# ІНФОРМАТИКА, ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА АВТОМАТИЗАЦІЯ

УДК 004.056.53
DOI https://doi.org/10.32782/2663-5941/2023.6/08

***Антощук С.Г.***
Національний університет «Одеська політехніка»

***Іванова О.М.***
Національний університет «Одеська політехніка»

## ПІДХІД ДО ЗБІЛЬШЕННЯ ОБСЯГУ ПРИХОВАНОГО ЗБЕРІГАННЯ КОНТРОЛЬНИХ ДАНИХ, ПРИЗНАЧЕНИХ ДЛЯ МОНІТОРИНГУ FPGA-КОМПОНЕНТІВ КОМП'ЮТЕРНИХ СИСТЕМ

*Стаття присвячена забезпеченню моніторингу програмного коду мікросхем FPGA. Головна мета моніторингу полягає в захист програмного коду від зловмисних спотворень, а також в контролі поширення програмного коду та легітимність його використання. Для виконання моніторингу використовуються контрольні дані, які зберігаються разом з об'єктом програмного коду та використовуються у процесі моніторингу. Розглянуто основні способи зберігання контрольних даних моніторингу: зберігання у складі об'єкта програмного коду, файлової системи, у віддаленій базі даних. Описано недоліки зазначених способів зберігання.*

*Як перспективний виділяється стеганографічний підхід до зберігання контрольних даних моніторингу. У рамках цього підходу контрольні дані вбудовуються у програмний код таким чином, що після вбудовування вони складають із програмним кодом єдине ціле. При цьому зовнішній спостерігач не має можливості відрізнити розряди контрольних даних від розрядів програмного коду. Зазначено, що до мікросхем FPGA стеганографічний підхід застосовується за рахунок вбудовування додаткових даних у програмні коди блоків LUT, шляхом еквівалентного перетворення їх програмного коду. Однак при такому підході можливе вбудовування у програмний код FPGA відносно невеликого обсягу додаткових даних. У роботі пропонується підхід до збільшення доступного для вбудовування обсягу даних за рахунок використання особливостей арифметичних операцій з плаваючою крапкою, що виконуються на FPGA.*

*При виконанні таких операцій часто результат має більшу розрядність, ніж операнди. При цьому специфікація операції вимагає приведення формату результату до формату операндів. Для здійснення такого приведення частина розрядів результату відкидається з наступною корекцією, що компенсує це відкидання. Наведено приклад відкидання розрядів та корекції для операції множення чисел з плаваючою крапкою. У роботі пропонується в якості додаткового ресурсу для стеганографічного вбудовування контрольних даних використовувати блоки LUT, які обчислюють розряди арифметичних операцій, що відкидаються. Пропонується модель стеганографічного вбудовування контрольних даних програмного коду FPGA з використанням зазначеного інформаційного ресурсу. Виконано експериментальну оцінку додаткового обсягу даних, що вбудовуються, отриманого за рахунок використання запропонованого інформаційного ресурсу. Зроблено висновки щодо ефективності підходу, запропонованого в роботі, для збільшення обсягу контрольних даних, які вбудовуються в програмний код FPGA.*

***Ключові слова:*** *моніторинг програмного коду, мікросхеми FPGA, стеганографічне вбудовування даних, наближена обробка даних, арифметичні операції з плаваючою крапкою.*

**Вступ та постановка проблеми.** Значну частину цифрових компонентів комп'ютерних систем складають програмовані компоненти. Істотної особливістю цих компонентів є те, що їх функціонування може бути змінено розробником на будь-якому етапі життєвого циклу. До таких компонентів відносяться мікропроцесори, мікроконтролери та програмовані логічні інтегральні схеми (ПЛІС). Ні відміну від мікропроцесорів та мікроконтролерів, які мають послідовний принцип виконання програми, мікросхеми ПЛІС можуть змінювати свою структуру під дією програмного коду, ство-

реного розробником [1]. Така можливість зміни структури мікросхеми, відповідно до задачі та паралельна організація мікросхеми, обумовлюють значно більшу продуктивність ПЛІС порівняно, з мікропроцесорами та мікроконтролерами [2].

