УДК 004.054

**В.В. СКЛЯР[1], В.С. ХАРЧЕНКО[1], А.С. ПАНАРИН[2], И. САНДЕР[3]**

[1]*Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Украина*
[2]*ЗАО «НПП «Радий», Украина*
[3]*Королевский технологический институт, Швеция*

# ПРИМЕНЕНИЕ КОНЦЕПЦИИ MODEL-BASED TESTING ДЛЯ ВЕРИФИКАЦИИ СИСТЕМ НА БАЗЕ IP-ЯДЕР

*Исследуется применение концепции Model-Based Testing на примере инструментального средства ForSyDe, основанного на формальном языке программирования Haskell, для разработки модели программы, созданной с помощью IP-ядра Nios фирмы Altera, и последующего её тестирования. Предложена последовательность разработки модели, а также ее конвертации в VHDL-код для имплементации в ПЛИС (программируемую логическую микросхему).*

***Ключевые слова:*** *ForSyDe, Model-Based Testing, IP-ядро, soft-процессор, моделирование.*

## Введение

Высокий уровень абстракции при разработке цифровых систем на кристалле (System on Programmable Chip – SoPC) является ответом на вызовы рынка, в результате которых срок разработки IT-продукта должен быть предельно сокращен.

Тестирование является необходимой составляющей процесса верификации систем критического применения [1]. Для таких систем, как правило, задаются критерии полноты тестового покрытия [2].

Одним из подходов к разработке тестов, соответствующих заданному критерию полноты, является концепція Model-Based Testing (тестирование на основе моделей) [3]. В качестве модели для тестирования может выступать модель или часть модели, описывающая функциональные характеристики системы (программного обеспечения). Свойства применяемой модели должны позволять генерировать тестовые данные. Таким образом, Model-Based Testing является разновидностью функционального тестирования [4].

Применение модели, кардинально отличающейся от базовой, реализации позволяет сформировать новую точку зрения на исследуемую систему, что является положительным аспектом при выявлении ошибок.

Кроме того, модель может являться основой для разработки альтернативной версии, применяемой в многоверсионных системах [5].

Целью настоящей статьи является исследование варианта реализации концепции Model-Based Testing,, обладающего следующими особенностями:

– модель строится для soft-процессора, являющегося IP-ядром в составе SoPC [6];

– для разработки модели применяется парадигма функционального программирования [7].

При использовании парадигмы функционального программирования процесс вычисления трактуется как вычисление значений функций в их математическом понимании, т.е. когда каждому значению элемента x из некоторого множества X ставится в соответствие единственный элемент y из множества Y. Таким образом, функциональное программирование предполагает неизменность данных при вызове функции с одними и теми же аргументами. Такой подход является, по-видимому, наиболее адекватным средством описания вычислений.

Примером применения функционального программирования для моделирования цифровых систем является инструментальное средство ForSyDe (Formal System Design), разработанное в Королевском техническом институте (KTH, Стокгольм, Швеция) [8].

## 1. Последовательность действий при реализации концепции Model-Based Testing

Общая методика моделирования цифровых систем с использованием функционального программирования (в частности, в среде ForSyDe) включает следующие шаги.

1. Выбор моделируемых функций.

2. Представление системы в виде набора модулей, предназначенных для преобразования входных сигналов в выходные.

3. Анализ комбинаций входных и выходных сигналов и формирование математических функций, реализуемых модулями обработки сигналов.

4. Построение блок-схемы системы, включающей модули (процессы) и сигналы.

5. Разработка программы на языке функционального программирования (обычно в качестве такого языка используется Haskell), описывающей процессы системы.

6. Компиляция текста программы.

7. Формирование массивов входных данных и выполнение моделирования.

8. Конвертация кода на языке Haskell в код на языке VHDL.

9. Анализ полученного кода на языке VHDL в среде разработке электронных проектов ПЛИС и анализ корректности компиляции.

10. Исследование цифровой системы в среде разработки электронных проектов ПЛИС и/или в составе микросхемы ПЛИС с имплементированным электронным проектом.

Проанализируем предложенный подход применительно к тестированию soft-процессора, реализующего функции ввода дискретных сигналов.

## 2. Анализ функционирования модуля ввода дискретных сигналов на базе soft-процессора

Основное назначение рассматриваемого модуля – считывание цифровых данных с портов ввода и их передача в модуль логического управления, а также передача диагностической информации в модуль диагностики.

