УДК 004.056.53 DOI: 10.30929/1995-0519.2018.1.59-64

МЕТОДИКА И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПОСТРОЕНИЯ ИНФОРМАЦИОННОЙ МОДЕЛИ LUT-СХЕМЫ, РАЗМЕЩЕННОЙ В СРЕДЕ FPGA

К. В. Защелкин
Одесский национальный политехнический университет
просп. Шевченко, 1, г. Одесса, 65044, Украина. E-mail: const-z@te.net.ua

Рассмотрена задача контроля целостности программного кода микросхем с LUT-ориентированной архитектурой типа FPGA. Отмечено, что одним из этапов решения этой задачи является построение информационной модели LUT-схемы. Предложена методика получения информации о схеме, необходимой для построения информационной модели. Рассмотрена структура САПР Altera Quartus, в среде которой, предлагается реализовывать методику. Отмечена возможность взаимодействия программного обеспечения, реализующего предложенную методику, с САПР Altera Quartus через соответствующий программный интерфейс API Quartus. Сформулирована последовательность действий методики и формализована информационная модель, строящаяся на основании данных, получаемых при помощи методики. Описана программная реализация каждого этапа методики, основанная на использовании TCL-команд API Quartus. Приведен пример выполнения этапов методики для конкретного FPGA-проекта.

Ключевые слова: программируемые логические интегральные схемы, контроль целостности, FPGA, LUT-ориентированная архитектура, программный код, Quartus API.

МЕТОДИКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ПОБУДОВИ ІНФОРМАЦІЙНОЇ МОДЕЛІ LUT-СХЕМИ, РОЗМІЩЕНОЇ В СЕРЕДОВИЩІ FPGA

К. В. Защолкін
Одеський національний політехнічний університет
просп. Шевченка, 1, м. Одеса, 65044, Україна. E-mail: const-z@te.net.ua

Розглянуто задачу контролю цілісності програмного коду мікросхем з LUT-орієнтованою архітектурою типу FPGA. Обґрунтовано важливість цієї задачі в контексті забезпечення безпеки функціонування FPGA-базованих інформаційних та керуючих систем. Відзначено, що одним з етапів вирішення даної задачі є побудова інформаційної моделі LUT-схеми. Зазначена модель повинна описувати деталі розміщення програмованих блоків LUT в просторі мікросхеми FPGA, їх програмні коди, зв'язки блоків між собою і з зовнішнім середовищем. Запропоновано методику отримання інформації про схему, необхідну для побудови інформаційної моделі. Розглянуто структуру САПР Altera Quartus, в середовищі якої, пропонується реалізовувати методику. Відзначено можливість взаємодії програмного забезпечення, що реалізує запропоновану методику, з САПР Altera Quartus через відповідний програмний інтерфейс API Quartus. Проаналізовано можливість отримання через API Quartus інформації, необхідної для побудови інформаційної моделі LUT-схеми. Сформульовано послідовність дій методики і формалізовано інформаційну модель, яка будується на основі даних, отриманих за допомогою методики. Описано програмну реалізацію кожного етапу методики, що базується на використанні сукупності TCL-команд API Quartus. Наведено приклад виконання етапів методики для конкретного FPGA-проекту. В роботі отримали подальший розвиток підходи до автоматизованого аналізу структури і програмного коду FPGA-проектів з метою контролю їх цілісності. Запропонована в роботі методика та програмне забезпечення, яке її реалізує, можуть знайти застосування для організації підсистеми підготовки даних в рамках системи контролю цілісності програмного коду мікросхем FPGA.

Ключові слова: програмовані логічні інтегральні схеми, контроль цілісності, FPGA, LUT-орієнтована архітектура, програмний код; Quartus API.

АКТУАЛЬНОСТЬ РАБОТЫ. Современные вычислительные и управляющие системы строятся на основе как специализированных, так и программируемых интегральных схем (ИС). Программируемые ИС обычно представлены:

а) микропроцессорами (МП) и микроконтроллерами (МК);

б) программируемыми логическими интегральными схемами (ПЛИС).

Специализированные ИС ориентированы на решение одной конкретной задачи. В отличие от них ИС, относящиеся к классам МП/МК и ПЛИС, могут менять свое поведение в динамике жизненного цикла. Такое свойство подобных ИС обеспечено возможностью их программирования.

