DOI: https://doi.org/10.32836/2521-6643-2018-1-56-1
УДК 004.056.53

К. В. Защелкин, кандидат технических наук, доцент кафедры компьютерных интеллектуальных систем и сетей Одесского национального политехнического университета
А. В. Дрозд, доктор технических наук, профессор кафедры компьютерных интеллектуальных систем и сетей Одесского национального политехнического университета
Ю. Ю. Сулима, кандидат технических наук, заведующий электронно-холодильным отделением Одесского технического колледжа Одесской национальной академии пищевых технологий
Е. Н. Иванова, старший преподаватель кафедры компьютерных систем Одесского национального политехнического университета

МЕТОД ФОРМИРОВАНИЯ СТЕГО-ПУТИ ПРИ РЕШЕНИИ ЗАДАЧИ КОНТРОЛЯ ЦЕЛОСТНОСТИ ПРОГРАММНОГО КОДА FPGA-БАЗИРОВАННЫХ УСТРОЙСТВ

Рассмотрены вопросы контроля целостности программного кода FPGA-базированных компонентов компьютерных систем. В качестве эффективного отмечен подход к контролю целостности, в рамках которого контрольная хэш-сумма внедряется в контролируемый информационный объект в виде цифрового водяного знака. Предлагается метод формирования в пространстве LUT-контейнера микросхемы FPGA стеганографического пути (стего-пути) внедрения цифрового водяного знака. Стего-путь представляет собой упорядоченное множество элементарных вычислительных блоков LUT, в программные коды которых выполняется непосредственное внедрение разрядов контрольного цифрового водяного знака. В статье обосновываются основные теоретические положения предлагаемого метода и формулируется последовательность этапов его выполнения.

Ключевые слова: *контроль целостности; FPGA; LUT; цифровой водяной знак; стеганографический путь; программируемые вычислительные блоки.*

Розглянуто питання контролю цілісності програмного коду FPGA-базованих компонентів комп'ютерних систем. В якості ефективного відзначено підхід до контролю цілісності, в рамках якого контрольна хеш-сума вбудовується в контрольований інформаційний об'єкт у вигляді цифрового водяного знака. Пропонується метод формування в просторі LUT-контейнера мікросхеми FPGA стеганографічного шляху (стего-шляху) вбудовування цифрового водяного знака. Стего-шлях являє собою впорядковану множину елементарних обчислювальних блоків LUT, у програмні коди яких виконується безпосереднє вбудовування розрядів контрольного цифрового водяного знака. В дослідженні обґрунтовуються основні теоретичні положення запропонованого методу, формулюється послідовність етапів його виконання.

Ключові слова: *контроль цілісності; FPGA; LUT; цифровий водяний знак; стеганографічний шлях; програмовані обчислювальні блоки.*

Issues of program code integrity monitoring of the FPGA-based components of computer systems is considered. As the most effective, the approach is chosen in which the control hash sum is embedded in the monitored information object as a digital watermark. This makes it possible to hide the fact that integrity monitoring is performed, as well as to hide the hash sum by which integrity is monitored. The disadvantage of the existing methods is that they do not specify the way of forming in the informational object a sequence of elementary units into which the digits of the digital watermark are embedded.

A method for forming in the space of a LUT-container (LUT – Look Up Table) the FPGA chip of a steganographic path (stego-path) of embedding the digital watermark is proposed. Stego-path is an ordered set of elementary computational LUT units, into the program codes of which the direct embedding of the digits of the digital watermark is performed. The paper substantiates the basic theoretical propositions of the proposed method, and formulates a sequence of stages for its implementation. The method determines the way and source of the ordering of the set of LUT units. The method also formalizes the natural constraints on the choice of LUT units for the stego-path. These natural constraints are due to the structure of the LUT-container. The method also defines artificial constraints on the choice of LUT units, which are caused by the bounding components of the stego-key. A rule is established that makes it possible to decide on the inclusion of a LUT unit in the stego path, depending on the connections with the blocks already included in the stego path. The theoretical positions of the method form a cyclic sequence of actions for choosing LUT units from the available set of LUT-container units.

A software implementation of the method is proposed, based on the analysis of the information model of the LUT-container obtained using the CAD system Intel

Quartus Prime. The implementation of the proposed method can be used as part of software systems that perform automated monitoring of the integrity of the program code of FPGA-based components.

