УДК 681.326

ГЕНЕРАЦИЯ ТЕСТОВ ДЛЯ ПОСЛЕДОВАТЕЛЬНОСТНЫХ СХЕМ, ИМЕЮЩИХ ТРИГГЕРНЫЕ СТРУКТУРЫ

ХАХАНОВ В.И., СКВОРЦОВА О.Б., ПУДОВ В.А., МАСУД МД. МЕХЕДИ

Предлагается информационное обеспечение систем логического моделирования и генерации тестов для цифровых устройств, ориентированное на обработку программируемых логических интегральных схем до 200 тыс. вентилей, включающих более 25 типов триггерных структур. Язык описаниия проектов – VHDL, поддержка систем проектирования фирм Aldec и Xilinx.

Введение

Наряду с интенсивным усовершенствованием технологий интегральных схем, позволяющим реализовывать все большее число компонентов на одном чипе, цифровые системы стали более сложными, а процесс детализированной разработки систем на уровне вентилей и триггеров превратился в рутинный и трудоемкий. Поэтому стало актуальным использование языков аппаратного описания в процессе разработки цифровых устройств. Эти языки помогают проектировщику разрабатывать и отлаживать цифровые системы вплоть до их реализации на вентильном и триггерном уровнях. Применение систем автоматизированного проектирования для такого преобразования становится все более распространенным. Это аналогично написанию программных средств на высокоуровневом языке, таком, как Си, с использованием компилятора для перевода программы в машинный код. На сегодняшний день существует два наиболее популярных языка аппаратного описания – VHDL и Verilog. Многие известные в мире фирмы, такие как ALTERA, XILINX и ALDEC, используют язык VHDL для описания поведения и структуры цифровых систем в своих программных продуктах. VHDL является классическим языком описания аппаратуры, который может быть использован для описания функционирования и моделирования широкого ряда цифровых систем, колеблясь в сложности от нескольких вентилей до совокупности большого количества сложных интегральных микросхем. Однако в настоящее время большинство программных продуктов известных фирм имеют важный недостаток – отсутствие автоматической генерации тестов, что приводит к большим трудностям для проектировщиков при отладке схем. Наличие такой генерации привело бы к обнаружению ошибок на этапе проектирования, а значит, к экономии времени проектирования, что во много раз бы повысило производительность и себестоимость проекта.

1. Требования, предъявляемые к структуре данных

Для программной реализации генерации теста необходимо иметь математическую модель схемы, позволяющую обрабатывать ее без дополнительной обработки данных [1].

Информационное обеспечение подразумевает предоставление полной информации о функциональных и структурных моделях цифрового устройства для применения в программных реализациях логического моделирования, алгоритмов генерации тестов и поиска неисправностей [2].

Исходя из этого, к структуре данных предъявляются следующие требования:

1. *Минимум необходимой информации.* Так как структура данных используется для решения специфических задач, то она должна содержать только используемую информацию.

2. *Упорядочивание информации.* Вся информация о структуре схемы должна быть упорядочена, чтобы свести дополнительные обработки данных к минимуму при ее использовании.

3. *Простой доступ к информации.* Подразумевает легкий доступ к отдельным элементам данных.

4. *Минимальное время получения информации.* Так как структура данных получается путем конвертирования из кода VHDL(или какого-нибудь другого формата данных), то, учитывая большой размер современных проектов, время получения информации должно быть минимальным.

2. Структура данных “Формат SCH”

Разработанная структура данных “Формат SCH” отвечает всем этим требованиям и призвана облегчить задачи моделирования и диагностики. Формат SCH состоит из четырех основных частей:

$$S=\{S_h,S_s,S_l,S_n\}, \qquad (1)$$

где S_h – раздел заголовка; S_s – структура схемы; S_l – библиотека примитивов; S_n – таблица соответствий линий.

Раздел заголовка. Включает общую информацию о схеме: количество линий, входов, выходов и т.д. Размер этого заголовка всегда одинаковый и не зависит от размера схемы:

$$S_h=\{N,N_{fb},N_{inp},N_{out},N_{typ}\}, \qquad (2)$$

здесь N – общее количество линий в схеме; N_{fb} – количество обратных линий; N_{inp} – количество входных линий; N_{out} – количество выходных линий; N_{typ} – количество типов примитивов.

Структура схемы. Дает представление о построении схемы. Включает список всех линий (кроме входных), номер примитива, формирующего ее, и схему включения. Линии пронумерованы по следующему принципу: первыми идут так называемые линии памяти, потом внутренние и последние выходные линии. Нумерация линий должна отвечать правилу ранжирования: численное значение номера выхода каждого элемента должно быть больше значения номера любого из входов данного элемента:

$$S_s = \{n,T,N_{arg},\alpha\}, \qquad (3)$$

где n – номер линии; T – тип примитива (номер кубического покрытия, формирующего данную линию); α – вектор-список аргументов (линий), формирующих данную линию.

Размер данного раздела определяется количеством невходных линий схемы.