В предметних областях, задачі яких потребують реалізації програмно-керованих високопродуктивних обчислювальних ресурсів, переважно використовуються програмовані логічні інтегральні схеми, найбільш досконалими представниками яких є мікросхеми FPGA (Field Programmable Gate Array). Здебільшого ці мікросхеми використовуються в таких галузях як телекомунікації, промислові технології та засоби транспорту. Також ці мікросхеми широко застосовані у вирішенні задач обробки даних, що потребують значної продуктивності обчислень [3].

Мікросхеми FPGA мають відмінності як архітектурного, так і безпекового плану, від мікропроцесорів та мікроконтролерів. Крім того, можливість забезпечення високої продуктивності робить FPGA вельми часто використовуваною елементною базою для реалізації комп'ютерних систем критичного застосування [4]. Серед засобів захисту FPGA компонентів від зловмисного втручання особливу роль відіграє оперативний моніторинг стану програмного коду цих компонентів. Моніторинг проводиться за окремими характеристиками (цілісністю, автентичністю, шляхами розповсюдження, легітимністю використання) або комплексно за групою характеристик. При цьому найбільш дієві підходи до організації моніторингу базуються на використанні контрольних даних, які зберігаються разом з програмним кодом або в якості його частини.

**Аналіз останніх досліджень і публікацій.** Для виконання моніторингу програмного коду використовуються контрольні дані, які обчислюються або призначаються визначеним чином. Обчислювальні контрольні дані повинні відповідати наступним вимогам:

1) контрольні данні повинні мати значно менший обсяг, ніж програмний код який контролюються за їх допомогою;

2) незначні зміни програмного коду, який контролюється, повинні приводити до суттєвих змін контрольних даних.

Прикладами обчислювальних контрольних даних є хеш-суми [5] для моніторингу цілісності програмного коду або коди аутентифікації для моніторингу його автентичності.

Атрибутивні контрольні дані, на відміну від обчислюваних, призначаються об'єкту програмного коду. Прикладом атрибутивних контрольних даних є контрольні дані, які застосовуються в контролі розповсюдження програмного коду або в контролі легітимності його використання [6].

На рис. 1 наведено схему виконання моніторингу програмного коду програмованих компонентів на етапі підготовки контрольних даних. На цьому етапі для об'єкта програмного коду формуються контрольні дані. Обчислювані контрольні дані створюються за допомогою відповідних алгоритмів, а атрибутивні контрольні дані призначаються об'єкту програмного коду. Отриманий набір контрольних даних оголошується еталонним і зберігається.

В момент виконання моніторингу програмного коду (рис. 2) обчислювані контрольні дані повторно обчислюються для програмного коду, щодо якого здійснюється моніторинг, а атрибутивні контрольні дані зчитуються з об'єкта програмного коду. Далі цей набір контрольних даних порівнюється з еталонними контрольними даними, які були збережені на етапі підготовки до мониторінгу. За результатами порівняння робиться висновок про проходження або не проходження моніторингу певного виду.

Аналіз інцидентів останніх років, пов'язаних зі дестабілізацією роботи технічних об'єктів підвищеного ризику показує, що найзначнішим чинником збоїв в роботі критично важливих комп'ютерних систем є зловмисне втручання в їх функціонування [7]. Одним з найефективніших підходів до протидії втручанню є оперативний моніторинг стану програмного коду цих компонентів. Виконання моніторингу потребує формування контрольних даних в момент підготовки програмного коду моніторингу, та їх зберігання з метою використання під час виконання актів моніторингу. Основним способом обходу моніторингу програмного коду для здійснення втручання є фальсифікація контрольних даних. Найбільш суттєвим фактором, який обумовлює можливість втручання є спосіб зберігання контрольних даних.

