УДК 004.43

Применение VHDL моделей неполностью определенных булевых функций при проектировании цифровых схем

П.Н. Бибило

Объединенный институт проблем информатики НАН Беларуси,

bibilo@newman.bas-net.by

***Аннотация* — Предлагаются синтезируемые VHDL модели систем неполностью определенных булевых функций, приводятся результаты экспериментов по различным способам схемной реализации VHDL описаний таких систем функций комбинационными схемами в библиотеке проектирования заказных СБИС и FPGA.**

***Ключевые слова* — система булевых функций, дизъюнктивная нормальная форма (ДНФ), диаграмма двоичного выбора, синтез логической схемы, VHDL, FPGA.**

I. Введение

Современные синтезаторы осуществляют высокоуровневый синтез логических схем заменой описания каждой VHDL конструкции функциональным описанием соответствующей логической подсхемы. После этапа высокоуровневого синтеза выполняются этапы оптимизации и отображения в заданный технологический базис логических элементов [1]. Результаты синтеза в значительной мере зависят от вида исходного VHDL описания, подаваемого на вход синтезатора. В данной работе предлагаются способы описания на языке VHDL систем неполностью определенных (*частичных*) булевых функций. В процессе синтеза исходные VHDL описания частичных булевых функций превращаются в комбинационные схемы, моделями поведения которых являются системы *полностью определенных* функций. В литературе известно, что использование при синтезе логических схем неопределенных значений булевых функций может приводить к более экономичным схемам [2]. Далее приводятся примеры синтезируемых VHDL моделей систем частичных булевых функций и сообщается о результатах экспериментов по схемной реализации VHDL описаний систем частичных функций. Эксперименты показали достаточно большую эффективность использования VHDL описаний частичных функций вместо соответствующих описаний полностью определенных булевых функций.

II. Частичные булевы функции и их VHDL описания

В практике проектирования достаточно часто приходится проектировать функциональные блоки комбинационной логики, характеризующиеся неопределенностью поведения для некоторых (многих) наборов значений входных сигналов. К таким блокам относятся кодовые преобразователи, комбинационные части конечных автоматов, цепочки последовательных арифметических VHDL операторов с операндами, характеризующимися неполными диапазонами значений, и т. д. Математическими моделями таких блоков являются системы частичных булевых функций. Области неопределенности таких функций могут занимать почти всю область их определения, в таких случаях говорят о *слабоопределенной* булевой функции или системе функций. Эффективность использования VHDL моделей частичных функций была изучена в работе [3], где описана методика выделения последовательных VHDL конструкций и замены таких конструкций эквивалентными по поведению матричными представлениями систем частичных булевых функций, при этом функции задавались на наборах значений аргументов.

В данной работе предлагается VHDL модель для системы частичных функций, заданных в интервальной форме.

Под *векторной* булевой функцией $\boldsymbol{f}(\boldsymbol{x})$ будем понимать упорядоченную систему частичных булевых функций $\boldsymbol{f}(\boldsymbol{x}) = (f^1(\boldsymbol{x}),\ldots, f^{\mathrm{m}}(\boldsymbol{x}))$, значениями векторных функций на элементах $\boldsymbol{x}^*$ булева пространства аргументов вектора $\boldsymbol{x} = (x_1,\ldots,x_{\mathrm{n}})$ являются m-компонентные троичные векторы $\boldsymbol{f}(\boldsymbol{x}^*)$. Пример интервального задания частичной векторной функции $\boldsymbol{f}(\boldsymbol{x}) = (f^1(\boldsymbol{x}), f^2(\boldsymbol{x}))$ представлен в табл. 1. Частичная булева функция $f^1(\boldsymbol{x})$ может быть задана парой дизъюнктивных нормальных форм (ДНФ): множество $M^1_{f^1}$ задается в виде $D^1_{f^1} = \bar{x}_1 x_3 x_4 \vee x_1 \bar{x}_3 \bar{x}_4 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4$; множество $M^0_{f^1}$ – в виде $D^0_{f^1} = x_1 x_2 x_3 x_4 \vee x_2 x_3 \bar{x}_4$. Аналогично, в виде пары ДНФ может быть задана частичная булева функция $f^2(\boldsymbol{x})$. Частичная функция $f^1(\boldsymbol{x})$ *реализуется* частичной функцией $f^2(\boldsymbol{x})$ (обозначается $f^1(\boldsymbol{x}) \prec f^2(\boldsymbol{x})$), если и только если $D^1_{f^1} \to D^1_{f^2}$, $D^0_{f^1} \to D^0_{f^2}$, где через $\to$ обозначена операция логической импликации. Для полностью определенных булевых функций отношение реализации является отношением равенства. Отношение реализации векторных

