ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

УДК 519.7

П.Н. Бибило, Д.Я. Новиков

ВЕРИФИКАЦИЯ ЛОГИЧЕСКИХ СХЕМ, РЕАЛИЗУЮЩИХ СИСТЕМЫ ЧАСТИЧНЫХ БУЛЕВЫХ ФУНКЦИЙ

Рассматривается задача верификации логических схем, реализующих системы частично определенных булевых функций. Описываются VHDL-модели различных форм задания таких систем. Предлагается программа для решения рассматриваемой задачи на основе совместного использования троичного параллельного моделирования и сведения к задаче проверки выполнимости конъюнктивной нормальной формы. Приводятся результаты экспериментов по верификации, показывающие высокую эффективность разработанной программы по сравнению с VHDL-моделированием, которое позволяет проводить система ModelSim.

Введение

В практике проектирования обычно приходится иметь дело с логическими схемами, реализующими системы полностью определенных булевых функций. Однако все чаще появляется необходимость в схемной реализации систем не полностью определенных (частичных) булевых функций (ЧБФ) [1], для которых развиваются соответствующие методы оптимизации их представлений, связанные с возможностью доопределения этих функций до полностью определенных [1]. Схемная реализация оптимизированных представлений функций приводит к логическим схемам, функции которых являются полностью определенными. В статье предлагаются две VHDL-модели системы ЧБФ: для интервальной формы задания системы функций и для задания системы функций на наборах значений аргументов. Рассматривается задача верификации исходных VHDL-моделей систем ЧБФ и логических схем, реализующих данные системы. Предлагается программа, позволяющая проводить верификацию значительно быстрее, чем стандартное VHDL-моделирование [2] на полном наборе всевозможных значений аргументов (входных сигналов), которое позволяет проводить система ModelSim [2] и другие системы моделирования, использующие язык VHDL для проектирования цифровых систем.

1. Основные определения

Полностью определенная булева функция (ПБФ) $f(x_1, x_2, \ldots, x_n)$ задается отображением $V^n \to V$, где $V = \{0, 1\}$. Элементами булева пространства V^n являются n-компонентные наборы (векторы) $\underline{x}^*$ нулей и единиц. ПБФ f принимает единичное значение на элементах $\underline{x}^*$ подмножества M_f^1 булева пространства V^n и нулевое значение на элементах подмножества M_f^0. ПБФ f можно задать в виде дизъюнктивной нормальной формы (ДНФ) [1], задающей область M_f^1 единичных значений функции, считая, что на остальных элементах булева пространства f принимает значение 0. Если же на некоторых элементах булева пространства V^n значения функции не определены, то такая функция называется не полностью определенной, или *частичной*. ЧБФ $f(\underline{x})$, где $\underline{x} = (x_1, x_2, \ldots, x_n)$, задается множествами M_f^0, M_f^1 и M_f^- интервалов (или элементов) булева пространства E^n, на которых она принимает соответственно нулевое, единичное и неопределенное значения, при этом $M_f^1 \cup M_f^0 \cup M_f^- = V^n$. Неопределенное значение функции обозначается символом «–».Частичная функция f_1 реализуется частичной либо полностью определенной функцией f_2, если выполняются следующие условия:

$$M_{f_1}^1 \subseteq M_{f_2}^1, \quad M_{f_1}^0 \subseteq M_{f_2}^0. \tag{1}$$

Для полностью определенных булевых функций f_1, f_2 отношение реализации является отношением *равенства*. Для задания ЧБФ f достаточно указать множества M_f^1 и M_f^0, при этом счи-

тая, что множество M_f^- задается неявно как доопределение до V^n. Каждое из множеств M_f^1, M_f^0 можно задать ДНФ D_f^1, D_f^0 соответственно. Пара ДНФ, задающих частичную функцию, обладает свойством ортогональности: $D_f^1 \wedge D_f^0 = 0$, знак $\wedge$ конъюнкции далее опускается. Задание полностью определенных функций в виде ДНФ и частичных функций в виде пар ДНФ будем называть *интервальной матричной формой* задания булевой функции в отличие от матричной формы задания булевой функции на наборах значений аргументов, когда полностью определенная функция задается в виде совершенной ДНФ (СДНФ), а частичная – в виде пары СДНФ [1].