В практиці побудови систем контролю програмного коду використовується декілька підходів до зберігання контрольних даних. Один з часто застосовуваних підходів базується на збереженні контрольних даних разом з інформаційним об'єктом програмного коду в файлової системі або пам'яті, чи які базуються на приєднанні контрольних даних до інформаційного об'єкта програмного коду і зберіганні в якості його частина [8]. Цей підхід має наступні недоліки: очевидність факту виконання моніторингу; доступність

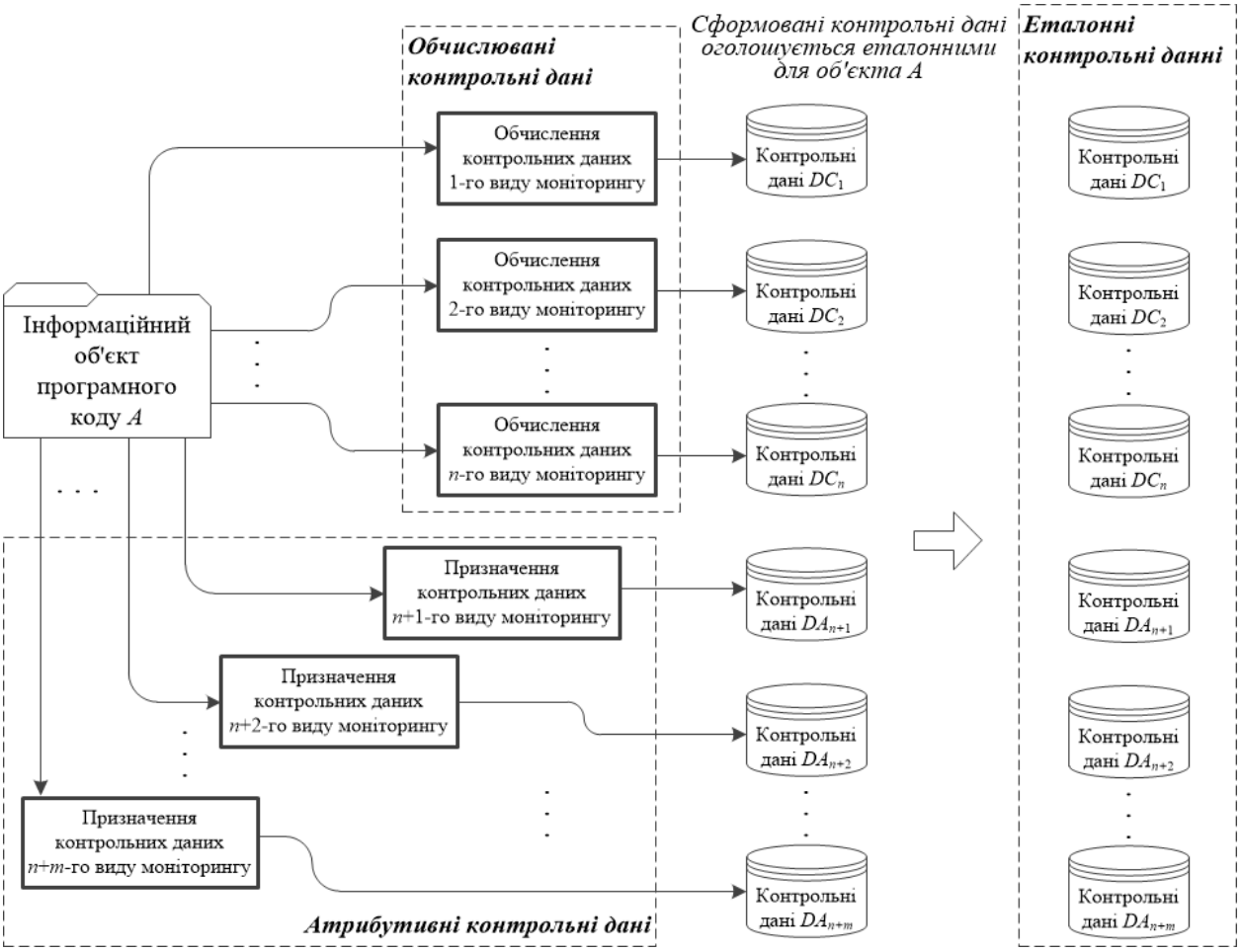


**Рис. 1. Підготовка та збереження контрольних даних при виконанні моніторингу програмного коду**

еталонних контрольних даних для зчитування, аналізу та можливої фальсифікації.

Також часто використовуються підходи, в межах яких контрольні дані зберігаються в централізованій базі даних та отримуються з неї в момент виконання процедури моніторингу. Ці підходи пов'язані з проблемою організації доступу до бази даних; складністю захисту бази даних від витоків та не зменшують можливість інсайдерського втручання [9].

Один з перспективних підходів до зберігання контрольних даних базується на використанні стеганографічного [10] приховування даних безпосередньо в середовищі об'єкта програмного коду FPGA. Цей підхід має переваги, які полягають у приховуванні контрольних даних та факту виконання моніторингу, а також у забезпеченні утворення єдиного цілого між об'єктом програмного коду та контрольними даними [11]. Однак вони мають і недоліки, що полягають у відносно малому доступному обсязі даних, які можна зберегти в стеганографічний спосіб [12]. Це обмежує обсяг контрольних даних та кількість видів моніторингу, які можна застосувати до програмного коду.

**Мета статті.** Збільшити обсяг контрольних даних, які можуть бути збереженими в програмному коді FPGA в стеганографічний способі за рахунок використання структурної надмірності в арифметичних пристроях у складі FPGA-проєктів.

**Виклад основного матеріалу.** Специфікації багатьох операцій наближеною обробки даних вимагають однаковості формату операндів та результату [13–14]. Однак при цьому вельми часто виникає ситуація, при якій результат операції природно має більшу розрядність в порівнянні з розрядністю операндів. У цьому випадку спочатку виконується обчислення повного результату, який потім доводиться до формату операндів шляхом відкидання деякої кількості молодших розрядів з наступним округленням.

Одним з прикладів таких операцій є арифметичні операції над числами з плаваючою крапкою. Числа з плаваючою крапкою містять

