# Алгоритмы и структуры данных в построении и анализе СБИС

доцент кафедры теоретической кибернетики Мубаракзянов Р.Г.

21 марта 2011 г.

# Содержание

# 1 Введение

Основными компонентами компьютеров и других цифровых систем являются *схемы*. В схемах провода соединяют определённым образом один или несколько заряженных входных портов с одним или несколькими выходными портами. В простейшей модели различаются два напряжения на входах и выходах: *заряжено* , которое обозначается «1», и *незаряжено* , которое обозначается «0». В действительности, такой бинарное представление символизирует, что заряд не выходит за пределы двух непересекающихся непрерывных интервалов зарядки.

*Сверхбольшие интегральные схемы, СБИС,* представляет собой сложную комбинацию (соединение) ограниченного числа основных, или функциональных, элементов, которые осуществляют простую логическую операцию. Хотя операции зависят не от точного значения входного сигнала, а от соответствующего интервала зарядки, можно моделировать (кодировать) входные и выходные значения функциональных элементов «0» и «1», таким образом, элементы и схемы с $n$ входами и одним выходом, соответствуют функциям $\{0,1\}^n \to \{0,1\}$.

Настоящее методическое пособие представляет собой конспект курса лекций, прочитанных в весеннем семестре 2005 г. для студентов третьего курса факультета ВМК КГУ и посвященных анализу алгоритмов и структур данных, используемых в построении и анализе СБИС [1]. Несмотря на актуальность данной тематики, в русскоязычной литературе практически отсутствует материал на эту тему. Курс базируется в основном на книге *"C. Meinel, T. Theobald. Algorithms and data Structures in VLSI Design. Springer, 1998"*, а также на статьях различных авторов и собственных результатах лектора. В составлении пособия оказали помощь студенты И.Бурмистров, Д.Складчиков, М.Абрамский.

[1] СБИС - сверхбольшая интегральная схема

# 2 Булевы алгебры и булевы функции

В 1936 году Клод Шеннон (C.E.Shennon) показал, что основные законы математической логики и теории множеств, сформулированные Джорджем Булем (G.Boole) в его книге в 1854 году, могут быть использованы в описании и анализе схем из функциональных элеметах. В данной главе рассматриваются основные определения и свойства булевых алгебр и булевых функций, которые понадобятся нам в дальнейшем.

**Определение.** Множество $A$, содержащее элементы 0 и 1, бинарные операции "$+$"и "$\cdot$"и унарную операцию "$\neg$ называется *булевой алгеброй*, если $\forall a, b \in A$ выполняются следующие утверждения:

1. Коммутативный закон:

   $a + b = b + a$,
   $a \cdot b = b \cdot a$;

2. Распределительный закон:

   $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$,
   $a + (b \cdot c) = (a + b) \cdot (a + c)$;

3. Существование нейтрального элемента:

   $a + 0 = a$ (для сложения),
   $a \cdot 1 = a$ (для умножения);

4. Существование обратного элемента:

   $a + \bar{a} = 1$ (для сложения),
   $a \cdot \bar{a} = 0$ (для умножения);

Будем рассматривать лишь конечные алгебры. Принято считать, что операция $^-$ имеет приоритет над операцией $\cdot$, а $\cdot$ имеет приоритет над $+$. Примем также запись $ab$ для $a \cdot b$.

**Примеры булевых алгебр.**

1. $2^s$ – множество подмножеств $S : \forall A \subseteq S, \bar{A} = S\ A \Rightarrow$ алгебра подмножеств $(2^s, \cup, \cap, \bar{}\ , \varnothing, S)$ – *Булева алгебра*;

2. для $n > 1$, пусть $n$ имеет лишь различные простые делители: $n = p_1 p_2 \dots p_k$, $p_1 < p_2 < \dots < p_k : T_n$ – множество делителей $n \Rightarrow (T_n$, НОК, НОД, $(\cdot)^{-1}$, 1, $n)$ – Булева алгебра.

Булева алгебра удовлетворяет принципу двойственности:

**Определение.** Если $G$ – некоторое равенство в булевой алгебре $(A, +, \cdot, 0, 1)$ то $G'$ – *двойственное* равенство, полученное из $G$ взаимной заменой $+$ на $\cdot$ и «0» на «1». Например, если $G : a + 0 = a$, то $G' : a \cdot 1 = a$.

**Теорема 1.** (*Принцип двойственности*) Пусть $G'$ – двойственное равенство для $G$. Если $G$ – верное высказывание в теории булевых алгебры, то $G'$ также истинно.

**Доказательство.** Доказательство очевидно, так как аксиомы сохраняют истинность булевой алгебры при переходе к двойственным равенствам. Таким образом, исходя из принципа двойственности достаточно доказать высказывания в булевой алгебре лишь в одной из двух версий.□

**Теорема 2.** (*Основной закон вычислений*) Для любых $a, b, c$ в булевой алгебре $(A, +, \cdot, 0, 1)$ выполняются:

1. Законы идемпотепции: $a \cdot a = a$, $a + a = a$;

2. Свойства нейтральности элементов: $a + 1 = 1$, $a \cdot 0 = 0$;

3. Законы поглощения: $a + (a \cdot b) = a$, $a \cdot (a + b)$;

4. Законы ассоциативности: $a + (b + c) = (a + b) + c$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$;

5. Законы Де – Моргана: $\overline{a + b} = \overline{a} \cdot \overline{b}$, $\overline{a \cdot b} = \overline{a} + \overline{b}$;

6. Законы двойственность отрицания: $\bar{\bar{a}} = a$.

Докажем, например, закон поглощения: $a + (a \cdot b) = a \cdot 1 + a \cdot b = a \cdot (1 + b) = a \cdot 1 = a$. Остальные доказательства опустим. (Домашнее задание)

**Определение.** *Булевы функции* — это функции, описываемые выражениями булевой алгебры или булевыми формулами.

**Определение.** Пусть $B = (A, +, \cdot, \bar{}\ , 0, 1)$ — булева алгебра. Выражение, содержащее переменные $x_1 \dots x_n$, символы $+, \cdot, \bar{}$ и элементы A называются *булевой формулой над переменными*, если выполняется следующее:

1. Элемент $A$ – булева формула;

2. $x_1 ... x_n$ – булевы формулы;

3. Если $F$ и $G$ – булевы формулы, то

   - $(F) + (G)$ – булева формула;

- $(F) \cdot (G)$ – булева формула;
- $(\bar{F})$ – булева формула

4. выражение является булевой формулой, если оно может быть получено конечным числом применений правил 1,2,3.

**Замечание.** Можно опускать скобки (приоритеты).

Булева формула *индуцирует* булеву функцию $f : A^n \to A$: $f(a_1, \ldots, a_n)$ есть элемент $A$ при замене $x_i$ на $a_j$.

**Определение.** Пусть $B = (A, +, \cdot, \bar{}\,, 0, 1)$ — булева алгебра. Функция от $n$ переменных: $A^n \to A$ — *булева функция*, если она может быть порождена булевой формулой. Будем говорить, что $F$ представляет $f$.

