УДК 004.056.53 DOI: 10.30929/1995-0519.2021.6.60-65

ПІДХІД ДО НЕЕКВІВАЛЕНТНОГО СТЕГАНОГРАФІЧНОГО ВБУДОВУВАННЯ ДОДАТКОВИХ ДАНИХ В ПРОГРАМНИЙ КОД БЛОКІВ LUT FPGA

О. М. Іванова, О. В. Дрозд, К. В. Защолкін, М. О. Кузнєцов
Національний університет «Одеська політехніка»
ORCID: 0000-0002-4743-6931; 0000-0003-2191-6758; 0000-0003-0427-9005; 0000-0002-3043-5924

Розглянуто задачу стеганографічного прихованого вбудовування додаткових даних в програмний код мікросхем FPGA. Зазначено, що таке вбудовування може бути використано для прихованого зберігання контрольних даних, які забезпечують моніторинг програмного коду. Визначено, що для вбудовування додаткових даних до програмного коду FPGA зазвичай використовують еквівалентні перетворення. Ці перетворення не змінюють цільову функцію програмного коду, і при цьому забезпечують вбудовування прихованих даних, які утворюють з програмним кодом єдине ціле. Однак корисний обсяг стего-контейнера в програмному коді FPGA, який можуть забезпечити еквівалентні перетворення, не завжди є достатнім для збереження контрольних даних одночасно для декількох різних видів оперативного моніторингу програмного коду. Запропоновано спільно з еквівалентними перетвореннями програмного коду використовувати нееквівалентні перетворення для збільшення корисного обсягу стего-контейнера в програмному коді FPGA. Виділено клас нееквівалентних перетворень, які не впливають на цільову функцію програмному коді FPGA. Перетворення такого роду мають зазначені властивості у випадку застосування до блоків FPGA-проєктів, що виконують обчислення для наближених операндів з кінцевим поданням результатів в форматі операндів. Розроблено програмну реалізацію теоретичних пропозицій нееквівалентного вбудовування додаткових даних. В середовищі розроблених програмних засобів виконано аналіз додаткового корисного обсягу стего-контейнера, який забезпечує запропонований нееквівалентний підхід. Зазначено, що в сукупності з традиційним підходом нееквівалентні перетворення програмного коду FPGA дозволяють забезпечити обсяг стего-контейнера, достатній для прихованого збереження контрольних даних декількох видів оперативного моніторингу програмного коду.

Ключові слова: стеганографічне вбудовування, мікросхеми FPGA, моніторинг програмного коду, контрольні дані; нееквівалентні перетворення.

АКТУАЛЬНІСТЬ РОБОТИ. Цифрові програмовані компоненти складають основу сучасних комп'ютерних систем. Такі компоненти налаштовуються на конкретний алгоритм функціонування за допомогою програмного коду. Зміна програмного коду приводить до зміни функціонування програмувального компонента й відповідно функціонування системи, що включає цей компонент до свого складу.

Це є значною перевагою програмованих компонентів, порівняно з компонентами, які мають жорстку логіку функціонування. Перевага обумовлена тим, що на будь-якій стадії життєвого циклу програмованого компонента існує можливість модернізації програмного коду, усунення виявлених в ньому помилок або пристосування функціонування компонента до нових зовнішніх умов. Однак можливість зміни поведінки компонентів через зміни програмного коду несе в собі загрози зловмисного втручання в роботу комп'ютерної системи шляхом впливу на програмний код. Для мінімізації цих загроз розроблена велика кількість методів, засобів і організаційних заходів. Серед них найбільш вагому роль відіграє оперативний контроль програмного коду.

Останнє десятиліття все більшу частку ринку програмованих компонентів займають мікросхеми FPGA (Field Programmable Gate Arrays) [1]. Ці мікросхеми є конкурентами мікропроцесорів. Для завдань, що вимагають високої продуктивності обчислень одночасно з можливістю перепрограмування FPGA мають переваги над мікропроцесорами. Через цю специфіку FPGA досить часто використовуються як компоненти комп'ютерних систем критичного застосування [2], для промислових, військових і аерокосмічних цілей. Мікросхеми FPGA програмуються шляхом завантаження програмного коду в конфігураційну пам'ять цих мікросхем. Через критичність і важливості застосування мікросхем FPGA оперативний контроль (наприклад, контроль цілісності) їх програмного коду здійснюється при кожній зміні коду в конфігураційній пам'яті, а також може здійснюватися регулярно за певним розкладом.

