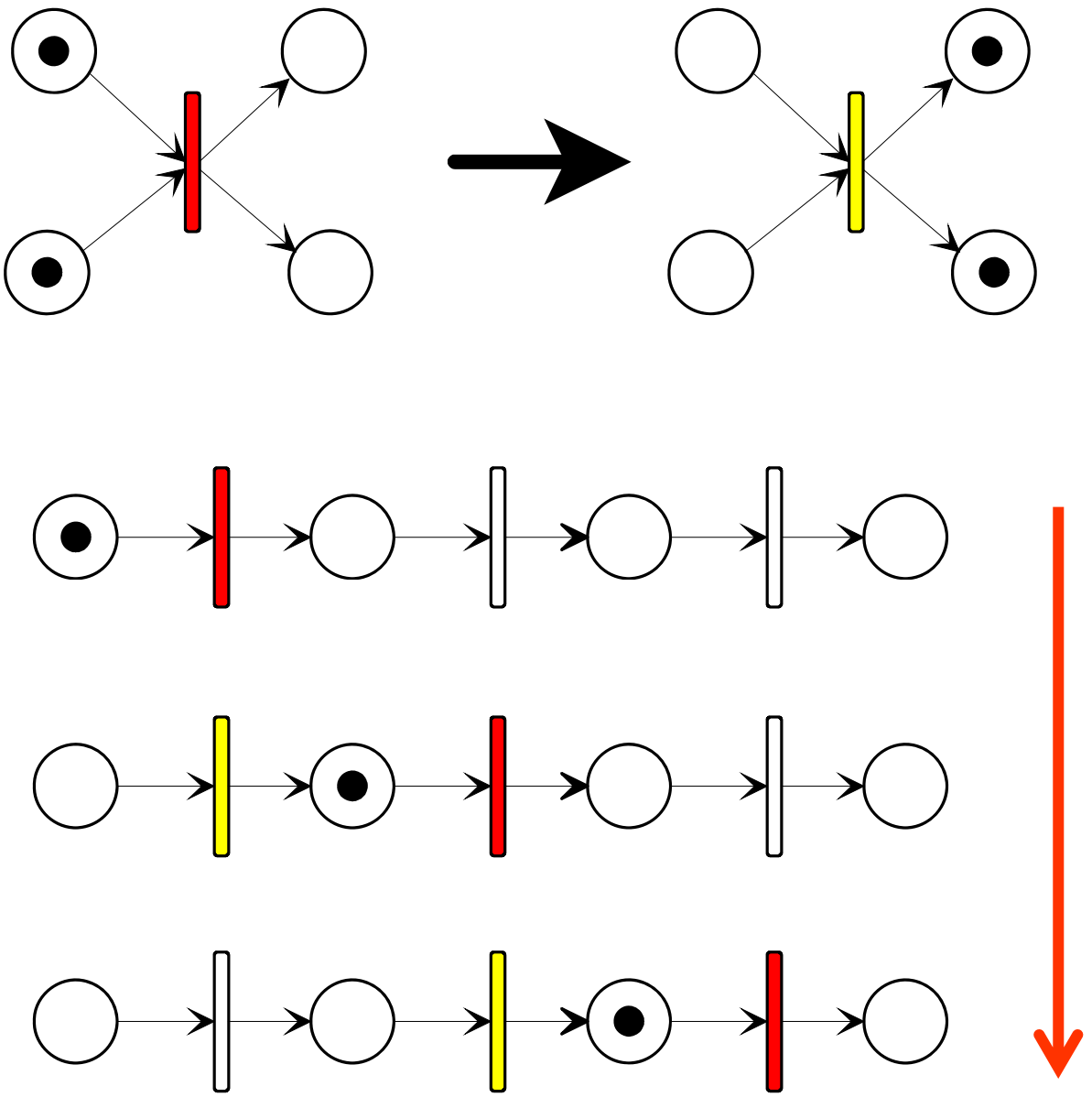
Логическое Проектирование

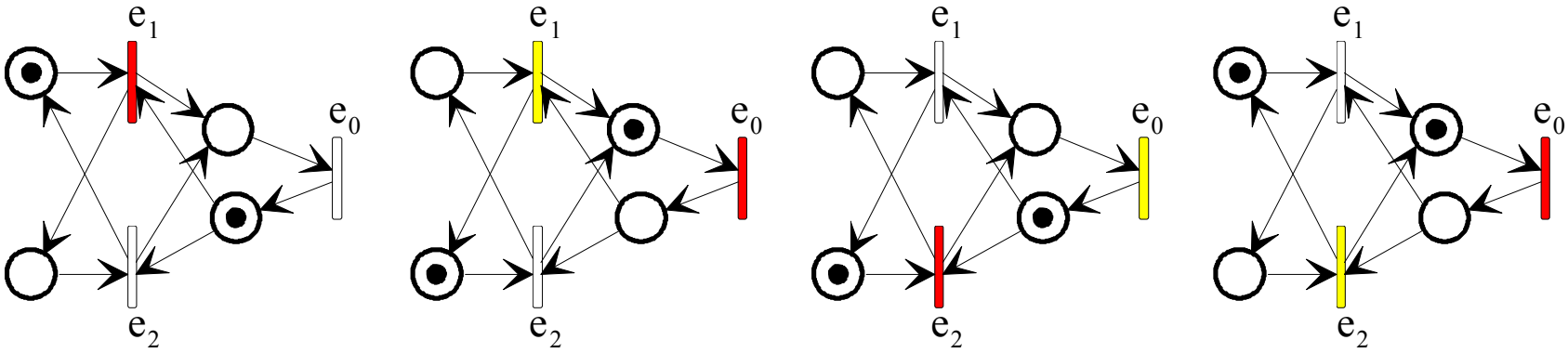
Лекция 14

Управление по событиям

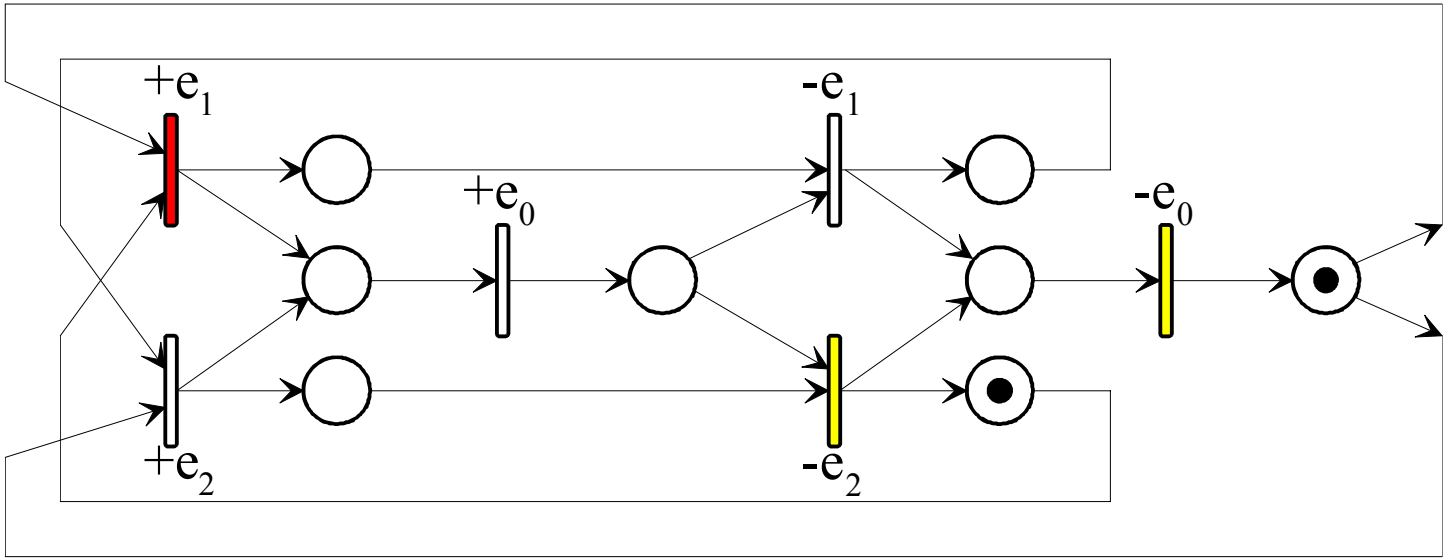
Сети Петри

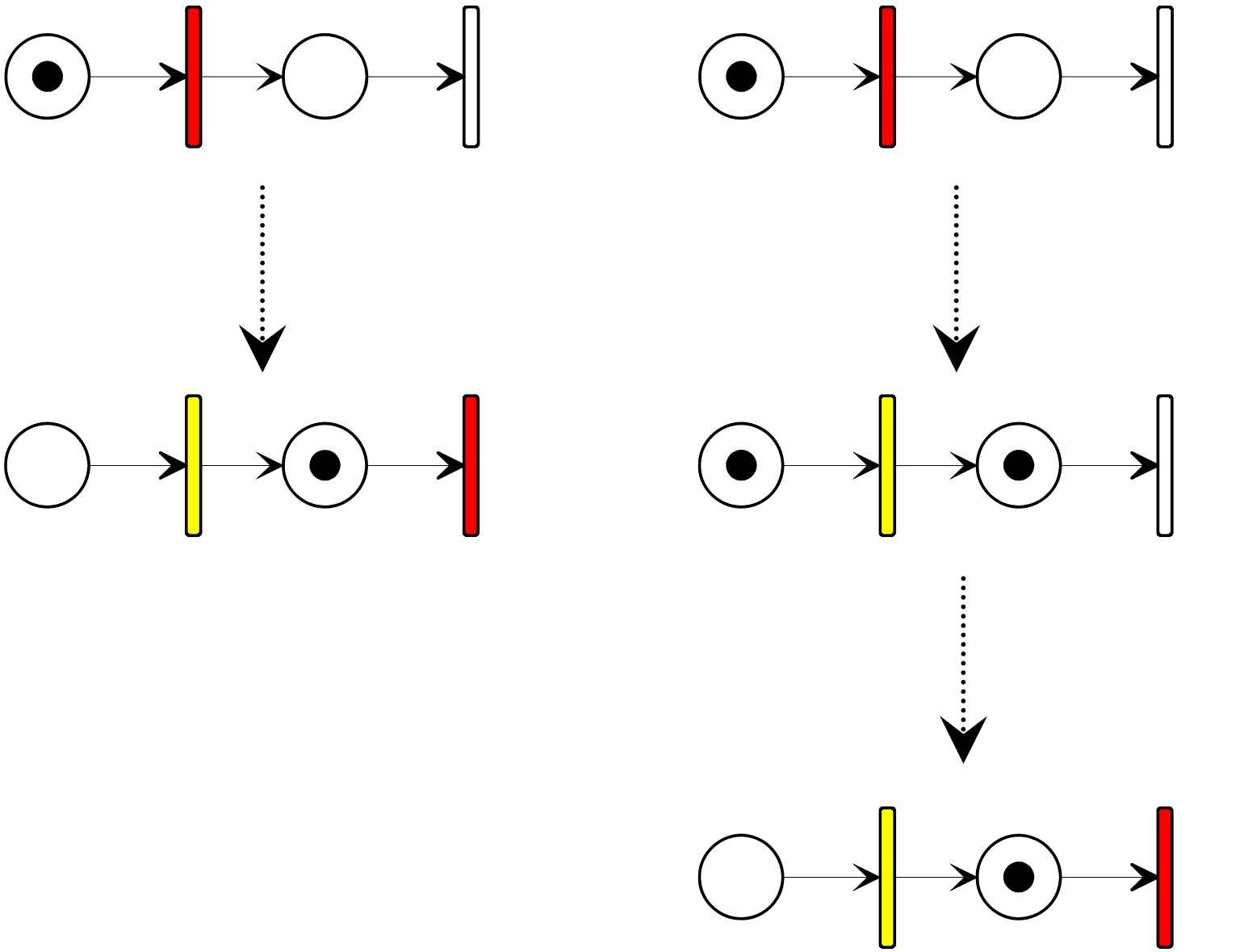


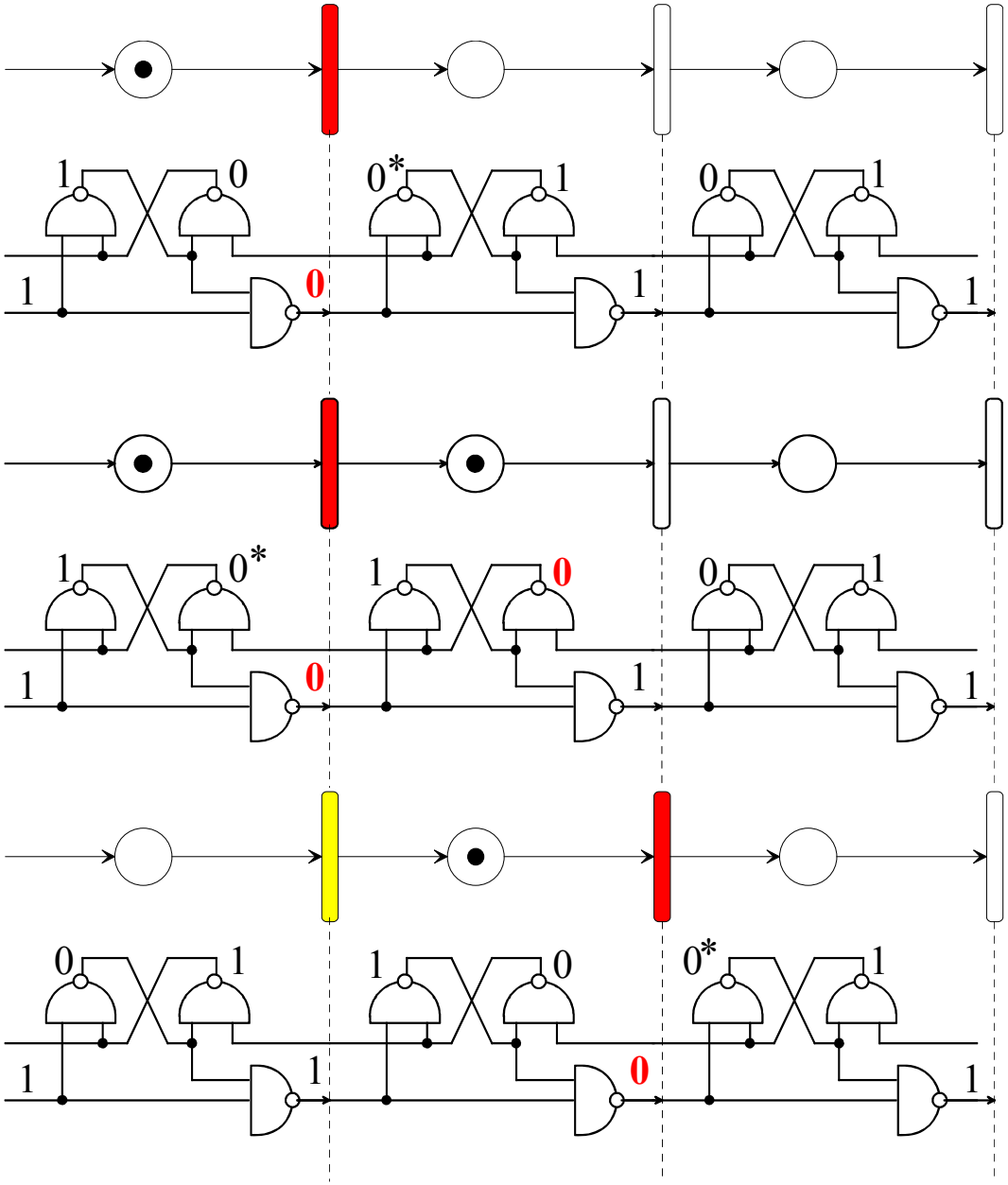




$$e_1 \rightarrow e_0 \rightarrow e_2 \rightarrow e_0 \rightarrow e_1 \rightarrow \ldots$$

