УДК 004.054

А.С. ПАНАРИН

ПАО НПП «Радий», Украина

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ SOFT-ПРОЦЕССОРОВ НА БАЗЕ КОНЦЕПЦИИ MODEL-BASED TESTING

Статья включает результат теоретического исследования и практического применения метода тестирования soft-процессоров, в частности, компилятора Nios ядра Altera Sopc Builder. Метод базируется на концепции Model-Based Testing. Инструментальным средством для разработки модели эталонной программы soft-процессора выбран формальный язык программирования ForSyDe. Рассматриваются этапы разработки модели с последующим анализом и сравнением результатов выполнения. В результате исследования подтверждена идентичность всех рассмотренных вариантов реализации цифровой системы. Полученные результаты целесообразно использовать для решения ряда практических задач.

Ключевые слова: *моделирование, Model-Based Testing, soft-процессор, ForSyDe, Altera SoPC Builder, Nios.*

Введение

Во время разработки программного обеспечения soft-процессоров для ПЛИС, часто поднимается вопрос верификации и тестирования конечного продукта. Именно благодаря результатам этих трудов можно сделать выводы устойчивости и надёжности работы устройства. Этот вопрос стал предметом многих исследований [1 – 3], но далеко не каждый результат является универсальным и подходит для каждого частного случая. В данной статье этот немаловажный вопрос рассматривается с практической стороны.

1. Model-Based Testing

Тестирование на основе моделей (Model-BasedTesting) [4, 5] представляет собой подход к тестированию, в рамках которого тесты строятся вручную, автоматизированным образом или генерируются полностью автоматически на основе модели поведения тестируемой системы и модели ситуаций, связанных с ее работой.

Построение тестов на основе моделей заключается в том, что создается (извлекается из проектных документов или, реже всего, берется в готовом виде) модель поведения тестируемой системы, создается модель ситуаций, отражающая основные приоритеты и риски проекта и, чаще всего, использующая структурные элементы модели поведения, и затем строится (генерируется автоматически, создается вручную, или разрабатывается с существенной помощью инструментов) ряд тестов, проверяющих соответствие между реальным поведением тестируемой системы и ее моделью поведения. При этом тестовый набор создается так, чтобы он удовлетворял критерию полноты тестирования, заданному моделью ситуаций.

Для того, чтобы понять как это всё работает, был взят проект, использующий софт-процессор, с последующей попыткой создать к нему модель на основе метода Model-Based Testing. В качестве испытуемого проекта выступает блок, входящий в систему безопасности АЭС.

Первоочерёдной задачей построения модели является выбор языка программирования или инструментального средства, которое бы удовлетворяло поставленным задачам. Эти задачи состоят в следующем:

-модель поведения и модель ситуаций должны максимально близко повторять эталонную систему;

-принцип построение модели или язык программирования должен быть отличным от базовой модели;

- инструментарии разработки проекта модели должны обеспечивать интеграцию модели в ПЛИС с последующим запуском;

- при помощи утилит или иных методов необходимо создать возможность проверки и сравнения результатов выполнения работы систем.

2. Применение среды ForSyDe в качестве инструментального средства для функционального программирования на языке Haskell

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 множест-

ва 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 (FormalSystemDesign), разработанное в Королевском техническом университете (KTH, Стокгольм, Швеция) [7].

Основная концепция ForSyDe заключается в решениипроблемы “gap” путем создания методологии системного проектирования высокого уровня абстракции, базирующейся на так называемом подходе трансформационного уточнения проекта (transformational design refinement).При этом система моделируется в виде сети взаимодействующих процессов, связанных между собой посредством сигналов. Процессы выполняют вычисления, преобразуя входные сигналы в выходные. Таким образом, коммуникационные и вычислительные функции отделены друг от друга. Применение абстрактного времени позволяет реализовывать несколько моделей вычислений, основной из которых является синхронная модель [3].