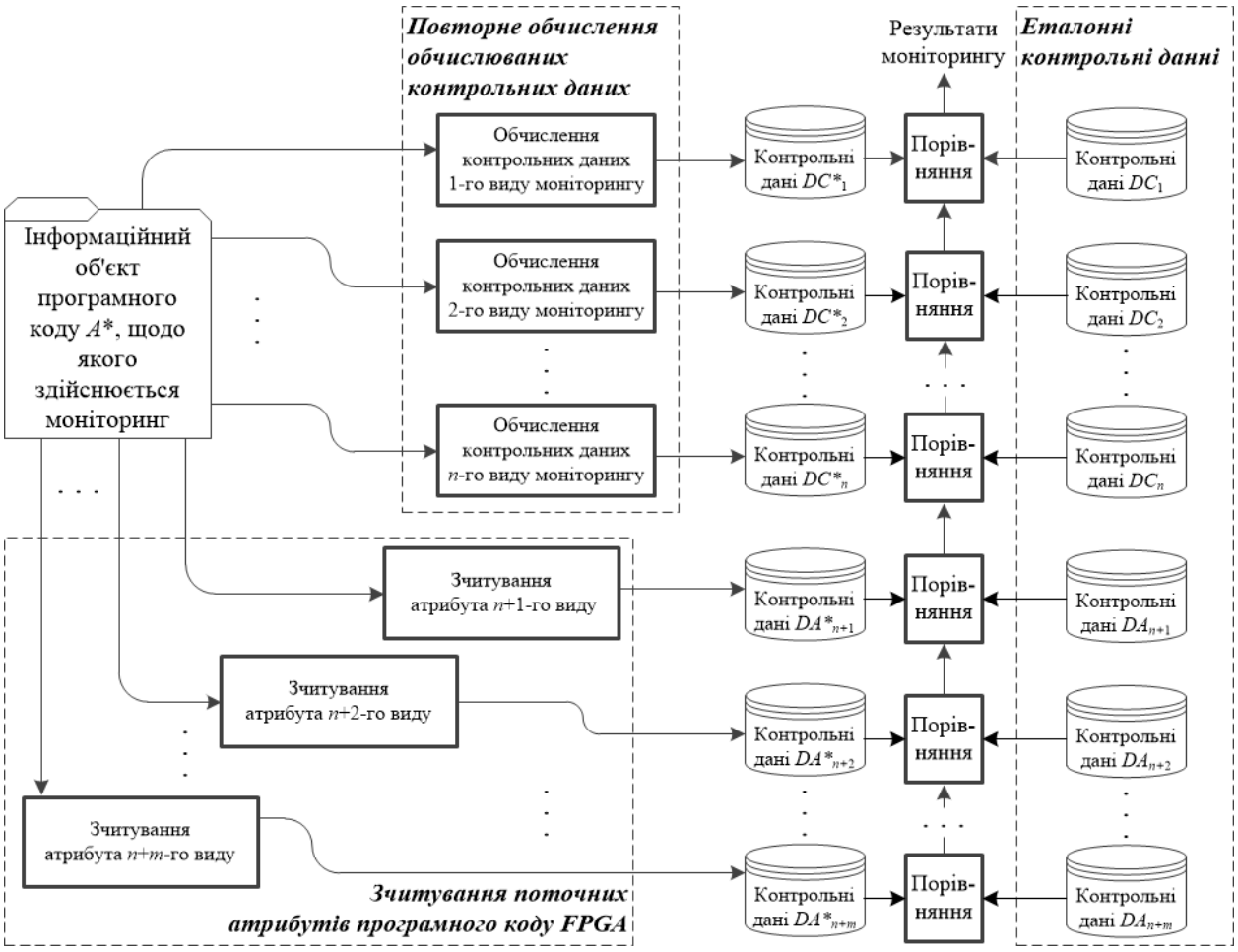


**Рис. 2. Використання збережених контрольних даних на етапі здійснення мониторінгу програмного коду**

в якості основних компонентів мантису та порядок. Порядки при виконанні арифметичних операцій обробляються точно, але мантиси обробляються наближено. При цьому найчастіше специфікація операції вимагає, щоб розрядність мантис результату дорівнювала розрядності мантис операндів.

Наприклад, для отримання результату помноженні чисел з плаваючою крапкою порядки операндів підсумовуються, а мантиси помножуються. Результатом помноження двох двійкових $n$-розрядних чисел є $2n$-розрядне двійкове число. Результат операції помноження чисел з плаваючою крапкою найчастіше повинен мати розрядність мантиси, що дорівнює розрядності мантиси операндів. Для приведення $2n$-розрядного результату помноження мантис до $n$-розрядного формату молодші з $2n$ розрядів відкидаються, а результуючий порядок коректується, щоб компенсувати це відкидання (рис. 3).

