ISSN 2519-2310

CS&CS Journal

KARAZIN UNIVERSITY
CLASSICS AHEAD OF TIME

1(19) 2021

COMPUTER SCIENCE AND CYBERSECURITY

КОМП'ЮТЕРНІ НАУКИ ТА КІБЕРБЕЗПЕКА

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені В.Н.КАРАЗІНА
ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ имени В.Н. КАРАЗИНА
V.N. KARAZIN KHARKIV NATIONAL UNIVERSITY

КОМП'ЮТЕРНІ НАУКИ ТА КІБЕРБЕЗПЕКА
КОМПЬЮТЕРНЫЕ НАУКИ И КИБЕРБЕЗОПАСНОСТЬ
COMPUTER SCIENCE AND CYBERSECURITY
(CS&CS)

Issue 1(19) 2021

Заснований 2015 року

Міжнародний електронний науково-теоретичний журнал
Международный электронный научно-теоретический журнал
International electronic scientific journal

The journal publishes research articles on theoretical, scientific and technical problems of effective facilities development for computer information communication systems and on information security problems based on advanced mathematical methods, information technologies and technical means.

The journal is published every six months.
Approved for placement on the Internet by Academic Council of the Karazin Kharkiv National University (June 29, 2021, protocol No.7)

The journal has Digital Object Identifier: **10.26565/2519-2310**.

Editorial office:
V.N. Karazin Kharkiv National University
Svobody Sq., 6, office 315a, Kharkiv, 61022, Ukraine *(North building of University, 3th floor)*
Phone: +38 (057) 705-10-83
E-mail: *cscsjournal@karazin.ua*
Web-page: *http://periodicals.karazin.ua/cscs (Open Journal System)*

Published articles have been internally and externally peer reviewed

TABLE OF CONTENTS

Issue 1(19) 2021

УДК 681.04 DOI: 10.26565/2519-2310-2021-1-01

МЕТОД ВИКОНАННЯ ОПЕРАЦІЇ ДОДАВАННЯ ЗАЛИШКІВ ЧИСЕЛ ЗА МОДУЛЕМ

Віктор Краснобаев, Катерина Кузнецова, Михайло Багмут

Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 4, Харків, 61022, Україна
v.a.krasnobaev@gmail.com, kate7smith12@gmail.com, mikhail56@ukr.net

Рецензент: В'ячеслав Калашников, д. ф.-м.н., проф., Технологічний університет Монтеррея, 64849 Монтеррей, Нуево-Леон, Мексика
kalash@itesm.mx

Надійшло: Листопад 2020.

Анотація: *Суматор двох чисел є одним з компонентів комп'ютерної системи (КС) в позиційній двійковій системі числення (ПСЧ). Зокрема, компонентами КС є також суматори за модулем m_i двох чисел. Даний тип суматори за модулем широко використовуються як в ПСЧ, так і в позиційній системі числення в залишкових класах (СЗК). Важливою і актуальною науково-прикладною задачею є завдання побудови суматорів, що працюють за довільним модулем m_i СЗК. Якщо залишки a_i і b_i чисел A і B в СЗК, представлені в двійковій ПСЧ, тоді акумулятор двох залишків a_i і b_i по модулю m_i є послідовна сукупність з двійкових однорозрядних суматорів (ДОС). Метою статті є розробка методу виконання операції модульного додавання $(a_i + b_i) \bmod m_i$ залишків двох залишків чисел, за довільним модулем на основі використання позиційного двійкового суматора за модулем $M = 2^n - 1$. Запропонований в статті метод виконання операції модульного додавання, заснований на використанні відомої структури позиційних двійкових суматорів за модулем $M = 2^n - 1$. Технічно, завдання побудови структури суматора полягає в необхідності забезпечити умови, при яких вихідний суматор в ПСЧ за модулем M, виконував би операцію складання за модулем m_i. Дана процедура здійснюється шляхом введення додаткових зв'язків виду $X_{\downarrow i \uparrow j}$ в позиційний суматор за модулем $M = 2^n - 1$, де вираз $X_{\downarrow i \uparrow j}$ позначає односторонню зв'язок між виходом j-го ДОС і входом i-го ДОС. Наведені приклади реалізації методу виконання операції модульного додавання для різних значень залишків a_i і b_i. Аналіз розглянутих прикладів показав практичну придатність запропонованого в статті методу. Він може бути використаний, як в ПСЧ, так і в СЗК.*

Ключові слова: *суматор двох чисел; суматор за довільним модулем; операція модульного додавання двох залишків; позиційна двійкова система числення (ПСЧ); непозиційних система числення в залишкових класах (СЗК).*

1 Вступ

Основою виконання арифметичної операції додавання двох чисел в непозиційній системі числення залишкових класів (СЗК) є операція $(a_i + b_i) \bmod m_i$ додавання відповідних залишків a_i та b_i по відповідним основам (модулям) m_i $(i = \overline{1,k})$ СЗК. Операція арифметичного додавання двох чисел в СЗК здійснюється незалежно від інших залишків і паралельно в часі по кожної i-ї основі СЗК [1-3]. Модульна операція складання $(a_i + b_i) \bmod m_i$ здійснюється на основі використання малоразрядних двійкових суматорів за модулем, на основі використання суматорів за модулем $M = 2^n - 1$. Даний підхід надає широкий вибір варіантів можливої реалізації внутрішньої структури такого суматору. Це дозволяє в повній мірі використовувати наявний теоретичний практичний досвід проектування двійкових суматорів [4-5]. Відомо, що одним з основних компонентів комп'ютерної системи (КС) в позиційній системі числення (ПСЧ) є суматор двох чисел. Зокрема, компонентами КС є також суматори двох чисел по модулю m_i [3-4]. Даний тип суматорів широко використовується, як в ПСЧ, так і в непозиційних системах числення [5]. Суматор чисел $A = (a_1 \| a_2 \| ... \| a_i \| ... \| a_k)$ та $B = (b_1 \| b_2 \| ... \| b_i \| ... \| b_k)$ в СЗК буде складатися з сукупності k $n = [\log_2 (m_i - 1) + 1]$ - розрядних суматорів за модулем m_i.

В цьому аспекті актуальною науково-прикладною задачею є задача побудови (*задача синтезу*) суматорів, які працюють за довільним модулем m_i, що виконані на логічних елементах з двома стійкими станами. Якщо залишки a_i та b_i, відповідно чисел A і B в СЗК, представлені в двійковій ПСЧ, тоді суматор двох залишків a_i та b_i по модулю m_i представляє собою послідовну сукупність з $n=[\log_2(m_i-1)+1]$ *двійкових однорозрядних суматорів* (ДОС), об'єднаних між собою зв'язками, подібно зв'язків позиційних двійкових суматорів.

Позиційні двійкові суматори мають фіксовану величину модуля $M=2^n-1$. Ця обставина не дає можливості їх безпосереднього застосування в якості модулів m_i СЗК. В ПСЧ найбільшого поширення набули два випадки. Для кожного випадку має місце наступні співвідношення модулів. *Перший випадок*. Має місце співвідношення модулів $M=m_i+1$. Для *другого випадку* має місце співвідношення модулів $M=m_i-1$. Для цих обох випадків реалізація операції модульного складання залишків здійснюється відносно просто. У той же час, в загальному випадку, операція модульного додавання двох залишків є досить трудомістким завданням, що обумовлює актуальність завдання синтезу суматорів за довільним модулем.

2 Метод побудови суматорів за модулем

Розглянемо метод побудови суматорів модульного додавання $(a_i+b_i)\bmod m_i$ двох залишків чисел за довільним модулем m_i. Даний метод побудови суматорів за модулем СЗК, заснований на використанні відомих структур суматорів за модулем $M=2^n-1$ в ПСЧ. Завдання побудови суматора модульного додавання $(a_i+b_i)\bmod m_i$ вирішується шляхом запровадження та організації додаткових зв'язків $X_{\downarrow i\uparrow j}$ меж j-м та i-м двійковими розрядами суматора за модулем *M*.

Метод побудови суматорів за довільним модулем m_i СЗК складається з двох етапів. Перший етап складається з рішення задачі побудови структури суматора модульного додавання $(a_i+b_i)\bmod m_i$. Другий етап. На основі отриманої структури суматора і чисельного значення модуля m_i, що представлений двійковим кодом, визначається схема модульного додавання двох залишків a_i та b_i чисел, обумовлену наявністю і використанням додаткових зв'язків $X_{\downarrow i\uparrow j}$ меж j-м та i-м ДОС. Детально розглянемо кожен з етапів методу побудови суматорів за довільним модулем m_i СЗК.

Перший етап: *- створення структури суматора модульного додавання* $(a_i+b_i)\bmod m_i$

Нехай є структура n-розрядного двійкового суматора в ПСЧ по модулю $M=2^n-1$ (рис. 1). Потрібно створити структуру суматора модульного складання $(a_i+b_i)\bmod m_i$, інакше, необхідно створити структуру суматора для реалізації операції додавання двох залишків чисел за довільним модулем m_i СЗК.

Технічно завдання побудови структури суматора за модулем формулюється в такий спосіб. Необхідно забезпечити умови, при яких вихідний суматор в ПСЧ по модулю *M* виконував би операцію складання за модулем m_i. Дана процедура здійснюється шляхом введення додаткових зв'язків виду $X_{\downarrow i\uparrow j}$ в позиційному суматорі за модулем $M=2^n-1$, де вираз $X_{\downarrow i\uparrow j}$ позначає односторонній зв'язок між виходом j-го ДОС та входом i-го ДОС.

Для побудови непозиційних суматора за довільним модулем, в структурі позиційного суматора за модулем *M* необхідно між певною парою ДОС вихідного суматора за модулем *M* сформувати додаткові зв'язку виду $X_{\downarrow i\uparrow j}$.

Додаткові зв'язки $X_{\downarrow i\uparrow j}$ формуються таким чином, щоб створений суматор здійснював операцію $(a_i+b_i)\bmod m_i$.

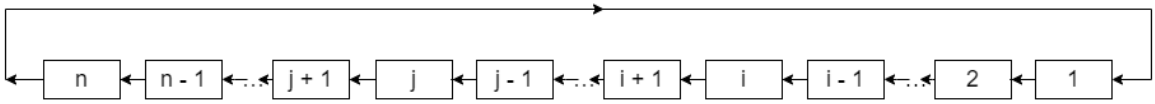


Рис. 1 – Схема розташування та нумерації ДОС

Схема організації в суматорі додаткового зв'язку $X_{\downarrow i \uparrow j}$ між виходом j-го ДОС та входом i-го ДОС відображено на рис. 2. Вочевидь, що завжди виконується умова $j > i$.

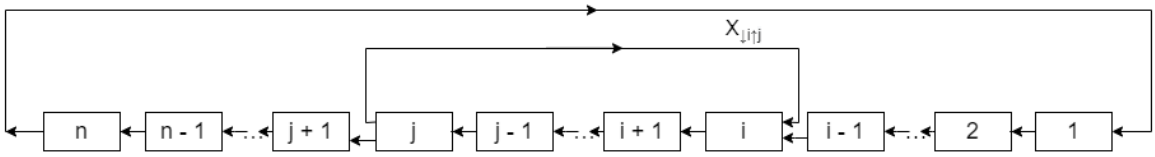


Рис. 2 – Схема двійкового суматора з одним додатковим зв'язком виду $X_{\downarrow i \uparrow j}$

Другий етап: *- визначення схеми додавання двох залишків a_i та b_i чисел*

В роботі [5] проведено дослідження впливу одного додаткового зв'язку $X_{\downarrow i \uparrow j}$, який встановлено між виходом j-го ДОС та входом i-го ДОС в структурі позиційного суматора по модулю $M = 2^n - 1$, на величину G_L вихідного вмісту суматора. Крім цього, показано, що числове значення $L = \{l_i\} = \{l_1, l_2, ..., l_m, ..., l_n\}$ $(i = \overline{1, n})$, що є вмістом G_L суматора, при введенні одного додаткового зв'язку $X_{\downarrow i \uparrow j}$ зменшується на величину $\Delta G_L = 2^{i-j-2} \cdot \sum_{m=j+1}^{n} 2^m \cdot l_m$. Введення і використання такого додаткового зв'язку $X_{\downarrow i \uparrow j}$ переводить обчислення чисел з двійкової системи числення (СЧ), в якій працює акумулятор по модулю M, в поліадичну СЧ з основами $\tau_1, \tau_2, ..., \tau_k$ с модулем $M = \tau_1 \cdot \tau_2 \cdot ... \cdot \tau_k - 1$. У цьому випадку число, наприклад $L = \{l_i\}$, $(i = \overline{1, k})$ представиться у вигляді (1):

$$G_L = \sum_{m=1}^{k} l_m \cdot \prod_{i=1}^{m-1} \tau_i = l_1 \cdot \tau_2 \cdot \tau_3 \cdot ... \cdot \tau_k + l_2 \cdot \tau_3 \cdot \tau_4 \cdot ... \cdot \tau_k + ... + l_{k-2} \cdot \tau_{k-1} \cdot \tau_k + l_{k-1} \cdot \tau_k + l_k . \quad (1)$$

Очевидно, що за відсутності в суматорі додаткових зв'язків $X_{\downarrow i \uparrow j}$ величина G_L вмісту суматора буде дорівнювати

$$G_L = \sum_{m=1}^{n} l_m \cdot q_m , \quad (2)$$

де величина l_m в m–му розряді вмісту суматора може приймати два значення $l_m = 0$ або $l_m = 1$, а значення q_m – є вага m–го розряду вмісту суматора, який визначається місцем розташування двійкового розряду суматора, що представлений на рис. 1. Якщо є додатковий зв'язок $X_{\downarrow i \uparrow j}$, що поєднує в структурі КС ДОС з номерами від i до j в єдиний (*узагальнений*) розряд суматора за модулем

$$\tau_{ij} = 2^{j-(i-1)} - 1 = 2^{j-i+1} - 1, \quad (3)$$

то вага q_m кожного розряду суматора з номерами від $(i-1)$-го до першого (молодшого розряду суматора) буде дорівнювати значенню $q_m = 2^{m-1}$ $(m = \overline{1, i-1})$ (див. рис. 3).

Виходячи зі структури суматора по модулю (рис. 4) вага розрядів суматора з номерами від $(j+1)$-го до старшого розряду суматора, буде визначатися виразом

$$q_m = 2^{i-1} \cdot \tau_{ij} \cdot 2^{m-j-1} . \quad (4)$$

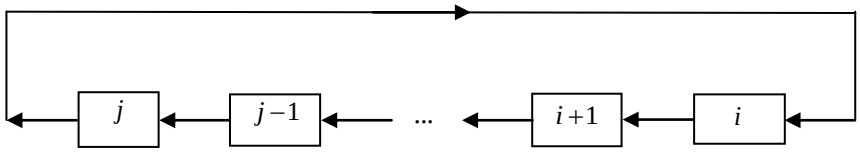


Рис. 3 – Схема узагальненого $(j-i+1)$-го розряду суматора за модулем τ_{ij}

З огляду на співвідношення (3), вираз (4) можна представити в наступному вигляді:

$$q_m = 2^{m+i-j-2} \cdot \tau_{ij} = 2^{m+i-j-2} \cdot (2^{j-i+1} - 1) = 2^{m-1} - 2^{m+i-j-2}. \quad (5)$$

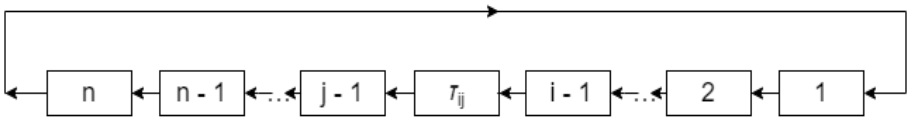


Рис. 4 – Еквівалентна схема суматора по модулю з додатковим зв'язком $X_{\downarrow i \uparrow j}$

В роботі [5] показано, що при введенні додаткової зв'язку виду $X_{\downarrow i \uparrow j}$, значення величини модулю $M = 2^n - 1$ позиційного суматора зменшується на величину

$$\Delta M = 2^{i-j-2} \cdot \sum_{m=j+1}^{n} S_m \cdot 2^m, \quad (6)$$

де S_m – значення m-го розряду числа, що міститься в суматорі. При цьому, природно, діапазон чисел, які представлені по модулю M, зменшується на величину ΔM. Дана обставина дає можливість, за рахунок введення в суматор за модулем M, додаткових зв'язків (або одного додаткового зв'язку), зменшити величину позиційного M модулю до необхідного значення модуля m_i СЗК.

Схема складання двох залишків визначається наступними правилами організації додаткових зв'язків $X_{\downarrow i \uparrow j}$.

1. Додатковий зв'язок суматора за модулем починається з виходу *n*-го (*старшого*) ДОС ($j = n$).

2. Додатковий зв'язок надходить на вхід ДОС, для якого $S_i = 0$.

3. В *n*-розрядному двійковому позиційному суматорі, що працює за модулем $M = 2^n - 1$, додатковий зв'язок $X_{\downarrow i \uparrow j}$ може мати місце лише в i-х двійкових розрядах S_i ($i = \overline{2, n}$), які відповідають нульовим значенням двійкової кодової комбінації модулю m_i.

4. У двійкових розрядах S_i позиційного суматора, котрі відповідають одиничним значенням, додаткові зв'язки $X_{\downarrow i \uparrow j}$ повинні бути відсутніми.

5. Визначаються двійкові розряди S_i позиційного суматора, для яких виконується умова $S_i = 0$. Процес визначення умови $S_i = 0$ проводиться виходячи з представлення модуля числа m_i в двійковому коді.

Розглянемо приклад побудови суматора, наприклад, по модулю $m_i = 11$ СЗК.

Приклад 1.

1. Згідно з величиною $m_i = 11$ модуля СЗК, визначимо кількість *n* ДОС. Для модулю $m_i = 11$ маємо, що $n = [\log_2(11-1)] + 1 = 4$. При цьому, вихідна (*без додаткових зв'язків*) структура позиційного суматора за модулем $M = 2^n - 1 = 15$ має вигляд, який представлений на рис. 5.

2. Для синтезу суматора по модулю $m_i = 11$ СЗК, попередньо визначимо значення двійкових розрядів S_i суматора, в запису значення модуля $m_i = 11$ котрих, містяться нулі (*тобто, коли* $S_i = 0$). Таким розрядом є третій розряд ($S_3 = 0$),так як в двійковому коді модуль $m_i = 11$ має вигляд: - 1011.

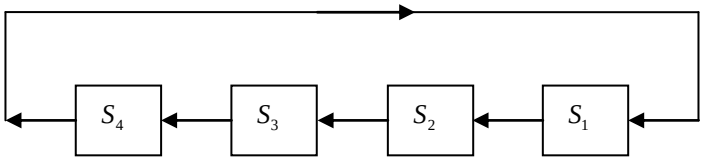


Рис. 5 – Вихідна структура суматора за модулем $M = 2^n - 1$

3. Так як $S_3 = 0$, то додатковий зв'язок в суматорі має вигляд $X_{\downarrow 3 \uparrow 4}$. Загальна кількість зв'язків дорівнює двом $X_{\downarrow 1 \uparrow 4}$ та $X_{\downarrow 3 \uparrow 4}$. Структура суматора за модулем $m_i = 11$ представлена на рис. 6.

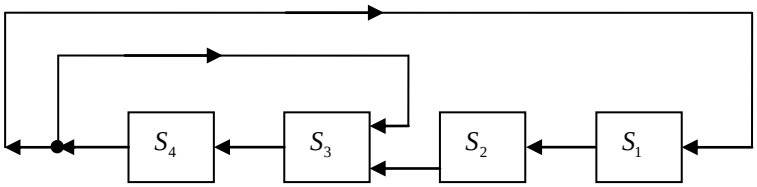


Рис. 6 – Структура суматора за модулем $m_i = 11$

Для перевірки ідентичності отриманої структури суматора за модулем $m_i = 11$ (рис. 6), попередньо, розглянемо частину (див. рис. 7) структури суматора виду

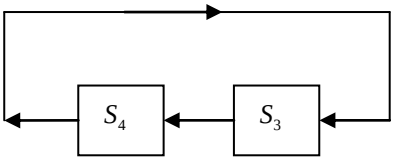


Рис. 7 – Частина структури суматора за модулем 11

Для цієї частини (рис. 7) структури суматора значення модуля M_1 визначиться як $M_1 = \tau_4 \cdot \tau_3 - 1$. Значення модуля СЗК суматора (рис. 6) визначається наступним чином (див. рис. 8): $m_i = M_1 \cdot \tau_2 \cdot \tau_1 - 1 = (\tau_4 \cdot \tau_3 - 1) \cdot \tau_2 \cdot \tau_1 - 1 = = (2 \cdot 2 - 1) \cdot 2 \cdot 2 - 1 = 11$.

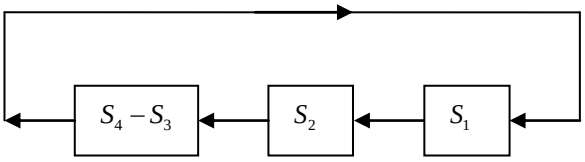


Рис. 8 – Фрагмент структури суматора для визначення значення модуля СЗК

Таким чином, синтез суматора за модулем $m_i = 11$ проведений вірно.

3 Приклади виконання операції додавання, залишків чисел за модулем

Існуючий [5] метод додавання $(a_i + b_i) \bmod m_i$ залишків a_i та b_i чисел за модулем m_i, базується на використанні двійкових суматорів, складається з сукупності таких дій, операцій і процедур:

1. Синтез суматора по заданому модулю m_i СЗК;
2. Визначається результат $S_n S_{n-1} \ldots S_2 S_1$ порозрядного додавання по модулю два залишків a_i та b_i чисел за модулем m_i, що представлені двійковим кодом;
3. Вміст двійкових розрядів, отриманої модульної суми $S_n S_{n-1} \ldots S_2 S_1$, заноситься до відповідних ДОС структури суматора по модулю m_i СЗК;
4. На підставі синтезованої структури суматора за модулем СЗК, реалізується алгоритм додавання двох залишків a_i та b_i чисел.

Однак в деяких випадках існуючий метод має обмежене застосування при додаванні двох залишків чисел за модулем m_i СЗК. Зокрема, в разі рівності двох залишків, тобто коли $a_i = b_i$, результат додавання $(a_i + b_i) \bmod m_i$ не у всіх випадках буде правильним. Це обумовлено тим, що отримана модульна сума $S_n S_{n-1} ... S_2 S_1$, яка використовується при визначенні результату модульного додавання $(a_i + b_i) \bmod m_i$ залишків a_i та b_i чисел, не враховує співвідношення між величинами значень модулів m_i, M і значенням $a_i + b_i$ позиційного додавання. В даному випадку для отримання правильного результату операції $(a_i + b_i) \bmod m_i$ необхідно враховувати величини значень m_i, M та $a_i + b_i$.

Очевидно, що при розробці методу додавання $(a_i + b_i) \bmod m_i$ залишків a_i та b_i необхідно враховувати варіанти співвідношення між величинами значень модулів m_i, M і значенням $a_i + b_i$ результату позиційного додавання залишків чисел. У таблиці 1 представлені можливі співвідношення між величинами модулів m_i, M та значенням $a_i + b_i$ позиційного додавання залишків чисел.

Таблиця 1 - Співвідношення значень величин

№ з.п.	***Співвідношення значень*** m_i, $a_i + b_i$ ***та*** $M = 2^n - 1 (n = 4, m_i = 11, M = 15)$		***Режим виконання операції додавання***
1	$a_i + b_i < m_i$	$a_i + b_i < 11$	Перший режим
2	$a_i + b_i = m_i$	$a_i + b_i = 11$	
3	$m_i < a_i + b_i < 2^n - 1$	$11 < a_i + b_i < 15$	Другий режим
4	$a_i + b_i = 2^n - 1$	$a_i + b_i = 15$	
5	$2^n - 1 < a_i + b_i$	$15 < a_i + b_i$	

Розглянемо запропонований метод на *прикладах* виконання операції $(a_i + b_i) \bmod m_i$ додавання залишків a_i та b_i чисел за модулем m_i. Узагальнюючи все вищевикладене для заданого значення модуля m_i, метод реалізації операції додавання $(a_i + b_i) \bmod m_i$ двох залишків a_i та b_i чисел буде складатися з сукупності таких дій, операцій і процедур:

1. Спочатку, для заданого значення модулю m_i, здійснюється побудова суматора за модулем m_i СЗК;

2. Формується результат позиційного підсумовування $a_i + b_i$ двох залишків a_i та b_i;

3. Проводиться порівняння значень $a_i + b_i$ та m_i;

4. Якщо $a_i + b_i \le m_i$, тоді значення $a_i + b_i$ – результат операції $(a_i + b_i) \bmod m_i$;

5. Якщо $a_i + b_i > m_i$, то для всіх можливих режимів виконання операції додавання (*див. табл. 1*), отриманий результат $a_i + b_i$ позиційного підсумовування двох залишків a_i та b_i, порозрядно заноситься до відповідних *n* ДОС, послідовна сукупність яких, становить структуру суматора по модулю m_i СЗК;

6. На підставі синтезованої, для заданого значення модуля m_i СЗК, структури суматора двох залишків a_i і b_i чисел (*див. п. 1*), реалізується алгоритм додавання двох залишків a_i та b_i чисел за модулем.

Розглянемо конкретні приклади виконання операції додавання $(a_i + b_i) \bmod m_i$ для двох довільних залишків a_i та b_i чисел розглянутим вище методом.

Приклади виконання операції додавання розглянемо в двох режимах (*див. табл. 1*).

Перший режим виконання операції додавання: - виконується умова $(a + b) \le m_i$ (*див. табл. 1*).

Приклад 2. Нехай залишки дорівнюють $a_i = 5$ та $b_i = 4$. Суматор реалізує операцію позиційного додавання залишків $a_i = 0101$ та $b_i = 0100$ у вигляді:

$$\begin{array}{r} a_i = 0101 \\ + \\ b_i = 0100 \\ \hline a_i + b_i = 1001 \end{array}.$$

Значення суми $a_i + b_i = 1001$ визначає результат операції.

Перевірка: $(0101 + 0100) = 1001 (\mathrm{mod} 11)$.

2-й режим виконання операції додавання. Виконується умова $(a+b) > m_i$ (*табл. 1*).

Приклад 3. Нехай $a_i = 5$ і $b_i = 6$. Суматор реалізує операцію позиційного додавання залишків $a_i = 0101$ та $b_i = 0110$ у вигляді:

$$\begin{array}{r} a_i = 0101 \\ + \\ b_i = 0110 \\ \hline a_i + b_i = 1011 \end{array}.$$

Значення позиційної суми $a_i + b_i = 1011$ залишків $a_i = 0101$ та $b_i = 0110$ порозрядно надходить на відповідні входи ДОС $5_4 - 5_1$. Таким чином, ДОС $5_4 - 5_1$ суматора містить значення 1011. Операція модульного додавання реалізується за схемою модульного додавання за модулем $m_i = 11$ (див. рис. 6). Алгоритм реалізації модульної операції представлений в таблиці 2 та на рис. 9. Одиниця двійкового розряду надходить на вхід ДОС 5_3 і 5_1.

Таблиця 2 - Алгоритм виконання результату операції

ДОС $5_4 - 5_1$	Вміст ДОС $5_4 - 5_1$	Наявність одиниці на входах ДОС $5_4 - 5_1$	Результат операції модульного додавання
5_1	1	+1	0
5_2	1	–	0
5_3	0	+1	0
5_4	1	–	0

На рис. 9 представлена схема додавання залишків $a_i = 0101$ та $b_i = 0110$ за модулем $m_i = 11$.

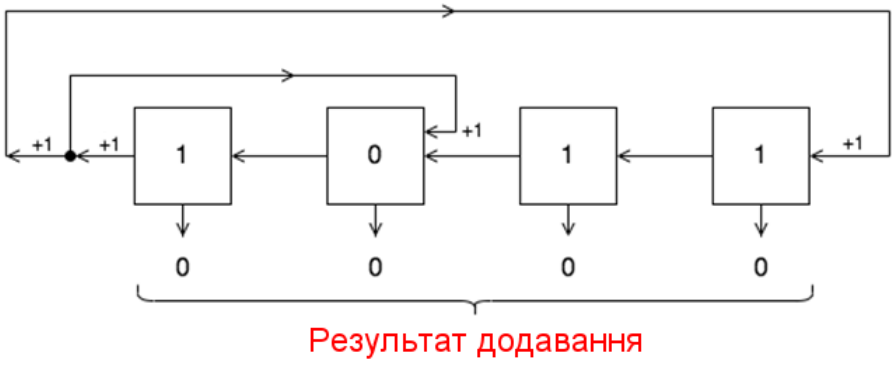


Рис. 9 – Схема додавання залишків $a_i = 0101$ та $b_i = 0110$ за модулем $m_i = 11$

Перевірка: $(0101 + 0110) = 0000 (\mathrm{mod} 11)$.

Приклад 4. Нехай $a_i = 5$ і $b_i = 7$. Суматор реалізує операцію позиційного додавання залишків $a_i = 0101$ та $b_i = 0111$ у вигляді:

$$\begin{array}{r} a_i = 0101 \\ + \\ b_i = 0111 \\ \hline a_i + b_i = 1100 \end{array}.$$

Значення позиційної суми $a_i + b_i = 1100$ залишків $a_i = 0101$ і $b_i = 0111$ порозрядно надходить на відповідні входи ДОС $5_4 - 5_1$. Таким чином, ДОС $5_4 - 5_1$ суматора містить значення 1100. Операція модульного додавання реалізується за схемою модульного додавання за модулем $m_i = 11$ (*див. рис. 6*).

Алгоритм реалізації модульної операції представлений в таблиці 3 і на рис. 10. Одиниця двійкового розряду надходить на вхід ДОС 5_3 і 5_1.

Таблиця 3 - Алгоритм виконання результату операції

ДОС $5_4 - 5_1$	Вміст ДОС $5_4 - 5_1$	Наявність одиниці на входах ДОС $5_4 - 5_1$	Результат операції модульного додавання
5_1	0	+1	1
5_2	0	–	0
5_3	1	+1	0
5_4	1	–	0

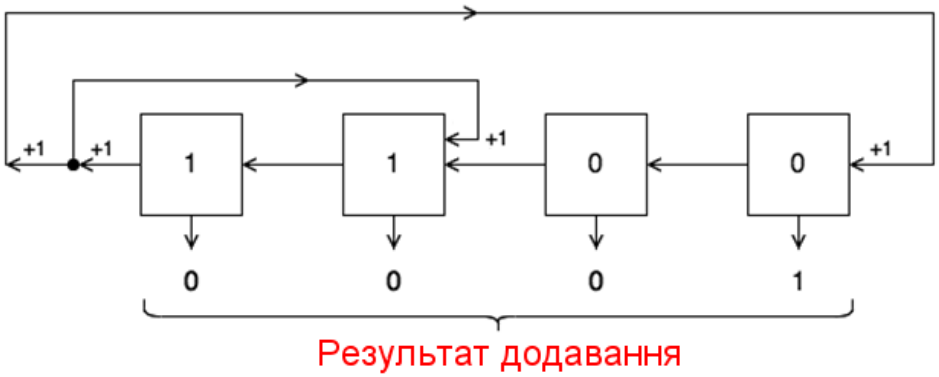


Рис. 10 – Схема додавання залишків $a_i = 0101$ і $b_i = 0111$ за модулем $m_i = 11$

Перевірка: $(0101 + 0111) = 0001 (\mathrm{mod} 11)$.

Приклад 5. Нехай $a_i = 5$ і $b_i = 9$. Суматор реалізує операцію позиційного додавання залишків $a_i = 0101$ и $b_i = 1001$ у вигляді:

$$\begin{array}{r} a_i = 0101 \\ + \quad b_i = 1001 \\ \hline a_i + b_i = 1110 \end{array}.$$

Значення позиційної суми $a_i + b_i = 1110$ залишків $a_i = 0101$ и $b_i = 1001$ порозрядно надходить до відповідних входів ДОС $5_4 - 5_1$. Таким чином, ДОС $5_4 - 5_1$ суматора, містить значення 1110. Операція модульного додавання реалізується за схемою модульного додавання за модулем $m_i = 11$ (рис. 6).

Алгоритм реалізації модульної операції представлений в таблиці 4 і схемі на рис. 11. Одиниця двійкового розряду надходить на вхід ДОС 5_3 та 5_1.

Перевірка: $(0101 + 1001) = 0011 (\mathrm{mod} 11)$.