функций сводится к выполнению отношения реализации между компонентными функциями: если каждая компонентная функция $f^{j*}(\boldsymbol{x})$, $j = 1, \ldots, m$, векторной функции $f^*(\boldsymbol{x}) = (f^{1*}(\boldsymbol{x}), \ldots, f^{m*}(\boldsymbol{x}))$ реализует $(f^j \prec f^{j*})$ соответствующую компонентную функцию f^j векторной функции $\boldsymbol{f} = (f^1, \ldots, f^m)$, то тогда векторная функция $\boldsymbol{f}^*$ реализует векторную функцию $\boldsymbol{f}$. В этом случае векторная функция $\boldsymbol{f}^*$ называется *доопределением* векторной функции $\boldsymbol{f}$. Подробное описание различных форм представлений систем полностью определенных и частичных булевых функций дано в [3].

Таблица 1

Интервальная форма системы частичных булевых функций

T^x				T^f	
x_1	x_2	x_3	x_4	f^1	f^2
0	–	–	1	–	1
0	–	1	1	1	–
1	–	0	0	1	–
–	–	1	0	–	0
0	0	0	0	1	0
1	1	1	1	0	1
–	1	1	0	0	–

В языке VHDL при синтезе логических схем используется девятизначный алфавит на базе перечислимого типа std_logic. Массив (вектор) значений типа std_logic обозначается как std_logic_vector [1]. На базе типов std_logic, std_logic_vector предлагается строить VHDL модели частичных функций. Дело в том, что одно из девяти значений данного типа называется don't care (неопределенное значение), обозначается как «–» и предназначается для оптимизации при логическом синтезе. Однако в литературе не показано, каким именно образом значение don't care используется при составлении VHDL описаний частичных функций и как использование неопределенности влияет на результаты синтеза по VHDL описаниям частичных булевых функций.

Предлагаемое VHDL описание интервальной формы системы частичных функций состоит из двух частей: в первой части с помощью логических операторов записываются ДНФ $D^1_{f^i}$, $D^0_{f^i}$, вторая часть – это вызов VHDL функции value_f, формирующей значения функций системы. Описание функции value_f приводится в пакете chast_pack. В листинге 1 задан пример VHDL модели системы функций из табл. 1.

A. Листинг 1. VHDL описание системы частичных функций (табл. 1), заданной на интервалах

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
package chast_pack is
constant N : natural := 4;
constant M : natural := 2;
Function value_f (constant M : natural;
f_nul : std_logic_vector; f_ed
:std_logic_vector )
return  std_logic_vector ;
end chast_pack;
package body chast_pack is
Function value_f (constant M : natural;
f_nul : std_logic_vector; f_ed  :
std_logic_vector )
return  std_logic_vector  is
variable z : std_logic_vector (1 to M);
variable e : std_logic_vector (1 to 2);
begin
for i in f_nul'range loop
    e := (f_nul(i), f_ed(i));
  case e is
    when "10" => z(i):='0';
    when "01" => z(i):='1';
    when "00" => z(i):='-';
    when others  => z(i):='X';
  end case;
end loop;
return z ;
end value_f ;
end;