**Определение.** Булевы функции от $n$ переменных *эквивалентны*, если их значения совпадают на всех $2^n$ входных векторах из $\{0, 1\}^n$.

**Теорема 3**. $f$ – эквивалентна $g$ $(f \approx g) \Leftrightarrow f = g$ :
$\forall (a_1, \ldots, a_n) \in A \quad f(a_1, \ldots, a_n) = g(a_1, \ldots, a_n)$.

**Доказательство.** Из аксиом и основных законов можно показать, что любая формула может быть переписана в виде суммы:

$$F = \sum_{(e_1 \ldots e_n) \in (0,1)^n}^{n} a(e_1 \ldots e_n) x_1^{e_1} \ldots x_n^{e_n},$$

где

$$x_i^1 = x_i, \quad x_i^0 = \bar{x}_i, \quad a(e_1 \ldots e_n) \in A.$$

Таким образом, любая булева функция однозначно определяется коэффициентами $a(e_1, \ldots, e_n)$, которые зависят лишь от $(e_1, \ldots, e_n) \in [0, 1]^n$.□

**Следствие 1.** Количество различных булевых функций равно $|A|^{2^n}$.

# 3 Переключательные функции

Будем рассматривать специальный случай булевой алгебры с двумя элементами, которая является теоретической основой построения схем. Важность этой алгебры была показана ещё в 1910 году физиком Эренфестом (Ehrenfest), занимавшимся разработкой переключательных схем в телефонных сетях. Наибольшее продвижение получили исследования в 1936-38 гг. в Японии, США, СССР, среди которых выделяются работы Клода Шеннона.

Итак, рассмотрим следующую булеву алгебру:

$$(B, +, \cdot, \bar{}\,, 0, 1),$$

где $B = \{0, 1\}$, $a + b = max[a, b]$, $a \cdot b = min[a, b]$, $\bar{0} = 1$, $\bar{1} = 0$.

**Определение.** Функция $f : B^n \longrightarrow B$ от $n$ переменных называется *переключательной функций*. Множество переключательных функций будем обозначать как $B_n$.

Легко показать, что количество различных переключательных функций также равно $2^{2^n}$. Из следствия 1 предыдущей главы следует, что количество различных булевых функций в $B_n$ равно $2^{2^n}$. Следовательно, любая переключательная функция является булевой функцией. Поэтому в дальнейшем будем использовать термин «булева функция», подразумевая переключательную функцию.

Булева функция с $n$ входами и $m$ выходами описывается «$m$»-кой $f = (f_1, \ldots, f_m)$, где $f_1, \ldots, f_m$ - булевы функции. Обозначим такие функции $B_{n,m}$ ($B_{n,1} = B_n$).

**Теорема.** Число различных булевых функций от $n$ переменных с $m$ выходами равно $(2^m)^{2^n} = 2^{m \cdot 2^n}$.

**Определение.** Для функции $f \in B_n$ набор $a = (a_1, \ldots, a_n) \in B^n$ называется *1-набором* или *выполнимым набором*, если $f(a) = 1$.

**Определение.** *1-множеством* функции $f \in B_n$ называется множество $L_f^1 = \{(a_1, \ldots, a_n) | f(a_1, \ldots, a_n) = 1\}$.

**Определение.** Переменная называется $x_i$ *существенной* для функции $f \in B_n$, если существует набор:

$$(a_1, \ldots, a_n) : f(a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n) \neq f(a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_n)$$

Функция $f$ может быть отождествлена со своим 1-множеством $L_f^1$. Фактически, $f$ является *характеристической функцией* своего 1-множества:

$$f(a) = X_L(a) = \begin{cases} 1, & a \in L \\ 0, & a \notin L \end{cases}$$

.

Булевы функции от 2 или меньше переменных:

a) от 0 переменных: константы 0, 1: $1(x_1, \ldots, x_n) = 1, 0(x_1, \ldots, x_n) = 0$

b) от 1 переменной: 4 функции $\{0, 1, x, \bar{x}\}$, $x, \bar{x}$ - литералы.

c) от 2 переменных: 16 функций: дизъюнкция ($\bigvee$, $+$), конъюнкция (&, $\bigwedge$, $\cdot$) и др.

Фиксируя значения некоторых переменных функции $f$, можно получать ее *подфункции*.

**Теорема (разложение Шеннона).** Пусть $f$ – булева функция от $n$ переменных, тогда для подфункций $g, h \in B_{n-1} : g(x_1, \ldots, x_{n-1}) = f(x_1, \ldots, x_{n-1}, 0)$, $h(x_1, \ldots, x_{n-1}) = f(x_1, \ldots, x_{n-1}, 1)$, выполняется

$$f = \bar{x}_n g + x_n h.$$

Доказательство очевидно. □

**Замечание.** Каждой переменной можно поставить в соответствии: элемент из $\{0, 1, n\} : c_j(x_i) \in \{0, 1, \}, c_j(x_1) = 0 \Leftrightarrow x_i = 0, c_j(x_i) = 1 \Leftrightarrow x_i = 1$, если $c_j(x_i) = n$, то $x_i$ нефиксирована (свободна): $f_{c_j}(x_1, \ldots, x_n) = f(c_j(x_1), \ldots, c_j(x_n)) \Rightarrow$ Существует $3^n$ подфункций от $n$ переменных (не обязательно различных).

**Следствие 2.** $\forall f \in B_n$ и $\forall i : 1 \leq i \leq n$ верно:

1. *Разложение Шеннона по i-му аргументу*:

$$f(x_1, \ldots, x_n) = x_i f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) + \bar{x}_i f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n);$$

2. *«Двойственное» разложение Шеннона*:

$$f(x_1, \ldots, x_n) = (x_i + f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n))(\bar{x}_i + f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n));$$

3. *Разложение Шеннона относительно* $\oplus$:

$$f(x_1, \ldots, x_n) = x_i f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) \oplus \bar{x}_i f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n);$$

**Доказательство следствия 2:**

1. очевидно;

2. следует из признака двойственности;

3. следует из того, что одно из слагаемых равно нулю. □

# 4 Важные свойства булевых функций

Существует ряд свойств булевых функций, известных из курса дискретной математики: монотонность, симметричность и т.д. В этой главе мы рассмотрим некоторые из них.

**Определение.** Пусть $f \in B_n$. $f$ *монотонно возрастает (убывает)* по $i$-му аргументу, если для любого $a \in B^n$:

$$f(a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n) \leq f(a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_n).$$

Если функция монотонно возрастает (убывает) по каждому своему аргументу, она называется *возрастающей (убывающей)*.

**Теорема (без д-ва).** Для того, чтобы $f$ была монотонно возрастающей (убывающей), необходимо и достаточно выполнения условия $\forall a, b \in B^n (a \leq b \Rightarrow f(a) \leq f(b) (f(a) \geq f(b)$ для убывающей).

