Методы синтеза комбинационных схем в базисе $\{\&, \vee, -\}$

Перязева Юлия Валерьевна
Доцент кафедры ВТ

Теория автоматов

Дискретные преобразователи без памяти

Алгоритм A_1, основанный на представлении булевых функций в виде СДНФ, СКНФ, ДНФ, КНФ

Содержание

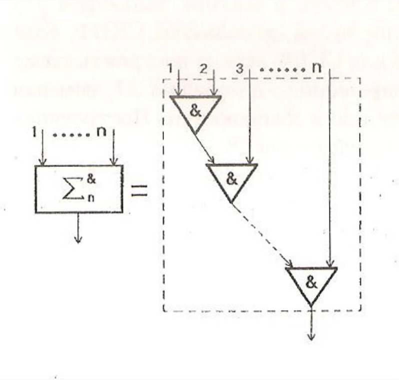
1. Алгоритм A_1, основанный на представлении булевых функций в виде СДНФ, СКНФ, ДНФ, КНФ

2. Алгоритм A_2, основанный на более компактной реализации множества всех полных элементарных конъюнкций

3. Алгоритм A_3, основанный на разложении Шеннона по остаточным функциям

Методы синтеза в базисе $\&, \vee, -$

Введем обозначение для многоместной конъюнкции:



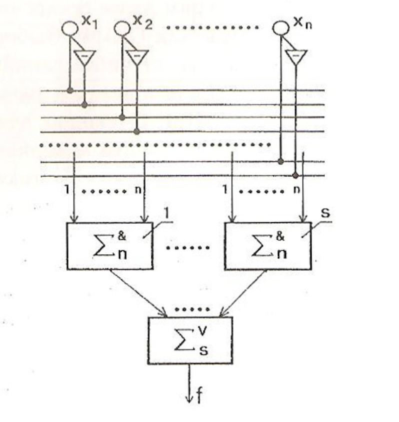
Видно, что сложность $\sum_n^{\&}$ равна $n-1$. Аналогично строится подсистема $\sum_n^{\vee}$.

Алгоритм A_1

Рассмотрим представление в виде СДНФ:

$$f(x_1, \cdots, x_n) = \sum_{f(\sigma_1,\ldots,\sigma_n)=1} x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot \ldots \cdot x_n^{\sigma_n}$$

Реализуя эту конструкцию, получим следующую схему:



Алгоритм A_1

Сложность данной схемы равна $n + s(n-1) + (s-1)$, n – число аргументов у функции, s – число полных элементарных конъюнкций. Сложность наибольшей по сложности схемы:

$$L_{A_1}(n) \leq n + ns - 1, \tag{1}$$

где $L_{A_1}(n)$ – сложность самой сложной схемы среди всех схем с n входами, полученных алгоритмом A_1.

Алгоритм A_1

Для СКНФ нужно поменять местами многоместные конъюнкции и дизъюнкции.

Если в векторе меньше нулей, используем СКНФ, если единиц – СДНФ.

Алгоритм A_1

Для произвольной булевой функции от n аргументов можно выбрать СДНФ или СКНФ, где число элементарных конъюнкций или элементарных дизъюнкций не превышает 2^{n-1}, т.к. худший случай, когда 0 и 1 поровну, следовательно $s \leq 2^{n-1}$, и, из (1), получим:

$$L_{A_1}(n) < n + ns < n + n\frac{2^n}{2}$$

Алгоритм A_1. Пример

Построим алгоритмом A_1 комбинационную схему над базисом $B_0 = \{\&, \vee, -\}$ для функции $f(x_1x_2x_3) = (1001\ 0101)$. Найдем сложность L.