Таблиця 4 - Алгоритм виконання результату операції

ДОС $5_4 - 5_1$	Вміст ДОС $5_4 - 5_1$	Наявність одиниці на входах ДОС $5_4 - 5_1$	Результат операції модульного додавання
5_1	0	+1	1
5_2	1	–	1
5_3	1	+1	0
5_4	1	–	0

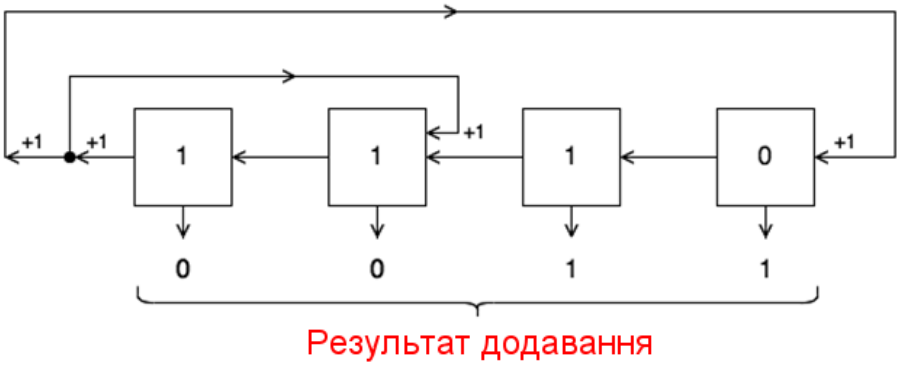


Рис. 11 – Схема додавання залишків $a_i = 0101$ і $b_i = 1001$ за модулем $m_i = 11$

Приклад 6. Нехай $a_i = 6$ і $b_i = 10$. Суматор реалізує операцію позиційного додавання залишків $a_i = 0110$ та $b_i = 1010$ у вигляді:

$$\begin{array}{r} a_i = 0110 \\ + \\ b_i = 1010 \\ \hline a_i + b_i = 10000 \end{array}.$$

Значення чотирьох 0000 молодших двійкових розрядів позиційної суми $a_i + b_i = 10000$ залишків $a_i = 0110$ і $b_i = 1010$, порозрядно надходить до відповідних входів ДОС $5_4 - 5_1$. Таким чином, ДОС $5_4 - 5_1$ суматора, містить значення 0000. Операція модульного додавання реалізується за схемою модульного додавання (рис. 6) за модулем $m_i = 11$.

Алгоритм реалізації модульної операції представлений в таблиці 5, а схема додавання на рис. 12. Одиниця двійкового розряду надходить на вхід ДОС 5_3 та 5_1.

Таблиця 5 - Алгоритм виконання результату операції

ДОС $5_4 - 5_1$	Вміст ДОС $5_4 - 5_1$	Наявність одиниці на входах ДОС $5_4 - 5_1$	Результат операції модульного додавання
5_1	0	+1	1
5_2	0	–	0
5_3	0	+1	1
5_4	0	–	0

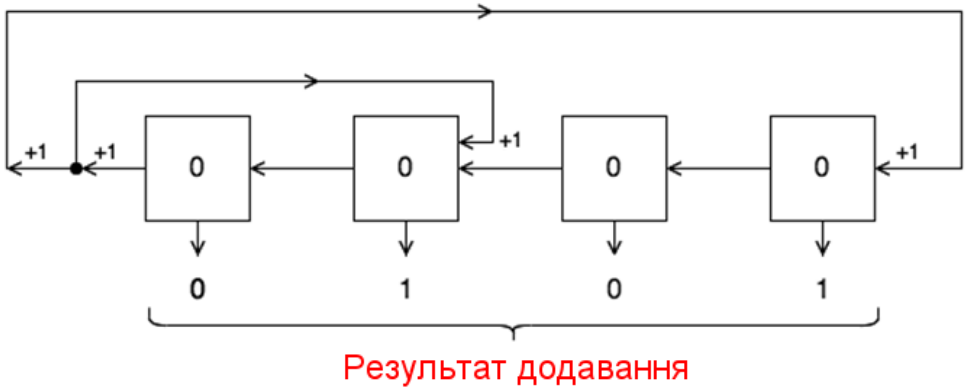


Рис. 12 – Схема додавання залишків

Перевірка: $(0110 + 1010) = 0101 (\mathrm{mod} 11)$.

Приклад 7. Пусть $a_i = b_i = 10$.

Суматор реалізує операцію позиційного додавання залишків $a_i = 1010$ та $b_i = 1010$ у вигляді:

$$\begin{array}{r} a_i = 1010 \\ + \\ b_i = 1010 \\ \hline a_i + b_i = 10100 \end{array}.$$

Значення 4-х 0100 молодших двійкових розрядів позиційної суми $a_i + b_i = 10100$ залишків $a_i = 1010$ і $b_i = 1010$ порозрядно надходить до відповідних входів ДОС $5_4 - 5_1$. Таким чином, ДОС $5_4 - 5_1$ суматора, містить значення Операція модульного додавання реалізується за схемою модульного додавання за модулем $m_i = 11$ (рис. 6).

Алгоритм реалізації модульної операції представлений в таблиці 6, а схема на рис. 13. Одиниця двійкового розряду надходить на вхід ДОС 5_3 та 5_1.

Таблиця 6 - Алгоритм виконання результату операції (*для* $a_i = b_i = 10$)

ДОС $5_4 - 5_1$	Вміст ДОС $5_4 - 5_1$	Наявність одиниці на входах ДОС $5_4 - 5_1$	Результат операції модульного додавання
5_1	0	+1	1
5_2	0	–	0
5_3	1	+1	0
5_4	0	–	1

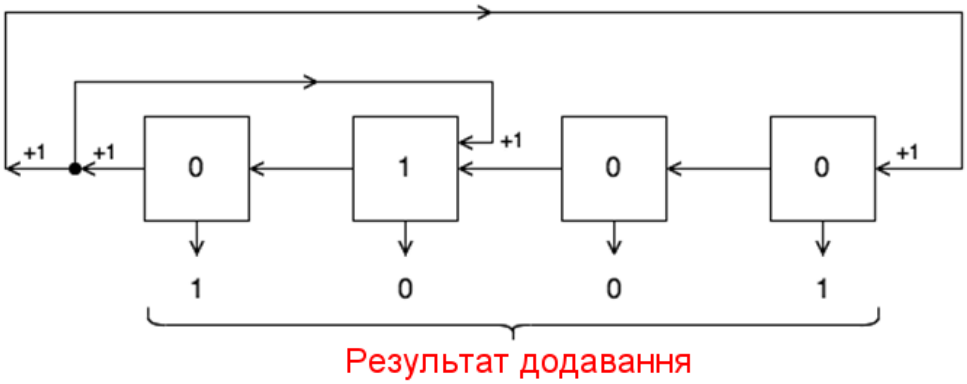


Рис. 13 – Схема додавання залишків $a_i = 1010$ та $b_i = 1010$ за модулем $m_i = 11$

Перевірка: $(1010 + 1010) = 1001 (\mathrm{mod} 11)$.

4 Висновки

У даній роботі представлений метод додавання залишків a_i та b_i двох чисел за довільним модулем m_i в СЗК. Для реалізації даного методу додавання за модулем, була модифікована структура модульного суматора за довільним модулем СЗК. В основу цих змін була покладена структура позиційного суматора за модулем $M = 2^n - 1$, яка складається із сукупності послідовних двійкових однорозрядних суматорів з 1-м зворотним зв'язком.

Для побудови зазначеного типу суматора, необхідно використовувати додаткові зв'язки $X_{\downarrow i \uparrow j}$ між j-м та i-м двійковими розрядами суматору за модулем M.

Розроблений метод виконання операції додавання $(a_i + b_i) \bmod m_i$ двох залишків a_i і b_i чисел за модулем m_i, ґрунтується на структурі двійкового суматору за довільним модулем m_i СЗК та схемою додавання залишків a_i та b_i двох чисел.

Розглянуті приклади реалізації методу модульного складання для залишків a_i і b_i для різних значень модуля m_i СЗК, підтверджують практичну реалізованість запропонованого методу.

Посилання

[1] Bayoumi M.A., Jullien G.A., Miller W.C. A VLSI Implementation of Residue. Adders IEEE Trans. on Circuits and Systems. 1987. V. 34, № 3. pp. 284-288.

[2] P.V. Ananda Mohan. Residue Number Systems: Theory and Applications. Birkhäuser Basel: Springer International Publishing Switzerland, 2016. 351 P.

[3] Krasnobayev V. A. and Koshman S. A. Method for implementing the arithmetic operation of addition in residue number system based on the use of the principle of circular shift // Cybernetics and Systems Analysis. – July, 2019. – Vol. 55, Is. 4, pp. 692-698.

[4] Krasnobayev V. A., Yanko A. S., Koshman S. A. A Method for arithmetic comparison of data represented in a residue number of system // Cybernetics and Systems Analysis. – January 2016. – Vol. 52, Is. 1, pp. 145-150.

[5] V. A. Krasnobayev, A. A. Kuznetsov, S. A. Koshman, and K. O. Kuznetsova "A method for implementing the operation of modulo addition of the residues of two numbers in the residue number system", Cybernetics and Systems Analysis, Vol. 56, No. 6, November, 2020, 1029-1038. https://doi.org/10.1007/s10559-020-00323-9.

[6] Azadeh Safari, James Nugent, Yinan Kong. Novel implementation of full adder based scaling in Residue Number Systems. IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS). 4-7 Aug. 2013. pp. 657–660.

[7] Shugang Wei. Fast signed-digit arithmetic circuits for residue number systems. IEEE International Conference on Electronics, Circuits, and Systems (ICECS). 6-9 Dec. 2015. pp. 344 – 347.

Reviewer: Vyacheslav Kalashnikov, Doctor of Sciences (Physics and Mathematics), Full Prof., Department of Systems and Industrial Engineering, Campus Monterrey, Monterrey, Mexico.
E-mail: kalash@itesm.mx

Received on November 2020.

Authors:
Victor Krasnobaev, Doctor of Sciences (Engineering), Full Prof., Academician of the Academy of Applied Radioelectronics Sciences, Kharkiv National University named after V.N. Karazin, Kharkiv, Ukraine.
E-mail: v.a.krasnobaev@gmail.com

Kateryna Kuznetsova, student of the Department of Security of Information Systems and Technologies, Kharkiv National University named after V.N. Karazin, Kharkiv, Ukraine.
E-mail: kate7smith12@gmail.com

Mykhaylo Bagmut, graduate student of the Department of Security of Information Systems and Technologies, Kharkiv National University named after V.N. Karazin, Kharkiv, Ukraine.
E-mail: Mikhail56@ukr.net

Method for performing the operation of adding the remainder of numbers modulo.

Abstract. One of the components of a computer system (CS) in a positional binary number system (PNS) is an adder of two numbers. In particular, adders modulo m_i of two numbers are also components of the CS. This type of modulo adders is widely used both in the PNS and in the non-positional number system in the residual classes (RNS). An important and urgent scientific and applied problem is the problem of constructing the adders, which operate by modulus m_i, that is an arbitrary RNS modulo. If the remainders a_i and b_i of both numbers A and B in RNS are represented in a binary PNS, then the adder of two residuals a_i and b_i by modulus m_i is a sequential set of n binary one-bit adders (BOBA). The purpose of the article is to develop a method for performing the operation of modular addition $(a_i + b_i) \bmod m_i$ of two remainders of numbers by an arbitrary modulo m_i based on the use of a positional binary adder modulo $M = 2^n - 1$. The proposed method is based on the use of the well-known structure of positional binary adders modulo $M = 2^n - 1$. Technically, the problem of creating the structure of the modular adder is formulated as follows. It is necessary to provide conditions under which the initial adder in PNS modulo M performs the addition operation modulo m_i. This procedure is carried out by introducing additional connections as $X_{\downarrow i \uparrow j}$ in the positional adder modulo $M = 2^n - 1$, where the expression $X_{\downarrow i \uparrow j}$ denotes one-way connection between the output of the j-th BOBA and the input of the i-th BOBA.

Keywords*:* adder of two numbers; adder for any module; operation of modular addition of two residues; positional binary number system (PNS); non-positional number system in residual classes (RNS).

Рецензент: Вячеслав Калашников, д.ф.-м.н., проф., Технологический университет Монтеррея, Монтеррей, Мексика.
E-mail: kalash@itesm.mx

Поступила: Ноябрь 2020.

Авторы:
Виктор Краснобаев, д.т.н., проф., академик Академии наук прикладной радиоэлектроники, Харьковский национальный университет имени В. Н. Каразина, Харьков, Украина.
E-mail: v.a.krasnobaev@gmail.com

Екатерина Кузнецова, студентка факультета компьютерных наук, Харьковский национальный университет имени В. Н. Каразина, Харьков, Украина.
E-mail: kate7smith12@gmail.com

Михаил Багмут, аспирант каф. Безопасности информационных систем и технологий, Харьковский национальный университет имени В. Н. Каразина, Харьков, Украина.
E-mail: Mikhail56@ukr.net

Метод выполнения операции сложения остатков чисел по модулю.

Аннотация. Сумматор двух чисел является одним из компонентов компьютерных систем (КС) в позиционной двоичной системе счисления (ПСС). В частности, компонентами КС являются, также, сумматоры по модулю m_i двух чисел. Данный тип сумматоров по модулю широко используются как в ПСС, так и в непозиционной системе счисления в остаточных классах (СОК). Важной и актуальной научно-прикладной задачей является задача построения сумматоров, работающих по произвольному модулю m_i СОК. Если остатки a_i и b_i чисел A и B в СОК, представлены в двоичной ПСС, тогда сумматор двух остатков a_i и b_i по модулю m_i есть последовательная совокупность из n двоичных одноразрядных сумматоров (ДОС). Целью статьи является, разработка метода выполнения операции модульного сложения $(a_i + b_i) \bmod m_i$ двух остатков чисел, по произвольному модулю, на основе использования позиционного двоичного сумматора по модулю $M = 2^n - 1$. Предложенный в работе метод выполнения операции модульного сложения, основан на использовании известной структуры позиционных двоичных сумматоров по модулю $M = 2^n - 1$. Технически, задача построения структуры сумматора, состоит в необходимости обеспечить условия при которых, исходный сумматор в ПСС по модулю М, выполнял бы операцию сложения по модулю m_i. Данная процедура осуществляется путем введения дополнительных связей вида $X_{\downarrow i \uparrow j}$ в позиционном сумматоре по модулю $M = 2^n - 1$, где выражение $X_{\downarrow i \uparrow j}$ обозначает одностороннюю связь между выходом j-го ДОС и входом i-го ДОС. Приведены примеры реализации метода выполнения операции модульного сложения для различных значений остатков a_i и b_i. Анализ рассмотренных примеров показал практическую реализуемость предложенного метода. Он может быть использован, как в ПСС, так и в СОК.

Ключевые слова: сумматор двух чисел; сумматор по произвольному модулю; операция модульного сложения двух остатков; позиционная двоичная система счисления (ПСС); непозиционная система счисления в остаточных классах (СОК).

UDC 004.056.55 DOI: 10.26565/2519-2310-2021-1-02

RESEARCH OF IMPLEMENTATION OF CANDIDATES OF THE SECOND ROUND OF NIST PQC COMPETITION FOCUSED ON FPGA XILINX FAMILY

Marina Yesina, Bogdan Shahov

V. N. Karazin Kharkiv National University, Svobody Sq., 4, Kharkiv, 61022, Ukraine
m.v.yesina@karazin.ua, bogdanshahov2000@gmail.com

Reviewer: Serhii Toliupa, Doctor of Sciences (Engineering), Full Prof., Taras Shevchenko National University of Kiev, 81 Lomonosova St., Kyiv, 03189, Ukraine.
tolupa@i.ua

Received on December 2020

Abstract: *Today, the question of the stability of modern existing cryptographic mechanisms to quantum algorithms of cryptanalysis in particular and quantum computers in general is quite acute. This issue is actively discussed at the international level. Therefore, in order to solve it, NIST USA has decided to organize and is currently holding a competition for candidates for post-quantum cryptographic algorithms NIST PQC. The result of the competition should be the acceptance for standardization of cryptographic algorithms of different types - asymmetric encryption, key encapsulation and electronic signature (at least one algorithm of each type). At the beginning of the competition for the standardization process, 82 algorithms were presented. Based on the minimum eligibility criteria defined by NIST, 69 algorithms were considered for the 1st round. Given several parameters – security, cost, performance, implementation characteristics, etc., 43 and 11 algorithms were excluded at the end of the 1st and 2nd rounds, respectively, and the other 15 algorithms were saved for the 3rd round. The algorithms left in the 2nd round can be divided into 5 different categories depending on the mathematical basis on which they are based: based on the isogeny of elliptic curves, based on algebraic lattices, based on mathematical code, based on multivariate transformations and based on hash functions. Security is the main evaluation criterion that determines competition in the NIST competition, and it is clear that candidates' software implementations are mainly focused on it. However, it is extremely important that the algorithm has an effective hardware implementation. And timely detection of hardware inefficiencies will help focus the cryptographic community's efforts on more promising candidates, potentially saving a lot of time that can be spent on cryptanalysis. This paper discusses and compares the FPGAs of Xilinx family. Data on the implementation of the candidates of the 2nd round in the process of standardization of post-quantum cryptography NIST, which are focused on the FPGA of the Xilinx family, are presented and compared.*

Keywords: *electronic signature; post-quantum cryptography; NIST PQC competition; FPGA, Xilinx.*

1 Introduction

Today, the problem of the stability of existing cryptographic defence mechanisms to quantum cryptanalysis algorithms and quantum computers in general is quite acute. This is an issue under discussion at the international level. And for its decision, NIST USA decided to organize and holds today a competition for post-quantum cryptographic algorithms NIST PQC. The result of the competition should be the adoption for standardization of algorithms such as asymmetric encryption, key encapsulation and electronic signature (*at least one algorithm from each type*).

At the time of the start of the competition for the standardization process, 82 algorithms were presented. Based on the minimum eligibility criteria defined by NIST, 69 algorithms were considered for the first round. Considering several parameters – safety, cost, productivity, implementation characteristics, etc., 43 and 11 algorithms were excluded at the end of the first and second rounds, respectively, and the remaining 15 algorithms were saved for the third round [1].

The algorithms that remained in the second round can be categorized into five different cryptographic hard problems: isogeny-based *(1 algorithm*), lattice-based (*12 algorithms*), code-based (*7 algorithms*), multivariate polynomial cryptography (*4 algorithms*) and hash-based digital signatures (*2 algorithms*) [1-2]. Security is the main evaluation criterion that determines competition in the NIST competition, and it is clear that the implementation of the candidate software is mainly focused on it. However, it is essential that the algorithm has an efficient hardware implementation. And timely identification of hardware inefficiency will help concentrate the efforts of the crypto-

graphic community on more promising candidates, potentially saving a large amount of time that can be spent on cryptanalysis [3].

2 Hardware and software

Cryptographic algorithms are routinely implemented using both software and hardware. By software, we mean implementations that can be executed using processors. These processors may vary from low-cost low-power embedded processors, such as ARM Cortex-M4, to high-performance general-purpose microprocessors, such as Intel Core i7, with Haswell microarchitecture, supporting Advanced Vector Extensions 2 (AVX2) and the AES New Instructions (AES-NI). The common feature is that all of these processors are typically programmed using high-level programming languages, such as C. Code written in these languages is portable among different processor types. Software implementations can be further optimized by using assembly language programming, involving instructions specific to a given processor (or more accurately to its Instruction Set Architecture (ISA)). Assembly language programs are not easily portable among processors based on different ISAs.

By hardware, we mean implementations that can be executed using Field Programmable Gate Arrays (FPGAs), Application-Specific Integrated Circuits (ASICs), Programmable Logic (PL) of System on Chip FPGAs (SoC FPGAs), Application-Specific Standard Products (ASSPs), etc. The common feature is that most of these implementations are developed using hardware description languages (HDLs), such as VHDL and Verilog. These languages differ substantially from high-level programming languages by introducing the concepts of an entity, connectivity, concurrency, and timing. HDL source code is transformed by a synthesis tool to a netlist composed of basic logic components and connections among these components. Because of its generic nature, HDL code can be easily ported among different technologies, such as FPGAs and ASICs. ASIC implementations are faster, use less power, and require less physical area. FPGA implementations have the advantage of less expensive development tools, much shorter design cycle, and reconfigurability, understood as an ability to change the function of all internal building blocks and connections among them, even after a given integrated circuit has been deployed in actual products.

3 FPGA

Although software implementations are likely to be dominant during the first phase of deploying PQC standards in real applications, hardware implementations will inevitably follow. They are likely to start from hardware accelerators for constrained environments, such as smart cards and Internet of Things devices. Low-cost low-power processors used in such applications may not be able to keep up with the increased demands for computational power and energy usage. Thus, these processors may need to be extended with hardware accelerators. In the medium term, high-performance security processors enhanced with new PQC standards will emerge. These processors will be optimized to process in hardware all the algorithms associated with secure communication (*such as those used in the post-quantum versions of TLS, IPSec, IKE, and WTLS/WAP protocols*) and secure storage. Finally, in the longer-term, support for new instructions, enabling the efficient and side-channel resistant implementations of PQC standards, is likely to be added to the most popular processor ISAs. Co-processors for such instructions are, effectively, hardware implementations of PQC. Taking into account that the new PQC standards are likely to remain in use for decades, all of the mentioned above use cases should be given considerable weight. In particular, the performance of a given algorithm in hardware may affect its long-term performance in software, on processors equipped with new specialized instructions. Even if Round 2 hardware implementations are not a final word in terms of the algorithm performance, they provide the first glimpse into each candidate's suitability for hardware acceleration. They establish [1] an open source-code base on which more optimized implementation and implementations protected against side-channel and fault attacks can be built in Round 3 and beyond.

Assuming the use of the same technology, hardware implementations outperform software implementations using at least one, and typically multiple metrics, such as speed, power consump-

tion, energy usage, and security against physical attacks. They also allow much higher flexibility in trading one subset of these metrics for another. From the point of view of benchmarking and ranking of candidates, such flexibility may become a curse, especially taking into account that no two metrics are likely to have a simple linear dependence on each other. A practical solution to this problem [4] is to focus during the evaluation process on two major types of implementations: high-speed and lightweight.

In high-speed implementations, the primary target is speed. For PQC schemes, this target amounts to minimizing the execution times of major operations involving the public and private key, respectively. For Key Encapsulation Mechanisms (KEMs), these operations are encapsulation and decapsulation; for digital signature schemes, signature verification and generation; for public-key encryption (PKE), encryption and decryption. The time of key generation may also play a major role in the case when a public-private key pair cannot be reused for security reasons. The resource utilization is secondary. Still, hardware designers typically aim at achieving the Pareto optimality, in which any further improvement in speed comes at the disproportionate cost in terms of resource utilization. The primary advantage of high-speed implementations is that they reveal the inherent potential of a given algorithm for parallelization. As long as the resource-utilization limit is sufficiently high, this limit does not affect the ranking of algorithms. As a result, the ranking is strongly correlated with the features of algorithms themselves and is not substantially influenced by any additional assumptions and technology choices. Additionally [4], only high-speed hardware implementations may effectively compete with optimized software implementations targeting high-performance processors with vector instructions (*e.g., AVX2*).

In lightweight implementations, the primary targets are typically minimum resource utilization and minimum power consumption, under the assumption that the execution time does not exceed a predefined maximum. Another way of formulating the goal is to achieve minimum execution time, assuming a given maximum budget in terms of resource utilization, power consumption, or energy usage. The maximum budget on resource utilization is related to the cost of implementation; the budget on power assures correct operation without overheating or devoting additional resources to cooling. The maximum energy usage affects how long a battery-operated device can function before the next battery recharge. In the context of the standardization process for cryptographic algorithms, the mentioned above maximum budgets are very hard to select. Any change in these thresholds may favor a different subset of candidates. With new standards remaining in use for decades, timing, cost, and power requirements of new applications are challenging to predict.

Additionally, changes in technology significantly affect which hardware architectures meet particular constraints. For example, an architecture capable of accomplishing the execution time of 0.1 seconds (*or below*), under a certain power or energy budget, may substantially change with the improvements in technology. As a result, the majority of current limits are selected somewhat arbitrarily by different designers, or left undefined in their reports. Consequently, the ranking of PQC candidates based on their lightweight implementations, especially those developed by different groups, is extremely challenging and assumption-dependent. These rankings have little to do with the parallelization allowed by each algorithm, as most of the operations must be executed sequentially due to the small resource budget. The primary feature of algorithms these implementations reveal is the number and complexity of its distinct elementary operations. Each major operation infers an additional functional unit, increasing resource utilization and power consumption. Additionally [4], lightweight hardware implementations can outperform software implementations targeting low-cost and low-power embedded processors (*for instance, such as Cortex-M4*).

In the case of FPGA implementations, resource utilization is a vector, such as (*#LUTs, #flip-flops, #DSP units, #BRAMs*). No single element of this vector can be expressed in terms of other elements. As a result, imposing a resource limit implies specifying the values of all components of this resource vector. One possible approach may be to choose the resources of the smallest FPGA of a given low-cost FPGA family. However, FPGA families and their resources change over time, so this limit has only a physical meaning during the limited time, covering the evaluation period, and may lose its significance just a few years after the standard is published and deployed. Besides, the

FPGA device may need to offset some of the costs associated with countering side-channel attacks (*moreover, this overhead and even countermeasures may remain unknown at the time of the candidates' evaluation*) [4].

4 FPGA Xilinx family

One of the major concerns is the NIST recommendation to focus on hardware benchmarking using the Xilinx Artix-7 FPGA family. This recommendation appeared in several NIST presentations related to Round 2 of the NIST standardization process, e.g., during PQCrypto 2019 in May 2019 and the Second PQC Standardization Conference in August 2019. We believe that, in its current form, this recommendation is counterproductive, and it impedes rather than supports fair and comprehensive hardware and software/hardware benchmarking [1].

FPGA family is a set of FPGA devices sharing the same internal structure and the same process technology (*also known as technology node or process node*), described by a number related to the size and density of transistors that can be fabricated using a given manufacturing process. With the steady improvements in process technology, described by Moore's Law, the maximum capacity and speed of FPGA devices have been steadily increasing while their prices have remained approximately the same. Every new generation of FPGA devices of a particular vendor receives a unique name, referred to as a family name. Every family consists of multiple devices with various distinct sizes to match the needs of different applications. All devices of a particular family share the same internal architecture and process technology but differ in terms of the number of resources of a particular type, such as Look-Up Tables (LUTs), flip-flops (FFs), block memories, and digital signal processing units (DSP units) or multipliers. Most vendors release both low-cost families (*such as Xilinx Artix-7) and high-performance families (such as Xilinx Virtex-7*). Most of them also release mid-range families, such as Xilinx Kintex-7. The maximum amount of resources available in the largest device of a low-cost family is naturally significantly smaller than the equivalent amount in the largest device of a high-performance family (*e.g., over 5 times smaller for Artix-7 vs. Virtex-7*).

Additionally, in recent years, FPGA vendors started releasing new types of programmable devices that enhance Programmable Logic of traditional FPGAs with the Processing System based on a hardwired embedded processor, such as ARM. Since this processor is custom designed, it takes full advantage of a given technological process and operates at a clock frequency significantly higher than Programmable Logic. With a fast processor and an efficient interface between this processor and Programmable Logic, these devices are ideal for software/hardware co-designs targeting high-speed. Although these types of devices appear under multiple commercial names, they are often collectively referred to as System on Chip FPGAs (SoC FPGAs). The first family of this type was Xilinx Zynq-7000, released in 2011, based on ARM Cortex-A9 embedded processors [4].

Hardware designs are described in hardware description languages. HDL code is typically identical for all FPGA families. As opposed to software, where each processor may require different optimized assembly language code, no such concepts exist for hardware. Thus, it is possible to synthesize the same HDL code targeting various FPGA families from various vendors, as long as the maximum capacity of the largest device of a given family is not exceeded.

5 General features of the FPGA Xilinx family

Today, the most modern is the series 7 FPGA Xilinx – Artix-7, Kintex-7, Virtex-7. In this series, the FPGA family with the ARM Cortex-A9 - Zynq-7000 processor core has been announced. In the new series, only Virtex-7 continues the existing line of high-performance FPGA, and the other two families – Artix and Kintex – replaced the Spartan line. FPGA Artix are designed for mass products, and are characterized by low power consumption and low cost, and Kintex is, to some extent, Spartan, specialized for digital signal processing. Until now, the Virtex series has traditionally been used in applications built around high-speed serial receivers and in projects based on digital signal processing. The Kintex-7 family «fits» well into a niche where a large number of parallel DSC units are required at a moderate price [5], and more expensive Virtex-7 are intended for systems with a large number of hardware receivers (see Table 1).

Zynq-7010 and Zynq-7020 chips are based on playable resources of the Artix family, and Zynq-7030 and Zynq-7040 are based on Kintex. This affects in the peak performance [6] of the digital signal processing subsystem (*the frequency of the lower FPGA Zynq is lower, they do not have PCI Express units and high-speed receivers*) (see Table 2).

A key feature of the next generation of FPGA is the unification of playable resources. It is assumed that for the next generation of FPGA, a quick migration between Virtex/Kintex/Artix families will be possible without adjusting the project.

Table 1 – FPGA Xilinx family, 7th series

Maximum parameters	**Artix-7**	**Kintex-7**	**Virtex-7**
Logical cells, thousand	*352*	*407*	*1955*
Block memory	*12*	*29*	*65*
DSP sections	*700*	*1540*	*3960*
Peak digital signal processing performance*, *GMAC/s*	*504*	*1965*	*5053*
Receivers	*4*	*16*	*88*
Maximum transfer rate, *Gb/s*	*3,75*	*10.325*	*28,05*
Peak capacity of receivers, *Gb/s*	*30*	*330*	*2784*
PCI Express interfaces	*Gen1x4*	*Gen2x8*	*Gen3x8*
Memory interface exchange speed, *MB/s*	*800*	*2133*	*2133*
External outputs	*450*	*500*	*1200*

** – for symmetric coefficient filters.*

Table 2 – FPGA Zynq-7000 family

Parameters	**Z -7010**	**Z -7020**	**Z -7030**	**Z -7040**
Programmable logic cells (ASIC gates)	*28 K (430 K)*	*85 K (1,3 M)*	*125 K (1,9 M)*	*235 K (3,5 M)*
Memory blocks (36 KB)	*60*	*140*	*265*	*760*
DSP sections (18x25 MACC)	*80*	*220*	*400*	*760*
Peak DSP performance*, GMAC/s	*58*	*158*	*480*	*912*
PCI Express blocks	-	-	*Gen2 x4*	*Gen2 x8*
ADC	*2x12 bits, 1 M samples/sec, 17 dif. channels*			
Encryption	*AES and SHA 256-bit*			
I/O units, 3.3 V	*100*	*195*	*100*	*200*
I/O units, 1.8 V	-	-	*150*	*150*
High speed receivers	-	-	*4*	*12*

** – for KIX with symmetric coefficients.*

6 FPGA Xilinx family details

Tables 3-6 show the characteristics of the FPGA family Xilinx, namely Spartan-7, Artix-7, Virtex-7, Kintex-7, respectively [6].

Giving preference to the Xilinx Artix-7 family has several undesired consequences summarized below:

1. Artix-7 is a low-cost FPGA family. As such, it is not very suitable for high-speed implementations. Hardware resources of even the largest device of this family are often insufficient to demonstrate the full potential for parallelizing operations a given PQC algorithm. Thus, the use of Artix-7 makes perfect sense for benchmarking lightweight implementations but may lead to suboptimal results for high-speed implementations.

Table 3 – Characteristic of Spartan-7

Parameters		I/O optimization at the lowest cost and highest performance-per-watt (1.0 V, 0.95 V)					
Part number		***XC7S6***	***XC7S15***	***XC7S25***	***XC7S50***	***XC7S75***	***XC7S100***
Logic resources	Logic cells	*6,000*	*12,800*	*23,360*	*52,160*	*76,800*	*102,400*
	Slices	*938*	*2,000*	*3,650*	*8,150*	*12,000*	*16,000*
	CLB flip-flops	*7,500*	*16,000*	*29,200*	*65,200*	*96,000*	*128,000*
Memory resources	Max. distributed RAM, (Kb)	*70*	*150*	*313*	*600*	*832*	*1,100*
	Block RAM/FIFO w/ ECC (36 Kb each)	*5*	*10*	*45*	*75*	*90*	*120*
	Total Block RAM (Kb)	*180*	*360*	*1,620*	*2,700*	*3,240*	*4,320*
Clock resources	Clock Mgmt Tiles (1 MMCM + 1 PLL)	*2*	*2*	*3*	*5*	*8*	*8*
I/O resources	Max. single-ended I/O pins	*100*	*100*	*150*	*250*	*400*	*400*
	Max. differential I/O pairs	*48*	*48*	*72*	*120*	*192*	*192*
Embedded hard IP resources	DSP slices	*10*	*20*	*80*	*120*	*140*	*160*
	Analog mixed signal (AMS) / XADC	*0*	*0*	*1*	*1*	*1*	*1*
	Configuration AES / HMAC blocks	*0*	*0*	*1*	*1*	*1*	*1*
Speed grades	Commercial temp, (C)	*-1 -2*	*-1 -2*	*-1 -2*	*-1 -2*	*-1 -2*	*-1 -2*
	Industrial temp (I)	*-1 -2 -1L*	*-1 -2 -1L*	*-1 -2 -1L*	*-1 -2 -1L*	*-1 -2 -1L*	*-1 -2 -1L*
	Expanded temp (Q)	*-1*	*-1*	*-1*	*-1*	*-1*	*-1*

Package	Body area (mm)	Ball pitch (mm)	Available user I/O: 3.3 V SelectIO™ HR I/O					
CPGA196	*8x8*	*0.5*	*100*	*100*				
CSGA225	*13x13*	*0.8*	*100*	*100*	*150*			
CSGA324	*15x15*	*0.8*			*150*	*210*		
FTGB196	*15x15*	*1.0*	*100*	*100*	*100*	*100*		
FGGA484	*23x23*	*1.0*				*250*	*338*	*338*
FGGA676	*27x27*	*1.0*					*400*	*400*

2. Artix-7 is a traditional FPGA, and not a SoC FPGA. As a result, the only way to develop a single-chip software/hardware implementation using Artix-7 is the use of so-called "soft" processor cores, i.e., processors implemented using programmable logic. Soft processors compatible with Artix-7 include MicroBlaze and lightweight versions of RISC-V. All of them operate at much lower clock frequency than hardwired embedded processors of SoC FPGAs.