Интервал представляется в виде троичного вектора z и порождает 2^k наборов (где k – число неопределенных компонент вектора z, получающихся заменой в векторе z компонент «–» на все возможные комбинации нулей и единиц. Например, интервал 1–0 порождает четыре следующих набора: 1000, 1001, 1100, 1101.

Под *векторной булевой функцией* $F(\underline{x})$ будем понимать упорядоченную систему частичных булевых функций $F(\underline{x})=(f^1(\underline{x}), \ldots, f^m(\underline{x}))$, значениями векторных функций на элементах булева пространства являются m-компонентные троичные векторы $F(\underline{x}^*)$. Будем говорить, что троичные векторы $\underline{a} = (a_1, a_2, \ldots, a_m)$, $\underline{b} = (b_1, b_2, \ldots, b_m)$ *ортогональны*, если найдется хотя бы одна компонента $i \in \{1, \ldots, m\}$, что a_i, b_i определены и не равны. Например, троичные векторы $\underline{a}$ = (0–10), $\underline{b}$ = (–100) ортогональны, так как для $i = 3$ выполняется условие ортогональности: $a_3 = 1$, $b_3 = 0$.

Форма *интервального* задания частичной векторной функции $F(\underline{x}) = (f^1(\underline{x}), f^2(\underline{x}))$ представлена в табл. 1.

Таблица 1
Пример интервального задания частичной векторной функции

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	1
1	1	1	1	0	0
-	1	1	0	0	-

Частичная функция f^1 (табл. 1) может быть задана парой ДНФ: множество $M_{f^1}^1$ – в виде ДНФ $D_{f^1}^1 = \bar{x}_1 x_3 x_4 \vee x_1 \bar{x}_3 \bar{x}_4 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4$ и множество $M_{f^1}^0$ – в виде ДНФ $D_{f^1}^0 = x_1 x_2 x_3 x_4 \vee x_2 x_3 \bar{x}_4$. Аналогично в виде пары ДНФ может быть задана функция f^2.

Интервальная форма задания векторной функции состоит из троичной матрицы T^x задания элементарных конъюнкций и троичной матрицы T^f вхождений конъюнкций в ДНФ, представляющих области нулевых и единичных значений функций f^1, f^2 – компонент векторной функции $F(\underline{x}) = (f^1(\underline{x}), f^2(\underline{x}))$. Неопределенное значение «–» элемента t_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, k$, матрицы T^f не означает, что функция f^j принимает неопределенное значение на наборах, которые порождает троичный вектор строки i матрицы T^x. Например, на пересечении первой строки и первого столбца матрицы T^f находится «–», однако это не означает, что на четырех наборах 0001, 0011, 0101, 0111 функция f^1 не определена. Рассмотрев вторые строки матриц T^x и T^f, можно убедиться, что на наборах 0011, 0111 функция f^1 принимает единичное значение. Если же все строки матрицы T^x являются попарно ортогональными, то в строках матрицы T^f задаются значения 0, 1, «–» компонентных частичных функций f^j векторной функции $F(\underline{x})$.

Логической схеме L соответствует суперпозиция W базисных полностью определенных булевых функций (функций логических элементов), поэтому процесс синтеза может рассматриваться как процесс нахождения суперпозиции, функционально эквивалентной исходной системе функций. Если функции исходной системы являются полностью определенными, то в ре-

зультате решения задачи логического синтеза значения функций, реализуемых на выходах логической схемы, совпадают со значениями соответствующих функций исходной системы. В этом случае говорят, что логическая схема реализует требуемые функции; другими словами, логическая схема L реализует систему полностью определенных функций $F(\underline{x})$, если для каждого набора $\underline{x}^* \in V^n$ значений входных сигналов реакция схемы (значения выходных сигналов) равна значению системы $F(\underline{x})$ на наборе $\underline{x}^*$. Если функции исходной системы являются частичными, то суперпозиция базисных *полностью определенных* функций должна реализовать исходные *частичные* функции. В данном случае под отношением реализации понимается отношение между функциями. Тот же термин «реализация» используется и для отношения между логической схемой L и системой частичных функций $F(\underline{x})$.