В качестве программируемого компонента применяются ПЛИС фирмы Altera. Электронные проекты ПЛИС разрабатываются в среде Altera Quartus II, с применением soft-процессора базирующегося на ядре Altera Nios.

На рис. 1 представлен алгоритм работы программы, который может быть преобразован в схему сигналов (рис. 2). Так как программа предназначена для непрерывной работы, она выполняется в бесконечном цикле. Завершение работы программы или её сброс, предусмотрен только с помощью отключения питания микросхемы ПЛИС. Выполнение важнейших операций программы осуществляется с периодичностью в 10 мс. Передача информации во внешние узлы происходит по запросу о необходимости передачи данных.

Алгоритм работы программы включает следующие действия:

1. Begin – запуск программы с инициализацией переменных и устройств ввода-вывода.

2. Rx Request from LCM – ожидание запроса на отправку данных в модуль логического управления. При его наличии программа переходит к формированию и передаче выходного пакета данных (Tx Message to LCM).

3. Rx Request from LCM_d – аналогичным образом осуществляется ожидание и выполнение запроса на отправку данных в модуль диагностики.

4. 10 ms timer – ожидание флага переполнения 10мс таймера. При его наличии программа переходит к выполнению функции считывания обновлённых данных с внешних портов ввода (Read Input Signals) в глобальные переменные.

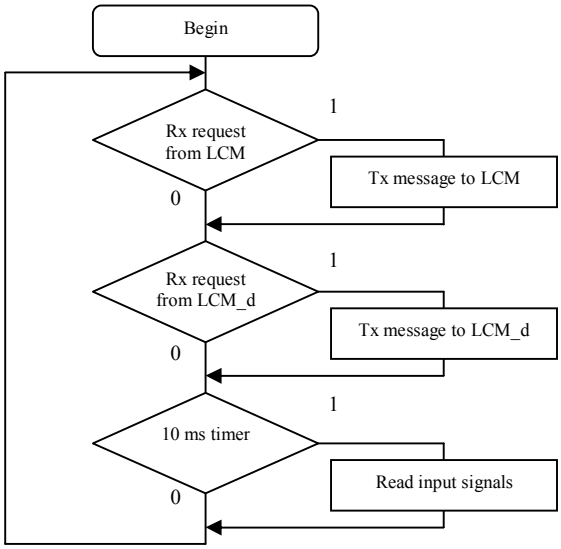



Рис. 1. Блок-схема алгоритма программы модуля ввода дискретных сигналов

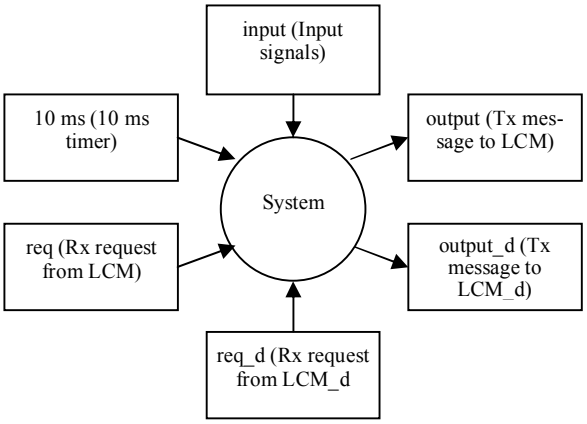



Рис. 2. Схема сигналов системы

## 3. Моделирование модуля ввода дискретных сигналов в среде ForSyDe

Выполним анализ комбинаций входных сигналов системы, представленной на рис. 2, и определим соответствующие значения выходных сигналов (табл. 1).

Таблица 1

Зависимость выходных сигналов от комбинаций входных сигналов

| 10 ms | input | req | output | req_d | output_d |
|---|---|---|---|---|---|
| True | New Signal | True | New Signal | True | New Signal |
| True | New Signal | True | New Signal | False | – |
| True | New Signal | False | – | True | New Signal |
| True | New Signal | False | – | False | – |
| False | Old Signal | True | Old Signal | True | Old Signal |
| False | Old Signal | True | Old Signal | False | – |
| False | Old Signal | False | – | True | Old Signal |
| False | Old Signal | False | – | False | – |

На рис. 3 изображена схема модели системы в среде ForSyDe, в которой можно выделить следующие сущности:

1. Входные сигналы системы.

2. Модуль эмуляции 10 мс таймера (Counter0_9), который совместно с компаратором (Comp9) предоставляет возможность получить таймер с заданной задержкой.

