УДК 004.056.53

Informatics and Mathematical Methods in Simulation
Vol. 3 (2013), No. 4, pp. 369-384

МЕТОД ВНЕДРЕНИЯ ЦИФРОВЫХ ВОДЯНЫХ ЗНАКОВ В АППАРАТНЫЕ КОНТЕЙНЕРЫ С LUT-ОРИЕНТИРОВАННОЙ АРХИТЕКТУРОЙ

К.В. Защелкин, Е.Н. Иванова

Одесский национальный политехнический университет,
просп. Шевченко, 1, Одесса, Украина; e-mail: const-z@te.net.ua

Рассмотрена задача внедрения цифровых водяных знаков в информационный объект с целью контроля его использования. Отмечены типичные подходы к организации такого внедрения. Предложен метод внедрения цифровых водяных знаков в аппаратные контейнеры с LUT-ориентированной архитектурой. Показаны алгоритмы реализации предложенного метода. Описана аппаратно-программная реализация метода и результаты экспериментов в среде этой реализации. Показаны возможности применения предложенного метода для организации цифровых водяных знаков в пространстве LUT-ориентированного контейнера с целью контроля его использования в динамике проектирования и жизненного цикла.

Ключевые слова: цифровые водяные знаки, стеганография, защита информации, аппаратный стего-контейнер, LUT-ориентированная архитектура, FPGA, контроль использования FPGA-проектов

Введение

Цифровой водяной знак (ЦВЗ) представляет собой данные, внедряемые в информационный объект с целью контроля его использования. Технология ЦВЗ основана на применении стеганографических приемов, в рамках которых скрывается факт наличия ЦВЗ в информационном объекте (контейнере). При этом ЦВЗ может быть считан из контейнера при наличии стего-ключа, определяющего правила доступа к элементам ЦВЗ [1, 2].

В современных информационных системах ЦВЗ получили широкое распространение для контроля использования мультимедийного контента: растровых графических файлов, видеофайлов, оцифрованного звука [3, 4]. Существенная особенность файлов-контейнеров для такого контента состоит в том, что все они являются *пассивными* информационными объектами, выполняющими только функцию хранения данных. Очевидно, что необходимость в контроле использования информационных объектов не ограничивается только контейнерами данного вида. Такая необходимость имеет место и для *активных* информационных объектов выполняющих некоторую вычислительную или управляющую функцию.

В последнее время активизировались исследования в области использования нетрадиционных активных стего-контейнеров, как для непосредственно стеганографических задач (скрытой передачи и хранения защищенной информации), так и для задач внедрения ЦВЗ в такие контейнеры. В частности появились работы предлагающие использовать в качестве стего-контейнеров или объектов для внедрения ЦВЗ исполняемые файлы [5–7] или исходные коды программ [8, 9] микропроцессоров и микроконтроллеров.

В данной работе предлагаются подходы к внедрению ЦВЗ в аппаратные контейнеры, построенные на основе LUT-ориентированной архитектуры (далее LUT-контейнеры). К таким контейнерам относятся, например, микросхемы FPGA (*Field Programmable Gate Array*), являющиеся на текущий момент весьма используемой элементной базой для построения компьютерных и управляющих систем. По многим параметрам FPGA конкурируют с микропроцессорами и микроконтроллерами, а по параметрам производительности и возможности организации параллельных вычислений превосходят их [10].

Постановка цели работы

На текущий момент, элементная база, основанная на LUT-ориентированной архитектуре (например, микросхемы FPGA) является крайне востребованной при построении компьютерных и управляющих систем. Исходя из этого, можно констатировать перспективность исследования возможности внедрения информации в LUT-контейнеры с целью контроля их использования, а так же для организации стего-защищенных систем передачи и хранения данных на их основе. *Цель* данной работы состоит в развитии технологии цифровых водяных знаков путем ее распространения на LUT-контейнеры.

Особенности контейнеров с LUT-ориентированной архитектурой

LUT (*Look Up Table* – таблица поиска) представляет собой структуру данных, используемую с целью заменить вычисления на операцию поиска заготовленных данных [10]. Подход, основанный на применении LUT, получил название «Вычисления с памятью» (*Computing with Memory*) [11]. Наибольшего своего развития этот подход достиг в структуре программируемых логических интегральных схем (ПЛИС), в частности в наиболее современной их разновидности FPGA [12]. Упрощенно архитектура FPGA представляет собой совокупность вычислительных модулей, упорядоченных в виде двухмерной матрицы. Основную вычислительную функцию этих модулей выполняют блоки LUT, имеющие обычно 4 (реже 5 или 6) входов и 1 или 2 выхода. Блоки LUT могут быть определенным образом соединены между собой, а так же со специализированными модулями (памяти, аппаратного умножения) и выводами микросхемы. Определенная конфигурация соединения блоков LUT, а так же запись в них определенного содержимого приводит к организации, требуемой для данной задачи, вычислительной среды.

Блоки LUT в FPGA обычно представляют собой одноразрядную оперативную память. Входы блока LUT при этом являются адресными входами такой памяти (рис. 1). При количестве входов, равном n, блок LUT хранит в себе 2^n бит информации и способен выполнить вычисление значения одной n-аргументной булевой функции.

Предлагаем рассматривать LUT-контейнер как четверку вида:

$$LC = \left(L, E, Interface, Extern\right), \tag{1}$$

где

L — множество блоков LUT;

E — множество связей между элементами множества L;

Interface — множество входов и выходов контейнера;

Extern — множество связей блоков LUT с входами и выходами контейнера.

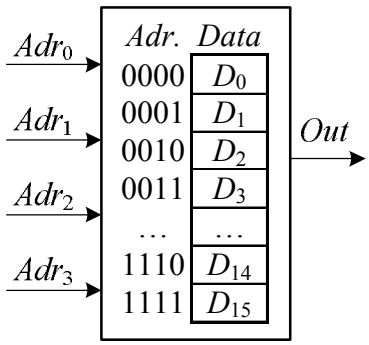

Рис. 1. Организация 4-х входового блока LUT микросхемы FPGA

Каждый элемент LUT_i множества L представляет собой следующую тройку:

$$LUT_i = \left(In_i, Out_i, Mem_i\right), \tag{2}$$

где

In_i — множество входов блока LUT_i;

Out_i — множество выходов блока LUT_i;

Mem_i — содержимое памяти блока LUT_i (внутреннее значение блока).

Таким образом, LUT-контейнер фактически представляет собой логическую схему, элементами которой являются настраиваемые блоки LUT. Следует отметить следующие особенности LUT-контейнеров, отличающие их от мультимедийных контейнеров, используемых в традиционных стеганографических методах.

1) LUT-контейнер является активным информационным объектом, выполняющим некоторую вычислительную или управляющую функцию.

2) Элементарные единицы контейнера (блоки LUT) в общем случае явно связанны друг с другом т.к. могут вычислять часть общей булевой функции. Таким образом, элементарные единицы контейнера не являются автономными, а оказывают взаимное влияние на функционирование друг друга.

3) Информация, находящаяся в каждой из элементарных единиц контейнера представляется точно. Произвольное изменение содержимого блока LUT приводит к разрушению контейнера, которое выражается в невозможности выполнения им целевой функции.

Далее предлагается метод внедрения ЦВЗ в LUT-контейнер. Метод основан на указанных особенностях контейнера, позволяющих выполнить его эквивалентное преобразование, сопряженное с внедрением секретных данных.

Основные положения предлагаемого метода внедрения данных в LUT-контейнер

Основное содержание метода далее излагается в виде совокупности специальных положений, определяющих последовательность и условия реализации процесса скрытия данных в LUT-контейнере.

Первое положение метода: для встраивания одного разряда секретной двоичной последовательности используется один из разрядов блока LUT, задействованного в выполнении целевой функции контейнера. Номер этого разряда (адрес) или правило его определения является элементом стего-ключа.

Второе положение метода: встраивание разряда секретной последовательности основано на следующем утверждении.

Пусть задана некоторая булева функция:

$$y = f\left(y_1^{\lambda_1}, y_2^{\lambda_2} \ldots, y_m^{\lambda_m}, a_1, a_2, \ldots, a_p\right). \quad (3)$$

При этом часть аргументов данной функции являются результатом вычислений значений следующей совокупности булевых функций:

$$\begin{gathered} y_1^{\lambda_1} = f_1^{\lambda_1}\left(x_1^1, x_2^1, \ldots, x_{n_1}^1\right) \\ y_2^{\lambda_2} = f_2^{\lambda_2}\left(x_1^2, x_2^2, \ldots, x_{n_2}^2\right) \\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots \\ y_m^{\lambda_m} = f_m^{\lambda_m}\left(x_1^m, x_2^m, \ldots, x_{n_m}^m\right) \end{gathered} \quad (4)$$

и для функций (3) и (4) установлена следующая система правил:

$$\begin{gathered} y_i^{\lambda_i} = \begin{cases} y_i, \text{ при } \lambda_i = 1; \\ \overline{y_i}, \text{ при } \lambda_i = 0; \end{cases} \\ f_i^{\lambda_i}\left(x_1^i, x_2^i, \ldots, x_{n_i}^i\right) = \begin{cases} f_i^{\lambda_i}\left(x_1^i, x_2^i, \ldots, x_{n_i}^i\right), \text{ при } \lambda_i = 1; \\ \overline{f_i^{\lambda_i}}\left(x_1^i, x_2^i, \ldots, x_{n_i}^i\right), \text{ при } \lambda_i = 0. \end{cases} \end{gathered} \quad (5)$$

где $\lambda_i \in \{0, 1\}$, $i = 1 \ldots m$.

В выражении (3) допускается отсутствие аргументов $a_1, a_2, \ldots, a_p$.

Утверждение 1. Значения булевой функции (3) при аргументах (4) и правилах (5) на любых двоичных наборах не зависят от значений переменных $\lambda_1, \lambda_2, \ldots, \lambda_m$.

Действительно, при инвертировании любой из функций $f_i^{\lambda_i}$ (4) и сопровождающейся повторным инвертированием (5) подстановке ее значения в функцию (3), в действительности в функцию (3) в соответствии с законом двойного отрицания будет подставлено прямое значение $f_i^{\lambda_i}$, которое имело место до инвертирования. Таким образом, инвертирование любых функций из (4) с учетом (5) не приводит к изменению значений функции (3).

Из данного утверждения следует, что любые комбинации значений переменных $\lambda_1, \lambda_2, \ldots, \lambda_m$ в системе выражений (3) – (5) порождают равносильные булевы функции (3), дающие одинаковые значения на любых наборах.

Принцип встраивания разряда секретной последовательности связан с приведенным *утверждением* следующим образом. Пусть функцию (3) реализует некоторый блок *LUT'*, а функции (4) некоторый набор блоков LUT_1, LUT_2, …, LUT_m. Тогда выборочным инвертированием значений LUT_1, LUT_2, …, LUT_m можно добиться наличия в каждом из них, таких значений определенных разрядов, которые соответствуют разрядам внедряемой в контейнер секретной двоичной последовательности. При этом значения, вычисляемые блоком *LUT'* с учетом выражений (5) не претерпят каких-либо изменений.

Третье положение метода определяет ограничения на структуру схемы LUT-контейнера, в который внедряется секретная информация. Предлагаемый метод может быть применен только к схеме, имеющей более одного уровня блоков LUT (ранг которой $r > 1$). Действительно, минимальный вариант схемы, совместно реализующей выражения (3) – (5) должен включать набор блоков LUT, расположенных на первом

уровне и предназначенных для вычисления функций (4), а также один или несколько блоков LUT, расположенных на втором уровне и предназначенных для вычисления функции (3).

Четвертое положение метода определяет ограничения на использование блоков LUT, входящих в контейнер, для задачи встраивания секретной последовательности.

Первое ограничение такого рода состоит в том, что для встраивания нельзя использовать блоки LUT, непосредственно подключенные к выходам схемы. Это ограничение обусловлено тем, что встраивание секретной информации выполняется в блоки LUT, реализующие выражения (4), к выходам которых, должны быть подключены блоки LUT следующего уровня, реализующие выражение (3). Если же блок LUT подключен к выходу схемы, то следующие за ним блоки, реализующие выражение (3) отсутствуют.

Второе ограничение состоит в том, что блок LUT, в который уже встроен разряд секретной последовательности, не может быть подвергнут инвертированию при выполнении встраивания данных в другой блок LUT данного контейнера. Из этого ограничения следует, что количество блоков LUT, которые можно задействовать для внедрения разрядов секретной последовательности зависит от порядка обхода контейнера.

Таким образом, первое указанное ограничение задает верхнюю оценку количества блоков LUT, которые можно задействовать для внедрения секретной информации, а второе ограничение – нижнюю оценку. Указанные верхнее и нижнее значения зависят от общего количества блоков, конфигурации их соединения и порядка обхода контейнера в процессе встраивания в него секретной последовательности.

Пятое положение метода определяет порядок формирования стего-ключа для встраивания и извлечения секретной последовательности. Ключ определяется как двойка следующего вида:

$$key = (set, order) \tag{6}$$

где

set – номер (адрес) разряда LUT, в который выполняется внедрение бита секретной последовательности. Вместо фиксированного значения *set*, этот компонент ключа может содержать некоторое правило, позволяющее получить номер разряда встраивания для каждого шага встраивания;

order – порядок обхода блоков LUT в контейнере для выполнения встраивания или извлечения секретной последовательности. Вместо фиксированного порядка обхода блоков LUT этот компонент ключа может содержать правило, задающее порядок обхода на каждом шаге встраивания.

Пример реализации предлагаемого метода

Рассмотрим пример, иллюстрирующий основные положения предлагаемого метода. На рис. 2 (а) представлена схема контейнера, состоящая из пяти блоков LUT и реализующая две логические функции y' и y''. Двоичный вес каждого из входов блоков LUT, обозначен рядом с соответствующим входом. Необходимо внедрить в данную схему, секретную последовательность $M = (1,0,0)$ используя для этого значения, расположенные в блоках LUT по адресу 3. Порядок обхода блоков схемы в ходе встраивания определен их нумераций.

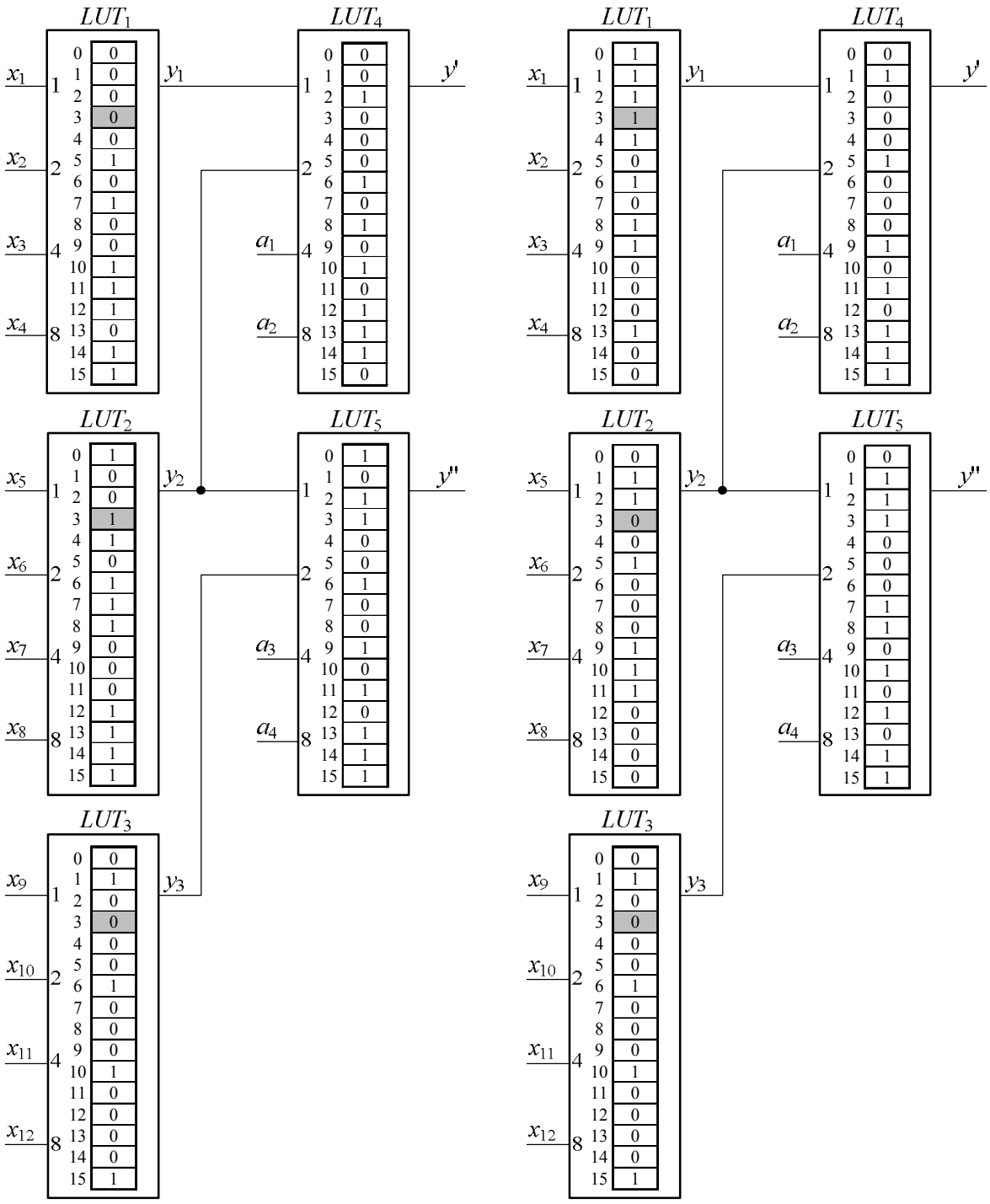

Рис. 2. Пример встраивания секретной двоичной последовательности $M = (1,0,0)$ в LUT-ориентированный контейнер, состоящий из 5-и блоков LUT: а – исходные значения контейнера; б – значения после встраивания

Структура данной схемы соответствует выражениям (3) и (4) для которых может быть применена система правил (5) 2-го положения метода. В соответствии с 3-м положением метода схема, показанная на рис. 2(а) может быть использована в качестве контейнера для встраивания секретной информации, т.к. она имеет два уровня. В

соответствии с 4-м положением для встраивания могут быть задействованы блоки LUT не подключенные к выходам схемы, т.е. блоки LUT_1, LUT_2, LUT_3.

Для внедрения разрядов последовательности будем в порядке нумерации использовать блоки LUT первого уровня. В блоке LUT_1 по адресу 3 хранится значение «0». По этому адресу необходимо поместить значение «1». В соответствии со 2-м положением метода выполним инвертирование всех значений, хранящихся в блоке LUT_1. После этого, в соответствии с правилами (5), выполним инвертирование значения на входе блока LUT, принимающего данные от блока LUT_1, т.е. блока LUT_4. В результате выполнения этих действий, функции, вычисляемые схемой, останутся неизменными, однако в блок LUT_1 по адресу 3 будет внедрен первый разряд секретной последовательности, равный значению «1» (рис. 2(б)).

Аналогичным образом, используя 1-е и 2-е положение предложенного метода, заменяем значение «1», хранящееся в блоке LUT_2 по адресу 3 на второй разряд секретной последовательности «0». Для этого инвертируем все значения, хранящиеся в блоке LUT_2 с одновременным инвертированием значений на входах блоков LUT, принимающих данные от блока LUT_2, т.е. блоков LUT_4 и LUT_5. Для внедрения третьего разряда секретной последовательности «0» в блок LUT_3 нет необходимости выполнять какие-либо изменения значений блоков LUT данной схемы, т.к. в блоке LUT_3 по адресу 3 уже хранится значение 0.

Рассмотренный пример показывает возможность внедрения секретной последовательности в LUT-контейнер в соответствии с предложенным методом. В результате такого внедрения функционирование контейнера не изменяется (схемы, изображенные на рис. 2(а) и рис. 2(б) формируют одинаковые значения на одинаковых входных наборах), т.е. его целевая функция остается неизменной. Однако контейнер становится носителем информации, которую можно использовать в качестве элемента ЦВЗ или для организации стего-защищенных систем передачи и хранения данных.

Алгоритм внедрения данных в контейнер в соответствии с предложенным методом

Основные положения предложенного метода встраивания данных в LUT-контейнер обуславливают рассмотренный далее алгоритм реализации этого метода.

Исходные данные алгоритма:

1) секретная двоичная последовательность $M = (m_1, m_2, \ldots, m_k)$;

2) LUT-контейнер LC, реализующий некоторую вычислительную или управляющую функцию. На множестве блоков LUT в контейнере LC установлено отношение порядка (нумерация, система координат) дающее возможность выполнить обход блоков контейнере в заданном порядке в ходе встраивания разрядов секретной последовательности;

3) ключ *key* для встраивания и извлечения секретной информации.

Результат применения алгоритма – LUT-контейнер LC^*, в который внедрена последовательность *M*. Функционирование контейнера LC^*, выражающееся в выполнении его целевой функции, не отличается от функционирования контейнера *LC*.

Последовательность действий по внедрению данных в LUT-ориентированный контейнер:

1) Определение возможности встраивание последовательности *M* в контейнер *LC* на основании 3-го положения предлагаемого метода. Если схема, содержащаяся в контейнере, имеет более одного уровня, то встраивание возможно.

2) Оценка возможности встраивания последовательности *M* целиком в контейнер *LC*, исходя из количества боков LUT, которые можно использовать для встраивания.

Оценка доли последовательности *M*, которую можно встроить в контейнере, в случае невозможности встроить последовательность целиком.

3) Формирование списков блоков LUT, предназначенных для проверки соблюдения ограничений 4-го положения метода:

- формирование списка *ExternalList*, в который заносятся идентификаторы блоков LUT, непосредственно подключенные к выходам схемы (4-е положение метода, 1-е ограничение);
- формирование списка *BlockList*, в который в ходе движения по контейнеру помещаются идентификаторы заблокированных для встраивания блоков LUT (4-е положение метода, 1-е ограничение).

4) В соответствии с заданным порядком обхода блоков LUT контейнера, с учетом ограничений, определенных 4-м положением метода, каждый разряд секретной последовательности m_j на основании следующего правила встраивается в индивидуальный блок LUT_i:

$$\forall LUT_i \mid \left(LUT_i \notin ExternalList \;\&\; LUT_i \notin BlockList \;\&\; OutList\left(LUT_i\right) \cap BlockList = \varnothing\right)$$
$$\text{if } LUT_i(set) \neq m_j \text{ then } Embed\left(LUT_i, m_j\right); \qquad (7)$$
$$LUT_i \rightarrow BlockList,$$

где

$OutList\left(LUT_i\right)$ — список блоков LUT, подключенных своими входами к выходу блока LUT_i;

set — номер (адрес) разряда LUT, в который выполняется внедрение секретного бита m_j;

Embed — процедура встревания разряда m_j в блок LUT_i по адресу *set*;

«$\rightarrow$» — операция включения идентификатора блока LUT в список.

Внешнее условие правила (7) определяет возможность использования блока для встраивания очередного разряда секретной последовательности (4-е положение метода). Данное сложное условие определяет то, что блок LUT_i, предназначенный для встраивания информации:

- не должен содержаться в списке блоков, непосредственно подключенных к выходам схемы;
- не должен содержаться в списке *BlockList* заблокированных для встраивания блоков;
- его выход не должен быть подключен на вход блоков из списка *BlockList*.

Таким образом, данное сложное условие проверяет отсутствие обоих ограничений, наложенных 4-м положением метода.

Внутреннее условие правила (7) означает, что *Embed* – процедура встревания разряда m_j в блок LUT_i по адресу *set*, выполняется только тогда, когда значение, содержащееся в данном блоке по указанному адресу, не совпадает со значением разряда, который необходимо встроить. В противном случае отсутствует необходимость выполнения встраивания в данный блок LUT_i.

Блоки LUT, не удовлетворяющие внешнему или внутреннему условию правила (7), игнорируются в процессе обхода контейнера. Встраивание информации в них не производится. Блоки LUT, удовлетворяющие внешнему условию, но не удовлетворяющие внутреннему условию правила (7) хранят элементы секретной последовательности, но встраивание в них не производится по причине совпадения исходного их значения со значением, подлежащим встраиванию. В ходе применения правила (7), независимо от того, каким образом в очередной блок LUT был помещен разряд секретной последовательности (при помощи процедуры *Embed* или этот разряд

находился в контейнере изначально, и применение процедуры *Embed* не требовалось), идентификатор данного блока LUT заносится в список заблокированных блоков *BlockList*.

Процедура встраивания Embed.

Процедура *Embed*, выполняющая встраивание разряда m_j в блок LUT_i по адресу *set* состоит из двух действий:

1) инвертирование значений блока LUT_i (каждое из значений, хранящихся в блоке меняется на противоположное);

2) выполнение *процедуры распространение инверсии* на входы всех блоков LUT, содержащихся в списке $OutList(LUT_i)$. Распространение инверсии состоит в инвертировании входа блока LUT. Такое инвертирование сводится к перестановке значений, хранящихся в блоке LUT, и определяется следующим образом.

Введем обозначения. Пусть блок LUT_i имеет набор из n входов $In_i = \left(in_i^{n-1}, in_i^{n-2}, \ldots, in_i^1, in_i^0\right)$. Каждый из входов in_i^k при $k = \overline{0, n-1}$ задает один из разрядов адреса блока LUT, и соответственно имеет вес 2^k. Количество значений, хранящихся в данном блоке LUT равно $N = 2^n$. Обращение (на чтение или на запись) к значению, хранящееся в блоке LUT_i по адресу s определим как $LUT_i(s)$, где s может задаваться в виде двоичного числа, количество разрядов которого, совпадает с количеством входов блока LUT или в виде десятичного эквивалента этого числа.

Следующим образом определим процедуру перестановки значений блока LUT:

$$Permutation(LUT_i, w):\ \forall s\ \ LUT_i\left(s_w^0\right) \leftrightarrow LUT_i\left(s_w^1\right), \tag{8}$$

где

s — любой допустимый адрес для данного блока LUT_i;

s_w^0 и s_w^1 — двоичные наборы, которые различаются только значением в разряде, имеющем вес w, а в остальных разрядах совпадают. Наборы s_w^0 и s_w^1 содержат в разряде с весом w ноль и единицу соответственно;

«$\leftrightarrow$» — операция обмена значений блока LUT (значение, указанное в качестве левого операнда, помещается на место правого операнда и наоборот).

Таким образом, принцип перестановки в ходе распространения инверсии зависит от двоичного веса входа, на который распространяется инверсия. Например, для распространения инверсии на вход блока LUT, имеющий минимальный вес $w = 1$, в соответствии с выражением (8), выполняется взаимный обмен значений, расположенных по адресам, отличающимся только в младшем разряде (с весом 1). Это приводит к обмену значениями, находящимися по ближайшим четным и нечетным адресам: $LUT(0) \leftrightarrow LUT(1)$; $LUT(2) \leftrightarrow LUT(3)$; $LUT(4) \leftrightarrow LUT(5)$ и т.д. Аналогичным образом, распространение инверсии на вход блока LUT, имеющий вес $w = 2$ приводит к взаимному обмену значениями, расположенными по адресам, отличающимся в разряде с весом 2: $LUT(0) \leftrightarrow LUT(2)$; $LUT(1) \leftrightarrow LUT(3)$; $LUT(4) \leftrightarrow LUT(6)$; $LUT(5) \leftrightarrow LUT(7)$ и т.д.

На рис. 3 показаны правила распространения инверсии для 4-х входового блока LUT. Столбцы *A*, *B*, *C*, *D* содержат разряды адреса, подаваемые на соответствующие входы. Веса входов увеличиваются слева направо: *A* – вес 1, *B* – вес 2, *C* – вес 4, *D* – вес 8. В столбце *out* находятся значения, содержащиеся в блоке LUT. На рис. 3 (а) показаны первоначальные значения блока LUT, на рис. 3(б–д) показаны результаты распространения инверсии на входы этого блока *D*, *C*, *B*, *A* соответственно.

Блоки LUT, используемые в современных средствах цифровой техники имеют небольшое количество входов (от 3 до 6). В силу этого нет необходимости выполнять поиск наборов, на которых значения LUT в соответствии с выражением (8) подлежат

обмену. Правила обмена для таких блоков LUT могут быть жестко определены в реализации алгоритма распространения инверсии.

D^8	C^4	B^2	A^1	out
0	0	0	0	h_0
0	0	0	1	h_1
0	0	1	0	h_2
0	0	1	1	h_3
0	1	0	0	h_4
0	1	0	1	h_5
0	1	1	0	h_6
0	1	1	1	h_7
1	0	0	0	h_8
1	0	0	1	h_9
1	0	1	0	h_{10}
1	0	1	1	h_{11}
1	1	0	0	h_{12}
1	1	0	1	h_{13}
1	1	1	0	h_{14}
1	1	1	1	h_{15}

а

D^8	C^4	B^2	A^1	out
0	0	0	0	h_8
0	0	0	1	h_9
0	0	1	0	h_{10}
0	0	1	1	h_{11}
0	1	0	0	h_{12}
0	1	0	1	h_{13}
0	1	1	0	h_{14}
0	1	1	1	h_{15}
1	0	0	0	h_0
1	0	0	1	h_1
1	0	1	0	h_2
1	0	1	1	h_3
1	1	0	0	h_4
1	1	0	1	h_5
1	1	1	0	h_6
1	1	1	1	h_7

б

D^8	C^4	B^2	A^1	out
0	0	0	0	h_4
0	0	0	1	h_5
0	0	1	0	h_6
0	0	1	1	h_7
0	1	0	0	h_0
0	1	0	1	h_1
0	1	1	0	h_2
0	1	1	1	h_3
1	0	0	0	h_{12}
1	0	0	1	h_{13}
1	0	1	0	h_{14}
1	0	1	1	h_{15}
1	1	0	0	h_8
1	1	0	1	h_9
1	1	1	0	h_{10}
1	1	1	1	h_{11}

в

D^8	C^4	B^2	A^1	out
0	0	0	0	h_2
0	0	0	1	h_3
0	0	1	0	h_0
0	0	1	1	h_1
0	1	0	0	h_6
0	1	0	1	h_7
0	1	1	0	h_4
0	1	1	1	h_5
1	0	0	0	h_{10}
1	0	0	1	h_{11}
1	0	1	0	h_8
1	0	1	1	h_9
1	1	0	0	h_{14}
1	1	0	1	h_{15}
1	1	1	0	h_{12}
1	1	1	1	h_{13}

г

D^8	C^4	B^2	A^1	out
0	0	0	0	h_1
0	0	0	1	h_0
0	0	1	0	h_3
0	0	1	1	h_2
0	1	0	0	h_5
0	1	0	1	h_4
0	1	1	0	h_7
0	1	1	1	h_6
1	0	0	0	h_9
1	0	0	1	h_8
1	0	1	0	h_{11}
1	0	1	1	h_{10}
1	1	0	0	h_{13}
1	1	0	1	h_{12}
1	1	1	0	h_{15}
1	1	1	1	h_{14}

д

Рис. 3. Правила распространения инверсии на входы 4-х входового блока LUT: а – исходные значения; б – распространение инверсии на вход D; в – распространение инверсии на вход C; г – распространение инверсии на вход B; д – распространение инверсии на вход A

В общем случае процедура распространения инверсии может иметь место не по одному входу блока LUT, а по нескольким его входам (групповое распространение инверсии). Выполнение такого группового распространения инверсии основано на следующем утверждении.

Утверждение 2. Результат группового распространения инверсии на входы блока LUT не зависит от порядка выполнения процедуры распространения инверсии на отдельные его входы.

Доказательство этого утверждения состоит в следующем. Распространение инверсии основано на перестановке, по правилу (8), значений, хранящихся в блоке LUT. Каждая такая перестановка приводит к взаимному изменению только одного разряда адреса для переставляемых значений. Разряды адреса в ходе выполнения перестановок не зависят друг от друга. В силу этого порядок изменения разрядов по выражению (8) при нескольких последовательных изменениях с разными значениями w не влияет на результат данных изменений. Следовательно, и порядок выполнения распространения инверсии на отдельные входы блока LUT не влияет на получаемый результат.

На рис. 4 графически показаны перестановки значений в ходе распространения инверсии на отдельные входы 4-х входового блока LUT.

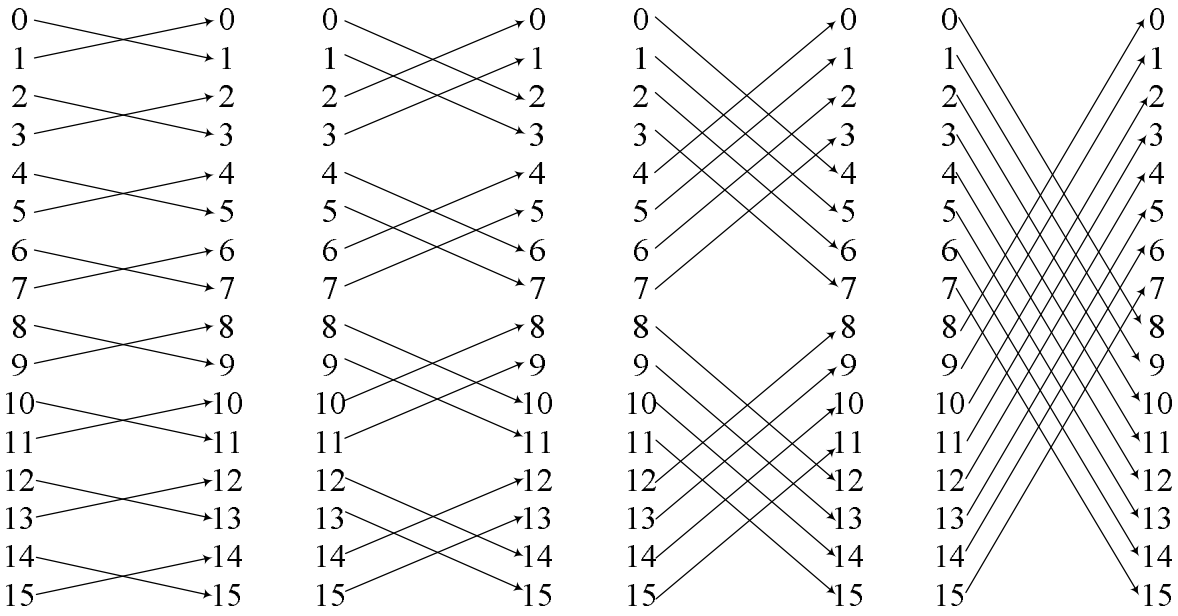

Рис. 4. Графическая интерпретация перестановок, возникающих в ходе распространения инверсии на входы 4-х входового блока LUT: а – распространение на вход с весом 1; б – распространение на вход с весом 2; в – распространение на вход с весом 4; г – распространение на вход с весом 8

Таким образом, на основании утверждения 2, в случае если процедура распространения инверсии осуществляется от нескольких блоков LUT на разные входы одного блока LUT, то необходимо по отдельности выполнить распространение инверсии на каждый вход, причем порядок обработки входов не имеет значения.

Обобщенная блок-схема алгоритма реализации предложенного метода встраивания секретной двоичной последовательности в LUT-контейнер приведена на рис. 5.

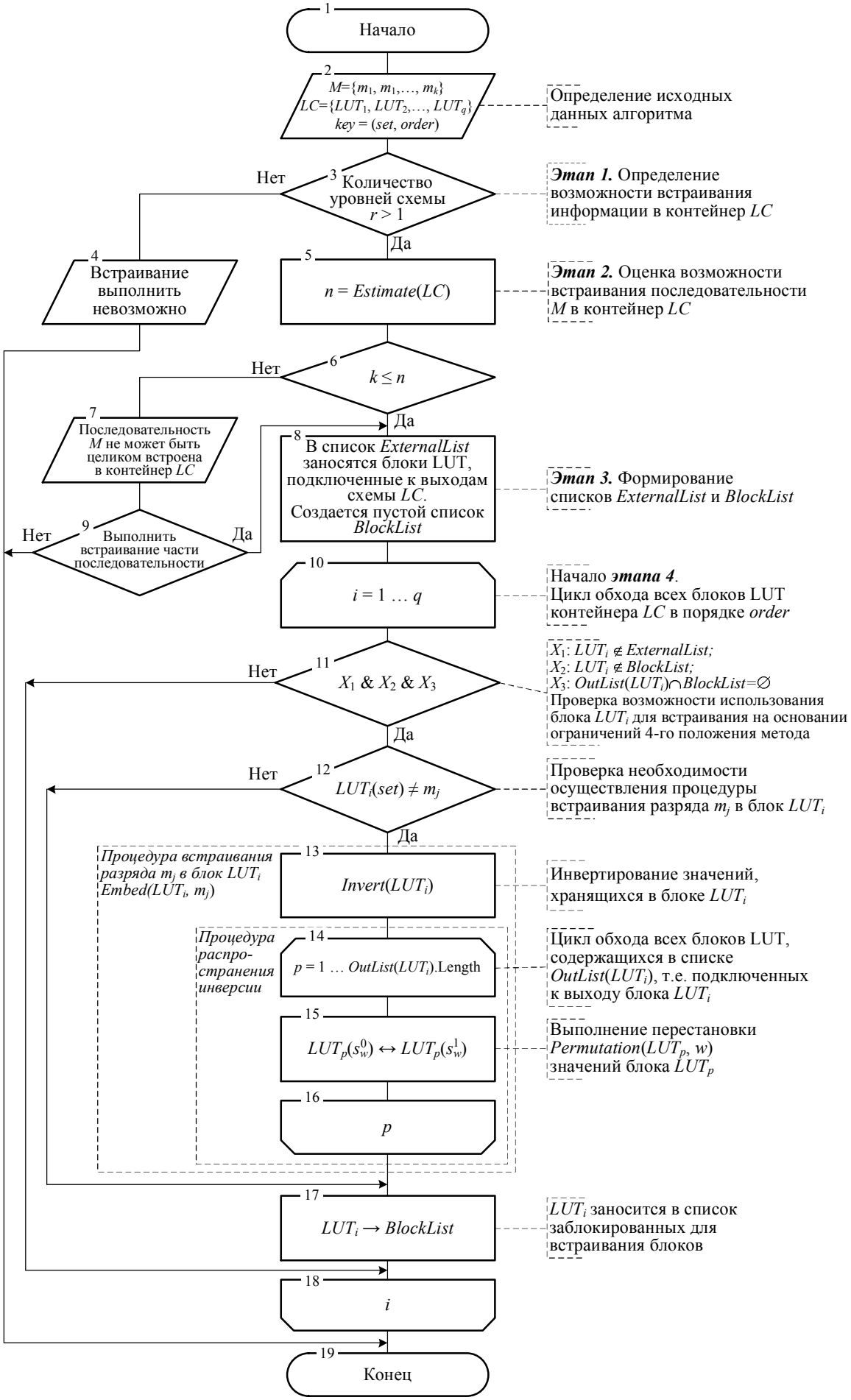

Рис. 5. Блок-схема алгоритма реализации предложенного метода встраивания

Алгоритм извлечения данных из контейнера в соответствии с предложенным методом

Далее рассматривается алгоритм извлечения из LUT-контейнера секретных данных, встроенных в соответствии с предлагаемым методом.

Исходные данные алгоритма:

1) LUT-контейнер LC^*, содержащий встроенную в него секретную двоичную последовательностью $M = (m_1, m_2, \ldots, m_k)$;

2) Ключ $key = (set, order)$ для извлечения встроенной информации.

Результат применения алгоритма – извлеченная из контейнера двоичная последовательность $M^* = (m_1^*, m_2^*, \ldots, m_k^*)$, которая при успешном извлечении должна совпадать с последовательностью $M = (m_1, m_2, \ldots, m_k)$.

Последовательность действий, необходимая для извлечения встроенных данных представлена на рис. 6 в виде блок-схемы. Несмотря на то, что информация о порядке обхода контейнера является составной частью стего-ключа, не все блоки LUT, лежащие на стего-пути, могли быть использованы для внедрения двоичной последовательности на этапе встраивания (ограничения 4-го положения метода). В силу чего (аналогично тому, как это было реализовано в алгоритме встраивания) при излечении необходимо использовать два служебных списка *ExternalList* и *BlockList*. Анализ этих списков позволяет определить мог ли блок LUT, лежащий на стего-пути, быть использован для внедрения бита секретной последовательности на этапе встраивания. Из блоков LUT, в отношении которых принято решение о том, что они могли быть использованы при встраивании, производится считывание значений по адресу *set*, являющемуся элементом стего-ключа. Считанные таким образом значения образуют извлеченную последовательность *M**.

Экспериментальное исследование предлагаемого метода

Для экспериментального исследования предлагаемого метода были разработаны аппаратно-программные средства, основанные на использовании микросхем FPGA Altera Cyclone II и САПР Altera Quartus II. На языке TCL была организована группа скриптов, выполняющих взаимодействие с САПР Altera Quartus II для считывания и записи содержимого блоков LUT. Непосредственно подсистема обработки считанных данных, в соответствии с предложенным методом, была реализована на языке C# в рамках платформы .Net.

Материалом для экспериментов выступили 25 FPGA-проектов разного объема и назначения. Эксперименты состояли во внедрении случайных секретных последовательностей в LUT-контейнеры FPGA-проектов; исследовании влияния такого внедрения на скоростные характеристики проекта, характеристики энергопотребления и тепловыделения; извлечении секретных последовательностей из заполненных контейнеров. Характеристики проекта измерялись средствами САПР Altera Quartus II Timing Analyzer и Power Play.

Экспериментальное исследование показало незначительное влияние применения метода на скоростные характеристики реализованных FPGA-проектов (изменение в среднем на 0.1%), характеристики энергопотребления и тепловыделения (изменение в среднем на 0.25%). Таким образом, установленные изменения указанных характеристик находится на уровне погрешности средств измерения и не могут считаться существенными.

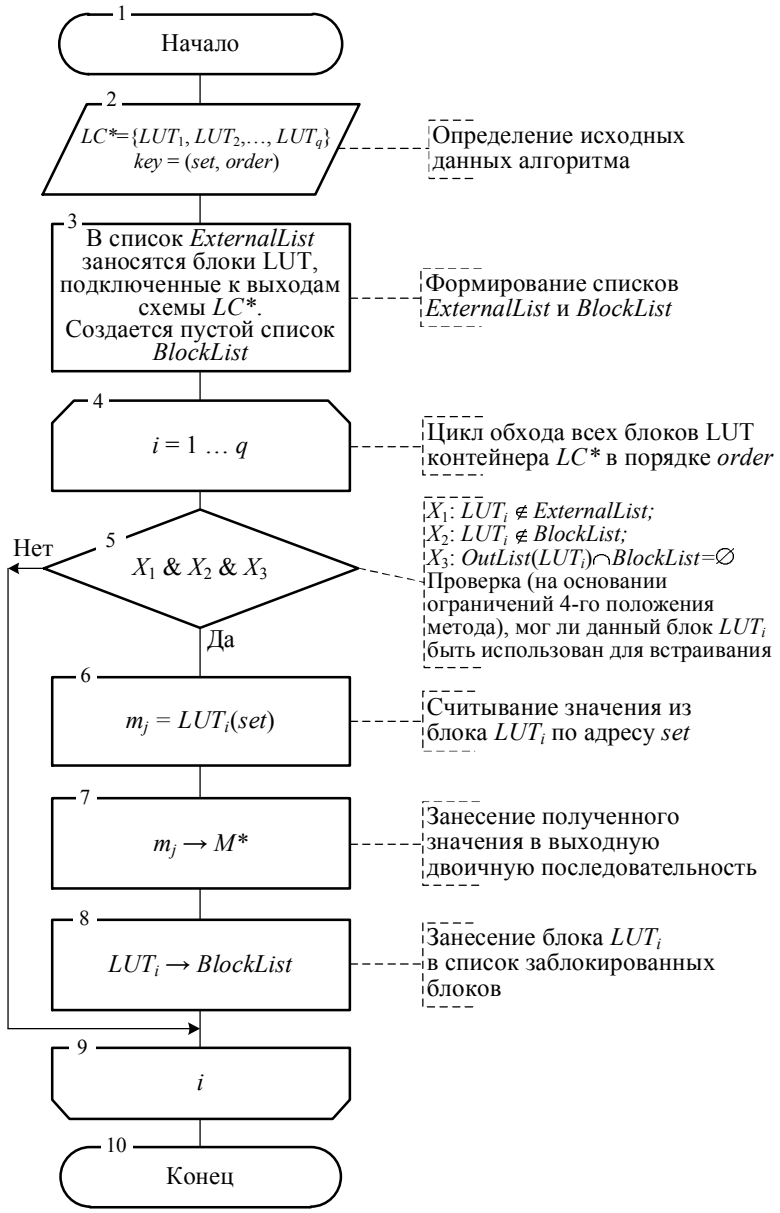

Рис. 6. Блок-схема алгоритма извлечения данных из LUT-контейнера

Преимущества и области использования предлагаемого метода

В отличие от традиционных мультимедийных стего-конвейеров, контейнеры, используемые в рамках предлагаемого метода являются активными, имеют влияющие друг на друга элементарные единицы, которые хранят в себе данные, представленные точно. Взаимное влияние элементарных единиц контейнера друг на друга дает возможность производить локальные изменения их содержимого, не меняя при этом глобальной функциональности контейнера. Точное представление данных порождает подходы к противодействию активным стего-атакам типа «стирание секретной информации», недоступные для традиционных мультимедийных контейнеров.

Природа данных, находящихся в элементарных единицах LUT-контейнеров не дает существенной статистической связи, как между разрядами отдельных единиц, так и между разрядами соседних единиц контейнера. Это не позволяет произвести

пассивную статистическую стего-атаку на такой контейнер методами стего-анализа, применяемыми для традиционных контейнеров.

Предложенный в данной работе метод может быть использован для внедрения ЦВЗ в компьютерные и управляющие устройства, построенные на основе элементной базы FPGA и ПЛИС со схожими архитектурами. Такое внедрение дает возможность контролировать правомерность использования проектной информации и самих устройств на различных этапах технологии проектирования и жизненного цикла (синтезированный FPGA проект, конфигурационный файл FPGA**,** действующее устройство).

Предложенный подход может найти применение при организации стего-защищенных систем передачи и хранения данных на основе LUT-контейнеров. Физически в качестве таких контейнеров могу выступать:

- файлы проектов в САПР FPGA устройств;
- конфигурационные файлы FPGA;
- действующие микросхемы FPGA в составе функционирующих устройств.

Кроме того, предложения данной работы могут быть использованы при организации стего-систем на основе LUT-контейнеров иной природы (не связанных с архитектурой микросхем FPGA). Однако выявление и исследование таких контейнеров требует дополнительных исследований.

Выводы

В работе предложен метод внедрения ЦВЗ в аппаратные контейнеры с LUT-ориентированной архитектурой. Метод позволяет внедрять двоичную информацию в LUT-контейнер, подвергая элементарные единицы контейнера локальным изменениям, не меняя при этом глобальную функциональность контейнера. Показаны алгоритмы реализации предложенного метода в части встраивания и извлечения секретной двоичной последовательности, которая представляет собой ЦВЗ. Метод предлагается использовать для внедрения секретных данных в LUT-контейнеры (на примере микросхем FPGA) с целью организации в их пространстве ЦВЗ, которые дают возможность контроля использования контейнера на всех этапах его жизненного цикла.

Список литературы

1. Конахович, Г.Ф. Компьютерная стеганография [Текст]: теория и практика / Г.Ф. Конахович, А.Ю. Пузыренко. — Киев : МК-Пресс, 2006. — 288 с.
2. Cox, I.J. Digital Watermarking and Steganography / I.J. Cox, *et al.* — 2nd edition. — Burlington: Morgan Kaufmann Publishers, 2008. — 624 p.
3. Fridrich, J. Steganography in Digital Media: Principles, Algorithms, and Applications / J. Fridrich. — New York: Cambridge University Press, 2010. — 462 p.
4. Shih, F.Y. Multimedia Security: Watermarking, Steganography, and Forensics / F.Y. Shih. — New York: CRC Press, 2012. — 424 p.
5. Skoudis, E. Malware: Fighting Malicious Code / E. Skoudis, L. Zeltser. — New Jersey: Prentice Hall, 2004. — 672 p.
6. El-Khalil, R. Hydan: Hiding Information in Program Binaries / R. El-Khalil, A.D. Keromytis // Proceedings of International Conference on Information and Communications Security (ICICS'2004), 27–29 October, Malaga, Spain. — PP. 187–199.
7. Hamilton, A. Survey of Static Software Watermarking / A. Hamilton, S. Danicic // Proceedings of World Congress on Internet Security (WorldCIS-2011), 21–23 February, London, UK. — PP. 100–107.
8. Hakun, L. New Approaches for Software Watermarking by Register Allocation / L. Hakun, K. Kaneko // Proceedings of the Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel Distributed Computing, 6–8 August 2008, Phuket, Thailand. — PP. 63–68.

9. XiaoCheng, L. Software Watermarking Algorithm Based on Register Allocation / L. XiaoCheng, C. Zhiming // Proceedings of the 2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES), 10–12 August, Hong Kong. — PP. 539–543.
10. Максфилд, К. Проектирование на ПЛИС: Архитектура, средства и методы. Курс молодого бойца [Текст] / К. Максфилд ; Пер. с англ. — М. : «Додэка-XXI», 2007. — 408 с.
11. Paul, S. Reconfigurable Computing Using Content Addressable Memory for Improved Performance and Resource Usage / S. Paul, S. Bhunia // Proceedings of the 45th annual Design Automation Conference ACM/IEEE (DAC-2008), 8–13 June 2008, Anaheim, USA. — PP. 786–791.
12. Грушвицкий, Р.И. Проектирование систем на микросхемах с программируемой структурой / Р.И. Грушвицкий, А.Х. Мурсаев, Е.П. Угрюмов. — 2-е изд., перераб. и доп. — СПб. : БХВ-Петербург, 2006. — 736 с.

МЕТОД ВБУДОВУВАННЯ ЦИФРОВИХ ВОДЯНИХ ЗНАКІВ В АПАРАТНІ КОНТЕЙНЕРИ З LUT-ОРІЄНТОВАНОЮ АРХІТЕКТУРОЮ

К.В. Защолкін, О.М. Іванова

Одеський національний політехнічний університет,
просп. Шевченко, 1, Одеса, 65044, Україна; e-mail: const-z@te.net.ua

Розглянуто задачу вбудовування цифрових водяних знаків в інформаційний об'єкт з метою контролю його використання. Відзначені типові підходи до організації такого вбудовування. Запропоновано метод вбудовування цифрових водяних знаків в апаратні контейнери з LUT-орієнтованою архітектурою. Показані алгоритми реалізації запропонованого методу. Описана апаратно-програмна реалізація методу і результати експериментів в середовищі цієї реалізації. Показано можливості застосування запропонованого методу для організації цифрових водяних знаків у просторі LUT-орієнтованого контейнера з метою контролю його використання в динаміці проектування та життєвого циклу.
Ключові слова: цифрові водяні знаки, стеганографія, захист інформації, апаратний стего-контейнер, LUT-орієнтована архітектура, FPGA, контроль використання FPGA-проектів

METHOD OF EMBEDDING OF DIGITAL WATERMARKS IN HARDWARE CONTAINERS WITH LUT-ORIENTED ARCHITECTURE

Kostyantyn V. Zashcholkin, Olena M. Ivanova

Odessa National Polytechnic University,
1 Shevchenko Ave., Odessa, 65044, Ukraine; e-mail: const-z@te.net.ua

A problem of embedding digital watermarks in information object to control the use of the latter was considered. Typical approaches to organizing such an embedding were noted. A technique to embed digital watermarks in hardware cover objects of LUT architecture was proposed. The algorithms to implement the technique proposed were demonstrated. Hardware-and-software implementation of the technique and the results of experiments within this implementation were described. The possibility of implementing the technique proposed for organization of digital watermarks in the domain of a cover object of LUT architecture to control its use within the project and life cycles was shown.
Keywords: digital watermarks, steganography, information security, hardware stego container, LUT-oriented architecture, FPGA, control of use FPGA-projects