Задача верификаци системы частичных функций и логической схемы заключается в проверке отношения реализации: требуется проверить, реализует ли логическая схема систему исходных частичных функций, по которой построена схема.

2. VHDL-модель интервальной формы задания системы ЧБФ

Предлагаемая VHDL-модель системы частичных функций состоит из двух частей: в первой части с помощью логических операторов записываются ДНФ $D^1_{f^1}$, $D^0_{f^1}$, вторая часть – это VHDL-функция value_f, формирующая значения системы функций для заданного набора значений аргументов. Например, в листинге 1 задана VHDL-модель системы функций из табл. 1.

Листинг 1. *VHDL-модель интервальной формы системы частичных функций*

```
package chast_pack is
constant N : natural := 4;
constant M : natural := 2;
constant K : natural := 7;
end chast_pack;

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE work.chast_pack.all;

ENTITY chast_matr   IS
PORT (x : IN std_logic_vector (1 to N);
      F : OUT std_logic_vector (1 to M));
END;

ARCHITECTURE BEH  of chast_matr   is

signal f_nul : std_logic_vector (1 to M);
signal f_ed : std_logic_vector (1 to M);
Function value_f  (f_nul : std_logic_vector (1 to M);
                   f_ed : std_logic_vector (1 to M) )
return  std_logic_vector  is
variable prom : std_logic_vector (1 to M);
begin
for i in f_nul'range loop
if    ((f_nul(i) = '1' ) and (f_ed(i) = '0')= true) then prom (i):= '0';
elsif ((f_nul(i) = '0' ) and (f_ed(i) = '1')= true) then prom(i) := '1';
elsif ((f_ed(i) = '0' ) and (f_nul(i) = '0')= true) then prom(i) := '-';
elsif ((f_ed(i) = '1' ) and (f_nul(i) = '1')= true) then prom(i) := 'X';
report "error";
end if;
end loop;
return prom ;
end value_f ;
```

```
BEGIN
f_nul(1)  <= (x(1) and x(2) and x(3) and x(4)) or
             (x(2) and x(3) and not x(4));
f_ed(1)  <=  (not x(1) and x(3) and x(4)) or
             (x(1) and not x(3) and not x(4) ) or
             (not x(1) and not x(2) and not x(3) and not x(4));
f_nul(2)  <= (x(3) and not x(4)) or
             (x(1) and x(2) and  x(3) and x(4));
f_ed(2)  <=  (not x(1) and  x(4) ) or
             (not x(1) and not x(2) and not x(3) and not x(4));
F <= value_f (f_nul, f_ed);
END;
```

Элемент $\underline{x}^* \in V^n$ булева пространства V^n будем называть при VHDL-моделировании *входным набором*. Обозначим через k полную элементарную конъюнкцию, соответствующую входному набору $\underline{x}^*$. Будем говорить, что входной набор $\underline{x}^*$ имплицирует ДНФ D, когда справедлива импликация $k \to D$.

Сигнал f_nul(1) имеет единичное значение тогда и только тогда, когда входной набор имплицирует ДНФ $D^0_{f^1}$, представленную в правой части оператора, а сигнал f_ed(1) имеет единичное значение тогда и только тогда, когда входной набор имплицирует ДНФ $D^1_{f^1}$. Аналогично определяются сигналы f_nul(2) и f_ed(2) для функции f^2. Значения выходных сигналов модели функция **`value_f`** формирует по следующим правилам:

1) значение компоненты f^j равно 1 тогда и только тогда, когда входной набор имплицирует ДНФ $D^1_{f^j}$ и не имплицирует ДНФ $D^0_{f^j}$;

2) значение компоненты f^j равно 0 тогда и только тогда, когда входной набор имплицирует ДНФ $D^0_{f^j}$ и не имплицирует ДНФ $D^1_{f^j}$;