Традиційно контроль цілісності програмного коду здійснюється шляхом дворазового обчислення й порівняння контрольних хеш-сум [3]. При підготовці програмного коду до контролю обчислюється еталонна хеш-сумма. В момент контролю еталонна хеш-сума порівнюється зі знов обчисленою хеш-суммой. При цьому спосіб і місце зберігання еталонної хеш-суми значно впливає на можливість фальсифікації контролю.

Ефективним підходом до зберігання контрольних даних для програмного коду FPGA є стеганографічний підхід [4, 5]. У рамках цього підходу контрольні дані (хеш-сума) вбудовуються в програмний код у вигляді цифрового водяного знака. Таке вбудовування зазвичай виконується шляхом еквівалентних перетворень програмного коду [6]. Тому розмір програмного коду, поведінка й характеристики FPGA-базованої системи в результаті вбудовування не зазнають змін. Цифровий водяний знак (який містить еталонну хеш-суму) при цьому є прихованим від зовнішнього спостерігача та не відрізняється від цільового програмного коду. В момент виконання контролю прихована еталонна хеш-сума може бути витягнута авторизованим суб'єктом, який володіє стего-ключем. Також елементи стеганографічного підходу можуть бути застосовані для обфускації [7] програмного коду FPGA. Причому приховування контрольної хеш-суми й обфускація можуть бути виконані за допомогою єдиної системи еквівалентних перетворень. Призначення обфускації в цьому випадку полягає в тому, щоб утруднити стегоаналіз, спрямований на виявлення цифрового водяного знака в програмному коді. В силу цього можна конста-

тувати, що розвиток підходів до стеганографічного приховування контрольних даних для процесу моніторингу програмного коду FPGA є важливим і актуальним завданням.

Стеганографічний підхід до захисту даних базується на непомітному (такому, що не привертає увагу) вбудовуванні даних в інформаційний об'єкт – стего-контейнер. Основними умовами такого вбудовування є:

– незмінність головної функціональності стего-контейнера в результаті вбудовування;

– нерозрізненість вбудованих даних від цільових даних стего-контейнера.

Найбільший розвиток одержав напрямок стеганографії в рамках якого в якості стего-контейнерів використовуються мультимедійні файли [8]: растрові зображення, цифрове відео та звук. Основа методів стеганографії в цьому напрямку полягає в тому, що мультимедійні дані мають аналогову та наближену природу. Зображення, відео й звук є аналоговими явищами, перетвореними у двійкове подання в результаті застосування відповідних аналогових датчиків з наступним оцифровуванням їх показників. В силу цього елементарні одиниці мультимедійних стего-контейнерів (пікселі – для растрових зображень і відео, семпли – для цифрового звуку) мають області різної важливості. Наявність цих областей обумовлене нерівнозначністю розрядів цифрового подання аналогових даних. Молодші розряди значень пікселів і семплів створюють дуже малий внесок у кількісний еквівалент цих значень. Через це викривлення зазначених розрядів не впливає на основну функцію мультимедійного стего-контейнера. Такі мало значущі області в даних мультимедійних стего-контейнерів проявляються як у просторовій області їх подання, і в області перетворення. Методи цифровий стеганографії використовують ці мало значущі області як місця вбудовування приховуваних даних.

На відміну від мультимедійних стего-контейнерів програмний код FPGA є даними не наближеними, а точними. Зміну розрядів програмного коду FPGA тими методами, які використовуються в мультимедійній стеганографії виконати неможливо. Це пов'язано з тим, що така зміна приведе до викривлення функціонування пристрою, побудованого на основі FPGA. В силу цього для програмного коду FPGA зазвичай застосовують стеганографічні методи, базовані на еквівалентних перетвореннях. Ці методи в ході вбудовування даних еквівалентно модифікують значення програмного коду, не впливаючи, при цьому, на функціонування пристрою і його характеристики.

Таким чином, можна констатувати, що склалося наступне закріплення стеганографічних методів за типами стего-контейнерів: для мультимедійних стего-контейнерів використовуються методи нееквівалентної зміни даних елементарних одиниць контейнерів; для контейнерів з точно поданими даними (наприклад, для програмного коду FPGA) використовуються методи еквівалентного перетворення даних контейнера.