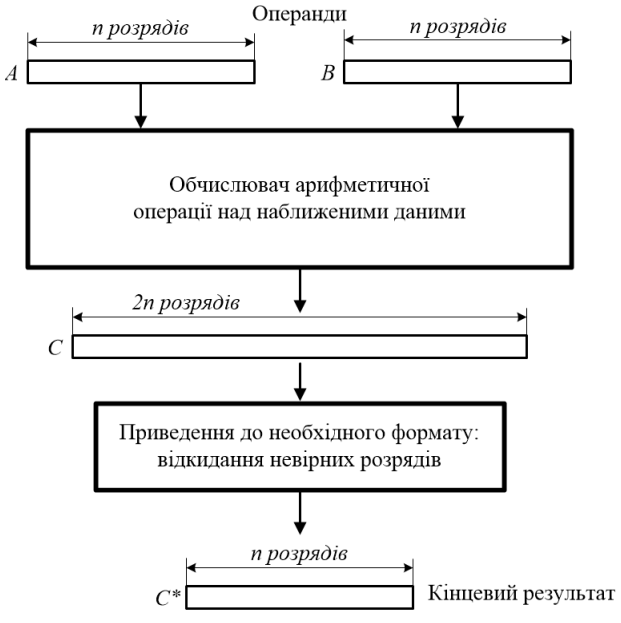


**Рис. 3. Приведення розрядності мантис результату до розрядності мантис операндів**

Таким чином, при виконанні арифметичних операцій над числами з плаваючою крапкою, у випадку необхідності приведення розрядності мантис результату до розрядності мантис операндів, виконується два крок:

– обчислюється повна версія мантиси;

– розрядність повної версії мантис приводиться до розрядності мантиси операндів, шляхом відкидання молодших розрядів.

При виконанні зазначених операцій, множина розрядів результату складається з двох непересічних підмножин: несуттєвих розрядів (що відкидаються) та суттєвих розрядів (що залишаються) в кінцевому результаті операції. Спотворення несуттєвих молодших розрядах результату не впливають на підсумковий результат операції. При реалізації таких арифметичних операцій на FPGA в її структурі можна виділити елементарні обчислювальні блоки LUT, які беруть участь у формуванні тільки несуттєвих розрядів результату. В силу цього програмний код таких блоків може бути модифікований і така модифікація не вплине на результат роботи пристрою.

Наявність множини несуттєвих розрядів, які відкидаються в процесі обчислень та наявність множини елементарних блоків LUT FPGA, які ці розряди обчислюють, робить можливим використання програмного коду таких блоків LUT в якості інформаційного ресурсу для стеганогрфічного збереження контрольних даних моніторингу програмного коду. Узагальнення виділення та використання зазначеного інформаційного ресурсу представлено у вигляді моделі стеганографічного зберігання контрольних даних в середовищі програмного коду LUT FPGA. Модель, що пропонується являє собою кортеж виду:

$$M_A = <A, B, C, C_E, C_{IE}, LUTs, L_E, L_{IE}, L_M>$$

де $A = <a_n, a_{n-1}, \dots, a_1>$ та $B = <b_n, b_{n-1}, \dots, b_1>$ – $n$-розрядні операнди арифметичної операції; $C = <c_{2n}, c_{2n-1}, \dots c_{n+1}, c_n, c_{n-1}, \dots, c_1>$ – $2n$-розрядний повний результат; $C_E$ та $C_{IE}$ – відповідно суттєві та несуттєві (ті, що відкидаються) розряди результату; $LUTs$ – множина блоків LUT FPGA, які використовуються для реалізації арифметичної операції; $L_E$ та $L_{IE}$ –підмножини блоків LUT, які беруть участь в обчисленні відповідно суттєвих та несуттєвих розрядів результату; $L_M$ – підмножина блоків LUT, які одночасно беруть участь в обчисленні, як суттєвих, так і несуттєвих розрядів результату.

Співвідношення між розрядами та підмножинами блоків LUT FPGA в межах моделі $M_A$ графічно показані на рис. 4. Блоки LUT $L_{OIE} = L_{IE} \setminus L_E$, які беруть участь в обчисленні тільки несуттєвих розрядів результату в подальшому будуть називатися несуттєвими блоками LUT. Програмні коди цих блоків складають надлишковий ресурс мікросхем FPGA, який виявляються моделюю $M_A$ та може бути використаний для стеганографічного зберігання контрольних даних.