3) значение компоненты f^j равно «–» тогда и только тогда, когда входной набор не имплицирует ДНФ $D^0_{f^j}$ и не имплицирует ДНФ $D^1_{f^j}$;

4) значение компоненты f^j является противоречивым, выдается сообщение об ошибке и формируется неизвестное значение «X» тогда и только тогда, когда входной набор имплицирует обе ДНФ $D^0_{f^j}$, $D^1_{f^j}$. В этом случае моделируемая система частичных булевых функций является противоречиво заданной.

Заметим, что предложенная модель интервальной формы системы частичных функций является синтезируемой – по данной модели имеется возможность автоматического построения логической схемы (например, в системе LeonardoSpectrum [2]), что является несомненным достоинством данной модели.

3. VHDL-модель задания системы ЧБФ на наборах значений аргументов

Если система ЧБФ задана на наборах (табл. 2), то элементы матрицы T^f задают значения функций на наборах из левой части таблицы. В этом случае синтезируемая VHDL-модель значительно упрощается и записывается в виде одного VHDL-оператора, пример которого представлен в листинге 2 после слова begin.

Листинг 2. *VHDL-модель системы частичных функций, заданных на наборах*

```
library ieee;
use ieee.std_logic_1164.all;
entity var2 is
  port (x : in  std_logic_vector (1 to 4);
        f : out std_logic_vector (1 to 3));
end;
architecture BEHAVIOR of var2 is
```

```
begin
  f <=
    "0-0" when x = "0000" else
    "001" when x = "0001" else
    "010" when x = "0010" else
    "-01" when x = "0100" else
    "010" when x = "0101" else
    "01-" when x = "1000" else
    "011" when x = "1001" else
    "100" when x = "1010" else
    "---";
end BEHAVIOR;
```

Таблица 2
Пример задания системы ЧБФ на наборах

T^x				T^f		
x_1	x_2	x_3	x_4	f^1	f^2	f^3
0	0	0	0	0	-	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	1	0	0	-	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	-
1	0	0	1	0	1	1
1	0	1	0	1	0	0

4. Верификация на основе логического VHDL-моделирования

Язык VHDL позволяет описывать не только алгоритмы и функции, но и синтезированные логические схемы. Например, логическая схема (рис. 1), синтезированная по VHDL-описанию системы функций (см. листинг 1), представляется структурным описанием, заданным в листинге 3. Функции логических элементов схемы приведены в табл. 3. Синтез логической схемы был осуществлен в системе LeonardoSpectrum [2].

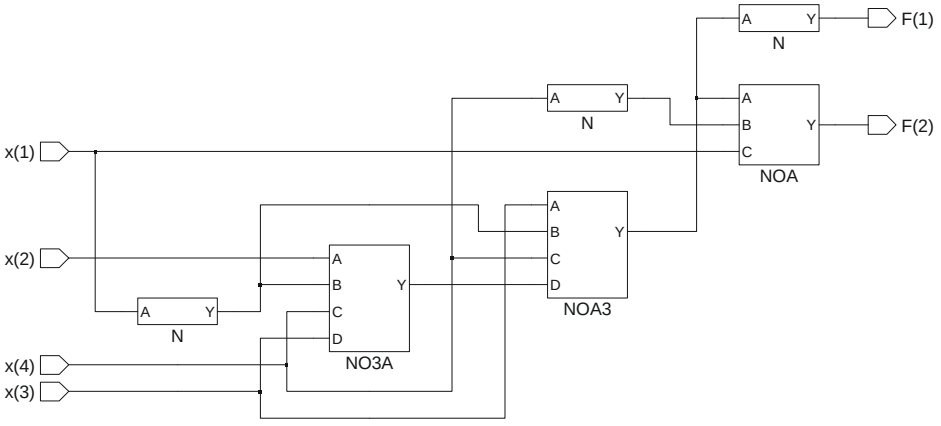

Рис. 1. Логическая схема

Листинг 3. *VHDL-описание логической схемы*

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
library work;
use work.power_pack.all;
entity chast_net is
   port (
      x : IN std_logic_vector (1 TO 4) ;
      F : OUT std_logic_vector (1 TO 2)) ;
end;
```