3. Модуль обработки данных – при поступлении импульса переполнения 10 мс таймера, осуществляется считывание данных с внешних портов ввода (input) в локальные переменные. При отсутствии сигнала таймера – данные считаны не будут (остаются предыдущие значения). Модуль учитывает сброс переменных (инициализацию) при запуске программы. Данный модуль представляет собой библиотечную функцию (процесс) Scanld3SY, которая управляет запуском функции nextState (декодер входных сигналов в систему), но хранит в себе прошлый результат её выполнения, и подставляет его при последующих запусках этой функции..

4. Модули приемо-передатчика Output, Output_d.

5. Выходные сигналы системы.

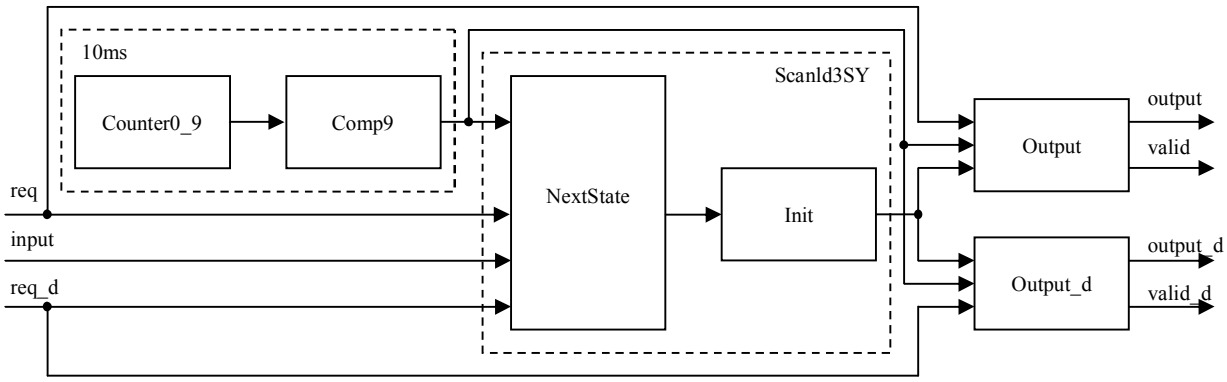



Рис. 3. Блок-схема модели системы в среде ForSyDe

Программа для реализации системы, представленной на рис. 3, пишется на языке функционального программирования Haskell.

С помощью встроенного в систему ForSyDe конвертора есть возможность конвертировать программу, написанную на языке Haskell, в код языка VHDL. Затем этот код может быть проанализирован в среде разработки электронных проектов ПЛИС. Программа на языке VHDL может быть проверена на работоспособность, а затем имплементирована в кристалл ПЛИС. Для конвертации программы в VHDL-код, выбрана упрощенная модель системы, включающая лишь модули NextState, Init и Output (рис. 4). После окончания процедуры конвертации, с помощью утилиты RTL viewer, встроенной в среду разработки Quartus II, был открыт электронный проект системы, и проанализированы схемы цифрового устройства, представленные на рис. 5.

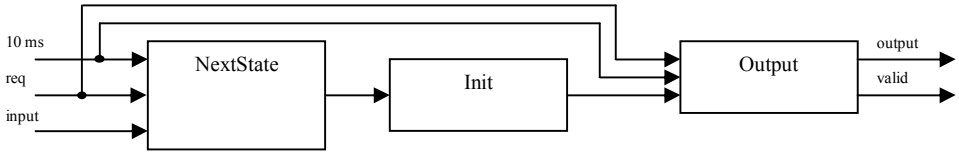
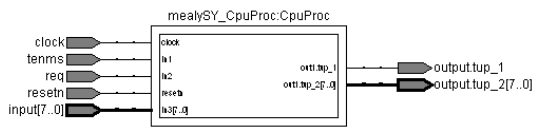
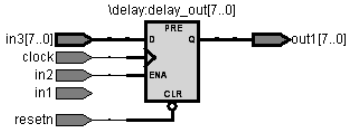