LIBRARY ieee;
USE ieee.std_logic_1164.all;
use work.chast_pack.all;
ENTITY EXAMPLE1 IS
PORT (x1, x2, x3, x4 : IN std_logic;
F : OUT std_logic_vector (1 to M));
end;
ARCHITECTURE BEH OF EXAMPLE1 IS
signal f_0 : std_logic_vector (1 to M);
signal f_1 : std_logic_vector (1 to M);
BEGIN
f_0(1) <= (x1 and x2 and x3 and x4) or
(x2 and x3 and not x4);
f_1(1) <= (not x1 and x3 and x4) or (x1
and not x3 and not x4) or (not x1 and
not x2 and not x3 and not x4);
f_0(2) <= (x3 and not x4) or (not x1
and not x2 and not x3 and not x4);
f_1(2) <= (not x1 and x4) or (x1 and x2
and x3 and x4);
F <= value_f (M, f_0, f_1);
end;
```

Если же функции системы заданы на наборах значений аргументов, то VHDL модель становится менее компактной (листинг 2). Предполагается, что на остальных наборах булева пространства, не приведенных после ключевого слова when, значения всех частичных функций не определены. Предлагаемые VHDL модели систем частичных функций являются *синтезируемыми* (по ним синтезатор может построить комбинационную

схему) и позволяют проводить проверку отношения реализации частичных функций полностью определенными функциями (логическими схемами) путем формальной верификации.

B. Листинг 2. VHDL описание системы частичных функций, заданных на наборах

```
Library ieee;
use ieee.std_logic_1164.all;
ENTITY EXAMPLE2 IS
PORT (x : IN std_logic_vector(1 to 4);
F : OUT std_logic_vector (1 to 2));
end;
ARCHITECTURE BEH OF EXAMPLE2 IS
begin
F <= "10" when x = "0000" else
     "-1" when x = "0001" else
     "-0" when x = "0010" else
     "11" when x = "0011" else
     "-1" when x = "0101" else
     "00" when x = "0110" else
     "11" when x = "0111" else
     "1-" when x = "1000" else
     "-0" when x = "1010" else
     "1-" when x = "1100" else
     "00" when x = "1110" else
     "01" when x = "1111" else
     "--";  end;
```

III. Оптимизация систем неполностью определенных булевых функций

Основными направлениями оптимизации представлений систем частичных булевых функций являются минимизация в классе ДНФ, т. е. оптимизация двухуровневых представлений, оптимизация многоуровневых представлений на основе разложения Шеннона (BDD-оптимизация) [4, 5] и декомпозиция [6, 7]. Теоретические методы по указанным направлениям оптимизации различных форм представлений систем частичных функций достаточно хорошо изучены [2, 3]. Влияние способов логической оптимизации на результаты логического синтеза в библиотеке проектирования отечественных заказных СБИС экспериментально исследовано в работах [3, 8] для систем полностью определенных функций. Эксперименты над программами логической оптимизации на потоке практических примеров показали, что для различных комбинационных схем целесообразно использовать различные (минимизацию в классе ДНФ, BDD-оптимизацию, декомпозицию) оптимизационные процедуры [8]. Однако BDD-оптимизация наиболее часто приводила к схемам, требующим меньшую площадь кристалла заказной СБИС, даже если сравниваться с результатами использования эффективной программы ESPRESSO [9] совместной минимизации систем полностью определенных функций в классе ДНФ.

BDD-оптимизация булевых функций основана на разложении Шеннона. *Разложением Шеннона* булевой функции $f(\boldsymbol{x}) = f(x_1,...,x_n)$ по переменной x_i называется представление $f(\boldsymbol{x})$ в виде

$$f(\boldsymbol{x}) = \bar{x}_i f_0 \vee x_i f_1 . \quad (1)$$

Если исходная функция $f(\boldsymbol{x})$ является частичной, то и коэффициенты f_0, f_1 разложения являются частичными функциями. Каждый из коэффициентов $f_0 = f(x_1,...,x_{i-1},0,x_{i+1},...x_n)$, $f_1 = f(x_1,...,x_{i-1},1,x_{i+1},...x_n)$ может быть разложен по одной из переменных из множества $\{ x_1,...,x_{i-1},x_{i+1},...x_n \}$. Процесс разложения коэффициентов заканчивается, когда все n переменных будут использованы для разложения. В процессе разложения коэффициенты вырождаются до констант 0, 1, «-». На каждом шаге разложения выполняется сравнение на равенство полученных коэффициентов, и из множества равных коэффициентов оставляется один. Под диаграммой двоичного выбора (BDD – Binary Decision Diagram) понимается ориентированный ациклический граф, задающий последовательные разложения Шеннона булевой функции $f(x_1,...,x_n)$ по всем ее переменным $x_1,...,x_n$ при заданном порядке (перестановке) переменных, по которым проводятся разложения. Функциональная вершина, соответствующая функции f, называется корнем. Под *сложностью (размером)* BDD будем понимать число функциональных вершин BDD.

Далее для векторной булевой функции $\boldsymbol{f}(\boldsymbol{x})$ будут рассматриваться BDD, которые построены по *общей* для всех компонентных функций $f^i(\boldsymbol{x})$ перестановке переменных. В результате BDD-оптимизации [3] матричная форма системы частичных функций заменяется графом BDD, представляющим систему полностью определенных булевых функций. Такая BDD имеет m корневых и две листовые вершины 0, 1, которые обычно дублируются для упрощения изображения графа. BDD, реализующая систему частичных функций (табл. 1), изображена на рис. 1. На языке VHDL описание BDD дано в листинге 3. Каждая функциональная вершина BDD, за исключением вершин-переменных либо их инверсий, задается соответствующей формулой разложения Шеннона.

Листинг 3. VHDL описание BDD (рис. 1)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY EXAMPLE3 IS
PORT (x1,x2,x3,x4,x5 : IN std_logic;
F : OUT std_logic_vector (1 to 2));
end;
ARCHITECTURE BDD OF EXAMPLE3 IS
signal p1: std_logic;
BEGIN
p1 <= not x1 and x4;
F(1) <= (not x3) or (x3 and p1);
F(2) <= (not x3 and p1) or x3 and x4;
end;
```