Однак структура програмного коду мікросхем FPGA має потенціал також і для виконання нееквівалентного (подібного тому, що застосовується для мультимедійних стего-контейнерів) вбудовування. Таке вбудовування можливе для програмного коду FPGA, що реалізує арифметичні операції: над наближеними даними, в ході яких формують результати з розрядністю, яка збігається з розрядністю операндів [9].

Таке нееквівалентне стеганографічне вбудовування даних може комбінуватися з еквівалентним вбудовуванням. В цьому випадку утворюється гібридна схема стеганографічного вбудовування даних у програмний код FPGA. У цій схемі обидва підходи до вбудовування (еквівалентний і нееквівалентний) можуть відігравати як однакові, так і різні ролі (приховання контрольних даних, обфускація програмного коду, фальш вбудовування). Наприклад, один з підходів виконує вбудовування з метою приховання контрольних даних, а інший використовується для обфускації програмного коду або для фальш вбудовування.

Однією з умов застосовності пропонованого нееквівалентного вбудовування є забезпечення достатнього для зазначених ролей обсягу стего-контейнера. Виходячи із цього *мета даної роботи* полягає у збільшенні корисного обсягу стего-контейнера, утвореного в середовищі програмного коду FPGA за рахунок введення додаткової процедури нееквівалентного перетворення програмного коду для блоків схеми, які виконують арифметичні операції над наближеними даними.

МАТЕРІАЛ І РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ. При виконанні арифметичних операцій з плаваючою крапкою типовою є вимога одержання результатів, розрядність яких збігається з розрядністю операндів. При виконанні операції над n-розрядними операндами повинен бути отриманий n-розрядний результат. Для ряду арифметичних операцій розрядність результату природно перевищує розрядність операндів. У цьому випадку спочатку виконується обчислення повного результату. Після цього виконується округлення й відкидання молодших розрядів. У підсумку розрядність результату стає такою, що дорівнює розрядності операндів [9]. Наприклад, при виконанні операції множення n-розрядних мантис чисел із плаваючою крапкою отримується $2n$-розрядний результат. Далі цей результат округляється з відкиданням молодших n-розрядів. Розряди, що залишилися утворюють необхідний n-розрядний результат.

У зазначених випадках розряди результату можна розділити на дві підмножини:

1) розряди, що відкидаються (несуттєві) розряди;

2) розряди, що залишаються (суттєві) розряди.

Викривлення значень несуттєвих молодших розрядах результату не впливають на підсумковий результат операції. При реалізації таких арифметичних операцій на FPGA у її структурі можна виділити елементарні обчислювальні блоки, які беруть участь у формуванні тільки несуттєвих розрядів результату. У силу цього програмний код таких блоків може бути нееквівалентно модифікований і така модифікація не вплине на результат роботи пристрою.

Далі пропонований підхід і його оцінка роблять-ся на прикладі операції множення, яка є базовою операцією для двійкової арифметики з плаваючою крапкою. Однак підхід може бути поширено й на інші арифметичні операції, у яких кінцевий результат утворюється з повного результату шляхом відкидання несуттєвих молодших розрядів.

Нехай операнди для операції множення мають наступний вигляд:

$$A = < a_n, a_{n-1}, \ldots, a_1 >; \; B = < b_n, b_{n-1}, \ldots, b_1 >,$$

а результат виконання зазначеної операції:

$$C = < c_{2n}, c_{2n-1}, \ldots, c_{n+1}, c_n, c_{n-1}, \ldots, c_1 >.$$

Для приведення результату C до необхідного формату (при наявності вимоги однакової розрядності операндів і результату) старші n-розрядів результату зберігаються, а молодші відкидаються. Таким чином, розряди повного результату утворюють дві впорядковані множини:

$$C_{EB} = < c_{2n}, c_{2n-1}, \ldots, c_{n+1} > \text{ та } C_{IEB} = < c_n, c_{n-1}, \ldots, c_1 >$$

де C_{EB} – множина суттєвих розрядів; C_{IEB} – множина несуттєвих (тих, що відкидаються при округленні) розрядів.

Основними елементарними одиницями структури мікросхем FPGA є блоки LUT (Look Up Table) [11, 12]. Ці блоки являють собою програмовані обчислювачі, призначені для обчислення однієї логічної функції від m змінних, де m – кількість входів блока LUT. Сучасні сімейства мікросхем FPGA містять блоки LUT з кількістю входів від 4 до 8. Блок LUT програмується на обчислення конкретної логічної функції 2^m-розрядним програмним кодом.