Каждый из указанных классов программируемых ИС занимает конкретную потребительскую нишу [1]. МП/МК используются в задачах, не требующих высокой производительности. ПЛИС применяются в проектах требующих высоких показателей производительности одновременно с возможностью перепрограммирования ИС [2].

Наиболее часто используемым на текущий момент видом ПЛИС являются ИС FPGA (Field Programmable Gate Array) [3]. Эти ИС представляют собой упорядоченную в виде двухмерной матрицы совокупность вычислительных блоков. Функции блоков и их соединения управляются программным кодом, размещаемым в конфигурационной памяти FPGA. Изменение содержимого конфигурационной памяти приводит к изменению поведения микросхемы FPGA, т.е. к ее перепрограммированию.

Для программируемых ИС характерна проблема обеспечения целостности их программного кода [4]. Нарушение целостности программного кода таких ИС потенциально может привести к возникновению

неприемлемых последствий. Особенно опасность подобных последствий характерна для ИС, обеспечивающих функционирования информационных управляющих систем для объектов повышенного риска [5]: объектов энергетики, транспортных средств, систем жизнеобеспечения.

Для возникшего на более ранних этапах развития элементной базы класса ИС МП/МК проблема обеспечения целостности программного кода достаточно хорошо проработана. Существует множество эффективных способов решения данной проблемы для МП/МК [6]. Что же касается, возникших на более поздних этапах ПЛИС, и в частности микросхем FPGA, то для них степень решения данной проблемы значительно отстает от уровня ее решения в классе МП/МК.

Микросхема FPGA образована совокупностью простых блоков различного назначения: вычислителей, блоков памяти, блоков умножения, блоков ввода-вывода и пр. Наиболее массовыми в структуре FPGA являются вычислительные блоки LUT (Look-Up Table). В типичной микросхеме FPGA количество блоков LUT может варьироваться от десятков тысяч до более миллиона штук [7]. Каждый из блоков LUT выполняет вычисление одной программируемой логической функции от n переменных. Программирование функции блока LUT производится 2^n разрядным двоичным кодом.

Контроль целостности программного кода микросхемы FPGA основан на: а) вычислении эталонной хеш-суммы программного кода [8]; б) сохранении этой хеш-суммы отдельно от конфигурационной информации FPGA или как ее части.

В данной работе рассматриваются методы сохранения хеш-суммы путем ее внедрения в программный код блоков LUT в виде цифрового водяного знака (ЦВЗ) [9]. Указанные методы обладают двумя особенностями, важными для обеспечения контроля целостности: а) ЦВЗ, будучи внедренным в программный код блоков LUT не изменяет функционирование микросхемы FPGA и не проявляет факт такого внедрения; б) при извлечении ЦВЗ, содержащего хеш-сумму, выполняется восстановление первоначального состояния программного кода блоков LUT [10].

В данной работе рассматривается решение, в рамках которого контролирующая информация (эталонная хеш-сумма) вычисляется для всего программного кода микросхемы FPGA, однако хранилищем этой контролирующей информации выступает программный код только отдельных блоков LUT. Для выполнения контроля целостности программного кода в рамках такого решения для FPGA проекта необходимо выполнить следующую последовательность действий:

а) считать список соединения блоков LUT;

б) считать программные коды блоков LUT;

в) сформировать информационную модель схемы, состоящей из блоков LUT;

г) на основе полученной информационной модели, используя заданный метод обеспечения контроля целостности, вычислить новые значения программных кодов целевых блоков LUT;

д) модифицировать программные коды блоков LUT в соответствии с выполненными вычислениями.

Цель данной работы состоит в разработке формализованной методики выполнения последовательности действий а) – в) с дальнейшим доведением ее до уровня программной реализации.

МАТЕРИАЛ И РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЙ. Проектирование и конечное программирование FPGA выполняется при помощи соответствующих САПР, которые предоставляет каждая компания-производитель микросхем FPGA. Далее примеры реализации методики будут показаны для САПР Quartus компании Altera (подразделение корпорации Intel), которая является одним из наиболее крупных производителей FPGA. САПР для FPGA других компаний-производителей функционируют по принципам весьма схожим с принципами функционирования САПР Quartus. В силу этого предлагаемая методика применима и для них.

САПР Quartus подставляет собой совокупность программных модулей, выполняющих отдельные задачи процесса проектирования и обменивающихся проектной информацией между собой. На рис. 1 показаны основные модули САПР Quartus.