```
architecture BEH of chast_net is
   signal w1, w2, w3, w4: std_logic ;
begin
   p1 : NOA port map ( Y=>F(2), A=>w1, B=> w4, C=>x(1));
   p2 : NOA3 port map ( Y=> w1, A=>x(3), B=>w3, C=>x(4), D=>w2);
   p3 : N port map ( Y=> w3, A=>x(1));
   p4 : NO3A port map ( Y=> w2, A=>x(2), B=> w3, C=>x(4), D=>x(3));
   p5 : N port map ( Y=> w4, A=>x(4));
   p6 : N port map ( Y=>F(1), A=> w1);
end BEH ;
```

Таблица 3

Функции логических элементов

Элемент	Логическая функция	VHDL-функция
NOA	$y = \overline{AB \vee C}$	Y <= not ((A and B) or C)
N	$y = \overline{A}$	Y <= not A
NOA3	$y = \overline{ABC \vee D}$	Y <= not ((A and B and C) or D)
NO3A	$y = \overline{ABC \vee D \vee E}$	Y <= not ((A and B and C) or D or E)

Решение сформулированной задачи верификации может быть сведено к моделированию и сравнению реакций поведения исходной системы функций с реакциями логической схемы [2] на всех 2^n наборах значений входных сигналов. Соответствующая VHDL-программа была разработана, результат ее работы в системе ModelSim [2] при верификации системы функций (листинг 1) и логической схемы (листинг 3) представлен в табл. 4. В рассматриваемом примере легко перебрать все $2^n = 16$ наборов значений входных сигналов, однако для больших значений n такой перебор трудоемок либо вообще невозможен за приемлемое время. Здесь проблема заключается не в специфике неопределенности функций, а в размерности n булева пространства – число входных наборов растет по экспоненте. Например, уже при $n = 30$ число наборов превышает миллиард. Если же не перебирать все наборы булева пространства, то необходимо сформировать множество тех наборов, на которых хотя бы одна из компонентных функций имеет определенное значение. Для больших значений n эта процедура также является трудоемкой.

Таблица 4

Результат работы VHDL-программы моделирования поведения системы частичных функций и логической схемы

Входные наборы	Значения системы частичных функций	Значения системы функций логической схемы
0000	11	11
0001	-1	**0**1
0010	-0	**0**0
0011	11	11
0100	--	**00**
0101	-1	**0**1
0110	00	00
0111	11	11
1000	1-	**1**0
1001	--	**00**
1010	-0	**0**0
1011	--	**00**
1100	1-	**1**0
1101	--	**00**
1110	00	00
1111	00	00

5. Верификация с помощью программы Verif_ChBF

В связи с трудоемкостью VHDL-моделирования была разработана программа верификации «Verif_ChBF», в основе которой лежат методы [3–5], базирующиеся на троичном параллельном моделировании и сведении задачи верификации к задаче проверки выполнимости конъюнктивной нормальной формы (КНФ) булевой функции. Программа Verif_ChBF позволяет решать задачу верификации логических схем, которые реализуют системы как ЧБФ, так и ПБФ.

В случае когда функции исходной системы F *полностью определены*, для решения задачи верификации программа Verif_ChBF использует метод [5, 9] сведения данной задачи к задаче проверки выполнимости КНФ. При этом строится так называемая схема сравнения [5], на единственном выходе которой появляется константа 0 тогда и только тогда, когда логическая схема L реализует систему F. Затем по методу Цейтина [6] формируется КНФ разрешения, описывающая все допустимые комбинации сигналов на полюсах схемы сравнения. Далее к КНФ разрешения добавляется дизъюнкт, устанавливающий выход схемы сравнения в 1. Полученная таким образом КНФ выполнима тогда и только тогда, когда схема L не реализует систему F.