Для мікросхеми FPGA, яка реалізує арифметичну операцію, виділимо блоки LUT, що беруть участь в обчисленні несуттєвих розрядів результату операції і при цьому не беруть участь в обчисленні суттєвих розрядів: $L_{IE} \subset L$, де L – множина усіх блоків LUT в FPGA-проєкті, що реалізує відповідні арифметичні операції; L_{IE} – підмножина блоків LUT, які беруть участь в обчисленні тільки несуттєвих розрядів і не беруть участі в обчисленні суттєвих розрядів.

Далі блоки LUT, що входять до множини L_{IE} буде називати несуттєвими блоками LUT. Програмний код несуттєвих блоків LUT може бути нееквівалентно модифікований з метою стеганографічного вбудовування в нього додаткових даних. Така модифікація спотворить результат обчислень, але тільки в несуттєвих розрядах. Оскільки ці розряди відкидаються при округленні, то модифікація програмного коду блоків L_{IE} не буде відбиватися на остаточному (приведеному до кількості розрядів операндів) результаті обчислень.

Необхідно оцінити те, наскільки ефективним є пропонований нееквівалентний підхід до вбудовування додаткових даних в частині забезпечення обсягу стеганографічного контейнера. Для цього виконано експериментальну оцінку, первісними даними якої в FPGA-проєкти, що містять двійкові помножувачі різної розрядності.

Мета експерименту полягає в оцінці частки несуттєвих (тих, що входять до множини L_{IE}) блоків LUT у загальній кількості блоків LUT арифметичного пристрою, реалізованого на FPGA. У якості цільової функції пристрою обрано операцію множення.

Первісні дані для експерименту формувалися за допомогою САПР Intel Quartus Prime 20.1 [13] і розробленого програмного забезпечення. Процес отримання первісних даних полягав у наступному. В САПР Intel Quartus з використанням бібліотечних компонентів було сформовано множину FPGA-проєктів. Кожний із цих проєктів реалізує помножувач відповідної розрядності. Для експерименту використовувалися помножувачі з розрядностями операндів 4, 6, 8, 12, 16 біт. Помножувачі були сконфігуровані таким чином, щоб за n-розрядними операндами формувався $2n$-розрядний результат. Таким чином, «сирий» результат обчислень, який має місце до відкидання розрядів, був доступний для аналізу.

Далі в середовищі САПР Intel Quartus Prime виконувався синтез, а також розміщення й трасування проєкту. Цільовою мікросхемою синтезу була обрана FPGA Intel Cyclone IV EP4CE15F23A7 [14]. Після закінчення розміщення й трасування було задіяно програмний додаток M1 (рис. 1), розроблений з використанням мови TCL.

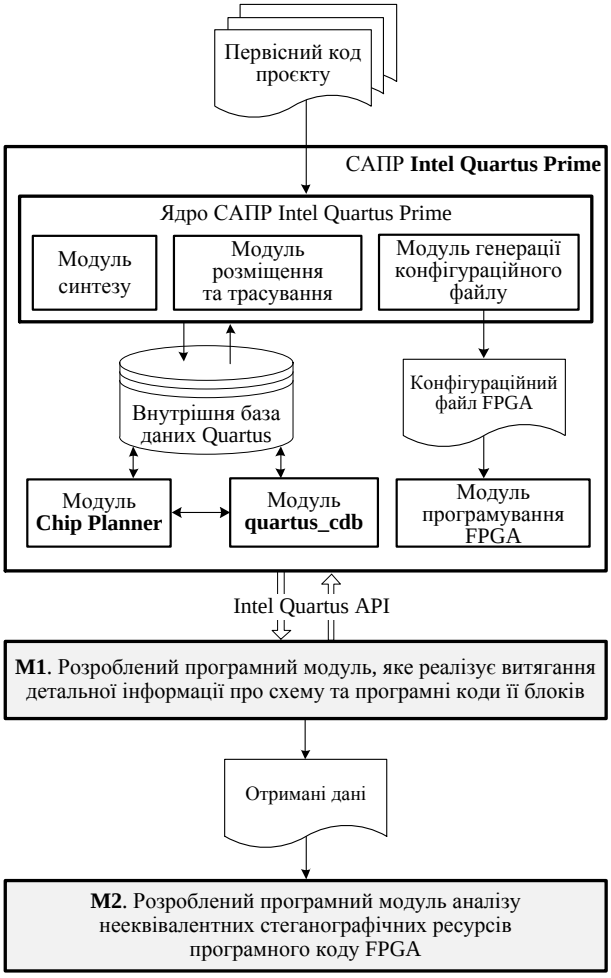

Рисунок 1 – Взаємодія розроблених програмних модулів з САПР Intel Quartus Prime

Додаток M1 дозволяє взаємодіяти з САПР Intel Quartus Prime використовуючи API (Application Programming Interface) цієї САПР. Основна функціональність зазначеного додатку полягає у витяганні

із внутрішньої бази даних САПР Intel Quartus детальної інформації про структуру FPGA-проєкту [15].