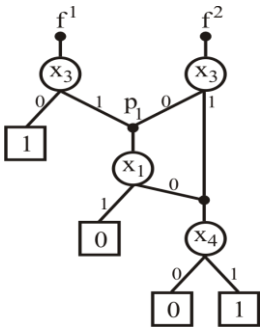

Рис. 1. BDD, реализующая функции (табл. 1)

IV. Экспериментальные исследования

В этом разделе приводятся результаты экспериментов по схемной реализации систем частичных булевых функций на заказных СБИС (ASIC) и на программируемых логических интегральных схемах типа FPGA. В качестве программы совместной минимизации векторных частичных функций в классе ДНФ использовалась программа [10], позволяющая минимизировать *общее число* элементарных конъюнкций, на которых задана минимизированная векторная полностью определенная функция, реализующая исходную частичную векторную функцию.

A. Эксперимент 1. Синтез схем кодовых преобразователей в библиотеке проектирования заказной СБИС

Синтез осуществлялся по двум видам исходных описаний: матричным формам систем частичных функций и BDD-представлениям полностью определенных функций, построенным в результате оптимизации исходных матричных форм систем функций в классе многоуровневых представлений. Исходные системы частичных функций, заданные на наборах значений аргументов, и BDD-представления задавались на языке VHDL. Исходные VHDL-описания матричных форм систем частичных функций поступали на вход синтезатора LeonardoSpectrum [1], который по этим описаниям строил схемы в библиотеке проектирования заказной СБИС. Состав библиотеки логических элементов приведен в [3]. Полученные схемы сравнивались со схемами, построенными по VHDL описаниям полностью определенных систем функций, полученных с помощью программы оптимизации BDD для систем частичных функций.