**Теорема (без д-ва).** $f \in B_n$ является монотонно возрастающей (убывающей) по $i$–му аргументу тогда и только тогда, когда $f$ может быть представлена в виде:

$$f = x_i g + h \quad (f = \bar{x}_i g + h) \quad \text{для g и h, независящих от x.}$$

**Определение.** Функция $f \in B_n$ называется *симметрической*, если любая перестановка $\pi$ значений переменных не изменят значения функции, т.е. $f(x_1, \ldots, x_n) = f(x_{\pi(1)}, \ldots, x_{\pi(n)}) \forall \pi$.

Таким образом, функция является симметрической тогда и только тогда, когда она зависит от только количества единиц среди переменных, но не от их позиций. Следовательно, симметрические функции однозначно определяются их значениями на векторах:

$$\nu_i = (0, 0, \ldots, 0, \underbrace{1, \ldots, 1}_{\text{i}}),$$

$$\nu_i \in B^n, i = \overline{0, n}$$

**Замечание.** Легко доказать, что существует $2^{n+1}$ симметрических функций.

**Примеры важных симметрических функций:**

1. Функция Четности ($Parity$):

$$Par_n(x_1, \ldots, x_n) = \bigoplus_{i=1}^{n} x_i$$

2. Функция голосования ($Majority$):

$$Maj_n(x_1, \ldots, x_n) = 1 \Leftrightarrow \sum_{i=1}^{n} x_i \geq \frac{n}{2}$$

3. Пороговая функция:

$$T_k^n \in B_n, \quad 0 \le k \le n$$

$$T_k^n(x_1, \ldots, x_n) = 1 \Leftrightarrow \sum_{i=1}^{n} x_i \ge k$$

4. Обратная пороговая функция:

$$T_{\le k}^n \in B_n, \quad 0 \le k \le n$$

$$T_{\le k}^n(x_1, \ldots, x_n) = 1 \Leftrightarrow \sum_{i=1}^{n} x_i \le k$$

5. Интервальная функция:

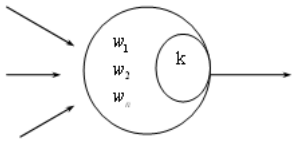
$$I_{k,m}^n(x_1, \ldots, x_n) = 1 \Leftrightarrow k \le \sum_{i=1}^{n} x_i \le m$$

**Замечание.** Каждая симметрическая функция может быть представлена, как дизъюнкция интервальных функций.

6. Взвешенная пороговая функция с весами $w_1, \ldots, w_n \in R_n$ и порогом $k \in R$:

$$T_{w_1,\ldots,w_n}(x_1, \ldots, x_n) = 1 \Leftrightarrow \sum_{i=1}^{n} w_i x_i \le k.$$

**Замечание.** Эта функция играет большую роль при моделировании нейронов и конструкции нейронных сетей.




В заключение хотелось бы отметить еще один важный класс булевых функций - *частично определенные функции*. Они возникают в том случае, когда некоторые входные наборы можно не рассматривать (т.е. они не могут возникнуть или реакция системы на них несущественна). В этом случае, кроме 1-множества и 0-множества, определяется $N$-множество - множество несущественных наборор.

# 5 Лекция 4. Вычислимые функции и сложность вычислений (неформальное введение)

Прежде чем говорить о представлении функции, нам нужно пояснить некоторые понятия теории вычислимости и сложности.

Одна из целей теории сложности – разбиение функций в зависимости от сложности на различные классы.

Из предыдущего курса вы знакомы с **иерархией Хомского**:

$$Reg \subset \text{КСЯ} \subset \text{КЗЯ} \subset Gram$$

Но существует и другое разбиение функций.

Какие функции вычислимы?

**Тезис Черча.** Интуитивно вычислимые функции – функции, вычислимые машинами Тьюринга.

Существуют различные модели вычислений:

1. *Интуитивная вычислимость* – вычислимость программами.
2. *Машина Тьюринга (МТ)* – $\langle Q, X, Y, \Gamma, \delta \rangle$ (известная модель)
3. ...

Но для нас важна не только вычислимость, но и сложность вычисления.

Перечислим важные моменты.

**Время вычисления.**

МТ – простое вычисление: время – количество тактов.

**Класс** $P$ (полином и экспонента).

Существуют функции, для которых не удается найти полиномиальный алгоритм.

**Класс** $NP$ – недетерминированное вычисление.

Задача распознавания эквивалентна задаче вычисления. Пример: Раскраска и $k$–раскраска.

$P = NP$?

*NP–полные проблемы.*

*Сводимость.*

**Задачи**. Разбиение; Гамильтонов цикл.

# 6 Лекция 5. Представление функций

Представление функций должно удовлетворять следующим требованиям:

1. Достаточно короткое и эффективное.
2. Позволяет манипулировать и вычислять функции.
3. Визуализация определенных свойств функции.
4. Подсказывать идеи для технологий реализации и т.д.

Нам известны некоторые представления: 1. *Таблицы истинности* (быстрое вычисление, быстрое выполнение би-нарных операций). Хотя сложность линейная относительно размера таблицы, но экспоненциально относительно количества переменных. Размерность таблицы экспоненциальна.

2. *ДНФ, КНФ (СДНФ, СКНФ), полином Жегалкина (без отрицаний)* – 2-х уровневые нормальные формы.

**Определение.** Представление называется

- *универсальным*, если для любой булевой функции существует представление такого типа;
- *совершенным*, если для любой булевой функции существует единственное представление такого типа.

ДНФ, КНФ – не являются совершенным представлением. Для совершенных представлений, чтобы определить эквивалентность функций достаточно проверить идентичность их представлений. Но с другой стороны, СДНФ (СКНФ) имеют большую длину: сложность ДНФ (длина ДНФ – количество входящих в нее литералов) для СДНФ максимальна среди всех эквивалентных ДНФ.

**Теорема 1.** $3-SAT$ $NP$-полна. (3-КНФ).
**Доказательство.** Если дизъюнкция содержит $k$ литералов.

1. $k=1 \Rightarrow$ добавляем две переменные $\{z, y_i^1, y_i^2\}, \{z, y_i^1, \bar{y}_i^2\}, \{z, \bar{y}_i^1, y_i^2\}, \{z, \bar{y}_i^1, \bar{y}_i^2\}$.
2. $k=2 \Rightarrow$ добавялем одну переменную $\{z_1, z_2, y_j\}$, $\{z_1, z_2, \bar{y}_j\}$
3. $k \geq 3 \Rightarrow \$\{z_1, z_2, y_i^1\}, \{\bar{y}_i^1, z_{i+2}, y_i^2\}, \ldots, \{\bar{y}_i^{k-3}, z_{k-1}, z_k\}$ $\square$

**Теорема 2.** $INEQU_{CNF}(\mathrm{C}_1, \mathrm{C}_1)$ $NP$-полна.
**Доказательство.** $3-SAT \leq INEQU$: $\mathrm{C}_1 = \mathrm{C}, \mathrm{C}_2 = 0$.$\square$

**Следствие.** Эквивалентность КНФ $coNP$-полна.

Операции $+, \cdot$ не всегда удобны. Например, ни равенство $f+g=h$ , ни $f \cdot g = h$ не может быть решено относительно $f$.