Key words: *integrity monitoring; FPGA; LUT; digital watermark; steganographic path; programmable computing units.*

Постановка проблемы. Важной составляющей безопасности компьютерных систем является обеспечение их целостности, которое состоит в исключении непредусмотренных изменений системы или предоставляемых ею сервисов [1]. Большинство современных компьютерных систем содержат в своем составе программируемые компоненты: микропроцессоры, микроконтроллеры, программируемые логические интегральные схемы (далее – ПЛИС) [2]. Функционирование систем такого рода обеспечивается как физическими связями компонентов, так и совокупностью программных кодов, настраивающих компоненты на решение требуемых задач. В отличие от компонентов с жесткой логикой функционирования обеспечение целостности программируемых компонентов усложняется в силу двух факторов:

1) относительной простоты процесса внесения изменений в программный код компонентов;

2) необходимости внесения легитимных (разрешенных) изменений программного кода, которые выполняются на всех этапах жизненного цикла системы.

В этих условиях нарушение целостности программного кода компонентов может быть замаскировано под его легитимное изменение или входить в состав легитимного изменения как закамуфлированная часть [3].

В работе [4] рассматривается проблема обеспечения целостности программного кода микросхем типа FPGA (Field Programmable Gate Array), которые являются наиболее современной и часто используемой разновидностью ПЛИС. Структура таких микросхем представляет собой двухмерную матрицу элементарных программируемых блоков нескольких видов: настраиваемых и специализированных вычислителей, блоков распределенной и сосредоточенной памяти, блоков ввода-вывода. Программный код микросхем FPGA задает настройку каждого из блоков матрицы на выполнение конкретной функции, а также формирует систему связей между блоками.

Несмотря на наличие во многих микросхемах FPGA встроенных механизмов защиты программного кода от перезаписи [5], существуют способы обхода такой защиты [6], позволяющие вносить в программный код нелегитимные изменения. В силу этого наиболее распространенным подходом к обеспечению целостности программного кода FPGA-базированных компонентов вы-

ступают приемы, основанные на использовании блоков дополнительной контрольной информации, которые позволяют сделать выводы о целостности кода.

Анализ последних исследований и публикаций. Наиболее часто применяемые подходы к контролю целостности программного кода основаны на применении контрольных хэш-сумм [7]. Хэш-суммы для этих целей вычисляют при помощи криптографических хэш-функций [8], основная особенность которых состоит в необратимости, то есть относительной простоте получения значения функции по ее аргументу, но крайней вычислительной сложности получения аргумента по значению функции. Все практически используемые подходы к контролю целостности базируются на двукратном вычислении хэш-суммы с последующим сравнением результатов этих вычислений [9]. Последовательность действий относительно контроля целостности при этом образует две стадии: стадию подготовки программного кода к контролю и непосредственно стадию контроля.

Стадия подготовки программного кода P к контролю:

1) для информационного объекта программного кода P при помощи заданной хэш-функции вычисляется хэш-сумма SP;

2) полученная хэш-сумма некоторым оговоренным образом присоединяется к информационному объекту P или сопоставляется с ним. В результате формируется информационный объект $CP = (P, SP)$, включающий в себя программный код и его хэш-сумму.

Стадия контроля целостности информационного объекта CP^*:

1) информационный объект $CP^* = (P^*, SP_1^*)$ разделяется на составляющие его программный код P^* и хэш-сумму SP_1^*.

2) для информационного объекта программного кода P^* при помощи заданной хэш-функции вычисляется хэш-сумма SP_2;

3) вновь вычисленная хэш-сумма SP_2 сравнивается с хэш-суммой SP_1^*, извлеченной из проверяемого объекта. В случае несовпадения этих хэш-сумм целостность считается нарушенной.

Недостаток традиционных методов контроля целостности программных кодов заключается в том, что контрольная хэш-сумма SP, сформированная на стадии подготовки информационного объекта к контролю, хранится либо полностью открыто, либо может быть найдена в структуре информационного объекта в результате его анализа. Так, известен подход [10], в рамках которого контрольная хэш-сумма размещается в памяти рядом с контролируемым программный кодом. Другой часто используемый подход [11] предполагает включение хэш-суммы в информационный объект программного кода в качестве одного из его полей. Указанные подходы не позволяют скрыть факт того, что контроль целостности выполняется и не дает возможности скрыть контрольную информацию, на основании которой этот

контроль проводится. Существует также подход [12], в рамках которого контрольная хэш-сумма присоединяется к информационному объекту программного кода не в открытом, а в зашифрованном виде при помощи оговоренного криптографического алгоритма. Такой подход скрывает контрольную информацию, но оставляет открытым для внешнего наблюдателя сам факт того, что контроль целостности производится. Это приводит к возможности применения достаточно большого спектра приемов (от полного перебора до методов социальной инженерии) для открытия и фальсификации хэш-суммы.

Перспективный подход к хранению контрольной хэш-суммы, устраняющий приведенные выше недостатки, заключается в том, что хэш-сумма не присоединяется к информационному объекту P программного кода, а погружается в него в виде цифрового водяного знака (далее – ЦВЗ) [13–15]. Достоинства такого подхода состоят в том, что:

1) ЦВЗ не увеличивает объем информационного объекта на размер хэш-суммы;

2) отсутствует возможность для внешнего наблюдателя выявить факт того, что программный код контролируется, а также выделить часть информационного объекта, которая содержит полезный программный код и часть, которая сдержит хэш-сумму;

3) внедрение ЦВЗ в программный код выполняется таким образом, что функционирование микросхемы FPGA не претерпевает изменений.

Методы [13–15], реализующие концепцию применения ЦВЗ для контроля программного кода FPGA-базированных компонентов, предполагают использование программных кодов блоков LUT (Look Up Table) в качестве среды, в которую погружается ЦВЗ. Блоки LUT являются наиболее массовыми элементарными вычислительными единицами FPGA. Их количество в современных микросхемах FPGA может варьироваться от десятков тысяч до нескольких миллионов. Блок LUT представляет собой программируемый вычислитель логической функции от n-аргументов (обычно от 4 до 8). Настройка блока LUT на реализацию конкретной логической функции производится при помощи 2^n-разрядного программного кода. В соответствии с положениями методов [13–15] совокупность программных кодов блоков LUT используется в качестве среды внедрения контрольного ЦВЗ. Внедрение в рамках указанных методов выполняется с использованием эквивалентных преобразований [16; 17], которые не изменяют реализуемые блоками LUT логические функции и не влияют на функционирование микросхемы FPGA. Методы [13–15] определяют, что для внедрения ЦВЗ из множества блоков LUT формируется упорядоченное множество блоков,

каждый элемент которых однозначно соответствует разряду ЦВЗ и используется для внедрения этого разряда. Указанное упорядоченное множество называется стеганографическим путем (стего-путем) в среде LUT-ориентированного информационного объекта. При этом правило формирования стего-пути выступает частью стего-ключа – совокупности секретной информации, определяющей формальные правила извлечения ЦВЗ из программных кодов блоков LUT. Методы [13–15] не конкретизируют способ формирования стего-пути и не определяют возможные ограничения, накладываемые на его элементы.

Целью статьи является формализация метода формирования стего-пути как части задачи внедрения ЦВЗ в программный код FPGA-базированных компонентов в процессе контроля целостности.

Изложение основного материала. Предлагаемый метод выполняет формирование стего-пути в LUT-контейнере. Под LUT-контейнером микросхемы FPGA понимается совокупность: а) блоков LUT этой микросхемы; б) связей между этими блоками; в) связей между блоками LUT и выводами микросхемы FPGA; г) программных кодов, поставленных в соответствие каждому из блоков LUT. LUT-контейнер является частью структуры микросхемы FPGA, непосредственно задействованной в процессе внедрения ЦВЗ.

Входные данные метода:

а) информационная модель LUT-контейнера [18], содержащая структурированную информацию о его компонентах;

б) контрольный ЦВЗ, предназначенный для внедрения в LUT-контейнер – двоичная последовательность $W = \langle w_1, \dots, w_n \rangle$, содержащая хэш контролируемого программного кода;

в) стего-ключ – секретный набор формально заданных правил внедрения ЦВЗ в LUT-контейнер.

Задача метода: получить упорядоченное множество (последовательность) блоков LUT, предназначенных для непосредственного внедрения разрядов ЦВЗ.

Выходные данные метода (результаты): стего-путь в LUT-контейнере, представляющий собой упорядоченное множество элементов:

$$P = \langle p_1, \dots, p_n \rangle, \quad (1)$$

в котором каждый элемент p_i описывает отдельный блок LUT микросхемы FPGA, а также содержит информацию, необходимую для внедрения одного разряда ЦВЗ в программный код этого блока; каждый элемент p_i однозначно соответствует разряду ЦВЗ w_i при $i = 1 \dots n$; p_i представляет собой тройку элементов:

$$p_i = <id_{\mathrm{LUT}i}, code_{\mathrm{LUT}i}, DList_{\mathrm{LUT}i}>, \quad (2)$$

где $id_{\mathrm{LUT}i}$ – идентификатор блока LUT;

$code_{\mathrm{LUT}i}$ – программный код блока LUT $id_{\mathrm{LUT}i}$;

$DList_{\mathrm{LUT}i}$ – список блоков LUT, источником входных данных для которых является блок с идентификатором $id_{\mathrm{LUT}i}$.

Первое положение метода заключается во введении отношения порядка на множестве элементов стего-пути за счет использования: а) уникальных идентификаторов блоков LUT, заданных информационной моделью LUT-контейнера; б) правила упорядочивания элементов, заданных стего-ключом.

Информационная модель [18] LUT-контейнера формируется как результат получения данных о FPGA-проекте при помощи соответствующей системы автоматизированного проектирования (далее – САПР) и программного обеспечения, выполняющего построение информационной модели. САПР-производителей микросхем FPGA идентифицируют блоки LUT в матрице микросхемы при помощи числового идентификатора. Этот идентификатор представляет собой натуральное число, сопоставленное блоку LUT и являющееся уникальным в пределах матрицы FPGA. Наличие такого идентификатора дает возможность ввести отношение порядка на множестве блоков LUT-контейнера.

Правило упорядочивания блоков LUT-контейнера задается одним из компонентов стего-ключа. Этот компонент формально описывает порядок обхода блоков LUT-контейнера в процессе внедрения ЦВЗ.

Второе положение метода заключается в том, что при формировании стего-пути учитываются *естественные* ограничения, накладываемые структурой LUT-контейнера на выбор блоков, включаемых в стего-путь.

Указанные ограничения обусловлены спецификой эквивалентных преобразований [16; 17] программных кодов блоков LUT, в результате выполнения которых осуществляется внедрение разрядов ЦВЗ в эти программные коды. При внедрении разряда ЦВЗ в программный код некоторого блока LUT_k производится условное инвертирование разрядов программного кода для достижения требуемого значения в заданном его разряде. Для компенсации этого инвертирования выполняется модификация программных кодов всех блоков LUT, входы которых подключены к выходу блока LUT_k (далее – блок LUT_k), в который выполняется внедрение разряда ЦВЗ, будет называться *целевым блоком*, а множество блоков LUT, получающих входные данные с выхода целевого блока LUT_k, *компенсирующим множеством блоков для блока* LUT_k). Если же блок LUT_k подключен к выходу микросхемы FPGA, то компенсирующая модификация программных кодов становится невозможной из-за отсутствия блоков LUT, находящихся между выходом блока LUT_k и выходом микросхемы FPGA. В силу этого данное

положение метода определяет ограничение, заключающееся в невозможности использования в качестве элементов стего-ключа блоков LUT, выходы которых непосредственно подключены к выходам микросхемы FPGA.

Третье положение метода заключается в том, что при формировании стего-пути учитываются *искусственные* ограничения, накладываемые стего-ключом на выбор блоков, включаемых в стего-путь.

Эквивалентные преобразования [16; 17], реализующие непосредственное внедрение разрядов ЦВЗ, требуют модификации программного кода как целевого блока, так и программных кодов всех компенсирующих блоков, относящихся к данному целевому блоку. В этих условиях может быть нецелесообразным внедрение разряда ЦВЗ в целевой блок, выход которого подключен к большому количеству компенсирующих блоков. Для управления возможностью внедрения разряда ЦВЗ в подобные целевые блоки стего-ключ может содержать компонент, задающий ограничивающий порог количества элементов компенсирующего множества блоков. Все целевые блоки LUT, у которых количество подключенных к их выходу компенсирующих блоков превышает ограничивающее значение, не включаются в состав стего-пути. Указанное искусственное ограничение, задаваемое компонентом стего-ключа, не является единственно возможным. Метод допускает наличие иных подобных ограничений, применяемых в соответствии с данным положением метода.

Четвертое положение метода заключается в том, что в процессе формирования стего-пути в его состав не может быть включен блок LUT, выход которого подключен к входу хотя бы одного блока LUT, ранее включенного в стего-путь.

Обоснование данного положения состоит в следующем. Пусть некоторый блок LUT_a включен в состав стего-пути. Пусть также выход другого блока LUT_b подключен к одному из входов блока LUT_a. Необходимо принять решение о возможности включения блока LUT_b в состав стего-пути. На этапе внедрения ЦВЗ в программный код блока LUT_a будет встроен один из разрядов ЦВЗ. В этом случае при внедрении разряда ЦВЗ в блок LUT_b потребуется выполнить модификацию программного кода, как самого блока LUT_b, так и всех его компенсирующих блоков, принимающих данные с выхода блока LUT_b. В число таких блоков, по условию описываемой ситуации, входит и блок LUT_a, в программный код которого уже внедрен разряд ЦВЗ. Модификация программного кода блока LUT_a приведет к искажению ранее внедренного в него разряда ЦВЗ. Для предотвращения такого искажения блок LUT_b не должен входить в состав блоков, в которые внедряются разряды ЦВЗ. Следовательно, в заданной ситуации блок LUT_b не может входить в состав стего-пути.

Предлагаемый метод включает следующую последовательность действий.

Этап 1. Из множества L идентификаторов блоков LUT-контейнера формируется упорядоченное множество идентификаторов:

$$L_{order} = <id_1, id_2, \dots, id_m>.$$

Правило упорядочивания задается компонентом стего-ключа, определяющего порядок обхода блоков LUT-контейнера в процессе внедрения ЦВЗ (первое положение метода).

Далее этапы 2–5 выполняются последовательно для каждого из элементов множества L_{order}.

Этап 2. Для текущего блока LUT с идентификатором id_j, где $j = 1 \dots m$, выполняется проверка естественных ограничений, накладываемых структурой LUT-контейнера на возможность использования этого блока в составе стего-пути (второе положение метода). Для этого производится проверка того, не имеет ли выход блока LUT с идентификатором id_j непосредственных подключений к выводам микросхемы FPGA? Если такие подключения имеются, то блок id_j в состав стего-пути не включается, и выполняется переход к анализу следующего блока из множества L_{order} (возвращение на этап 2). Иначе, осуществляется переход к этапу 3.

Этап 3. Для текущего блока LUT с идентификатором id_j, где $j = 1 \dots m$, выполняется проверка искусственных ограничений, накладываемых стего-ключом на возможность использования этого блока в составе стего-пути (третье положение метода). Если блок id_j не удовлетворяет ограничивающее условие стего-ключа, то этот блок в состав стего-пути не включается, а выполняется переход к анализу следующего блока из множества L_{order} (возвращение на этап 2). Иначе, осуществляется переход к этапу 4.

Этап 4. Для текущего блока LUT с идентификатором id_j, где $j = 1 \dots m$, выполняется проверка того, имеются ли в LUT-контейнере блоки, подключенные к выходу блока id_j и при этом уже включенные в состав стего-пути (четвертое положение метода)? Если такие блоки имеются, то блок id_j в состав стего-пути не включается, а выполняется переход к анализу следующего блока из множества L_{order} (возвращение на этап 2). Иначе, осуществляется переход к этапу 5.

Этап 5. Идентификатор блока id_j включается в состав стего-пути (включается в упорядоченное множество P (1), описывающее стего-путь). Если id_j – последний обработанный элемент множества L_{order}, то стего-путь сформирован и полное его описание содержится во множестве P (1). Иначе, выполняется переход к обработке следующего элемента множества L_{order} (переход на этап 2).

Предлагаемый в данной статье метод формирования стего-пути в пространстве LUT-контейнера был реализован программно. Реализация выполнялась на языке C# в рамках программной платформы Net. В среде разработанного программного обеспечения выполнено тестирование предлагаемого метода. В качестве целевых микросхем использовались микросхемы FPGA Intel (Altera) Cyclone II–IV [19]. Для получения информационной модели LUT-контейнера была применена САПР Intel Quartus Prime [20] совместно с разработанными собственными программными модулями. Результаты тестирования свидетельствуют о целесообразности использования реализации предлагаемого метода в составе системы контроля целостности программного кода FPGA-базированных компонентов.

Выводы по данному исследованию и перспективы дальнейших исследований в данном направлении. Предложенный в статье метод позволяет выполнить построение стего-пути в пространстве LUT-контейнера. Метод позиционируется как составная часть решения задачи контроля целостности программных кодов FPGA-базированных компонентов путем использования ЦВЗ. В исследовании обоснованы основные положения метода и сформулированы этапы его выполнения. Предложена программная реализация метода, основанная на анализе информационной модели LUT-контейнера, получаемой при помощи САПР Intel Quartus Prime. Реализация метода может найти применение в составе программных систем, выполняющих автоматизированный контроль целостности программного кода FPGA-базированных компонентов.

Дальнейших исследований требует вопрос построения в пространстве LUT-контейнера оптимального в соответствии с заданными параметрами стего-пути. Например, построения стего-пути, обеспечивающего для заданного LUT-контейнера внедрение ЦВЗ максимального объема, или стего-пути, использующего минимальное количество блоков LUT (целевых и компенсирующих) для внедрения ЦВЗ фиксированного объема.

Список использованных источников:

1. *Бохмач Е. С., Герасименко А. Д., Головир В. А.* Отказобезопасные информационно-управляющие системы на программируемой логике / под ред. В. С. Харченко, В. В. Скляра. Харьков : НАУ “ХАИ”, НПП “Радий”, 2008. 380 с.

2. *Vanderbauwhede W., Benkrid K.* High-performance computing using FPGAs. New-York: Springer, 2016. 525 p.

3. *Mishra P., Bhunia S., Tehranipoor M.* Hardware IP Security and Trust. New-York: Springer, 2017. 353 p.

4. *Rodriguez Andina Juan Jose, Torre Arnanz Eduardo de la, Dolores Valdes Maria* FPGAs: Fundamentals, Advanced Features, and Applications in Industrial Electronics. CRC Press, 2017. 450 p.

5. Using the Design Security Features in Intel FPGAs. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an556.pdf

6. Handbook of FPGA design security / T. Huffmire, C. Irvine, T. D. Nguyen, T. Levin, R. Kastner, T. Sherwood. Dordrecht: Springer, 2013. 178 p.

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

8. *Stallings W.* Cryptography and Network Security: Principles and Practice, 7th Edition. United Kingdom, Harlow: Pearson Education Limited, 2017. 768 p.

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

10. *Katz J., Lindell Y.* Introduction to Modern Cryptography, Second Edition Hall/CRC, 2014. 554 p.

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

12. *Katz J.* Digital signatures. New York: Springer, 2010. 192 p.

13. *Zashcholkin K., Ivanova O.* The Control Technology of Integrity and Legitimacy of LUT-Oriented Information Object Usage by Self-Recovering Digital Watermark // CEUR Workshop Proceedings. 2015. Vol. 1356. P. 486–497.

14. *Zashcholkin K., Ivanova O.* LUT-object integrity monitoring methods based on low impact embedding of digital watermark // Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET-2018): Proceedings of the 14th International Conference. Lviv-Slavske, 2018. P. 519–523.

15. Checkable FPGA Design: Energy Consumption, Throughput and Trustworthiness / A. Drozd, S. Antoshchuk, J. Drozd, K. Zashcholkin, M. Drozd, N. Kuznietsov, M. Al-Dhabi, V. Nikul // In book: Green IT Engineering: Social, Business and Industrial Applications, Studies in Systems, Decision and Control / V. Kharchenko, Y. Kondratenko, J. Kacprzyk (Edits). Berlin, Heidelberg: Springer International Publishing, 2018. P. 73–94, DOI: 10.1007/978-3-030-00253-4_4.

16. *Drozd A., Drozd M., Kuznietsov M.* Use of natural LUT redundancy to improve trustworthiness of FPGA design // CEUR Workshop Proceedings. 2016. Vol. 1614. P. 322–331.

17. *Drozd O., Drozd M., Martynyuk O., Kuznietsov M.* Improving of a circuit checkability and trustworthiness of data processing results in LUT-based FPGA components of safety-related systems // CEUR Workshop Proceedings. 2017. Vol. 1844. P. 654–661.

18. *Защелкин К. В., Иванова Е. Н.* Поиск целевых блоков LUT в информационной модели FPGA-устройства в рамках задачи контроля целостности программного кода // Електротехнічні та комп'ютерні системи. 2018. № 28 (104). С. 215–222.

19. Intel Cyclone IV Device Handbook. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyclone4-handbook.pdf

20. Intel Quartus Prime Standard Edition Handbook. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/qts/archives/qts-qps-handbook-16.0.pdf

References:

1. Bokhmach E.S., Gerasimenko A.D. and Golovir V.A. (2008), *Otkazobezopasnyye informatsionno-upravlyayushchiye sistemy na programmiruyemoy logike* [Fail-safe information control systems on programmable logic] / ed. by V. S. Kharchenko, V. V. Sklyar. Press NAU "KhAI", NPP "Radiy", Kharkov. 380 p. [Ukraine].

2. Vanderbauwhede W. and Benkrid K. (2016), High-performance computing using FPGAs. New-York: Springer, 525 p. [USA].

3. Mishra P., Bhunia S. and Tehranipoor M. (2017), Hardware IP Security and Trust. New-York: Springer, 353 p. [USA].

4. Rodriguez Andina Juan Jose, Eduardo de la Torre Arnanz and Dolores Valdes Maria (2017), FPGAs: Fundamentals, Advanced Features, and Applications in Industrial Electronics. CRC Press, 450 p.

5. Using the Design Security Features in Intel FPGAs (2019), available at: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an556.pdf

6. Huffmire T., Irvine C., Nguyen T. D. and Levin T. et al (2013), Handbook of FPGA design security, Sherwood, Dordrecht: Springer, 178 p.

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

8. Stallings W. (2017), Cryptography and Network Security: Principles and Practice, 7th Edition. United Kingdom, Harlow: Pearson Education Limited, 768 p.

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

10. Katz J. and Lindell Y. (2014), Introduction to Modern Cryptography, Second Edition Hall/CRC, 554 p.

11. Ferguson N., Schneier B. and Kohno T. (2013), Cryptography engineering. Hoboken: Wiley, 354 p.

12. Katz J. (2010), Digital signatures. New York: Springer, 192 p.

13. Zashcholkin K. and Ivanova O. (2015), "The Control Technology of Integrity and Legitimacy of LUT-Oriented Information Object Usage by Self-Recovering Digital Watermark" // CEUR Workshop Proceedings, vol. 1356, pp. 486–497.

14. Zashcholkin K. and Ivanova O. (2018), "LUT-object integrity monitoring methods based on low impact embedding of digital watermark" // Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET-2018), Proceedings of the 14th International Conference. Lviv-Slavske, pp. 519–523.

15. Drozd A., Antoshchuk S., Drozd J. and Zashcholkin K. et al (2018), Checkable FPGA Design: Energy Consumption, Throughput and Trustworthiness // In book: Green IT Engineering: Social, Business and Industrial Applications, Studies in Systems, Decision and Control / V. Kharchenko, Y. Kondratenko, J. Kacprzyk (Edits). Berlin, Heidelberg: Springer International Publishing, pp. 73–94, DOI: 10.1007/978-3-030-00253-4_4.

16. Drozd A., Drozd M. and Kuznietsov M. (2016), "Use of natural LUT redundancy to improve trustworthiness of FPGA design" // CEUR Workshop Proceedings, vol. 1614, pp. 322–331.

17. Drozd O., Drozd M., Martynyuk O. and Kuznietsov M. (2017), "Improving of a circuit checkability and trustworthiness of data processing results in LUT-based FPGA components of safety-related systems" // CEUR Workshop Proceedings, vol. 1844, pp. 654–661.

18. Zaschelkin K. V. and Ivanova E. N. (2018), *"Poisk tselevykh blokov LUT v informatsionnoy modeli FPGA-ustroystva v ramkakh zadachi kontrolya tselostnosti programmnogo koda"* ["Search for target LUT blocks in the information model of an FPGA device as part of the task of monitoring the integrity of a program code"] // Journal *Elektrotekhnichní ta komp'yuterní systemy* [Electronic Systems and Computer Systems], vol. 28(104), pp. 215–222 [Ukraine].

19. Intel Cyclone IV Device Handbook (2019), available at: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyclone4- handbook.pdf

20. Intel Quartus Prime Standard Edition Handbook (2019), available at: https://www:intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/qts/archives/qts-qps-handbook-16.0.pdf