Витягнута інформація містить в собі:

а) опис блоків LUT, що входять до схеми FPGA-проєкту;

б) програмні коди кожного із блоків LUT;

в) інформацію про зв'язки блоків LUT між собою, а також із входами й виходами мікросхеми. Результатом функціонування розробленого TCL-додатка є структура FPGA-проєкту, представлена у формі, придатної для подальшого аналізу.

Основна частина експерименту була реалізована в середовищі другого розробленого додатка M2 (рис. 1). Цей додаток отримує вихідні дані (інформацію про структуру схеми FPGA-проєкту) і визначає, які із блоків LUT проєкту є несуттєвими. Діаграму послідовності взаємодії розроблених програмних модулів M1та M2 з компонентами САПР Intel Quartus Prime в ході виконання експерименту наведено на рис. 2.

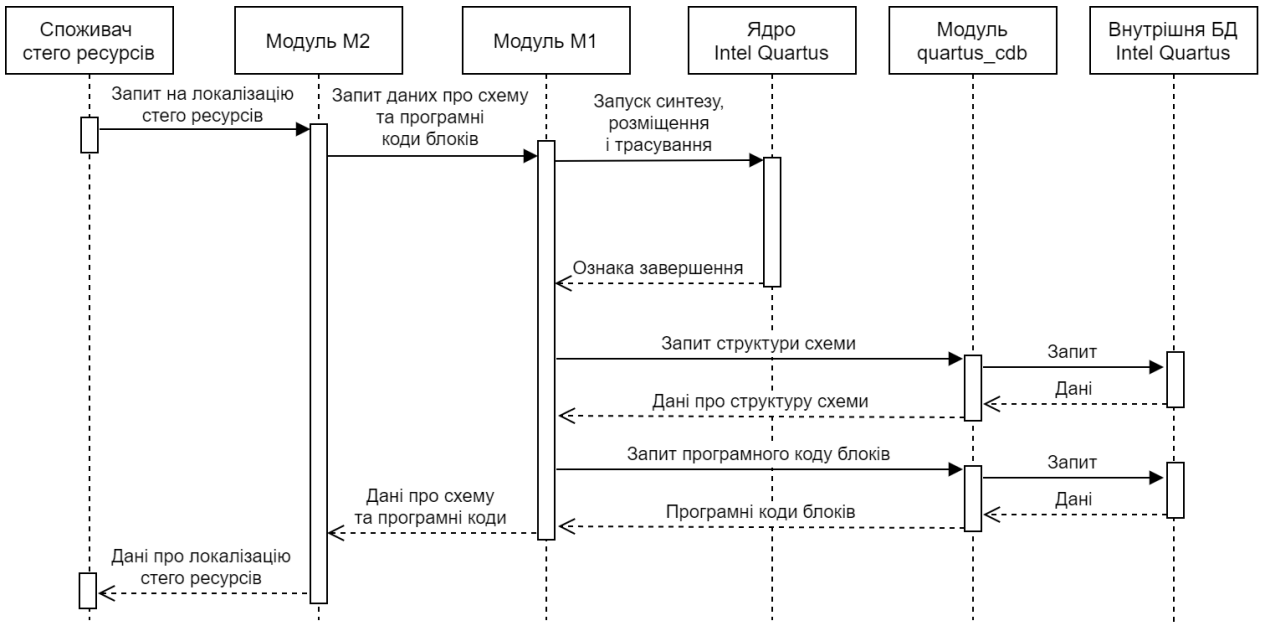

Рисунок 2 – Діаграма послідовності взаємодії розроблених програмних модулів та САПР Intel Quartus Prime

Процес аналізу блоків LUT програмним модулем M2 полягає в наступному. Для кожного з можливих спільних значень операндів *A* і *B* помножувача виконуються наступні кроки.