Результаты эксперимента 1 для двух практических примеров Verg1, Verg2 систем частичных функций (таблиц для хранения микропрограмм команд микропроцессоров) представлены в табл. 2. Каждая система частичных функций была задана на наборах значений аргументов двумя булевыми матрицами B^x, B^f, т. е. троичная матрица T^f выродились до булевой матрицы B^f. Первая булева матрица B^x имела размерность $n \times k$ (n столбцов, k строк), вторая матрица B^f значений функций имела размерность $m \times k$. В табл. 2 используются следующие обозначения: n – число аргументов; m – число функций; k – число двоичных наборов в задании системы функций; L – число элементов схемы; S_{ASIC} – суммарная площадь всех элементов схемы (далее площадь схемы), размещаемых на кристалле заказной СБИС; τ – задержка схемы (нс).

Таблица 2

Результаты эксперимента 1

Пример	n	m	k	Интерпретация пары булевых матриц	Схемная реализация матричной формы			Схемная реализация минимизированных BDD		
					S_{ASIC}	L	τ	S_{ASIC}	L	τ
Verg1	17	61	2003	Система частичных булевых функций	8940	2183	17.79	**3741**	**991**	**12.15**
				Система полностью определенных булевых функций	Синтез не выполнен			5562	1488	17.38
Verg2	18	63	2129	Система частичных булевых функций	8884	2272	20.29	**5809**	**1508**	**12.59**
				Система полностью определенных булевых функций	Синтез не выполнен			7670	1924	20.39

Затем пара булевых матриц была интерпретирована как система совершенных ДНФ (СДНФ) *полностью определенных* булевых функций, заданных в матричной форме, и проведен синтез по оптимизированной BDD. Заметим, что синтезатор LeonardoSpectrum не смог построить логическую схему по исходному VHDL описанию, интерпретируемому как система полностью определенных функций, поэтому в табл. 3 отсутствуют данные по результатам синтеза по таким описаниям. При поиске лучшей перестановки последовательности переменных, по которым строится BDD, для примера Verg1 эвристический алгоритм испытал 500 перестановок, для более сложного примера Verg2 – 100 перестановок аргументов.

Результаты эксперимента 1 показывают, что использование BDD позволяет уменьшить площадь схемы в несколько раз по сравнению со схемной реализа-

цией исходного неоптимизированного VHDL описания системы частичных функций. Кроме того, использование частичной определенности функций, даже без BDD оптимизации, позволяет в полтора-два раза уменьшить площадь схем.

B. Эксперимент 2. Схемная реализация систем псевдослучайных частичных булевых функций в библиотеке проектирования заказной СБИС

Генерируемые случайным образом частичные векторные функции представлялись в матричной форме, пример которой представлен в табл. 1. При генерации варьировались значения: α – число определенных (0, 1) значений в строке матрицы T^x; β – число определенных значений в строке матрицы T^f. Матричные представления конвертировались в VHDL описания, после чего синтезировались логические схемы. Затем исходные матричные представления минимизировались в классах ДНФ и BDD, преобразовывались в VHDL описания, по которым синтезировались комбинационные схемы. Результаты эксперимента 2 представлены в табл. 3. Анализ результатов показывает, что для систем псевдослучайных частичных функций предварительная совместная минимизация в классе ДНФ чаще приводит к схемам меньшей площади. Величина площади уменьшается при уменьшении чисел определенных элементов в матрицах T^x и T^f. При небольшом (α = 4) числе определенных элементов в процессе BDD оптимизации частичные функции системы вырождаются (доопределяются) до констант 0, 1.