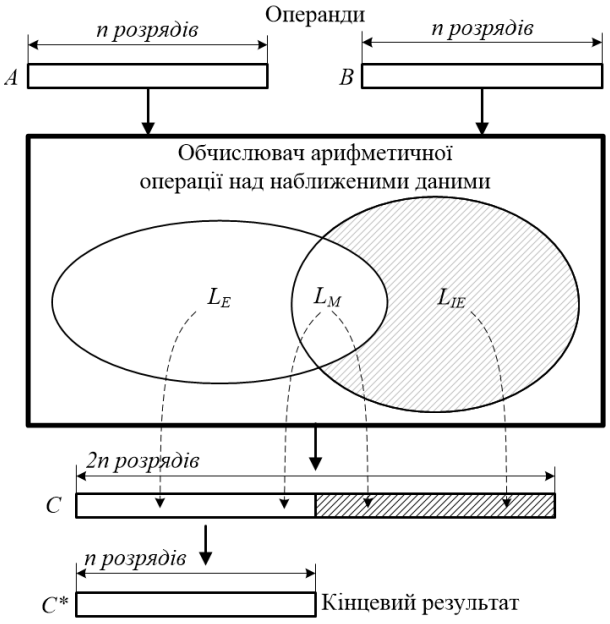


**Рис. 4. Множини блоків LUT FPGA, задіяних при обчисленні розрядів, що залишаються та відкидаються**

Для оцінки обсягу даних, які можуть бути вбудовані в програмний код блоків LUT FPGA відповідно до моделі $M_A$ с залученням надмірних ресурсів було проведено експериментальне дослідження. Для здійснення оцінки розроблено програмну модель, яка відтворює схеми арифметичних обчислювачів з наближеними даними в середовищі FPGA. В якості цільової експериментальних схем обрані помножувачі мантис з плаваючою крапкою з розрядністю операндів 4, 6, 8, 12, 16, в якості мікросхеми синтезу використовувалася FPGA Intel Cyclone IV EP4CE15F23A7, синтез здійснювався в середовищі автоматизованого проектування Intel Quartus Prime [15].

Розроблена програмна модель послідовно перебирає значення операндів на вході схеми помножувача. При цьому зберігаються правильний $2n$-розрядний результат множення і правильні значення на виходах кожного з блоків LUT схеми. Після цього значення на виходах блоків LUT послідовно інвертуються. При цьому модель виконує аналіз того, на які з бітів результату (ті, що відкидаються або ті, що залишаються) впливає кожне з інвертувань. Блоки LUT, інвертування правильних значень яких, призводило до

спотворення тільки тих розрядів результату, що відкидаються, відносяться до несуттєвих блоків. При цьому використано наступне вирішальне правило віднесення блоків LUT до множини $L_{IE}$:

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

де $max\_error(LUT_i)$ – значення максимальної помилки, поставлене у відповідність блоку $LUT_i$ – модуль різниці правильного значення результату і значення результату, отриманого при інвертуванні вихідного значення блоку $LUT_i$.

В результаті дослідження розробленої програмної моделі визначено, що частка несуттєвих блоків LUT в експериментальних FPGA-проектах склала від 39,3% до 42,5%. Оскільки спотворення програмного коду таких блоків LUT не призводить до спотворення кінцевих результатів обчислень, то ці блоки складають ресурс для стеганографічного вбудовування даних.

**Висновки.** Для прихованого зберігання контрольних даних моніторингу програмного коду мікросхем FPGA зазвичай використовуються методи еквівалентного перетворення програмного коду. Пропонується удосконалити ці методи за рахунок використання додаткового надмірного інформаційного ресурсу. В якості такого ресурсу пропонується застосувати програмні коди блоків LUT FPGA, які призначенні для несуттєвих розрядів мантис результату арифметичних операцій з плаваючої крапкою. Оцінено обсяг додаткового ресурсу стеганографічного приховування даних. Зазначений обсяг дозволяє підвищити загальний обсяг контрольних даних моніторингу програмного коду FPGA та збільшити кількість видів моніторингу, що можуть бути реалізовані на основі використання збережених контрольних даних.