Раздел описания примитивов. Включает библиотеку примитивов, содержащую информацию о примитиве и его функциональное описание, заданное в виде кубического покрытия. Библиотека примитивов содержит описание элементов, содержащихся в схеме. Преимущество такого представления заключается в том, что при использовании нескольких элементов одного типа мы храним только одно описание. Поэтому размер библиотеки определяется не общим количеством элементов в схеме, а количеством типов в ней:

$$S_l = \{T, N_{arg}, N_{vec}, C\}, \qquad (4)$$

где T – тип примитива(порядковый номер элемента в библиотеке); N_{arg} – количество аргументов – число входов элемента данного типа; N_{vec} – количество векторов в кубическом покрытии – определяет размер кубического покрытия; C – кубическое покрытие.

Кубическое покрытие записывается символами четырехзначного алфавита {0, 1, X, U}. Его размер равен:

$$L_{Sl} = N_{arg} * N_{vec}. \qquad (5)$$

Размер раздела определяется количеством типов примитивов и размером их кубических покрытий.

Раздел имен линий. Включает список имен линий и их соответствующие номера.

Известно, что легче работать с переменными целого типа, чем с переменными строкового или символьного типа. Переменные целого типа требуют также меньше памяти. Поэтому в самом начале обработки схемы строковые имена линий заменяются уникальным числовым номером. Соответствия значений имен линий записываются в таблицу и хранятся в отдельном разделе:

$$S_n = \{V, n\}, \qquad (6)$$

где V – имя линии; n – номер линии.

Размер раздела зависит от количества линий и от длин их имен.

Рассмотрим подробнее формат SCH на примере простой схемы (рис.1), описанной на языке VHDL и представленной в листинге 1.

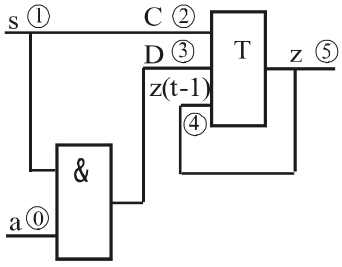

Рис.1. Пример схемы триггера

Листинг 1

Пример схемы триггера, описанной на языке VHDL

```
entity test8 is
  port (
    a,s: in STD_LOGIC;
    z: out STD_LOGIC
  );
end test8;
architecture arch of test8 is
begin
process
constant t : STD_LOGIC :='1';
Begin
 if s =t then
 z<=t and a;
 end if;
End process;
end arch;
```

Схему, описанную в листинге 1, можно представить в виде системы булевых уравнений (7) и схематично, как показано на рис.1:

$$\begin{array}{l} C = s \\ D = s \,\&\, (a) \\ z(t) = (D \,\&\, C) \mid \overline{((C \,\&\, \overline{D}) \,\&\, z(t-1))} \end{array}. \qquad (7)$$

Рассматриваемая схема хранится в формате SCH таким образом:

Раздел ‘Заголовок схемы’:

$$S_h = \left| \begin{array}{|ccccc|} \hline N & N_{inp} & N_{fb} & N_{out} & N_{typ} \\ \hline 6 & 2 & 1 & 1 & 3 \\ \hline \end{array} \right|. \qquad (8)$$

Раздел ‘Структура схемы’:

$$S_s = \left| \begin{array}{|ccc|c|} \hline n & T & N_{arg} & \alpha \\ \hline 2 & 0 & 1 & 5 \\ 3 & 1 & 2 & 0,1 \\ 4 & 0 & 1 & 1 \\ 5 & 2 & 3 & 4,3,2 \\ \hline \end{array} \right|. \qquad (9)$$

Раздел ‘Соответствия линий’:

$$S_n = \left| \begin{array}{|c|cccccc|c|} \hline V & a & s & z(t-1) & D & C & z \\ \hline n & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline \end{array} \right|. \qquad (10)$$

Раздел ‘Библиотека примитивов’:

$$S_l = \left| \begin{array}{|ccc|c|} \hline T & N_{arg} & N_{vec} & C \\ \hline 0 & 1 & 2 & \begin{array}{cc} 0 & 0 \\ 1 & 1 \end{array} \\ \hline 1 & 2 & 3 & \begin{array}{ccc} 0 & X & 0 \\ X & 0 & 0 \\ 1 & 1 & 1 \end{array} \\ \hline 2 & 3 & 6 & \begin{array}{cccc} 1 & 0 & X & 0 \\ 0 & X & 0 & 0 \\ X & 0 & 0 & 0 \\ 1 & 1 & X & 1 \\ 0 & X & 1 & 1 \\ X & 1 & 1 & 1 \end{array} \\ \hline \end{array} \right|. \qquad (11)$$

Исходя из изложенного материала, можно сделать вывод, что формат SCH соответствует требованиям, выдвинутым в начале этой статьи. Разработана программа Conversion, конвертирующая систему логических уравнений, описанных при помощи языка VHDL, в формат SCH. На рис.2 изображен пример использования программы в системе TestBuilder, генерирующей тесты для ОКН комбинационных схем.

```
Test:
    a    b       c       d
 ____________________________
    012345  012345
    ======  ======
 0: 01X010  1..1.1   25.00%
    ——      1..1.1 = 25.00%
 1: 110111  00.000   41.67%
    ——      X0.X0X = 66.67%
 2: 11X111  0..0.0   25.00%
    ——      X0.X0X = 66.67%
 3: 001001  010010   50.00%
    ——      XX0XXX = 91.67%
 4: 01X010  1..1.1   25.00%
    ——      XX0XXX = 91.67%
 5: 100000  1111.1   41.67%
    ——      XXXXXX =100.00%
```