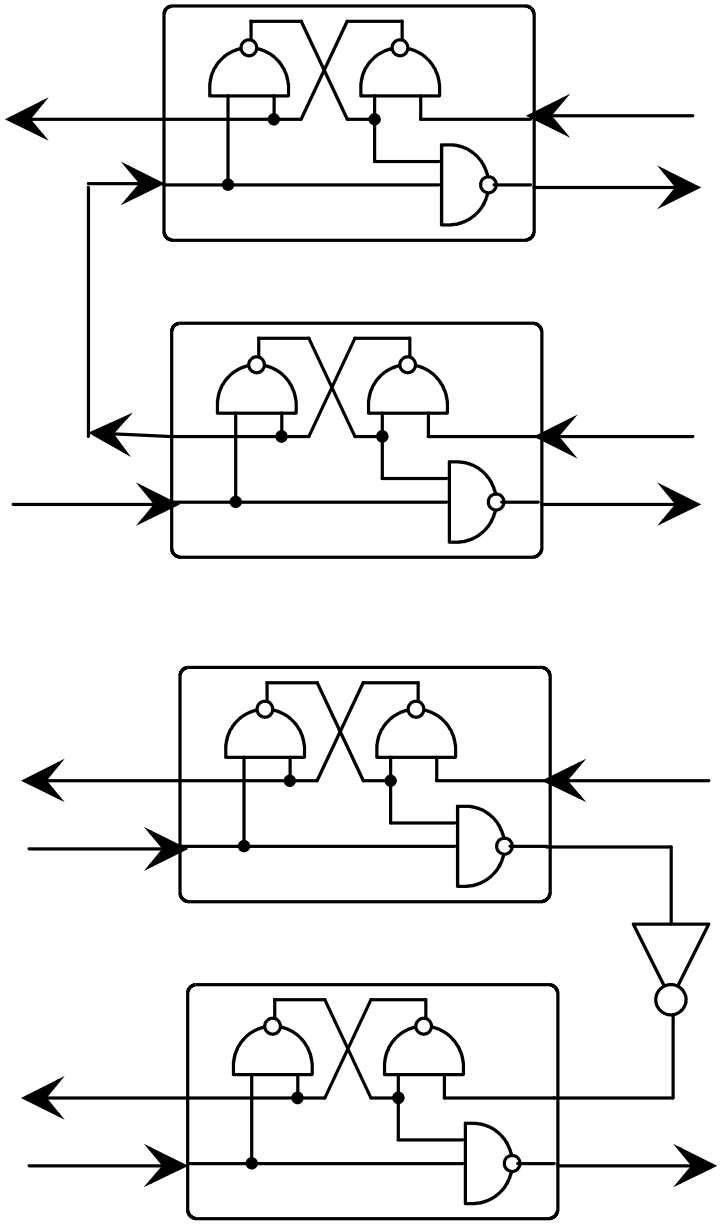


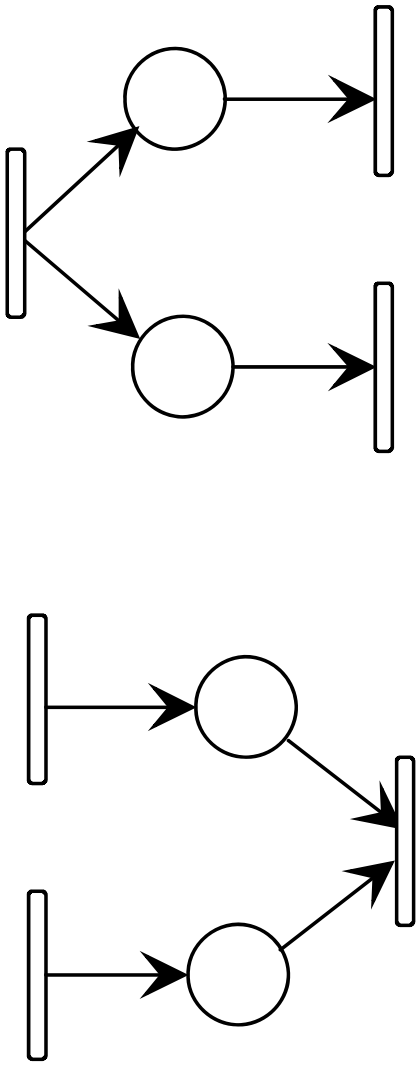


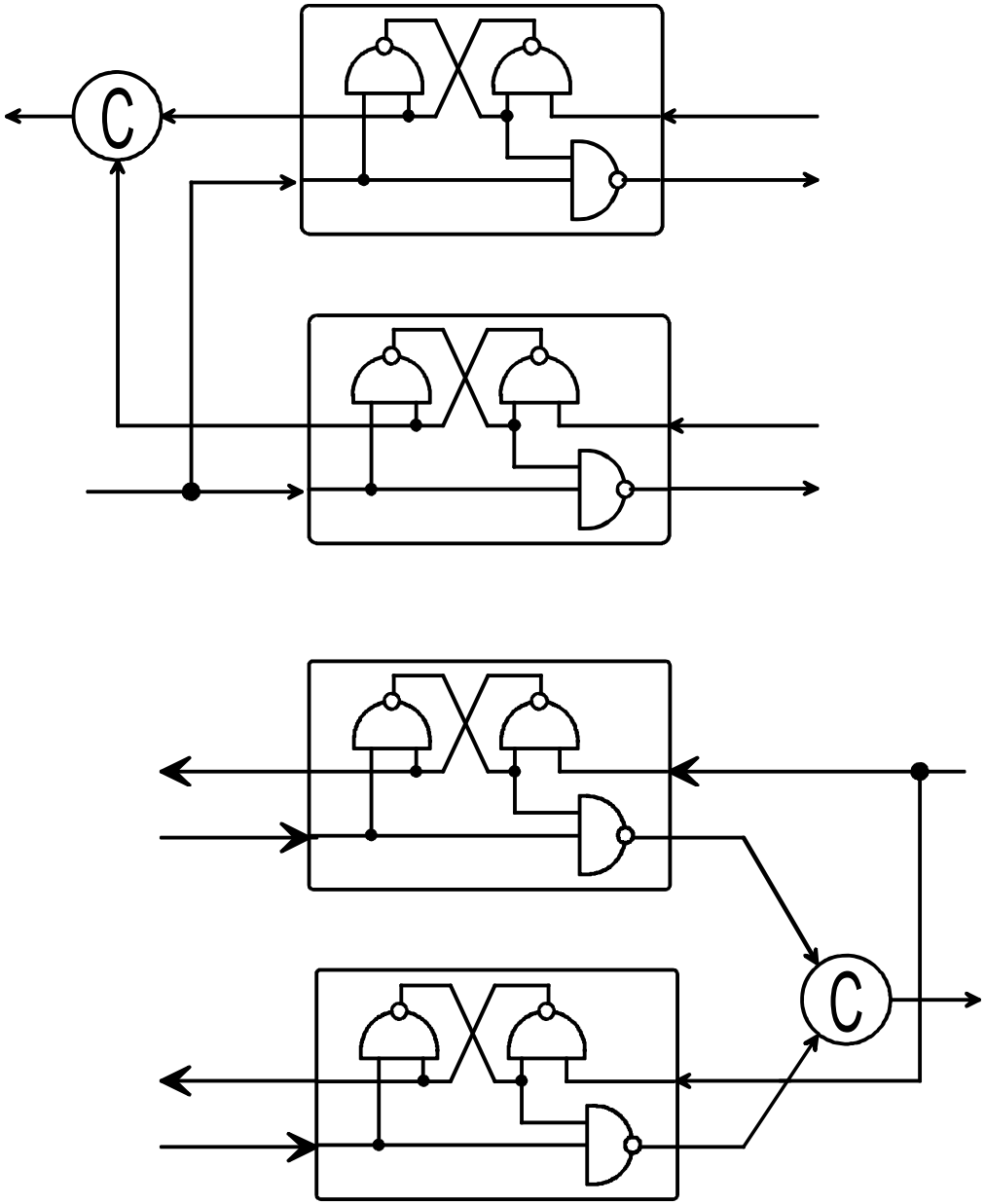

1
0
0*
1
0
1
1
0
1
1
1
0*
1
0
0
1
1
1
0
1
0
1
1
0
0*
1
1
1
0
1

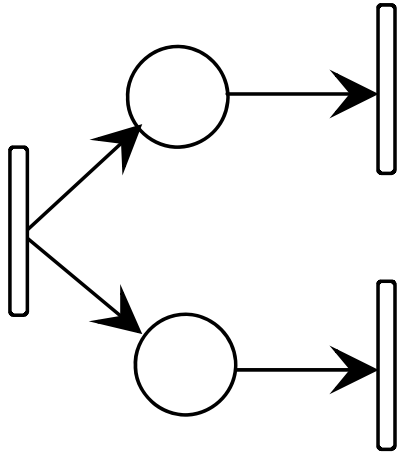
События:

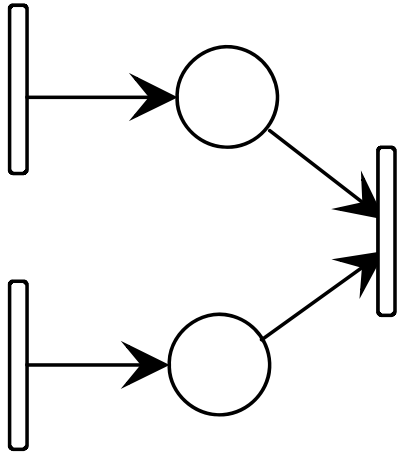
$0 \rightarrow 1$	$+ x$
$1 \rightarrow 0$	$- x$
$1 \rightarrow 0 \rightarrow 1 \quad (0 \rightarrow 1 \rightarrow 0)$	X