**Список літератури:**

1. Ledin J. Architecting High-Performance Embedded Systems: Design and build high-performance real-time digital systems based on FPGAs and custom circuits. Packt Publishing, 2021. 376 p.

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

3. Waidyasooriya H., Hariyama M., Uchiyama K. Design of FPGA-Based Computing Systems with OpenCL. Switzerland, Cham: Springer, 2018. doi: 10.1007/978-3-319-68161-0

4. Drozd, O., Kuznietsov M., Martynyuk O., Drozd M.: A method of the hidden faults elimination in FPGA projects for the critical applications. *Proceedings of the IEEE International Conference DESSERT'2018*. Kyiv, Ukraine. pp. 231–234 (2018). doi: 10.1109/DESSERT.2018.8409131.

5. Awad A., Fairhurst M. Information Security. Foundations, technologies and applications. UK, Lon-don: The Institution of Engineering and Technology, 2018. 426 p.

6. Bhunia S., Tehranipoor M. Hardware Security: A Hands-on Learning Approach. USA, Boston: Morgan Kaufmann Publ., 2018. 526 p.

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

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

9. Bossuet L., Torres L. Foundations of Hardware IP Protection. USA, New-York: Springer, 2018.

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

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

12. Fridrich J. Steganography in Digital Media, Cambridge University Press, New York, 2010.

13. Ullah S., Kumar A. Approximate Arithmetic Circuit Architectures for FPGA-based Systems. Springer, 2023. 194 p.

14. 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: 10.1109/TC.2017.2716355.

15. Intel Quartus, https://www.intel.com/content/www/us/en/products/details/ fpga/development-tools/ quartus-prime.html

**Antoschuk S.H., Ivanova O.M. APPROACH TO INCREASING THE VOLUME OF HIDDEN STORAGE OF CONTROL DATA FOR MONITORING FPGA COMPONENTS OF COMPUTER SYSTEMS**

*The article is devoted to providing monitoring of the program code of FPGA chips. The main purposes of monitoring are: protection of program code from malicious distortions, control of program code distribution, legitimacy of its use. To perform monitoring uses control data that are stored together with the object of program code and used in the implementation of monitoring. The main ways of storing the monitoring control*

*data are considered: storage as part of the program code object, in the file system, in a remote database. The disadvantages of these storage methods are described.*

*The steganographic approach to the storage of monitoring control data is a promising one. Within the framework of this approach the control data are embedded into the program code in such a way that after embedding they form a single whole with the program code. In this case an external observer has no possibility to distinguish the bits of control data among the bits of the program code. It is noted that in relation to the program code of FPGA chips steganographic approach is applied by embedding additional data in the program codes of LUT units by equivalent transformation of their program code. However, with this approach, it is possible to embed a relatively small amount of additional data into the FPGA program code. This paper proposes an approach to increase the amount of data available for embedding by exploiting the features of floating point arithmetic operations performed on FPGA.*

*When performing operations of this kind, the result is often larger than the operands. In this case, the specification of the operation requires that the format of the result be brought to the format of the operands. In this case, some bits of the result are discarded, followed by a correction that compensates for this discarding. An example of such bit discarding and correction for the operation of multiplication of floating-point numbers is given. The paper proposes to use LUT units that compute the discarded bits of arithmetic operations as an additional resource for steganographic data embedding. We propose a model for steganographic embedding of control data into the FPGA program code using the specified information resource. Experimental estimation of the additional amount of embedded data, which is provided through the use of the proposed information resource. Conclusions are made regarding the effectiveness of the approach proposed in the paper to increase the volume of control data embedded in the program code.*

***Key words:*** *program code monitoring, FPGA chips, steganographic data embedding, approximate data processing, floating point arithmetic.*