$\langle B, \oplus, \cdot \rangle$ — поле в алгебраическом смысле, таким образом, результаты теории алгебраических полей могут быть перенесены на $\langle B, \oplus, \cdot \rangle$.

**Теорема 3.** Полином Жегалкина – совершенное представлений булевых функций.

**Доказательство.** Любая ДНФ представима через полином Жегалкина, следовательно любая булева функция представима в виде полинома Жегалкина. С другой стороны, количество полиномов Жегалкина равно $2^{2^N}$, т.е. количеству всех булевых функций, что и доказывает теорему.□

**Теорема 4.** Функция $x_1 + x_2 + \cdots + x_n = \bigoplus_{I \subset \{1 \cdot n\}} {}_{\varnothing}\, m_I = P(x_1, \ldots, x_n)$, где $m_I = x_{i_1} \ldots x_{i_s}$ для $I = \{i_1, \ldots, i_s\}$

**Доказательство.** $P(0, \ldots, 0) = 0$.

Пусть в наборе ровно $l$ единиц, тогда в $P$ количество мономов, равных 1 и содержащих $k$ переменных, равно $\binom{l}{k}$, $k \in (1, l)$, следовательно, количество единичных мономов $\sum_{k=1}^{l} \binom{l}{k} = \sum_{k=0}^{n} \frac{l}{k} - \binom{l}{0} = 2^l - 1 \Rightarrow P(x_1, \cdot, c_n) = 1$.

**Следствие.** Функция $x_1 + \cdots + x_n$ - экспоненциально сложна в представлении через полином жегалкина.

3. **СФЭ.**

Переход от 2-х уровневой к многоуровневым формам позволяет найти более компактное представление булевых функций. Пусть $\Omega = \{w_i \in B_n, i \in I\}$ – множество базовых функций (называется *базисом*, если нельзя удалить ни одной функции из $\Omega$ без уменьшения замыкания $\Omega$).

**Определение.** $\Omega$-СФЭ $S$ от $n$ переменных — ациклический граф с вершинами 2-х типов:

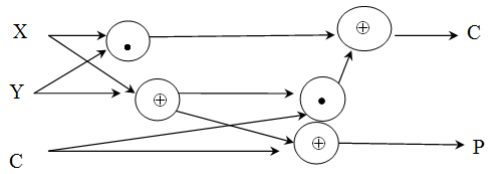
- *входные вершины* (без входящих дуг), помеченные 0, 1 или $x_i$, $i = 1, \ldots, n$;
- *функциональные вершины* – вершины, помеченные $w_i$, имеют $n_i$ входящих дуг: $w_i \in \Omega$, $i \in I$.

Каждая вершина $v$ представляет булеву функцию, соответствующую следующим индуктивным правилам:

1. Если $v$ помечена $0(1)$, то $f_v(x_1, \ldots, x_n) = 0(1)$;
2. если $v$ помечена $x_i$, то $f_v(x_1, \ldots, x_n) = x_i$;
3. если $v$ помечена $w_i$ и имеет $n_i$ входящих дуг, выходящих из $v_1 \ldots v_{n_i}$: $f_v(x_1, \ldots, x_n) = w_i(f_{v_1}(x), \ldots, f_{v_{n_i}}(x))$.

Обозначив $m$ узлов $v_1, \ldots, v_m$ как выходные узлы СФЭ, эта схема определяет функцию $f_s(x) = (f_{v_1}(x), \ldots, f_{v_m}(x))$.

**Пример.** Полный сумматор – хорошо известная базовая компонента в построении чипов. Эта схема вычисляет $x + y + c$ для 3 входных битов $x, y, c$(c – бит переноса): $\Omega = \{\oplus, \cdot, \bar{}\}$ – стандартный базис.




**Определение.** *Глубина СФЭ* – число функциональных уровней в СФЭ – соответствует временным затратам.

**Определение.** Базис *полон*, если Ω-СФЭ соответствует универсальному представлению, то есть любая булева формула может быть представлена в Ω-СФЭ.

**Пример.**

1. $\{+, \cdot, \bar{}\}$ — стандартный базис, полон (ДНФ, КНФ);
2. $\{\oplus, \cdot\}$ — полон;
3. $\{+, \cdot\}$ — неполон, так как он генерирует лишь монотонно возрастающие функции.

Даже для фиксированного базиса представление булевой функции неоднозначно. Нас интересует представление в СФЭ с небольшим количеством вершин.

**Определение.** Ω-СФЭ-сложность булевой функции $f$ — минимальное число вершин Ω-СФЭ, представляющей $f$.

ДНФ и КНФ — специальные формы СФЭ, поэтому СФЭ-сложность в стандартном базисе не более ДНФ-(КНФ-)сложности. С другой стороны, в большинстве случаев СФЭ-сложность менее ДНФ-(КНФ-) сложности.

СФЭ-модель по-прежнему имеет недостаток: неприемлемая времен-ная сложность проверки эквивалентности схем.

**Теорема.** Эквивалентность СФЭ над стандартным базисом со$NP$-полна.

**Доказательство.** Проблема $\in$ со$-NP$, $EQU_{DNF} \leq EQU_{\text{СФЭ}}$.□

4) **Формулы как СФЭ.**

В СФЭ экономия происходит, в частности, в связи с тем, что выход элемента может подаваться на вход нескольких элементов. То есть, вершины могут иметь несколько выходных дуг. Именно это свойство часто позволяет построить компактное представление в виде СФЭ. Но это приводит к трудности решения различных задач.

**Определение.** Для Ω-базиса Ω-*формула* — это Ω-СФЭ, степень исхода каждой вершины которой равна 1.

**Следствие.** Ω-СФЭ — формула $\Leftrightarrow$ она состоит из деревьев.

Очевидно, существует взаимно-однозначное соотношение между $\Omega$-формулами и булевыми формулами.

Анализ представления функции в виде $\Omega$-формулы намного проще, чем в виде СФЭ, например нижние оценки сложности конкретных функций. Но эквивалентность булевых формул со$NP$-полна.

# 7 Лекция 6. Бинарные диаграммы решений

5) **Бинарные диаграммы решений.**

**Определение.** Бинарное дерево решений (б.д.р.) —- дерево, в котором:

- внутренние вершины помечены переменными $x_i$ и имеют ровно 2 выходные дуги.
- листья помечены 0 и 1.