Учитывая достаточную технологическую зрелость ForSyDe и фундаментальность, положенную в его основу теоретических концепций, представляется целесообразным его использование для альтернативной разработки и верификации IP-Cores для SoPC, реализующих функции параллельных цифровых систем. Поскольку функциональное программирование дает удобное представление цифровых параллельных систем, при помощи ForSyDeможно представить SoPC или его часть (встроенное IP-Core) в диверсном исполнении. Если результаты функционирования основного и диверсного представления SoPCсовпадают, то результаты такой верификации признаются успешными. Диверсное представления SoPC может быть использовано в рабочих проектах для защиты от отказов по общей причине и для усиления так называемой «защиты в глубину». Такой подход позволил бы исправлять ошибки в проекте SoPC уже на начальных этапах его создания.

Целью исследования является анализ возможности применения ForSyDe для реализации концепции формирования альтернативных IP-Cores, позволяющих выполнять независимую верификацию, а также диверсную реализацию для SoPC.

В настоящем исследовании в качестве SoPC рассматриваются индустриальные контроллеры на базе программируемых логических интегральных схем (ПЛИС).

3. Анализ функционирования модуля ввода дискретных сигналов на базе ПЛИС

В качестве моделируемого проекта был выбран блок (модуль) ввода дискретных сигналов. Задачи обработки дискретных сигналов являются типичными для индустриальных систем управления. В частности, рассматриваемый модуль применяется в цифровой информационно-управляющей платформе РАДИЙ для систем безопасности АЭС [3]. Основное назначение 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 IDEAltera Quartus II с применением процессора, базирующегося на ядре Altera Nios.

На рис. 1 представлен краткий алгоритм работы программы, написанной на языке программирования C для компилятора Nios ядра Altera Sopc Builder.

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.

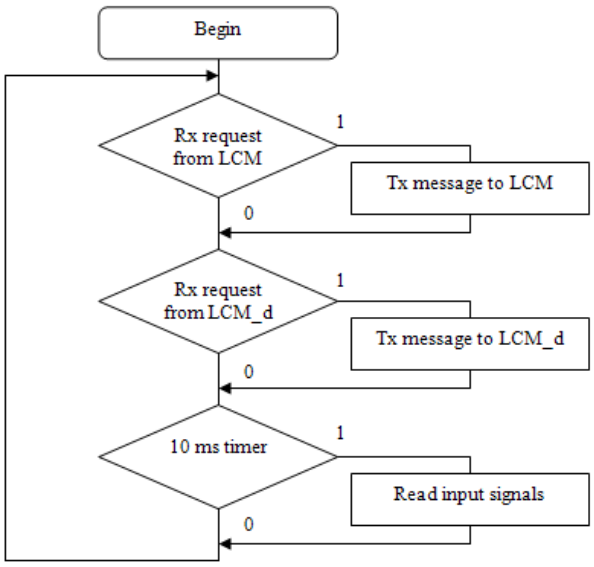


Рис. 1. Блок-схема алгоритма программы модуля ввода дискретных сигналов

4. Подход к моделированию модуля ввода дискретных сигналов на базе ПЛИС в среде ForSyDe

Все входные сигналы имеют определенное логическое состояние, влияющее на их обработку и соответственно выходные сигналы. Так, например, при переполнении 10 мс таймера (10 ms = True) осуществляется считывание обновлённых данных порта (input = New Signal), и при наличии запроса на передачу данных во внешние модули системы (req = True) происходит передача обновлённых данных (output = NewSignal). При отсутствии флага переполнения 10 мс таймера (10 ms = False) сохраняются старые считанные значения, и при наличии запроса на передачу данных (req = True) происходит передача старых данных (output = OldSignal). При отсутствии запроса на передачу данных во внешние модули (req = False) передача данных не осуществляется (output = '– ') следовательно, переменная данных порта игнорируется.

Программа, написанная на языке Haskell, состоит из функций, для вызова которых необходимо передавать входные параметры (переменные или сигналы) и на выходе получить результат её выполнения.