3. Artix-7 is unsuitable for HLS designs. Such designs typically take significantly more resources than designs based on writing code manually in HDL. As a result, assuming the Pareto optimization for high-speed, they are unlikely to fit in the largest Artix-7 FPGA.

4. Artix-7 is a relatively old FPGA family, released by Xilinx in 2010. By the time of the release of the PQC standard, this family will be at least 12 years old. While still relatively popular for low-cost applications, this family does not represent the state-of-the-art in FPGA technology.

5. It is not customary to base ranking of candidates in cryptographic contests on results obtained for a single family of a single vendor. Although Xilinx is the largest developer of FPGAs and SoC FPGAs, Intel comes a strong second, and other vendors, such as Microchip and Lattice Semiconductor, also develop FPGAs suitable for implementing cryptographic algorithms. During the SHA-3 competition, the results were reported for seven FPGA families from two major vendors, Xilinx and Altera. During the CAESAR contest, four Xilinx families and four Altera families were employed. For all of these families, results were generated based on the same HDL code. There was no need to purchase multiple tools or boards. Free or trial versions of tools were sufficient. The designs ended with the generation of post-place-and-route reports, which correctly described the worst-case performance of any particular instance of the given FPGA device.

Table 4 – Characteristic of Artix-7

				Transceiver optimization at the lowest cost and highest DSP bandwidth (1.0 V, 0.95 V, 0.9 V)							
Part number				***XC7A12T***	***XC7A15T***	***XC7A25T***	***XC7A35T***	***XC7A50T***	***XC7A75T***	***XC7A100T***	***XC7A200T***
Logic resources	Logic cells			12800	16,640	23,360	33,280	52,160	75,520	101,440	215,360
	Slices			2,000	2,600	3,650	5,200	8,150	11,800	15,850	33,650
	CLB flip-flops			16,000	2,800	29,200	41,600	65,200	94,400	126,800	269,200
Memory resources	Maximum distributed RAM (Kb)			171	200	313	400	600	892	1,188	2,888
	Block RAM/FIFO w/ ECC (36 Kb each)			20	25	45	50	75	105	135	365
	Total block RAM (Kb)			720	900	1,620	1,800	2,700	3,780	4,860	13,140
Clock resources	CMTs (1 MMCM + 1 PLL)			3	5	3	5	5	6	6	10
I/O resources	Maximum single-ended I/O			150	250	150	250	250	300	300	500
	Maximum differential I/O pairs			72	120	12	120	120	144	144	240
Embed-ded hard IP resources	DSP slices			40	45	80	90	120	180	240	740
	PCIe® Gen2			1	1	1	1	1	1	1	1
	Analog mixed signal (AMS) / XADC			1	1	1	1	1	1	1	1
	Configuration AES / HMAC blocks			1	1	1	1	1	1	1	1
	GTP Transceivers (6.6 Gb/s max rate)			2	4	4	4	4	8	8	16
Speed grades	Commercial temp (C)			-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2
	Extended temp (E)			-2L -3	-2L -3	-2L -3	-2L -3	-2L -3	-2L -3	-2L -3	-2L -3
	Industrial temp (I)			-1 -2 -1L	-1 -2 -1L	-1 -2 -1L	-1 -2 -1L	-1 -2 -1L	-1 -2 -1L	-1 -2 -1L	-1 -2 -1L
	Package	Dimen-sions (mm)	Ball pitch (mm)	Available user I/O: 3.3 V SelectIO™ HR I/O (*GTP transceivers*)							
Footprint compati-ble	**CPG236**	10x10	0.5		106(2)		106(2)	106(2)			
	CPG238	10x10	0.5	112(2)		112(2)					
	CSG324	15x15	0.8		210(0)		210(0)	210(0)	210(0)	210(0)	
	CSG325	15x15	0.8	150(2)	150(4)	150(4)	150(4)	150(4)			
	FTG256	17x17	1.0		170(0)		170(0)	170(0)	170(0)	170(0)	
	SBG484	19x19	0.8								285(4)
	FGG484	23x23	1.0		250(4)		250(4)	250(4)	285(4)	285(4)	
	FBG484	23x23	1.0								285(4)
	FGG676	27x27	1.0						300(8)	300(8)	
	FBG676	27x27	1.0								400(8)
	FFG1156	35x35	1.0								500(16)

Table 5 – Characteristic of Kintex-7

				Optimized for best price-performance (1.0 V, 0.95 V, 0.9 V)						
Part number				***XC7K70T***	***XC7K160T***	***XC7K325T***	***XC7K355T***	***XC7K410T***	***XC7K420T***	***XC7K480T***
Logic resources	Slices			10,250	25,350	50,950	55,650	63,550	63,150	74,650
	Logic cells			65,600	162,240	326,080	356,160	406,720	416,960	477,760
	CLB flip-flops			82,000	202,800	407,600	445,200	508,400	521,200	597,200
Memory resources	Maximum distributed RAM, (Kb)			838	2,188	4,000	5,088	5,663	5,938	6,778
	Block RAM/FIFO w/ ECC (36 Kb each)			135	325	445	715	795	835	955
	Total block RAM, (Kb)			4,860	11,700	16,020	25,740	28,620	30,060	34,380
Clock resources	CMTs (1 MMCM + 1 PLL)			6	8	10	6	10	8	8
I/O resources	Maximum single-ended I/O			300	400	500	300	500	400	400
	Maximum differential I/O pairs			144	192	240	144	240	192	192
Integrated IP resources	DSP48 slices			240	600	840	1,440	1,540	1,680	1,920
	PCIe® Gen2			1	1	1	1	1	1	1
	Analog mixed signal (AMS) / XADC			1	1	1	1	1	1	1
	Configuration AES / HMAC blocks			1	1	1	1	1	1	1
	GTX transceivers (12.5 Gb/s max rate)			8	8	16	24	16	32	32
Speed grades	Commercial Temp, (C)			-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2
	Extended temp (E)			-2L -3	-2L -3	-2L -3	-2L -3	-2L -3	-2L -3	-2L -3
	Industrial temp (I)			-1 -2 -2L	-1 -2 -2L	-1 -2 -2L	-1 -2 -2L	-1 -2 -2L	-1 -2 -2L	-1 -2 -2L
	Package	Dimensions (mm)	Ball pitch (mm)	Available user I/O: 3.3 V HR I/O, 1.8 V HP I/Os (*GTX*)						
Footprint compatible	**FBG484**	23x23	1.0	185, 100 (4)	185, 100 (4)					
	FBG676	27x27	1.0	200, 100 (8)	250, 150 (8)	250, 150 (8)		250, 150 (8)		
	FFG676	27x27	1.0		250, 150 (8)	250, 150 (8)		250, 150 (8)		
	FBG900	31x31	1.0			350, 150 (16)		350, 150 (16)		
	FFG900	31x31	1.0			350, 150 (16)		350, 150 (16)		
	FFG901	31x31	1.0				300, 0 (24)		380, 0 (28)	380, 0 (28)
	FFG1156	35x35	1.0						400, 0 (32)	400, 0 (32)

6. Based on the authors' experiences, multiple reviewers of papers devoted to implementations of Round 2 PQC candidates treated the NIST's choice of Artix-7 as an absolute requirement. Submissions not complying with this requirement were subject to rejection or requests for major revisions. As a result, a noble goal of making the results more comparable with one another was turned into a reason for suppressing or delaying the publication of relevant results.

Table 6 – Characteristic of Virtex-7

		Optimized for highest system performance and capacity (1.0 V)										
Part number		***XC7V585T***	***XC7V2000T***	***XC7VX330T***	***XC7VX415T***	***XC7VX485T***	***XC7VX550T***	***XC7VX690T***	***XC7VX980T***	***XC7VX1140T***	***XC7VH580T***	***XC7VH870T***
Logic Resources	Slices	91,050	305,400	51,000	64,400	75,900	86,600	108,300	153,000	178,000	90,700	136,900
	Logic cells	582,720	1,954,560	326,400	412,160	485,760	554,240	693,120	979,200	1,139,200	580,480	876,160
	CLB flip-flops	728,400	2,443,200	408,000	515,200	607,200	692,800	866,400	1,224,000	1,424,000	725,600	1,095,200
Memory Resources	Maximum distributed RAM (Kb)	6,938	21,550	4,388	6,525	8,175	8,725	10,888	13,838	17,700	8,850	13,275
	Block RAM/FIFO w/ ECC (36 Kb each)	795	1,292	750	880	1,030	1,180	1,470	1,500	1,880	940	1,410
	Total block RAM (Kb)	28,620	46,512	27,000	31,680	37,080	42,480	52,920	54,000	67,680	33,840	50,760
Clocking	CMTs (1 MMCM + 1 PLL)	18	24	14	12	14	20	20	18	24	12	18
I/O Resources	Maximum single-ended I/O	850	1,200	700	600	700	600	1,000	900	1,100	600	300
	Maximum differential I/O pairs	408	576	336	288	336	288	480	432	528	288	144
Integrated IP Resources	DSP slices	1,260	2,160	1,120	2,160	2,800	2,880	3,600	3,600	3,360	1,680	2,520
	PCIe® Gen2	3	4	-	-	4	-	-	-	-	-	-
	PCIe Gen3	-	-	2	2	-	2	3	3	4	2	3
	Analog mixed signal (AMS) / XADC	1	1	1	1	1	1	1	1	1	1	1
	Configuration AES / HMAC blocks	1	1	1	1	1	1	1	1	1	1	1
	GTX transceivers (12.5 Gb/s max rate)	36	36	-	-	56	-	-	-	-	-	-
	GTH transceivers (13.1 Gb/s max rate)	-	-	28	48	-	80	80	72	96	48	72
	GTH transceivers (28.05 Gb/s max rate)	-	-	-	-	-	-	-	-	-	8	16
Speed Grades	Commercial temp, (C)	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2
	Extended temp (E)	-2L -3	-2L -2G	-2L -3	-2L -3	-2L -3	-2L -3	-2L -3	-2L	-2L -2G	-2L -2G	-2L -2G
	Industrial temp (I)	-1 -2	-1	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1	-1	-	-

Continuation of Table 6

			Optimized for highest system performance and capacity (1.0 V)										
Part number			***XC7V585T***	***XC7V2000T***	***XC7VX330T***	***XC7VX415T***	***XC7VX485T***	***XC7VX550T***	***XC7VX690T***	***XC7VX980T***	***XC7VX1140T***	***XC7VH580T***	***XC7VH870T***
Package	Dimen-sions (mm)	Ball Pitch (mm)	Available user I/O: 3.3 V HR I/O, 1.8 V HP I/Os (*GTX*, *GTH*)									1.8 V HP I/O (*GTH*, *GTZ*)	
FFG1157	35x35	1.0	0, 600 (20, 0)		0, 600 (20, 0)	0, 600 (20, 0)	0, 600 (20, 0)		0, 600 (20, 0)				
FFG1761	42.5x42.5	1.0	100, 750 (36, 0)		50, 650 (0, 28)		0, 700 (28, 0)		0, 850 (0, 36)				
FHG1761	45x45	1.0		0, 850 (36, 0)									
FLG1925	35x35	1.0		0, 1200 (16, 0)									
FFG1158	45x45	1.0				0, 350 (0, 48)	0, 350 (0, 48)	0, 350 (0, 48)	0, 350 (0, 48)				
FFG1926		1.0							0, 720 (0, 64)	0, 720 (0, 64)			
FLG1926		1.0									0, 720 (0, 64)		
FFG1927		1.0				0, 600 (0, 48)	0, 600 (56, 0)	0, 600 (0, 80)	0, 600 (0, 80)				
FFG1928		1.0								0, 480 (0, 72)			
FLG1928		1.0									0, 480 (0, 96)		
FFG1930		1.0					0, 700 (24, 0)		0, 1000 (0, 24)	0, 900 (0, 24)			
FLG1930		1.0									0, 1100 (0, 24)		
FLG1155	35x35	1.0										400 (24, 8)	
FLG1931	45x45	1.0										600 (48, 8)	
FLG1932		1.0											300 (72, 16)

Taking these concerns into account, our recommendation for Round 3 is to encourage reporting results for at least the following FPGA families:

1. For lightweight hardware implementations and lightweight software/hardware implementations based on soft processor cores: Xilinx Artix-7 (f*or compatibility with Round 2 results*) and Intel Cyclone 10 LP.

2. For lightweight software/hardware implementations based on the use of hard processor cores: Xilinx Zynq 7000-series and Intel Cyclone V SoC FPGAs.

3. For high-speed hardware and high-speed software/hardware implementations: Zynq Xilinx UltraScale+ and Intel Stratix 10 SoC.

One of the reasons for selecting Zynq Xilinx UltraScale+, even for pure hardware implementations that do not require SoC capabilities, is the support for these devices by the free version of the Xilinx toolset, called Vivado HL WebPACK, which is sufficient to generate all required benchmarking results. Xilinx Virtex-7 UltraScale+ FPGAs, which could be considered as a natural candidate, are not supported by the same free version of tools. The Zynq Xilinx UltraScale+ family is also recommended for high-speed software/hardware implementations based on the use of hard processor cores because of moderate cost of suitable prototyping boards and the availability of a free Benchmarking Setup for Software/Hardware Implementations of PQC Schemes, developed at George Mason University [7].

7 FPGA-focused implementations

In Tables 7-8, summarize implementations targeting Xilinx Artix-7 FPGAs and related Xilinx Zynq-7000 SoC FPGAs (*indicated with the superscript* ^{Z}). For the security level 1, six candidates – Classic McEliece, CRYSTALS-Kyber, FrodoKEM, NewHope, SIKE, and Saber – have implementations of all three operations reported. The preliminary implementations of BIKE focused on key generation only. For security level 3, NewHope does not have a variant. For security level 5, the results are missing for Classic McEliece.

For most KEMs, the time of decapsulation is longer than the time of encapsulation. Table entries are ordered according to the time of decapsulation in μs (*and, if needed, according to the decapsulation time in clock cycles*).

The ranking of candidates listed in Tables 7 and 8 is very challenging to determine based on available results. First, it may be unfair to compare pure hardware implementations with software/hardware implementations. Secondly, it is hard to compare lightweight implementations with high-speed implementations, as they are optimized with different primary metrics in mind. Third, software/hardware implementations based on different processors are very challenging to compare with one another. Finally, even for implementations using exactly the same type of implementation (*software/hardware*) and the same type of processor (RISC-V), the comparison may be unintentionally biased. Significantly different hardware support was provided for algorithms that can take advantage of the Number Theoretic Transform – Kyber and NewHope – vs. the algorithm that cannot – Saber. An additional, relatively minor factor is that several results for Classic McEliece and NewHope concern their IND-CPA-secure PKEs rather than IND-CCA-secure KEMs.

Taking all these factors into account, almost the only ranking that is quite clear from Tables 7 and 8 is the ranking of candidates that have results available for pure hardware implementations targeting high-speed. In this specific category, the ranking for the security level 1 is: 1. NewHope, 2. Classic McEliece, 3. FrodoKEM. If we assume that a software/hardware implementation of SIKE with a custom processor is almost as efficient as a pure hardware implementation, then we can also add SIKE at position 4. At level 3, NewHope does not have a variant, and at level 5, Classic McEliece and FrodoKEM, do not have high-speed pure hardware implementations reported.

In Tables 9 and 10, summarize implementations targeting Xilinx Virtex-7 FPGAs. Unfortunately, the only conclusion that can be drawn from these tables is an advantage of Classic McEliece over SIKE in terms of all performance metrics other than the number of LUTs and flip-flops.

In Table 11, we compare results reported by our own group at the end of 2019 in [4, 8-9], with results reported by other groups for Saber and NewHope, respectively. All results were obtained using the same SoC FPGA, Zynq UltraScale+. The software/hardware implementation of Round5 was very close to the pure hardware implementation. The same was not the case for the software/hardware implementation of Saber, were a significant percentage of the execution time was devoted to functions remaining in software and to the transfer of data and control between software and hardware. As a result, the most accurate comparison between Round5 and Saber is possible at the security level 3, for which the pure hardware implementation of Saber was reported in [3, 5]. Based on this implementation Saber outperforms Round5 by a small margin in terms of the execution times for encapsulation and decapsulation.

Table 7 – Level 1 KEMs and PKEs on Artix-7 (*default*) & Zynq-7000

Algorithm	Type	Target	Max. Freq.	LUT	FF	Slice	DSP	BRAM	Key Generation		Encaps. / Enc.^{cpa}		Decaps. / (Dec.+Enc.)^{cpa}	
									cycles	μs	cycles	μs	cycles	μs
Security Level 1														
NewHope-512^{cpa}	HW	HS	200	6,780	4,026	–	2	7.0	4,200	21.0	6,600	33.0	9,100	45.5
mceliece348864^{cpa}	HW	HS	106	81,339	132,190	–	0	236.0	202,787	1,920.3	2,720	25.8	12,743	120.7
mceliece348864^{cpa}	HW	HS	108	25,327	49,383	–	0	168.0	1,599,882	14,800.0	2,720	25.2	18,358	169.8
Kyber-512	SW/HW^{RV}	LW	–	23,925	10,844	–	21	32.0	150,106	–	193,076	–	204,843	–
FrodoKEM-640	HW	HS	172	2,587	2,994	855	16	0	–	–	–	–	–	–
			171	5,796	4,694	1,692	16	0	204,766	1,190.5	207,269	1,212.1	209,867	1,408.5
16x			149	6,881	5,081	1,947	16	12.5	–	–	–	–	–	–
Kyber-512	SW/HW^{RV}	LW	25*	14,975	2,539	4,173	11	14.0	74,519	2,980.8	131,698	5,267.9	142,309	5,692.4
NewHope-512	SW/HW^{RV}	LW	–	23,925	10,844	–	21	32.0	123,860	–	207,299	–	226,742	–
NewHope-512	SW/HW^{RV}	LW	25*	14,975	2,539	4,173	11	14.0	97,969	3,918.8	236,812	9,472.5	258,872	10,354.9
LightSaber	SW/HW^{RV}	LW	–	23,925	10,844	–	21	32.0	366, 837	–	526, 496	–	657,583	–
Kyber-512	SW/HW^{RV}	LW	59	1,842	1,634	–	5	34.0	710,000	11,993.2	971,000	16,402.0	870,000	14,695.9
NewHope-512	SW/HW^{RV}	LW	59	1,842	1,634	–	5	34.0	904,000	15,270.3	1,424,000	24,054.1	1,302,000	21,993.2
SIKEp434	SW/HW^{c}	HS	162	22,595	11,558	7,491	162	37.0	1,474,200	9100	2,494,800	15,400.0	2,656,800	16,400.0
SIKEp503	SW/HW^{c}	HS	162	22,595	11,558	7,491	162	37.0	1,733,400	10,700.0	2,932,200	18,100.0	3,126,600	19,300.0
FrodoKEM-640	HW	LW	191	971	433	290	1	0						
			190	4,246	2,131	1,180	1	0	3,237,288	16,949.2	3,275,862	17,241.4	3,306,122	20,408.2
1x			162	4,446	2,152	1,254	1	12.5	–	–	–	–	–	–
	SW/HW^{c}								–	–	–	–	–	–
SIKEp434		LW	143	10,976	7,115	3,512	57	21.0	2,187,902	15,300.0	3,718,004	26,000.0	3,946,804	27,600.0
SIKEp503	SW/HW^{c}	LW	143	10,976	7,115	3,512	57	21.0	2,602,603	18,200.0	4,390,104	30,700.0	4,676,105	32,700.0
FrodoKEM-640	SW/HW^{RV}	LW	25*	14,975	2,539	4,173	11	14.0	11,453,942	458,157.7	11,609,668	464,386.7	12,035,513	481,420.5
BIKE-1 Level 1^{cs}	HW	HS	165	1,907	1,049	608	0	7.0	95,500	578.0	–	–	–	–
BIKE-3 Level 1^{cs}	HW	HS	170	1,397	925	453	0	4.0	98,500	579.0	–	–	–	–
BIKE-2 Level 1^{cs}	HW	HS	160	3,874	2,141	1,312	0	10.0	2,150,000	13,437.0	–	–	–	–
BIKE Level 1	HW	HS	135	1,865	589	590	0	4.0	7,370,429	54,540.0	–	–	–	–

Table 8 – Level 3 & 5 KEMs and PKEs on Artix-7 (*default*) & Zynq-7000

Algorithm	Type	Target	Max. Freq.	LUT	FF	Slice	DSP	BRAM	Key Generation		Encaps. / Enc.^{cpa}		Decaps. / (Dec.+Enc.)^{cpa}	
									cycles	μs	cycles	μs	cycles	μs
Security Level 3														
mceliece460896^{cpa}	HW	HS	107	38,669	74,858	–	0	303.0	5,002,044	46,704.4	3,360	31.4	31,005	289.5
FrodoKEM-976	HW	HS	169	2,869	3,000	908	16	0	–	–	–	–	–	–
16x			168	6,188	4,678	1782	16	0	476,056	2,816.9	479,993	2,857.1	483,073	3,076.9
			157	7,213	5,087	2042	16	19.0	–	–	–	–	–	–
Saber^{Z}	SW/HW^{A9}	HS							–	–	–	–	–	–
			125	7,400	7,331	–	28	2.0	–	3,273.0	–	4,147.0	–	3,844.0
Kyber-768	SW/HW^{RV}	LW	25*	14,975	2,539	4,173	11	14.0	111,525	4,461.0	177,540	7,101.6	190,579	7,623.2
SIKEp610	SW/HW^{c}	HS	162	22,595	11,558	7,491	162	37.0	2,916,000	18,000.0	5,443,200	33,600.0	5,508,000	34,000.0
FrodoKEM-976	HW	LW	189	1,243	441	362	1	0	–	–	–	–	–	–
1x			187	4,650	2,118	1,272	1	0	7,560,000	40,000.0	7,480,000	40,000.0	7,714,286	47,619.0
			162	4,888	2,153	1,390	1	19.0	–	–	–	–	–	–
SIKEp610	SW/HW^{c}								–	–	–	–	–	–
		LW	143	10,976	7,115	3,512	57	21.0	4,347,204	30,400.0	8,108,108	56,700.0	8,208,208	57,400.0
FrodoKEM-976	SW/HW^{RV}	LW	25*	14,975	2,539	4,173	11	14.0	26,005,326	1,040,213.0	29,749,417	1,189,976.7	30,421,175	1,216,847.0
BIKE Level 3	HW	HS	135	1,884	557	593	0	5	30,447,947	231,400.0	–	–	–	–

Continuation of Table 8

Algorithm	Type	Target	Max. Freq.	LUT	FF	Slice	DSP	BRAM	Key Generation		Encaps. / Enc.^{cpa}		Decaps. / (Dec.+Enc.)^{cpa}	
									cycles	μs	cycles	μs	cycles	μs
Security Level 5														
NewHope-1024^{cpa}	HW	HS	200	6,781	4,127	–	2	8.0	8,000	40.0	12,500	62.5	17,300	86.5
NewHope-1024^{cpa}	HW	HS	190	13,244	8,272	–	24	18.0	–	–	34,000	178.0	30,600^{KD}	160.0^{KD}
Kyber-1024	SW/HW^{RV}	LW	25*	14,975	2,539	4,173	11	14.0	148,547	5,941.9	223,469	8,938.8	240,977	9,639.1
NewHope-1024	SW/HW^{RV}	LW	25*	14,975	2,539	4,173	11	14.0	97,969	3,918.8	236,812	9,472.5	258,872	10,354.9
Kyber-1024	SW/HW	LW	–	23,925	10,844	–	21	32.0	349,673	–	405,477	–	424,682	–
NewHope-1024	SW/HW	LW	–	23,925	10,844	–	21	32.0	235,420	–	392,734	–	450,541	–
NewHope-1024^{cpa}	SW/HW	HS	25	26,606	26,303	–	32	1.0	357,052	14,282.1	589,285	23,571.4	756,932	30,277.3
FireSaber	SW/HW	LW	–	23,925	10,844	–	21	32.0	1,300,272	–	1,622,818	–	1,898,051	–
Kyber-1024	SW/HW^{RV}	LW	59	1,842	1,634		5	34.0	2,203,000	37,212.8	2,619,000	44,239.9	2,429,000	41,030.4
SIKEp751	SW/HW^{c}	HS	162	22,595	11,558	7,491	162	37.0	3,742,200	23,100.0	6,188,400	38,200.0	6,658,200	41,100.0
NewHope-1024	SW/HW^{RV}	LW	59	1,842	1,634		5	34.0	1,776,000	30,000.0	2,742,000	46,317.6	2,528,000	42,702.7
SIKEp751	SW/HW^{c}	LW	143	10,976	7,115	3,512	57	21.0	7,965,108	55,700.0	13,156,013	92,000.0	14,185,614	99,200.0
FrodoKEM-1344	SW/HW^{RV}	LW	25*	14,975	2,539	4,173	11	14.0	67,994,170	2,719,766.8	71,501,358	2,860,054.3	72,526,695	2,901,067.8

At the same time, even the fastest reported implementation of Saber uses 1.6x fewer LUTs than Round5, with the same number of BRAMs and DSP units. FrodoKEM is demonstrated to be by far slower than Saber and Round5 for all security levels.

Somewhat differently, for the security level 5, the pure hardware implementation of NewHope, reported in [9], is not fast enough to outperform the software/hardware implementation of Round5 from [10]. However, the comparison is somewhat complicated by the fact that, in [9], the results are reported the IND-CPA-secure PKE (*rather than the IND-CCA-secure KEM*), and only the sum of the key generation and decryption (rather than the decryption itself) is reported in the paper.

In Tables 12, summarize results available for the implementations of digital signatures. The implementations targeting FPGAs are considered first in Table 9.

Table 9 – Level 1 KEMs on Virtex-7 (default) & Virtex-6 (*indicated with the superscript* ^{V6})

Algorithm	Type	Target	Max. Freq.	LUT	FF	Slice	DSP	BRAM	Key Generation		Encap./Enc.^{cpa}		Decaps./Dec.^{cpa}	
									cycles	μs	cycles	μs	cycles	μs
Security Level 1														
SIKEp503	HW	HS	171	25,094	26,971	9,514	264	34.0	640,000	3,738.3	1,120,000	6,542.1	1,210,000	7,067.8
SIKEp434	SW/HW	HS	142	21,210	13,657	7,408	162	38.0	981,180	6,900.0	1,677,960	11,800.0	1,777,500	12,500.0
SIKEp503	SW/HW	HS	142	21,210	13,657	7,408	162	38.0	1,166,040	8,200.0	1,976,580	13,900.0	2,104,560	14,800.0
LEDAkem-128^{o,cpa,V6}	HW	LW	235	104	53	33	0	1.0	–	–	712,000	3,029.8	2,620,000	18,714.3
SIKEp434	SW/HW	LW	152	10,937	7,132	3,415	57	21.0	2,191,781	14,400.0	3,713,851	24,400.0	3,957,382	26,000.0
SIKEp503	SW/HW	LW	152	10,937	7,132	3,415	57	21.0	2,602,740	17,100.0	4,383,562	28,800.0	4,672,755	30,700.0

^{cpa} - Design of a KEM variant resistant against Chosen-Plaintext Attack (CPA)
^{V6} - Design implemented on Virtex-6
^{o} - Design for an old parameter set changed by the submitters on March 19th, 2020

Unfortunately, multiple results available for qTESLA concern heuristic parameter sets that have been withdrawn by submitters on Aug. 20, 2019. Among the remaining designs, for Artix-7, the ranking of candidates for the security level 1 is 1. Picnic, 2. Dilithium, and 3. qTESLA. The differences among these candidates in terms of the execution time for the signature generation (*more critical*) and signature verification are very significant. At the same time, only the implementation of Picnic is a high-speed and pure hardware implementation. The remaining implementations are software/hardware implementations based on RISC-V. Additionally, the number of LUTs for Picnic is approximately 6 times larger than for Dilithium, and the number of BRAMs, 3.75 times larger. At the same time, compared to Picnic, the execution time for signature generation is 12 times longer for Dilithium-I and 16 times longer for Dilithium-II.

Table 10 – Level 3 & 5 KEMs & PKEs on Virtex-7

Algorithm	Type	Target	Max. Freq.	LUT	FF	Slice	DSP	BRAM	Key Generation		Encap./Enc.^{cpa}		Decaps./Dec.^{cpa}	
									cycles	μs	cycles	μs	cycles	μs
Security Level 3														
mceliece460896^{cpa}	HW	HS	131	109,484	168,939	–	0	446.0	515,806	3,943.5	3,360	25.7	17,931	137.1
SIKEp610	SW/HW	HS	142	21,210	13,657	7,408	162	38.0	1,962,360	13,800.0	3,654,540	25,700.0	3,711,420	26,100.0
SIKEp610	SW/HW	LW	152	10,937	7,132	3,415	57	21.0	4,353,120	28,600.0	8,097,412	53,200.0	8,219,178	54,000.0
Security Level 5														
mceliece6960119^{cpa}	HW	HS	130	116,928	188,324	–	0	607.0	974,306	7,500.4	5,413	41.7	25,135	193.5
mceliece6688128^{cpa}	HW	HS	137	122,624	186,194	–	0	589.0	1,046,139	7,658.4	5,024	36.8	29,754	217.8
mceliece8192128^{cpa}	HW	HS	130	123,361	190,707	–	0	589.0	1,286,179	9,901.3	6,528	50.3	32,765	252.2
mceliece6960119^{cpa}	HW	HS	141	44,154	88,963	–	0	563.0	11,179,636	79,570.4	5,413	38.5	46,141	328.4
mceliece6688128^{cpa}	HW	HS	136	44,345	83,637	–	0	446.0	12,389,742	91,034.1	5,024	36.9	52,333	384.5
mceliece8192128^{cpa}	HW	HS	134	45,150	88,154	–	0	525.0	15,185,314	113,154.4	6,528	48.6	55,330	412.3
SIKEp751	HW	HS	167	45,893	50,390	17,53	512	43.5	1,240,000	7,407.4	2,170,000	12,963.0	2,330,000	13,918.8
SIKEp751	SW/HW	HS	142	21,210	13,657	7,408	162	38.0	2,516,940	17,700.0	4,166,460	29,300.0	4,479,300	31,500.0
SIKEp751	SW/HW	LW	152	10,937	7,132	3,415	57	21.0	7,960,426	52,300.0	13,150,685	86,400.0	14,185,693	93,200.0

Table 11 – All KEMs and PKEs on Zynq Ultrascale +

Algorithm	Type	Target	Max. Freq.	LUT	FF	Slice	DSP	BRAM	Key Generation		Encapsulation		Decapsulation	
									cycles	μs	cycles	μs	cycles	μs
Security level 1														
R5ND_1KEM_0d	SW/HW	HS	260	55,442	82,341	10,627	0	2	–	–	–	19.0	–	24.0
LightSaber	SW/HW	HS	322	12,343	11,288	1,989	256	3.5	–	–	–	53.0	–	56.0
FrodoKEM-640	SW/HW	HS	402	7,213	6,647	1,186	32	13.5	–	–	–	1,223.0	–	1,319.0
Security level 3														
Saber	HW	HS	250	45,895	18,705	–	0	2	4,320	17.3	5,231	20.9	6,461	25.8
Saber	HW	HS	250	25,079	10,750	–	0	2	5,435	21.8	6,618	26.5	8,034	32.1
R5ND_3KEM_0d	SW/HW	HS	249	73,881	109,211	14,307	0	2	–	–	–	24.0	–	33.0
Saber	SW/HW	HS	322	12,566	11,619	1,993	256	3.5	–	–	–	60.0	–	65.0
FrodoKEM-976	SW/HW	HS	402	7087	6693	1190	32	17	–	–	–	1,642.0	–	1,866.0
Security level 5														
R5ND_5KEM_0d	SW/HW	HS	212	91,166	151,019	18,733	0	2	–	–	–	32.0	–	42.0
NewHope-1024^{cpa}	HW	HS	406	13,961	8,149	–	25	18	–	–	34,000	83.0	30,600^{KD}	75.0^{KD}
FireSaber	SW/HW	HS	322	12,555	11,881	2,341	256	3.5	–	–	–	74.0	–	80.0
FrodoKEM-1344	SW/HW	HS	417	7,015	6,610	1,215	32	17.5	–	–	–	2,186.0	–	3,120.0

Table 12 – Digital Signature Schemes on Artix-7, Kintex-7 and Virtex-7

Algorithm	Type	Target	Max. Freq.	LUT	FF	Slice	DSP	BRAM	Key Generation		Signature Verification		Signature Generation		Family
									cycles	us	cycles	us	cycles	us	
Security Level 1 & 2															
Picnic-L1-FS	HW	HS	91	90,535	23,516	25,160	0	52.5	–	–	29,600	325.6	31,300	344.3	
qTESLA-I [o2]	SW/HW	LW	25*	14,975	2,539	4,173	11	14.0	4,846,949	193,878.0	38,922	1,556.9	168,273	6,730.9	
Dilithium-I	SW/HW	LW	25*	14,975	2,539	4,173	11	14.0	95,202	3,808.1	142,576	5,703.0	376,392	15,055.7	Artix-7
Dilithium-II	SW/HW	LW	25*	14,975	2,539	4,173	11	14.0	130,022	5,200.9	184,933	7,397.3	514,246	20,569.8	
qTESLA-p-I	SW/HW	LW	121	7,212	4,378	2,438	15	139.0	925,431	7,648.2	946,520	7,822.5	4,165,160	34,422.8	
Rainbow-Ic [o1]	HW	HS	90	52,895	32,476	15,112	0	67.0	–	–	–	–	979	10.9	
Rainbow-Ia	HW	HS	111	27,712	27,679	8,939	0	59.0	–	–	–	–	1,980	17.8	Kintex-7
Picnic-L1-FS	HW	HS	125	90,037	23,105	–	0	52.5	–	–	29,600	237.0	31,300	250.0	
Rainbow-Ic [o1]	HW	HS	167	52,721	32,475	15,976	0	67.0	–	–	–	–	979	5.9	Virtex-7
Rainbow-Ia	HW	HS	181	27,556	27,675	7,065	0	59.0	–	–	–	–	1,980	10.9	
Security Level 3															
qTesla-III-speed [o2]	SW/HW	LW	25*	14,975	2,539	4,173	11	14.0	11,898,241	475,929.6	67,712	2,708.5	317,083	12,683.3	
qTesla-III-size [o2]	SW/HW	LW	25*	14,975	2,539	4,173	11	14.0	11,479,190	459,167.6	69,154	2,766.2	348,429	13,937.2	Artix-7
Dilithium-III	SW/HW	LW	25*	14,975	2,539	4,173	11	14.0	167,433	6,697.3	229,481	9,179.2	634,763	25,390.5	
qTESLA-p-III	SW/HW	LW	121	7,475	4,518	2,473	15	147.0	2,305,220	19,051.4	2,315,950	19,140.1	7,745,088	64,009.0	
Security Level 4 & 5															
Picnic-L5-FS	HW	HS	125	167,530	33,164	–	0	98.5	–	–	146,600	1,173.0	154,500	1,236.0	Kintex-7
Dilithium-IV	SW/HW	LW	25*	14,975	2,539	4,173	11	14.0	223,272	8,930.9	276,221	11,048.8	815,636	32,625.4	Artix-7

For security level 3, no implementation of Picnic is available. The implementations of Dilithium-III and qTESLA-p-III are comparable in terms of type, target, and resource utilization. At the same time, the implementation of Dilithium is an order of magnitude more efficient. The implementations of digital signature schemes targeting Kintex-7 and Virtex-7 are summarized in the same table. For the Kintex-7 implementations, Rainbow substantially outperforms Picnic at the security level 1. For all remaining families and security levels, only one candidate with the up-to-date parameter set is reported.