Если же исходная система F является системой *частичных функций*, программа Verif_ChBF производит параллельное троичное моделирование [3] логической схемы L одновременно на всех интервалах, на которых определена система F. При таком моделировании сигналы на полюсах схемы описываются троичными переменными и вычисления в процессе моделирования производятся соответственно над троичными переменными. Преимущество троичного моделирования над двоичным VHDL-моделированием достигается по ряду причин: интервалы не требуется преобразовывать в наборы, что позволяет резко сократить длину требуемых для моделирования векторов; элементы моделируемой логической схемы просматриваются только один раз. Однако по результатам троичного моделирования не всегда удается ответить на вопрос, реализуется ли система F схемой L на всех интервалах, т. е. могут остаться интервалы, на которых реализуемость системы F схемой L необходимо анализировать дополнительно. Для такого анализа используется метод [4] сведения к задаче проверки выполнимости КНФ K, заключающейся в поиске набора значений аргументов, на котором КНФ K принимает значение 1, или в доказательстве, что такого набора не существует. КНФ K формируется следующим образом. Для исходной системы F ЧБФ строится КНФ R запрета [4], а для логической схемы строится КНФ P разрешения [5]. КНФ K получается конъюнктивным объединением КНФ R и КНФ P: $K = R \wedge P$. Полученная КНФ K выполнима тогда и только тогда, когда схема L не реализует систему F.

Следует отметить, что в программе Verif_ChBF для проверки выполнимости формируемых КНФ используется известная программа MiniSat [7,8].

6. Экспериментальные исследования

Рассмотрим экспериментальные результаты сравнения на практических примерах быстродействия программы Verif_ChBF с VHDL-программой логического моделирования на всех 2^n наборах значений аргументов (табл. 5). Каждая строка таблицы соответствует некоторому верифицируемому примеру. Каждый такой пример включает систему F ЧБФ или ПБФ, заданную на наборах или интервалах значений аргументов, и логическую схему из библиотечных элементов [2]. Рассматриваемые примеры характеризуются следующими параметрами:

– числом N_v аргументов системы F;

– числом N_f функций системы F;

– числом N_c наборов или интервалов из области задания системы F;

– числом N_{bl} элементов схемы;

– средним числом N_v^{bl} входов элементов схемы;

– средним числом N_f^{bl} выходов элементов схемы;

– средним числом N_c^{bl} конъюнкций, используемых для функционального описания элементов схемы;

– числом N_c^{sim} моделируемых наборов VHDL-программой.

При проведении эксперимента измерялись время t_v и время t_{msim}, затрачиваемые на верификацию рассматриваемых примеров испытываемыми программными средствами Verif_ChBF и ModelSim соответственно. Пример verg1 взят из практики проектирования промышленных контроллеров [10], примеры (benchmarks) RCKL, MAX1024, INTB, TIAL – из [11], примеры R_30_15_300, R_25_10_250 являются псевдослучайными.

Таблица 5

Сравнение программы Verif_ChBF с VHDL-программой

Имя схемы	N_v	N_f	N_c	N_{bl}	N_v^{bl}	N_f^{bl}	N_c^{bl}	N_c^{sim}	t_{msim}, с	t_v, с
RCKL	32	7	96	745	1,4	1	1,1	2^{32}	>10000	0,05
MAX1024	10	6	1024	2496	2,3	1	1,5	2^{10}	0,63	0,08
INTB	15	7	664	1124	2,7	1	2	2^{15}	2,5	0,15
TIAL	14	8	640	900	2,8	1	2	2^{14}	5,0	0,13
R_30_15_300	30	15	300	6483	2,9	1	1,9	1794272	1800,0	0,35
R_25_10_250	25	10	250	3241	2,8	1	1,8	42590	85,0	0,08
verg1	17	61	938	1410	2,7	1	1,8	2^{17}	10,0	0,08

Из табл. 5 видно, что программа Verif_ChBF обладает более высоким быстродействием, выполняя верификацию рассматриваемых примеров на порядки быстрее по сравнению с VHDL-программой моделирования на всех наборах булева пространства, а именно такое моделирование приходится осуществлять, не имея специально разработанной программы верификации, примером которой и является Verif_ChBF.

Заключение