Для моделирования скомпилированной программы необходимо создать массивы входных данных. При запуске функции system, подставляются 3 параметра в виде массивов входных данных: req, req_d и input. После обработки модели программы, функция выводит на экран результат выполнения в виде 2х мерного массива, в котором первым аргументом является десятичное число данных в буфере передатчика, вторым аргументом выступает булевское значение активации передатчика. Так как построенная модель предусматривает работу с двумя приёмо-передатчиками, результатом выполнения программы является передача (вывод на экран) сообщений от двух передатчиков:

({(0,True), (0,False), (0,True), (0,False), (0,True), (0,False), (0,True), (0,False), (0,True), (0,False), (10,True), (10,False), (10,True), (10,False), (10,True), (10,False), (10,True), (10,False), (10,True), (10,False), (20,True)},

{(0,True), (0,True), (0,False), (0,False), (0,True), (0,True), (0,False), (0,False), (0,True), (0,True), (10,False), (10,False), (10,True), (10,True), (10,False), (10,False), (10,True), (10,True), (10,False), (10,False), (20,True)})

В состав системы ForSyDe входит конвертор, который позволяет конвертировать программу, написанную на языке Haskell в код на языке VHDL. Затем этот код может быть проанализирован в IDE для разработки электронных проектов ПЛИС. Программа на языке VHDL может быть проверена на работоспособность, а затем имплементирована в кристалл ПЛИС.

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	–

Для конвертации в VHDL выбрана упрощенная модель системы (рис. 2).

После окончания процедуры конвертации, с помощью утилиты RTL viewer, встроенной в среду разработки Quartus II, был открыт электронный проект программной модели системы, и проанализированы схемы цифрового устройства, представленные на рис. 3.

При детальном рассмотрении сигналов можно провести сравнительный анализвариантов представления исследуемой цифровой системы:

– в виде блок-схемы алгоритма программы (исходные данные для моделирования);

– в виде функциональной программы на языке Haskell в среде ForSyDe;

– в виде электронного проекта ПЛИСв среде разработки Quartus II.

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 с полным соответствием исходным данным.

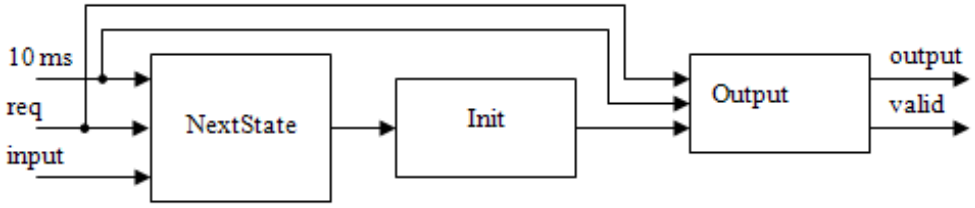
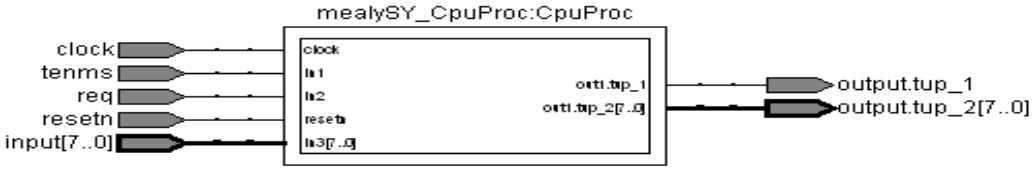
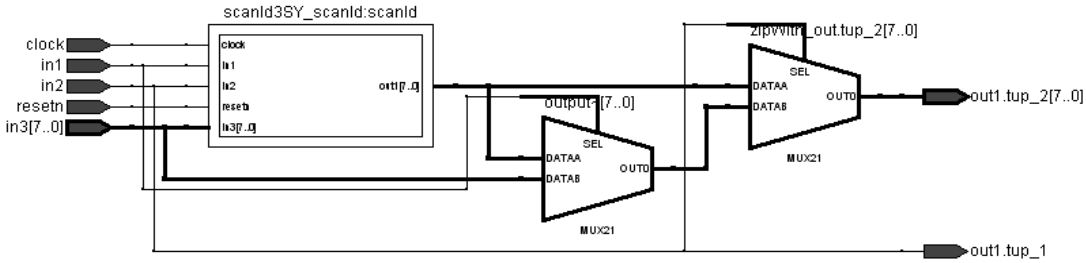