7 Conclusions

In this paper, we first reviewed the previous work on hardware and software/hardware implementations of Round 2 PQC schemes. Out of 26 candidates, six – NewHope, CRYSTALS-Kyber, FrodoKEM, Saber, Round5, and SIKE – received the highest coverage in terms of the number of implementations and related publications. All of them have high-speed and simplified implementations reported. Applied software/hardware co-design to high-speed rather than lightweight implementations, which led to the choice of Xilinx Zynq UltraScale+, a state-of-the-art SoC FPGA family, as our primary platform. What matters is that this platform includes a «hardwired» ARM Cortex-A53 processor operating at the frequency of 1.2 GHz and a significant amount of programmable logic supporting hardware accelerators operating at the clock frequencies up to 500 MHz.

For each candidate, an attempt was made to offload as many as possible operations to hardware. For 50% of investigated KEMs, this percentage reached 100%. Thus, the corresponding implementations could be treated as hardware implementations, assuming that a random seed (*of 16, 24, or 32 bytes*) was transferred to the hardware module during encapsulation. KEMs implemented using this approach included Kyber, LAC (v3a and v3b), NewHope, and Round5 (*with and without error-correcting code*). Their code was benchmarked using Artix-7 and Virtex-7 FPGAs.

In terms of both the execution times and resource utilization, Round5 with an error-correcting code (R5ND_5d) outperformed Round5 without an error-correcting code (R5ND_0d). Similarly, LAC-v3b appeared superior over LAC-v3a in terms of both speed and use of FPGA resources. Then, when the best representatives of four candidates – Kyber, LAC, NewHope, and Round5 – were compared, the following conclusions could be drawn. The execution times of these candidates were extremely close to one another. For encapsulation, the execution times were within 10% from one another at the security level 5, within 22% at the security level 3, and within 32% at the security level 1. For decapsulation, the largest differences were 26% at level 5, 22% at level 3, and 48% at level 1. In multiple instances, just a change of an FPGA family from low-cost Artix-7 to high-performance Virtex-7 caused a significant change in the rankings, even though the HDL code remained exactly the same. As a result, we must conclude that the differences among these candidates in terms of speed are too small to give preference to any particular candidate. These results contradict one of the earlier reports placing LAC well behind NewHope and Kyber.

In terms of resource utilization, a small advantage belongs to NewHope and Kyber. Both of them use fewer LUTs and flip-flops than LAC and Round5, and their use of DSP units and BRAMs, although slightly higher, is very moderate. Additionally, both NewHope and Kyber use almost the same amount of resources independently of the security level. In the case of both LAC and Round5, resource usage increases sharply with the increase in security level. The former property appears to be an advantage for applications requiring support for the highest or all security levels. In particular, the k-in-1 designs, which support all k security levels and allow modifying them at run time, typically have only slightly higher resource utilization than that for the maximum security level. Thus, the flat dependence of the resource utilization on the security level implies a potential for very cost-effective k-in-1 designs. At the same time, this potential should still be confirmed through complete designs.

A detailed characterization of the FPGA Xilinx family was also provided. Each particular FPGA should be used based on purpose, expected cost, and performance.

References

[1] J.-S. Coron, A. Joux, Cryptanalysis of a provably secure cryptographic hash function, Cryptology ePrint Archive Report 2004/013, 2004. http://eprint.iacr.org/2004/013

[2] Post-quantum cryptography, round 2 submissions. [Електронний ресурс]. – Режим доступу: https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[3] Malik Imran A Systematic Study of Lattice-based NIST PQC Algorithms: from Reference Implementations to Hardware Accelerators / Malik Imran, Zain Ul Abideen, Samuel Pagliarini //. – Режим доступу: https://arxiv.org/pdf/2009.07091.pdf.

[4] Viet Ba Dang Implementation and Benchmarking of Round 2 Candidates in the NIST Post-Quantum Cryptography Standardization Process Using Hardware and Software/Hardware Co-design Approaches / Viet Ba Dang, Farnoud Farahmand, Michal Andrzejczak, Kamyar Mohajerani, Duc Tri Nguyen, Kris Gaj //. – Режим доступу: https://eprint.iacr.org/2020/795.pdf.

[5] И. Тарасов ПЛИС Xilinx и Цифровая обработка Сигналов Особенности, преимущества, перспективы. – Режим доступу: https://www.electronics.ru/files/article_pdf/2/article_2788_434.pdf.

[6] Xilinx. 7 Series Product Selection Guide. [Електронний ресурс]. – Режим доступу: https://www.xilinx.com/support/documentation/selection-guides/7-series-product-selection-guide.pdf.

[7] Farnoud Farahmand et al. Software/Hardware Codesign of the Post Quantum Cryptography Algorithm NTRUEncrypt Using High-Level Synthesis and Register-Transfer Level Design Methodologies. In: 29th International Conference on Field Programmable Logic and Applications, FPL 2019. Barcelona, Spain: IEEE, Sept. 2019, pp. 225–231. ISBN: 978-1-72814-884-7. DOI: 10.1109/FPL.2019.00042.

[8] Kris Gaj Challenges and Rewards of Implementing and Benchmarking Post-Quantum Cryptography in Hardware. In: 2018 Great Lakes Symposium on VLSI, GLSVLSI 2018. Chicago, IL, USA: ACM Press, 2018, pp. 359–364. ISBN: 978-1-4503-5724-1. DOI: 10/ggbscs.

[9] Jens-Peter Kaps et al. Lightweight Implementations of SHA-3 Candidates on FPGAs. In: 12th International Conference on Cryptology in India, Indocrypt 2011. Vol. 7107. LNCS. Chennai, India, Dec. 2011, pp. 270–289. ISBN: 978-3-642-25577-9 978-3-642-25578-6. DOI: 10.1007/978-3-642-25578-6_20. – Режим доступу: https://2011.indocrypt.org/slides/gurung.pdf.

[10] Viet B Dang et al. Implementing and Benchmarking Three Lattice-Based Post-Quantum Cryptography Algorithms Using Software/Hardware Codesign. In: 2019 International Conference on Field Programmable Technology, FPT 2019. Tianjin, China: IEEE, Dec. 9-13, 2019, pp. 206–214. DOI: 10.1109/ICFPT47387.2019.00032.

Рецензент: Сергій Толюпа, д.т.н., проф., Київський національний університет імені Т. Шевченка, м. Київ, Україна.
E-mail: tolupa@i.ua

Надійшло: Грудень 2020.

Автори:

Марина Єсіна, к.т.н., доцент кафедри безпеки інформаційних систем і технологій, Харківський національний університет імені В. Н. Каразіна, майдан Свободи 6, м. Харків, 61022, Україна.
E-mail: m.v.yesina@karazin.ua

Богдан Шахов, студент кафедри безпеки інформаційних систем і технологій, Харківський національний університет імені В.Н. Каразіна, майдан Свободи 4, м. Харків, 61022, Україна.
E-mail: bogdanshahov2000@gmail.com

Дослідження реалізацій кандидатів другого раунду конкурсу NIST PQC, що орієнтовані на сімейства FPGA Xilinx.

Анотація. Сьогодні досить гостро постає питання щодо стійкості сучасних існуючих криптографічних механізмів до квантових алгоритмів криптоаналізу зокрема та квантових комп'ютерів взагалі. Ця проблема активно обговорюється на міжнародному рівні. Тому, задля її вирішення, NIST США вирішив організувати та проводить на сьогоднішній день конкурс на кандидатів на постквантові криптографічні алгоритми NIST PQC. Результатом конкурсу повинне стати прийняття до стандартизації криптографічних алгоритмів різного типу – асиметричне шифрування, інкапсуляція ключів та електронний підпис (як мінімум по одному алгоритму з кожного типу). На момент початку конкурсу на процес стандартизації було представлено 82 алгоритми. На основі критеріїв мінімальної прийнятності, визначених NIST, для 1-го раунду було розглянуто 69 алгоритмів. Враховуючи декілька параметрів – безпеку, вартість, продуктивність, характеристики реалізації тощо, 43 і 11 алгоритмів були виключені при завершенні 1-го і 2-го раундів відповідно, а інші 15 алгоритмів були збережені для 3-го раунду. Алгоритми, які залишилися у 2-му раунді можна розділити на 5 різних категорій залежно від математичного базису, на якому вони засновуються: на основі ізогеній еліптичних кривих, на основі алгебраїчних решіток, на основі математичного коду, на основі багатовимірних перетворень і на основі геш-функцій. Безпека є основним критерієм оцінки, що визначає конкуренцію в конкурсі NIST, і, зрозуміло, що реалізації програмного забезпечення кандидатів в основному зосереджені на ній. Однак, вкрай важливо аби алгоритм мав й ефективну апаратну реалізацію. А своєчасне виявлення апаратної неефективності допоможе сконцентрувати зусилля криптографічної спільноти на більш перспективних кандидатах, потенційно заощадивши велику кількість часу, що може бути витрачена на криптоаналіз. У даній роботі розглядаються та порівнюються між собою FPGA сімейства Xilinx. Наводяться та порівнюються між собою дані щодо реалізацій кандидатів 2-го раунду в процесі стандартизації постквантової криптографії NIST, що орієнтовані на FPGA сімейства Xilinx.

Ключові слова: електронний підпис; постквантова криптографія; конкурс NIST PQC; FPGA, Xilinx.

Рецензент: Сергей Толюпа, д.т.н., проф., Киевский национальный университет имени Т. Шевченко, г. Киев, Украина.
E-mail: tolupa@i.ua

Поступила: Декабрь 2020.

Авторы:
Марина Есина, к.т.н., доцент кафедры безопасности информационных систем и технологий, ХНУ им. В. Н. Каразина, пл. Свободы 6, Харьков, 61022, Украина.
E-mail: m.v.yesina@karazin.ua
Богдан Шахов, студент факультета компьютерных наук, ХНУ им. В. Н. Каразина, пл. Свободы, 4, Харьков, 61022, Украина.
E-mail: bogdanshahov2000@gmail.com

Исследование реализаций кандидатов второго раунда конкурса NIST PQC, ориентированных на семейства FPGA Xilinx.

Аннотация. Сегодня достаточно остро стоит вопрос о стойкости современных существующих криптографических механизмов к квантовым алгоритмам криптоанализа в частности и квантовым компьютерам вообще. Эта проблема активно обсуждается на международном уровне. Поэтому, для ее решения, NIST США решил организовать и проводит на сегодняшний день конкурс на кандидатов на постквантовые криптографические алгоритмы NIST PQC. Результатом конкурса должно стать принятие к стандартизации криптографических алгоритмов разного типа – асимметричное шифрование, инкапсуляция ключей и электронная подпись (как минимум по одному алгоритму с каждого типа). На момент начала конкурса на процесс стандартизации было представлено 82 алгоритмы. На основе критериев минимальной приемлемости, определенных NIST, для 1-го раунда было рассмотрено 69 алгоритмов. Учитывая несколько параметров – безопасность, стоимость, производительность, характеристики реализации и т.п., 43 и 11 алгоритмов были исключены при завершении 1-го и 2-го раундов соответственно, а остальные 15 алгоритмов были сохранены для 3-го раунда. Алгоритмы, которые остались во 2-м раунде можно разделить на 5 различных категорий в зависимости от математического базиса, на котором они основываются: на основе изогений эллиптических кривых, на основе алгебраических решеток, на основе математического кода, на основе многомерных преобразований и на основе хеш-функций. Безопасность является основным критерием оценки, определяет конкуренцию в конкурсе NIST, и, понятно, что реализации программного обеспечения кандидатов в основном сосредоточены на ней. Однако, крайне важно, чтобы алгоритм имел и эффективную аппаратную реализацию. А своевременное выявление аппаратной неэффективности поможет сконцентрировать усилия криптографического сообщества на более перспективных кандидатах, потенциально сэкономив большое количество времени, которое может быть потрачено на криптоанализ. В данной работе рассматриваются и сравниваются между собой FPGA семейства Xilinx. Приводятся и сравниваются между собой данные по реализаций кандидатов 2-го раунда в процессе стандартизации постквантовой криптографии NIST, ориентированные на FPGA семейства Xilinx.

Ключевые слова: электронная подпись; постквантовая криптография; конкурс NIST PQC; FPGA, Xilinx.

УДК 004.056.55 DOI: 10.26565/2519-2310-2021-1-03

ПОРІВНЯЛЬНИЙ АНАЛІЗ ТА ДОСЛІДЖЕННЯ ВЛАСТИВОСТЕЙ НОСІЇВ ІНФОРМАЦІЇ ДЛЯ СТЕГАНОГРАФІЧНОГО ПРИХОВУВАННЯ ДАНИХ В КЛАСТЕРНИХ ФАЙЛОВИХ СИСТЕМАХ

Кирил Шеханін, Людмила Горбачова, Катерина Кузнецова

Харківський національний університет імені В.Н. Каразіна, Харків, 61022, Україна
lusyag23@gmail.com, kate7smith12@gmail.com

Рецензент: Микола Карпінський, д.т.н., проф., університет Бельсько-Бяла, Бельсько-Бяла, Польща.
mkarpinski@ath.bielsko.pl

Надійшла: Грудень 2020.

Анотація: *У роботі розглянуто сучасні технології збереження інформації, а саме HDD, Flash-USB, SSD. Проаналізовано такі показники, як кількість реалізованих виробів, ціна, швидкість зчитування та запису. Досліджено деякі показники ефективності носіїв інформації, з точки зору можливості застосування стеганографічних методів приховування інформації у кластерних файлових системах. Виконано аналіз швидкості послідовного зчитування/запису та швидкості доступу до випадкового кластеру, що відповідає швидкості доступу до фрагментованного файлу. Для цього використовувалися результати тестувань з ресурсу UserBenchmark. Тестування виконувалися методами Sequential та Random4k. Запропонована оцінка носіїв інформації і надані рекомендації, стосовно використання носія інформації та методу приховування даних шляхом перемішування кластерів у структурі файлової системи. Проведені аналіз взаємозв'язку і залежності параметрів швидкості доступу до кластеру, та рівня фрагментованості файлу. Уточнено, яким чином збільшення чи зменшення рівня фрагментованості (переплетеності) впливає на швидкість доступу до файлу, що є доволі важливим показником при використанні методу приховування даних у структурі файлової системі. Розглянуті переваги і недоліки накопичувачів інформації різних типів, та проведено їх порівняльний аналіз. Виконано, аналіз особливостей процесу дефрагментації накопичувачів, та вплив різних факторів на загальний рівень фрагментованості на носії інформації. Підкреслено, що чим більший рівень фрагментованості на носії даних, тим більше інформації можна приховати. Стверджується, що завдяки широкому розповсюдженню SSD/HDD накопичувачів, метод приховування інформації у структуру файлових систем, шляхом перемішування кластерів покриваючих файлів, є актуальним.*

Ключові слова: *файлові носії інформації; фрагментація; швидкість доступу; приховування даних; стеганографія.*

1 Вступ

Із розвитком інформаційних технологій питання зберігання, обміну і обробки інформації, як і раніше, не втрачають своєї актуальності. При цьому на передній план виходять напрямки, що пов'язані з процесами зменшення експлуатаційних витрат, підвищення швидкодії та якості відповідних онлайн сервісів та служб. Як свідчить практика, в багатьох випадках інформація повинна бути захищена різними криптографічними методами, а доступ до неї забезпечуватися, тільки авторизованим користувачам. Але у разі використання криптографічних методів потенційним зловмисникам відомо про факт збереження або передачі інформації [1–3]. Це, у подальшому, може призвести до розкриття змісту чутливої інформації шляхом використання методів криптоаналізу. Суттєво ускладнити потенційним зловмисникам процес доступу чутливих даних можна за рахунок додаткового використання різних стеганографічних методів [4–6]. Використання стеганографії доповнює криптографічні методи, тож ці методи є компліментарними один до одного з точки зору захищеності інформації. В цьому сенсі, як приклад, можна згадати сумісне використання шифру Цезаря (криптографічний метод) і «невидимих» чорнил (стеганографічний метод) [4, 5, 7]. А отже, якщо сумісно використати ці два методи, то зловмиснику необхідно буде спочатку визначити і підтвердити факт використання прихованого тексту, що написаний невидимими чорнилами, і тільки вже потім зробити спробу безпосередньо дешифрувати використовуваний шифр.

Звісно у сучасності данні методи вже не є ефективними, а більшої популярності набули методи комп'ютерної, або цифрової стеганографії [4, 5, 7–10], де інформаційні повідомлення подаються у вигляді цифрових даних, що приховуються в інших цифрових даних. Так звані

покрівельні файли, які маскують факт інкапсуляції прихованого вмісту, передаються та зберігаються у відкритому вигляді. Це можуть бути, наприклад, цифрові зображення, які у великій кількості передаються сучасними каналами електронної пошти. При цьому вповноважений отримувач, який має секретний ключ дешифрування, приймає покрівельний файл та може відновити таємне повідомлення. Звісно, що при цьому приховується сам факт існування прихованого повідомлення, а відслідкувати та дослідити всі покрівельні (*тобто, маскуючі*) файли у в всесвітньої мережі фізично неможливо [9–11]. Нарізі стеганографічні методи постійно вдосконалюються та розвиваються. Основна вимога до покрівельних файлів є їхня надмірність (збитковість). Наприклад, надмірність цифрових зображень дозволяє приховати досить великі за обсягом інформаційні повідомлення.

Останніми роками набули розвитку методи технічної стеганографії. В таких системах приховування інформації досягається шляхом використання властивостей, які штучно зроблено людиною при побудові різних технічних засобів. Як приклад, можна навести мережеву стеганографію [12–16], і якій застосовуються різні особливості побудови сучасних телекомунікаційних систем та мереж, в тому числі штучна надмірність при визначенні форматів пакетів даних на способів їхньої передачі [17–19]. В 3D стеганографії використовуються надмірності цифрових 3D моделей та створених за їх допомогою фізичних об’єктів [20–24]. Наприклад, у роботах [25, 26] запропоновано приховувати інформацію шляхом створення фізичних об’єктів всередині інших об'єктів.

Ще одним прикладом технічної стеганографії є застосування особливостей побудови кластерних файлових систем. Зокрема, у роботах [27–29] було запропоновано методи, які дозволяють ефективно приховувати інформацію шляхом зміни чергування окремих кластерів т.з. покрівельних файлів. Імена (назви) таких файлів є ключовою інформацією і відновити приховуване повідомлення без посилань (тобто без назв) покрівельних файлів вкрай важко. Подальші дослідження кластерних файлових систем [30–36] дозволяють впевнено стверджувати, що застосування особливостей організації зберігання інформації дозволяє отримати надійний та безпечних механізм стеганографічного приховування інформаційних повідомлень. Звісно, що стійкість та швидкодія кластерних стеганосистем безпосередньо спирається на конкретні властивості файлової системи. Зокрема, безпека спирається на фрагментарність та переплетеність (в термінах робіт [27, 27, 30, 33]) покрівельних файлів, а швидкодія залежить від фрагментарності файлової системи, та характеристик конкретного носія інформації. Отже актуальним є питання аналізу різних носіїв інформації і відповідних файлових систем на предмет їх можливого застосування в стеганографічних методах приховування, та дослідження впливу окремих показників на ефективність кластерних стеганосистем.

Метою даної роботи є аналіз відомих технологій та методів зберігання інформації та дослідження властивостей фрагментарності файлової системи і переплетеності окремих файлів, на швидкість процесів запису/зчитуванні даних. Практичне значення роботи обумовлено визначенням можливих перспектив використання досліджених процедур для цілей побудови кластерних стеганосистем.

2 Техніки приховування інформації у кластерні файлові системи

Файлова система встановлює порядок, спосіб організації, зберігання та іменування даних на носіях інформації в інформаційних системах, а також в іншому електронному обладнанні: цифрових фотоапаратах, мобільних телефонах тощо [37–39]. Файлова система визначає формат вмісту і спосіб фізичного зберігання інформації, яка групується у файли. Конкретна файлова система визначає розмір імен файлів і (каталогів), max можливий розмір файлу і розділу, та атрибути файлу, тобто визначає метадані файлів. Деякі файлові системи надають сервісні можливості, наприклад, розмежування доступу або шифрування файлів.

Найпростіші методи приховування інформації у структуру файлових систем розглянуто в [32, 34–36]. Дані методи застосовують вільні кластери (*службові поля даних*) для запису прихованої інформації, але такий спосіб, з точки зору конфіденційності інформації, є ненадійним [27, 29]. Інші методи, наприклад [27–31], ґрунтуються на використанні покриваючих

файлів і приховуванні інформаційних даних за допомогою взаємного перемішування файлів, тобто зміни відносних позицій кластерів декількох покриваючих файлів один до одного.

Приховування інформації через перемішування кластерів різних покриваючих файлів.

Приховування інформації через перемішування кластерів різних покриваючих файлів досліджено у роботах [27, 29]. Приховувана інформація представляється у вигляді бітового масиву: $M=\{b_0,b_1,...,b_{n-1}\}$, $b_i\in\{0,1\}$. На інформаційному носії обирають $p=2^m$, $m\in N$ покриваючих файлів: $F_0,F_1,...,F_{p-1}$. Порядок кластерів покриваючих файлів приховує інформаційне повідомлення, тобто після вбудовування покриваючи файли не можна модифікувати, видаляти та переміщувати. Натуральне число m та імена покриваючих файлів є секретним ключем. Важливим є, також, порядок упорядкування покриваючих файлів [29].

Формується масив номерів кластерів покриваючих файлів:

$$C=\begin{pmatrix} c_{0,0} & c_{0,1} & ... & c_{0,L_0-1} & & & \\ c_{1,0} & c_{1,1} & ... & ... & ... & ... & c_{1,L_1-1} \\ ... & ... & ... & ... & ... & ... & ... \\ c_{p-1,0} & c_{p-1,1} & ... & ... & c_{p-1,L_{p-1}-1} & & \end{pmatrix},$$

де кожен рядок масиву містить номера кластерів відповідного файлу. Наприклад, файлу F_i відповідає i-й рядок масиву C, тобто номера кластерів i-го покриваючого файлу можуть бути подані у вигляді масиву $C_{F_i}=\{c_{i,0},c_{i,1},...,c_{i,L_i-1}\}$, де L_i - число кластерів в i-у покриваючому файлі. Якщо при приховуванні інформації необхідно зберегти без зміни вміст покриваючих файлів, тоді потрібно, щоб виконувалися умови: $\forall i: L_i\ge k$, $k=n/m$.

Формується масив D номерів вільних кластерів файлової системи: $D=\{c_1,c_2,...,c_{L_D}\}$, так щоб виконувалася умова $c_1<c_2<...<c_{L_D}$. Число L_D дорівнює кількості вільних кластерів файлової системи, причому потрібно, щоб виконувалася умова: $L_D\ge\sum_{i=0}^{p-1}L_i$.

Інформаційне повідомлення M розбивається на блоки по m бітів: $M=\{B_1,B_2,...,B_k\}$, де $k=\lceil n/m\rceil$ та якщо $k=n/m$, то $B_1=\{b_0,b_1,...,b_{m-1}\}$, $B_2=\{b_m,b_{m+1},...,b_{2m-1}\}$, …, $B_k=\{b_{(k-1)m},b_{(k-1)(m+1)},...,b_{km-1}\}$. Якщо $k<n/m$, то останній блок необхідно дописати неінформаційними значеннями, наприклад, нулями $B_1=\{b_0,b_1,...,b_{m-1}\}$, $B_2=\{b_m,b_{m+1},...,b_{2m-1}\}$, …, $B_k=\{b_{(k-1)m},b_{(k-1)(m+1)},...,b_{n-1},\underbrace{0,0,...,0}_{km-n}\}$. Кожен блок B_i, $i=1,2,...,k$ подається як натуральне число, тобто $\forall i: 0\le B_i\le p-1$. Кожне натуральне число B_i, $i=1,2,...,k$ подається як номер покриваючого файлу з множини файлів $F_0,F_1,...,F_{p-1}$.

Всі кластери покриваючих файлів перезаписуються у вільні кластери файлової системи, тобто масив D заповнюється номерами кластерів із масиву C. Порядок перезапису кластерів покриваючих файлів відповідає послідовності натуральних чисел $\{B_1,B_2,...,B_k\}$, що задаються повідомленням, яке приховується. Наприклад, у перший вільний кластер записуємо 1-й кластер покриваючого файлу із номером B_1, у 2-й вільний кластер – наступний кластер покриваючого файлу із номером B_2 і так далі.

Натуральні числа B_i можуть співпадати і в цьому разі записуємо наступний кластер того ж самого покриваючого файлу із номером B_i. Для посилення захисту від детектування та декодування стеганосистеми додатково можуть застосовуватися певні механізми, наприклад [29], обирається ключ ініціалізації B_0, а порядок перезапису кластерів покриваючих файлів задається послідовністю натуральних чисел $\{N_1,N_2,...,N_k\}$, $N_i=B_{i-1}+B_i \bmod p$, $0\le N_i\le p-1$. Тоді, у 1-й вільний кластер перезаписуються 1-й кластер покриваючого файлу із номером N_1, а у 2-й вільним кластер – черговий кластер покриваючого файлу із номером N_2 і т.д.

В результаті виконання алгоритму перші k вільних кластерів файлової системи будуть записані кластерами покриваючих файлів. Отже повинна виконуватися умова $k \le L_D$.

Для вилучення прихованого повідомлення M формується масив D номерів кластерів покриваючих файлів: $D = \{c_1, c_2, ..., c_{L_D}\}$, причому необхідно, щоб виконувалась умова $c_1 < c_2 < ... < c_{L_D}$. Кожен номер кластеру з цього масиву співвідноситися тільки з одним кластером одного покриваючого файлу. Саме така відповідність пов'язана із логікою вбудовування інформації і використовується для вилучення прихованої інформації. Формується послідовність натуральних чисел $\{B_1, B_2, ..., B_k\}$, які відповідають блокам прихованого повідомлення: $B_1 = \{b_0, b_1, ..., b_{m-1}\}$, $B_2 = \{b_m, b_{m+1}, ..., b_{2m-1}\}$, …, $B_k = \{b_{(k-1)m}, b_{(k-1)(m+1)}, ..., b_{km-1}\}$. З цих бітових блоків формується інформаційне повідомлення $M = \{b_0, b_1, ..., b_{n-1}\}$, $b_i \in \{0,1\}$. Якщо $k < n/m$, тоді останній блок «обрізується» - його останні $km - n$ біт не несуть інформаційного значення.

Недоліком даного методу є незначний обсяг розміру прихованої інформації, який залежить від кількості покриваючих файлів та розміру покриваючих файлів у кластерах. Так, у кожному кластері покриваючих файлів може бути приховано $\log_2 p = m$ інформаційних біт.

Подальший розвиток розглянутого методу наведено у роботах [30, 31], де для збільшення обсягу прихованого повідомлення, в модифікованому методі, запропоновано додаткове перемішування кластерів кожного покриваючого файлу.

Модифікований метод приховування інформації в кластерних файлових системах ґрунтується на використанні одного або декількох покриваючих файлів і приховуванні інформаційного повідомлення шляхом зміни порядку позицій кластерів різних покриваючих файлів, та зміні чергування кластерів у межах одного покриваючого файлу. Даний метод дозволяє досягти збільшення прихованої інформації на один біт для кожного кластеру, при одних і тих значеннях ключових параметрів. Приховувана інформація подається у вигляді бітового масиву: $M = \{b^*_0, b^*_1, ..., b^*_{L_1+L_2+...+L_{p-1}-1}, b_0, b_1, ..., b_{n-1}\}$, $b^*_i, b_i \in \{0,1\}$. На носії інформації обирається $p = 2^m$, $m \in N$ покриваючих файлів: $F_0, F_1, ..., F_{p-1}$. Після цього формується масив номерів кластерів покриваючих файлів, як в вище описаному методі.

В даному разі для кожного покриваючого файлу змінюється порядок «чергування» кластерів у кожному покриваючому файлі. Порядок «чергування» задається відповідною послідовністю M. Для цього, формується p бітових масивів інформаційних бітів:

$$M_1 = \{b^*_0, b^*_1, ..., b^*_{L_1-1}\},$$
$$M_2 = \{b^*_{L_1}, b^*_{L_1+1}, ..., b^*_{L_1+L_2-1}\},$$
$$\dots,$$
$$M_{L_p} = \{b^*_{L_1+L_2+...+L_{p-1}}, b^*_{L_1+L_2+...+L_{p-1}+1}, ..., b^*_{L_1+L_2+...+L_{p-1}-1}\},$$

кожен з яких, зіставляється із масивом номерів кластерів покриваючих файлів

$$C_{F_1} = \{c_{1,0}, c_{1,1}, ..., c_{1,L_1-1}\},$$
$$C_{F_2} = \{c_{2,0}, c_{2,1}, ..., c_{2,L_2-1}\},$$
$$\dots,$$
$$C_{F_p} = \{c_{p,0}, c_{p,1}, ..., c_{p,L_p-1}\}.$$

Позиції кластерів кожного покриваючого файлу перезаписуються, тобто номери кластерів у кожному з масивів C_{F_1}, C_{F_2}, … , C_{F_p} змінюють своє чергування, відповідно до значень бітових масивів M_1, M_2, … , M_{L_p}. У результаті отримують нові масиви номерів кластерів $C^*_{F_1}$, $C^*_{F_2}$, … , $C^*_{F_p}$.

Перезапис позицій кластерів у кожному покриваючому файлі може здійснюватися різними способами. Наприклад, шляхом розбиття всіх номерів позицій кластерів $\{c_{i,0}, c_{i,1}, ..., c_{i,L_i-1}\}$ на дві половини, та співставлення кожної половини із значенням інформа-

ційного біту. Тоді, наприклад, якщо $b_j^*=1$, $L_1+L_2+...+L_{i-1}-1<j\le L_1+L_2+...+L_i-1$, на j-у позицію в масиві $C_{F_i}^*$ розміщують кластер з 1-ої половини впорядкованих номерів, якщо $b_0^*=0$ - з другої половини номерів.

Сформовані таким чином масиви $C_{F_i}^*=\{c_{i,0}^*,c_{i,1}^*,...,c_{i,L_i-1}^*\}$ перезаписаних позицій номерів покриваючих файлів утворюють масив

$$C^*=\begin{pmatrix} c_{0,0}^* & c_{0,1}^* & ... & c_{0,L_0-1}^* & & & \\ c_{1,0}^* & c_{1,1}^* & ... & ... & ... & ... & c_{1,L_1-1}^* \\ ... & ... & ... & ... & ... & ... & ... \\ c_{p-1,0}^* & c_{p-1,1}^* & ... & ... & c_{p-1,L_{p-1}-1}^* & & \end{pmatrix}.$$

Зміна порядку чергування кластерів в кожному покриваючому фалі дозволяє приховати перші $L_1+L_2+...+L_{p-1}$ інформаційних бітів з масиву M, тобто інформаційну послідовність $\{b_0^*,b_1^*,...,b_{L_1+L_2+...+L_{p-1}-1}^*\}$. Решту n інформаційних бітів необхідно приховати так само, як і у розглянутому вище методі [29].

Формується масив D номерів вільних кластерів файлової системи: $D=\{c_1,c_2,...,c_{L_D}\}$, $c_1<c_2<...<c_{L_D}$. Послідовність із інформаційних бітів $\{b_0,b_1,...,b_{n-1}\}$ розбивається на блоки по m бітів кожен: $\{B_1,B_2,...,B_k\}$, кожен блок B_i, $i=1,2,...,k$ подається як натуральне число, тобто $\forall i: 0\le B_i\le p-1$. Кожне натуральне число B_i, $i=1,2,...,k$ подається як номер покриваючого файлу з множини файлів $F_0,F_1,...,F_{p-1}$.