**Замечание.** Можно предположить, что каждая переменная читается не более одного раза.

**Пример.** $(x_1 \oplus x_2)(x_3 \oplus x_4)$

а) Полное дерево

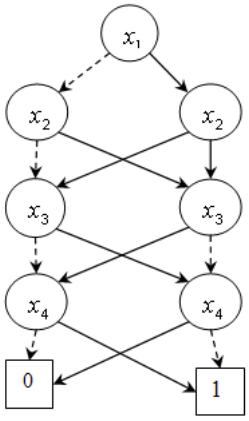
б) Неполное дерево

в) Изменение порядка чтения переменных существенно влияет на раз-мер б.д.р (порядок $x_1, x_2, x_3, x_4$)

**Определение.** *Бинарные диаграммы решений (Lee, 1959), ветвящиеся программы (branching programs) или бинарные программы (BP) (Mask, 1976)*: ориентированный ациклический граф с двумя стоками, помеченными 0 и 1, и внутренними вершинами, помеченными $x_i$ и имеющими две выходных дуги, помеченные 0 и 1.

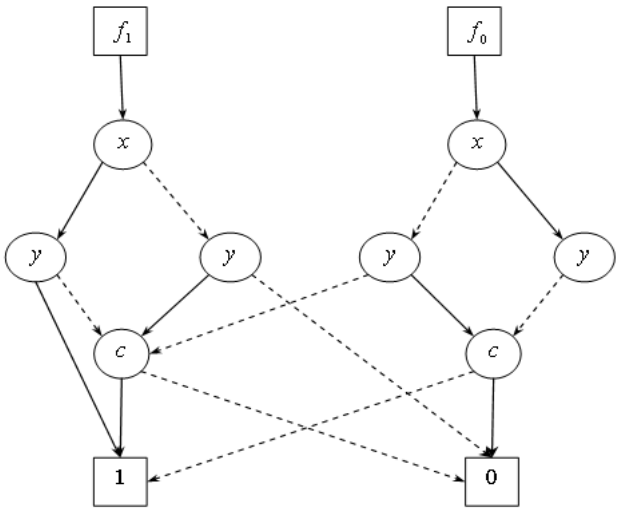
**Определение.** *Размер (сложность)* бинарной программы – количество её вершин.

**Пример.** $x_1 \oplus x_2 \oplus x_3 \oplus x_4$




В отличие от бинарных деревьев решений, в бинарных программах повторное чтение переменных может дать бинарную программу меньшего размера (невозможно удалить вершину, в которой переменная читается повторно, так как ее можно достичь различными путями).

Бинарные программы можно рассматривать и для представления функций с несколькими выходами из $B_{n,m}$. При этом выделяется $m$ начальных вершин.




**Проблема перевода одного представления булевой функции в другое.**

1. *Бинарная программа → ДНФ.*

   Строим моном для каждого пути. Затем рассмотрим дизъюнкцию мономов.

2. *Бинарная программа → Ω-СФЭ (Ω-полный базис).*

   Рассмотрим булеву функцию $sel(x, y, z) = \bar{x}y + xz$

   а) Заменим каждую вершину, помеченнную $x_i$, на *sel*-вершину с одним из входов, равным $x_i$, двигаясь от финальных вершин.

   б) Изменим направление всех дуг.

   в) Заменяем все *sel*-вершины на Ω-СФЭ.

# 8 Лекция 7. Ограниченные классы бинарных программ. Сложность проблем для бинарных программ

blank

# 9 Лекция 8. Требование к структурам данных в формальной верификации схем

Итак, существует несколько типов представления булевых функций:

1. Задание значений функции (таблица истинности);
2. Метод вычисления булевой функции (СФЭ);
3. Описание схемы определения значений функции (ВР).

Эти представления имеют существенные различия:

1. *Размер*: таблица истинности и ДНФ ($x_1 + x_2 + \cdots + x_n$), с другой стороны $x_1 + x_2 + \cdots + x_n$, представленная в виде полинома Жегалкина имеет слагаемыми всевозможные мономы;
2. *Ресурсы*, требующиеся для вычисления функции: например, время (Таблица истинности – сразу, формула - вычисление).
3. *Ресурсы*, требуемые для представления результата булевых операций над двумя функциями (f * g). (Для СФЭ быстро, для ВР1 — $NP$-сложно).
4. *Сложность определения свойств функции* (например, «функция – константа?»: просто для ВР1; $NP$-сложно для КНФ: $f \notin SAT \Leftrightarrow f = const, f(0, \ldots, 0) = 0$ ).
5. *Сложность определения фиктивности переменной* (полином Жегалкина: переменная существенна $\Leftrightarrow$ входит в полином Жегалкина, для КНФ — NP-сложно: $f \in SAT$ для новой переменной $x_0$, $f$ существенно зависит от $x_0$).

**Верификация схем**

Таким образом, невозможно найти идеальную форму представления булевых функций. Поэтому важно расставить приоритеты. Нас интересует построение схем. Рассмотрим две проблемы из области верификации схем. Эти проблемы очень важны, но кроме того, являются подпроблемами больших проблем.

Логические схемы: комбинационные схемы, схемы с элементами памяти.

Комбинационные схемы не имеют элементов памяти.

Хотя схемы с элементами памяти более функциональны, исследование комбинационных схем очень важно:

1. Сами по себе: как вычисление функции за один шаг.
2. Схемы с элементами памяти являются расширением комбинационных схем. При построении схем важна их функциональная корректность. Описание свойств схемы называется спецификацией. Разработка логической схемы сво-дится к итерационным шагам: спецификация – реализация: реализация на $i$-ом шаге задаёт спецификацию $i + 1$ шага.

Доказательство функциональной корректности осуществляется.

1. Оценка поведения спецификации и реализации на большом числе входящих векторов.

2. Формальная верификация (математическое доказательство).

3. Частичная верификация (математическое доказательство, что реализация удовлетворяет по крайней мере некоторым важным свойствам поведения спецификации): свойство надежности (определенные «плохие» события не могут произойти); свойство жизнеспособности (возможно, что определенные «хорошие» события произойдут).

**Формальная верификация комбинационных схем.**

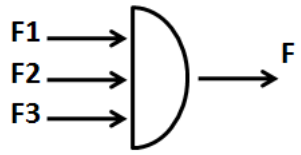
При моделировании схем рассматриваются сети логических элементов. Логический элемент – это элементарная схема, соединяющая транзисторы так, чтобы вычислялась определенная Булева операция на входах. Задача «логического синтеза» — оптимизировать схему соответственно определенным критериям: количество элементов, размер чипа, экономия энергии, задержки и т.п.

Для решения этих задач разрабатывались системы логического синтеза. Разрабатывались такие системы и в академических центрах:

- MINI (IBM Reslorch,1974)
- ESPRESO (Berkley,1984) (Калифорния)
- MIS (Berkley,1987)
- BOLD (Un Colorado at Boulder,1989)
- SIS (Berkley,1992).

**Основные требования к представлениям функций:**

1. т.к. в основном схемы – это сети логических элементов, то возникает необходимость представления функции, являющейся выходной для элемента на входы которого поданы другие булевы функции:

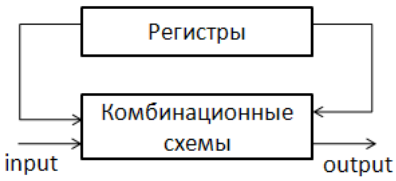



*Булевы операции должны осуществляться эффективно.*

2. Минимизация числа элементов: спецификация, реализация – эквивалентны ли они?