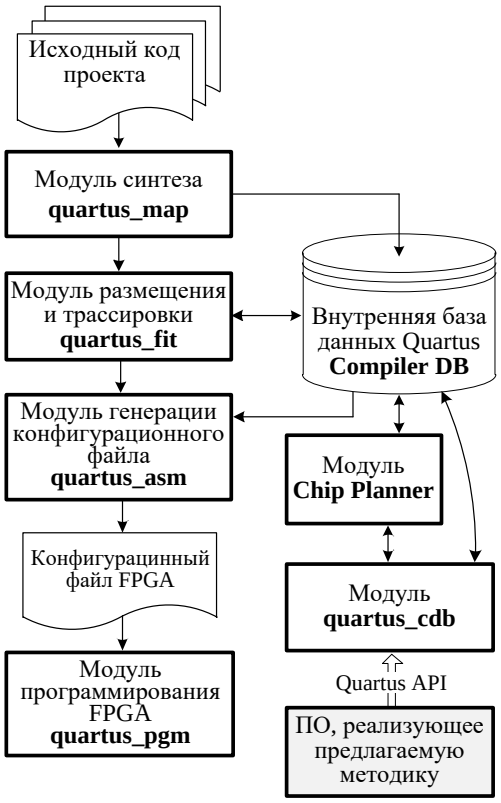

Рисунок 1 – Взаимодействие основных модулей САПР Altera Quartus

В данной работе предлагается методика, осуществляющая получение информации о схеме, размещенной в пространстве матрицы FPGA. Схема размещается на FPGA модулем размещения и трассировки **quartus_fit**. Полная информация о размещении схемы заносится во внутреннюю базу данных САПР Quartus **Compiler DB**. Получение этой информации из внутренней базы данных реализуется при помощи модуля **Chip Planner**. Действия по получению данных от **Chip Planner**, в свою очередь,

обеспечиваются модулем **quartus_cdb** при помощи Quartus API (Application Programming Interface). Quartus API представлен совокупностью команд на языке TCL. В рамках предлагаемой методики программное приложение, которое реализует ее этапы, формирует TCL-команды и передает их модулю **quartus_cdb**. В результате обработки этих команд модуль **quartus_cdb** извлекает из внутренней базы данных Quartus необходимую для выполнения методики информацию и возвращает ее приложению.

Схема во внутренней базе данных Quartus, хранится в виде совокупности узлов (Nodes). Узлами считаются как логические ячейки с блоками LUT, так и входы/выходы схемы. Блоки LUT относятся к узлам **lcell**-типа, а входы/выходы схемы к узлам **io**-типа. Каждый узел в базе данных имеет уникальный числовой идентификатор, символьное имя, а также атрибуты, характерные для конкретного типа узлов.

Предлагаемая методика представляет собой следующую последовательность действий.

1. Загрузка списка соединений узлов из внутренней базы данных Quartus.

2. Получение списка узлов lcell-типа, соответствующих блокам LUT.

3. Для LUT-узлов, полученных на этапе 2, определение следующих атрибутов: идентификатор узла, символьное имя узла, программный код, информация о локализации узла в матрице микросхемы FPGA.

4. Для LUT-узлов, полученных на этапе 2, извлечение информации об их входных и выходных портах: идентификатор порта, имя порта, источник сигнала для порта.

5. Получение списка узлов io-типа, соответствующих входным и выходным выводам микросхемы FPGA, задействованным в проекте.

6. Для узлов, полученных на этапе 5, определение следующих атрибутов: идентификатор узла, имя узла.

7. Для узлов, полученных на этапе 5, извлечение информации об их входных и выходных портах: идентификатор порта, имя порта, источник сигнала для порта.