Номери позицій кластерів покриваючих файлів перезаписуються у вільні кластери, тобто масив D заповнюється номерами кластерів з масиву C^* (*перезаписаними кластерами, тобто із зміненим чергуванням кластерів у кожному покриваючому файлі*). При цьому, порядок перезапису кластерів покриваючих файлів відповідає послідовності натуральних чисел $\{B_1,B_2,...,B_k\}$, які задаються приховуваним повідомленням. Наприклад, у перший порожній кластер перезаписуються 1-й кластер покриваючого файлу із номером B_1, у другий порожній кластер – черговий кластер покриваючого файлу із номером B_2 і т.д. Натуральні числа B_i можуть співпадати і в цьому разі записуються чергові кластери того ж самого покриваючого файлу із номером B_i. В результаті перші k вільних кластерів файлової системи будуть записані кластерами покриваючих файлів.

Для вилучення прихованого повідомлення M формується масив D номерів позицій кластерів покриваючих файлів: $D=\{c_1,c_2,...,c_{L_D}\}$. Кожен номер кластеру з цього масиву співвідноситися тільки з одним кластером одного покриваючого файлу. При цьому формується послідовність натуральних чисел $\{B_1,B_2,...,B_k\}$, які відповідають блокам прихованого повідомлення, тобто формується інформаційна послідовність $\{b_0,b_1,...,b_{n-1}\}$, $b_i\in\{0,1\}$.

Далі вилучається інформаційна послідовність $\{b_0^*,b_1^*,...,b_{L_1+L_2+...+L_{p-1}-1}^*\}$, $b_i^*\in\{0,1\}$.

Для цього аналізуємо масиви $C_{F_i}^*=\{c_{i,0}^*,c_{i,1}^*,...,c_{i,L_i-1}^*\}$ номерів кластерів кожного покриваючого файлу. Правило вилучення інформації відповідає логіці приховування. Наприклад, може застосовуватися розбиття всіх впорядкованих позицій номерів $\{c_{i,0},c_{i,1},...,c_{i,L_i-1}\}$ на дві половини і співставлення кожної половини із значенням інформаційного біту. Тоді, якщо на j-й позиції в масиві $C_{F_i}^*$ розміщено кластер з 1-ї половини масиву впорядкованих номерів $\{c_{i,0},c_{i,1},...,c_{i,L_i-1}\}$, приймають бітове значення $b_j^*=1$. Якщо з другої половини, то приймають бітове значення $b_0^*=0$.

Таким чином, за рахунок додаткової зміни порядку чергування номерів кластерів у кожному із покриваючих файлів вдається збільшити обсяг прихованої інформації. Зокрема, в

порівнянні із способом-прототипом, додатково вдається приховати по одному інформаційному біту на кожен кластер покриваючого файлу.

3 Аналіз відомих технологій зберігання інформації

При підготовці матеріалів даної роботи було проаналізовано носії інформації, які використовують різні технології способу збереження інформації:

1. SSD (*Solid-State Drive*) – це енергонезалежний немеханічний запам'ятовуючий пристрій на основі мікросхем пам'яті та керуючому контролері
2. HDD (*Hard (magnetic) Disk Drive*) – це енергонезалежний запам'ятовуючий пристрій оснований на принципі магнітного запису на диск;
3. Flash-USB (*Universal Serial Bus*) – це енергонезалежний запам'ятовуючий пристрій на основі мікросхем, що має структуру подібну до SSD накопичувачів, але має менші показники місткості і швидкодії, тому здебільшого використовуються для перенесення інформації з пристрою на пристрій, та/або відокремленого збереження незначної кількості важливої інформації (паролі, ключи доступу та ін).

Так як Flash-USB є технологією подібною до SSD то переваги й недоліки у даних технології будуть схожі. Але, Flash-USB обмежені максимальним об'ємом зберігаємої інформації, а також інтерфейсом USB (1.5 Мбіт/с ÷ 5 Гбіт/с), що є значно повільнішим за SATA (1.5÷6 Гбіт/с) чи M.2 PCIe (8÷32 Гбіт/с), через які зазвичай підключають SSD накопичувачі до комп'ютерів. В цілому, порівнюючи переваги та недоліки різних SSD та HDD накопичувачів можна стверджувати, що SSD мають більше переваг ніж недоліків (табл. 1).

Таблиця 1 – Порівняння основних параметрів SSD та HDD пристроїв

Архітектура накопичувача	**SSD**	**HDD**
Рівень	Майже відсутній	Значний, так як є рухомі частини
Механічна стійкість	Висока	Низька (*при падінні та ударах рухомі частини можуть бути пошкодженні*)
Енергоспоживання	2-3 Ватт/годину	5-6 Ватт/годину
Магнітна чутливість	Майже відсутня	Значна (*електромагнітне поле впливає на магнітний диск*)
Розміри	Менші розміри та вага	Більші розміри через присутність рухомих частин
Паралельні операції	Підтримуються (*пришвидшує запис/зчитування декількох файлів*)	Відсутні
Швидкість запису/зчитування (Мб/с)	1000-3200/2000-4000	100-320/200-400
Ціна ($/Гб)	0.5	0.1
Циклів перезапису *(разів)*	10000	> 100000
Вплив фрагментації на роботу	Майже відсутній *(швидкість зчитування/запису не залежить від фрагментованості файлів)*	Значний

Дані цієї таблиці мають дещо приблизний характер через те, що кожен виробник має свої характеристики стосовно пристроїв SSD та HDD. Так, одна й та ж технологія, навіть у одного виробника, може мати різні показники у залежності від цінової категорії пристрою.

Головною метою відомостей табл. 1, було показати *загальну тенденцію* та *різницю* між SSD та HDD накопичувачами.

Варто зазначити, що з розвитком технологій SSD накопичувачі, у порівнянні з аналогами п'ятирічної давнини, мають значний приріст у показниках. У той час як показники HDD пристроїв майже не змінились. Так, якщо ще у 2015 році кількість HDD пристроїв значно переважала на ринку накопичувачів даних, то вже у 2020 році, кількість пристроїв HDD та SSD практично порівнялась. Зведена гістограма наведена на рис. 1 (*реалізація вироблених накопичувачів інформації по всьому світі, млн. одиниць*).

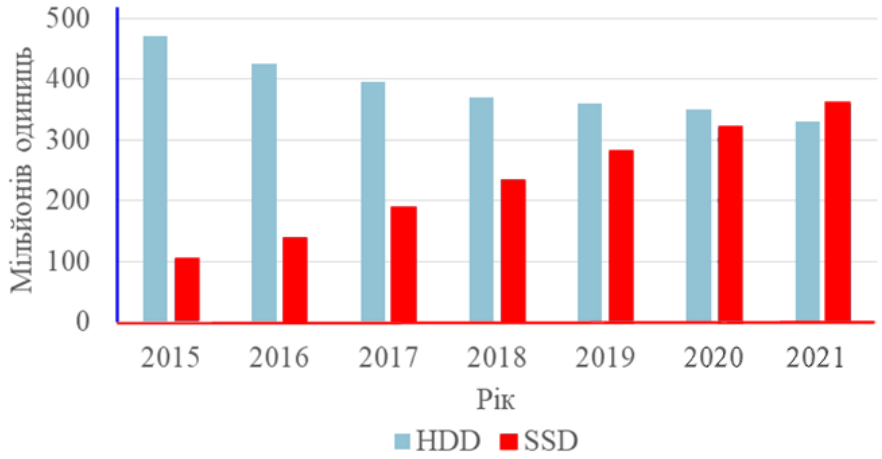


Рис. 1 – Показники реалізації вироблених накопичувачів інформації

Таким чином можна стверджувати, що з кожним роком кількість SSD накопичувачів росте, поглинаючи долю ринка HDD. Динаміка кількості HHD та SSD накопичувачів приведено на рис. 2.

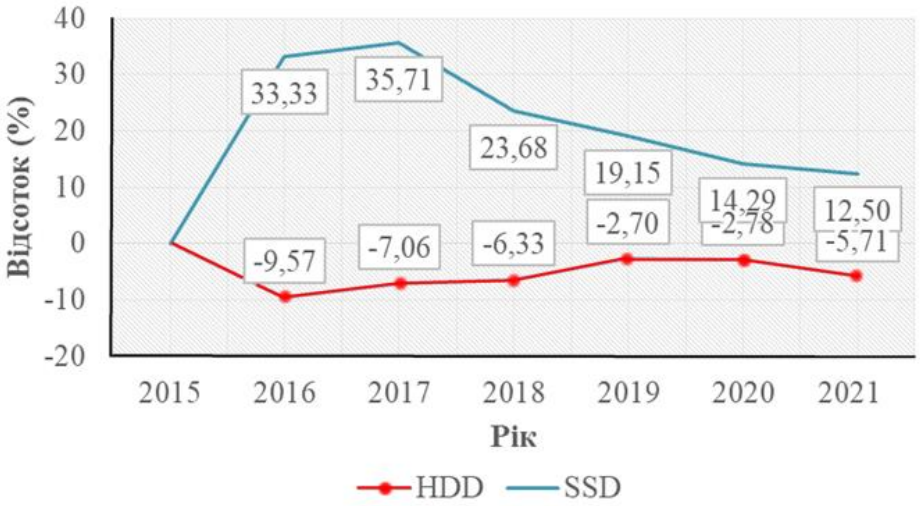


Рис. 2 – Динаміка зміни кількості HDD та SDD пристроїв

Вплив технології накопичувача на можливості методу приховування інформації у структуру файлових систем [31, 33], більш детально оцінимо швидкодію кожної технології. Так як єдиної специфікації по SSD, HDD технологіям не існує, а швидкодія кожного пристрою залежить від виробника та цінової категорії, тож порівняльна оцінка зроблена спираючись на результати електронного ресурсу UserBenchmark (*https://www.userbenchmark.com*).

Даний ресурс збирає дані по компонентам комп'ютера (CPU, GPU, SSD, HDD, RAM, USB) проводячи тестування «Open Source» програмним забезпеченням. Дані з UserBenchmark є об'єктивними так як на ресурсі вказані результати більш ніж 160 млн. користувачів.

4 Методика дослідження та результати експериментів

Слід підкреслити, що в межах виконання даної роботі нас цікавили лише SSD та HDD накопичувачі, так як для методу приховування інформації [31], важливим є параметр швидкості запису/зчитування з носія інформації. Тому, в якості тестових було обрано найкращі пристрої за цими показниками: - для SSD, це – Gigabyte, mod. GP-ASM2NE6100TTTD Aorus NVMe PCIe M.2 1TB (190$); – для HDD, це – WD, mod. WD6001FZWX Black 6TB (225$).

Тестування даних пристрої проводилось наступними методами:

1. *Послідовного запису* (*Sequential*) – це модель доступу до диску, при якій великі блоки даних записуються у сусідні блоки на поверхні пристрою з глибиною черги, рівній одиниці. Даний термін використовується здебільшого у контексті порівняльного аналізу, швидкість вимірюють у МБ/с. Даний тип доступу часто використовують при зчитуванні/запису великих за розміром файлів, таких як відео, музика та зображення. Як демонструє практика, близько 50% звичайного доступу користувачів до дискового накопичувача на їх персональному комп'ютері (ПК), складається з послідовних операцій зчитування/запису. Таки накопичувачі, що використовуються для зберігання великих мультимедійних файлів та/або резервних копій, повинні мати відносно велику швидкість послідовного доступу до файлів;
2. *Випадкового запису 4K* (*Random 4k*) – це шаблон доступу до диску, при якому невеликі (*4k*) блоки даних записуються у випадкові місця на поверхні тестового пристрою, з глибиною черги рівній одиниці. Даний термін використовується здебільшого у контексті порівняльного аналізу, швидкість вимірюють у МБ/с. Цій метод оцінки швидкодії можна використовувати для визначення того, наскільки ефективно пристрій зчитує/записує невеликі фрагменти даних із випадкових місць. Даний шаблон доступу характерний під час завантаження операційної системи, коли з накопичувача необхідно зчитати велику кількість файлів конфігурації та драйверів. Як демонструє практика, близько 20% звичайного доступу користувачів до дисків ПК, складається з запису/зчитування, саме з випадкових блоків накопичувача.

Результат тестування *Random 4k* є більш важливим для методу приховування інформації шляхом перемішування кластерів у структурі файлової системи так як при приховуванні даних запис виконується у «випадкові» блоки накопичувача у залежності від приховуваної інформації. Що відповідає швидкості доступу до файлу із значним рівнем фрагментації.

Також необхідно зазначити що модифікований метод приховування даних у структуру файлової системи FAT (який запропоновано у [31]) дозволяє використовувати 1 покриваючий файл з неперервним ланцюгом кластерів але з певним рівнем переплетеності.

Переплетеність – це властивість файлу при якому кластери файлу розміщуються безперервно, без проміжків між кластерами, але можливий зворотній напрямок у індексації кластерів. Як приклад переплетений файл, це файл кластери якого розміщені з 1 по 10 кластери, але ланцюг кластерів може мати вигляд як $Chain_F = [1,9,2,3,10,4,8,7,5,6]$. У загальному вигляді вираз, який визначає рівень переплетеності файлу має наступний вигляд (1):

$$Ent_F = \sum_{i=1}^{F_{len}} \left| Chain_F[i] - Chain_F[i+1] \right| - 1, \quad (1)$$

де:

- Ent_F (*entanglement*) – рівень переплетеності файлу F;
- F_{len} – довжина файлу F у кластерах;
- $Chain_F[i]$ – i елемент ланцюгу кластерів файлу F.

Т.ч., рівень переплетеності файлу у попередньому прикладі буде дорівнювати:

$$Ent_F = (|1-9|-1) + (|9-2|-1) + \ldots + (|5-6|-1) = 28.$$

Так як в межах даної роботи виконувався аналіз фізичних властивостей носіїв інформації, то можна прирівняти вплив фрагментованості файлу до впливу переплетеності файлу на швидкість запису/зчитування. Адже, як фрагментованість, так і переплетеність змушують зміщуватися датчик зчитуючого пристрою (для HDD) для доступу до наступного кластеру.

Зведені результати тестувань швидкодії SSD та HDD пристроїв за результатами UserBenchmark наведені у табл. 2 і 3, де: - *T – технологія пристрою;* - *Мт – метод оцінювання;* - *R (read) – швидкість зчитування даних;* - *W (write) – швидкість запису даних;* - *M (mixed) – швидкість почергового зчитування/запису даних.*

Таблиця 2 – Результати UserBenchmark методом Sequential

T	**SSD** *(МБ/с)*			**HDD** *(МБ/с)*		
Мт	**Sequential**			**Sequential**		
	Min.	Avg.	Max.	Min.	Avg.	Max.
R	809	1950	2318	102	167	215
W	1705	3115	3783	138	202	246
M	704	1998	2315	67.5	102	215

Таблиця 3 – Результати UserBenchmark методом Random 4k

T	**SSD** *(МБ/с)*		**HDD** *(МБ/с)*		
Мт	**Random 4k**		**Random 4k**		
	Min.	Avg.		Min.	Avg.
R	33	49.5	R	33	49.5
W	103	176	W	103	176
M	50.7	78.2	M	50.7	78.2

Таким чином, якщо порівнювати швидкості накопичувачів, то безпосередньо SSD є більшь швидким, у абсолютних значеннях. Якщо порівнювати зниження швидкості між Random 4k і Sequential, то SSD при фрагментованому запису має швидкість у 5-10%, у той час, як HDD втрачають свою швидкість навіть до 1.5-3%. Звісно 5-10% це мало, але це значно краще ніж для випадку використання HDD. Наведемо характерний приклад впливу фрагментованості носія, на швидкість доступу до файлу. Для цього проаналізуємо середній час запису і читання даних з диску при його фрагментації, та без неї. При цьому зауважимо, що середній час читання з диску, це час, який у середньому необхідний на здійснення однієї операції читання даних з носія інформації (*в секундах*). Відповідні відомості вказані на рис. 3 та 4 (*за даними, https://club.directum.ru/post/140*).

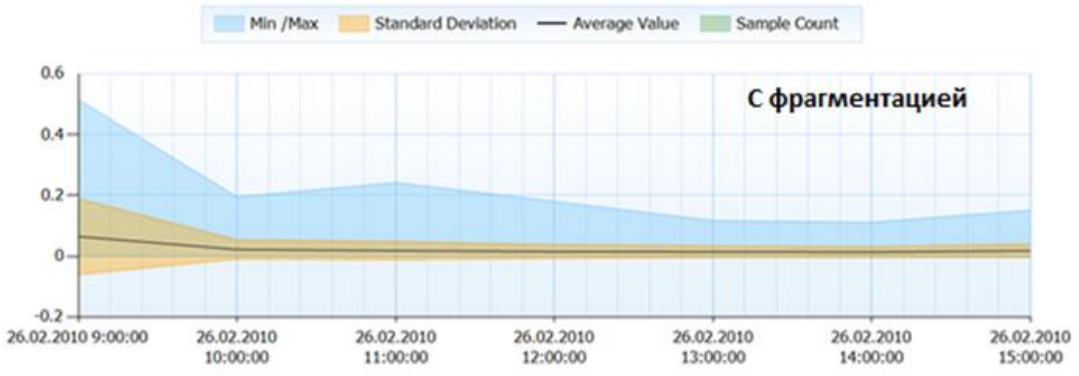


Рис. 3 – Середній час зчитування даних з диску при фрагментації даних

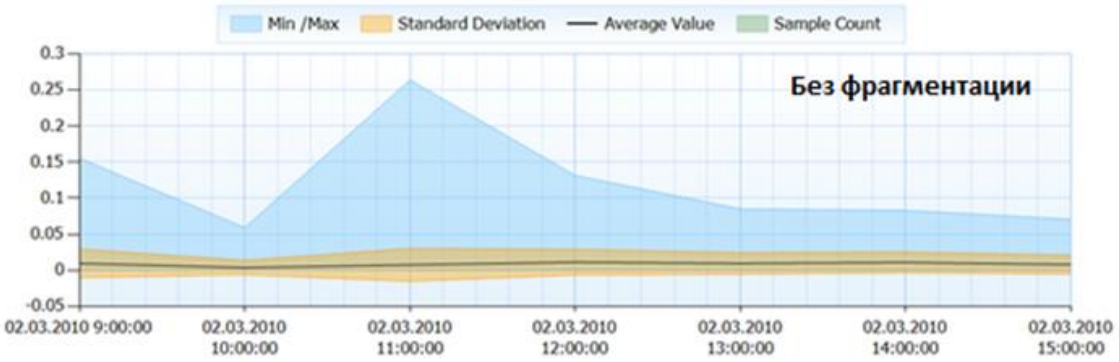


Рис. 4 – Середній час зчитування даних з диску без фрагментації даних

Додаткові результати, також, наведені у таблиці 4.

Таблиця 4 – Оцінка затраченого часу при читанні даних з диску

	Мінімальне значення *(с)*	**Середнє значення** *(с)*	**Максимальне значення** *(с)*
З фрагментацією	0	0.23	0.5
Без фрагментації	0	0.08	0.26

Також проаналізуємо середній час запису на диск. Так само оцінюючи час при фрагментованості та без фрагментованості. Результати вказані на рисунку 5 та таблиці 5 (за даними *https://club.directum.ru/post/140*).

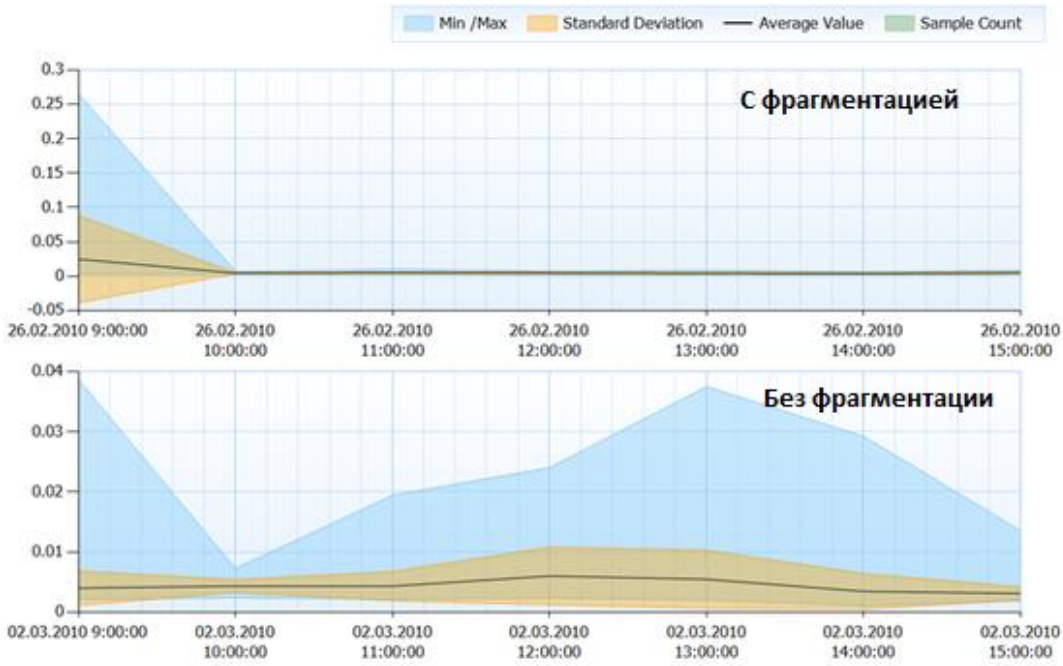


Рис. 5 – Середній час запису даних на носій при фрагментації та без неї

Таблиця 5 – Оцінка затраченого часу при запису даних на диск

	Мінімальне значення (с)	**Середнє значення (с)**
З фрагментацією	0,001787	0,007211
Без фрагментації	0,001044	0,004368

Порівнюючи результати обробки даних з фрагментацією та без неї, можна стверджувати, що обробка даних без фрагментації ефективніша на 40% - 60%, ніж обробка фрагментованих даних. Але використання SSD здатне зменшити негативний вплив фрагментованості даних на швидкість обробки цих даних. Виходячи з результатів методів *Random 4k* та *Sequential* можна стверджувати, що чим більший рівень фрагментації, тим менша швидкість доступу до файлу. Особливо це впливає на швидкодію HDD накопичувачів, у той час як SSD накопичувачі дозволяють проводити запис/зчитування файли майже без втрати швидкості, навіть при значному рівні фрагментованості даних.

5 Висновки

На закінчення можна сказати, що технологія SSD має значні переваги, завдяки чому накопичувачі з даною технологією стають більш розповсюдженими. SSD має у рази більшу швидкість доступу до файлів/секторів, навіть при значному показнику фрагментованості – це сприятливий показник для використання стеганографічних методів приховування даних. Також SSD має обмежену кількість циклів перезапису, і хоча це й недолік, але для методу приховування даних цей показник сприятливий, адже виконання дефрагментації є небажаною операцією для SSD.

Отже як висновок можна рекомендувати використання SSD носіїв інформації для приховування повідомлення у структуру файлової системи:

1 - через те що швидкість доступу до кластеру значно вища, що забезпечить більше швидке виконання стеганографічного методу;

2 - при збільшенні рівня фрагментованості (переплетеності) швидкість доступу до файлу втрачає не так багато як у порівнянні з HDD технологією, що є значно важливішим показником при використанні методу приховування даних у структурі файлової системі;

3 - виконання дефрагментації SSD накопичувачів є небажаною процедурою, що призводить до збільшення загального рівня фрагментованості на носії інформації, що у свою чергу дозволяє приховати більше інформації без ризику розкриття (чим більший рівень фрагментованості на носії тим більше інформації можна приховати [33]).

Таким чином завдяки розповсюдженню SSD метод приховування інформації у структуру файлових систем шляхом перемішування кластерів покриваючих файлів є актуальним.

Посилання

[1] Klima, R.E., Klima, R., Sigmon, N.P., Sigmon, N., Klima, R., Sigmon, N.P., Sigmon, N.: Cryptology : Classical and Modern. Chapman and Hall/CRC (2018). https://doi.org/10.1201/9781315170664.

[2] Delfs, H., Knebl, H.: Introduction to Cryptography. Springer Berlin Heidelberg, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47974-2.

[3] Childs, L.N.: Cryptology and Error Correction: An Algebraic Introduction and Real-World Applications. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-15453-0.

[4] Manoj, I.V.S.: Cryptography and Steganography. IJCA. 1, 63–68 (2010). https://doi.org/10.5120/257-414.

[5] Yahya, A.: Introduction to Steganography. In: Yahya, A. (ed.) Steganography Techniques for Digital Images. pp. 1–7. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-319-78597-4_1.

[6] Qin, J., Luo, Y., Xiang, X., Tan, Y., Huang, H.: Coverless Image Steganography: A Survey. IEEE Access. 7, 171372–171394 (2019). https://doi.org/10.1109/ACCESS.2019.2955452.

[7] Schöttle, P., Böhme, R.: Game Theory and Adaptive Steganography. IEEE Transactions on Information Forensics and Security. 11, 760–773 (2016). https://doi.org/10.1109/TIFS.2015.2509941.

[8] Yahya, A.: Steganography Techniques. In: Yahya, A. (ed.) Steganography Techniques for Digital Images. pp. 9–42. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-319-78597-4_2.

[9] Fridrich, J.: Steganography in Digital Media: Principles, Algorithms, and Applications. Cambridge University Press, Cambridge ; New York (2009).

[10] Cox, I., Miller, M., Bloom, J., Fridrich, J., Kalker, T.: Digital Watermarking and Steganography, 2nd Ed. Morgan Kaufmann, Amsterdam ; Boston (2007).

[11] Kim, C.R., Lee, S.H., Lee, J.H., Park, J.-I.: Blind decoding of image steganography using entropy model. Electronics Letters. 54, 626–628 (2018). https://doi.org/10.1049/el.2017.4276.

[12] Rowland, C.H.: Covert channels in the TCP/IP protocol suite, https://firstmonday.org/ojs/index.php/fm/article/download/528/449?inline=1, Last Accessed 2020/11/01.

[13] Mazurczyk, W., Lubacz, J.: LACK—a VoIP steganographic method. Telecommun Syst. 45, 153–163 (2010). https://doi.org/10.1007/s11235-009-9245-y.
[14] Lubacz, J., Mazurczyk, W., Szczypiorski, K.: Principles and Overview of Network Steganography. IEEE Communications Magazine. 52, (2012). https://doi.org/10.1109/MCOM.2014.6815916.
[15] Mazurczyk, W., Smolarczyk, M., Szczypiorski, K.: On information hiding in retransmissions. Telecommun Syst. 52, 1113–1121 (2013). https://doi.org/10.1007/s11235-011-9617-y.
[16] Cauich, E., Gómez Cárdenas, R., Watanabe, R.: Data Hiding in Identification and Offset IP Fields. In: Ramos, F.F., Larios Rosillo, V., and Unger, H. (eds.) Advanced Distributed Systems. pp. 118–125. Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/11533962_11.
[17] Wang, M., Gu, W., Ma, C.: A Multimode Network Steganography for Covert Wireless Communication Based on BitTorrent, https://www.hindawi.com/journals/scn/2020/8848315/, last accessed 2020/11/08. https://doi.org/10.1155/2020/8848315.
[18] Seo, J.O., Manoharan, S., Mahanti, A.: Network steganography and steganalysis - a concise review. In: 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT). pp. 368–371 (2016). https://doi.org/10.1109/ICATCCT.2016.7912025.
[19] Noskov, A.: Analysis of Network Protocols: The Ability of Concealing the Information. Computer and Network Security. (2020). https://doi.org/10.5772/intechopen.88098.
[20] A High Capacity 3D Steganography Algorithm, https://www.computer.org/csdl/journal/tg/2009/02/ttg2009020274/13rRUwdIOUD, Last Accessed 2020/10/20. https://doi.org/10.1109/TVCG.2008.94.
[21] Paramasivan, T., Natarajan, V., Gnanasekaran, A., Venkatesan, V., Anitha, R.: Pattern based 3D image Steganography. 3D Research. 4, (2014). https://doi.org/10.1007/3DRes.01(2013)1.
[22] Chao, M.-W., Lin, C., Yu, C.-W., Lee, T.-Y.: A high capacity 3D steganography algorithm. IEEE Trans Vis Comput Graph. 15, 274–284 (2009). https://doi.org/10.1109/TVCG.2008.94.
[23] Li, N., Hu, J., Sun, R., Wang, S., Luo, Z.: A High-Capacity 3D Steganography Algorithm With Adjustable Distortion. IEEE Access. 5, 24457–24466 (2017). https://doi.org/10.1109/ACCESS.2017.2767072.
[24] Thiyagarajan, P., Natarajan, V., Aghila, G., Prasanna Venkatesan, V., Anitha, R.: Pattern based 3D image Steganography. 3D Res. 4, 1 (2014). https://doi.org/10.1007/3DRes.01(2013)1.
[25] Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Smirnov, O., Krasnobaev, V., Kuznetsova, K.: Information Hiding Using 3D-Printing Technology. In: 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). pp. 701–706 (2019). https://doi.org/10.1109/IDAACS.2019.8924352.
[26] Kuznetsov, A.A., Stefanovych, O.O., Prokopovych-Tkachenko, D.I., Kuznetsova, K.O.: 3D STEGANOGRAPHY INFORMATION HIDING. TRE. 78, (2019). https://doi.org/10.1615/TelecomRadEng.v78.i12.30.
[27] Khan, H., Javed, M., Mirza, F., Khayam, S.: Evading Disk Investigation and Forensics using a Cluster-Based Covert Channel. (2012).
[28] Khan, H., Javed, M., Khayam, S.A., Mirza, F.: Designing a cluster-based covert channel to evade disk investigation and forensics. Computers & Security. 30, 35–49 (2011). https://doi.org/10.1016/j.cose.2010.10.005.
[29] Venčkauskas, A., Morkevicius, N., Petraitis, G., Ceponis, J.: Covert Channel for Cluster-based File Systems Using Multiple Cover Files. Information technology and control. 42, (2013). https://doi.org/10.5755/j01.itc.42.3.3328.
[30] Kuznetsov, A., Shekhanin, K., Kolhatin, A., Mikheev, I., Belozertsev, I.: Hiding data in the structure of the FAT family file system. In: 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT). pp. 337–342 (2018). https://doi.org/10.1109/DESSERT.2018.8409155.
[31] Shekhanin, K.Y., Kolhatin, A.O., Demenko, E.E., Kuznetsov, A.A.: ON HIDING DATA INTO THE STRUCTURE OF THE FAT FAMILY FILE SYSTEM. TRE. 78, (2019). https://doi.org/10.1615/TelecomRadEng.v78.i11.50.
[32] Vokorokos, L., Madoš, B., Ádám, N., Baláž, A., Porubän, J., Chovancová, E.: Multi-Carrier Steganographic Algorithm Using File Fragmentation of FAT FS. COMPUTING AND INFORMATICS. 38, 343-366–366 (2019).
[33] Shekhanin, K., Kuznetsov, A., Krasnobayev, V., Smirnov, O.: Detecting Hidden Information in FAT. IJCNIS. 12, 33–43 (2020). https://doi.org/10.5815/ijcnis.2020.03.04.
[34] Aycock, J., de Castro, D.M.N.: Permutation Steganography in FAT Filesystems. In: Shi, Y.Q. (ed.) Transactions on Data Hiding and Multimedia Security X. pp. 92–105. Springer, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46739-86.
[35] Davis, J., MacLean, J., Dampier, D.: Methods of Information Hiding and Detection in File Systems. (2010). https://doi.org/10.1109/SADFE.2010.17.
[36] Neuner, S., Voyiatzis, A.G., Schmiedecker, M., Brunthaler, S., Katzenbeisser, S., Weippl, E.R.: Time is on my side: Steganography in filesystem metadata. Digital Investigation. 18, S76–S86 (2016). https://doi.org/10.1016/j.diin.2016.04.010.
[37] FAT File System, https://www.keil.com/pack/doc/mw/FileSystem/html/fat_fs.html, Last Accessed 2020/10/01.
[38] FAT File Systems. FAT32, FAT16, FAT12 - NTFS.com, https://www.ntfs.com/fat_systems.htm, Last Accessed 2020/10/10.
[39] Overview of FAT, HPFS, and NTFS File Systems, https://support.microsoft.com/en-us/help/100108/overview-of-fat-hpfs-and-ntfs-file-systems, Last Accessed 2020/10/10.

Reviewer: Mikołaj Karpiński, Dr. of Sciences (Eng.), Full Prof., University of Bielsko-Biala, Bielsko-Biala, Poland.
E-mail: mkarpinski@ath.bielsko.pl

Received on December 2020.

Authors:
Kirill Shekhanin, Postgraduate Student, Faculty of Computer Science, V.N.Karazin Kharkiv National University, Kharkov, Ukraine.
Lyudmila Gorbachovaa, student, Faculty of Computer Science, V.N.Karazin Kharkiv National University, Kharkov, Ukraine.
E-mail: lusyag23@gmail.com
Katerina Kuznetsova, student, Faculty of Computer Science, V.N.Karazin Kharkiv National University, Kharkov, Ukraine.
E-mail: kate7smith12@gmail.com

Comparative analysis and study of the properties of information carriers for steganographic data hiding in clustered file systems.