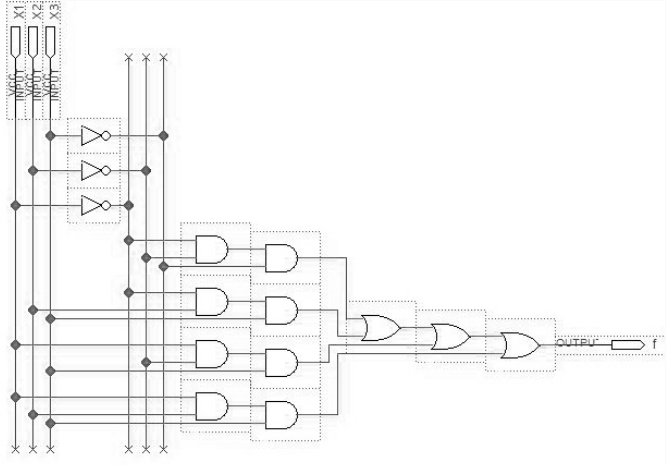
Найдем СДНФ для функции f:

$x_1x_2x_3$	f	
0 0 0	1	$\overline{x}_1\overline{x}_2\overline{x}_3$
0 0 1	0	
0 1 0	0	
0 1 1	1	$\overline{x}_1x_2x_3$
1 0 0	0	
1 0 1	1	$x_1\overline{x}_2x_3$
1 1 0	0	
1 1 1	1	$x_1x_2x_3$

$$f(x_1x_2x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 \vee \overline{x}_1x_2x_3 \vee x_1\overline{x}_2x_3 \vee x_1x_2x_3$$

Представим формулу в виде схемы, сложность схемы: $L = 14$.

$f = \overline{x}_1\overline{x}_2\overline{x}_3 \vee \overline{x}_1 x_2 x_3 \vee x_1\overline{x}_2 x_3 \vee x_1 x_2 x_3$



Алгоритм A_1. Пример

Построим минимальную ДНФ для функции $f(x_1x_2x_3) = (1001\ 0101)$.

Поместим функцию f в матрицу и найдем минимальное покрытие.

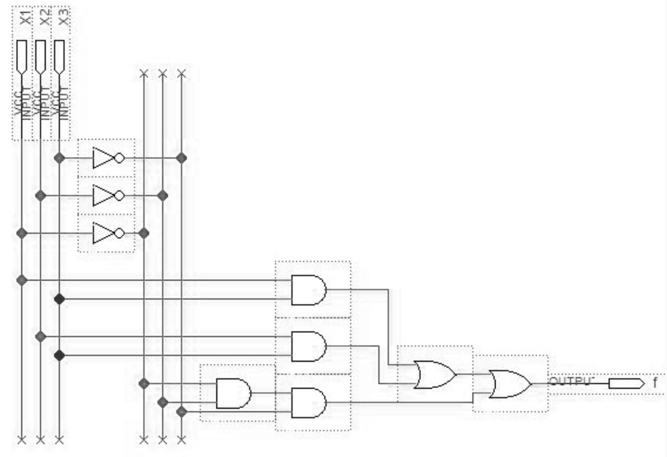
x_2	0	0	1	1
x_3 / x_1	0	1	1	0
0	1	0	1	0
1	0	1	1	0

Таким образом, $f(x_1x_2x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 \vee x_2x_3 \vee x_1x_3$.

Алгоритм A_1. Пример

Представим формулу в виде схемы, сложность схемы $L = 9$.

$f(x_1x_2x_3) = x_1x_3 \vee x_2x_3 \vee \overline{x}_1\overline{x}_2\overline{x}_3$



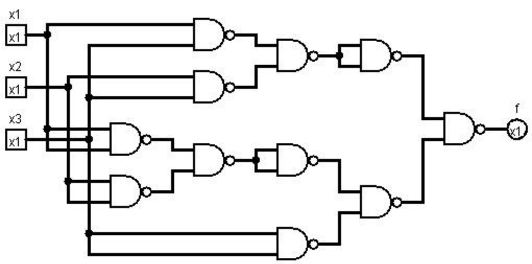
Переход к NAND элементам. Строим минимальную ДНФ и применяем следующие тождества:

$$\overline{x} = \overline{x \cdot x}$$

$$x \vee y = \overline{\overline{x} \cdot \overline{y}}$$

$$x \cdot y = \overline{\overline{x \cdot y}}$$

$f(x_1x_2x_3) = x_1x_3 \vee x_2x_3 \vee \overline{x}_1\overline{x}_2\overline{x}_3 = \overline{\overline{x_1x_3} \cdot \overline{x_2x_3} \cdot \overline{\overline{x}_1\overline{x}_2\overline{x}_3}} =$
$\overline{\overline{\overline{\overline{x_1x_3} \cdot \overline{x_2x_3}}} \cdot \overline{\overline{\overline{\overline{x_1x_1}\,\overline{x_2x_2}}}\,\overline{x_3x_3}}}$