8. Построение информационной модели схемы на основании данных, полученных на этапах 1-7. Информационная модель содержит структурированную совокупность данных о схеме, обеспечивающую дальнейшую процедуру контроля целостности программного кода микросхемы FPGA. Модель, строящаяся на данном этапе, представляет собой пятикомпонентный кортеж:

$$M = <L, C_L, I, O, C_{LIO}>,$$

где L – множество блоков LUT, образующих схему; каждый элемент l множества L представляет собой кортеж следующего вида:

$$l = <id_L, name_L, code_L, location_L, iPorts_L, oPort_L>,$$

где id_L – уникальный идентификатор блока LUT, $name_L$ – символьное имя блока; $code_L$ – программный код блока; $location_L$ – информация о локализации блока в матрице микросхемы FPGA; $iPorts_L$ множество входных портов блока; $oPort_L$ – выходной порт блока. Для блока с четырьмя входами:

$$iPorts_L \subseteq \{dataA, dataB, dataC, dataD\}.$$

C_L – множество связей блоков LUT между собой. Каждый элемент c_L множества C_L представляет собой пару двухкомпонентных картежей вида

$$c_L = <(id_{Ln}, oPort_{Ln}), (id_{Lm}, ip_m \in iPort_{Lm})>,$$

которая определяет связь между выходным портом блока LUT, имеющим идентификатор id_{Ln}, и одним из входных портов блока LUT, имеющим идентификатор id_{Lm}.

I – множество входов схемы. Каждый элемент i множества I является двухкомпонентным кортежем вида:

$$i = <id_I, name_I>,$$

где id_I – уникальный идентификатор входа; $name_I$ – символьное имя входа;

O – множество выходов схемы. Каждый элемент o множества O является двухкомпонентным кортежем вида:

$$o = <id_O, name_O>,$$

где id_O – уникальный идентификатор выхода; $name_O$ – символьное имя выхода;

C_{LIO} – множество связей блоков LUT с внешней средой (входами и выходами схемы). Множество C_{LIO} состоит из двух непересекающихся подмножеств: $C_{LIO} = C_{LO} \cup C_{LI}$, где C_{LO} – множество связей блоков LUT с выходами схемы, C_{LI} – множество связей блоков LUT со входами схемы. Каждый элемент c_{LO} подмножества C_{LO} представляет собой картеж вида:

$$c_{LO} = <(id_L, oPort_L), id_O>.$$

Каждый элемент c_{LI} подмножества C_{LI} представляет собой картеж следующего вида:

$$c_{LI} = < id_I, (id_L, ip \in iPort_L)>.$$

Предложенная методика реализуется при помощи API Quartus на языке TCL. Реализующий методику TCL-скрипт, передается в качестве параметра исполняемому модулю **quartus_cdb.exe**. Загрузка списка соединений узлов из внутренней базы данных Quartus выполняется при помощи TCL команды **`read_netlist`**.

Получение списка узлов (этапы методики 2 и 5) выполняется при помощи TCL-команды

```
get_nodes -type $nodeType
```

Данная команда имеет единственный параметр `$nodeType`, который определяет требуемый тип узлов: `lcell` или `io`. Команда возвращает коллекцию узлов требуемого типа.

Этапы методики 3, 4, а также 6, 7 выполняются для каждого узла коллекций, полученных на этапах 2 и 5 соответственно. Перебор всех элементов данных коллекций производится при помощи оператора цикла **`foreach_in_collection`** языка TCL.

Получение атрибутов узлов (этапы 3 и 6) производится при помощи TCL-команды

```
get_node_info -node $nodeId -info $attr
```

Параметр `$nodeId` данной команды задает уникальный идентификатор узла, а параметр `$attr` конкретизирует вид необходимого атрибута. Для получения символьного имени узла, его программного кода, информации о локализации узла в матрице FPGA, параметр `$attr` определяется значениями `"name"`, `"LUT Mask"` и `"Location String"` соответственно.

Получение списка входных и выходных портов узла (этапы методики 4 и 7) производится при помощи двух TCL-команд:

```
get_iports -node $nodeId
get_oports -node $nodeId
```

Эти команды имеют единственный параметр `$nodeId` – идентификатор узла, для которого выполняется получение списка портов. Команды возвращают коллекцию требуемых портов.

Получение атрибутов портов выполняется при помощи TCL команды

```
get_port_info -node $nodeId -port_id
 $portId -type $portType -info $attr
```

Данная команда имеет четыре параметра: `$nodeId` – идентификатор узла; `$portId` – идентификатор порта; `$portType` – тип порта (`iport` для входных портов и `oport` для выходных портов); параметр `$attr` конкретизирует вид необходимого атрибута. Для получения символьного имени порта и информации об источнике сигнала для этого порта параметр `$attr` определяется значениями `"port_name"` и `"Input Port Source Atom Id"` соответственно.

Полученные на этапах методики 1-7 данные о структуре схемы и программных кодах ее блоков LUT позволяют построить информационную модель схемы, необходимую для осуществления контроля целостности программного кода FPGA.