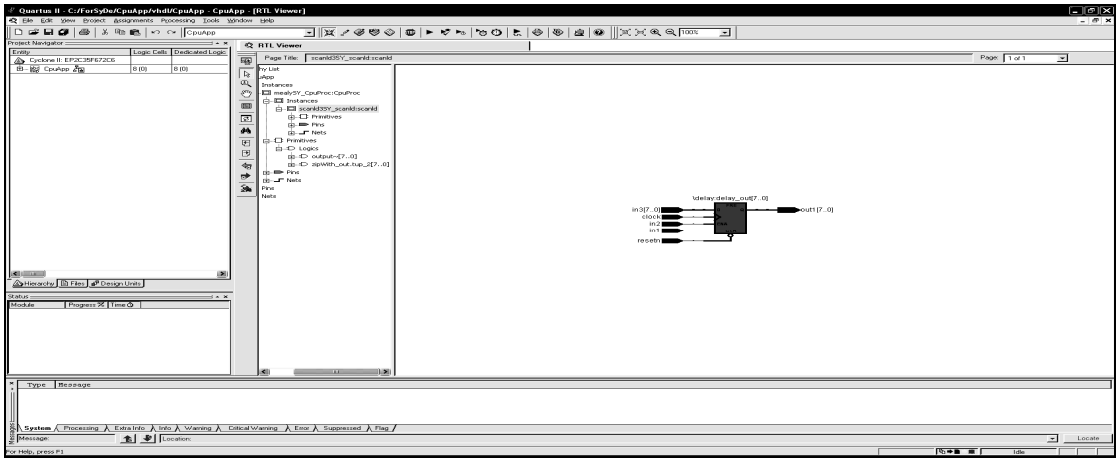
Рис. 2. Упрощенная блок-схема программной модели системы в среде ForSyDe, предназначенная для конвертации в код на языке VHDL



a – общий вид представления системы (функция mealySY)



b – представление функции scanld3SY



c – представление функции scanld3SY на уровне триггеров

Рис. 3. Представление программной модели системы в среде AlteraQuartusII

5. Сравнительный анализ результатов функционирования разработанной системы и стандартного IP-Core

После прохождения основных этапов создания диверсной модели с помощью инструментального средства ForSyDe необходимо скомпилировать проект с помощью оболочки QuartusII и запустить симуляцию. Результат выполнения симуляции в виде осциллограммы приведён на рис. 4.

Результат выполнения программы, скомпилированной с помощью компилятора Nios Gnu Pro и выведенный результат выполнения программы на терминал отладчика (рис. 5) представлен в виде: «X:YYYY», где X = F – принят байт запроса функционального канала; X = D – принят байт запроса диагностического канала; X = S – принят флаг переполнения 10мс таймера считывания обновлённых данных внешнего порта; YYYY – содержимое переменной Signal.

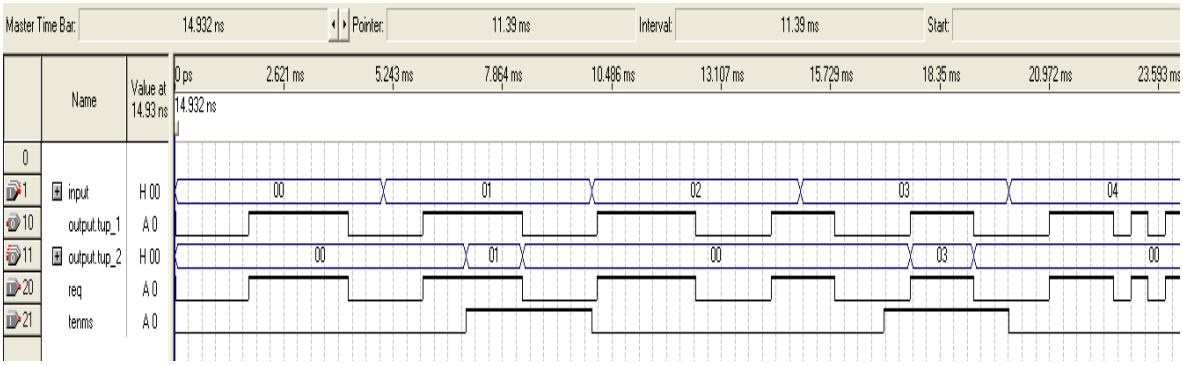


Рис. 4. Симуляция проекта ForSyDe в среде разработки Quartus II

Рис. 5. Результаты моделирования программы на языке C
на терминале отладчика компилятора GERMS Monitor Gnu Pro