1. Обчислюється 2*n*-розрядний результат помноження операндів *A* та *B*.

2. Фіксуються правильні значення на виходах кожного із блоків LUT FPGA-проєкту помножувача.

3. Значення на виході кожного із блоків LUT FPGA-проєкту послідовно інвертуються (змінюються з вірного значення на невірне). При кожному з таких викривлень фіксується величина помилки результату. Величина помилка обчислюється як модуль різниці правильного значення результату й значення результату, отриманого при інвертуванні. При цьому кожному із блоків LUT проєкту ставиться у відповідність максимальна величина помилки, що виникла в результаті викривлення значення виходу цього блока. Збережена величина помилки обновляється при отриманні поточної помилки, більшої, ніж збережена.

Величини максимальної помилки, отримані в результаті перебору значень операндів, дозволяють прийняти рішення щодо того, чи відноситься блок LUT до підмножини несуттєвих блоків. Якщо максимальна помилка, яка поставлена у відповідність блоку LUT_i *n*-розрядного помножувача, є меншою за 2^n, то ця помилка проявляється тільки в розрядах результату, що відкидаються $C_{IEB} = < c_n, c_{n-1}, \ldots, c_1 >$. Це означає, що при будь-яких значеннях операндів викривлення значень на виході блока LUT_i не приводять до викривлення розрядів, що залишаються (суттєвих розрядів, які не відкидаються) результату. Таким чином, продукційне правило для віднесення блока LUT *n*-розрядного помножувача до підмножини несуттєвих блоків має наступний вигляд:

$$\text{if } max\_error(LUT_i) < 2^n \text{ then } LUT_i \in L_{IE},$$

де $max\_error(LUT_i)$ – значення максимальної помилки, поставлене у відповідність блокові LUT_i.

Результати проведеного експерименту дозволили отримати значення кількості несуттєвих блоків LUT у схемах помножувачів, реалізованих на FPGA. Оскільки значення на виходах несуттєвих блоків не впливають на суттєві (ті, що залишаються після округлення) розряди результату, програмний код цих блоків може бути нееквівалентно модифікований. Таким чином, кількість несуттєвих блоків дозволяє оцінити розмір стего-контейнера, утвореного в середовищі LUT-схеми помножувача.

В ході експериментів було виконано аналіз нееквівалентних стего ресурсів для помножувачів з розрядністю операндів 4, 6, 8, 12 і 16 біт. Частка несут-

тєвих блоків LUT в експериментальних FPGA-проєктах склала від 39,3% до 42,5%.

Функціонування блока LUT з *m* входами задається 2^m бітовим програмним кодом (зазвичай *m* становить від 4 до 8). Таким чином, у кожний несуттєвий блок LUT може бути вбудовано мінімум 1 і максимум 2^m розрядів додаткових даних. Верхня й нижня оцінка обсягу стего-контейнера, отриманого за рахунок застосування пропонованого нееквівалентного вбудовування даних склала відповідно 12 та 192 розрядів для одного помножувача мантис чисел з плаваючою крапкою, який виконує операції над 4-розрядними мантисами та відповідно 145 та 2320 розрядів для помножувача 16-розрядних мантис.

Отримані оцінки дозволяють констатувати достатність додаткових обсягів стего-контейнера, (отриманих за рахунок застосування нееквівалентного підходу) для вбудовування контрольних даних окремого виду моніторингу програмного коду. Що в сукупності з обсягами, які забезпечуються традиційним еквівалентним підходом дає можливість збільшити кількість одночасно застосованих видів оперативного моніторингу.

ВИСНОВКИ. Встановлено, що для стеганографічного вбудовування додаткових даних в програмний код FPGA може бути разом з традиційним підходом застосований підхід, який базується на нееквівалентних перетвореннях програмного коду. В роботі показано, що частина блоків LUT в FPGA-проєкті не впливає на результат при виконанні обчислень з наближеними даними. Через це програмний код таких блоків LUT може бути нееквівалентно змінений в процесі стеганографічного вбудовування додаткових даних. При цьому функціонування мікросхеми FPGA не зазнає змін.

Пропонується підхід для локалізації таких блоків LUT. Визначено продукційне правило, за допомогою якого блоки LUT можуть бути віднесені до множини несуттєвих блоків. В середовищі програмного забезпечення, яке реалізує зазначений підхід, проведено експериментальне дослідження. Результати дослідження показали додаткове збільшення ефективного обсягу стего-контейнера, яке можливо використати для прихованого зберігання контрольних даних декількох видів оперативного моніторингу програмного коду FPGA.

ЛІТЕРАТУРА

1. Amano H. Principles and Structures of FPGAs. Heidelberg : Springer, 2018. 240 p.

2. Kharchenko V., Kovalenko A., Siora O., Sklyar V., Security assessment of FPGA-based safety-critical systems: US NRC requirements context. *Proceedings of the Information Digital Technologies International Conference*. Slovakia, Zilina, 2015. P. 132–138.

3. Awad A., Fairhurst M. Information Security. Foundations, technologies and applications. UK, London : The Institution of Engineering and Technology, 2018. 426 p.

4. Shih F. Digital Watermarking and Steganography: Fundamentals and Techniques. 2nd ed. USA, Boca Raton : CRC Press, 2017. 320 p.

5. Ke Y., Liu J., Zhang M., Su T., Yang X. Steganography Security : Principle and Practice. *IEEE Access*. 2018. Vol. 6. P. 73009–73022. DOI: https://doi.org/10.1109/ ACCESS.2018.2881680.

6. Zashcholkin K., Drozd O., Shaporin R., Ivanova O., Drozd M. Co-Embedding Additional Security Data and Obfuscating Low-Level FPGA Program Code. *Proceedings of IEEE East-West Design and Test Symposium*. 2020. DOI: https://doi.org/10.1109/EWDTS50664.2020.92251112020.

7. Bishop M. Computer Security. 2nd edn. USA, Boston : Addison-Wesley, 2018.

8. Yahya A. Steganography Techniques for Digital Images. Heidelberg : Springer, 2018.

9. Anderson A., Muralidharan S., Gregg D. Efficient Multibyte Floating Point Data Formats Using Vectorization. *IEEE Transactions on Computers*. 2017. Vol. 66. No. 12. P. 2081–2096. DOI: https://doi.org/10.1109/TC.2017.2716355.

10. IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2019 (Revision of IEEE 754-2008). 2019. P. 1–84. DOI: https://doi.org/10.1109/IEEESTD.2019.8766229.

11. Vanderbauwhede W., Benkrid K. High-performance computing using FPGAs. USA, New-York : Springer, 2016.

12. Raj A., Bazil Arockia. FPGA-Based Embedded System Developer's Guide. USA, Boca Raton : CRC Press, 2018.

13. Intel Quartus Prime Standard Edition User Guide. URL: https://www.intel.com/content/dam/alterawww/global/en_US/pdfs/literature/ug/ug-qps-getting-started.pdf. (дата звернення: 22.12.2021).

14. Cyclone IV Device Handbook. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyclone4-handbook.pdf. (дата звернення: 22.12.2021).

15. Защелкин К.В. Методика и программная реализация построения информационной модели LUT-схемы, размещенной в среде FPGA. *Вісник Кременчуцького національного університету імені М. Остроградського*. 2018. №1 (108). С. 59–64.

AN APPROACH TO NON-EQUIVALENT STEGANOGRAPHIC EMBEDDING OF ADDITIONAL DATA INTO THE PROGRAM CODE OF FPGA LUT UNITS

O. Ivanova, O. Drozd, K. Zashcholkin, M. Kuznietsov

Odessa Polytechnic National University

ORCID: 0000-0002-4743-6931; 0000-0003-2191-6758; 0000-0003-0427-9005; 0000-0002-3043-5924

Purpose. To increase the effective volume of stego containers in the FPGA program code. To achieve this purpose, an additional procedure of non-equivalent transformations of FPGA program code is added. Non-equivalent transformations complement the traditional procedure of equivalent transformations of program code. The joint use of these two