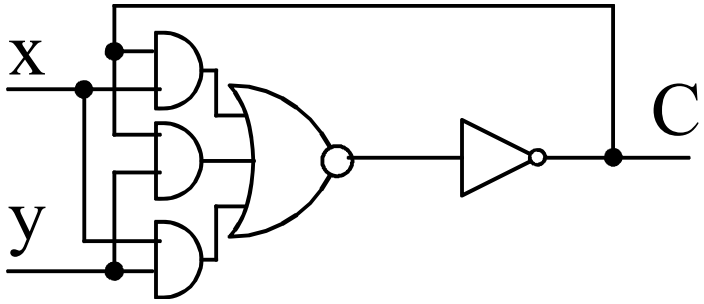

C
C

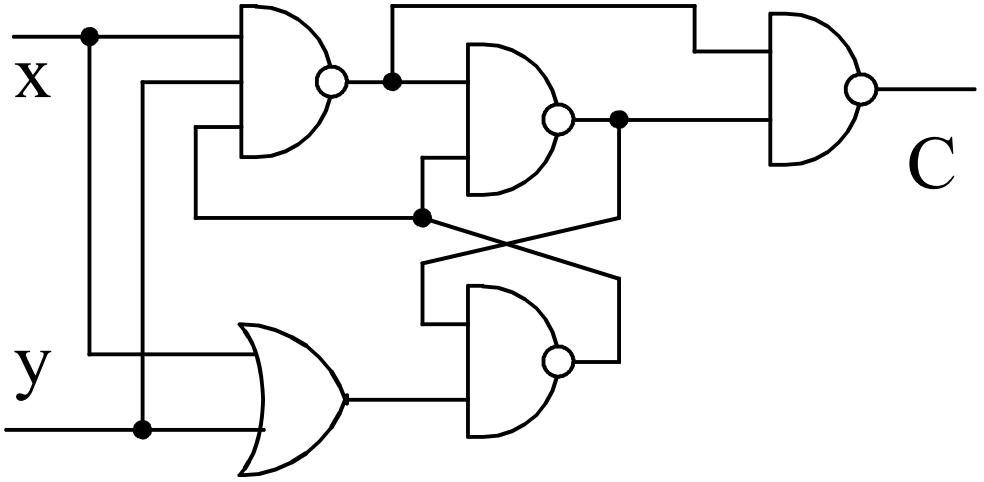


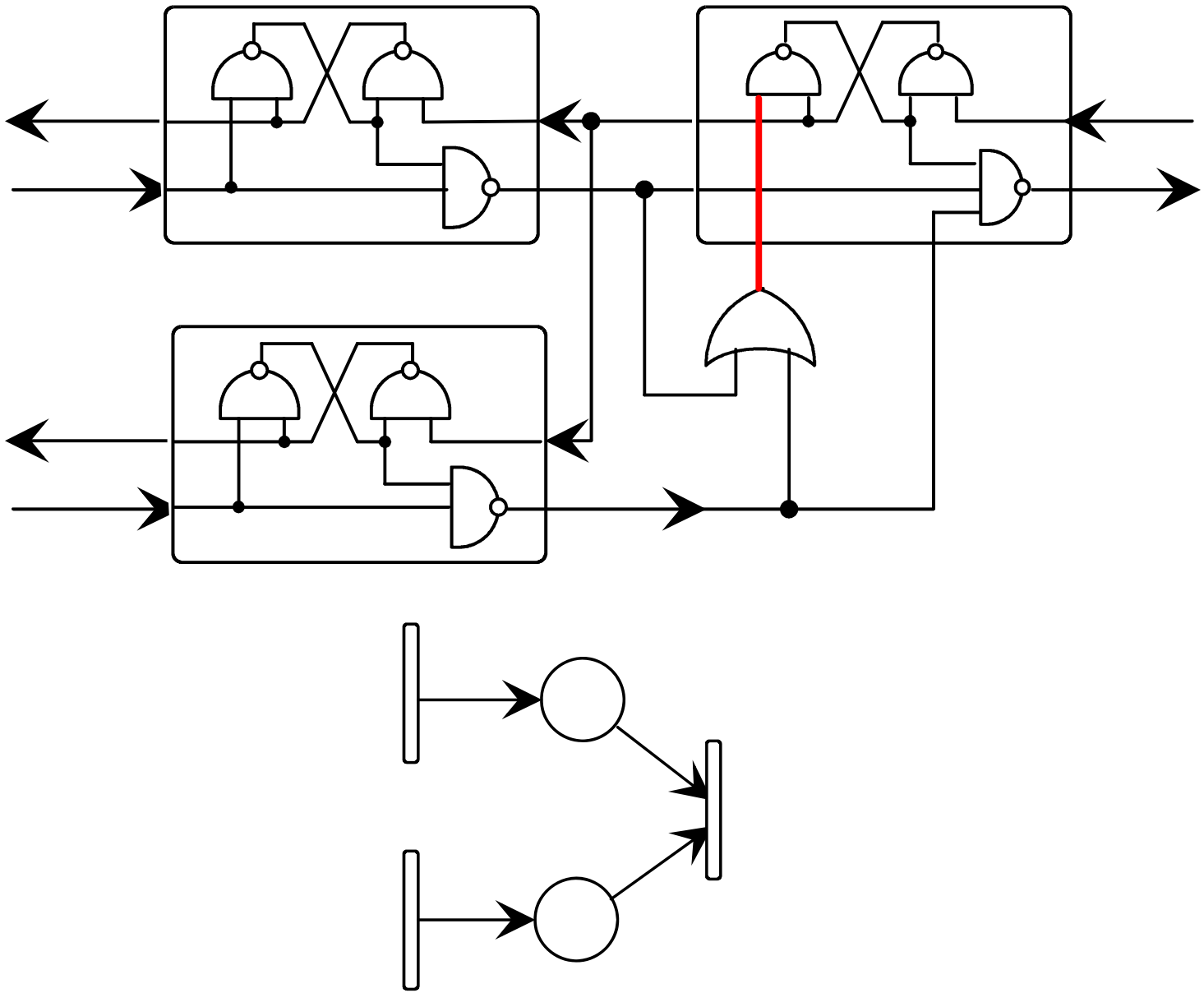


C-элемент

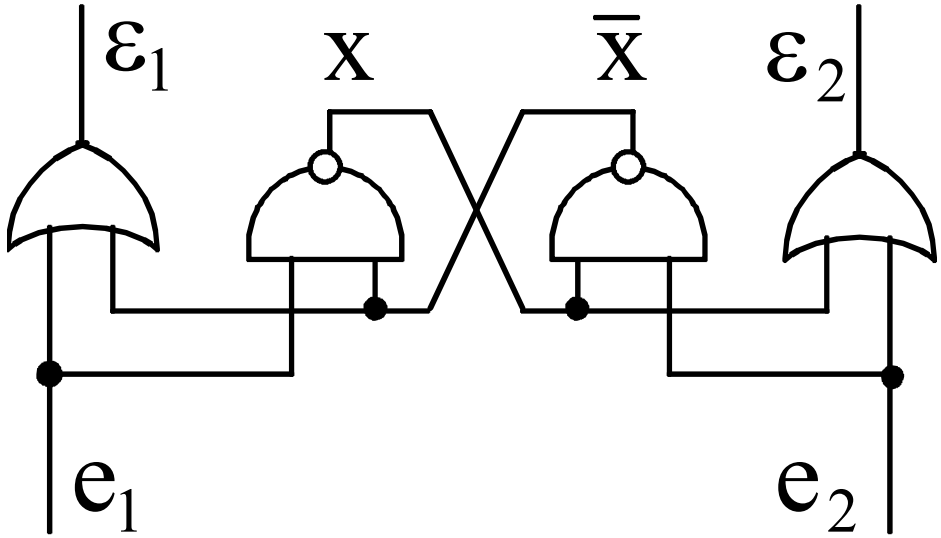
$$C = \bigcap_j x_j + C(\bigcup_j x_j);$$







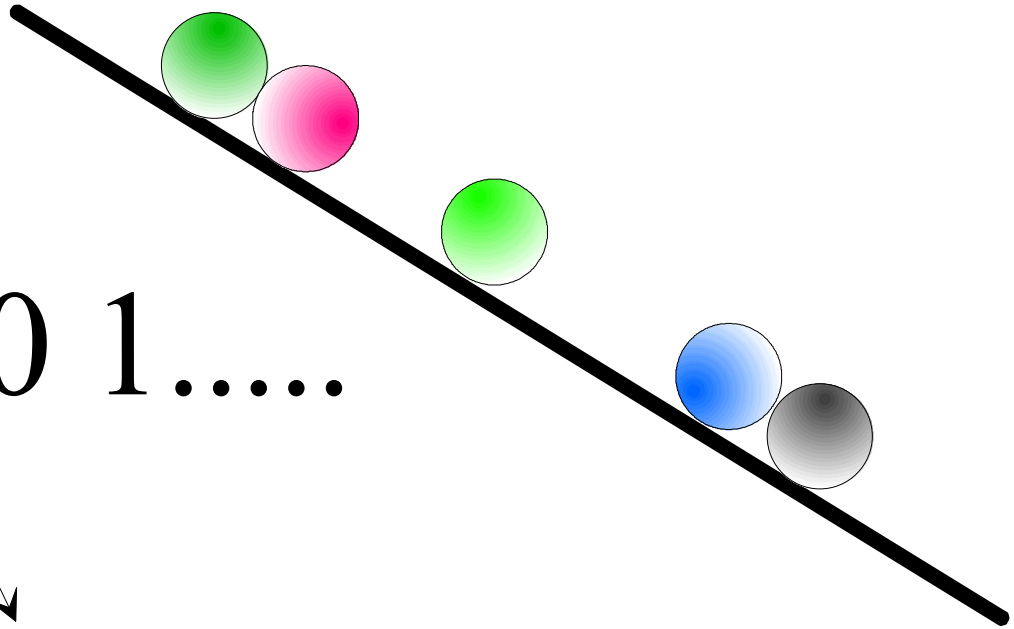
Переключение переменных



$$-e_1 \rightarrow +x \rightarrow -\bar{x} \rightarrow -\varepsilon_1 \rightarrow +e_1 \rightarrow +\varepsilon_1$$

$$-e_2 \rightarrow +\bar{x} \rightarrow -x \rightarrow -\varepsilon_2 \rightarrow +e_2 \rightarrow +\varepsilon_2$$

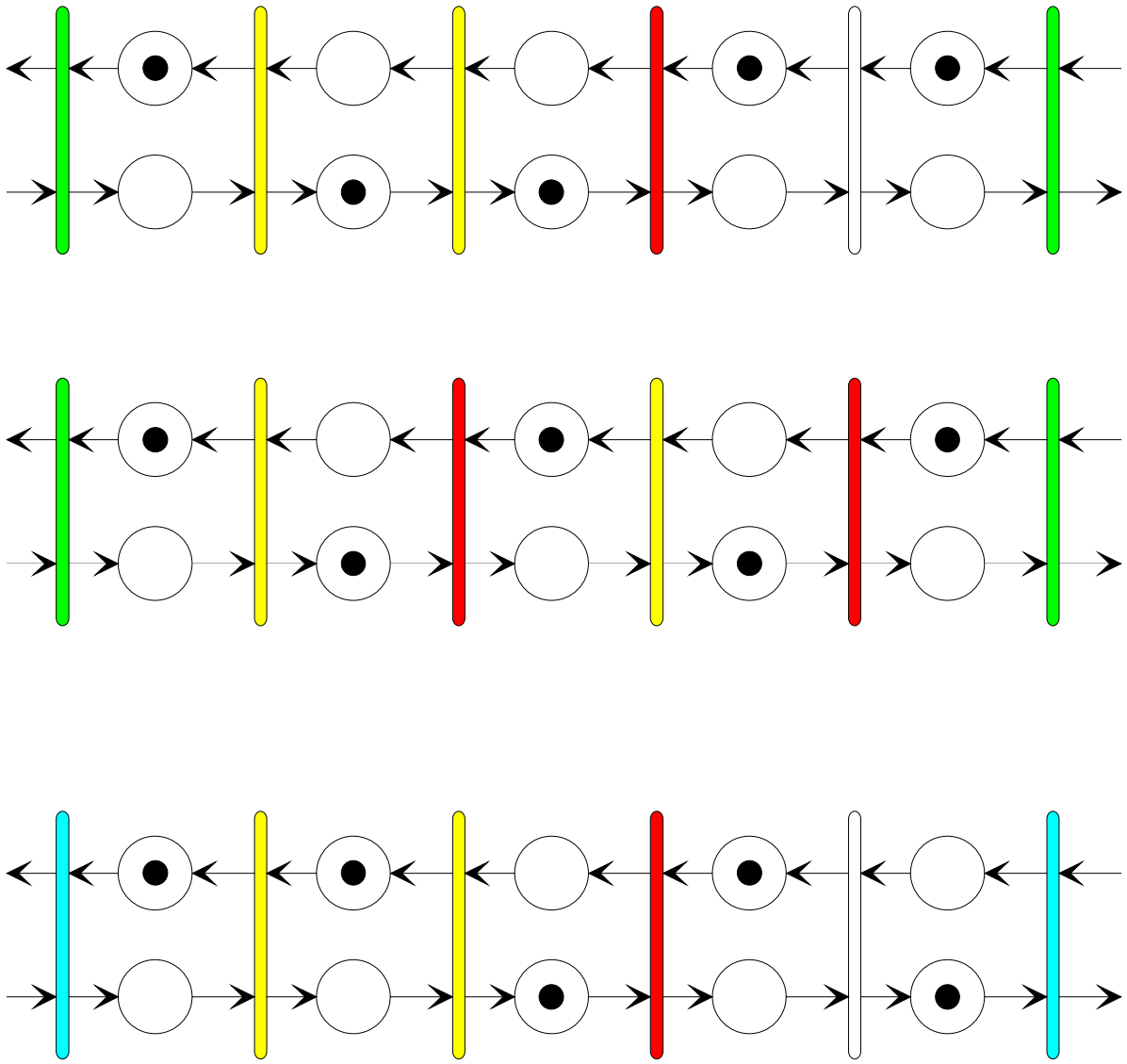
Асинхронный конвейер

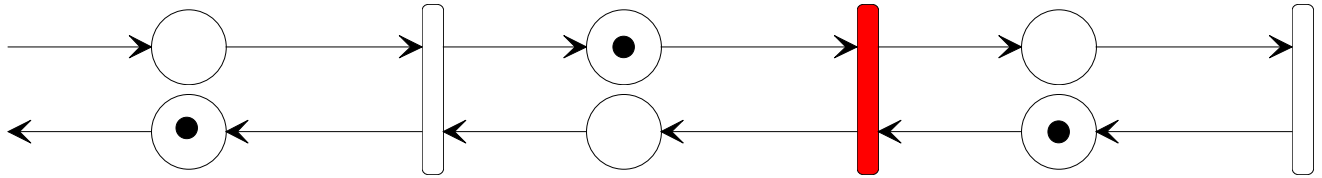


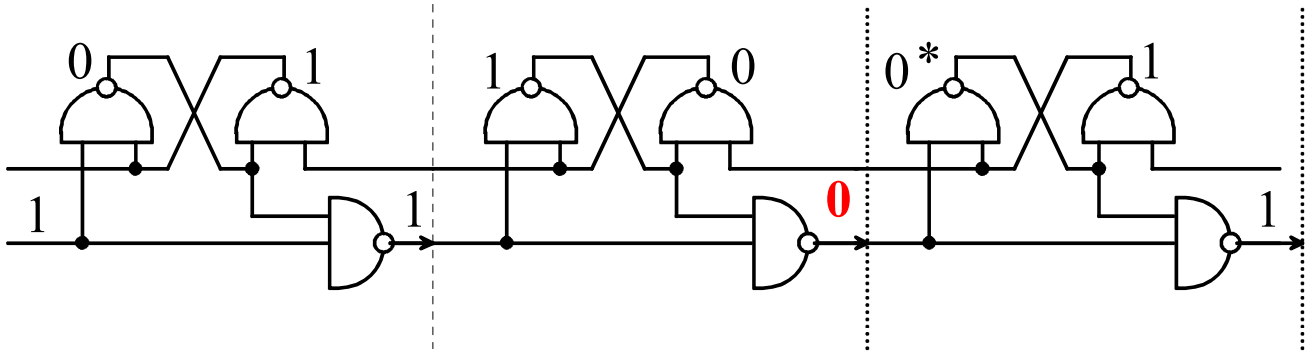
1 0 0 1 1 0 1.....