Таблица 3

Результаты эксперимента 2 (n = 16, m = 16, k_{mp} = 256)

Пример	α	β	Схемная реализация исходных частичных функций			Схемная реализация минимизированных ДНФ (полностью определенных функций)			Схемная реализация минимизированных BDD (полностью определенных функций)			
			S_{ASIC}	L	τ	S_{ASIC}	L	τ	S_{ASIC}	L	τ	Размер BDD
Gen1	16	4	483875	1351	8.76	**246815**	**662**	**6.36**	605737	1609	9.48	1486
Gen2	12	4	474908	1320	8.63	**167880**	**448**	**5.70**	336837	874	7.56	983
Gen3	8	4	343304	917	10.74	**11260**	**33**	3.39	14620	43	**3.33**	75
Gen4	4	4	454268	1164	11.36	Доопределение функций до констант			Доопределение функций до констант			
Gen5	16	8	527271	1495	12.23	**373257**	**1041**	**9.23**	822492	2072	10.57	2161
Gen6	12	8	464479	1299	11.65	**256334**	**687**	**8.17**	274542	716	8.75	911
Gen7	8	8	387793	1057	11.64	1836	**7**	2.22	**1735**	**7**	**2.08**	8
Gen8	4	8	258583	687	10.69	Доопределение функций до констант			Доопределение функций до констант			

Таблица 4

Результаты эксперимента 3 (n = 16, m = 16, k_{mp} = 256)

Пример	α	β	Схемная реализация исходных частичных функций		Схемная реализация минимизированных ДНФ (полностью определенных функций)			Схемная реализация минимизированных BDD (полностью определенных функций)		
			LUT	τ	LUT	τ	k_{min}	LUT	τ	Размер BDD
Gen1	16	4	891	**3.67**	**406**	**3.73**	212	926	4.88	1486
Gen2	12	4	846	**3.66**	**279**	3.78	166	575	4.33	983
Gen3	8	4	663	3.77	**24**	2.30	27	26	2.56	75
Gen4	4	4	98	3.43	Доопределение функций до констант			Доопределение функций до констант		
Gen5	16	8	1047	3.85	**569**	**3.79**	244	1440	4.49	2161
Gen6	12	8	956	3.89	**418**	**3.86**	227	489	4.25	911
Gen7	8	8	662	4.00	**3**	**1.00**	10	**3**	**1.00**	8
Gen8	4	8	Доопределение функций до констант		Доопределение функций до констант			Доопределение функций до констант		

Таблица 5

Результаты эксперимента 4 (n = 16, m = 16, k_{mp} = 256)

Пример	α	β	Схемная реализация исходных функций (полностью определенных функций)		Схемная реализация минимизированных ДНФ (полностью определенных функций)			Схемная реализация минимизированных BDD (полностью определенных функций)		
			LUT	τ	LUT	τ	k_{min}	LUT	τ	Размер BDD
Gen9	16	4	**803**	**3.91**	**803**	3.92	256	6919	4.32	9479
Gen10	12	4	787	**3.82**	**783**	3.88	256	10384	4.39	13700
Gen11	8	4	**688**	**3.89**	695	4.00	255	18029	4.66	23227
Gen12	6	4	628	**3.72**	**541**	3.76	235	5359	4.83	7951
Gen13	4	4	57	3.18	**54**	**3.13**	77	59	3.23	203

C. Эксперимент 3. Схемная реализация систем псевдослучайных частичных функций на FPGA

Эксперимент 3 отличается от эксперимента 2 только тем, что в качестве технологического базиса была выбрана программируемая логическая схема (ПЛИС) типа FPGA Spartan 3-1000, в качестве примеров систем функций были взяты те же примеры Gen1 – Gen8, что и в эксперименте 2. Синтез осуществлялся с помощью синтезатора XST, входящего в состав ISE (САПР фирмы XILINX). Результаты эксперимента 3 представлены в табл. 4, где в столбцах «LUT» приводится число программируемых элементов FPGA, требуемых для реализации комбинационной схемы, через τ обозначена задержка схемы (нс), k_{min} – число конъюнкций в минимизированной системе ДНФ.