В статье представлены синтезируемые модели частичных функций и программа верификации, позволяющая эффективно решать задачи верификации практической размерности. Показано, что сведение задач верификации к логическому моделированию имеет применение для задач ограниченной размерности, когда число входов схемы не более 30.

Работа выполнена при частичной финансовой поддержке Белорусского республиканского фонда фундаментальных исследований (проект Ф11ОБ–041).

Список литературы

1. Закревский, А.Д. Логические основы проектирования дискретных устройств / А.Д. Закревский, Ю.В. Поттосин, Л.Д. Черемисинова. – М. : Физматлит, 2007. – 589 с.

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

3. Черемисинова, Л.Д. Верификация многоблочных структур с функциональной неопределенностью / Л.Д. Черемисинова, Д.Я. Новиков // Известия Нац. акад. наук Беларуси. Сер. физ-техн. наук. – 2009. – № 2. – С. 98–105.

4. Черемисинова, Л.Д. Проверка схемной реализации частичных булевых функций / Л.Д. Черемисинова, Д.Я. Новиков // Вестник Томского государственного университета. Сер. Управление, вычислительная техника и информатика. – 2008. – № 4. – С. 102–111.

5. Novikov, Ya. Foundations of Hierarchical SAT-Solving / Ya. Novikov, R. Brinkmann // 6th International Workshop on Boolean Problems : Proc. of the Workshop, Freiberg, 23–24 September 2004 / University of Mining and Technology. – Freiberg, 2004. – P. 103–142.

6. Tsetin, G.S. On the Complexity of Derivation in Propositional Calculus / G.S. Tsetin // Studies in Constructive Mathematics and Mathematical Logic / ed. by A.O. Slisenko ; Steklov Mathematical Institute. – Leningrad, 1968. – Part 2. – P. 115–125.

7. Een, N. An Extensible SAT-solver / N. Een, N. Sorensson // Theory and Applications of Satisfiability Testing (SAT 2003) : Proc. of the Intern. Conf., Santa Margherita Ligure, Italy, May 5–8, 2003 / Microsoft Research. – Santa Margherita Ligure, 2003. – P. 502–518.

8. Sorensson, N. MiniSat v1.13 – A SAT Solver with Conflict-Clause Minimization / N. Sorensson, N. Een // The International Conference on Theory and Applications of Satisfiability Testing (SAT 2005) [Electronic resource]. – 2005. – Mode of access : http://www.lri.fr/~simon/contest/results/descriptions/solvers/minisat_static.pdf. – Date of access : 16.07.2010.

9. Kuehlmann, A. Combinational and Sequential Equivalence Checking / A. Kuehlmann, A.J. Cornelis van Eijk // Logic synthesis and Verification. – Norwell : Kluwer Academic Publishers, 2002. – P. 343–372.

10. Бибило, П.Н. Совместное использование синтезаторов Leonardo и XST при проектировании цифровых схем на FPGA / П.Н. Бибило // Информационные технологии. – 2008. – № 3. – С. 7–12.

11. Espresso examples / University of California Berkeley [Electronic resource]. – 2006. – Mode of access : http://www1.cs.columbia.edu/~cs4861/sis/espresso-examples/ex. – Date of access : 06.01.2010.

Поступила 26.05.11

Объединенный институт
проблем информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: bibilo@newman.bas-net.by

P.N. Bibilo, D.Ya. Novikov

VERIFICATION OF LOGIC CIRCUITS IMPLEMENTING SYSTEMS OF INCOMPLETELY SPECIFIED BOOLEAN FUNCTIONS

A task of verification of systems of partially defined Boolean functions is considered. VHDL-models for describing different forms of such systems are proposed. We also propose a software for solving the considered task which is based on combining ternary parallel simulation and reduction to checking out satisfiability of a conjunctive normal form. Experimental results are presented which demonstrate high efficiency of the developed software comparing to the VHDL-simulation in ModelSim system.