В качестве примера для целей компактной демонстрации предложенной методики выбран простой FPGA-проект. Проект представляет собой реализацию двух логических функций от восьми переменных каждая. В логических ячейках FPGA при реализации проекта элементы памяти не используются. Все задействованные логические ячейки FPGA сконфигурированы на функционирование в нормальном режиме (normal mode).

Пример выполнения действий методики.

1. Выполняется загрузка списка соединений узлов из внутренней базы данных Quartus.

2. Получена коллекция из пяти узлов типа lcell (LUT), имеющих идентификаторы 6, 9, 12, 13, 14.

3. Для LUT-узлов, полученных на этапе 2, определены основные атрибуты (табл. 1): символьное имя узла, 16-разрядный программный код (для компактности далее показан в шестнадцатеричной системе), информация о локализации узла в матрице микросхемы FPGA.

Таблица 1 – Атрибуты узлов типа lcell

Node id	Node name	Code	Location
6	y1~0	0002	X4_Y35_N24
9	y1~1	CC00	X4_Y35_N2
12	y1~2	F8F0	X4_Y35_N4
13	y2~0	8000	X4_Y35_N6
14	y2~1	AAAE	X4_Y35_N0

4. Для LUT-узлов, полученных на этапе 2, определены атрибуты входных и выходных портов (табл. 2): имя порта и источник сигнала порта.

5. Получена коллекция из 8 узлов, соответствующих входам схемы. Эти узлы имеют идентификаторы 17, 18, 19, 20, 21, 22, 23, 24. Также получена коллекция из 2 узлов, соответствующих выходам схемы, их идентификаторы 15 и 16.

Таблица 2 – Атрибуты портов для узлов типа lcell

Node id	Input ports		Output port	
	Name	Source Node id	Name	Source Node id
6	dataA dataB dataC dataD	17 10 19 18	dataOut	—
9	dataB dataD	21 22	dataOut	—
12	dataA dataB dataC dataD	23 24 6 9	dataOut	—
13	dataA dataB dataC dataD	17 20 19 18	dataOut	—
14	dataA dataB dataC dataD	13 9 23 24	dataOut	—

6. Для узлов, полученных на этапе 5, определены их символьные имена (столбец "**Node name**" в табл. 3).

7. Для узлов, полученных на этапе 5, определена информация об их входных и выходных портах: имя порта, источник сигнала для порта (столбцы "**Input ports**" и "**Output ports**" в табл. 3).

Таблица 3 – Атрибуты узлов типа io и их портов

Node id	Node name	Input ports		Output port	
		Name	Source Node id	Name	Source Node id
input nodes					
17	x[7]	padIO	—	dataOut	—
18	x[4]	padIO	—	dataOut	—
19	x[5]	padIO	—	dataOut	—
20	x[6]	padIO	—	dataOut	—
21	x[1]	padIO	—	dataOut	—
22	x[3]	padIO	—	dataOut	—
23	x[0]	padIO	—	dataOut	—
24	x[2]	padIO	—	dataOut	—
output nodes					
15	y1	dataIn	12	dataOut	padIO
16	y2	dataIn	14	dataOut	padIO

8. На основе полученных данных (табл. 1 – табл. 3) строится информационная модель схемы. Информационная модель в графической форме показана на рис. 2. Она состоит из: пяти узлов, соответствующих блокам LUT (показаны прямоугольниками); восьми входов и двух выходов схемы (показаны пятиугольниками); связей между ними. Для задачи дальнейшего контроля целостности блоки LUT в информационной модели содержат атрибуты значений программного кода, локализации блоков в матрице FPGA и списки задействованных портов.