*Функциональная эквивалентность должна выполняться эффективно (проблема $SAT$, т.е. выполнимость схемы).*

3. Схемы с памятью.




Описание таких схем может быть осуществлено при помощи KDA. Эквивалентность KDA: Если состояние кодируется 80 битами, то получаем $2^80$ состояний. Время существования Вселенной $2^34$ лет $\approx 2^44$ часов $\approx 2^56$ секунд, следовательно, если даже обрабатывать 2 миллиона состояний в секунду, то не хватило бы всего времени существования Вселенной для анализа эквивалентности такого автомата.

Довольно долго оценивалась работа системы на большом множестве входов (1994, Intel Pentium — ненадежность проверки). Современная формальная верификация основана на эффективном представлении множества состояний, или характеристической функции множества состояний (т.е. булевой функции).

*Основные проблемы KDA (напр. проверка эквивалентности) могут решаться эффективно.*

# 10 Лекция 9. OBDD - эффективная структура данных

Начало изучения OBDD для VLSI-дизайна положили работы Брайнта (Bryant, 1986г.). Следующие свойства очень важны.

1. Редуцированная OBDD совершенное представление Булевой функции.
2. Манипулирование редуцированнами OBDD может выполняться эффективно.
3. Для многих практических интересных функций OBDD имеют маленький размер.

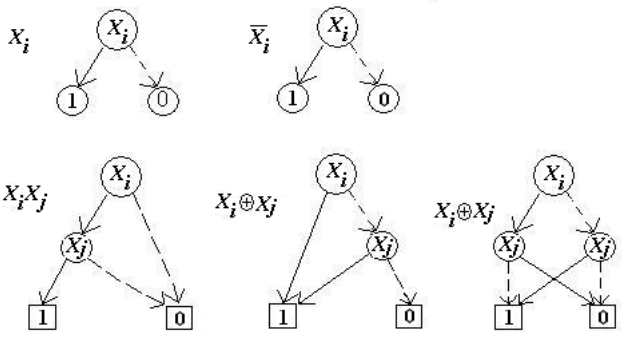
**Определения.**

**Определение 1.** Пусть $\pi$ порядок множества переменных $\{x_1, \ldots, x_n\}$:$\{x_{\pi(1)}, \ldots, x_{\pi(n)}\}$. *OBDD (Ordered Binary Decision Diagram — упорядоченная бинарная диаграмма решений) с порядком чтения переменных $\pi$ — это ациклический орграф, имеющим ровно 1 корень, 2 финальные вершины, не имеющие выходных дуг.* Эти финальные вершины помечены 0 и 1 (0-вершина, 1-вершина). Каждая нефинальная вершина помечена переменной $x_i$ и имеет 2 выходные дуги, помеченные 0 и 1. Порядок, в котором переменные встречаются на пути в графах соответствуют порядку $\pi$, т.е. если дуга ведет от вершины, помеченной $x_i$, к вершине, помеченной $x_j$, следовательно $\pi^1(i) < \pi^1(j)$.

**Определение 2.** OBDD *представляет* булеву функцию $f \in B_n$, если $\forall a \in B_n$ вычисленный путь на a, т.е. путь от корня до финальной вершины в соответствии с a, достигает вершину, помеченную $f(a)$.

Вершина OBDD с меткой $x_i$ определяет *разложение Шеннона:*
Если OBDD имеет корень, помеченный $x_i$, и представляет $f(x_1, \ldots, x_n)$, то $f = x_i f_{x_i} + \bar{x}_i f_{\bar{x}_i}$, где $f_{x_i} = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$; $f_{\bar{x}_i} = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$

**Пример.**




**Определение 3.** Пусть $P_1$, $P_2$ - OBDD. $P_1$ и $P_2$ - *изоморфны* ($P_1 \approx P_2$), если существует биекция $\phi$ между вершинами $P_1$ и $P_2$:

1. пометка $v$ и $\phi(v)$ совпадает;

2. для $a \in \{0, 1\}$, если $a$-дуга от $v$ ведет к $w$, то $a$-дуга от $\phi(v)$ ведёт к $\phi(w)$.

**Определение 4.** OBDD называется *редуцированной*, если

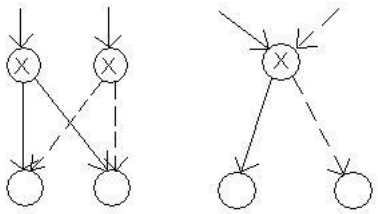
- не существует вершины $v$: 1-дуга и 0-дуга от $v$ ведет к одной и той же вершине $w$;

- не существует 2-х вершин $v$ и $w$: OBDD с корнями $v$ и $w$ - изоморфны: OBDD(v) $\approx$ OBDD(w).

**Определение 5.** *2 правила редукции OBDD*:

- *Правило «удаления»*: если 1-дуга и 0-дуга от вершины $v$ ведут к одной и той же вершине $W$, то можно перенаправить дуги, входящие в $v$, на вход $W$.

- *Правило «склеивания»*: если $u$ и $v$ помечены одной и той же переменной, и 1-дуги от $u$ и $v$ ведут к одной и той же вершине, и 0-дуги - - - - ? можно отожествить $u$ и $v$, удалив одну из этих вершин, перенаправив все ее входящие дуги к оставшейся вершине.



# 11 Лекция 10. Свойства редуцированных OBDD

**Теорема 1.** OBDD - редуцирована тогда и только тогда, когда неприменимо ни одно из правил редуцирования.

**Доказательство.** *Необходимость.* OBDD редуцирована, следовательно ни одно из правил не может быть применено.

*Достаточность.* Пусть не может быть применимо правило «удаления». Следовательно, 1 свойство редуцированных OBDD выполнено.

Пусть OBDD имеет 2 вершины $v$ и $w$ и OBDD($v$)≈OBDD($w$). Следовательно возможны два случая:

- 1-сын(u) = 1-сын($v$), 0-сын($u$) = 0-сын($v$), следовательно, может быть применимо правило «склеивания».
- Пусть $u'$ = 1-сын(u) ≠ 1-сын(v)= $v'$ или 0-сын(u) ≠ 0-сын(v).

Пусть 1 неравенство выполнено, следовательно $OBDD(u') \approx OBDD(v')$

Повторим процедуру с $u'$ и $v'$. На каждом шаге уменьшается множество переменных, которые могут встречаться в под-OBDD, поэтому не более, чем за $n$ шагов процесс приведет к возможности применить правило «склеивания», т.к. 1-вершина и 0-вершина одинаковы.□

Покажем, что любая булева функция имеет совершенное представление в виде OBDD при фиксированном порядке чтения переменных.

Далее для простоты будем рассматривать естественный порядок: $x_1, \ldots, x_n$. Рассмотрим $x_i$-вершину $v$.

На пути от корня до v читаются $x_j : j \leq i-1$.

