ДЕКОМПОЗИЦИЯ БУЛЕВЫХ ФУНКЦИЙ МЕТОДОМ РАСШИРЕНИЯ

Баркалов А.А., Красичков А.А.
University Zelenogursky Institution of Informatics and Electronics, Poland
A.Barkalov@iie.uz.zgora.pl
Кафедра ЭВМ, ДонНТУ
krasich@cs.dgtu.donetsk.ua

Abstract

Barkalov A.A., Krasichkov A.A. Decomposition of Boolean functions by a method of expansion. *The method of decomposition of Boolean functions by function expansion is proposed. Method is based on the artificial increase of Boolean function variables and its subsequent minimization. The method is illustrated by example of Boolean function realization in FPGA basis.*

1. Введение

Функциональный базис ПЛИС с архитектурой FPGA и FLEX подразумевает множество методов для реализации в нем систем булевых функций (БФ) [1]. Такие методы основаны на декомпозиции булевых функций по аргументам. В настоящее время известны традиционные методы декомпозиции, являющиеся классическими, а также методы, представляющие их комбинации. Они представляют чисто академический интерес и непригодны для реализации систем булевых функций с большим числом аргументов. Более производительные алгоритмы функционального синтеза являются ноу-хау зарубежных фирм-производителей ПЛИС и заложены в соответствующие САПР [2]. Поэтому актуальной задачей является создание продуктивных методов реализации систем булевых функций на ПЛИС с архитектурой FPGA и FLEX. Главными критериями методов являются уменьшение аппаратурных затрат при реализации и приемлемое время нахождения решения. В настоящей работе предлагается новый метод декомпозиции, в основе которого лежит искусственное расширение БФ по аргументам.

2. Основные положения и методы декомпозиции

Функциональный базис ПЛИС FPGA и FLEX представляет собой множество идентичных логических блоков (ЛБ), которые реализуют любую БФ с числом аргументов $R = \overline{2,5}$ и могут произвольным образом соединяться друг с другом [3]. Функция каждого ЛБ и их соединение в

логическую схему программируются независимо при проектировании. Поэтому, для реализации системы БФ в данном базисе необходимо представить каждую функцию в виде совокупности подфункций с меньшим числом аргументов. Такое преобразование называется функциональной декомпозицией и может быть выполнено множеством методов [4].

Базовым является метод декомпозиции Рота-Карпа (Roth-Karp) [4] (Рис.1).

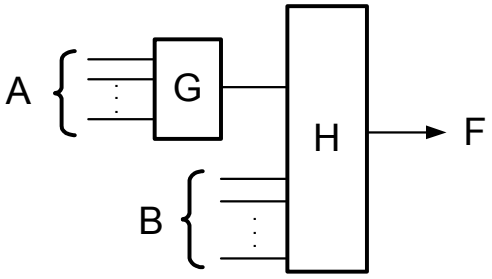


Рис. 1. Представление функции F методом Рота-Карпа

Для такой реализации функция F должна быть представлена в виде:

$$F = H[G(A_0, A_1, ..., A_M), B_0, B_1, ..., B_N], \quad (1)$$

где множество аргументов A называется связанным (bound), а множество аргументов B – свободным (free). Если множества A и B не имеют общих аргументов, то декомпозицию называют непересекающейся (disjoint), в обратном случае – пересекающейся (non disjoint).

Для нахождения функций G и H выполняется разложение Шеннона [1] по аргументам множества A:

$$F = H_0(B_0, B_1, ..., B_M) \,\&\, \overline{A}_0 \,\&\, \overline{A}_1 \,\&\, ... \,\&\, \overline{A}_N \vee H_1(B_0, B_1, ..., B_M) \,\&\, \overline{A}_0 \,\&\, \overline{A}_1 \,\&\, ... \,\&\, A_N \vee ...$$
$$... \vee H_K(B_0, B_1, ..., B_M) \,\&\, A_0 \,\&\, A_1 \,\&\, ... \,\&\, A_N,$$

где $K = 2^N$.

Если среди всех K подфункций H_i существует более, чем две различные, то декомпозиция для заданных множеств аргументов A и B невозможна.

Для количественной оценки эффективности метода выведем зависимость числа реализуемых функций для фиксированных множеств A и B.

Пусть функция F реализуется в виде (1) и все K ее подфункций H_i равны. Тогда получаем частный случай реализации: $F = H(B_0, B_1, ..., B_M)$. Число V всех различных функций такого вида равняется:

$$V = 2^{2^M}. \quad (2)$$

Пусть функция F реализуется в виде (1) и среди K подфункций H_i встречается не более двух разных. Число перестановок всех K подфункций H_i в этом случае составляет $2^K - 2$. Это все возможные комбинации, за исключением двух частных случаев, когда все K

подфункций равны одной или другой из двух существующих различных подфункций H_i. При этом число различных пар подфункций H_i определяется как число сочетаний C_V^2. Очевидно, что в этом случае существует $(2^K - 2) \cdot C_V^2$ реализуемых функций.

Таким образом, среди N_1 функций (M+N) аргументов ($N_1 = 2^{2^{M+N}}$) в виде (1) реализуется N_2 функций:

$$N_2 = (2^K - 2) \cdot C_V^2 + V = (2^{2^N} - 2) \cdot C_{2^{2^M}}^2 + 2^{2^M}. \quad (3)$$

Вероятность P реализации БФ вычисляется следующим образом:

$$P = \frac{N_1}{N_2} = \frac{(2^{2^N} - 2) \cdot C_{2^{2^M}}^2 + 2^{2^M}}{2^{2^{M+N}}}. \quad (4)$$

Для нахождения возможного представления БФ F в виде (1) необходимо последовательно перебирать все различные комбинации разбиения аргументов на множества A и B и выполнять разложение Шеннона. Общее число всех таких комбинаций равно $L = C_{M+N}^M = \frac{(M+N)!}{M! \cdot N!}$ и достаточно велико для последовательного перебора. Вероятность существования функций G и H при каждой комбинации вычисляется по формуле (4) и значительно уменьшается с увеличением числа аргументов множеств A и B.

Так, при M=1, N=2 (функция трех аргументов, реализуемая на ЛБ с R=2) N_1=88, а L=3. То есть, при каждом из трех различных разбиений аргументов на множества A и B существует 88 различных функций, представляемых в виде (1). Однако, как показали проведенные исследования, общее число функций составляет 152, а не 88×3. Это объясняется тем, что в L случаях некоторые функции повторяются, то есть реализуются при нескольких разбиениях аргументов на множества A и B.

При M=4, N=4 (начало реализации на ЛБ с R=4) $N_1 = 1.41 \cdot 10^{14} \approx 2^{47}$, а L=70, тогда как $N_2 = 2^{2^{M+N}} = 2^{2^8} = 2^{256}$.
Следовательно, вероятность реализации такой функции равняется $2^{47} / 2^{256} = 2^{-209}$ – практически нулю. При пересекающейся декомпозиции вероятность реализации функции выше, однако, как показали исследования, незначительно.

Очевидно, что данный метод неоптимален с точки зрения быстродействия, а главное, мощности множества реализуемых функций.

Модификацией данного метода является PUB-декомпозиция [4], представление реализации которой приведено на рис.2.

При помощи данного представления можно реализовать любую БФ, однако число переборов L определяется также, как и в предыдущем методе. Кроме того, в большинстве случаев число функций G достаточно велико, и число аргументов функции H не всегда приемлемо

для реализации на ЛБ. В любом случае такая реализация обладает избыточностью, так как множество функций *G* зависит от одних и тех же аргументов.

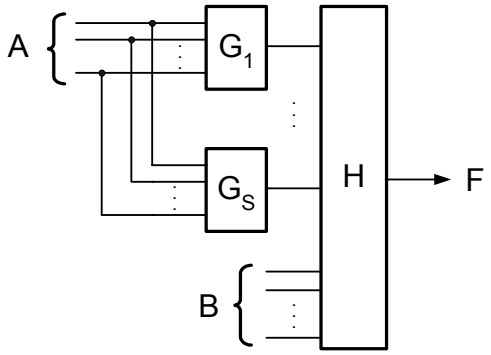


Рис. 2. PUB-декомпозиция функции F

Другой модификацией метода Рота-Карпа является декомпозиция Куртиса-Ашенхерста (Curtis-Ashenhurst) [4] (Рис.3).

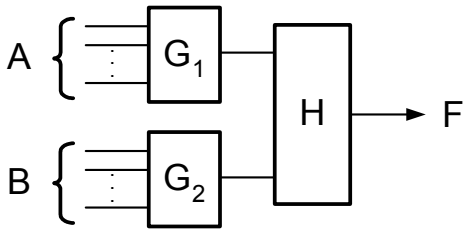


Рис. 3. Декомпозиция Куртиса-Ашенхерста

Необходимым условием применимости метода является то, что множества *A* и *B* пересекаются, причем, обычно множество *A* либо *B* включает в себя все аргументы функции *F*. В зависимости от базовых реализаций функции *H* различают AND-, OR-, EXOR-декомпозиции Куртиса-Ашенхерста.

Данным методом может быть представлена любая БФ, однако он обладает максимальной избыточностью реализации и требует оптимизации, а поиск любой существующей реализации может быть выполнен множеством способов и при этом достаточно сложен [4].

Исходя из недостатков рассмотренных методов, предлагается следующий метод декомпозиции, ориентированный на конкретную реализацию БФ на ЛБ.

3. Декомпозиция методом расширения

Исходными данными для декомпозиции служит каскадная структура неопределенных ЛБ, которая включает в себя минимальное число ЛБ, необходимых для реализации любой БФ с заданным числом

аргументов. Далее последовательно определяются логические функции каждого блока, до тех пор, пока все ЛБ не будут определены. В процессе определения функции каждого ЛБ, остальные блоки могут быть исключены из структуры. Тем самым, одновременно выполняется и оптимизация логической схемы. Найденная реализация является оптимальной с точки зрения аппаратурных затрат.

Определение функций ЛБ начинается с блока, входы которого реализуют некоторую промежуточную функцию нескольких аргументов исходной БФ F и заканчивается определением функции ЛБ, непосредственным выходом которого является выход F. Процесс является итерационным и включает в себя следующие этапы.

1. Определение аргументов функции очередного ЛБ G_i.
2. Расширение БФ на аргумент G_i.
3. Выбор области определения функции G_i.
4. Определение аргументов оставшейся части логической схемы.
5. Минимизация оставшейся части логической схемы.

В том случае, если на каком-либо этапе декомпозиции, минимизация невозможна, необходимо вернуться к предыдущему ЛБ и повторить пункт 3 для другой области определения функции G_i.

4. Пример применения метода

Авторами проведены исследования реализации всех функций трех аргументов при реализации на ЛБ с R=2. Установлено, что 104 функции реализуются в виде структуры ЛБ, приведенной на рис.4,*а*. Независимо от этого 232 функции из 256-ти реализуются структурой, приведенной на рис.4,*б*, а 24 оставшиеся функции можно реализовать только структурой, приведенной на рис.4,*в*.

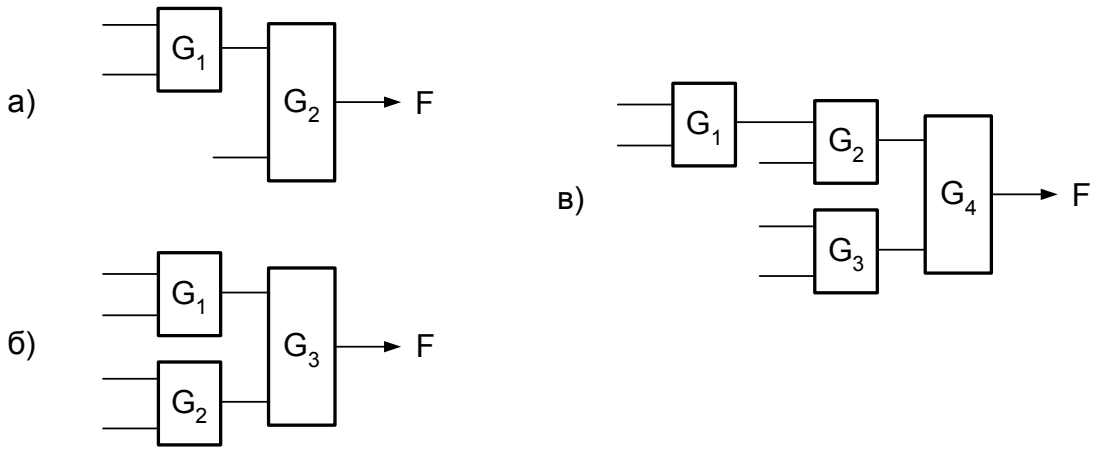


Рис. 4. Варианты реализации БФ трех аргументов на ЛБ с R=2

Рассмотрим примеры применения метода при реализации структур, приведенных на рис.4.

Пример 1. Пусть БФ F_1 задана ДНФ: $F_1 = \bar{a}\bar{c} \vee \bar{a}bc \vee a\bar{b}\bar{c} \vee ab\bar{c}$ и ее необходимо реализовать на ЛБ с R=2. Исходная структура ЛБ приведена на рис.4*в*.

1. Определение аргументов функции G_1. Аргументы выбираются произвольно.

Пусть $G_1 = G_1(a,b)$.

2. Расширение БФ на аргумент G_1. Поскольку функция F_1 не должна фактически зависеть от G_i, то функция должна принимать одни и те же значения на соответствующих наборах a,b,c как при $G_1=0$, так и при $G_1=1$. Таблица истинности БФ F_1 приведена в табл.1.

Таблица 1

Таблица истинности БФ

G_1	a	b	c	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Таблица 2

Отмеченная таблица истинности

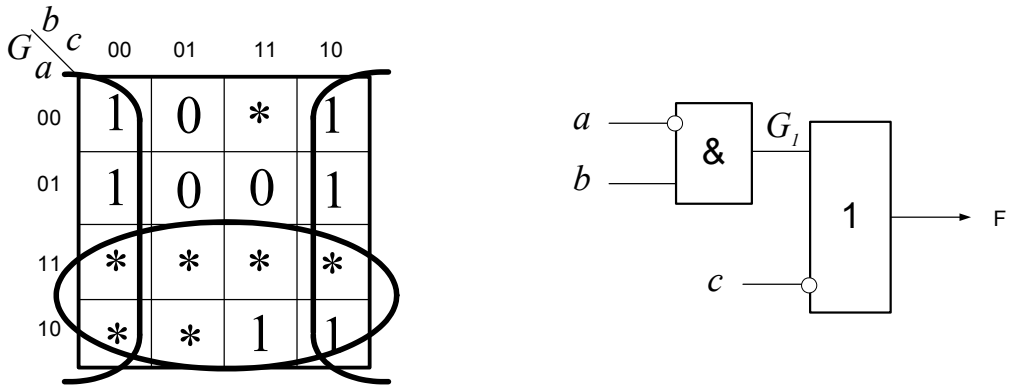
G_1	a	b	c	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	*
0	0	1	1	*
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	*
1	0	0	1	*
1	0	1	0	1
1	0	1	1	1
1	1	0	0	*
1	1	0	1	*
1	1	1	0	*
1	1	1	1	*

3. Выбор области определения функции G_1. Таблица истинности разделена на две части ($G_1=0$ и $G_1=1$). Поскольку функция фактически должна зависеть от трех аргументов, то половину значений функции F_1 можно отметить неопределенностями (*). Однако должно выполняться следующее правило. Неопределенности должны ставиться на одном из наборов аргументов – либо в верхней половине таблицы, либо в нижней. При этом набор неопределенностей должен задаваться только на

аргументах, определенных в п.1. Отметим неопределенностями верхнюю половину таблицы на наборе $\bar{a}b$, а нижнюю, соответственно на наборах $\bar{a}\bar{b}$, ab и $a\bar{b}$. Отмеченная таблица истинности БФ F_1 приведена в табл.2 Область определения функции G_1 соответствует наборам, не отмеченным неопределенностями в нижней части таблицы, то есть там, где G_1=1. Таким образом, $G_1 = \bar{a}b$.

4. Определение аргументов оставшейся части логической схемы. Пусть F_1= =$F_1(G_1,b,c)$. То есть, оставшаяся, пока неопределенная часть ЛБ реализует функцию, не зависящую непосредственно от аргумента a.

5. Минимизация оставшейся части логической схемы. Нагляднее всего в данном случае проводить минимизацию при помощи карты Карно (рис. 5, а). По условию п.4, склеивать можно только такие группы единиц, чтобы аргумент a склеивался и не присутствовал в минимальной формуле. Получаем $F_1 = G_1 \vee \bar{c}$. Поскольку данная часть функции может быть реализована на блоке G_2, то реализация функции на этом закончена (Рис.6,*б*) а ЛБ G_3 и G_4 исключаются из схемы.



а) *б*)

Рис. 5. Минимизация и реализация БФ $F_1=F_1(G_1,b,c)$

Пример 2. Пусть БФ F_2 задана ДНФ: $F_2 = \bar{a}\bar{c} \vee a\bar{b}\bar{c} \vee a\bar{b}c$ и ее необходимо реализовать на ЛБ с R=2.

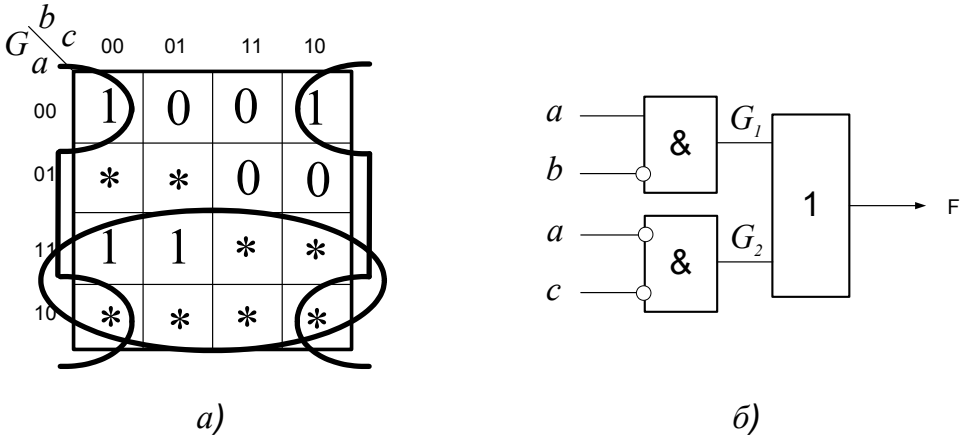
1. Определение аргументов функции G_1. Пусть $G_1 = G_1(a,b)$.

2. Расширение БФ на аргумент G_1 выполняется аналогично первому примеру.

3. Выбор области определения функции G_1. Отметим неопределенностями верхнюю половину таблицы истинности на наборе $a\bar{b}$, а нижнюю, соответственно на наборах $\bar{a}\bar{b}$, ab и $\bar{a}b$. Отсюда имеем: $G_1 = a\bar{b}$.

4. Определение аргументов оставшейся части логической схемы. Пусть $F_2 = F_2(G_1,a,c)$. То есть, оставшаяся часть ЛБ реализует функцию, не зависящую от b.

5. Минимизация оставшейся части логической схемы. По условию п.4, склеивать можно только такие группы единиц, чтобы аргумент b склеивался и не присутствовал в формуле. На рис. 6, а приведена минимизация БФ F_2. Получаем $F_2 = G_1 \vee \bar{a}\bar{c}$. Данная часть функции может быть реализована в виде структуры, представленной на рис. 4,*б* и реализация функции на этом закончена (рис. 6, *б*) (ЛБ G_4 исключается из схемы).



а) *б)*

Рис. 6. Минимизация (*а*) и реализация (*б*) БФ $F_2=F_2(G_1,a,c)$

5. Заключение

Исследования авторов показали, что применение описанного метода приводит к более быстрой и эффективной реализации БФ на ЛБ, чем при использовании известных методов декомпозиции. Критерием оценки служило число ЛБ, использованных при реализации БФ и скорость нахождения решения. Предлагаемый метод может быть использован при написании САПР ПЛИС с архитектурами FPGA и FLEX.

Литература

1. Соловьёв В. В. Проектирование цифровых систем на основе программируемых логических интегральных схем. – М.: Горячая линия – Телеком, 2001– 636 с.

2. M. Perkowski, S. Grygiel. A survey of literature on function decomposition. Portland State University, Oregon, 1995. – 188 p.
3. Грушвицкий Р.И., Мурсаев А.Х., Угрюмов Е.П. Проектирование систем на микросхемах программируемой логики. – СПб.: БХВ-Петербург, 2002. – 608 с.
4. M.Perkowski, T.Luba, S.Grygiel. Unified approach to functional decompositions of switching functions. Portland State University, Oregon, 1995. – 143 p.

Дата надходження до редколегії: 16.10.2003 р.