Как видно из рис. 5, передача ответного байта, считанного из порта данных, соответствует алгоритму реализуемой задачи. При наличии запроса на передачу осуществляется передача сохраненных данных порта, считанных только при переполнении соответствующего таймера.

При сравнении результата выполнения программ, написанных на языке C и языке Haskell, можно отметить идентичность результатов выполнения программ.

Также стоит отметить определенную сложность построения одинакового алгоритма на принципиально разных языках, так как реализация похожих функций происходит разными методами. Несмотря на это, проведенное моделирование можно считать успешным.

В учебных целях, для определения величины диверсности, были проведены соответствующие исследования.

Постановка задачи: имеется два IP-ядра, которые необходимо исследовать на предмет их диверсности. При сравнении ядер, комиссией выделено n критериев оценки, дана оценка значения показателя диверсности d, и весовой коэффициент L в масштабах системы для каждого i-го критерия оценки.

Справедливы ограничения:

$$d_i \in [0,1];$$

$$\sum_{i=1}^{n} L_i = 1.$$

Следуя вышеописанным правилам, комиссией была дана оценка ядер, приведённая в табл. 2.

Таблица 2

Анализ различий архитектуры исследуемых проектов.

№	Параметр	Nios	ForSyDe	Весовой коэффициент	Значение диверсности
1.	Входные данные	сигнал	массив	0,2	1
2.	Выполнение программы	послед	парал.	0,3	1
3.	Использование шины	да	нет	0,1	1
4.	Использование ПЛИС	да	да	0,1	0
5.	Разрядность процессора	16 бит	нет	0,2	1
6.	Использование DMA	да	нет	0,1	1

С помощью полученных исходных данных и приведённой формулы можно вычислить коэффициент диверсности ядер K_d:

$$K_d = \sum_{i=1}^{n=6} L_i \cdot d_i.$$

Как результат проведения процедуры вычислений, получен коэффициент диверсности для ядер soft-процессора Nios и его ForSyDe-модели, который равен K_d= 0,9, что говорит о достаточной диверсности используемых ядер и подходов к постоению модели.

Выводы

В статье рассмотрена возможность реализации цифровых систем, традиционно разрабатываемых на императивных языках в среде процессоров, на функциональных языках программирования.

Для выполнения исследования предложена методика моделирования цифровых систем в среде ForSyDe.

Моделирование выполнялось для модуля ввода дискретных сигналов на базе ПЛИС. Для данного модуля получены и исследованы следующие варианты реализации цифровой системы:

1) описание на языке Haskell в среде ForSyDe – предназначено только для моделирования; однако,является исходными данными для получения второго вариата реализации;

2) электронный проект на языке VHDL в среде AlteraQuartus;

3) код на языке C предназначен для выполнения в среде эмулятора процессора Nios, являющегося частью электронного проекта ПЛИС.

В результате исследования подтверждена идентичность всех трех вариантов реализации цифровой системы.

Полученные результаты целесообразно использовать для решения следующих практических задач.

Во-первых, реализация функций SoPC альтернативным путем и сравнение результатов тестирования может применяться для независимой верификации таких SoPC в составе систем, важных для безопасности критических объектов. Во-вторых, полученные альтернативные реализации SoPC могут быть использованы для разработки диверсных систем, важных для безопасности критических объектов. Наличие диверсных составляющих системы, выполняющих параллельно общие функции, позволяет усилить защиту системы в глубину и снизить вероятность отказов по общей причине.