Abstract. The paper studies and analyzes various modern information storage technologies, namely HDD, Flash-USB, SSD. We`ve analyzed different indicators such as the number of implemented products, price, speed of reading and writing. Besides, we`ve considered some indicators of the information carriers' efficiency in terms of view of the possibility of using steganographic methods for hiding information in clustered file systems. It have been analyzed the speed of sequential reading / writing and the speed of access to a random cluster, corresponding to the speed of access to a fragmented file. For this task, we used the test results from the UserBenchmark resource. The testing has made using the Sequential and Random4k methods. In addition, we have provided an assessment of information carriers and have gave recommendations of using the particular information carrier and method for hiding data by mixing clusters in the structure of the file system. Besides, it was analyzed the dependence of the speed parameters of access to the cluster on the level of file fragmentation. Refinements are made of how an increase or decrease in the level of fragmentation (entanglement) affects the speed of access to the file, which is an important indicator when using the method of hiding data in the file system structure. The advantages and disadvantages of various types of storage devices have been discussed, and its comparative analysis was made. Moreover, we analyzed the features of the process of defragmentation of drives, and the influence of various factors on the overall level of fragmentation on the storage medium. We placed emphasis on the greater the level of fragmentation on the storage medium, the more information could be hidden. It was concluded that due to the widespread use of SSD / HDD drives, the method of hiding information in the structure of file systems, by mixing clusters of covering files, is relevant.

Keywords: File storage media; Fragmentation; Speed of access; Data hiding, Steganographic.

Рецензент: Николай Карпинский, д.т.н., проф., Университет Бельсько-Бяла, ул. Виллова 2, 43-309 Бельсько-Бяла, Польша.
E-mail: mkarpinski@ath.bielsko.pl

Поступила: Декабрь 2020.

Авторы:
Кирил Шеханин, аспирант, Харьковский национальный университет имени В. Н. Каразина, Харьков, Украина.
Людмила Горбачова, студентка факультета компьютерных наук, Харьковский национальный университет имени В. Н. Каразина, Харьков, Украина.
E-mail: lusyag23@gmail.com
Екатерина Кузнецова, студентка факультета компьютерных наук, Харьковский национальный университет имени В. Н. Каразина, Харьков, Украина.
E-mail: kate7smith12@gmail.com

Сравнительный анализ и изучение свойств носителей информации для стеганографических данных, скрываемых в кластерных файловых системах.

Аннотация: В работе рассмотрены современные технологии хранения информации, а именно *HDD, Flash-USB, SSD*. Проанализированы такие показатели, как количество реализованных изделий, цена, скорость считывания и записи. Исследованы некоторые показатели эффективности носителей информации, с точки зрения возможности применения стеганографических методов сокрытия информации в кластерных файловых системах. Выполнен анализ скорости последовательного чтения / записи и скорости доступа к случайному кластеру, соответствующей скорости доступа к фрагментованному файлу. Для этого использовались результаты тестирования с ресурса *UserBenchmark*. Тестирование выполнялись методами *Sequential* и *Random4k*. Предложена оценка носителей информации и даны рекомендации по использованию носителя информации и метода сокрытия данных путем перемешивания кластеров в структуре файловой системы. Проведен анализ взаимосвязи и зависимости параметров скорости доступа к кластеру от уровня фрагментированности файла. Уточнено, каким образом увеличение или уменьшение уровня фрагментированности (*переплетенности*) влияет на скорость доступа к файлу, что является важным показателем при использовании метода сокрытия данных в структуре файловой системы. Рассмотрены преимущества и недостатки накопителей информации различных типов, и проведен их сравнительный анализ. Выполнен анализ особенностей процесса дефрагментации накопителей, и влияние различных факторов на общий уровень фрагментированности на носителе информации. Подчеркнуто, что чем больше уровень фрагментированности на носителе данных, тем больше информации можно скрыть. Утверждается, что благодаря широкому распространению *SSD / HDD* накопителей, метод сокрытия информации в структуру файловых систем, путем перемешивания кластеров покрывающих файлов, является актуальным.

Ключевые слова: файловые носители информации; фрагментация; скорость доступа; сокрытие данных; стеганография.

УДК 004.5: 316.472.4 DOI: 10.26565/2519-2310-2021-1-04

АНАЛІЗ ФАКТОРІВ І УМОВ РЕАЛІЗАЦІЇ КІБЕРБУЛІНГУ З УРАХУВАННЯМ МОЖЛИВОСТЕЙ СУЧАСНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Валерія Гайкова, Сергій Малахов
Харківський національний університет імені В.Н. Каразіна, майдан Свободи 4, Харків, 61022, Україна
valeriagaikova98@gmail.com, mailgate@meta.ua

Рецензент: Володимир Хома, д.т.н., проф., Опольський політехнічний Університет, Ополе, Польща
xoma@wp.pl

Надійшла: Січень 2021.

Анотація: *Представлений перелік основних узагальнень і чинників, які характерні для різних варіантів та умов здійснення акцій мережевий цькування (кібербулінгу). Запропонована аргументація певних аналогій і співпадінь між процесом мережевого цькування та умовами, і цілями проведення сучасних інформаційних операцій. Звернено увагу те, що умови кібертравлі багато в чому збігаються з парадигмою експерименту С. Мілгрема. Запропоновано аргументація суті деяких аналогій між проявами кібербулінгу та основною фабулою експерименту про покору. Підкреслено, що рівень ієрархічної важливості об'єктів кібертравлі, обумовлює ступінь необхідної інтеграції функцій адміністрування контентом і основних складових ресурсного забезпечення сучасних інформаційних систем. Зазначено, що інтеграція в одному центрі прийняття рішень основних складових ресурсного забезпечення та функцій адміністрування контенту, є серйозною проблемою навіть для рівня окремих держав, не кажучи вже про персональний рівень впливу. Звернуто увагу на той факт, що кібербулінг може реалізовуватися, як шляхом використання можливостей окремих інформаційних технологій, так і втілювати концепцію інтегрованої атаки. Підкреслено, що явище кібербулінгу є значно недооціненим і тому являє собою серйозну проблему сучасності.*

Ключові слова*: булінг; кібербулінг; інформаційна повістка; контент; інформаційні технології; технічна платформа; мережева поведінка; мережевий статус; експеримент Мілгрема.*

1 Вступ

Стрімкий розвиток інформаційних технологій (ІТ) привносить в сучасне суспільство багато позитивних та корисних тенденцій, однак, нажаль, він має і свої «темні» сторони, що приховані від непосвячених в реалії існування сучасного цифрового світу. Однією з таких сторін, є явище булінгу, яке в наслідок стрімкого науково-технічного розвитку останніх десятиліть, поступово «перейшло» в кіберпростір, та відкрило нову, неприваблиу сторінку технологічної історії людства – сторінку кібербулінгу [1-3]. Процес протистояння булінгу в кіберпросторі [4-5], обумовлює необхідність постійної розробки нових та вдосконалення вже існуючих технологій і засобів протидії, які спрямовані, в тому числі, на завчасне коригування мережевої поведінки користувачів різних інформаційних систем, та онлайн сервісів [6].

2. Основна частина

Результати всебічного аналізу інцидентів, що пов'язані з різними формами проявів булінгу в кіберпросторі, дозволяє зробити кілька принципових узагальнень:

1 – цькування (або травля) в кіберпросторі має два рівня впливу на обрані об'єкти психологічної атаки : - груповий *(глобальний)* та індивідуальний *(персональний)*;

2 – об'єкти кібертравлі значною мірою збігаються з основними об'єктами впливу/атаки в ході ведення сучасних інформаційних операцій [7];

3 – практично всі випадки кібербулінгу, мають структуру і основні складові, які притаманні для сучасних[1] умов ведення інформаційних операцій [7, 8];

4 – рівень значущості заявленої жертви мережевого булінгу, визначає необхідний обсяг фінансової складової ресурсного забезпечення заходів кібертравлі;

5 – доступність і неперервність використовуваних комунікаційних сервісів і онлайн послуг, що надаються різними інформаційними системами, є визначальним (*критично важливим*) фактором для досягнення кінцевих цілей будь-яких акцій булінгу в кіберпросторі;

6 – характер поведінки і ролі [9] основних учасників процесу кібертравлі, багато в чому ідентичні умовам і цілям проведення експерименту Стенлі Мілгрема (*відомий, як «Експеримент про покору»*) [10-12]. У разі акцій кібертравлі, в незалежності від рівня впливу на обрану жертву та рівня її ієрархічної значущості, ми є свідками процесу масової реінкарнації цього експерименту в умовах існування сучасного інформаційного суспільства [13-14], з відповідною «еволюцією»[2] уявлень учасників цього процесу, про норми соціальної і мережевої поведінки людини[3] в кіберпросторі.

Примітки:

[1] – *мається на увазі, використання можливостей існуючих інформаційних систем (технічних комунікаційних платформ) та IT-технологій;*

[2] – *може і не являтися такою, а бути, лише, похідною поточних комерціалізованих тенденцій, сформульованих домінуючими постачальниками телекомунікаційних послуг і сервісів (наприклад, гігантами IT-індустрії: Twitter, Facebook, YouTube та ін.);*

[3] – *однієї зі сторін процесу може являтися комп'ютеризована система (бот), що реалізує, як наперед задані поведінкові алгоритми, так і підтримувати функції самонавчання (штучного інтелекту). У останньому випадку складно говорити про «соціалізацію» поведінкових алгоритмів боту, зважаючи на значну складність формалізації його поведінкових реакцій.*

Розглянемо запропоновані узагальнення та зробимо деякі пояснення по кожному з них.

1. В рамках зазначених рівнів впливу, в якості заявленої цілі атаки (*тобто жертви кібертравлі*) можуть виступати:

- конкретні фізичні персони (*окремі користувачі будь-яких інформаційних систем і онлайн сервісів, представники бізнес структур, політичних партій і громадських об'єднань, представники ланок державної влади, культових організацій та ін.*);

- мережеві об'єднання груп користувачів в межах існуючих інформаційно-комунікаційних платформ *(учасники соціальних мережевих співтовариств (чати, форуми та ін.) з різними критеріями групоутворення (політичні, культурні, професійні та ін.))*;

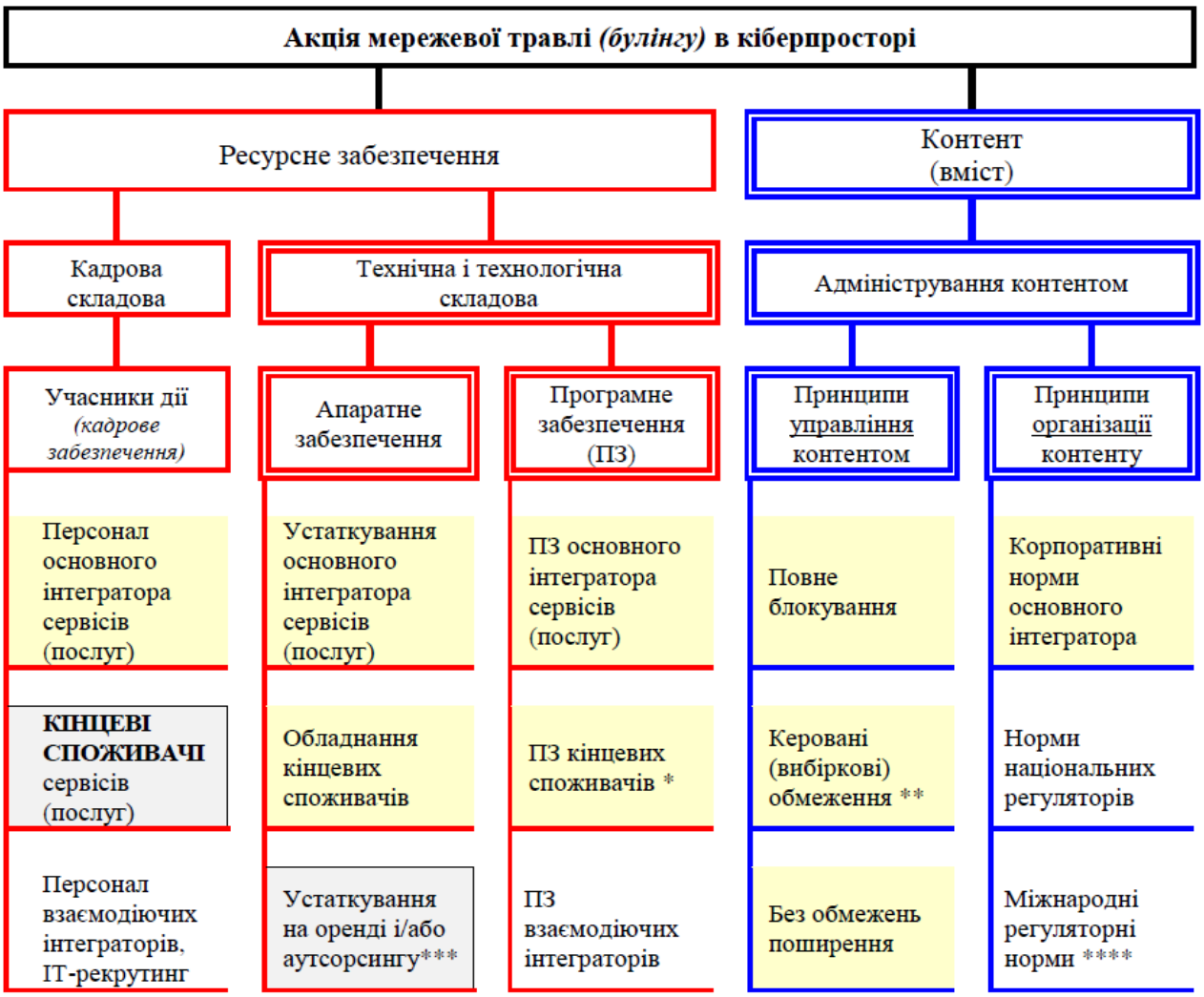
- корпоративний сегмент (*представництва окремих бізнес структур, галузевих об'єднань, медійних холдингів, політичних партій, представництва культових організацій, окремі ланки державного управління та ін.*);

- цивілізаційно-ціннісні основи суспільства (*історико-культурні та релігійні традиції, міжрасові і соціальні норми відносин тощо*).

В наведеному переліку «потенційних жертв» особливого коментарю потребує питання, що стосується співвіднесення ієрархічної значущості об'єктів атаки при реалізації акцій булінгу, стосовно існуючих цивілізаційно-ціннісних основ суспільства. З огляду на очевидний суспільний резонанс від спроб проведення заходів подібного характеру, рівень представництва основних учасників відповідних атак, вимагає високого ступеня їх повноважень (*протегування*). Характерним прикладом акцій подібного рівня значущості, може слугувати епізод, що пов'язаний з трагічними подіями 2015 року в редакції щотижневика *Charlie Hebdo* [15].

2. Рівень важливості (*ієрархічної значущості*) об'єктів, обраних для кібертравлі, зумовлює ступінь необхідної інтеграції (*об'єднання*) ***функцій управління*** контентом та основних складових ресурсного забезпечення (рис. 1) використовуваних інформаційних систем. При цьому, реалізація акцій мережевої травлі проти об'єктів з високим рівнем ієрархічної значущості, на відміну від атак цілей з низькою ієрархічної важливістю, потребує безумовного об'єднання в одному центрі прийняття рішень функцій адміністрування контентом та технічного забезпечення (*подвійний контур на рис. 1*). Характерним прикладом подій відповідного ієрархічного рівня може слугувати епізод політичної кризи в Венесуелі у 2019 році [16].

Можна стверджувати, що відсутність у представників атакуючої сторони (*перш за все рольових груп «агресор» та «помічник агресора»* [9]) функцій управління та контролю використовуваних інформаційно-комунікаційних платформ (рис. 2), обумовлює локальний характер для будь-яких заходів кібертравлі проти жертв з низькими рівнями їх ієрархічної значущості (*наприклад, випадки підліткової цькування на ґрунті боротьби за домінування в своїх мікрогрупах*).



* - *додатки користувачів, програмна «прошивка» мобільних гаджетів, комунікаційних терміналів і мультимедійних пристроїв;*

** - *за часом, геолокацією, мовою і/або каналами видачі контенту, пошуковими запитами, типом і/або швидкістю трафіку та ін.;*

*** - *тимчасові посередники телекомунікаційних послуг, IT-аутсорсинг, хостинг, оренда додаткових апаратних і/або канальних ресурсів інформаційно-комунікаційних систем (наприклад, розширення серверних можливостей, використання «віртуального» волокна, збільшення потужності TV- і радіо передавачів систем мовлення, збільшення кількості транспондерів супутників зв'язку та ін.);*

**** - *наприклад, відповідні визначення ООН, ЮНЕСКО та ін.*

Рис. 1 – Тотожність структури складових кібербулінгу та сучасних інформаційних операцій

3. Для аргументації узагальнення, стосовно тотожності структури кібербулінгу (*особливо для випадків глобального рівня впливу*) та основних складових інформаційних операцій, слід проаналізувати відомості, які систематизовані на рис. 1.

Аналіз наведеної схеми дозволяє стверджувати, що основні складові процесу булінгу в кіберсфері, співпадають з відповідними елементами, які притаманні для сучасних умов ведення інформаційних операцій [7]. Існування можливих відмінностей, в частині ієрархічної значущості обраної жертви, складу кадрового забезпечення та ролей учасників цькування, абсолютно не принципово, так як в будь-якому випадку присутні безпосередні виконавці акції мережевої травлі, та є маркована ціль для атаки.

В разі високої ієрархічної значущості жертви, основні учасники заходів мережевої травлі присутні в усіх 3-х складових кадрового забезпечення (*див. рис. 1*). Так, представники основного інтегратора (*або інтеграторів*) комунікаційних послуг, на базі котрого (-*рих*), безпосередньо, здійснюється інформаційний вплив, представлені на рівні її технічних спеціалістів і фахівців, котрі здійснюють концептуальну модерацію контенту та визначають етапність його подачі (*тобто, фактично, визначають інформаційну повістку всієї акції*).

Другій рівень – «***КІНЦЕВІ СПОЖИВАЧІ***», за рахунок щільної взаємної інтеграції використовуваних телекомунікаційних сервісів і послуг, формує наймасовішу кадрову складову подій. Даний рівень поєднує заявлену жертву атаки, та саму представницьку рольову групу – «спостерігачів», яка, власне, і є прихованою (*неявною*) головною метою здійснюваного інформаційного впливу.

Третій рівень – «***Персонал взаємодіючих інтеграторів …***», складають тимчасові IT–аутсорсери [17] та фахівці на IT–рекрутингу, які залучаються для підтримки необхідних форм (*чат-боти, стрім-канали, тематичні форуми тощо*), масштабів і термінів проведення акції. Представники цієї групи здійснюють проміжну обробку і розповсюдження потрібного контенту, та можуть приймати безпосередню участь в здійсненні певних заходів цькування.

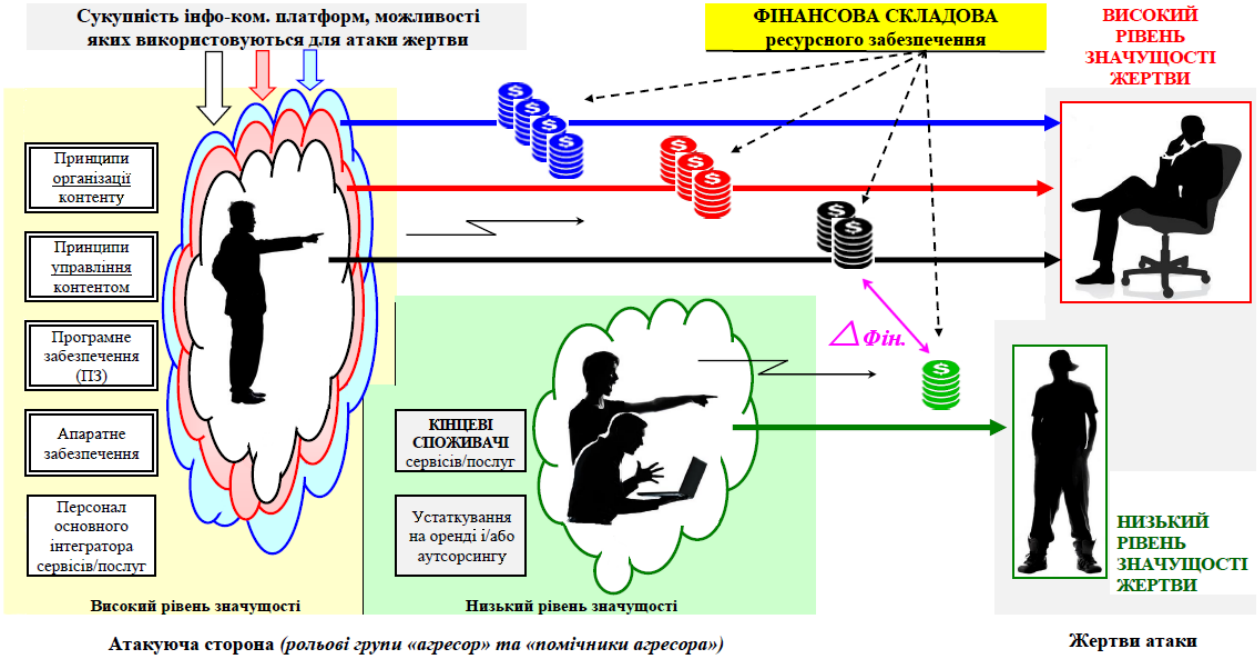


Рис. 2 – Відмінності повноважень адміністрування контенту та складових ресурсного забезпечення агресора для різних рівнів значущості жертви

При низьких рівнях ієрархічної значущості жертв атаки *(наприклад, у випадках підліткового цькування),* основні учасники травлі, об'єктивно обмежені представництвом рівня «***КІНЦЕВІ СПОЖИВАЧІ***». Іншими словами, всі учасники будь-яких можливих подій подібного масштабу та рівня значущості, повністю позбавлені компетенцій зі зміни параметрів технічної складової ресурсного забезпечення та функцій адміністрування "***принципів управління контентом***" (рис. 1-2). Власне ця обставина і обумовлює допустимі межі умовних "правил гри". Незначні відхилення від цих обмежень можливі в разі проведення акцій булінгу на рівні корпоративного сегменту, що може бути обумовлено наявністю відповідних елементів ресурсного забезпечення у окремих учасників процесу (*наприклад, наявність власного медійного ресурсу*) та/або їх можливістю здійснювати хоча б часткове адміністрування контенту *(наприклад, визначати власні "принципів управління контентом")*. Також, при здійсненні булінгу на корпоративному рівні впливу та/або проти об'єктів з високим рівнем ієрархічної значущості, категорія «***Персонал взаємодіючих інтеграторів…***» (рис. 1) може поєд-

нувати в собі представників ролевих груп «***помічники агресорів***» і «***хамелеони***», які будуть діяти згідно загальної концепції заходів мережевого цькування (*буде розглянуто нижче*).

4. Вочевидь, що необхідність збільшення кількості задіяних технічних ресурсів *(наприклад, використання можливостей відразу декількох інформаційно-комунікаційних платформ (див. рис.2))*, зумовлює зростання потрібного обсягу "**ФІНАНСОВОЇ СКЛАДОВОЇ**" **ресурсного забезпечення** заходів мережевий цькування, які проводяться або плануються. Тобто фінансове забезпечення заходів кібертравлі проти жертв з низьким рівнем значущості, може носити вельми умовний характер. В даному випадку, обмеження можливостей по контролю апаратного і програмного забезпечення тільки на рівні "**Кінцевих споживачів**", обумовлює "скромні" рівні бюджету подібних акцій (*в даному випадку, взагалі, можна говорити про ситуативний булінгсорсинг*). У той же час, атака жертви з низьким рівнем ієрархічної значущості, може здійснюватись шляхом "злому" та подальшого використання сторонніх апаратних ресурсів (*наприклад, шляхом формування бот-мережі, що поширює наклепницький контент*). В даному випадку всі витрати, що стосуються оплати мережевого трафіку і електроживлення скомпрометованих пристроїв, несуть власники цих (*зламаних*) пристроїв.

Таким чином, склад, структура і масштаб (*або кількість*) залучених ресурсів (рис. 1), визначають різницю в потрібних обсягах "**ФІНАНСОВОЇ СКЛАДОВОЇ**" для різних акцій мережевого булінгу (рис. 2). Крім того, маючи на увазі значну кореляцію структури і основних складових інформаційних операцій та кібербулінгу, можна стверджувати, що розмір "**ФІНАНСОВОЇ СКЛАДОВОЇ**" **ресурсного забезпечення**, безпосередньо визначає рівень потенційної складності структури заходів кібертравлі. Іншими словами, враховуючи багаторівневий (*або пошаровий*) характер сучасних інформаційних операцій, наявний розмір фінансового забезпечення заходів мережевого цькування, обумовлює кількість відповідних верств на кожному з етапів інформаційного впливу. Усунення або інформаційне купірування одного або ж відразу декількох шарів «атакуючого» контенту (*безвідносно причини події*) не змінює загального вектору проведеної атаки (кібертравлі) та гарантує збереження необхідної інформаційної повістки. Вочевидь, що послідовність, інтенсивність та тривалість інформаційного тиску на жертву, безпосередньо пов'язані з "**ФІНАНСОВОЮ СКЛАДОВОЮ**" **ресурсного забезпечення**, що є в розпорядженні потенційного агресора [18].

Так, наприклад, при необхідності легітимації джерела даних з заздалегідь неправдивою інформацією (*для умов булінгу - джерела даних з наклепницьким контентом*), в одному шарі, необхідно послідовно реалізувати кілька процедур:

а) синтезувати (*а згодом оновлювати*) «атакуючий» контент;

б) створити відповідний інформаційний ресурс - «прокладку» (*наприклад сайт або ж тематичну гілку на форумі або паблік в соціальній мережі*);

в) забезпечити механізм постійної апеляції/відсилання до контенту на даному ресурсі (*для умов кібербулінгу це функція рольової групи «**помічники агресора**»[9]*).

Для нарощування ефекту необхідна організація другого шару, що дозволить з необхідною періодичністю забезпечувати взаємний репост та/або посилання/згадки «атакуючого» контенту. Вочевидь, що зі збільшенням кількості подібних шарів збільшується кількість взаємних посилань, підтримується необхідна тематична повістка (*в нашому прикладі, дані з наклепницьким змістом*), та експресія цькування. При цьому локалізація і підтримка необхідної інформаційної повістки, в рамках конкретної інформаційно-комунікаційної платформи (рис. 2), забезпечується шляхом розміщення інформаційного ресурсу - «прокладки» на потрібної технічної платформі. Важливо підкреслити, що локація (*присутність*) цілі атаки та локація інформаційних ресурсів - «прокладок», можуть не збігатися! Іншими словами, атака обраної жертви може проводитися:

- при її повній відсутності на обраної технічній платформі;

- при її часткової присутності на цій платформі (*наприклад, епізод з обмеженням «присутності» колишнього президента Сполучених Штатів Америки Д. Трампа на основних національних інформаційних ресурсах в ході виборчої кампанії 2020 року*);

- в умовах паритетного представництва агресора і жертви, принаймні, на початковій фазі атаки. Очевидно, що такі (*дуельні*) умови невигідні для агресора, тому що знижують ефект від вжитих ним дій.

Комплексна атака жертви, з використанням можливостей відразу декількох інформаційно-комунікаційних платформ (рис. 2), потенційно, дозволяє реалізувати багатоетапну та багатошарову структуру інформаційного впливу. Проте, така схема булінгу потребуватиме використання загального координуючого центру і залучення більшої кількості відповідних фахівців (рис. 1), що зумовить зростання фінансового забезпечення потрібних заходів.

5. Фактор доступності (*неперервності*) використовуваних онлайн сервісів і послуг визначає не тільки спосіб, масштаб і тривалість реалізації процесу кібертравлі, а й обумовлює саму можливість його здійснення. Характерним прикладом успішно реалізованих заходів кібертравлі, що експлуатують фактор доступності комунікаційних сервісів, є події по одночасної «ізоляції» (*виключенню та ігноруванню*) колишнього президента Сполучених Штатів Америки Д. Трампа, на основних національних інформаційних і глобальних комунікаційних ресурсах (*TV компанії, соціальні мережі та ін.*). Такий варіант реалізації атаки забезпечує підтримку головної умови успішності проведеної акції – нерівність сил агресора[4] і жертви.

Примітка:

[4] – *уточнення рольової категорії «агресор», в разі монополізації основних складових процесу кібербулінгу, буде надано нижче, при розгляді питань, стосовно аналогій з умовами проведення експерименту С. Мілґрема.*

Показовим прикладом критичності фактора доступності, є можливість потенційного об'єднання, в рамках однієї компанії, холдингу або держави, основних складових «***Ресурсного забезпечення***» та функцій «***Адміністрування контенту***» (*виділено жовтим тоном на рис. 1*). Подібний варіант інтеграції складових досить імовірний, наприклад, в разі повного розгортання низькоорбітальної глобальної супутникової системи *Starlink* (*від компанії SpaceX*). Поєднання в одному центрі прийняття рішень основних складових ресурсного забезпечення (*перш за все її програмної і апаратної компонент*) та функцій адміністрування контенту стає серйозною проблемою навіть для рівня окремих держав, не кажучи вже про персональний рівень впливу. Атакуюча сторона, що має такі ресурси, отримує в «свої руки» всі ключові функції процесу: - планування; - супровід; - контроль результату. Більш того, домінування корпоративних норм основного інтегратора послуг в питаннях адміністрування контенту, фактично зумовлює кінцевий результат. В цих умовах, можлива протидія обмежується заходами в двох основних напрямках: - правовому та технічному. Зусилля по першому напрямку повинні бути сконцентровані на спробах спонукання (*згодом і примусу (податкового, тарифного тощо))* закордонних ІТ-компаній [5] відкривати свої представництва на територіях під юрисдикцією суб'єкта [6], котрий демонструє подібний «мотивуючий тиск». Зусилля в рамках 2-го напряму, скоріш за все, будуть обмежуватись, заходами фрагментарної фільтрації контенту, ліцензування та контролю постачання абонентського обладнання (*в т.ч. попереднього встановлення і контролю джерел поширення відповідного ПЗ*), і то, тільки за умови дієздатності відповідних відомств, та служб.

Примітка:

[5] – *сервіси, служби та послуги яких, використовуються, або можуть бути використані, в тому числі, для цілей проведення акцій мережевого булінгу;*

[6] – *вирішення проблеми транспарентності кордонів у кіберпросторі, шляхом його сегментації нормативно-правовими засобами, в межах окремих територіальних юрисдикцій.*

6. Відповідно до умов експерименту Мілґрема (*експеримент про покору*), до його проведення залучались: – «експериментатор» (*персона з повноваженнями коригувати хід експерименту*); – «вчитель» (*випробуваний, який лише частково уявляв сенс експерименту*); – «учень» (*актор, що грав свою роль та мав уявлення про сенс експерименту*). Які ж аналогії можна виділити, порівнюючи учасників експерименту про покору та ролі основних фігурантів акцій цькування в кіберпросторі (див. рис. 3)?

Доречно зауважити, що рольова структура учасників звичайного булінгу набагато простіше, ніж рольова структура учасників кібербулінгу. Так, в ситуації «традиційного» булінгу присутні три основні ролі: – жертва, переслідувач і свідок. При цьому рольова структура кібербулінгу передбачає присутність наступних акторів [9]: 1 – «агресори»; 2 – «помічники агресорів», які забезпечують підтримку процесу цькування; 3 – «жертви», це учасники процесу, що піддаються публічному цькуванню; 4 – «захисники і хамелеони», представники цих груп не стабільні і можуть мігрувати між групами «спостерігачів» і «агресорів» (*або «помічників агресорів»*); 5 – «спостерігачі» це найчисленніша і ситуативно не ангажована група учасників (*принаймні, на початковому етапі подій*).

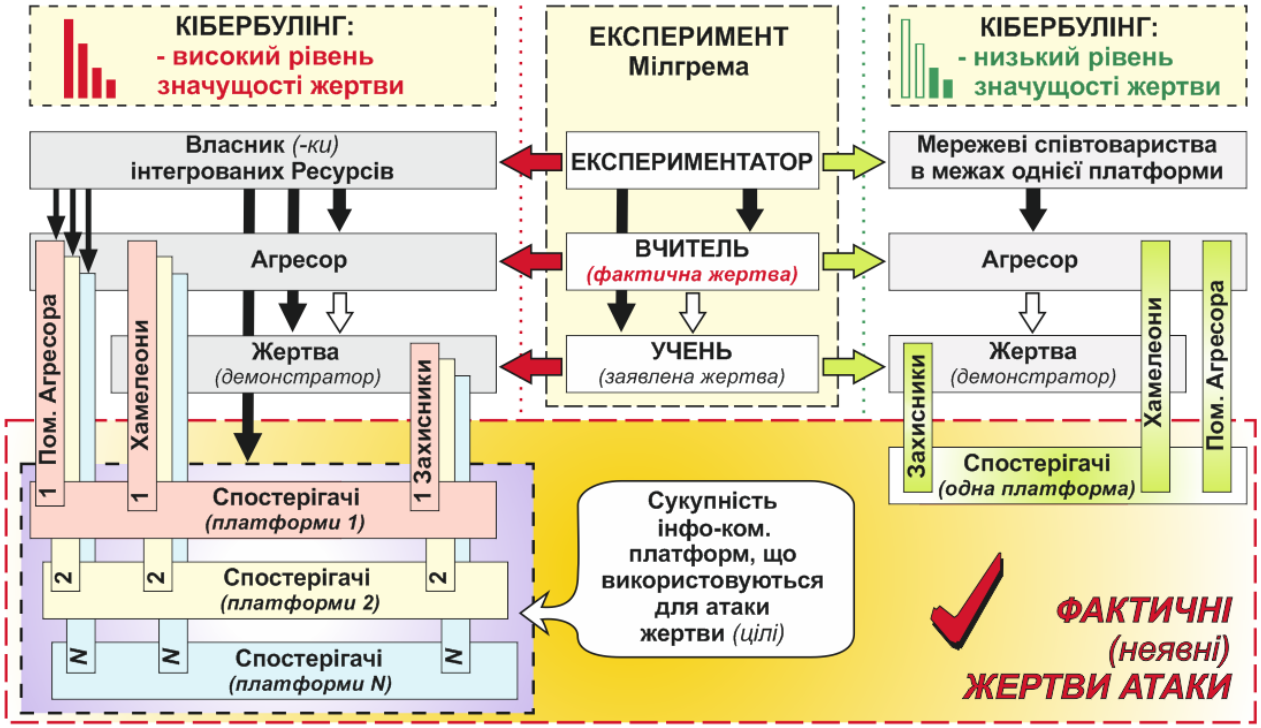


Рис. 3 – Передбачувана структура аналогій

Враховуючи фактор анонімності присутності представників 5-ї групи (*в ході здійснення заходів мережевої цькування*), можна зазначити, що на відміну від ситуації «традиційного» булінгу, в даному разі відсутній прямий контакт між «агресором» і «спостерігачами». При цьому, віддалена присутність учасників цієї групи та можливість швидкої зміни їх чисельності, визначають основну рольову функцію даної групи, заради якої, власне, організовується і підтримується все дійство. Таким чином, з огляду на швидкість, масштабованість та повторюваність процесу цькування, ця група учасників, де-факто, є кінцевою ціллю проведення всієї акції, за результатами якої повинні бути скориговані їх соціальний (мережевий) статус або соціальна (мережева) поведінка, або ж і те й інше одночасно. Тобто, в даному випадку, можна говорити про приховану насильницьку корекцію самоідентифікації учасників даної групи, шляхом послідовної і навмисної зміни їх уявлень, про норми соціальної (*мережевої*) поведінки в кіберпросторі.