Если вход $c_1, \ldots, c_n$ соответствует вычисленному пути, ведущему через $v$, следовательно $OBDD(v)$ вычисляет подфункцию $f_{x_1=c_1,\ldots,x_{i-1}=c_{i-1}}$.

**Теорема 2.** Пусть $S_i$ множество подфункций $f$, полученных при фиксированнии переменных $\{x_j, j \leq i-1\}$, для которых $x_i$ – существенна. Вплоть до изоморфизма существует единственная OBDD минимального размера для $f$ с порядком $x_1, \ldots, x_n$. Эта OBDD имеет $|S_i|$ вершин, помеченных $x_i$.

**Доказательство:**

1) Сначала построим минимальную OBDD $P$ для $f$, которая для любого $i$ содержит ровно $|S_i|$ вершин, помеченных $x_i$.

- Если $f$ константа, то $P$ является графом с 2-мя вершинами (обе финальные).
- Индукционное предположение: для любого $j \geq i+1$, для любого $g \in S_j$ в $P$ имеется ровно 1 вершина $x_j$-вершина $v$: $OBDD(v)$ вычисляет $g$.
- Индукционный переход: рассмотрим $i$. Пусть $h \in S_i$, следовательно, $h_{X_i}$ и $h_{\bar{X}_i}$ является константами или $S_j$ для некоторого $j \geq i+1$.

  Для $h$ введем вершину $v_h$. Пометим её $x_i$ и поправим 1-дугу в $h_{X_i}$, а 0-дугу в $h_{\bar{X}_i}$.

$P$ вычисляет $f$, что доказывается также индуктивно, т.к. $P$ правильно вычисляет подфункции: это верно для финальных вершин: $h = x_i h_{x_i} + \bar{x}_i h_{\bar{x}_i}$.

2) Минимальность $P$

Доказательство от противного. Если $P$ не минимальна, то существует $P'$, для которой существует $i : P'$ содержит $< |S_i|$ вершин, помеченных $x_i$.

Но существует $|S_i|$ различных подфункций $f$ вида $f_{X_1=C_1,\ldots,X_{i-1}=C_{i-1}}$, существенно зависящих от $x_i$.

Рассмотрим $Q = \{(c_1, \ldots, c_{i-1}) | f_{X_1=C_1,\ldots,X_{i-1}=C_{i-1}} \in S_i\}$. Т.к. $f_{X_1=C_1,\ldots,X_{i-1}=C_{i-1}}$ принадлежат $S_i$, следовательно, пути, помеченные $x_1 = c_1, \ldots, x_{i-1} = c_{i-1}$, ведут в вершины, помеченные $x_i$, следовательно, существуют $\bar{a}, \bar{b} \in Q : f_{\bar{a}} \neq f_{\bar{b}}$, но пути, помеченные $\bar{a}$ и $\bar{b}$ ведут в одну и ту же вершину, что приводит к противоречию, т.к. корень $OBDD$ соответствует лишь одной подфункции.

3) Минимальная OBDD изоморфна P.

Следуя предыдущим рассуждениям : любая минимальная $OBDD$ $R$ с порядком $x_1, \ldots, x_n$ содержит для любой подфункции $g \in S_i$ вершину, помеченную $x_i$ с сыновьями, $g_{X_i=1}$ и $g_{X_i=0}$, следовательно, любая $OBDD$ с тем же количеством вершин изоморфна $P$, а любая $OBDD$ $R$ неизоморфная $P$ имеет дополнительные вершины - следовательно, $R$ неминимальна.□

**Теорема 3**. Пусть $P$ – $OBDD$ булевой функции $f$ с порядком $\pi$, $P$ – изоморфна минимальный $P^|$ для $f$ с $\pi$. Тогда и только тогда когда ни одно из правил редуцирования не может быть применимо к P. Справедлива и обратная теорема.

Поясним смысл утверждения.


1) для минимальных $OBDD$ невозможно использовать правила редуцирования. 2) если $P$ - неминимальна, то можно принять одно из правил редуцирования.

**Доказательство.**

Пусть $f$ – не константа, а $\pi = (x_1, \ldots, x_n)$. Из теоремы 2 следует, что для любой $OBDD$ для $f$ для каждой подфункции из $S_i$ содержит хотя бы одну $x_i$-вершину. Если $P \neq P'$ - минимальная, то следовательно, существует $i \in \{1, \ldots\}$, $P$ содержит $> |S_i|$ $x_i$-вершин. Пусть $k$ – максимум из таких $i$, тогда в $P$ любая подфункция из $S$, $j > k$ и любая подфункция функции представляется в точности одной вершиной. Т.к. существует более $|S_k|$ $x_k$-вершин.

1. или существует $x_k$-вершина $u$, вычисляющая $g \notin S_k$, т.е. $g$ не зависит от $x_k$.

2. или существуют $x_k$-вершины $v$ и $w$, в которых вычисляется одна и та же функция $h \in S_k$.

В случае

1. $g = g_{x_k} = g_{\bar{x}_k}$ можно применить правило удаления для $u$, т.к $g_{x_k} = g_{\bar{x}_k}$.
2. сыновья $v$,$w$ вычисляет $h_{x_k}$ и $h_{\bar{x}_k}$ - следовательно, применим правило склеивания. Получаем противоречие. $\square$

**Следствие.** Для любого порядка чтения переменных (п.ч.п.) $\pi$ редуцированная $OBDD$ для любой булевой функции с п.ч.п. $\pi$ определяется однозначно (с точностью до изоморфизма).

# 12 Лекция 11. Алгоритм редукции

**Алгоритм редукции**

Идея алгоритма редукции ($\pi = (x_1, \ldots, x_n)$)

1. Перенумеруем все вершины OBDD.

2. Для всех $i := n, n-1, \ldots, 1$

2.1 Находим $V_i$ – множество $x_i$-вершин.

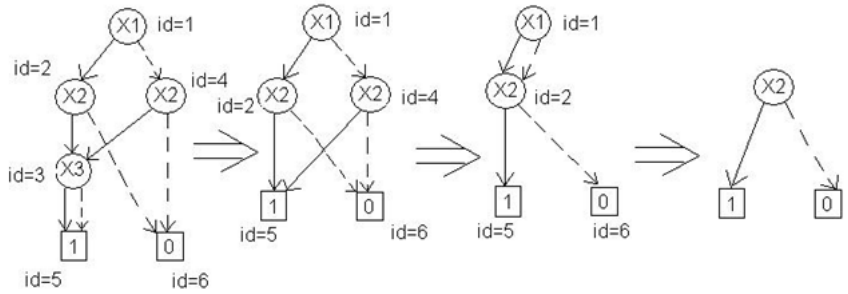
2.2 Для всех $v \in V_i$

2.2.1. Если $id$(0-сын($v$))=$id$(1-сын($v$)), то применяем к $v$ правило удаления, в противном случае $key(v) = (id$(0-сын($v$),1-сын ($v$))

2.3 Сортируем $key(v)$ для $V_i$ ($key(v_j) \leq key(v_{j+1}s)$)

2.4 Для всех $v_j \in V, j \geq 2$

2.4.1. Если $key(v_j) = key(v_{j-1})$ , то удаляем $v_{j-1}$, переводя все входящие в неё дуги в $v_j$




**Основные конструкции.**

Существует 2 основных метода построения редуцированной OBDD

1 метод:

а) Строим бинарное дерево решений;

б) Отождествляем финальные вершины, имеющие одинаковые пометки;

в) Применяем алгоритм редуцирования (недостаток : большой размер).