Дальнейшие исследования могут быть выполнены в следующих направлениях:

– формальная оценка различия между версиями, полученными с применением разных технологий разработки; такая оценка может быть выполнена, например, методом засева дефектов (faultinjection);

– исследование диверсных архитектур электронных проектов, включающих диверсные IP-Cores (диверсные soft-процессоры, диверсные шины передачи данных, диверсные структуры, выполняющие функции логического управления и т.п.).

Литература

1. Ушаков, А.А. Анализ рисков проектирования и эксплуатации цифровых систем на ПЛИС [Текст] / А.А. Ушаков, А.В. Желтухин, В.В. Скляр и др. – Радіоелектронні і комп'ютерні системи. – 2006. – № 7 (19). – С. 88–98.

2. Оценка и обеспечение качества программных средств космических систем [Текст] / В.С. Харченко, В.В. Скляр, Б.М. Конорев, Ю.Г. Алексеев, и др. – Национальное космическое агентство Украины. – Х. : ХАИ. – 2007. – 244 с.

3. Отказобезопасные информационно-управляющие системы на программируемой логике [Текст]: моногр. / Е.С. Бахмач, А.Д. Герасименко, В.А. Головир, А.А. Сиора, и др.; под ред. В.С. Харченко, В.В. Скляр. – Х.: Нац. аэрокосмический университет "ХАИ"; Кировоград: НПП "Радий". – 2008. – 380 с.

4. Model-Based Testing of Reactive Systems. Advanced Lectures [Text] / M. Broy, B. Jonsson, J.P. Katoen, M. Leucker, et al. – Springer-Verlag, 2005. – 659 p.

5. Utting, M. Practical Model-Based Testing: A Tools Approach [Text] / M. Utting, B. Legeard. – Morgan-Kaufmann. – 2007.

6. Sander, I. System Modeling and Design Refinement in ForSyDe [Text] / I. Sander // PhD thesis. Stockholm. Royal Institute of Technology. – 2003.

7. Raudvere, T. Application and verification of local non-semantic-preserving transformations in system design [Text] / T. Raudvere, I. Sander, A. Jantsch // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – 2008. – P. 1091 – 1103.

Поступила в редакцию 19.01.2012

Рецензент: д-р техн. наук, проф. И.А. Фурман, Харьковский национальный технический университет сельского хозяйства им. Петра Василенко, Харьков, Украина.

ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ SOFT-ПРОЦЕСОРІВ НА БАЗІ КОНЦЕПЦІЇ MODEL-BASED TESTING

А.С. Панарін

Стаття включає результат теоретичного дослідження та практичного застосування методу тестування soft-процесорів, безпосередньо компілятора Nios ядра Altera Sopc Builder. Метод базується на концепції Model-Based Testing. Інструментальним засобом для розробки моделі еталонної програми soft-процесора використовується формальна мова програмування ForSyDe. Розглядаються етапи розробки моделі з наступним аналізом і порівнянням результатів використання. В результаті дослідження підтверджена ідентичність всіх розглянутих варіантів реалізації цифрової системи. Отримані результати доцільно використовувати для вирішення ряду практичних завдань.

Ключові слова: моделювання, Model-Based Testing, soft-процесор, ForSyDe, Altera Sopc Builder, Nios.

SOFT-PROCESSORS SIMULATION USING MODEL-BASED TESTING CONCEPT

A.S. Panarin

The given article includes results of the theoretical research and practical application of soft-processors testing, in particular Nios compiler of Altera Sopc Builder core. Model is based on Model-Based Testing concept. ForSyDe programming formal language is used as instrument for development of soft-processor reference program model. Stages of model development with the follow-up analysis and comparison of performance results are considered. As a result of research the identity of all of the considered variants of realization of the digital system is confirmed. It is expedient to draw on the got results for the decision of row of practical tasks.

Key words: modeling, Model-Based Testing, soft-processor, ForSyDe, Altera Sopc Builder, Nios.

Панарин Артём Сергеевич – инженер-программист КБ АСУ ТП, Научно-производственное предприятие «Радий», Кировоград, Украина, e-mail: panarin83@gmail.com.