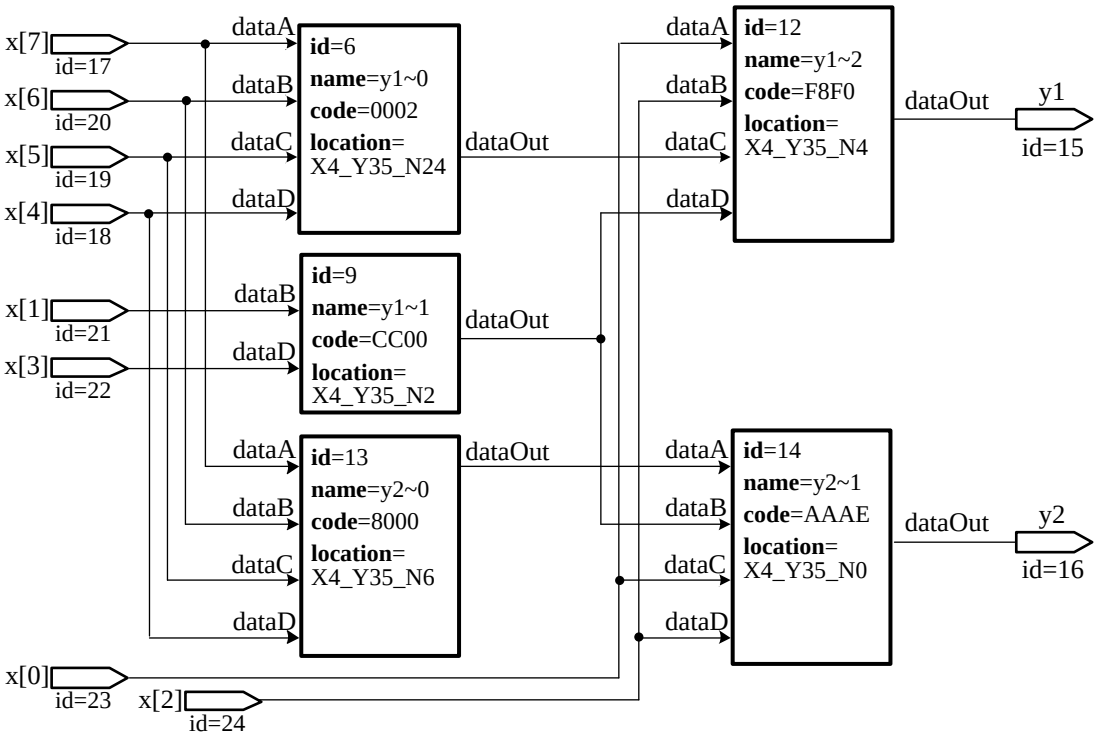

Рисунок 2 – Информационная модель схемы, размещенной в среде FPGA

ВЫВОДЫ. Предложенная в работе методика позволяет получить информационную модель LUT-схемы, описывающей детали размещения блоков LUT в пространстве чипа FPGA, их программные коды, связи блоков между собой и с внешней средой.

В работе обоснована возможность практического применения предложенной методики посредством программного приложения, которое взаимодействует с САПР FPGA через соответствующий интерфейс API.

Методика может найти применение для организации подсистемы подготовки данных в рамках системы контроля целостности программного кода микросхем FPGA. Дальнейших исследований требует вопрос возможных способов обмена данными об информационной модели LUT-схемы с модулями системы контроля целостности программного кода микросхем FPGA, реализующими основную функциональность этой системы.

ЛИТЕРАТУРА

1. Грушвицкий Р.И., Мурсаев А.Х., Угрюмов Е.П. Проектирование систем на микросхемах с программируемой структурой. Санкт-Петербург: БХВ, 2010. 650 с.

2. Защелкин К.В., Кузнецов Н.А., Дрозд А.В. Исследование основных характеристик FPGA-проектов при изменении кодов в блоках LUT. *Науковий вісник Чернівецького університету: Комп'ютерні системи та компоненти.* 2014. Т. 5. Вип. 2. С. 41–45.

3. Andina J., Arnanz E., Valdes M. FPGAs: Fundamentals, Advanced Features, and Applications in Industrial Electronics. Boca Raton: CRC Press, 2017. 266 p.

4. Vacca J. Computer and information security, 2nd edition. USA, Waltham: Morgan Kaufmann Publishers, 2013. 1280 p.

5. Оценка контролепригодности цифровых компонентов встроенных систем критического применения / А.В. Дрозд та ін. *Радіоелектронні і комп'ютерні системи.* 2012. № 6. С. 184–190.

6. Kleidermacher D., Kleidermacher M. Embedded Systems Security: Practical Methods for Safe and Secure Software and Systems Development. Boston: Newnes, 2012. 416 p.

7. Максфилд К. Проектирование на ПЛИС: архитектура, средства, методы. Москва: Додека-XXI, 2007. 408 с.