Можна припустити, що саме таке трактування прихованих цілепокладань процесу кібербулінгу, є тією сполучною, котра забезпечує його концептуальну аналогію з основною парадигмою експерименту С. Мілгрема. У заявленому контексті неявній ролі «спостерігачів», роль «жертви» трансформується в інструмент для досягнення іншої, прихованої глобальної мети – забезпечення абсолютного домінування в найбільш представницької мережевій рольової групі. При такій постановці питання, роль «жертви», як це не дивно виглядає, це всього лише «ширма», основним призначенням якої, є нав'язування мережевого дискурсу із заздалегідь визначеним наративом і прихованим кінцевим цілепокладанням. Іншими словами,

в даному випадку, можна говорити про одну з можливих реалізацій стратегії непрямих дій, що адаптована до умов існування інформаційного суспільства і можливостям ведення інформаційних операцій в кіберпросторі.

А тепер, власне, про аналогії з «експериментом про покору». Вочевидь, що найпростіше виділити учасників групи «***вчитель***» (*головний випробовуваний цього експерименту*). Відносно процесу булінгу в кіберсфері, такі функції виконує «***агресор***». При цьому визначальною якістю представників цієї групи (*мережевого співтовариства*) повинна бути готовність «йти до кінця», в рамках реалізації всього комплексу заходів цькування, об'єднаних загальною стратегією дій. Крім того, група «***вчитель***» може поглинати і рольову категорію «***помічники агресорів***», яка в більшості випадків знайома з порядком та умовами проведення заходів травлі (рис. 3). Однак, у випадках проведення заходів з «глобальним рівнем впливу», представники цієї рольової групи, швидше за все, не будуть ознайомлені з кінцевою метою проведеної акції (*залучаються для завдань підготовки контенту або підтримки процесу цькування*). Іншими словами завдання акторів групи «***вчитель***» можуть бути фрагментовані, а їх використання обмежуватися рамками передбачених етапів/стадій мережевої атаки.

Також слід мати на увазі, що на відміну від умов експерименту С. Мілгрема, де «***учень***» мав певні уявлення про задум та хід експерименту, в випадку здійсненні заходів кібербулінгу маркована ***жертва*** атаки не має відповідних преференцій. В умовах мережевої травлі таку обізнаність можуть мати тільки представники рольової групи *«**хамелеони**»*, які виступають своєрідними каталізаторами або провокаторами потрібних дій ***жертви***. Перебуваючи, на різних стадіях атаки, в різних рольових іпостасях *(«спостерігач», «захисник жертви» або «помічник агресора»*), учасники рольової групи *«**хамелеони**»* є еквівалентом поведінкових якостей «***учня***» з експерименту Мілгрема. Вочевидь, що представники цієї групи повинні бути досить інформовані з приводу реалізованих заходів цькування (*принаймні, про заплановані стадії заходів травлі та маркери власної рольової міграції*). Зрозуміло, що краудсорсінг в даному випадку виключений, тому залучення фахівців відповідних кваліфікації, професійних якостей та мотивацій, можливо тільки на основі підряду або IT-аутсорсінгу (рис. 1-2).

Виникає логічне запитання: - хто ж визначає «***жертву***» та формує стратегію всієї акції (*тобто, правіла адміністрування контенту, стадії, час і масштаб проведення заходів*)?

В пошуках відповіді ми виходимо на повноваження «***експериментатора***» (*в термінології експерименту про покору*). Враховуючи тенденцію останніх років, до «зрощення» джерел контенту та основних складових ресурсного забезпечення інформаційних систем, вочевидь, що учасниками рольової групи «***експериментатор***», при високих рівнях ієрархічної важливості об'єктів кібертравлі, можуть бути тільки вигодоутримувачі (*або монополісти*) результатів подібного об'єднання. Крім того, незнання «***жертвою***» мережевого цькування факту присутності в схемі атаки (*заходів психологічного тиску*), учасників рольової категорії «***експериментатор***» (*в нашому випадку монополіста інтегрованих ресурсів*), додатково посилює існуючі аналогії з умовами проведення експерименту Мілгрема. При цьому, в заходах мережевого цькування на нижчих рангах ієрархії (*тобто в «традиційному» розумінні випадків кібербулінгу*), в ролі «***експериментатора***», нажаль, можуть виступати представники тимчасових мережевих співтовариств, які визначають локальну інформаційну повістку (*онлайн голосування, рейтингові оцінювання, опитування та ін.*) та формують потрібну для них форму уявлень, про норми мережевої поведінки в «їх» кіберпросторі.

3 Висновки

1. Об'єкти кібертравлі значною мірою корелюють з основними об'єктами впливу в ході ведення сучасних інформаційних операцій.

2. При реалізації кібербулінгу роль «жертви» зводиться до функцій демонстратора можливостей «агресора», стосовно змін мережевого статусу та коригування уявлень про норми мережевої поведінки представників найбільш численної рольової групи - «спостерігачів».

3. Головна мета представників рольової групи «агресор» при реалізації акцій кібербулінгу, це забезпечення абсолютного домінування в ролевій групі «спостерігачів».

4. Рівень ієрархічної значущості представників рольової групи «жертва», обумовлює необхідну ступінь інтеграції (*монополізації*) функцій адміністрування контенту і основних складових ресурсного забезпечення залучуваних інформаційно-комунікаційних систем.

5. Рівень значущості маркованої жертви атаки, в значній мірі визначає необхідний обсяг фінансової складової ресурсного забезпечення заходів кібертравлі.

6. Сучасні акції мережевого булінгу, особливо у випадках високих рівнів значущості об'єктів кібертравлі, мають багаторівневий та багатошаровий характер. Нейтралізація одного або ж відразу декількох шарів «атакуючого» контенту не змінює загального тренду кібертравлі і гарантує збереження необхідної інформаційної повістки.

7. Розмір фінансової складової ресурсного забезпечення агресора, визначає потенційний рівень складності структури заходів кібертравлі.

8. Характер поведінки і ролі основних учасників процесу кібертравлі, багато в чому ідентичні умовам та цілям проведення експерименту Мілгрема.

9. В залежності від фактичного рівня ієрархічної важливості об'єктів кібертравлі, можливі істотні відмінності в представництві групи «експериментатор».

10. При високих рівнях ієрархічної важливості об'єктів кібертравлі, представниками рольової групи «експериментатор», є ***власники*** інтегрованих активів (*перш за все інформаційних і технічних*).

11. Принциповою умовою «успішності» інформаційного впливу, в рамках проведення акцій булінгу в кіберпросторі, є забезпечення потрібного рівня диспаритету можливостей (ресурсів) агресора і жертви.

12. Одночасне використання атакуючою стороною можливостей відразу декількох інформаційно-комунікаційних платформ, є показовим прикладом забезпечення нерівності можливостей агресора і жертви.

Посилання

[1] Цькування [Електронний ресурс]: Wikipedia. Вільна енциклопедія. – Режим доступу: http://surl.li/omyd (дата звернення: 20.01.2021).

[2] What is cyberbullying? [Online]. Available: https://nuedusec.com/blog/cyberbullying/ Accessed on: January 07, 2021.

[3] Интернет-травля [Электронный ресурс]: Wikipedia. Свободная энциклопедия. – Режим доступа: http://surl.li/omyr (*дата обращения*: 29.01.2021).

[4] Закон України «Про основні засади забезпечення кібербезпеки України» № 2163-VIII від 5 жовтня 2017 року. [Електронний ресурс]. Режим доступу: https://zakon.rada.gov.ua/laws/show/2163-19

[5] Vdovenko, S., Danik, Y., & Faraon, S. (2019). Дефініційні проблеми термінології у сфері кібербезпеки і кібероборони та шляхи їх вирішення. *Комп'ютерні науки та кібербезпека*, (1), 18-30. https://doi.org/10.26565/2519-2310-2019-1-02 (дата звернення: 21.12.2020).

[6] Гайкова, В., & Малахов, С. (2020). Дослідження явища кібербулінгу і шляхів протидії його проявам. *Комп'ютерні науки та кібербезпека*, 1(1), 14-32. https://doi.org/10.26565/2519-2310-2020-1-02 (*дата звернення: 30.12.2020*).

[7] Макаренко С. И. Информационные операции: Фрагмент монографии. [Електронний ресурс] / С. И. Макаренко – Режим доступа: https://psyfactor.org/psyops/makarenko-304.htm. *Дата обращения: Декабрь 31, 2020.*

[8] Інформаційна операція [Електронний ресурс]: Wikipedia. Вільна енциклопедія. – Режим доступу: http://surl.li/onen (*дата звернення: 29.01.2021*).

[9] Ролевая структура кибербуллинга [Электронный ресурс]: Wikipedia. Свободная энциклопедия. – Режим доступа: http://surl.li/ocjv (*дата обращения*: *29.01.2021*).

[10] Овсянникова А. Эксперимент Милгрэма полвека спустя [Электронный ресурс] / А. Овсянникова – Режим доступа: http://surl.li/onca. *Дата обращения: 29.01.2021.*

[11] Експеримент Мілгрема [Електронний ресурс]: Wikipedia. Вільна енциклопедія. – Режим доступу: http://surl.li/oubj (дата звернення: 29.01.2021).

[12] Conducting the Milgram experiment in Poland, psychologists show people still obey. [Online]. Available: https://www.eurekalert.org/pub_releases/2017-03/sfpa-ctm030917.php. Accessed on: January 08, 2021.

[13] Информационное общество [Электронный ресурс]: Wikipedia. Свободная энциклопедия. – Режим доступа: http://surl.li/onlr (*дата обращения: 29.01.2021*).

[14] Информационное общество. Новая философская энциклопедия: в 4 т. / Электронная библиотека Института философии РАН. – Режим доступа: http://surl.li/onnv. *Дата обращения: Январь 29, 2021.*

[15] Террористический акт в редакции Charlie Hebdo [Электронный ресурс]: Wikipedia. Свободная энциклопедия. – Режим доступа: http://surl.li/qzgt *(дата обращения: 29.01.2021).*

[16] Политический кризис в Венесуэле (с 2019 года) [Электронный ресурс]: Wikipedia. Свободная энциклопедия. – Режим доступа: http://surl.li/qyzc *(дата обращения: 29.01.2021).*

[17] Дідух Т.М. Глобальні ризики використання IT-аутсорсингу [Електронний ресурс] / Т. М. Дідух – Режим доступу: http://global-national.in.ua/archive/20-2017/8.pdf. *Дата звернення: 29.01.2021.*
[18] Байден побил рекорд по анонимным пожертвованиям на предвыборную кампанию [Электронный ресурс] – Режим доступа: https://www.rbc.ru/politics/24/01/2021/600cd63c9a794789ada2136a Дата обращения: 29.01.2021.

Рецензент: Владимир Хома, д.т.н., проф., Опольский Политехнический Университет, Ополе, Польша.
E-mail: xoma@wp.pl

Поступила: Январь 2021.

Авторы:
Валерия Гайкова, студентка магистратуры кафедры безопасности информационных систем и технологий, ХНУ имени В. Н. Каразина, Харьков, Украина. E-mail: valeriagaikova98@gmail.com
Сергей Малахов, к.т.н., с.н.с., факультет комп'ютерних наук, ХНУ имени В. Н. Каразина, Харьков, Украина.
E-mail: mailgate@meta.ua

Анализ факторов и условий реализации кибербуллинга, с учетом возможностей современных информационных систем.

Аннотация. Представлен перечень основных обобщений и факторов, характерных для различных вариантов и условий осуществления акций сетевой травли (кибербуллинга). Предложена аргументация определенных аналогий и совпадений между процессом сетевой травли, а также условиями и целями проведения современных информационных операций. Обращено внимание на то, что условия кибертравли во многом совпадают с парадигмой эксперимента С. Милгрэма. Предложена аргументация сути некоторых аналогий между проявлениями кибербуллинга и основной фабулой эксперимента о повиновении. Подчеркнуто, что уровень иерархической важности объектов кибертравли, обусловливает степень необходимой интеграции функций администрирования контента и основных составляющих ресурсного обеспечения современных информационных систем. Отмечено, что интеграция в одном центре принятия решений основных составляющих ресурсного обеспечения и функций администрирования контента, является серьезной проблемой даже для уровня отдельных государств, не говоря уже о персональном уровне воздействия. Обращено внимание на тот факт, что кибербуллинг может реализовываться, как путем использования возможностей отдельных информационных технологий, так и воплощать концепцию интегрированной атаки. Подчеркнуто, что явление кибербуллинга является значительно недооцененным и поэтому, представляет собой серьезную проблему современности.

Ключевые слова: буллинг; кибербуллинг; информационная повестка; контент; информационные технологии; техническая платформа; сетевое поведение; сетевой статус; эксперимент Милгрэма.

Reviewer: Volodymyr Khoma, Dr. of Sciences (Eng.), Full Prof., The Opole University of Technology, Opole, Poland.
E-mail: xoma@wp.pl

Received: January 2021.

Authors:
Valeriia Haikova, Student of the Department of Security of Information Systems and Technologies, V.N. Karazin Kharkiv National University, Ukraine. E-mail: valeriagaikova98@gmail.com
Serhii Malakhov, Ph.D., Senior Researcher, Computer Science Department, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine. E-mail: mailgate@meta.ua

Analysis of factors and conditions of implementation for cyberbullying, taking into account the capabilities of modern information systems.

Annotation. A list of the main generalizations and factors characteristic of various options and conditions for the implementation of actions of network bullying (cyberbullying) is presented. The argumentation of certain analogies and coincidences between the process of network bullying, as well as the conditions and goals of modern information operations is proposed. Attention drawn to the fact that the conditions of cyberbullying largely coincide with the paradigm of S. Milgram experiment. The argumentation of the essence of some analogies is offered between the manifestations of cyberbullying and the main concept of the experiment of S. Milgram's experiment. It is emphasized that the level of hierarchical importance of cyberbullying objects, determines the degree of integration of content administration functions and the main components of the resource support of information systems. It is noted that integration in one decision-making center all major technical resources and content administration functions, is a serious problem even for the level of individual states, not to mention the personal level of exposure. Attention is drawn to the fact that cyberbullying can be implemented, both by using the capabilities of individual information technologies, and embody the concept of integrated attack. It was emphasized that the phenomenon of cyberbullying is significantly underestimated and therefore, represents a serious problem of our time.

Keywords: Bullying; Cyberbullying; Information agenda; Content; Information Technology; Technical platform; Network behavior; Network status; Milgram's experiment.

УДК 681.04 DOI: 10.26565/2519-2310-2021-1-05

АЛГОРИТМ ПОБУДОВИ СТРУКТУРИ СУММАТОРУ ДВОХ ЗАЛИШКІВ ЧИСЕЛ ПО МОДУЛЮ

Михайло Багмут, Катерина Кузнєцова, Людмила Горбачова

Харківський національний університет імені В.Н. Каразіна, пл. Свободи, 4, Харків, 61022, Україна
Mikhail56@ukr.net, kate7smith12@gmail.com, lusyag23@gmail.com

Рецензент: В'ячеслав Калашников, д.ф.-м.н., проф., Технологічний університет Монтеррея, 64849 Монтеррей, Нуево-Леон, Мексика
kalash@itesm.mx

Надійшла: Лютий 2021.

***Анотація**: Відомо, що завдання побудови структури суматора, який працює по довільному модулю m_i і виконаний на логічних елементах з двома стійкими станами, є актуальною науково-прикладною задачею. Даний тип суматора використовується, як в позиційній двійковій системі числення (ПСЧ), так і в непозиційній системі числення в залишкових класах (СЗК). Якщо залишки a_i і b_i чисел $A = (a_1 || a_2 || ... || a_i || ... || a_k)$ і $B = (b_1 || b_2 || ... || b_i || ... || b_k)$, представлених в СЗК, дані в двійковій ПСС, тоді суматор двох залишків ai і bi по модулю ті являє собою сукупність з $n = [\log2 (m_i - 1) +1]$ двійкових однорозрядних суматорів (ДОС). При цьому всі ДОС об'єднані між собою зв'язками, подібно зв'язків позиційних двійкових суматорів. Метою статті є розробка алгоритму побудови структури суматору двох залишків a_i і b_i чисел A та B для довільного значення m_i модуля СЗК. Це процес реалізований, шляхом організації нових міжрозрядних зв'язків ДОС, з використанням позиційного суматора по модулю $M = 2^n - 1$. Відзначено, що існують спеціальні набори модулів, які застосовуються при обробці даних в СЗК. Так, при виконанні операції модульного складання залишків чисел, може використовуватися один з 3-х взаємно попарно простих чисел (виду $M = 2^n - 1$, $M = 2^n$ або $M = 2^n + 1$). Показано, що для синтезу суматора по модулю m_i СЗК, в структурі суматора по модулю М, необхідно відповідним чином сформувати додаткові зв'язки.*

***Ключові слова:** непозиційна система числення; однобітовий суматор; система залишкових класів; суматор залишків; цілочисельні арифметичні операції.*

1 Вступ

Виконання цілочисельних арифметичних операцій додавання, віднімання і множення двох чисел $A=(a_1||a_2||...||a_i||...||a_k)$ і $B=(b_1||b_2||...||b_i||...||b_k)$ в непозиційній системі числення в залишкових класах (СЗК) здійснюється шляхом реалізації відповідних залишків a_i і b_i чисел A і B за відповідними модулями (основами) m_i $(i=\overline{1,k})$ незалежно один від одного і паралельно в часі по кожному з k основ СЗК [1-3]. Відомо, що основною операцією, яка реалізується комп'ютерною системою (КС), як в позиційній двійковій системі числення (ПСЧ), так і в СЗК, є операція додавання двох чисел. В цьому аспекті одним з головних компонент КС є суматор двох чисел. Зокрема, компонентами КС в СОК, поряд з позиційними суматорами, є також суматори двох чисел по модулю m_i. В СОК модульна операція складання $(a_i+b_i)\bmod m_i$ реалізується на основі використання малорозрядних суматорів за модулем m_i. Один з методів реалізації модульної операції складання $(a_i+b_i)\bmod m_i$ ґрунтується на використанні структур малорозрядних двійкових суматорів [4]. Даний підхід надає широкий вибір варіантів реалізації структури таких суматорів. Це дозволяє в повній мірі використовувати наявний практичний досвід проектування і вибору структур довічних суматорів.

Таким чином, важливою і актуальною науково-прикладною задачею є задача по-будови структур суматорів, що працюють за довільним модулем m_i, виконаних на логічних елементах з двома стійкими станами. Якщо залишки a_i і b_i чисел $A=(a_1||a_2||...||a_i||...||a_k)$ та $B=(b_1||b_2||...||b_i||...||b_k)$ по модулю m_i СЗК представлені в двійковій ПСС, тоді суматор двох залишків a_i і b_i по модулю m_i представляє собою послідовну сукупність з $n=[\log_2(m_i-1)+1]$

двійкових однорозрядних суматорів (ДОС), які об'єднані між собою зв'язками, подібно зв'язків позиційних двійкових суматорів.

Мета статті – представити алгоритм побудови структури суматора двох залишків a_i і b_i чисел *A* та *B*, для довільного значення модулю m_i СЗК, шляхом нової організації міжрозрядних зв'язків ДОС, з використанням позиційного суматора по модулю $M = 2^n - 1$.

2 Основна частина

Відомо, що двійкові суматори в ПСЧ мають фіксовану величину модуля, рівну значенню числа $M = 2^n - 1$. Дана обставина виключає можливість їх безпосереднього використання для довільного модуля m_i СЗК. Якщо модуль *M* суматора, відрізняється від модулю m_i СЗК на одиницю, то існує два варіанти практичної реалізації операції додавання $(a_i + b_i) \bmod m_i$ двох залишків a_i і b_i по модулю СЗК. Розглянемо більш детально ці варіанти [1].

Перший варіант. Має місце наступне співвідношення модулів: $M = m_i + 1$. В цьому випадку розглянемо можливі умови виконання процедур реалізації модульного складання двох залишків чисел.

Перша умова. Якщо залишки чисел відповідають вимозі $a_i + b_i < m_i$, то корекція результату операції модульного складання не потрібна.

Друга умова. Якщо виконується співвідношення $a_i + b_i = m_i$, то вміст суматора по модулю обнуляється (*скидається*).

Третя умова. Якщо виконується співвідношення $a_i + b_i = M$, то при наявності нуля суматора, в молодшому розряді суматора встановлюється одиниця з одночасним обнулінням (*скиданням*) інших двійкових розрядів суматора.

Четверта умова. Якщо $a_i + b_i > M$, то результат операції буде дорівнювати значенню $a_i + b_i - M = (a_i + b_i - m_i - 1) \bmod m_i = a_i + b_i - 1$. В цьому випадку, в суматорі, ланцюг перенесення з виходу старшого розряду переходить на вхід молодшого розряду.

Другий варіант. Має місце наступне співвідношення модулів $M = m_i - 1$. У цьому разі є необхідність в коректуванні результату операції модульного складання залишків.

Перша умова. Якщо виконується співвідношення $a_i + b_i < m_i$, то корекція результату операції не вимагається.

Друга умова. Якщо виконується співвідношення $a_i + b_i = m_i$, то результат операції дорівнює одиниці. Здійснюється попередня видача сигналу, щодо перенесення вмісту зі старшого розряду. Вміст суматора скидається.

Третя умова. Якщо $a_i + b_i = M$, то обнуляється вміст суматора з паралельним занесенням в суматор одиниці.

Четверта умова. Якщо $a_i + b_i > M$, то отримуємо результат операції модульного додавання в наступному вигляді: $a_i + b_i - M = (a_i + b_i + 1) \bmod m_i$.

Вочевидь, що при розглянутих варіантах співвідношень між модулями m_i та *M*, реалізація операції додавання залишків $(a_i + b_i) \bmod m_i$, здійснюється відносно просто. Але відзначимо, що в разі наявності різниці значень величин модулів m_i і *M*, операція модульного складання $(a_i + b_i) \bmod m_i$ двох залишків чисел, є досить складним завданням. Це призводить до необхідності постановки і вирішення окремого завдання побудови структури суматора за довільним модулем m_i.

3 Структура суматора за довільним модулем

Розглянемо метод побудови суматорів за довільним модулем m_i СЗК, що заснований на структурі суматора по модулю $M = 2^n - 1$, шляхом організації і використання додаткових між-

розрядних зв'язків $X_{\downarrow i\uparrow j}$, в загальному випадку, між j-м та i-м ДОС суматора, по модулю *M*. Сформулюємо задачу побудови суматора 2-х залишків чисел по модулю, наступним чином.

Нехай задана довільна вихідна структура *n* - розрядним двійкового суматора по модулю $M = 2^n - 1$ (рис. 1). Необхідно створити структуру суматора для реалізації операції складання двох залишків чисел за довільним модулем m_i СЗК. Іншими словами, необхідно забезпечити, умову, щоб суматор за модулем $M = 2^n - 1$, виконував операцію додавання двох залишків, a_i і b_i чисел, $A = (a_1 \| a_2 \| ... \| a_i \| ... \| a_k)$ та $B = (b_1 \| b_2 \| ... \| b_i \| ... \| b_k)$, по модулю m_i. Це досягається шляхом введення додаткових зв'язків $X_{\downarrow i\uparrow j}$, в суматорі за модулем *M (де знак* $X_{\downarrow i\uparrow j}$ *позначає зв'язок між виходом* j*-го та входом* i*-го ДОС)*.

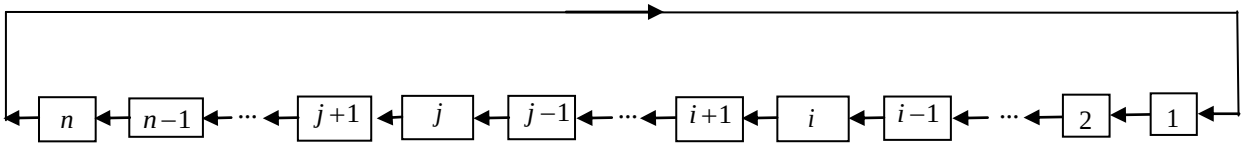


Рис. 1 – Структура двійкового суматора по модулю $M = 2^n - 1$

Задача побудови суматора за модулем m_i СЗК, буде здійснена шляхом вирішення задачі встановлення в суматорі за модулем, додаткових міжрозрядних зв'язків між певними ДОС виду $X_{\downarrow i\uparrow j}$. Для побудови суматора за модулем m_i СЗК, в структурі суматора необхідно, між певною парою ДОС (*вихідного суматора по модулю M*), сформувати додаткові зв'язки $X_{\downarrow i\uparrow j}$ таким чином, щоб за допомогою суматора по модулю *M*, здійснювалася операція додавання двох залишків чисел по модулю m_i. Схема організації додаткової зв'язку $X_{\downarrow i\uparrow j}$ між виходом j-го та входом i-го ДОС, представлена нижче, на рис. 2 ($j > i$).

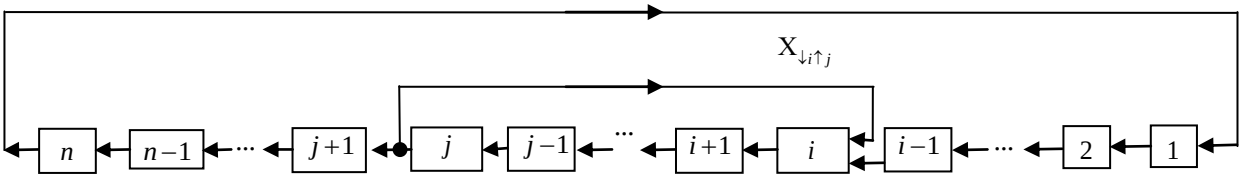


Рис. 2 – Схема двійкового суматора по модулю $M = 2^n - 1$ з одним зворотнім зв'язком

Розглянемо вплив наявності одного додаткового зв'язку/зв'язків $X_{\downarrow i\uparrow j}$ на величину вмісту суматора за модулем *M*.

4 Структура суматора по модулю *M* з додатковими зв'язками $X_{\downarrow i\uparrow j}$

Розглянемо вплив одного додаткового зв'язку $X_{\downarrow i\uparrow j}$, який встановлюється між виходом j-го та входом i-го ДОС в суматорі за модулем $M = 2^n - 1$ (рис. 2), на величину G_L вихідного вмісту суматору. Покажемо, що число $L = \{l_i\}$, $i = \overline{1, n}$, що є вмістом G_L суматора, при введенні одного додаткового зв'язку $X_{\downarrow i\uparrow j}$, зменшується на величину $\Delta G_L = 2^{i-j-2} \cdot \sum_{m=j+1}^{n} 2^m \cdot l_m$ [1]. При цьому, слід зазначити, що введення додаткового зв'язку $X_{\downarrow i\uparrow j}$, переводить обчислення значень з двійкової системи числення (СЧ), в якій працює суматора по модулю *M*, в поліадичну СЧ з основами $\tau_1, \tau_2, ..., \tau_k$ та модулем $M = \tau_1 \cdot \tau_2 \cdot ... \cdot \tau_k - 1$. В цьому випадку, одиниця m-го розряду числа $L = \{l_i\}$, $i = \overline{1, k}$, визначається наступним чином:

$$G_L = \sum_{m=1}^{k} l_m \cdot \prod_{i=1}^{m-1} \tau_i = l_1 \cdot \tau_2 \cdot \tau_3 \cdot ... \cdot \tau_k + l_2 \cdot \tau_3 \cdot \tau_4 \cdot ... \cdot \tau_k + ... + l_{k-2} \cdot \tau_{k-1} \cdot \tau_k + l_{k-1} \cdot \tau_k + l_k . \quad (1)$$

Співвідношення (1) можна представити в наступному вигляді:

$$G_L = l_1 \cdot \prod_{i=2}^{k} \tau_i + l_2 \cdot \prod_{i=3}^{k} \tau_i + ... + l_{k-2} \cdot \prod_{i=k-1}^{k} \tau_i + l_{k-1} \cdot \prod_{i=k}^{k} \tau_i + l_k . \quad (2)$$

У двійковій СЧ ($\tau_1 = \tau_2 = ... = \tau_k = 2$) вираз (2) прийме наступний вигляд:

$$G_L = \sum_{m=1}^{k} l_m \cdot 2^{k-m} = l_1 \cdot 2^{k-1} + l_2 \cdot 2^{k-2} + ... + l_{k-1} \cdot 2 + l_k . \quad (3)$$

В разі відсутності в суматорі додаткових зв'язків $X_{\downarrow i \uparrow j}$, величина G_L, вмісту суматора, дорівнює (*вираз 4*):

$$G_L = \sum_{m=1}^{n} l_m \cdot q_m , \quad (4)$$

де значення l_m є число одиниць, що містяться в m -ому розряді двійкового суматора, а значення q_m є вага m-го розряду двійкового суматора, який визначається положенням двійкового розряду суматора (рис. 1).

Якщо є додаткова зв'язок ($X_{\downarrow i \uparrow j}$), який об'єднує виконавчі розряди суматора з номерами від i до j, в єдиний (*узагальнений*) розряд суматора за модулем

$$\tau_{ij} = 2^{j-(i-1)} - 1 = 2^{j-i+1} - 1, \quad (5)$$

то, тоді, вага q_m кожного розряду суматора з номерами від $(i-1)$-го до першого (*молодшого розряду суматора*), буде дорівнює значенню $q_m = 2^{m-1}$ $(m = \overline{1, i-1})$ (*див. рис. 3*).

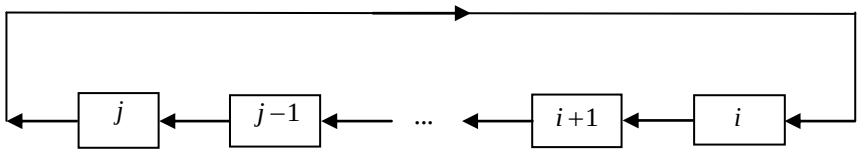


Рис. 3 – Схема узагальненого $(j-i+1)$-го розряду суматора по модулю τ_{ij}

Виходячи зі структури суматора по модулю (рис. 4), вага розрядів суматора з номерами від $(j+1)$-го до n (*старшого розряду суматора*) буде визначатися виразом (6):

$$q_m = 2^{i-1} \cdot \tau_{ij} \cdot 2^{m-j-1} . \quad (6)$$

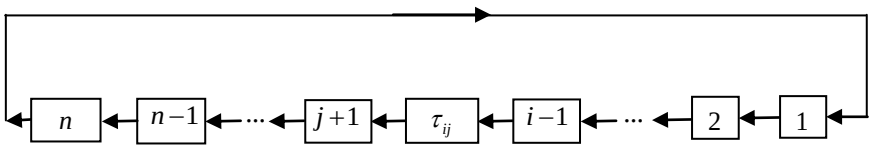


Рис. 4. Еквівалентна схема суматора по модулю з додатковим зв'язком $X_{\downarrow i \uparrow j}$

Враховуючи на співвідношення (5), вираз (6) можна представити в наступному вигляді

$$q_m = 2^{m+i-j-2} \cdot \tau_{ij} = 2^{m+i-j-2} \cdot (2^{j-i+1} - 1) = 2^{m-1} - 2^{m+i-j-2} . \quad (7)$$

Величина G_L числа L дорівнює різниці (8)

$$G_L = \sum_{m=1}^{n} l_m \cdot 2^{m-1} - \sum_{m=j+1}^{n} l_m \cdot 2^{m+i-j-2}. \quad (8)$$

Зі співвідношення (8) видно, що введення в суматор одного додаткового зв'язку. виду $X_{\downarrow i \uparrow j}$, зменшує його вміст G_L на величину, що дорівнює значенню

$$\Delta G_L = \sum_{m=j+1}^{n} l_m \cdot 2^{m+i-j-2} = 2^{i-j-2} \cdot \sum_{m=j+1}^{n} l_m \cdot 2^m.$$

Таким чином, при введенні одного додаткового зв'язку $X_{\downarrow i \uparrow j}$ початковий вміст суматора по модулю M , зменшується на величину ΔG_L, де

$$\Delta G_L = 2^{i-j-2} \cdot \sum_{m=j+1}^{n} l_m \cdot 2^m. \quad (9)$$

Одним з важливих ефектів виконаних досліджень є те, що при введенні додаткового зв'язку виду $X_{\downarrow i \uparrow j}$, значення величини модуля $M = 2^n - 1$, вихідного суматора, зменшується на величину

$$\Delta M = 2^{i-j-2} \cdot \sum_{m=j+1}^{n} S_m \cdot 2^m, \quad (10)$$

де S_m – значення *m*-го розряду числа, що міститься в суматорі [1]. При цьому, вочевидь, що діапазон чисел за модулем *M*, які представляються, зменшується на величину $\triangle M$. Ця обставина надає можливість, за рахунок введення в суматор за модулем *M* певних зв'язків (*або одного додаткового зв'язку*), зменшити величину *M* модуля до необхідного значення модуля m_i СЗК. В цьому випадку має виконуватися наступна умова

$$m_i = M - \Delta M \text{ или } m_i = 2^n - 1 - 2^{i-j-2} \cdot \sum_{m=j+1}^{n} l_m \cdot 2^m. \quad (11)$$

Виходячи з виразу (11), чисельне значення заданого модуля СЗК буде визначено набором значень n, j та i. Таким чином, необхідно визначити значення n, j та i, як такі, що задовольняють виконання рівності (11).