D. Эксперимент 4. Схемная реализация систем псевдослучайных полностью определенных функций на FPGA

Системы псевдослучайных полностью определенных функций в эксперименте 4 генерировались в виде систем ДНФ, заданных в матричной форме парой матриц T^x и B^f, где T^x – троичная матрица, строки которой задают троичные векторы, соответствующие элементарным конъюнкциям ДНФ функций системы; B^f – булева матрица, в которой единичные значения элементов в строке отмечают факт вхождения соответствующей элементарной конъюнкции в ДНФ функций системы [2]. Результаты эксперимента 4 приведены в табл. 5. Анализ результатов показывает, что предварительная минимизация в классе ДНФ псевдослучайных систем полностью определенных функций практически не улучшает (а процедура BDD-оптимизации ухудшает) результаты последующего синтеза в базисе FPGA в отличие от их значительной эффективности [3, 8] при синтезе примеров [9] схем (benchmarks), взятых из практики проектирования.

V. Заключение

Доопределение частичных булевых функций при оптимизации позволяет в несколько раз (по сравнению с реализацией полностью определенных функций) уменьшить сложность логических схем как при проектировании заказных СБИС, так структур FPGA. В отличие от случая полностью определенных функций, предварительная совместная минимизация в классе ДНФ систем псевдослучайных частичных функций в данном эксперименте позволила получить при синтезе лучшие результаты по сравнению с предварительной оптимизацией их BDD-представлений.

Литература

[1] Бибило П.Н. Системы проектирования интегральных схем на основе языка VHDL. StateCAD, ModelSim, LeonardoSpectrum. - М.: СОЛОН-Пресс, 2005. - 384 с.

[2] Закревский А.Д. Логический синтез каскадных схем. – М.: Наука, 1981. - 416 с.

[3] Бибило П.Н. Применение диаграмм двоичного выбора при синтезе логических схем - Минск: Беларуская навука, 2014. - 231 с.

[4] Yang S., Ciesielski M. BDS: a BDD-based logic optimization system. // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. - 2002. - Vol. 21. - № 7. - P. 866 – 876.

[5] Stojkovich S., Stanković M., Stanković R. Determining Assignment of Incompletely Specified Boolean Functions for Compact Representations by Binary Decision Diagrams. // 10th International Workshop on Boolean Problems. Freiberg (Sachsen), 2012. - P. 233-238

[6] Taghavi Afshord S., Pottosin Yu.V., Arasteh B. An input variable partitioning algorithm for functional decomposition of a system of Boolean functions based on the tabular method // Discrete Applied Mathematics. - 2015. - № 185. - P. 208-219.

[7] Lee R.-R., Jiang J.-H. R., Hung W.-L. Bi-decomposing large Boolean functions via interpolation and satisfiability solving // Proceedings of the 45th annual Design Automation Conference, Anaheim, California, 8-13 Jun. 2008 / ACM. - New York, 2008. - P. 636-641.

[8] Авдеев Н.А., Бибило П.Н. Эффективность логической оптимизации при синтезе комбинационных схем из библиотечных элементов // Микроэлектроника. - 2015. - Т. 44. - № 5. - С. 383 – 399.

[9] Brayton K.R., Hactel G.D., McMullen C.T., Sangiovanni-Vincentelli A.L. Logic minimization algorithm for VLSI synthesis. - Boston, e.a.: Kluwer Academic Publishers, 1984. - 193 p.

[10] Торопов Н.Р. Минимизация систем булевых функций в классе ДНФ // Логическое проектирование. Минск: Ин-т техн. кибернетики НАН Беларуси, 1999. Вып. 4. - С. 4 – 19.

The use of VHDL models of partial Boolean functions for the digital circuits design

P.N. Bibilo

United Institute of Informatics Problems of NAS of Belarus

bibilo@newman.bas-net.by

Keywords **- Boolean function, disjunctive normal form, binary decision diagram (BDD), logic circuit synthesis, VHDL, FPGA.**

ABSTRACT

Modern synthesizers perform high-level synthesis of logic circuits by a compiled method – each construction of original VHDL description is replaced by the description of corresponding logical subcircuits [1]. The results of synthesis depend substantially on the original VHDL description which is given to the input of the synthesizer. In this work the methods for description of incompletely specified (*partial*) Boolean functions in VHDL systems are proposed. During the synthesis process, original VHDL descriptions turn into combinational logic circuits whose behavior models are the systems of *completely specified* functions. It is known from the literature that the use of "don't-care-values" of Boolean functions in the logic circuit synthesis may result in more economic circuits [2]. Examples of the synthesized VHDL models of partial Boolean functions are presented; so are the results of experiments on circuit implementation of VHDL descriptions to systems of partial functions. The feasibility of original partial functions in logical circuits was verified by formal verification [6].