8. Ferguson N., Schneier B., Kohno T. Cryptography engineering. Hoboken: Wiley, 2013. 354 p.

9. Shih F. Multimedia Security: Watermarking, Steganography, and Forensics. Boston: CRC Press, 2013. 424 p.

10. Защелкин К.В., Иванова Е.Н. Информационная технология внедрения самовосстанавливающих цифровых водяных знаков в LUT-ориентированные контейнеры. *Электротехнические и компьютерные системы.* 2014. №16 (92). С. 78–84.

THE TECHNIQUE AND SOFTWARE IMPLEMENTATION FOR CREATION THE LUT-CIRCUIT INFORMATION MODEL PLACED IN THE FPGA ENVIRONMENT

K. Zashcholkin
Odessa National Polytechnic University,
prosp. Shevchenko, 1, Odessa, 65044, Ukraine. E-mail: const-z@te.net.ua

Purpose. To develop the technique, with the help of which the creation of LUT-circuit information model for FPGA chip program code integrity monitoring (the resulting model should describe the details of the location of the LUTs in the space of the FPGA chip, their program codes, the interconnection of the units with each other and with the external environment), to bring the details of this technique to the level sufficient for software implementation. **Methodology**. The analysis of CAD Altera Quartus structure in the environment, where the target technique is to be implemented, has been made. As a result of analysis the possibility of interaction of software realizing the proposed technique with CAD Altera Quartus through the corresponding software interface API Quartus has been found out. The possibility to obtain the information necessary for the creation of LUT-circuit information model through API Quartus has been researched. **Results.** The sequence of actions of the proposed technique has been formulated. The information model, which is created on the basis of data obtained with the help of the technique, has been formalized. Software implementation of each stage of the technique, based on the use of TCL-commands of the Quartus API, has been described. **Originality.** The approaches to the automated analysis of program code and structure of FPGA-projects with the view of their integrity monitoring were further developed. **Practical value.** The technique offered in the work and the software, which implements it, can be applied in organizing the data preparation subsystem within the framework of the system of FPGA chip program code integrity monitoring. References 10, tables 3, figures 2.

Key words: programmable logic integrated circuits, integrity monitoring; FPGA, LUT-oriented architecture, program code, Quartus API.

REFERENCES

1. Grushvitsky, R.I., Mursaev, A.H., Ugryumov, E.P. (2010), *Proektirovanie sistem na mikro-skhemakh s programmiruemoi strukturoi* [Design of systems based on microchips with programmable structure], BHV, St. Petersburg, Russia.

2. Zashcholkin, K., Kuznetsov, N., Drozd, A. (2014), "Research in Key Features of FPGA-projects in Case of Changing the Codes in LUT-blocks". *Naukovyi visnyk Chernivetskoho universytetu: Kompiuterni systemy ta komponenty*, vol. 5, issue 2, pp. 82-86.

3. Andina, J., Arnanz, E., Valdes, M. (2017), FPGAs: Fundamentals, Advanced Features, and Applications in Industrial Electronics, CRC Press, 266 p.

4. Vacca, J. (2013), Computer and information security, 2nd edition, Morgan Kaufmann Publishers, USA, Waltham, 1280 p.

5. Drozd, O.V., Kharchenko, V.S., Antoshchuk, S. G., Drozd, M. O., Sulima, J. J. (2012), "Assessment in checkability of safety-critical embedded systems components", *Radioelektronni i kompiuterni systemy*, no. 6, pp. 184-190.

6. Kleidermacher, D., Kleidermacher, M. (2012), Embedded Systems Security: Practical Methods for Safe and Secure Software and Systems Development, Newnes, 416 p.

7. Maxfield, C. (2007), *Proektirovanie na PLIS: arkhitektura, sredstva, metody* [The Design Warrior's Guide to FPGAs: Devices, Tools and Flows], Dodeka-XXI, Moscow, Russia.

8. Ferguson, N., Schneier, B., Kohno, T. (2013), Cryptography engineering, Wiley, USA, Hoboken, 354 p.

9. Shih, F. (2013), Multimedia Security: Watermarking, Steganography, and Forensics, CRC Press, Boston, 424 p.

10. Zashcholkin, K.V., Ivanova, E.N. (2014), "Information technology of embedding self-recovery digital watermark in LUT-oriented containers", *Elektrotekhnichni ta kompiuterni systemy*, no. 16 (92), pp. 78-84.

Стаття надійшла 14.02.2018.