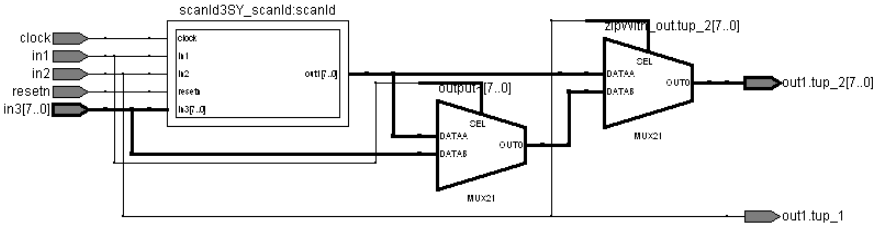

Рис. 4. Упрощенная блок-схема программной модели системы в среде ForSyDe, предназначенная для конвертации в VHDL-код




a – общий вид представления системы (функция mealySY)




b –представление функции scanld3SY




c – представление функции scanld3SY на уровне триггеров

Рис. 5. Представление программной модели системы в среде Altera Quartus II

Общий вид проекта отображен на рис. 5а. Система имеет следующие входные сигналы:

clock – частота тактирования системы;

tenms – импульсы переполнения 10 мс таймера;

req – запрос на передачу данных;

resetn – аппаратный порт сброса кристалла;

input – входной порт данных.

Результатом обработки сигналов функцией mealySY является 2 выходных сигнала:

output.tup_1 – сигнал активации передатчика;

output.tup_2 – сигнал данных передатчика.

На рис. 6 отображена осциллограмма, полученная в результате тестирования системы.

Результаты анализа подтвердили полное соответствие всех трех вариантов представления исследуемой цифровой системы. По результатам анализа сделан вывод о том, что все этапы предложенной методики моделирования цифровых систем в среде ForSyDe выполнены в полном соответствии с исходными данными.

Для более удобного рассмотрения результата выполнения программы, с помощью аппаратных команд, на терминал отладчика (рис. 7) осуществляется вывод информации в виде: «X:YYYY», где:

X = F – принят байт запроса функционального канала;

X = S – принят флаг переполнения 10мс таймера считывания обновлённых данных внешнего порта;

YYYY – содержимое переменной Signal.

Как следует из рис. 7, передача ответного байта считанных из порта данных порта соответствует алгоритму реализуемой задачи. При наличии запроса на передачу осуществляется передача сохраненных данных порта, считанных только при переполнении соответствующего таймера.

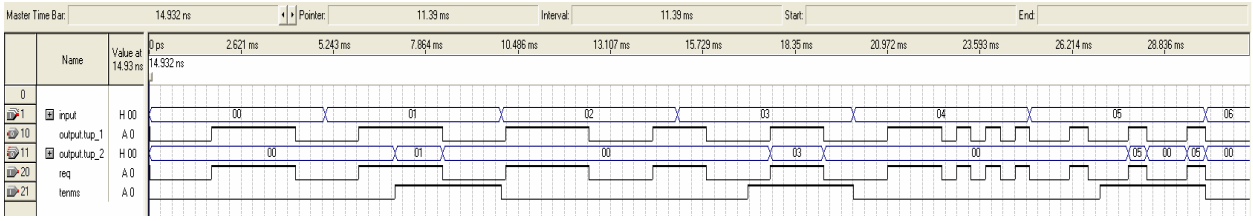



Рис. 6. Результаты Model-Based Software Testing в среде разработки электронных проектов ПЛИС

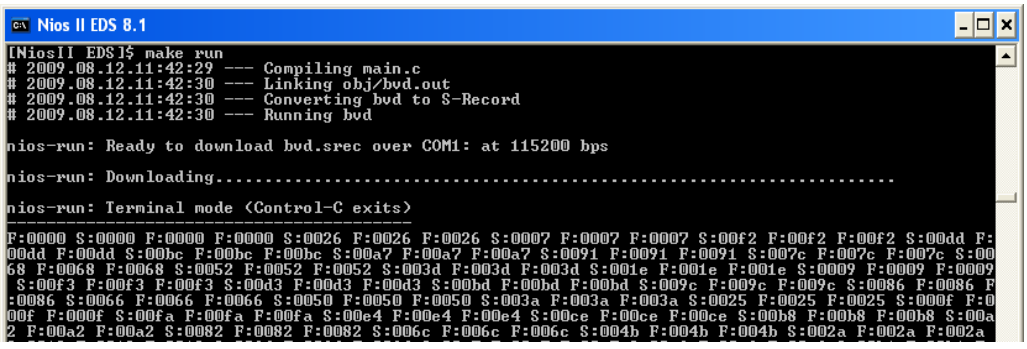



Рис. 7. Вывод отладочной информации с помощью терминала компилятора Gnu Pro

## Заключение

Основной задачей выполняемого исследования являлось разработка элементов технологии реализации концепции Model-Based Testing для проектов ПЛИС на базе процессорных IP-ядер с использованием средства ForSyDe. При этом производилось сравнение полученной модели с программой, разработанной стандартным методом.