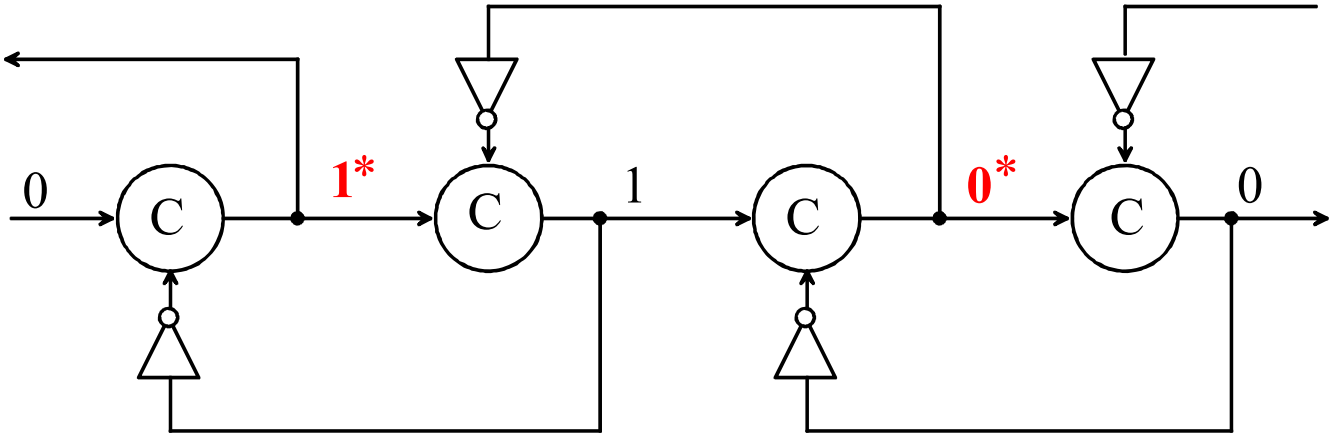
0 1 0 1 0 1 1.....

0 0 1 0 1 1 0.....

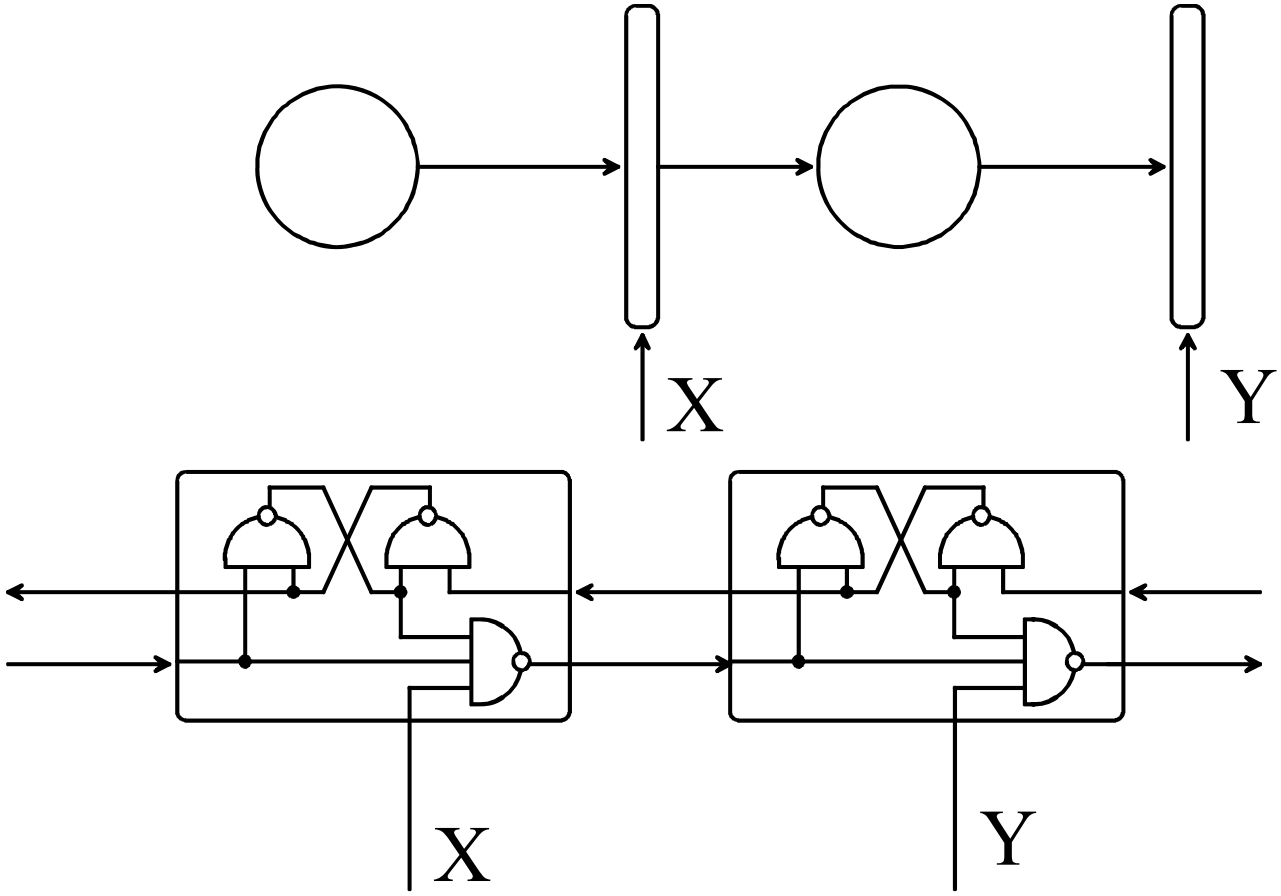





0
1
1
1
1
0
0*
1
1
0
1


0
C
1*
C
1
C
0*
C
0

Внешние переменные



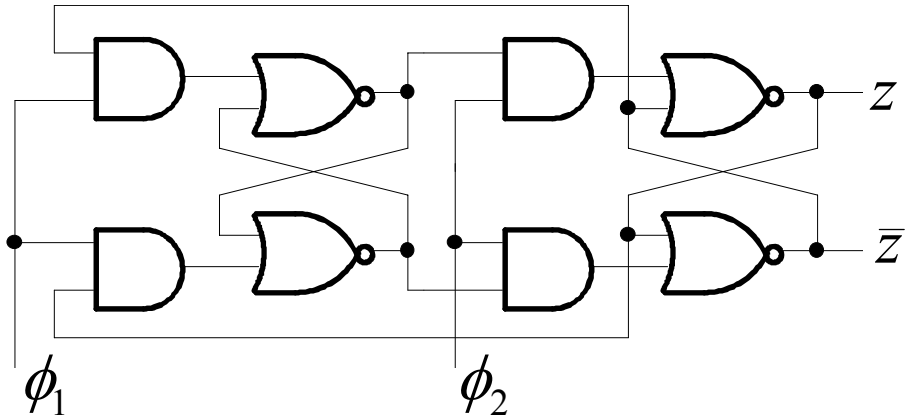
Счетные устройства (примеры схем)

- Счетчик – это устройство, состояние которого соответствует числу циклов изменения входного сигнала.
- Счетчик по модулю N – это устройство, состояние которого соответствует остатку от деления числа циклов входного сигнала на N.

Обычно счетчики строятся из счетных триггеров (Т-триггеров). Счетный триггер – это простейший счетчик по модулю 2.

Счетные триггеры

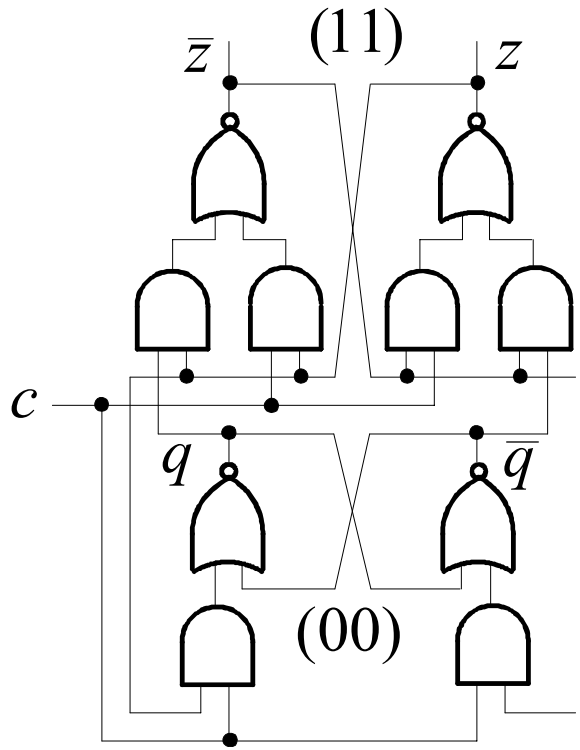
Двухтриггерная реализация с униполярным управлением



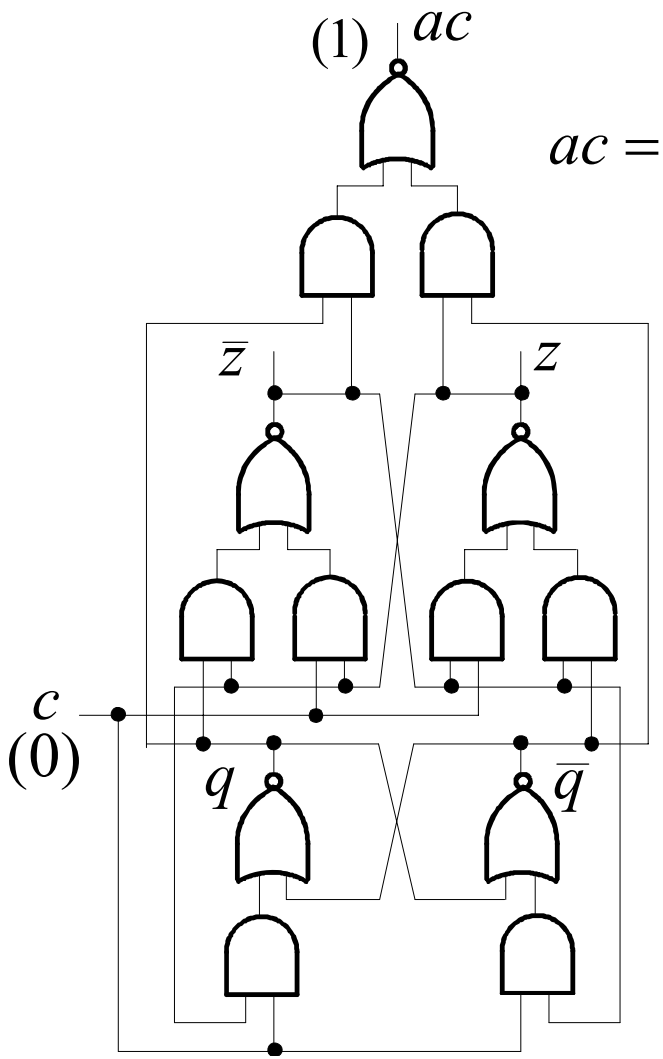
Этот триггер считает по модулю 2 пары счетных сигналов ϕ_1 и ϕ_2. Он бесполезен, если необходимо считать изменения только одного входного сигнала.