Алгоритм A_2, основанный на более компактной реализации множества всех полных элементарных конъюнкций

Содержание

1. Алгоритм A_1, основанный на представлении булевых функций в виде СДНФ, СКНФ, ДНФ, КНФ

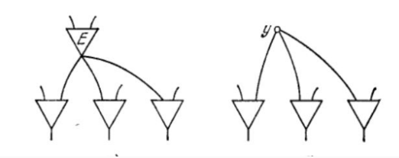
2. Алгоритм A_2, основанный на более компактной реализации множества всех полных элементарных конъюнкций

3. Алгоритм A_3, основанный на разложении Шеннона по остаточным функциям

Алгоритм A_2

Использование ветвлений позволяет, как правило, значительно уменьшить сложность схем.

Пусть в схеме для функции $f(\tilde{x}) = f(x_1, \ldots, x_n)$ выход элемента E подается на входы нескольких элементов, и на нем реализуется функция $h(\tilde{x}^{'})$, где $\tilde{x}^{'}$ – некоторое подмножество $\tilde{x}$. Заменим в схеме элемент E на новый полюс y.



Функцию реализуемую на выходе обозначим через $g(\tilde{x}^{''}, y)$. Если на полюс y подать $h(\tilde{x}^{'})$, то будет реализована $f(\tilde{x}) = g(\tilde{x}^{''}, h(\tilde{x}^{'}))$. Указанное представление получено на основе схемы, однако обычно требуется обратное, найти декомпозицию (разложение).

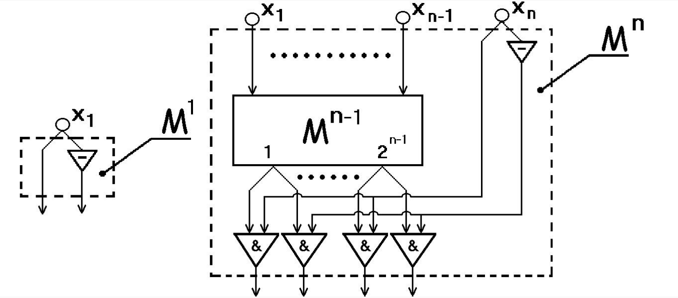
Алгоритм A_2

Если строить не каждый минтерм для СДНФ в отдельности, а всё их множество сразу, то можно значительно уменьшить сложность схемы, находя ветвления при построении.

Для этого в алгоритме A_2 используется индуктивное построение, на каждом шагу добавляется по следующей переменной в элементрные конъюнкции, получая минтермы.

Алгоритм A_2

Для n индуктивное построение схемы, которая реализует на своих выходах все 2^n минтерма, будет выглядеть следующим образом:

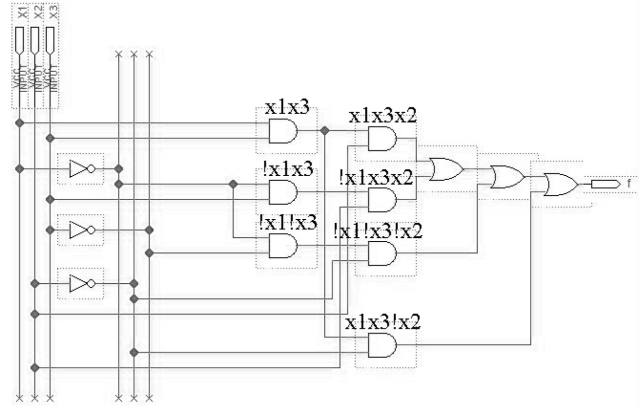


Далее минтермы соединяются дизъюнкцией, как и в предыдущем алгоритме.

Алгоритм A_2. Пример

Рассмотрим на примере все той же функции $f(x_1x_2x_3) = (1001\ 0101)$, $f(x_1x_2x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 \vee \overline{x}_1x_2x_3 \vee x_1\overline{x}_2x_3 \vee x_1x_2x_3$.