2 метод. Начинаем с корня:

а) Определяем существенна ли переменная $x_{i_1}$, если «да», то строим вершину с двумя отростками;

б) Для каждой новой вершины определяем подфункции, т.е. пометку и новую вершину (или отождествляем с какой-то старой): (недостаток: проверка существенности переменной и эквивалентности функций – сложны)

**Пример:**

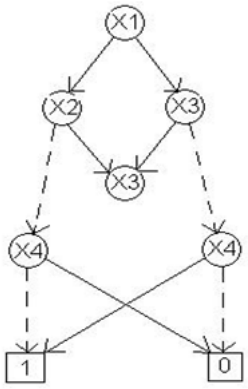
$f(x_1, x_2, x_3, x_4) = x_2(x_3 + \bar{x}_4) + \bar{x}_1\bar{x}_2x_4 + x_1\bar{x}_2\bar{x}_4$

$\pi = (1, 2, 3, 4)$ :

$f_{X1} = \bar{x}_2\bar{x}_4 + x_2(x_3 + \bar{x}_4)$

$f_{\bar{X}1} = \bar{x}_2\bar{x}_4 + x_2(x_3 + \bar{x}_4)$

$f_{X1X2} = f_{\bar{X}1X2} = x_3 + \bar{x}_4$
$f_{X1\bar{X}2} = \bar{x}_4$
$f_{\bar{X}1\bar{X}2} = x_4$
$f_{X1X2X3} = f_{\bar{X}1X2X3} = 1$
$f_{\bar{X}1\bar{X}2X3} = f_{\bar{X}1\bar{X}2X3} = 1$
$f_{\bar{X}1\bar{X}2X3} = f_{\bar{X}1\bar{X}2\bar{X}3} = x_4$
$f_{X1X2\bar{X}3} = f_{\bar{X}1X2\bar{X}3} = f_{X1\bar{X}2X3} = f_{X1\bar{X}2\bar{X}3} = \bar{x}_4$




Пусть * – булева операция: например, конъюнкция или дизъюнкция.
$f * g = x_i(f_{x_i=1} * g_{x_i=1}) + \bar{x}_i(f_{x_i=0} * g_{x_i=0})$
Рекурсивная конструкция:
Строим OBDD $P_1$ и $P_0$ для $(f_{x_i=1} * g_{x_i=1})$ и $(f_{x_i=0} * g_{x_i=0})$.
Вводим $x_i$-вершину, а-сын которой есть корень $P_a, a \in \{0, 1\}$.

Разложение $f$ и $g$ на подфункции связанно с необходимостью рассматривать $2^n$ подфункций. Выходом является рекурсивная процедура с движением по узлам $P_1$ и $P_0$. Каждому узлу новой $OBDD$ ставится в соотношение пара $(f, g)$, где $f$, $g$ – узлы $P_1$ и $P_0$ соответственно (пары образуют Table).

**Алгоритм:**

Oper(F,G,*) /* вход: OBDD F и G для f и g с п.ч.п. $\pi$ бинарная операция */
/* выход: OBDD для f ? g */
if (F и G – финальные вершины)
then Return (F*G)
else
if (F,G) $\in$ Table
then Return (F*G)
else /* - первая в $\pi$ существенная для $F$ или $G$ */
{Строим новую $x_i$-вершину $v$
0-сын(v)=oper $(F_{x_i=0}, G_{x_i=0}, *)$
1-сын(v)=oper $(F_{x_i=1}, G_{x_i=1}, *)$
InsertTable(F,G,v)
Return(v)}
Полученные $OBDD$ в общем случае не редуцированны.

**Теорема.** Пусть $f_1$ и $f_2$ представлены OBDD $P_1$ и $P_0$ соответственно, тогда для любой булевской операции $*$ редуцированная OBDD $P$ функции $f_1 * f_2$ может быть построена за время порядка $O(size(P_1) * size(P_0)$.

Без доказательства.

**Теорема.** Пусть булевы функции $f_1 * f_2$ представлены OBDD $P_1$ и $P_2$ соответственно, тогда тест на эквивалентность может быть выполнен за время $O(size(P_1) * size(P_2)$.

Для этого сначала редуцируем $P_1$ и $P_2$. Затем проверяется изоморфизм полученных $OBDD$ (переход в глубину по обеим $OBDD$ параллельно).

# 13 Лекция 12. Эффективная реализация OBDD

Хотя мы уже показали, что ряд операций над $OBDD$ могут быть выполнены эффективно для практических приложений было необходимо осуществлять эффективно по времени и памяти реализацию $OBDD$.

**Основные идеи.**

1. Центральной идеей является *представление отдельного узла* OBDD при помощи трёх полей:

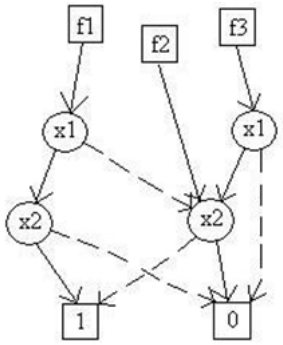
   - index (2 байта) – индекс $i$ переменной $x_i$;
   - 2-поля (по 1-му слову) – ссылки на сыновей узла.

   Таким образом количество переменных $\leq 65536$. «Слово» позволит получать ссылку на адреса полного виртуального адресного пространства (при 32–битной архитектуре слово = 4 байта, а полное адресное пространство имеет размерность $2^{32} \approx 4 \times 10^9 \approx 4Gb$ узлов. Для практической реализации может быть полезна дополнительная информация, связанная с каждым узлом. Но большие редуцированные OBDD требует осторожного увеличения хранимой информации. Компромисс между эффективностью введения дополнительной информации для любого узла и общим размером OBDD требует большой работы.

2. **Разделяемые (shared, «расшаренные») OBDD.**

   Некоторые функции могут быть представлены одним и тем же ациклическим орграфом с несколькими корнями. Такое представление назовём разделяемыми OBDD.

   **Пример.** $f_1 = (x_1 \equiv x_2), f_2 = \bar{x}_2, f_3 = x_1\bar{x}_2$




   В таких разделяемых OBDD можно добиться того, что каждая подфункция представляется не несколько раз, а лишь единожды. При этом представление OBDD будет не только совершенным, но и строго совершенным. При этом две эквивалентные функции $f$ и $g$ имеют не только единую редуцированную $OBDD$, но при реализации соответствуют ссылке на один и тот же элемент памяти, следовательно, проверка на эквивалентность требует лишь одной операции сравнения.