types of transformations leads to the achievement of the purpose of this paper. **Methodology**. The steganographic approach is used to covertly embed additional data into the FPGA chip program code. In this case the FPGA program code is a stego container. The stego container embedding procedure is used to perform hidden monitoring of the FPGA program code. For this purpose, monitoring data is embedded into the program code and secretly stored there. The traditional approach to steganographic embedding in FPGA program code is based on the use of equivalent transformations of the program code. Such transformations do not change the target function of the program code or the operation of the FPGA chip. However, the traditional approach provides a relatively small effective volume of the stego container. This leads to the fact that it is usually possible to steganographically store only control data for one type of monitoring. It is proposed to additionally use a non-equivalent approach to steganographic embedding of data into the FPGA program code. Such transformations, despite their non-equivalence, do not change the target function of the program code. This is achieved by applying transformations to the program code of units that perform arithmetic operations on approximate data. For arithmetic operations on approximate data there is often a requirement that the operands and the result are the same size. To satisfy this requirement the complete result of the operation is computed first. After that some bits of the result are discarded and rounding is performed. It is proposed to allocate elementary LUT units in the FPGA structure, which participate only in the calculation of discarded bits (and do not participate in the calculation of the remaining bits). Program codes of such LUTs can be non-equivalently changed during steganographic embedding. Distortion of the FPGA program code of such units does not distort the behavior of the FPGA chip. This is a consequence of the fact that such LUTs are not involved in the formation of the discarded bits. **Results.** We have developed software that, together with Intel Quartus CAD system, extracts detailed information about an FPGA project. This information includes the structure of the circuit in the project and the program codes of the units of this circuit. An application has also been developed that uses this information to determine non-equivalent steganographic resources. With the help of the developed software, an experimental estimation of the additional volume of the stego container is performed. **Originality.** An approach to the use of non-equivalent transformations of the program code of FPGA chips for steganographic embedding of additional data is proposed. This approach is proposed to be used together with the traditional approach, which is based on equivalent transformations of the program code. **Practical value.** The steganographic embedding approach proposed in this paper allows increasing the effective volume of the stego container in the FPGA program code. By applying this approach, it becomes possible to secretly store monitoring data of several types of monitoring in FPGA stego containers. References 10, figures 2.

Key words: steganographic embedding, FPGA chips, monitoring of program code, control data, non-equivalent transformations.

REFERENCES

1. Amano, H. (2018). Principles and Structures of FPGAs. Heidelberg, 240 p.

2. Kharchenko, V., Kovalenko, A., Siora, O., Sklyar, V. (2015). Security assessment of FPGA-based safety-critical systems: US NRC requirements context. *Proceedings of the Information Digital Technologies International Conference*. Slovakia, Zilina, pp. 132–138.

3. Awad, A., Fairhurst, M. (2018). Information Security. Foundations, technologies and applications. UK, London, 426 p.

4. Shih F. (2017). Digital Watermarking and Steganography: Fundamentals and Techniques. 2nd ed., USA, Boca Raton, 320 p.

5. Ke, Y., Liu, J., Zhang, M., Su, T., Yang, X. (2018). Steganography Security: Principle and Practice. *IEEE Access*. Vol. 6, pp. 73009–73022. DOI: https://doi.org/10.1109/ACCESS.2018.2881680.

6. Zashcholkin, K., Drozd, O., Shaporin, R., Ivanova, O., Drozd, M. (2020). Co-Embedding Additional Security Data and Obfuscating Low-Level FPGA Program Code. *Proceedings of IEEE East-West Design and Test Symposium*. DOI: https://doi.org/10.1109/EWDTS50664.2020.92251112020.

7. Bishop, M. (2018). Computer Security. 2nd edn. USA, Boston.

8. Yahya, A. (2018). Steganography Techniques for Digital Images. Heidelberg.

9. Anderson, A., Muralidharan, S., Gregg, D. (2017). Efficient Multibyte Floating Point Data Formats Using Vectorization. *IEEE Transactions on Computers*. Vol. 66. No. 12, pp. 2081–2096. DOI: https://doi.org/10.1109/TC.2017. 2716355.

10. IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2019. (2019). (Revision of IEEE 754-2008), pp. 1–84. DOI: https://doi.org/10.1109/IEEESTD.2019.8766229.

11. Vanderbauwhede, W., Benkrid, K. (2016) High-performance computing using FPGAs. USA, New-York.

12. Raj, A., Bazil, A. (2018). FPGA-Based Embedded System Developer's Guide. USA, Boca Raton.

13. Intel Quartus Prime Standard Edition User Guide. URL: https://www.intel.com/content/dam/alterawww/global/en_US/pdfs/literature/ug/ug-qps-getting-started.pdf.

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

15. Zashcholkin, K.V. (2018). Metodika i programmnaya realizaciya postroeniya informacionnoj modeli LUT-skhemy, razmeshchennoj v srede FPGA [The technique and software implementation of creation of the LUT-circuit information model placed in FPGA environment]. *Transaction of Kremenchuk Mykhailo Ostrohradskyi National University*. №1 (108), pp. 59–64 [in Russian]

Стаття надійшла 22.12.2021