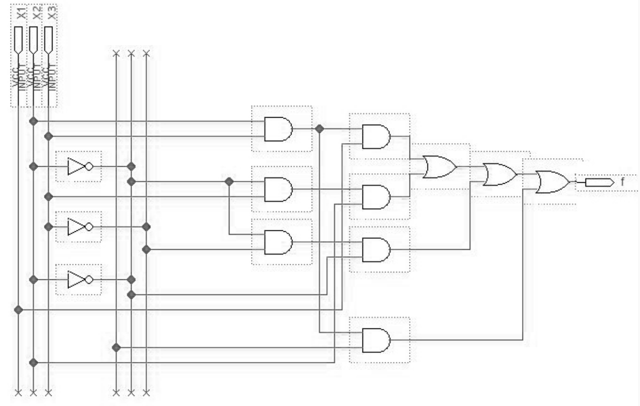
При построении важно в каком порядке добавляются переменные, при разном порядке получаем разные сложности! Рассмотрим порядок x_1, x_3, x_2. L=13.



Алгоритм A_2. Пример

$f(x_1x_2x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 \vee \overline{x}_1x_2x_3 \vee x_1\overline{x}_2x_3 \vee x_1x_2x_3.$

Рассмотрим порядок x_2, x_3, x_1. Опять получаем схему сложности 13.



Минимальный вариант по такому построению можно найти полным перебором.

Алгоритм A_2

Оценим теперь сложность схем, которые получаются данным алгоритмом. Видно, что

$$L(\mathrm{M}^1) = 1,$$

$$L(\mathrm{M}^n) = L(\mathrm{M}^{n-1}) + 2^n + 1.$$

Раскрывая последнее рекуррентное соотношение до M^1, получим:

$$L(\mathrm{M}^n) = L(\mathrm{M}^1) + 2^2 + 1 + 2^3 + 1 + \cdots + 2^n + 1 = n + 2^{n+1} - 4.$$

Так как s не превосходит 2^n, то $L(\Sigma_s^\vee)$ не превосходит $2^n - 1$. Таким образом, в итоге получим

$$L_{A2}(n) \leq 3 \cdot 2^n + n - 5.$$

Отметим, что все приведенные рассуждения, сделанные для СДНФ, можно по аналогии сделать и для СКНФ.

Алгоритм A_3, основанный на разложении Шеннона по остаточным функциям

Содержание

1. Алгоритм A_1, основанный на представлении булевых функций в виде СДНФ, СКНФ, ДНФ, КНФ

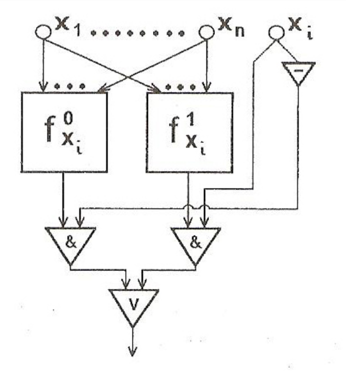
2. Алгоритм A_2, основанный на более компактной реализации множества всех полных элементарных конъюнкций

3. Алгоритм A_3, основанный на разложении Шеннона по остаточным функциям

Алгоритм A_3

Рассмотрим разложение Шеннона: $f = \overline{x_i} f^0_{x_i} \vee x_i f^1_{x_i}$.

Для реализации схемы нужно воспользоваться следующей индуктивной процедурой:



С помощью данного метода можно строить схему без получения формулы.

Алгоритм A_3

Каждую остаточную, в свою очередь можно разложить тем же способом:

$$f = \overline{x_i x_j} f^{0\ 0}_{x_i x_j} \vee \overline{x_i} x_j f^{0\ 1}_{x_i x_j} \vee x_i \overline{x_j} f^{1\ 0}_{x_i x_j} \vee x_i x_j f^{1\ 1}_{x_i x_j}$$

Когда все остаточные станут размерности 2, воспользуемся готовыми представлениями:

$(0000) = x \cdot \overline{x}$	$(1000) = \overline{x \vee y}$
$(0001) = x \cdot y$	$(1001) = \overline{x \vee y} \vee xy$
$(0010) = x \cdot \overline{y}$	$(1010) = \overline{y}$
$(0011) = x$	$(1011) = x \vee \overline{y}$
$(0100) = \overline{x} \cdot y$	$(1100) = \overline{x}$
$(0101) = y$	$(1101) = \overline{x} \vee y$
$(0110) = \overline{x \cdot y} \cdot (x \vee y)$	$(1110) = \overline{x \cdot y}$
$(0111) = x \vee y$	$(1111) = x \vee \overline{x}$

Алгоритм A_3

Сложность бинарной булевой функции в рассматриваемом базисе не превосходит 4. Оценим сложность схемы, получаемой данным алгоритмом.

$$L(n) \leq 2 \cdot L(n-1) + 4 \leq 2 \cdot (2 \cdot L(n-2) + 4) + 4 \leq \ldots =$$
$$2^{n-2} \cdot L(2) + 2^{n-3} \cdot 4 + \cdots + 2^0 \cdot 4 = 2 \cdot 2^n - 4$$

Таким образом:

$$L(n) \sim 2 \cdot 2^n$$

Алгоритм A_3. Пример

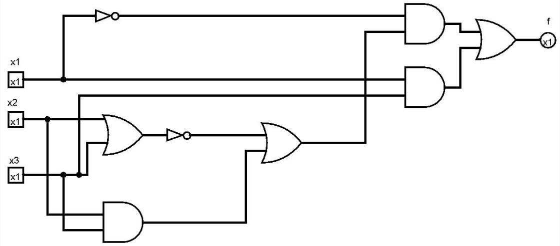
Рассмотрим на примере все той же функции $f(x_1x_2x_3) = (1001\,0101)$. Как и для предыдущего алгоритма важен порядок переменных в разложении, и минимальный вариант можно найти полным перебором.

Рассмотрим переменные в порядке x_1, x_2, x_3.

$f(x_1x_2x_3) = \overline{x_1}f^0_{x_1} \vee x_1f^1_{x_1} = \overline{x_1} \cdot (\overline{x_2 \vee x_3} \vee x_2x_3) \vee x_1x_3$

$f^0_{x_1} = (1001) = \overline{x_2 \vee x_3} \vee x_2x_3$ $\qquad$ $f^1_{x_1} = (0101) = x_3$

Получили схему сложности 8:



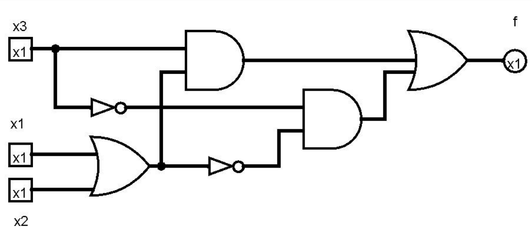
$f(x_1x_2x_3) = (1001\ 0101)$.

Рассмотрим переменные в порядке x_3, x_1, x_2.

$f(x_1x_2x_3) = \overline{x_3} f^0_{x_3} \vee x_3 f^1_{x_3} = \overline{x_3} \cdot \overline{x_1 \vee x_2} \vee x_3 \cdot (x_1 \vee x_2)$

$f^0_{x_3}(x_1x_2) = (1000) = \overline{x_1 \vee x_2}$ $\qquad f^1_{x_3} = (0111) = x_1 \vee x_2$

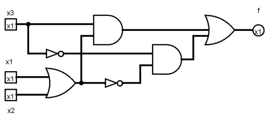
Получили схему сложности 6:



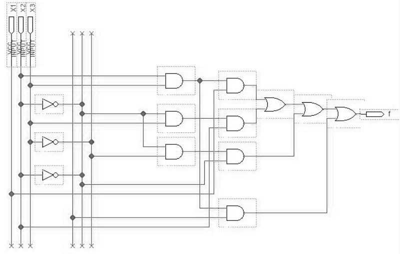
Алгоритмы A_1, A_2, A_3

$f(x_1x_2x_3) = (1001\ 0101)$ $B = \{\&, \vee, -\}$

Алгоритм A_3, сложность $L = 6$:



Алгоритм A_2, сложность $L = 13$:



Алгоритм A_1, сложность $L = 14$: