# ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

УДК 004.056.53 DOI https://doi.org/10.35546/kntu2078-4481.2025.1.2.1

С. Г. АНТОЩУК
доктор технічних наук, професор, директор
Інститут комп'ютерних систем
Національного університету «Одеська політехніка»
ORCID: 0000-0002-9346-145X

О. М. ІВАНОВА
старший викладач кафедри комп'ютерних систем
Національний університет «Одеська політехніка»
ORCID: 0000-0002-4743-6931

## МЕТОД ФОРМУВАННЯ СТЕГО-КЛЮЧА ДЛЯ ЗБІЛЬШЕННЯ ОБСЯГУ ПРИХОВАНОГО ЗБЕРІГАННЯ ДАНИХ В СЕРЕДОВИЩІ ПРОГРАМНОГО КОДУ FPGA

*У статті розглядаються питання прихованого зберігання контрольних даних в середовищі низькорівневого програмного коду мікросхем FPGA при виконанні моніторингу цього програмного коду. Моніторинг характеристик безпеки програмного коду, таких як цілісність, автентичність, шляхи його розповсюдження є однією з основних складових забезпечення безпеки програмованих систем. В статті зазначено, що перспективними є методи моніторингу характеристик безпеки програмного коду FPGA, в рамках яких контрольні дані, що використовуються цими методами, вбудовуються в програмний код в стеганографічний спосіб у вигляді цифрового водяного знака. В результаті таке вбудовування не впливає на поведінку мікросхем FPGA і не змінює характеристики системи, побудованої на основі цих мікросхем. Перевагою зазначеного підходу є те, що факт наявності контрольних даних у програмному коді та факт виконання моніторингу є скритими. Однак при використанні для моніторингу контрольних даних, які вбудовуються в програмний код, існує проблема відновлення початкового стану цього програмного коду. Проблема полягає в необхідності стеганографічного збереження як самих контрольних даних, так і інформації для відновлення початкового стану програмного коду. Однак обсяг інформації, необхідної для відновлення, може займати дуже велику частину обсягу цифрового водяного знака. Це значно зменшує частину обсягу цифрового водяного знака, яка містить безпосередньо контрольні дані моніторингу. В результаті часто виникає ситуація при якій ефективний обсяг цифрового водяного знака є недостатнім для зберігання контрольних даних з необхідним для моніторингу розміром. В статті пропонуються шляхи вирішення цієї проблеми шляхом застосування інтервального підходу до формування стего-ключа вбудовування даних в програмний код. Описано експериментальне дослідження підходу, запропонованого в статті, та на його основі показані переваги цього підходу.*

***Ключові слова:*** *стеганографічне вбудовування даних, цифрові водяні знаки, стего-ключ, моніторинг програмного коду, мікросхеми FPGA, інформаційний об'єкт програмного коду.*

S. G. ANTOSCHUK
Doctor of Technical Sciences, Professor, Director
Institute of Computer Systems of Odesa Polytechnic National University
ORCID: 0000-0002-9346-145X

O. M. IVANOVA
Senior Lecturer at the Department of Computer Systems
Odesa Polytechnic National University
ORCID: 0000-0002-4743-6931

## THE INTERVAL STEGO-KEY METHOD FOR INCREASING THE VOLUME OF HIDDEN DATA STORAGE IN THE ENVIRONMENT OF FPGA CHIPS PROGRAM CODE

*The article considers the issues of hidden storage of monitoring data in the environment of low-level program code of FPGA chips when performing monitoring of this program code. Monitoring the security characteristics of program code, such as integrity, authenticity, and ways of its distribution, is one of the main components of ensuring the security of programmable systems. The article notes that the methods of monitoring the security characteristics of FPGA*

*program code are promising, in which the monitoring data used by these methods are embedded in the program code in a steganographic way in the form of a digital watermark. As a result, such embedding does not affect the behavior of FPGA chips and does not change the characteristics of the system built on the basis of these chips. The advantage of this approach is that the fact of the presence of monitoring data in the program code and the fact of monitoring are hidden. However, when monitoring data embedded in program code is used for monitoring, there is a problem of recovering the initial state of this program code. The problem is that it is necessary to steganographically save both the monitoring data and the information to recover the initial state of the program code. However, the volume of information required for recovery can take up a very large part of the volume of the digital watermark. This significantly reduces the part of the digital watermark that contains the monitoring data itself. As a result, a situation often arises when the effective volume of the digital watermark is insufficient to save monitoring data with the size required for monitoring. The paper proposes ways to solve this problem by applying an interval approach to the formation of a stego-key for embedding data in program code. The paper describes an experimental research of the approach proposed in the article and shows the advantages of this approach.*

***Key words:*** *steganographic data embedding, digital watermarks, stego-key, program code monitoring, FPGA chips, program code information object.*

**Постановка проблеми**

Мікросхеми FPGA (Field Programmable Gate Array) є програмованими мікросхемами з іншим, ніж у мікропроцесорів принципом програмування [1]. Вони складаються з великої кількості елементарних програмованих обчислювачів, комутація між якими здійснюється засобами ієрархічної програмованої комутаційної матриці. Низькорівневий програмний код, який завантажується до мікросхеми FPGA, налаштовує кожний з елементарних обчислювачів в структурі мікросхеми на реалізацію конкретної логічної функції. Крім того програмний код конфігурує комутаційну матрицю FPGA таким чином, щоб організувати потрібні для вирішення обчислювальної задачі зв'язки між елементарними блоками мікросхеми. Таким чином внутрішня конфігурація мікросхеми FPGA і функції її блоків змінюються під дією програмного коду. Такий вплив програмного коду на функціонування мікросхеми суттєво відрізняє принцип програмування FPGA від принципу програмування мікропроцесорів. Мікропроцесори в процесі реалізації обчислювальної задачі послідовно виконують програму, записану в пам'яті програм. Мікросхеми FPGA змінюють внутрішню конфігурацію під дією програмного коду після чого всі елементарні обчислювальні блоки, які складають мікросхему, функціонують паралельно. Такі особливості функціонування дають змогу отримати значно більшу продуктивність при вирішенні обчислювальних задач на FPGA порівняно з мікропроцесорами [2].

Процес функціонування програмованих цифрових пристроїв, таких, як мікропроцесори та FPGA, керується програмним кодом. Через це існує потенційна можливість зловмисного втручання в функціонування цих компонентів через використання маніпуляцій з програмним кодом. Одним з дієвих засобів протидії зловмисній маніпуляції програмним кодом є оперативний моніторинг характеристик безпеки програмного коду, таких як цілісність, автентичність та шляхи розповсюдження коду [3]. Тому підвищення ефективності такого моніторингу є актуальною та важливою задачею в умовах потенційної можливості зловмисного втручання в функціонування програмованих систем.

**Аналіз останніх досліджень і публікацій**

Найчастіше застосовуються методи моніторингу характеристик безпеки програмного коду, базовані на використанні контрольних даних [4]. До початку процесу моніторингу відповідно до певних алгоритмів моніторингу обчислюються та зберігаються еталонні контрольні дані. При виконанні кожного акту моніторингу еталонні контрольні дані зчитуються та порівнюються з даними, отриманими безпосередньо під час моніторингу програмного коду. Недолік традиційних методів моніторингу полягає в тому, що факт наявності еталонних контрольних даних є відкритим. Так контрольні дані можуть зберігатися в пам'яті системи, яка здійснює моніторинг [5]. Також контрольні дані можуть приєднуватися до інформаційного об'єкта програмного коду, щодо якого здійснюється контроль [6]. Крім того існує підхід в межах якого контрольні дані зберігаються у віддаленій базі даних та запитуються з неї під час здійснення моніторингу [7]. У випадку зберігання у відкритому вигляді еталонних контрольних даних в межах зазначених підходів існують можливості зловмисної маніпуляцій цими даними для обходу моніторингу.

Також відомим є підхід в межах якого еталонні контрольні дані зберігаються не у відкритому, а в зашифрованому вигляді [8]. Такий підхід приховує контрольні дані, але залишає відкритим факт виконання моніторингу та факт наявності контрольних даних. Це відкриває можливість застосування достатньо великої кількості прийомів для відкриття та фальсифікації еталонних контрольних даних.

Є відомим підхід, який передбачає зберігання еталонних контрольних даних в стеганографічний спосіб [9, 10]. Цей підхід позбавлений недоліків зазначених вище підходів. Вій полягає в тому, що еталонні контрольні дані вбудовуються в інформаційний об'єкт програмного коду у вигляді цифрового водяного знака [11]. В результаті таке вбудовування не впливає на поведінку мікросхем FPGA і не змінює характеристики системи, побудованої на

основі цих мікросхем. Перевагою зазначеного підходу є те, що факт наявності контрольних даних у програмному коді та факт виконання моніторингу є скритими. Однак при використанні для моніторингу контрольних даних, які вбудовуються в програмний код, існує проблема відновлення початкового стану цього програмного коду [12]. Проблема полягає в необхідності стеганографічного збереження як самих контрольних даних, так і інформації для відновлення початкового стану програмного коду. Однак обсяг інформації, необхідної для відновлення, може займати значну частину обсягу цифрового водяного знака. Це значно зменшує частину обсягу цифрового водяного знака, яка містить безпосередньо контрольні дані моніторингу [13, 14]. В результаті часто виникає ситуація при якій ефективний обсяг цифрового водяного знака є недостатнім для зберігання контрольних даних з необхідним для моніторингу розміром.

**Формулювання мети дослідження**

Збільшити обсяг контрольних даних, які можуть бути приховано збереженими в програмному коді FPGA у вигляді цифрового водяного знака, за рахунок застосування інтервального підходу до формування стего-ключа, що використовується при збереженні та зчитуванні контрольних даних.

**Викладення основного матеріалу дослідження**

Для формування основних положень методу, що пропонується, виконано аналіз двох основних факторів, які впливають на ефективний обсяг цифрового водяного знака: фактора довжини шляху вбудовування та фактора стего-ключа.

Цифровий водяний знак, що використовується для розглянутих методів моніторингу, складається з трьох компонентів (рис. 1): *CD* – контрольні дані; *ISRec* – інформація, необхідна для відновлення початкового стану програмного коду FPGA у момент виконання актів моніторингу, *S* – дані, необхідні для визначення місця розташування компонентів цифрового водяного знака під час моніторингу.

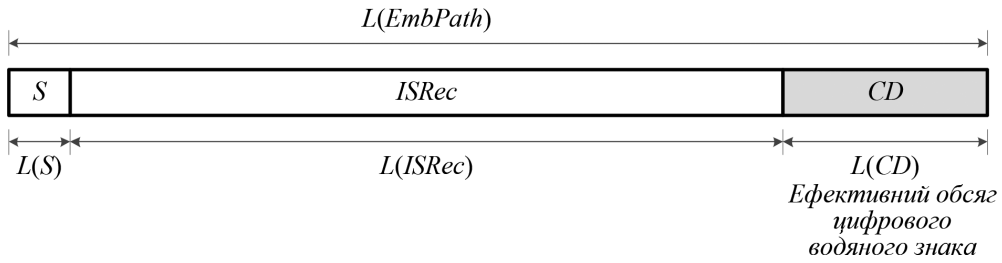


**Рис. 1. Базовий формат цифрового водяного знака, який містить контрольні дані моніторингу**

Цифровий водяний знак, що вбудовується у програмний код FPGA в стеганографічний спосіб, розміщується в просторі коду у вигляді сукупності розрядів, які утворюють шлях вбудовування *EmbPath*. Довжина шляху вбудовування $L(EmbPath)$, виражена в кількості розрядів, залежить як від розміру програмного коду FPGA, так і від параметрів стего-ключа. При цьому ефективний обсяг цифрового водяного знака $L(CD)$, це обсяг доступний для зберігання даних моніторингу в його складі:

$$L(CD) = L(EmbPath) - L(ISRec) - L(S); \quad (1)$$

де $L(EmbPath)$ – кількість розрядів шляху вбудовування; $L(ISRec)$ – кількість розрядів даних, що використовуються для відновлення початкового стану інформаційного об’єкта програмного коду FPGA ; $L(S)$ – довжина поля *S* цифрового водяного знака.

Основним відомим підходом [15], що застосовується для відновлення початкового стану, є метод, заснований на стисненні сукупності розрядів, які знаходяться на шляху вбудовування та збереженні результатів стиснення в полі *ISRec* цифрового водяного знака (рис. 2).

При використанні такого підходу вираз (1) набуває наступного вигляду:

$$L(CD) = L(EmbPath) - L(Com(EmbPath)) - L(S), \quad (2)$$

де $L(Com(EmbPath))$ – обсяг результатів стиснення без втрат сукупності розрядів шляху вбудовування.

Нехай необхідний для виконання моніторингу програмного коду FPGA ефективний обсяг цифрового водяного знака повинен становити *LN* розрядів. Тоді має місце наступне співвідношення:

$$L(EmbPath) - L(Com(EmbPath)) - L(S) \geq LN. \quad (3)$$

Виконання цього співвідношення залежить від довжини шляху вбудовування та ступеня стиснення даних, що знаходяться на цьому шляху. Довжина шляху вбудовування істотно залежить від властивостей стего-ключа. Збільшення довжини шляху вбудовування призводить до збільшення заповненої частини стего-контейнера, що потенційно може негативно позначитися на ступені здатності контейнера приховувати вбудовану в нього

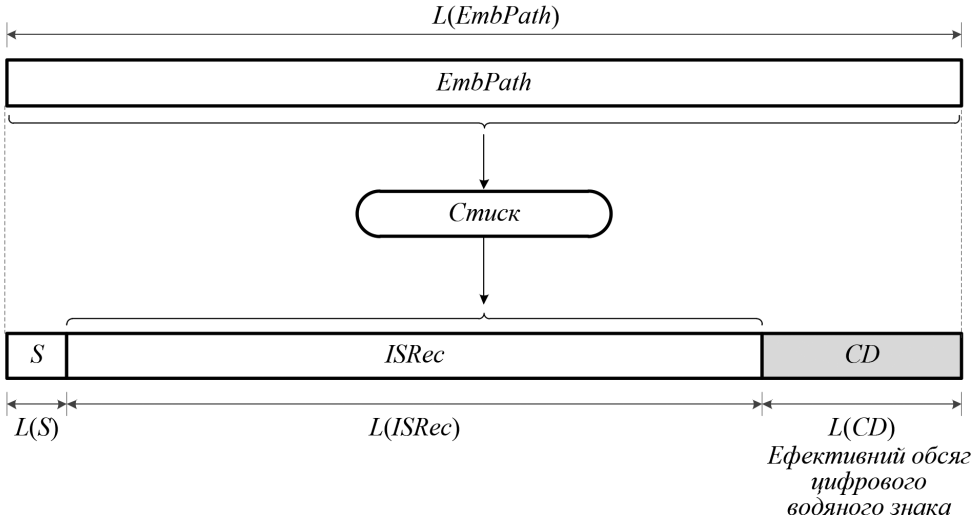


**Рис. 2. Принцип використання стиску без втрат для збереження початкового стану программного коду FPGA**

інформацію. Виходячи з цього, параметри стего-ключа повинні бути підібрані таким чином, щоб при виконанні співвідношення (3) мінімізувати довжину шляху вбудовування.

Базовий стего-ключ для LUT-контейнера, яким є програмний код FPGA, є сукупністю трьох компонентів: *EnumRule* – правила, яке задає порядок залучення блоків LUT-контейнера для формування шляху вбудовування; *DThreshold* – обмеження на кількість підключень блоків LUT до виходу кожного блоку, що входить у шлях вбудовування; *AddrRule* – правило, що визначає адресу розряду програмного коду блоків LUT, який використовується для формування шляхів вбудовування. Параметр *EnumRule* стего-ключа задає нумераційну відстань між блоками LUT, які мають бути включені до шляху вбудовування. Параметр *DThreshold* – числовий поріг, який визначає можливість включення блоку LUT у шлях вбудовування залежно від кількості зв'язків цього блоку з іншими блоками LUT-контейнера. Зменшення значення параметра *EnumRule* і порога *DThreshold* приводить до збільшення довжини шляху вбудовування, а також збільшення обсягу змін програмного коду FPGA.

Грунтуючись на наведених вище факторах, пропонується метод формування стего-ключа, що дозволяє адаптувати ефективний обсяг цифрового водяного знака до структури LUT-контейнера. Основні положення запропонованого методу полягають у наступному.

1. Традиційні методи вбудовування цифрового водяного знака у програмний код FPGA використовують стего-ключ із точковими параметрами. В даному методі пропонується використовувати інтервальні параметри стего-ключа. При цьому кожен параметр являє собою інтервал значень, з якого за правилами методу здійснюється вибір параметрів, адаптованих під конкретний стего-контейнер.

В рамках цього положення запропонованого методу для всіх компонентів стего-ключа формується інтервал значень в такий спосіб, що значення компонента дає шлях вбудовування, найменшої довжини для даної задачі вбудовування цифрового водяного знака. При цьому, застосування кожного наступного значення з даного інтервалу у якості компоненти стего-ключа дає збільшення довжини шляху вбудовування. При виконанні процедури вбудовування цифрового водяного знака, послідовно, починаючи з мінімальних, обробляються значення інтервалів. та перевіряється виконання співвідношення (3). При використанні інтервального методу до декількох компонентів стего-ключа в ключ вводиться додатковий компонент *priority*, який визначає, в якому порядку збільшуються компоненти з декількох інтервалів. Відповідно до зазначеного положення пропонується перейти від точкових значень компонентів *EnumRule* та *DThreshold* стего-ключа до їх інтервальних версій.

Якщо в результаті виконання дій, регламентованих першим положенням запропонованого методу, співвідношення (3) не виконається, далі застосовується друге положення методу.

2. Для вбудовування цифрового водяного знака потенційно можна використати декілька методів, що дають різні довжини шляхів вбудовування. При цьому кожен із зазначених методів має індивідуальний набір параметрів , розглянутих у попередньому положенні методу. Дане положення запропонованого методу передбачає впорядкування методів вбудовування у вигляді послідовності та виконання вбудовування за допомогою компонентів цієї послідовності з перевіркою співвідношення (3). Якщо всі методи послідовності були застосовані з урахуванням першого положення запропонованого методу, але параметри цифрового водяного знака не задовольняють співвідношенню (3), застосовуються дії, регламентовані третім положенням запропонованого методу.

3. Якщо використання першого та другого положення запропонованого методу не привело до отримання необхідного ефективного обсягу цифрового водяного знака, використовується додатковий однорозрядний компонент стего-ключа $m_{prep}$. Цей компонент визначає можливість застосування методу [16] попередньої підготовки інформаційного об'єкта програмного коду FPGA для даних, які містяться в стего-контейнері на шляху вбудовування. Якщо за умов задачі вбудовування цифрового водяного знака використання методу попередньої підготовки не є допустимим, виконується редукція контрольних даних, що змінює вимоги до ефективного обсягу цифрового водяного знака.

4. При витяганні цифрового водяного знака, який було вбудовано в програмний код FPGA відповідно до зазначених вище положень методу, виконується процедура визначення точкових значень з інтервальних компонентів стего-ключа. Точкові значення визначаються відповідно до положень пропонованого методу, використаних при вбудовуванні. Після цього виконується витягання цифрового водяного знака з програмного коду FPGA і розкладання його на компоненти. На основі компонента *ISRec* здійснюється відновлення початкового стану інформаційного об'єкта програмного коду. Далі цей інформаційний об'єкт та контрольні дані, отримані із цифрового водяного знака, приймаються системою моніторингу характеристик безпеки програмного коду.

Послідовність виконання запропонованого методу відповідно до наведених положень, представлена на рис. 3 у вигляді блок-схеми. Блок-схема показана для двокомпонентної послідовності методів вбудовування , що відповідає другому положенню запропонованого методу. Компоненти цієї послідовності – базовий метод. вбудовування

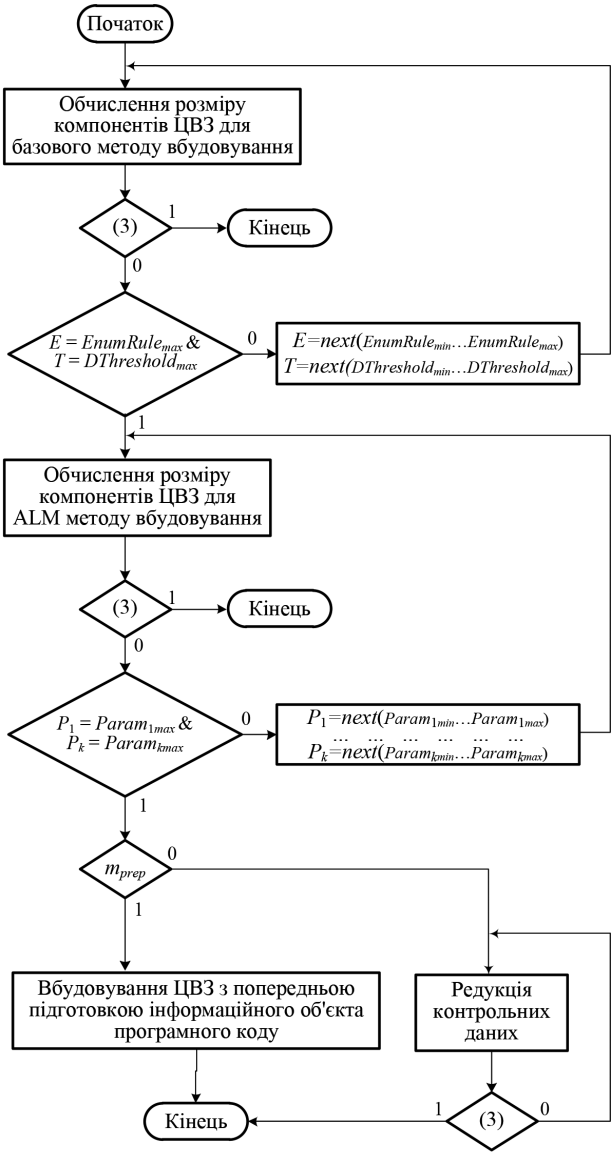


**Рис. 3. Блок схема виконання інтервального методу отримання стего-ключа для вбудовування цифрового водяного знака (ЦВЗ)**

цифрового водяного знака у програмний код FPGA [17] та метод вбудовування, орієнтований на додаткове використання спеціалізованих блоків Adaptive Logic Модулі (ALM) [18] FPGA.

Для експериментального дослідження методу, запропонованого у статті, було розроблено відповідне програмне забезпечення. Дане програмне забезпечення використовує в якості компонентів програмні модулі, які реалізують базовий метод вбудовування цифрового водяного знака в програмний код FPGA, ALM-базований метод вбудовування та метод вбудовування з попередньою підготовкою інформаційного об'єкта програмного коду FPGA. Розроблений програмний додаток приймає інтервальний стего-ключ, ітераційно, відповідно до послідовності дій методу, перебирає інтервальні компоненти цього ключа та отримує розміри компонентів цифрового водяного знака для застосовуваних методів вбудовування.

На основі розробленого програмного додатку було виконано експериментальне дослідження запропонованого методу. Під час проведення експерименту було задіяно 8 FPGA-проєктів різного розміру та призначення. Синтез проєктів та отримання низькорівневого програмного коду FPGA виконувались у системі Intel Quartus Prime [19].

Методика проведення експерименту полягала в порівнянні результатів, отриманих з використанням звичайного та інтервального стего-ключів. В якості компонентів звичайних стего-ключів при цьому використовувались мінімальні значення параметрів інтервальних стего-ключів.

За допомогою обох ключів формувався цифровий водяний знак у просторі кожного з FPGA-проєктів, що беруть участь в експерименті. При достатньому для вирішення задачі моніторингу ефективному обсязі цифрового водяного знака виконувалося вбудовування водяного знака з контрольними даними в інформаційний контейнер програмного коду FPGA. Після цього за стего-ключами проводилося витягання цифрового водяного знака з контейнера, поділ його на компоненти та отримання контрольних даних.

На рис. 4 наведено порівняння, отриманих в результаті експерименту, ефективних обсягів цифрових водяних знаків, при застосуванні звичайного підходу до формування стего-ключа та підходу, запропонованого в даній роботі. На рис. 4, *а* на горизонтальній осі показані номери експериментальних FPGA-проєктів, впорядкованих за зростанням обсягу апаратних ресурсів, на вертикальній осі відкладено отриманий ефективний обсяг цифрового водяного знака, виражений у кількості розрядів. Оскільки FPGA-проєкти на рис. 4, *а* впорядковані за зростанням обсягу апаратних ресурсів, має місце збільшення ефективного обсягу цифрового водяного знака відповідно до кількості блоків LUT в проєкті. Однак, як видно, ефективний обсяг цифрового водяного знака, отриманий при застосуванні запропонованого підходу, для кожного експериментального проєкту є більшим за обсяг, отриманий традиційним шляхом. На рис. 4, *б* показано відносне збільшення ефективного обсягу цифрового водяного знака, отримане за рахунок застосування пропозицій даної роботи. Відносне збільшення ефективного обсягу не має прямої залежності від обсягу ресурсів проєкту, але для великих, за кількостью блоків LUT, проєктів (проєкти 5–8) збільшення обсягу перевищує 20 %. Дуже мале збільшення, отримане для проєкту 4 є результатом специфіки його структури, в якій на множині блоків LUT має місце значно більше зв'язків, ніж для інших проєктів, що були використані в експерименті.

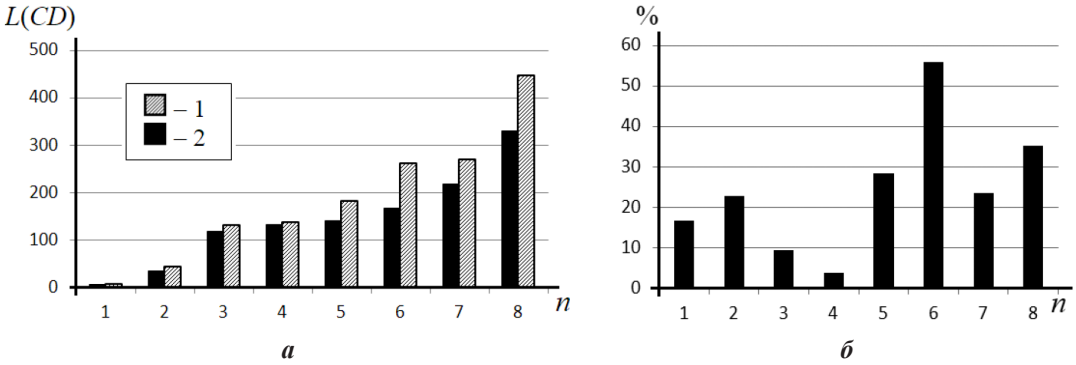


**Рис. 4. Результати експериментального дослідження запропонованого методу:**
**1 – ефективний обсяг цифрового водяного знака при застосуванні запропонованого методу;**
**2 – обсяг при застосуванні звичайного стего-ключа**

В цілому результати експериментів показують, що запропонований метод дає збільшення ефективного обсягу цифрового водяного знака. В середньому за результатами проведеного експериментального дослідження використання запропонованого методу дозволило на 22,8 % збільшити ефективний обсяг цифрового водяного знака. Це дає можливість використати контрольні дані більшої сукупної розрядності для прихованого моніторингу характеристик безпеки програмного коду FPGA. Таким чином, експериментальне дослідження запропонованого підходу показало, що мету даної роботи досягнуто завдяки тому, що метод дозволяє збільшити ефективний обсяг цифрового водяного знака, який використовуються для прихованого зберігання контрольних даних.

**Висновки**

У статті пропонується метод інтервального формування стего-ключа під час вбудовування цифрового водяного знака у програмний код мікросхем FPGA. Метод спрямований на збільшення ефективного обсягу цифрового водяного знака, призначеного для стеганографічного зберігання контрольних даних при виконанні прихованого моніторингу характеристик безпеки програмного коду FPGA.

Особливість запропонованого методу полягає у використанні інтервальних компонентів стего-ключа замість точкових компонентів. Це дозволяє адаптувати ключ до кожного конкретного стего-контейнера програмного коду FPGA з метою збільшення ефективного обсягу цифрового водяного знака, який вбудовується в контейнер.

Проведене експериментальне дослідження методу показало його ефективність порівняно з методами, що базуються на точкових параметрах стего-ключа. Для експериментальних проєктів вдалося досягти сумарного збільшення розрядності контрольних даних, що дозволяє збільшити кількість видів прихованого моніторингу характеристик безпеки програмного коду мікросхем FPGA.

**Список використаної літератури**

1. Tu K., Tang S., Yu X., Josipovic L., Chu Z. FPGA EDA: Design Principles and Implementation. Singapore: Springer, 2024. 248 p.
2. Hajji B., Mellit A., Bouselham L. A Practical Guide for Simulation and FPGA Implementation of Digital Design. Singapore : Springer, 2022. 312 p.
3. Tehranipoor M., Zamiri Azar K., Asadizanjani N., Rahman F., Mardani Kamali H., Farahmandi F. Hardware Security: A Look into the Future. Cham : Springer, 2024. 546 p.
4. Pfleeger C., Pfleeger S., Margulies J. Security in Computing. 6th ed. Boston: Pearson, 2019. 800 p.
5. Anderson R. Security Engineering: A Guide to Building Dependable Distributed Systems. 3rd ed. Indianapolis: Wiley, 2020. 1232 p.
6. Bishop M. Computer Security. 2nd edn. USA, Boston: Addison-Wesley, 2018.
7. Stallings W. Cryptography and Network Security: Principles and Practice. 7th edn. United Kingdom, Harlow : Pearson Education Limited, 2017.
8. Bossuet L., Torres L. Foundations of Hardware IP Protection. USA, New-York: Springer, 2018.
9. Yahya A. Steganography Techniques for Digital Images. Springer, 2018. 122 p.
10. Shih F. Digital Watermarking and Steganography: Fundamentals and Techniques. 2nd ed. USA, Boca Raton : CRC Press. 2017. 320 p.
11. Fan L., Chan C., Yang Q. Digital Watermarking for Machine Learning Model: Techniques, Protocols and Applications. Singapore: Springer, 2023. 300 p.
12. Fridrich J. Steganography in Digital Media, Cambridge University Press, New York, 2010.
13. 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
14. Zhou, H., Zhang, Y., & Wang, X. Data Integrity Verification and Monitoring in Cloud Storage. *Journal of Cloud Computing: Advances, Systems and Applications*. 2020. 9(1). P. 1–12.
15. Fridrich J., Goljan, M., Du, R. Lossless Data Embedding – New Paradigm in Digital Watermarking. *EURASIP Journal on Advanced Signal Processing*. 2002. P. 185–196.
16. Zashcholkin K., Drozd O, Shaporin R, Ivanova O, Sulima Y. Increasing the Effective Volume of Digital Watermark Used in Monitoring the Program Code Integrity of FPGA-Based Systems. *IEEE East-West Design and Test Symposium (EWDTS)*. 2019. P. 53–58.
17. Zashcholkin K., Drozd O., Ivanova O., Sulima Y. Compositional method of FPGA program code integrity monitoring based on the usage of digital watermarks. *Applied Aspects of Information Technology*. Odessa, 2019. Vol. 2, № 02. P. 138–152.
18. Zashcholkin K., Drozd O., Ivanova O., Shaporin R., Veselska O., Stepova H. Embedding the Digital Watermarks into FPGA-Projects Containing the Adaptive Logic Modules. *Dependable Systems, Services and Technologies (DESSERT'2019)*: Proceedings of 10th IEEE International Conference. United Kingdom, Leeds, 2019. P. 179–183.
19. Intel Quartus, https://www.intel.com/content/www/us/en/products/details/ fpga/development-tools/quartus-prime.html

**References**

1. Tu K., Tang S., Yu X., Josipovic L., Chu Z. (2024) FPGA EDA: Design Principles and Implementation. Singapore : Springer.
2. Hajji B., Mellit A., Bouselham L. (2022) A Practical Guide for Simulation and FPGA Implementation of Digital Design. Singapore : Springer.
3. Tehranipoor M., Zamiri Azar K., Asadizanjani N., Rahman F., Mardani Kamali H., Farahmandi F. (2024) Hardware Security : A Look into the Future. Cham: Springer.

4. Pfleeger C., Pfleeger S., Margulies J. (2019) Security in Computing. 6th ed. Boston : Pearson.
5. Anderson R. (2020) Security Engineering: A Guide to Building Dependable Distributed Systems. 3rd ed. Indianapolis : Wiley.
6. Bishop M. (2018) Computer Security. 2nd edn. Boston : Addison-Wesley
7. Stallings W. (2017) Cryptography and Network Security: Principles and Practice. 7th edn. United Kingdom, Harlow : Pearson Education Limited.
8. Bossuet L., Torres L. (2018) Foundations of Hardware IP Protection. USA, New-York : Springer.
9. Yahya A. (2018) Steganography Techniques for Digital Images. Springer.
10. Shih F. (2017) Digital Watermarking and Steganography: Fundamentals and Techniques. 2nd ed. USA, Boca Raton: CRC Press.
11. Fan L., Chan C., Yang Q. (2023) Digital Watermarking for Machine Learning Model: Techniques, Protocols and Applications. Singapore: Springer.
12. Fridrich J. (2010) Steganography in Digital Media, Cambridge University Press, New York.
13. Ke Y., Liu J., Zhang M., Su T., Yang X. (2018) Steganography Security: Principle and Practice. *IEEE Access*, Vol. 6, 73009–73022. DOI: https://doi.org/10.1109/ ACCESS.2018.2881680
14. Zhou, H., Zhang, Y., & Wang, X. (2020) Data Integrity Verification and Monitoring in Cloud Storage. *Journal of Cloud Computing: Advances, Systems and Applications*, 9(1), 1–12.
15. Fridrich J., Goljan, M., Du, R. (2002) Lossless Data Embedding – New Paradigm in Digital Watermarking. *EURASIP Journal on Advanced Signal Processing*, 185–196.
16. Zashcholkin K., Drozd O, Shaporin R, Ivanova O, Sulima Y. (2019) Increasing the Effective Volume of Digital Watermark Used in Monitoring the Program Code Integrity of FPGA-Based Systems. *IEEE East-West Design and Test Symposium (EWDTS)*, 53–58.
17. Zashcholkin K., Drozd O., Ivanova O., Sulima Y. (2019) Compositional method of FPGA program code integrity monitoring based on the usage of digital watermarks. *Applied Aspects of Information Technology*, Vol. 2, N 02, 138–152.
18. Zashcholkin K., Drozd O., Ivanova O., Shaporin R., Veselska O., Stepova H. (2019) Embedding the Digital Watermarks into FPGA-Projects Containing the Adaptive Logic Modules. *Dependable Systems, Services and Technologies (DESSERT'2019)*: Proceedings of 10th IEEE International Conference. United Kingdom, Leeds, P. 179–183.
19. Intel Quartus, https://www.intel.com/content/www/us/en/products/details/ fpga/development-tools/quartus-prime.html