The main optimization directions of descriptions of systems of partial Boolean functions are the minimization in DNF class, the multi-level representations based on Shannon's expansion optimization (BDD-optimization) [4, 5] and the decomposition [6, 7]. Theoretical methods on above optimization directions are studied in the literature quite well [2, 3]. The influence of methods for logic optimization on the results of logic synthesis in the design library of domestic VLSI custom circuits has been investigated by experiment in [3, 8] for systems of completely specified functions. The experiments with logical optimization programs on the flow of practical examples show that for various combinational circuits it is rational to use various optimization procedures (minimization in the DNF class, BDD-optimization, decomposition) [8]. However, BDD-optimization resulted more often in circuits requiring lesser area of the ASIC chip even comparing with the results of running the effective program ESPRESSO for joint minimization in DNF class of systems of completely specified functions.

We performed the experiments with circuit implementation of partial Boolean functions systems in the ASIC design library [3] and with programmable logic integrated circuits Spartan 3-1000 of FPGA type. For joint minimization of partial functions in DNF class we used the program [9].

Using the facility of specification completion of partial functions for optimization allows reducing substantially the complexity of combinational logic circuits both in the case of designing circuit example of library elements and in the case of FPGA structures. Unlike the case of completely specified functions, joint minimization in DNF class of systems of partial functions in the experiments allowed to obtain better synthesis results compared to the optimization of BDD representations.

REFERENCES

[1] Bibilo P. N. Systems of Design of Integrated Circuits on the Basis of the VHDL. StateCAD, ModelSim, LeonardoSpectrum. - M.: SOLON Press, 2005. - 384 p. (in Russian).

[2] Zakrevsky A.D. Logical Synthesis of Cascade Circuits. – M.: Science, 1981. - 416 p. (in Russian).

[3] Bibilo P. N. Application of Binary Decision Diagrams at Synthesis of Logical Circuits - Minsk: Belaruskaya navuka 2014. - 231 p. (in Russian).

[4] Yang S., Ciesielski M. BDS: a BDD-based logic optimization system. // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. - 2002. - Vol. 21. - № 7. - pp. 866 - 876.

[5] Stojkovich S., Stanković M., Stanković R. Determining Assignment of Incompletely Specified Boolean Functions for Compact Representations by Binary Decision Diagrams. // 10th International Workshop on Boolean Problems. Freiberg (Sachsen), 2012. – pp. 233-238.

[6] Taghavi Afshord S., Pottosin Yu.V., Arasteh B. An input variable partitioning algorithm for functional decomposition of a system of Boolean functions based on the tabular method // Discrete Applied Mathematics. - 2015. - № 185. - pp. 208-219.

[7] Lee R.-R., Jiang J.-H. R., Hung W.-L. Bi-decomposing large Boolean functions via interpolation and satisfiability solving // Proceedings of the 45th annual Design Automation Conference, Anaheim, California, 8-13 Jun. 2008 / ACM. - New York, 2008. - pp. 636-641.

[8] Avdeev N. A., Bibilo P. N. Effectiveness of logical optimization at synthesis of combinational schemes from library elements // Microelectronics. - 2015. - T. 44. - No. 5. – pp. 383 – 399 (in Russian).

[9] Toropov N. R. Minimization of systems of Boolean functions in DNF class // Logical design. Minsk: Institute of technical cybernetics of NAN of Belarus, 1999. Issue 4. – pp. 4–19 (in Russian).