Счетные устройства (примеры схем)

Master-slave реализация счетного триггера с разнополярным управлением

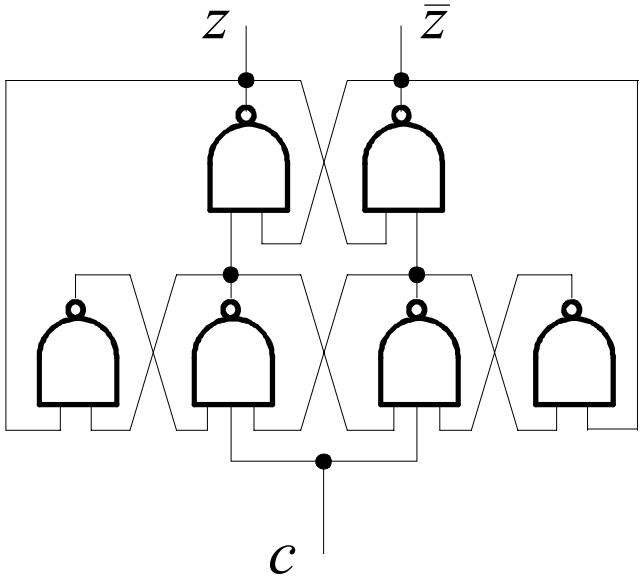


Самосинхронная реализация того же триггера

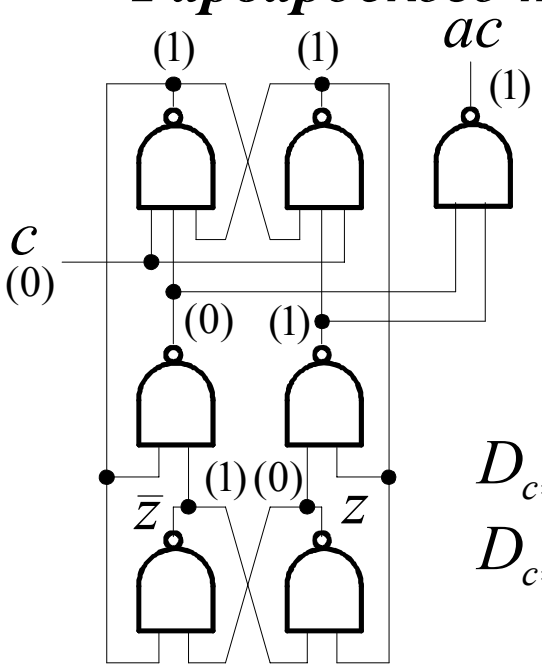


Счетные устройства (примеры схем)

Гарвардский счетный триггер (трехтриггерная схема)



Самосинхронная реализация Гарвардского триггера



$$D_{c=1} = 4\tau_g + \tau_{ind}$$

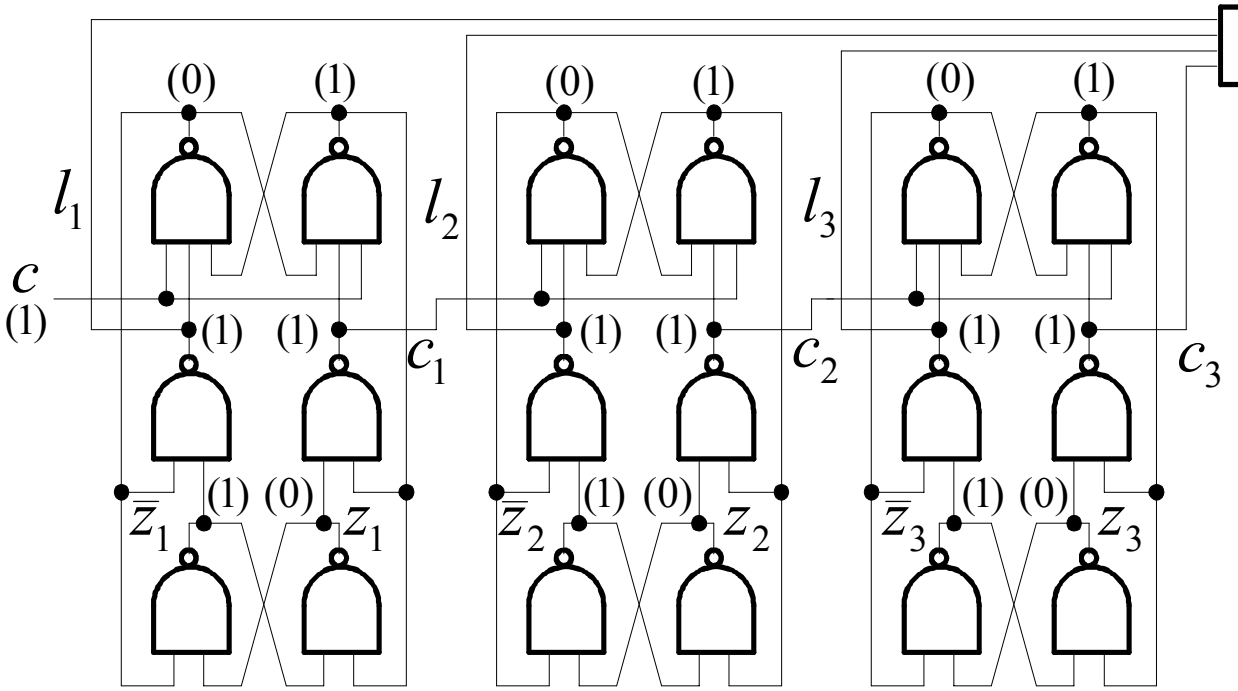
$$D_{c=0} = 2\tau_g + \tau_{ind}$$

$$\begin{array}{ccccccccccccccccccc} 11 & & 11' & & 10 & & 10 & & 10 & & 10 & & 10 & & 10' & & 11 & & 11 \\ 0'01 & \rightarrow & 101 & \rightarrow & 101 & \rightarrow & 101 & \rightarrow & 10'1 & \rightarrow & 111 & \rightarrow & 1'11 & \rightarrow & 011 & \rightarrow & 011' & \rightarrow & 010 & \rightarrow \\ 10 & & 10 & & 10' & & 1'1 & & 01 & & 01 & & 01 & & 01 & & 01 & & 01 & \end{array}$$

$$\begin{array}{ccccccccccccccccccc} 11 & & 1'1 & & 01 & & 01 & & 01 & & 01 & & 01 & & 0'1 & & 11 & & 11 \\ 0'10 & \rightarrow & 110 & \rightarrow & 110 & \rightarrow & 110 & \rightarrow & 110' & \rightarrow & 111 & \rightarrow & 1'11 & \rightarrow & 011 & \rightarrow & 01'1 & \rightarrow & 001 & \rightarrow \\ 01 & & 01 & & 0'1 & & 11' & & 10 & & 10 & & 10 & & 10 & & 10 & & 10 & \end{array}$$

Счетные устройства (примеры схем)

Последовательный двоичный счетчик на базе Гарвардских триггеров



$$ac = \overline{l_1 l_2 l_3 c_3}$$

1)	01 01 01	2)	11 01 01	3)	10 01 01	4)	11 11 01	5)	01 10 01	6)	11 10 01
c=1	11 11 11	c=0	01 11 11	c=1	11 11 11	c=0	10 01 11	c=1	11 11 11	c=0	01 11 11
	10 10 10		10 10 10		01 10 10		01 10 10		10 01 01		10 01 10
7)	10 10 01	8)	11 11 11	9)	01 01 10	10)	11 01 10	11)	10 01 10	12)	11 11 10
c=1	11 11 11	c=0	10 10 01	c=1	11 11 11	c=0	01 11 11	c=1	11 11 11	c=0	10 01 11
	01 01 10		01 01 10		10 10 01		10 10 01		01 10 01		01 10 01
13)	01 10 10	14)	11 10 10	15)	10 10 10	16)	11 11 11	1)	01 01 01		
c=1	11 11 11	c=0	01 11 11	c=1	11 11 11	c=0	10 10 10	c=1	11 11 11		
	10 01 01		10 01 01		01 01 01		01 01 01		10 10 10		

Счетные устройства (примеры схем)

Заметим, что в половине случаев длина переноса (в числе бит счетчика) равна нулю, в 1/4 случаев – единице, in 1/8 – двум и т.д.. Следовательно, когда $n \to \infty$, средняя длина переноса определяется как

$$G = (1/4)1 + (1/8)2 + (1/16)3 + (1/32)4 + \ldots =$$
$$= 1/2 + (1/2)G,$$

откуда G=1.

Следовательно, математическое ожидание числа переключающихся разрядов счетчика равно двум.

Средняя задержка самосинхронного счетчика равна $12\tau_g + 2\tau_{ind}$.

Минимальная задержка равна $6\tau_g + 2\tau_{ind}$.

Максимальная задержка равна $6n\tau_g + 2\tau_{ind}$.

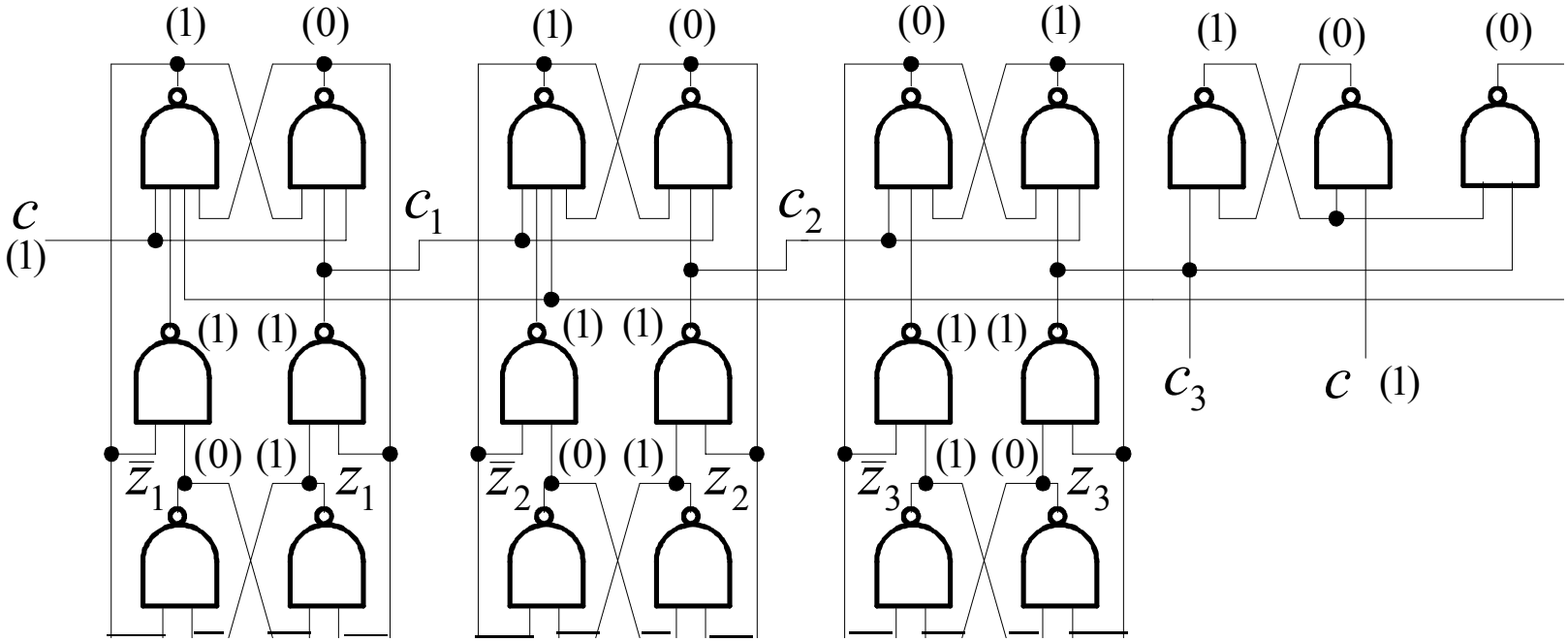
Для синхронного счетчика необходимо выбрать частоту счетного сигнала на основе худшего случая, если нужно считывать значение двоичного кода во время счета.

Счетные устройства (примеры схем)

Последовательный счетчик по модулю N из Гарвардских триггеров

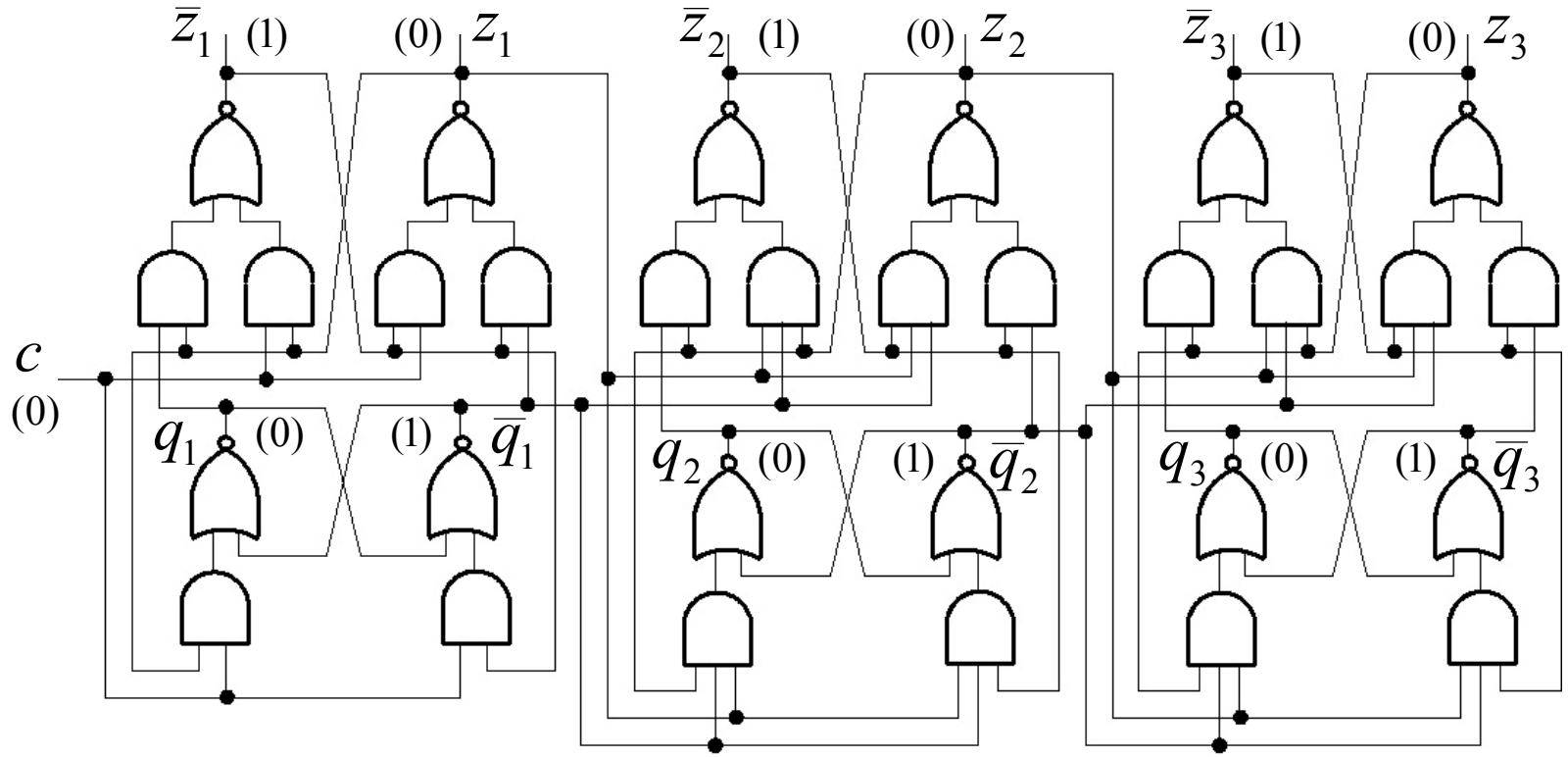
Если счетчик должен считать по модулю N, возможный подход заключается в установлении счетчика в состояние $2^n - N$ каждый раз после выработки сигнала переполнения.

В случае последовательного счетчика такую установку удобно производить, когда счетчик находится в состоянии 000… и c=1. Сигнал переполнения вырабатывается, когда c=0 и счетчик находится в состоянии 111…, следовательно, необходимо запомнить факт переполнения, чтобы использовать его, когда c=1 и сигнал переполнения исчезает. На рисунке представлен счетчик по модулю 5:



Счетные устройства (примеры схем)

Последовательный двоичный счетчик на основе счетных триггеров с разнополярным управлением

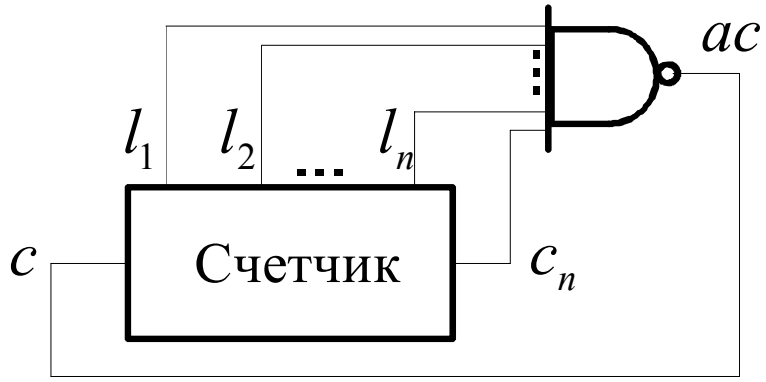


Возможно построение самосинхронного варианта этого счетчика. Функция завершения переходных процессов в счетчике имеет вид:

$$\overline{ac = q_1\bar{z}_1 \vee q_2\bar{z}_2 \vee q_3\bar{z}_3 \vee \bar{q}_3 z_3}.$$

Счетные устройства (примеры схем)

Замечательные свойства самосинхронных счетчиков на основе Гарвардских триггеров



c_n – сигнал переполнения,

l_i – сигналы заёма разрядов счетчика

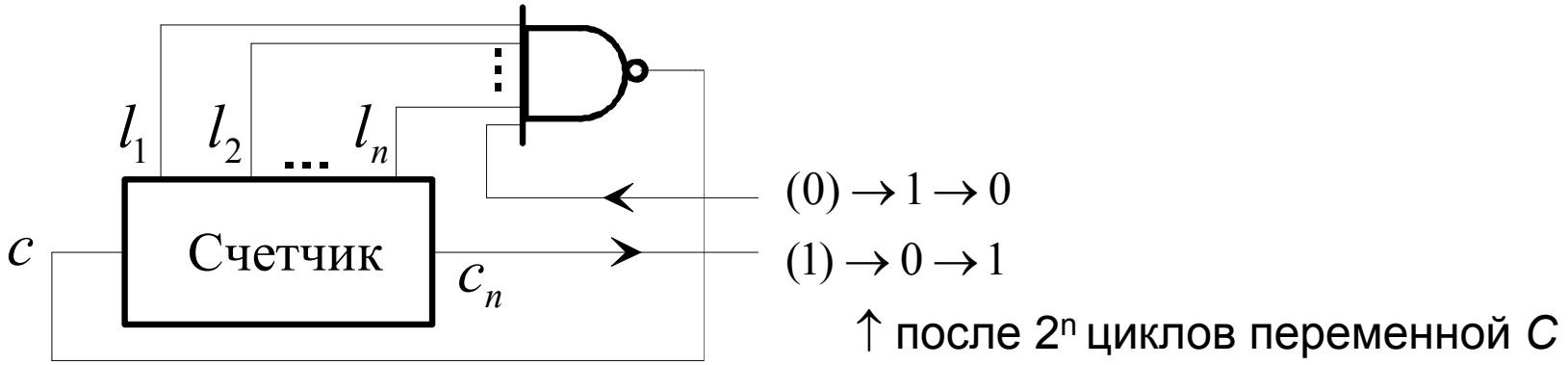
При c=0 переходной процесс в счетчике завершается, когда на одном из выходов $l_1, l_2, ..., l_n$ счетчика появляется значение «0» или $c_n = 0$. При c=1 переходный процесс завершается, когда все эти сигналы становятся равными «1».

Если замкнуть выход ***ac*** на вход ***c***, то счетчик будет работать в режиме генератора с полным рабочим циклом двоичного n-разрядного счетчика, равным 2^n.

В одном цикле счетчика содержится 2^n циклов сигнала *c*, 2^{n-1} циклов сигнала l_1 2^{n-2}циклов сигнала l_2 2^{n-i}циклов сигнала l_i один цикл сигнала l_n и один цикл сигнала переполнения c_n .

Счетные устройства (примеры схем)

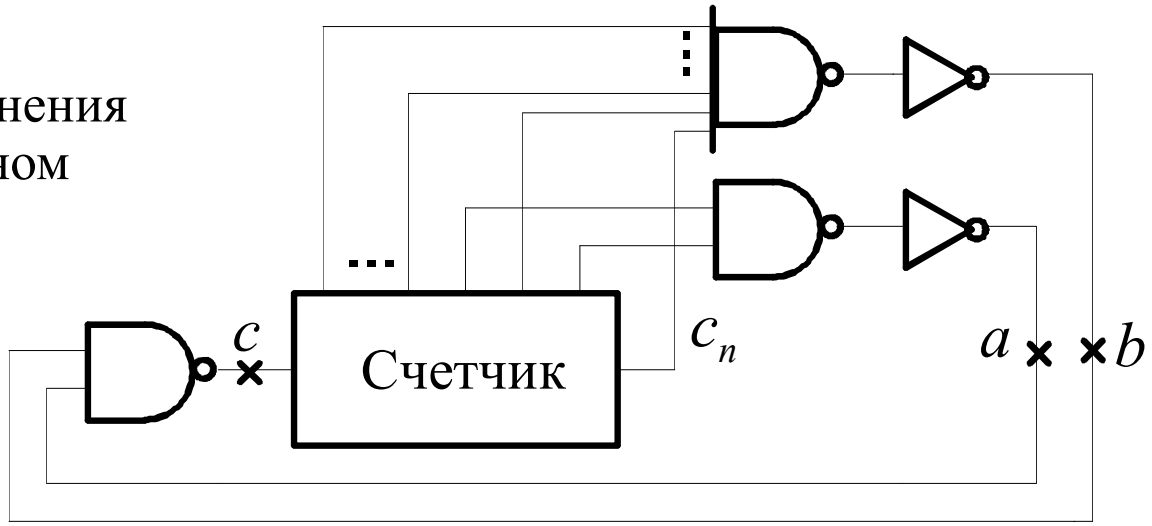
Разорвем провод сигнала of c_n . При этом образуются два полюса. Такая схема может быть использована для умножения на 2^n числа циклов переменной *c* (умножитель частоты).



Разобьем индикатор окончания переходных процессов на две части, например, следующим образом:

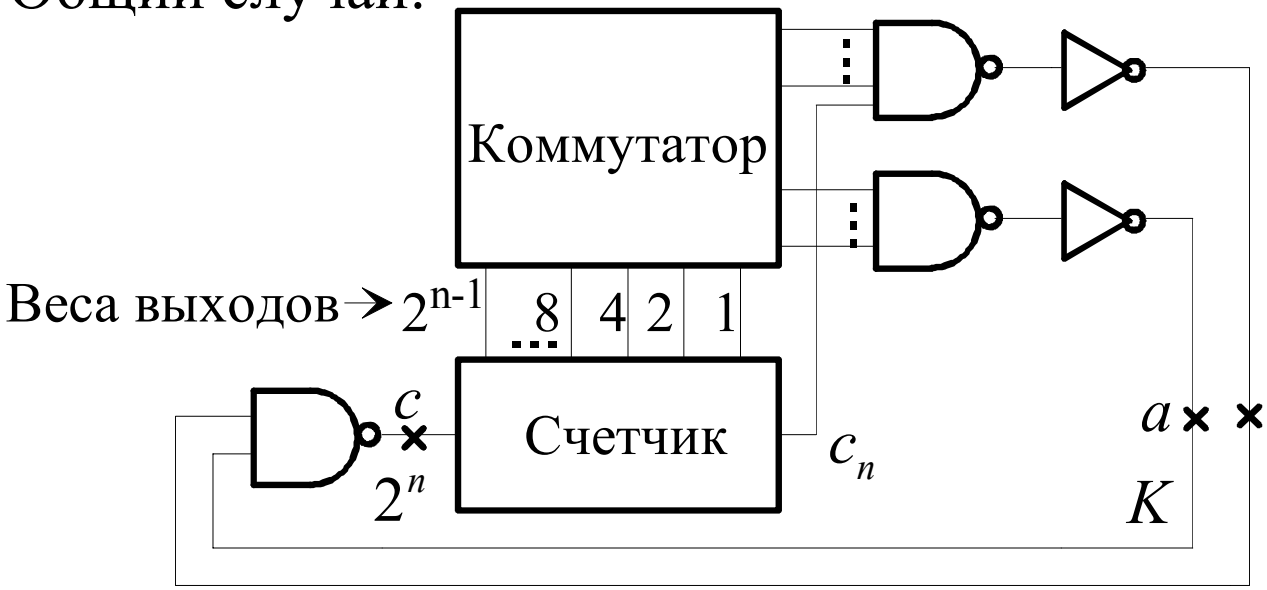
N_x – число циклов изменения переменной *x* в одном цикле счетчика

$$N_c = 2^n$$
$$N_a = 5$$
$$N_b = 2^n - 5$$



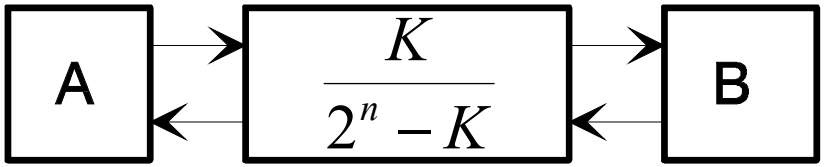
Счетные устройства (примеры схем)

Общий случай:

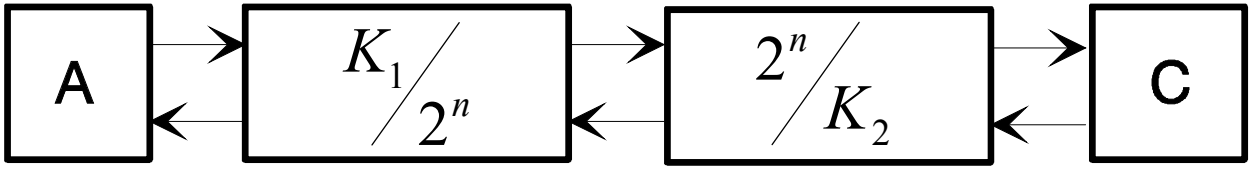


$$\frac{N_a}{N_b} = \frac{K}{2^n - K}$$

$$\frac{N_a}{N_c} = \frac{K}{2^n}$$



$$N_B = \frac{K}{2^n - K} N_A$$



$$N_C = \frac{K_1}{K_2} N_A$$

Отношение K_1 / K_2 может быть любым рациональным числом. Очень интересный результат.

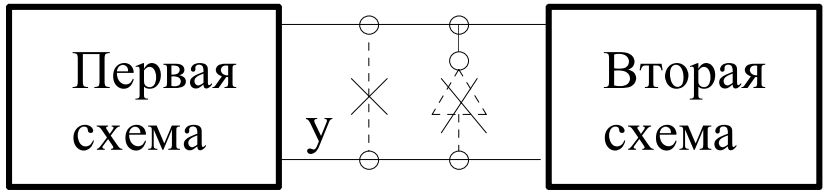
Счетные устройства (примеры схем)

Одна интересная реализация самосинхронного счетчика

В этой реализации схемы разрядов счетчика встраиваются одна в другую.

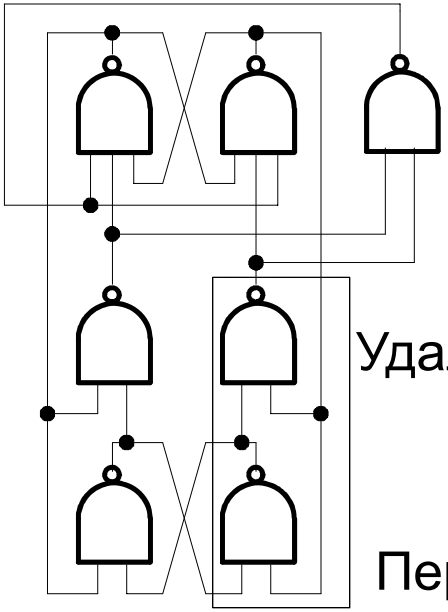
При построении счетчика использовалось расширение теоремы Маллера о соединении самосинхронных схем.

Смысл теоремы может быть объяснен следующим образом. Пусть имеются две самосинхронные схемы. В одной из схем выделяем провод между выходом “*y*” какого-либо элемента и разветвлением и разрываем его, образуя два полюса. Во второй схеме выделяем инвертор и удаляем его. После этого соединяем схемы так, как это показано на рисунке. Получившаяся схема также является самосинхронной.



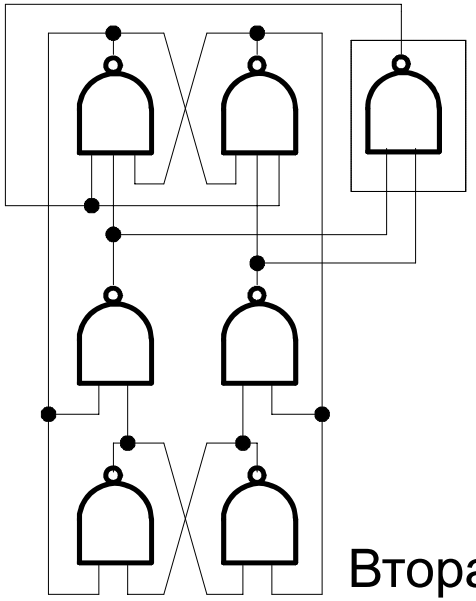
Расширенная теорема рекомендует, как соединять две самосинхронные схемы с полюсами, образованными удалением подсхем в обоих схемах, и как вводить дополнительные связи, чтобы новая схема была самосинхронной.

Счетные устройства (примеры схем)



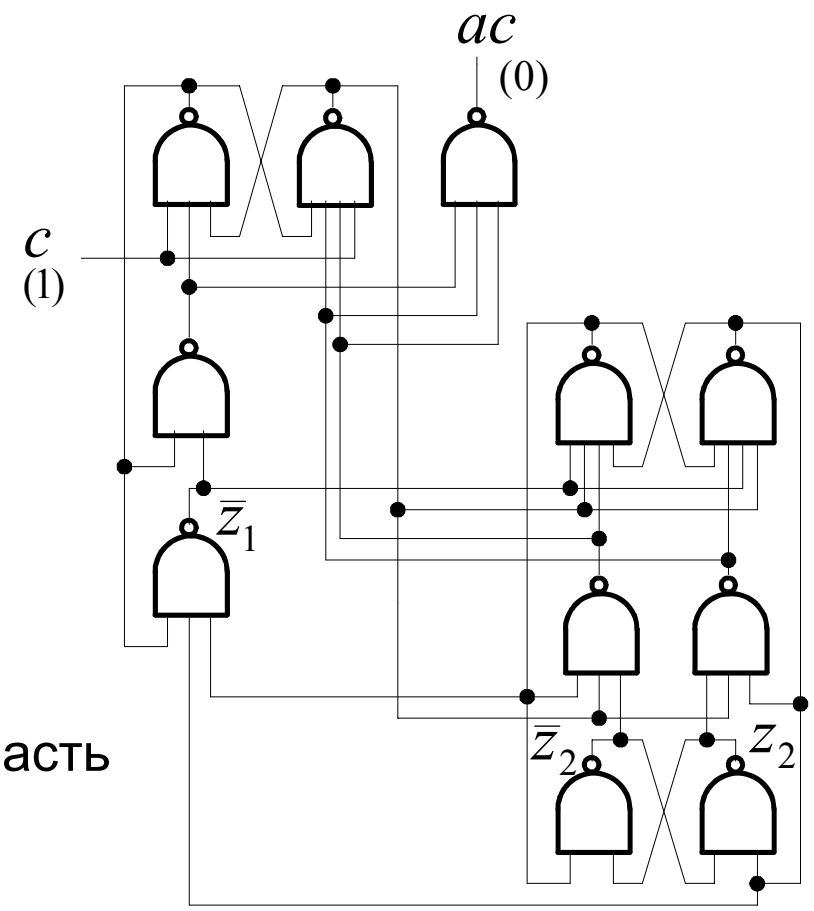
Удаляемая часть

Первая схема



Удаляемая часть

Вторая схема



Двухразрядный счетчик, у которого второй разряд встроен в первый.

Счетные устройства (примеры схем)

Счетчик с постоянным временем ответа

Недостатком последовательных счетчиков является большое время распространения переносов.

При использовании счетчиков в процессах циклического управления важной проблемой является минимизация задержки между счетным сигналом и моментом окончания переходных процессов в счетчике.

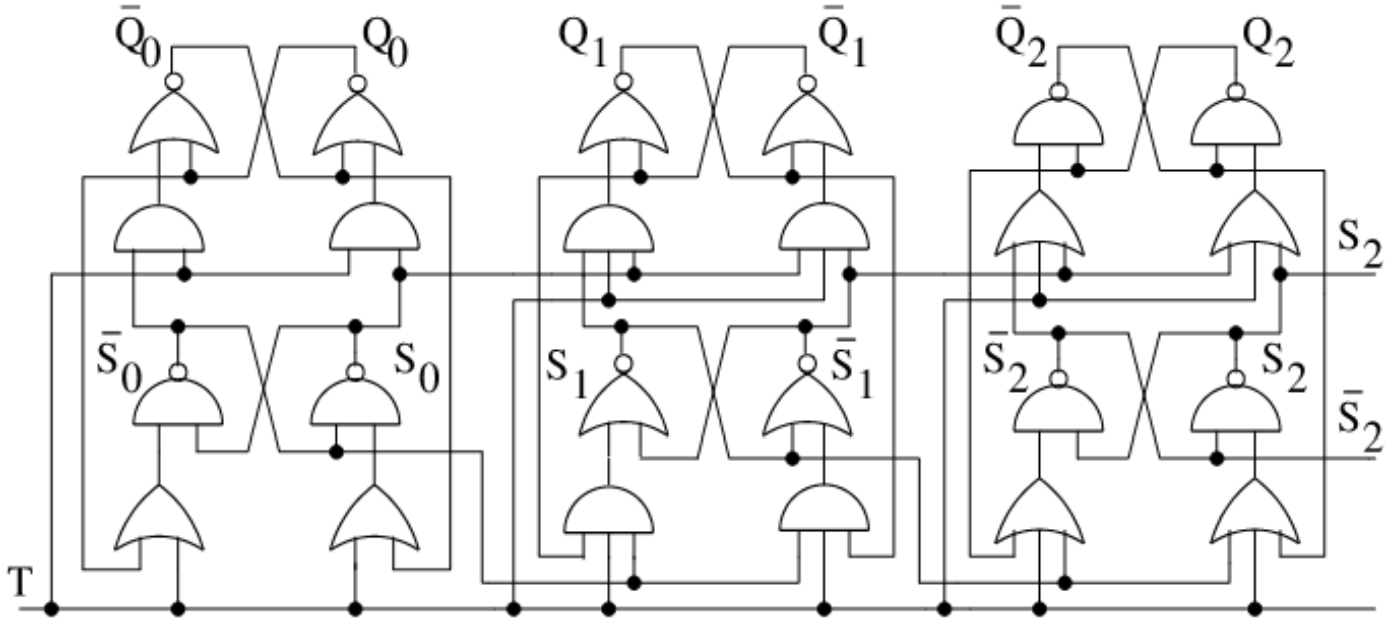
С этой целью могут быть использованы схемы ускорения переноса, которые сами вносят дополнительную задержку в цикл счета, и асинхронные пайплайнизированные счетчики с постоянным временем ответа, имеющие очень высокую сложность.

Профессором Варшавским совместно с коллегами были предложены синхронные счетчики с отложенными переносами. Длительность переходных процессов в таких счетчиках не превосходит времени переключения одного простого триггера (равного задержке двух элементов).

Один из таких счетчиков представлен на слайде. Нулевой разряд счетчика представляет собой обычный счетный триггер с разнополярным управлением . Остальные разряды построены по двухтриггерной схеме с двухпроводным униполярным управлением.

Счетные устройства (примеры схем)

Трехразрядный синхронный счетчик с отложенными переносами



Нечетные разряды переключаются, когда счетный сигнал T=1; четные разряды выполнены по двойственной схеме и переключаются, когда T=0.

Последовательность состояний основных триггеров счетчика отличается от последовательности двоичных чисел.

Счетные устройства (примеры схем)

Последовательности состояний разрядов счетчика

T	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	...
Q_0	**0**	1	**1**	0	**0**	1	**1**	0	**0**	1	**1**	0	**0**	1	**1**	0	**0**	1	**1**	0	**0**	1	**1**	0	**0**	...
Q_1	1	**1**	**1**	0	0	**0**	**0**	1	1	**1**	**1**	0	0	**0**	**0**	1	1	**1**	**1**	0	0	**0**	**0**	1	1	...
Q_2	**0**	**0**	1	1	1	1	**1**	**1**	**1**	**1**	0	0	0	0	**0**	**0**	**0**	**0**	1	1	1	1	**1**	**1**	**1**	...
Q_3	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1	1	...
Q_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	...

φ_2

φ_3

φ_4

Каждый разряд счетчика может находиться в одном из четырех состояний.
Введем следующие обозначения:

{0,1} – состояния разряда, когда состояния его простых триггеров различны,

{**0,1**} – состояния разряда, когда состояния его простых триггеров одинаковы.

На рисунке представлены последовательности состояний разрядов 5-разрядного счетчика .

Счетные устройства (примеры схем)

Когда счетный сигнал T изменяет свое значение, нулевой разряд счетчика переходит в одно из состояний {0,1}, если T=1, и в одно из состояний {**0,1**}, если T=0. Каждый следующий разряд, когда он работает, переходит в одно из состояний {**0,1**} , если предыдущий разряд находится в состоянии **0** или 1, и в одно из состояний{0,1}, если предыдущий разряд находится в состоянии **1** или 0. В моменты времени T=0 счетчик дает следующую последовательность чисел: 10,15,12,13,6,3,0,1,2,7,4,5,14,11,8,9,26,31,28,29,… . Однако полный цикл счета такого n-разрядного счетчика равен 2^n.

Легко видеть, что последовательность состояний предложенного счетчика имеет постоянные фазовые сдвиги, начиная с Q_2. Будем измерять фазовый сдвиг числом изменений значений сигнала T.

Фазовый сдвиг j-го бита $\varphi_j = \varphi_{j-1} + 2^{j-1} - 1$ для $j \geq 2$. При начальном условии $\varphi_1 = 0$ решением этого уравнения является $\varphi_j = 2^j - j - 1$. Согласно этому решению имеем последовательность фазовых сдвигов: *0, 0, 1, 4, 11, 26, 57, ...* . Простое выражение для фазового сдвига дает возможность построить алгоритм преобразования двоичных чисел в состояния разрядов счетчика.

Счетные устройства (примеры схем)

Заметим, что состояние счетчика, соответствующее нулевому двоичному числу зависит от длины счетчика. Для 5-разрядного счетчика это состояние **0**1**01**0. Это состояние является сигналом переполнения при счете по модулю 2^n.

Счетчики в коде Грея

Код Грея может быть получен из двоичного кода по формуле:

$$g_i = b_i \oplus b_{i+1},\ i = 1,2,3,\ldots,n,\ b_{n+1} = 0.$$

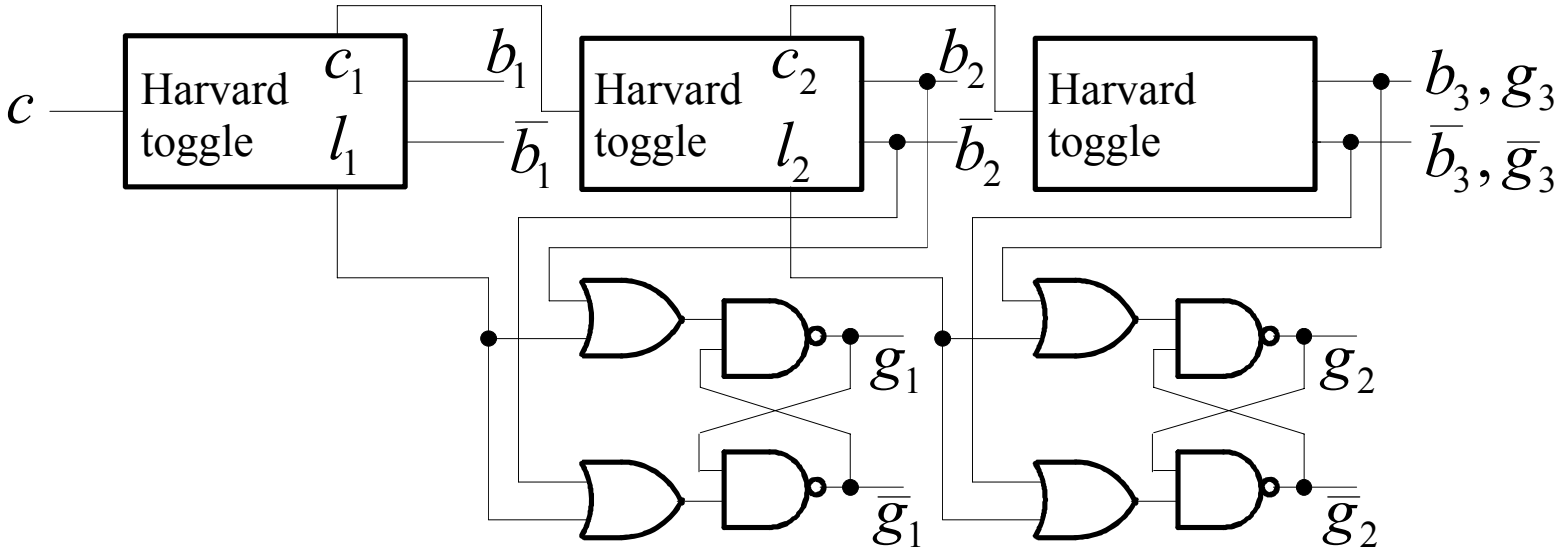
где b_i – значение i-го бита двоичного кода, а g_i – значение i-го бита кода Грея.

Двоичный код	Код Грея	Двоичный код	Код Грея
0000	0000	0001	0011
1000	1000	1001	1011
0100	1100	0101	1111
1100	0100	1101	0111
0010	0110	0011	0101
1010	1110	1011	1101
0110	1010	0111	1001
1110	0010	1111	0001

Счетные устройства (примеры схем)

Основным преимуществом кода Грея является то, что в нем любые два смежных кода отличаются значением только одного бита. Это дает возможность считывать двоичные числа, представленные кодом Грея, с точностью половины значения двоичного разряда с наименьшим весом.

Трехразрядный счетчик, считающий одновременно в двоичном коде и коде Грея

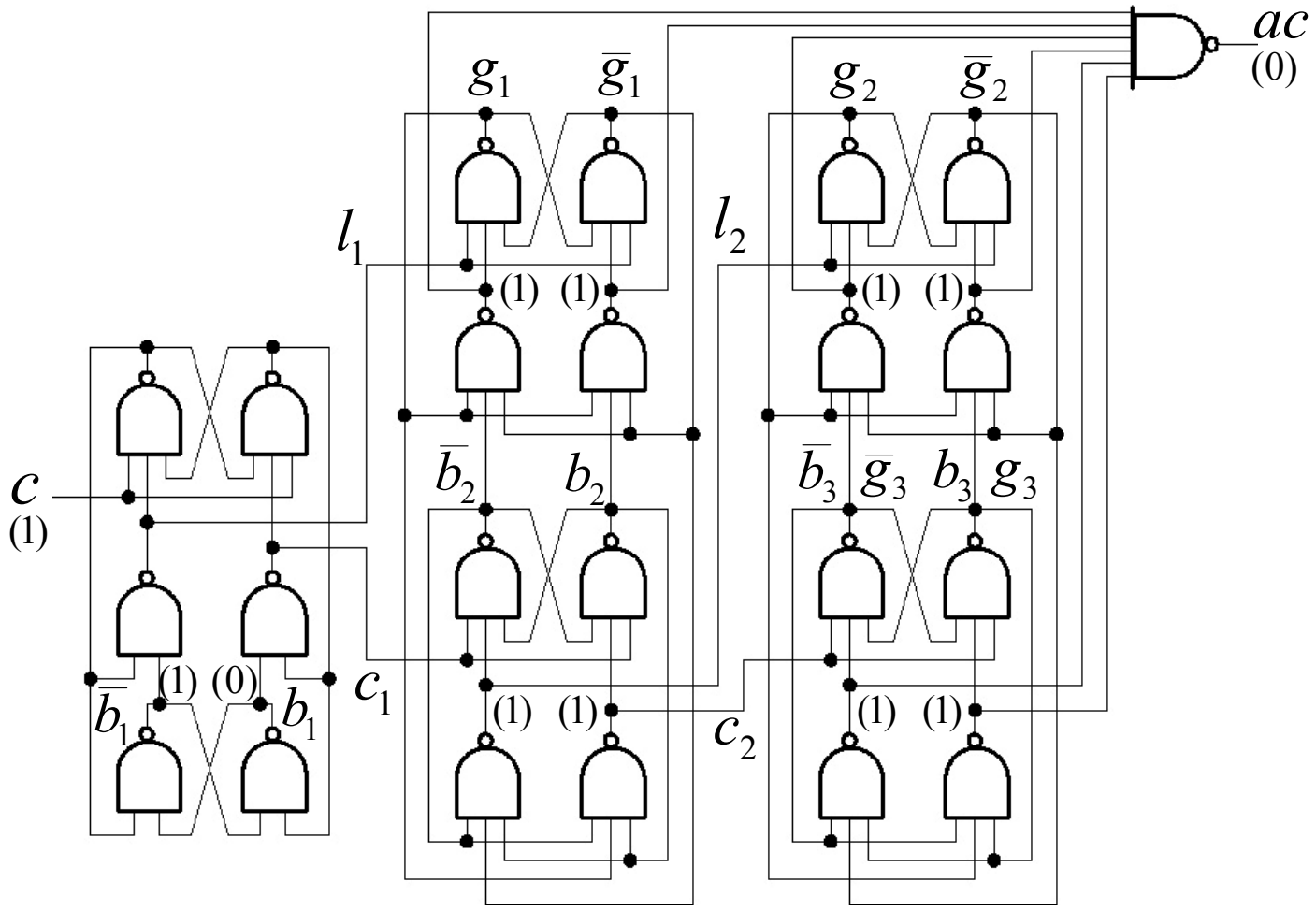


Номер двоичного разряда, в котором заканчивается перенос, указывает номер бита кода грея, изменяющего свое значение.

Попытаемся упростить реализацию этого счетчика.

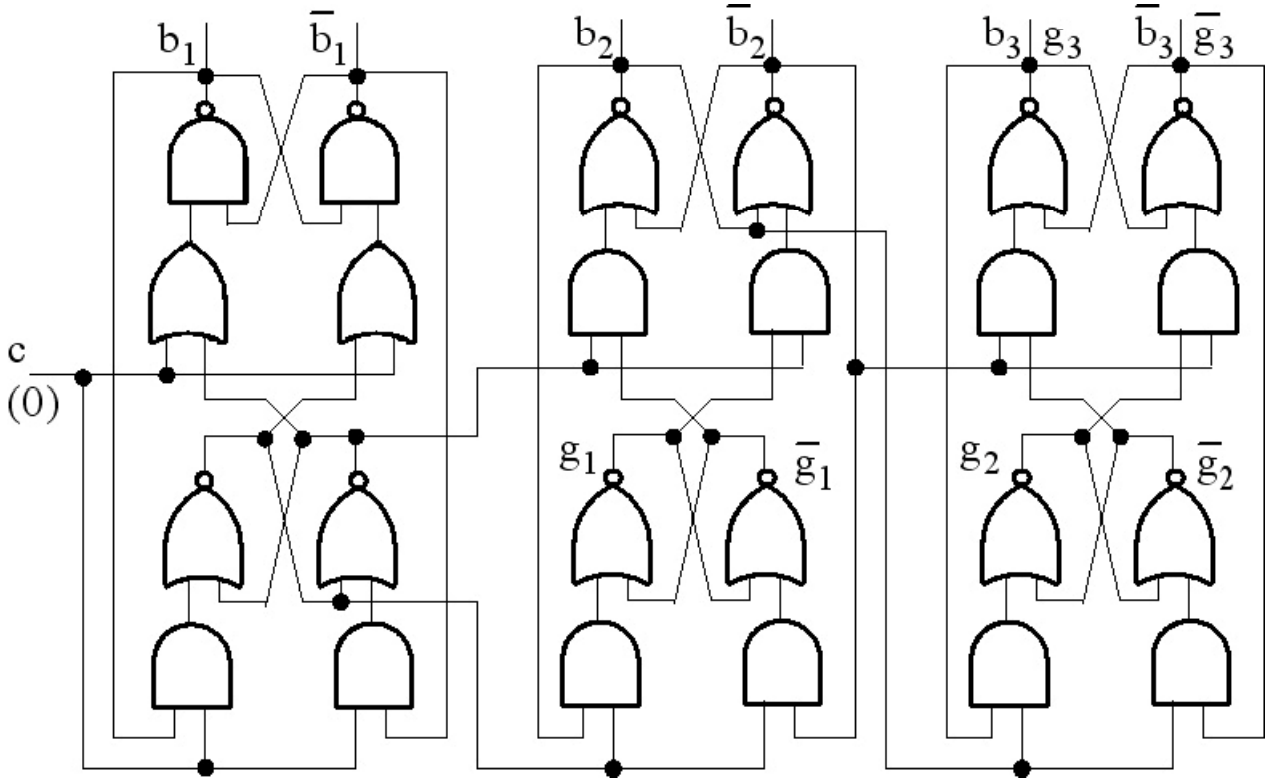
Счетные устройства (примеры схем)

Упрощенный самосинхронный трехразрядный счетчик, считающий одновременно в двоичном коде и коде Грея



Счетные устройства (примеры схем)

Минимальный трехразрядный счетчик, считающий одновременно в двоичном коде и коде Грея



Логика

1. Совершенная дизъюнктивная нормальная форма.
2.Совершенная конъюнктивная нормальная форма.
3. Правила де Моргана.
4. NPN-классификация булевых функций, классы эквивалентности.
5. Понятие элемента n-куба размерности k, примеры, внутренние и внешние переменные, характеристическая функция элемента (импликанта).
6. Разложение Шеннона, пример, двойственное разложение, бинарные деревья (BDD).
7. Понятие производной логической функции и её вычисление.
8. Разложение Рида-Маллера, показать на примере функции T={0,1,5,6,7,8,14,15}, F={2,3,4,9,10,11,12,13}.
9. Диагональная декомпозиция Варшавского, показать на примере функции T={0,1,5,6,7,8,14,15}, F={2,3,4,9,10,11,12,13}.
10. Понятие простого элемента, представление простого элемента с помощью двух векторов.
11. Покрытие функции, сокращенная форма булевой функции, метод Блейка-Парецкого получения сокращенных форм.
12. Звезда из элементов, существенный элемент, неизбыточное покрытие, тупиковое и минимальное покрытие.
13. Неэлементарное поглощение. Визуальная минимизация.
14. Арифметические операции с номерами вершин. Вычисление номеров диагональных соседей.
15. Процедура Урбано-Мюллера для вычисления простых элементов булевых функций.
16. Слабо определенные булевы функции и их минимизация.
17. Минимизация слабо определенных булевых функций с помощью таблицы связей.
18. Вычисление существенных элементов при минимизации слабо определенных булевых функций.
19. Понятие функциональной декомпозиции. Двухуровневая декомпозиция.
20. Простая разделительная декомпозиция и теорема о ее существовании.
21. Многократная разделительная декомпозиция.
22. Изотонные, антитонные и монотонные булевы функции.Свойства монотонных функций.
23. Двойственные и самодвойственные функции. Самодвойственные функции трех и четырех переменных. Свойства самодвойственных функций.
24. Симметрические функции и их свойства.
25. Монотонные симметрические функции и их свойства.
26. Полиномиальное представление логических функций.
27. Функциональная полнота системы логических функций (теорема Поста).
28. Состязания в логических схемах и их виды.
29. Функциональные состязания. Статические функциональные состязания.
30. Функциональные состязания. Динамические функциональные состязания.
31. Функциональные состязания. Критерии отсутствия динамических функциональных состязаний.
32. Функциональные состязания. Классы допустимых траекторий переходов: с соседними переходами, со сравнимыми переходами.
32. Квази-монотонные и супер-монотонные функции. Двухфазная дисциплина смены входных наборов.
33. Организация внешней среды при различных допустимых дисциплиах смены входных наборов (соседние переходы, сравнимые переходы, с двухфазной дисциплиной).
34. Логические состязания. Статические и динамические логические состязания. Анализ логических состязаний с помощью БТФ.

35. Логические состязания. Методы борьбы с логичесеими состязаниями.
36. Пороговые функции и их геометрическое представление.
37. Пороговые функции. Вращение гиперплоскости инвертированием переменных. Менотонность пороговых функций.
38. Связь весовых коэффициентов пороговых функций со стандартной задачей линейного программирования. Задание пороговых функций опорными множествами.
39. Реализуемость пороговых элементов.
40. Суммирующий усилитель как пороговый элемент. Анализ параметрической ошибки.
41. Основные области применения пороговых и мажоритарных функций.
42. Логический синтез из трехвходовых мажоритарных элементов.
43. Мажоритарная декомпозиция.
44. Мостиковые схемы.
45. Инвертирование мостиковых схем.

Автоматы

46. Что такое последовательностные схемы?
47. Модель конечного автомата. Автоматы Мили и Мура.
48. Прямое и обратное преобразование автоматов Мили и Мура.
49. Спецификации конечных автоматов.
50. Распознавание автоматом свойств конечных последовательностей.
51. Распознавание автоматом свойств неограниченных последовательностей. Язык регулярных выражений.
52. Два подхода к пониманию распознавания автоматом регурярных выражений.
53. Построение графа переходов автомата по регулярному выражению, прямой алгоритм МакНотона-Ямады.
54. Нахождение регулярного выражения по графу переходов автомата Мура, обратный алгоритм МакНотона-Ямады.
55. Минимизация состояний полностью определенного автомата. К-эквавалентность и к-различимость состояний.
56. Минимизация состояний полностью определенного автомата с помощью таблиц разбиений.
57. Минимизация состояний полностью определенного автомата с помощью таблицы пар состояний.
58. Неполностью определенная спецификация автомата и недетерминизм, частичный автомат.
59. Минимизация состояний частичного автомата.
60. Логическое и физическое время, синхронизация.
61. Асинхронные, синхронные и самосинхронные автоматы. Структура синхронного автомата.
62. Стандартная процедура синтеза простых синхронных автоматов.
62. Проблема декомпозиции автоматов. Типы декомпозиции.
63. Размещение состояний автоматов. Оценка сложности.
64. Размещение состояний автоматов. Рекомендуемые правила размещения.
65. Размещение состояний автоматов. Использования графа импликаций. Порядок использования правил при размещении состояний.
66. Общность проблемы декомпозиции и кодирования автоматов. Разбиения и операции над разбиениями. Построение алгебраической структуры разбиений.
67. Разбиения со свойством подстановки. Построение алгебраической структуры разбиений со свойством подстановки. Условия существования параллельной и последовательной декомпозиции автоматов.

68. Общая декомпозиция и кодирование состояний. Пары разбиений.
69. Автоматы, не зависящие от скорости. Модель элемента. Допущения Маллера.
70. Диаграммы Маллера. Конфликты в диаграммах Маллера. Построение функций элементов по диаграммам Маллера.
71. Дистрибутивные и недистрибутивные диаграммы Маллера, сигнальные графы и диаграммы изменений.
72. Управление событиями. Моделирование сетей Петри.
73. Асинхронный конвейер.
74. Счетные устройства. Самосинхронные счетчики. Математическое ожидание длины переноса.
75. Замечательное свойство самосинхронных счетчиков. Структурная схема устройства с рациональным числом пересчета.
76. Пересчетная схема с постоянным временем ответа.
78. Счетчики, одновременно считающие в двоичном коде и коде Грея.