При сравнении результата выполнения программ, написанных на языке C и языке Haskell, отмечена идентичность результатов. Отмечается сложность построения одинакового алгоритма на принципиально разных языках, так как реализация похожих функций происходит разными методами.

## Литература

*1. Scott J. Testing Existing Software for Safety - Related Applications / J. Scott, J. Lawrence – Lawrence Livermore National Laboratory, 1995. – P. 77.*

*2. Канер С. Тестирование программного обеспечения / С. Канер, Д. Фолк, Е. Нгуен – К.: Диа-Софт, 2000. – 554 с.*

*3. Utting M. Practical Model-Based Testing: A Tools Approach / M. Utting, B. Legeard - Morgan-Kaufmann, 2006. – P. 456.*

*4. Jacky J. Model-Based Software Testing and Analysis with C# / Jonathan Jacky, Margus Veanes, Colin Campbell, Wolfram Schulte - Cambridge University Press, 2008.*

*5. FPGA-based NPP Instrumentation and Control Systems: Development and Safety Assessment / V. Kharchenko, V. Sklya - Kharkiv, Ukraine: National Aerospace University, NPP "Radiy", SSTC, 2008. – P. 188.*

*6. Hamblen J.O. Rapid Prototyping of Digital Systems: SOPC edition / J.O. Hamblen, T.S. Hall, M.D. Furman. - New York: Springer Science+Business Media, LLC, 2008. – P. 67-103.*

*7. Филд А. Функциональное программирование / А. Филд, П. Харрисон – М.: Мир, 1993. – 637 с.*

*8. Sander I. System Modeling and Design Refinement in ForSyDe. PhD thesis / I. Sander – Stockholm, Sweden: Royal Institute of Technology, 2003. – P. 228.*

*Поступила в редакцию 20.01.2010*

**Рецензент:** д-р техн. наук, проф., декан факультета компьютерной инженерии и управления В.И. Хаханов, Харьковский национальный университет радиоэлектроники, Харьков.

### ЗАСТОСУВАННЯ КОНЦЕПЦІЇ MODEL-BASED TESTING ДЛЯ ВЕРИФІКАЦІЇ СИСТЕМ НА БАЗІ IP-ЯДЕР

***В.В. Скляр, В.С. Харченко, А.С. Панарін, І. Сандер***

Досліджується застосування концепції Model-Based Testing з використанням інструментального засобу ForSyDe, що базується на формальній мові програмування Haskell, для розробки моделі програми, створеної за допомогою IP-ядра Nios фірми Altera, та подальшого її тестування. Запропонована послідовність розробки моделі, а також її конвертації в VHDL-код для імплементації в ПЛІС..

**Ключові слова:** ForSyDe, Model-Based Testing, IP-ядро, soft-процесор, моделювання.

### APPLICATION OF MODEL-BASED TESTING CONCEPT FOR VERIFICATION OF IP-CORE BASED SYSTEMS

***V.V. Sklyar, V.S. Kharchenko, A.S. Panarin, I. Sander***

Application of Model-Based Testing is considered at the example of ForSyDe tool based on formal programming language Haskell for development of program model, created using Altera IP-core Nios, and its further testing. Stages of model development, as well as its conversion into VHDL-code for implementation into PLD (Programmable Logic Device), are proposed.

**Key words**: ForSyDe, Model-Based Testing, IP-core, soft-processor, modeling.

**Скляр Владимир Владимирович** – канд. техн. наук, доцент, доцент кафедры компьютерных систем и сетей Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: vvsklyar@mail.ru.

**Харченко Вячеслав Сергеевич** – д-р техн. наук, профессор, заведующий кафедрой компьютерных систем и сетей Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: V.Kharchenko@khai.edu.

**Панарин Артём Сергеевич** – инженер-программист КБ АСУ ТП, Научно-производственное предприятие «Радий», Кировоград, Украина, e-mail: temchick@yandex.ru.

**Сандер Инго** – д-р философии, старший преподаватель кафедры микроэлектроники и информационных технологий, Королевский технологический институт, Стокгольм, Швеция, e-mail: ingo@kth.se.