Рис.2. Результат генерации тестов для ОКН комбинационной схемы

Программа TestBuilder разработана на основании использования модифицированного К-алгоритма [2]. Из рис. 2 видно, что построенный программой TestBuilder тест покрывает 100% неисправностей в схеме. Столбец ‘a’ содержит пронумерованные линии из рис.1. Столбец ‘b’ - это входные тестовые воздействия. В ‘c’ показаны неисправности, проверяемые построенным тестом. И, наконец, ‘d’ содержит процент неисправностей, проверяемых конкретной тестовой последовательностью. Для того чтобы сделать полученный тест читабельным в среде VHDL, необходима его конвертация в требуемый формат. Полученный TestBentch представлен в листинге 2.

Листинг 2

Файл VHD, содержащий тестовые воздействия

```
entity test8_tb is
end test8_tb;
  architecture TB_ARCHITECTURE of test8_tb is
  — Component declaration of the tested unit
  component test8
  port(
     a : in STD_LOGIC;
     s : in STD_LOGIC;
     z : out STD_LOGIC);
end component;
  signal a : STD_LOGIC;
  signal s : STD_LOGIC;
  signal z : STD_LOGIC;
begin
    — Unit Under Test port map
  UUT : test8
    port map
     (a => a,
     s => s,
     z => z);
process begin
    a<='0'; s<='1'; wait for 5 ns;
    a<='1'; s<='1'; wait for 5 ns;
    a<='0'; s<='0'; wait for 5 ns;
wait;
end process;
end TB_ARCHITECTURE;
configuration TESTBENCH_FOR_test8 of test8_tb is
 for TB_ARCHITECTURE
   for UUT : test8
       use entity work.test8(arch);
   end for;
 end for;
end TESTBENCH_FOR_test8;
```

Total checked 12 of 12 SF (100.00%):

На рис. 3 показан результат генерации тестов для рассматриваемой схемы, представленный в виде диаграммы.

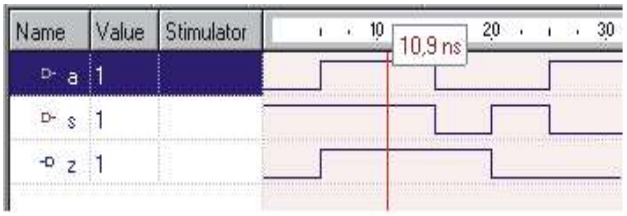

Рис.3. Тест Bentch (диаграмма)

Заключение

Таким образом, предложенное информационное обеспечение производит логическое моделирование и генерацию тестов для цифровых устройств, ориентировано на обработку программируемых логических интегральных схем и имеет реальную возможность интегрирования в области CAD-систем. Что касается практического применения полученных результатов, то с появлением научного и практического интереса к языку VHDL предлагаемое информационное обеспечение реализовано в качестве базового для тестирования схемных реализаций структурно-функционального уровня описания (примитивные элементы, 25 типов триггеров, вентильные схемы, функциональные комбинационные примитивы, заданные покрытиями).

Литература: 1. *Хаханов В.И.* Техническая диагностика элементов и узлов персональных компьютеров. К.: ІЗМН. 1997.308 с. 2. *Хаханов В.И., Ковалев Е.В., Ханько В.В., Масуд Мд. Мехеди.* Система генерации тестов для проектирования цифровых автоматов в среде VHDL-Active. АСУ и ПА. Вып. 111. 2000. с. 15-21. 3. *Ashenden, Peter.* Designer's Guide to VHDL. Morgan Kaufman, 1994. 500p.

Поступила в редколлегию 14.01.01

Рецензент: д-р техн. наук, проф. Кривуля Г.Ф.

Хаханов Владимир Иванович, д-р техн. наук, профессор кафедры автоматизации проектирования вычислительной техники ХТУРЭ. Научные интересы: техническая диагностика вычислительных устройств, систем, сетей. Увлечения: баскетбол, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26. E-mail: Hahanov@kture.kharkov.ua

Скворцова Ольга Борисовна, аспирантка кафедры автоматизации проектирования вычислительной техники ХТУРЭ. Научные интересы: техническая диагностика вычислительных устройств, систем, сетей. Увлечения: аэробика, иностранные языки, музыка. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26. E-mail: <o_skv2000@mail.ru>

Пудов Виталий Анатольевич, соискатель кафедры автоматизации проектирования вычислительной техники ХТУРЭ. Научные интересы: техническая диагностика вычислительных устройств, систем. Увлечения: программирование. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26. E-mail:<vapudov@mail.ru>

Масуд Мд. Мехеди, аспирант кафедры автоматизации проектирования вычислительной техники ХТУРЭ. Научные интересы: техническая диагностика вычислительных устройств, систем, сетей. Увлечения: путешествия. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.