5 Алгоритм побудови суматорів за довільним модулем m_i

У загальному вигляді, в представленні модулю m_i, вміст двійкових розрядів має наступний вигляд:

$$m_i = S_n \cdot 2^{n-1} + S_{n-1} \cdot 2^{n-2} + ... + S_2 \cdot 2 + S_1, \quad (12)$$

та нехай початковий вміст G_L суматора по модулю $M = 2^n - 1$ має вид:

$$G_L = l_n \cdot 2^{n-1} + l_{n-1} \cdot 2^{n-2} + ... + l_2 \cdot 2 + l_1. \quad (13)$$

В цьому випадку, для виконання умови (11) в суматор за модулем $M = 2^n - 1$ вводять додаткові зв'язку $X_{\downarrow i \uparrow j}$. При цьому *(див. вирази (12) і (13))*, i-й ДОС прийме значення $l_i + c_{i-1} + x_{ij}$, де l_i – це вміст l_i -го ДОС вихідного стану суматора по модулю *M*; c_i – значення сигналу перенесення вмісту $(i-1)$-го ДОС в i-й ДОС; x_{ij} – це значення l_j вмісту j-го ДОС суматора; S_i – значення вмісту i-го двійкового розряду модуля m_i СЗК. У цьому випадку маємо (14):

$$l_i + c_{i-1} + x_{ij} + S_i = l_i. \quad (14)$$

Враховуючи одно з властивостей модуля суматора (*модуль суматора є його другим нулем*) вираз (14) набуває вигляду (15)

$$c_{i-1} + x_{ij} + S_i = 0, \quad (15)$$

що для двійкового представлення чисел, рівнозначно співвідношенню (16)

$$S_i = c_{i-1} + x_{ij}. \quad (16)$$

Беручи до уваги все вищезазначене, та виходячи з результатів аналізу виразів (15) і (16), можна зробити наступні висновки, стосовно впливу додаткових встановлених зв'язків $X_{\downarrow i \uparrow j}$, на вміст позиційного суматора по модулю $M = 2^n - 1$.

Вплив встановлених зв'язків $X_{\downarrow i \uparrow j}$, на вміст позиційного суматора по модулю $M = 2^n - 1$, обумовлює правила синтезу структури суматора по модулю m_i СОК.

Сформулюємо ***правила*** побудови структури суматора двох залишків чисел по модулю m_i.

1. Наявність додаткових зв'язків $X_{\downarrow i \uparrow j}$ і їх вплив на зменшення величини $\Delta M = 2^{i-j-2} \cdot \sum_{m=j+1}^{n} S_m \cdot 2^m$ вмісту G_L суматора по модулю $M = 2^n - 1$ не залежить від вихідного стану цього позиційного суматора.

2. У позиційному суматорі по модулю $M = 2^n - 1$ додатковий зв'язок $X_{\downarrow i \uparrow j}$ має місце (*вихід доп. зв'язку* $X_{\downarrow i \uparrow j}$) лише для i-х S_i ДОС (S_i; $i = \overline{2, n}$), відповідних нульових значень двійкової кодової комбінації модулю m_i, за яким працює суматор.

3. У двійкових розрядах S_i кодової комбінації, модулю m_i суматора, які відповідають одиничним значенням, додаткові зв'язки $X_{\downarrow i \uparrow j}$ повинні бути відсутніми. Тобто необхідно, щоб виконувалася умова $x_{ij} = 0$.

4. Вміст G_L суматора не зміниться, якщо додатковий зв'язок $X_{\downarrow i \uparrow n}$ буде «взят» з виходу старшого ДОС (старшого розряду суматора).

5. При введенні одного додаткового зв'язку $X_{\downarrow i \uparrow j}$, модуль $M = 2^n - 1$ позиційного суматора зменшується на величину $\Delta M = 2^{i-j-2} \cdot \sum_{m=j+1}^{n} S_m \cdot 2^m$.

6. Число, яке описує вміст G_L суматора по модулю $M = 2^n - 1$ з одним додатковим зв'язком $X_{\downarrow i \uparrow j}$, може приймати не більше $(n + i - j)$ різних числових значень, що відповідають $(n + i - j)$ можливим конструкціям суматору по модулю m_i. Фактично, майже для кожної основи m_i СЗК, можна побудувати кілька варіантів структур суматорів. В цьому випадку, може виникнути необхідність вибору «найкращої» структури суматору за модулем m_i, із всіх можливих варіантів. Однак, в рамках даної роботи, оптимізація та вибір суматора по модулю m_i, з усіх можливих варіантів, не розглядаються.

Таким чином, суть алгоритму побудови суматорів за модулем m_i СЗК полягає в наступному: - в вихідному суматорі за модулем $M = 2^n - 1$, на підставі вищенаведених правил, формуються додаткові зв'язки суматора, а використання додаткових зв'язків дозволяє реалізувати вихідний суматор для виконання операції додавання вирахувань чисел за модулем m_i СЗК. При цьому слід мати на увазі, що введення додаткових зв'язків $X_{\downarrow i \uparrow j}$ змінює категорію ваги окремих розрядів суматора та зменшує величину модуля від M до m_i.

Алгоритм побудови суматора по модулю m_i СЗК складається з наступних операцій:

1. Представлення структури суматору по модулю $M = 2^n - 1$, де $n = [\log_2(m_i - 1)] + 1$. В цьому випадку визначається розрядність n (*кількість ДОС*) суматора по модулю m_i;

2. Визначення двійкових розрядів S_i суматора для яких виконується умова $S_i = 0$. Процес визначення умови $S_i = 0$, проводиться виходячи з подання модуля числа m_i, в двійковому коді;

3. Як правило, додатковий зв'язок починається з виходу n-го (старшого) двійкового розряду ($j = n$);

4. Додатковий зв'язок $X_{\downarrow i \uparrow j}$ надходить на вхід ДОС, для якого $S_i = 0$ *(див. п. 2 методу)*.

6 Приклади виконання операції додавання залишків чисел за модулем

Відповідно до представленого вище алгоритму, розглянемо приклади синтезу суматорів для різних модулів m_i СЗК.

Приклад 1. Нехай $m_i = 11$. Розглянемо етапи синтезу суматора за модулем $m_i = 11$ СЗК.

1. Відповідно до величини $m_i = 11$ модуля СЗК, визначимо кількість n ДОС. Для модуля $m_i = 11$ маємо, що $n = [\log_2(11-1)]+1 = 4$. При цьому, структура суматора по модулю буде $n = [\log_2(11-1)]+1 = 4$ мати вигляд, який наведено на рис. 5.

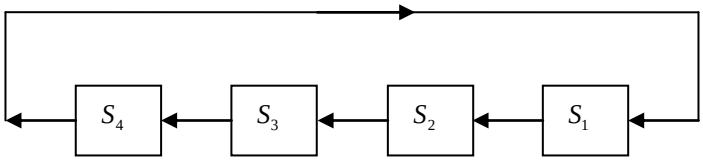


Рис. 5 – Вихідна структура суматора за модулем $M = 2^n - 1$

2. Для синтезу суматора по модулю $m_i = 11$ СЗК, попередньо, визначимо значення двійкових розрядів S_i суматора, в запису модуля $m_i = 11$ яких, містяться нулі, тобто для випадку, коли $S_i = 0$. Таким розрядом буде третій розряд, тобто $S_3 = 0$, так як в двійковому коді модуль $m_i = 11$ має наступний вигляд 1011.

3. Виходячи з того, що $S_3 = 0$, додатковий зв'язок в суматорі матиме вигляд $X_{\downarrow 3 \uparrow 4}$.

Структура суматора за модулем $m_i = 11$ представлена нижче, на рис. 6.

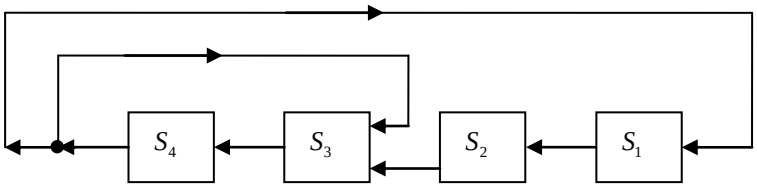


Рис. 6 – Структура суматора за модулем $m_i = 11$

Для структури суматора, що представлена на рис. 6, визначимо значення модуля $M = m_i$ СЗК. Попередньо розглянемо частину структури (*див. рис. 7*) цього суматора.

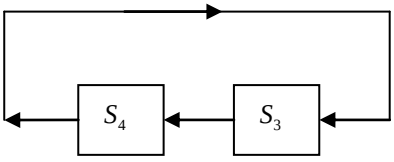


Рис. 7 – Частина структури суматора за модулем m_i

Для цієї частини структури суматора значення модуля M_1 визначиться, як $M_1 = \tau_4 \cdot \tau_3 - 1$. Значення модуля $M = m_i$ СЗК суматора (*рис. 6*) визначиться наступним чином (*див. рис. 8*)

$$m_i = M_1 \cdot \tau_2 \cdot \tau_1 - 1 = (\tau_4 \cdot \tau_3 - 1) \cdot \tau_2 \cdot \tau_1 - 1 = (2 \cdot 2 - 1) \cdot 2 \cdot 2 - 1 = 11.$$

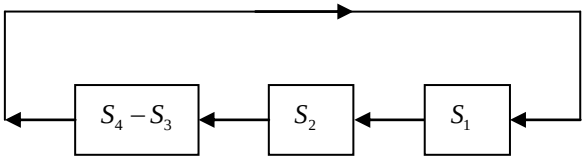


Рис. 8 – Частина структури суматора за модулем m_i

Таким чином, синтез суматора за модулем $m_i = 11$ проведений вірно.

Приклад 2. Нехай m_i =37. Етапи синтезу суматора по модулю СЗК.

1. Відповідно до величини модуля $m_i = 37$, визначимо кількість n двійкових розрядів (*кількість ДОС*) суматора по модулю. Для модуля $m_i = 37$ маємо, що $n = [\log_2(37-1)]+1 = 6$.

2. Для синтезу структури суматора по модулю $M = 63$ введемо додаткові зв'язки $X_{\downarrow i \uparrow j}$. Попередньо визначимо двійкові розряди S_i суматора, в яких в запису модуля m_i, в двійковому коді, містяться нулі, тобто $S_i = 0$. Так як модуль в двійковому коді має вигляд 100101, то нульовими двійковими розрядами суматора будуть наступні: $S_2 = 0$, $S_4 = 0$ та $S_5 = 0$.

3. Спираючись на отримані в п. 2 результати (*опис метода синтезу суматора по модулю* m_i), введемо в суматор за модулем $M = 2^n - 1$ три додаткові зв'язку: $X_{\downarrow 5 \uparrow 6}$, $X_{\downarrow 4 \uparrow 6}$ та $X_{\downarrow 2 \uparrow 6}$. У цьому випадку структура суматора за модулем $m_i = 37$ має вид, що представлений на рис. 9.

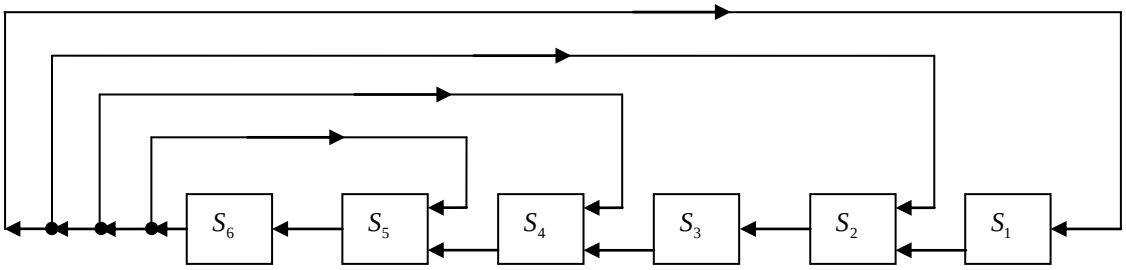


Рис. 9 – Структура суматора за модулем $m_i = 37$

Визначимо для даної структури (рис. 9) значення модуля $M = m_i$ СЗК. Для цього, попередньо, складемо ряд структур окремих частин суматора, що представлений на рис. 9.

Перша частина структури суматора представлена на рис. 10.

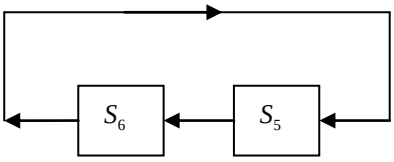


Рис. 10 – Перша частина структури суматора за модулем m_i

Для 1-ої частини структури суматора, модуль M_1 визначиться наст. чином $M_1 = \tau_6 \cdot \tau_5 - 1$.

Друга частина структури суматора представлена на рис. 11. Для цієї структури суматора, модуль M_2 визначається наступним чином: $M_2 = M_1 \cdot \tau_4 - 1 = (\tau_6 \cdot \tau_5 - 1) \cdot \tau_4 - 1$.

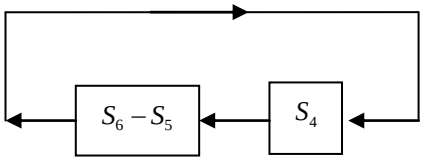


Рис. 11 – Друга частина структури суматора за модулем m_i

Третя частина структури суматора представлена на рис. 12.

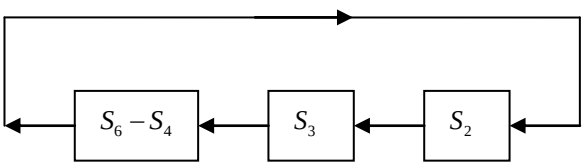


Рис. 12 – Третя частина структури суматора за модулем m_i

Для третьої частини структури суматора, модуль M_3 визначиться наступним чином $M_3 = M_2 \cdot \tau_3 \cdot \tau_2 - 1 = \left[(\tau_6 \cdot \tau_5 - 1) \cdot \tau_4 - 1\right] \cdot \tau_3 \cdot \tau_2 - 1$.

Четверта частина структури суматора представлена на рис. 13

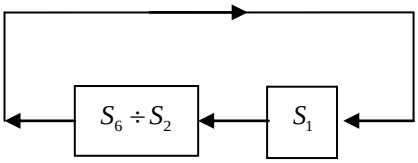


Рис. 13 – Четверта частина структури суматора за модулем m_i

Значення модуля $M = m_i$ СЗК визначається наступним чином $m_i = M_3 \cdot \tau_1 - 1 = \left\{\left[(\tau_6 \cdot \tau_5 - 1) \cdot \tau_4 - 1\right] \cdot \tau_3 \cdot \tau_2 - 1\right\} \cdot \tau_1 - 1 = \left\{\left[(2 \cdot 2 - 1) \cdot 2 - 1\right] \cdot 2 \cdot 2 - 1\right\} \cdot 2 - 1 = 37$. Т.ч., синтез суматора по модулю m_i =37 проведений вірно.

Приклад 3. Нехай $m_i = 53$. Розглянемо етапи синтезу суматора по модулю СЗК.

1. Відповідно до величини модуля $m_i = 53$, визначимо кількість n ДОС суматора по модулю $M = 2^n - 1$. Для модуля $m_i = 53$ маємо, що $n = [\log_2(m_i - 1)] + 1 = [\log_2(53 - 1)] + 1 = 6$. Структура суматора по модулю $M = 2^n - 1 = 63$ представлена на рис. 14. Вихідна структура суматора по модулю $m_i = 53$ без додаткових зв'язків $X_{\downarrow i \uparrow j}$ матиме той же вигляд.

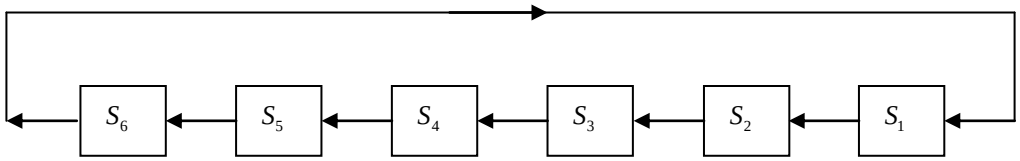


Рис. 14 – Вихідна структура суматора за модулем $M = 2^6 - 1$.

2. Модуль $m_i = 53$, в двійковому коді $S_6\, S_5\, S_4\, S_3\, S_2\, S_1$, представляється у вигляді 110101, тобто $S_6 = 1$, $S_5 = 1$, $S_4 = 0$, $S_3 = 1$, $S_2 = 0$ та $S_1 = 1$. З вигляду, який представлений в двійковому коді модулю $m_i = 53$ визначимо, що $S_2 = S_4 = 0$.

3. На основі отриманих результатів, структура суматора за модулем $m_i = 53$ має вигляд, який представлений нижче,е на рис. 15.

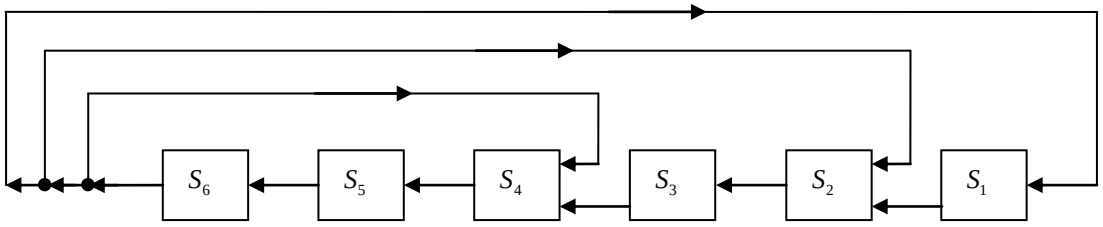


Рис. 15 – Структура суматора за модулем $m_i = 53$

Відповідно до методу синтезу, в суматор за модулем $M = 2^6 - 1$ введені два додаткові зв'язку $X_{\downarrow 4 \uparrow 6}$ та $X_{\downarrow 2 \uparrow 6}$. З метою перевірки правильності синтезу суматора по модулю $m_i = 53$, визначимо для даної структури суматора, значення модулю $M = m_i$ СЗК.

Враховуючи схему на рис. 15 складемо ряд окремих частин суматора по модулю $m_i = 53$.

Перша частина структури такого суматору, представлена на рис. 16.

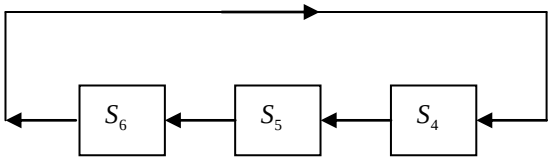


Рис. 16 – Перша частина структури суматора за модулем m_i

Для першої частини структури суматора, модуль M_1 визначиться наступним чином $M_1 = \tau_3 \cdot \tau_5 \cdot \tau_4 - 1$. Друга частина структури суматора представлена на рис. 17.

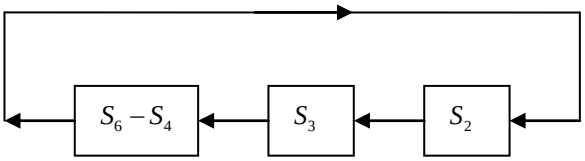


Рис. 17 – Друга частина структури суматора за модулем m_i

Для цієї частини структури, модуль M_2 визначиться виразом $M_2 = M_1 \cdot \tau_3 \cdot \tau_2 - 1 = (\tau_6 \cdot \tau_5 \cdot \tau_4 - 1) \cdot \tau_3 \cdot \tau_2 - 1$.

Для суматора по модулю, значення модуля $M = m_i$ СЗК визначається наступним чином (*див. рис. 15-17*) $m_i = M_2 \cdot \tau_1 - 1 = \left[(\tau_6 \cdot \tau_5 \cdot \tau_4 - 1) \cdot \tau_3 \cdot \tau_2 - 1\right] \cdot \tau_1 - 1 = \left[(2^3 - 1) \cdot 2^2 - 1\right] \cdot 2 - 1 = 53$. - Т.ч. на основі проведених розрахунків, можна зробити висновок, що синтез суматору за модулем $m_i = 53$ (рис. 15) проведений вірно!

Пример 4. Пусть $m_i = 97$. Этапы синтеза сумматора по модулю СОК следующие

1. У відповідності зі значенням модулю, $m_i = 97$ визначимо кількість *n* ДОС суматора за модулем $M = 2^n - 1$. Для модулю $m_i = 97$ маємо, що $n = \left\lceil \log_2 (m_i - 1) \right\rceil + 1 = \left\lceil \log_2 (97 - 1) \right\rceil + 1 = 7$. Структура суматору по модулю $M = 2^n - 1 = 2^7 - 1 = 127$ має вигляд, який представлений на рис. 18.

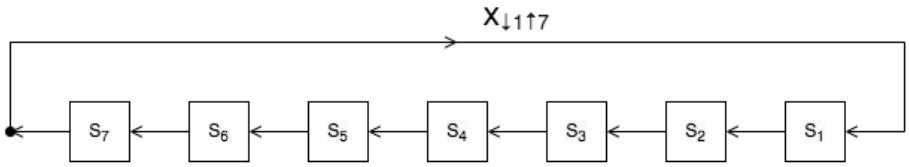


Рис. 18 – Вихідна структура суматора за модулем $M = 2^7 - 1$

2. Модуль $m_i = 97$ в двійковому коді $S_7 S_6 S_5 S_4 S_3 S_2 S_1$ представляється у вигляді 1100001, тобто $S_7 = 1, S_6 = 1, S_5 = S_4 = S_3 = S_2 = 0, S_1 = 1$.

3. На підставі отриманих результатів, структура суматора по модулю $m_i = 97$ представлена на рис. 19.

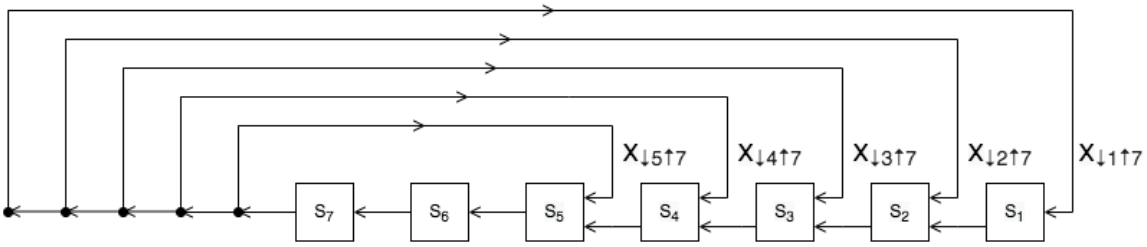


Рис. 19 – Структура суматора за модулем $m_i = 97$

Відповідно до отриманих результатів в суматор за модулем $M = 2^7 - 1$ введені 4 додаткові зв'язку: - $X_{\downarrow 5 \uparrow 7}, X_{\downarrow 4 \uparrow 7}, X_{\downarrow 3 \uparrow 7}, X_{\downarrow 2 \uparrow 7}$. Для перевірки правильності створеної структури суматору за модулем $m_i = 97$, визначимо значення модуля для створеної структури (*див. рис. 19*). Перша частина структури суматора (*рис. 19*) представлена на рис. 20.

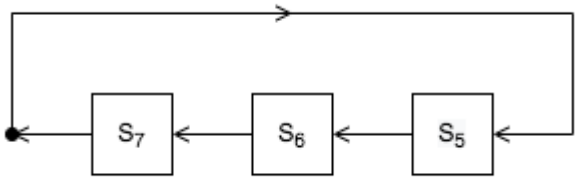


Рис. 20 – Перша частина структури суматора за модулем $m_i = 97$

Для першої частини структури суматора модуль M_1 визначається наступним чином $M_1 = \tau_7 \cdot \tau_6 \cdot \tau_5 - 1$.

Друга частина структури суматора представлена на рис. 21.

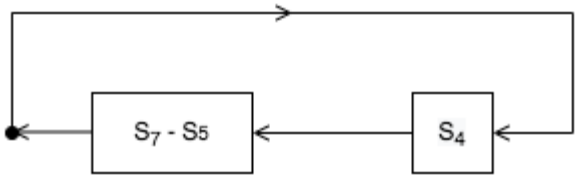


Рис. 21 – Друга частина структури суматора за модулем $m_i = 97$

Для 2-ої частини структури відповідного суматора модуль M_2 визначається, як $M_2 = M_1 \cdot \tau_4 - 1$.

Третя частина структури суматора представлена на рис. 22.

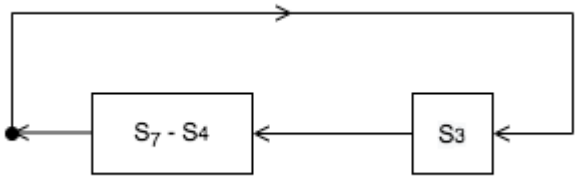


Рис. 22 – Третя частина структури суматора за модулем $m_i = 97$

Для третьої частини структури суматора, модуль M_3 визначається, як $M_3 = M_2 \cdot \tau_3 - 1$.

Четверта частина структури суматора представлена на рис. 23. Для неї модуль M_4 визначається, як $M_4 = M_3 \cdot \tau_2 - 1$.

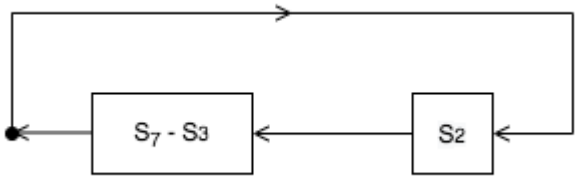


Рис. 23 – Четверта частина структури суматора за модулем $m_i = 97$

П'ята частина структури суматора представлена на рис. 24.

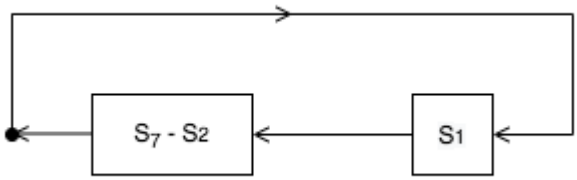


Рис. 24 – П'ята частина структури суматора за модулем $m_i = 97$

Для п'ятої частини структури суматора, модуль M_5 визначається, як $M_5 = M_4 \cdot \tau_1 - 1$.

Визначимо значення модуля m_i:

- $M_1 = \tau_7 \cdot \tau_6 \cdot \tau_5 - 1$ *(див. рис. 20)*;
- $M_2 = M_1 \cdot \tau_4 - 1 = (\tau_7 \cdot \tau_6 \cdot \tau_5 - 1) \cdot \tau_4 - 1$ *(див. рис. 21)*;
- $M_3 = M_2 \cdot \tau_3 - 1 = \left[(\tau_7 \cdot \tau_6 \cdot \tau_5 - 1)\tau_4 - 1\right]\tau_3 - 1$ *(див. рис. 22)*;
- $M_4 = M_3 \cdot \tau_2 - 1 = \left\{\left[(\tau_7 \cdot \tau_6 \cdot \tau_5 - 1) \cdot \tau_4 - 1\right] \cdot \tau_3 - 1\right\} \cdot \tau_2 - 1$ *(см. рис. 23)*;
- $M_5 = M_4 \cdot \tau_1 - 1 = \left(\left\{\left[(\tau_7 \cdot \tau_6 \cdot \tau_5 - 1) \cdot \tau_4 - 1\right] \cdot \tau_3 - 1\right\} \cdot \tau_2 - 1\right) \cdot \tau_1 - 1$ *(см. рис. 24)*.

В цьому випадку можна представити вираз для визначення значення модуля m_i у наступному вигляді: $m_i = m_5 = \left(\left\{\left[(2 \cdot 2 \cdot 2 - 1) \cdot 2 - 1\right] \cdot 2 - 1\right\} \cdot 2 - 1\right) \cdot 2 - 1 = 97$.

Таким чином, синтез суматора за модулем $m_i = 97$ (*рис. 19*) виконаний правильно.

7 Висновки

1. В роботі запропонований алгоритм побудови суматорів за модулем m_i СЗК.

2. Запропонований алгоритм заснований на використанні вже існуючих позиційних суматорів за модулем $M = 2^n - 1$ (що складаються із сукупності послідовно розташованих двійкових однорозрядних суматорів), за допомогою введення та подальшої реалізації, додаткових міжрозрядних зв'язків (*виду* $X_{\downarrow i \uparrow j}$).

3. Авторами роботи сформульовані правила введення додаткових міжрозрядних зв'язків виду $X_{\downarrow i \uparrow j}$. Підкреслено, що використання додаткових зв'язків *(на основі структури суматора за модулем* $M = 2^n - 1$*)*, дозволяє синтезувати суматор, який реалізує операцію додавання двох залишків a_i та b_i чисел.

4. Сукупність із k суматорів за модулем, представляє собою суматор двох чисел $A = (a_1 \| a_2 \| ... \| a_i \| ... \| a_k)$ та $B = (b_1 \| b_2 \| ... \| b_i \| ... \| b_k)$ в СЗК.

5. Розглянуті в межах цієї статті приклади побудови двійкових суматорів для різних значень модулів m_i СЗК, підтверджують можливість практичного використання запропонованого алгоритму.

Посилання

[1] V. A. Krasnobayev, A. A. Kuznetsov, S. A. Koshman, and K. O. Kuznetsova "A method for implementing the operation of modulo addition of the residues of two numbers in the residue number system", Cybernetics and Systems Analysis, Vol. 56, No. 6, November, 2020, 1029-1038. https://doi.org/10.1007/s10559-020-00323-9.

[2] Krasnobayev V. A., Yanko A. S., Koshman S. A. A Method for arithmetic comparison of data represented in a residue number of system // Cybernetics and Systems Analysis. – January 2016. – Vol. 52, Is. 1, pp. 145-150.

[3] Krasnobayev V. A. and Koshman S. A. Method for implementing the arithmetic operation of addition in residue number system based on the use of the principle of circular shift // Cybernetics and Systems Analysis. – July, 2019. – Vol. 55, Is. 4, pp. 692-698.

[4] Bayoumi M.A., Jullien G.A., Miller W.C. A VLSI Implementation of Residue. Adders IEEE Trans. on Circuits and Systems. 1987. Vol. 34, № 3. pp. 284-288.

[5] Azadeh Safari, James Nugent, Yinan Kong. Novel implementation of full adder based scaling in Residue Number Systems. IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS). 4-7 Aug. 2013. pp. 657–660.

[6] Shugang Wei. Fast signed-digit arithmetic circuits for residue number systems. IEEE International Conference on Electronics, Circuits, and Systems (ICECS). 6-9 Dec. 2015. pp. 344 – 347.
[7] P.V. Ananda Mohan. Residue Number Systems: Theory and Applications. Birkhäuser Basel: Springer International Publishing Switzerland, 2016. - 351 P.

Рецензент: Вячеслав Калашников, д.ф.-м.н., проф., Технологический университет Монтеррея, Монтеррей, Мексика.
E-mail: kalash@itesm.mx

Поступила: Февраль 2021.

Авторы:
Михаил Багмут, аспирант кафедры Безопасности информационных систем и технологий, Харьковский национальный университет имени В.Н. Каразина, Харьков, Украина.
E-mail: Mikhail56@ukr.net

Екатерина Кузнецова, студентка кафедры Безопасности информационных систем и технологий, Харьковский национальный университет имени В.Н. Каразина, Харьков, Украина.
E-mail: kate7smith12@gmail.com

Людмила Горбачова, студентка кафедры Безопасности информационных систем и технологий, Харьковский национальный университет имени В.Н. Каразина, Харьков, Украина.
E-mail: lusyag23@gmail.com

Алгоритм построения структуры сумматора двух остатков чисел по модулю.

Аннотация. Известно, что задача построения структуры сумматора, который работает по произвольному модулю m_i и выполнен на логических элементах с двумя устойчивыми состояниями, является актуальной научно-прикладной задачей. Данный тип сумматора используется, как в позиционной двоичной системе счисления *(ПСС),* так и в непозиционной системе счисления в остаточных классах *(СОК).* Если остатки a_i и b_i чисел $A = (a_1||a_2||...||a_i||...||a_k)$ и $B = (b_1||b_2||...||b_i||...||b_k)$, представленных в СОК, даны в двоичной ПСС, тогда сумматор двух остатков a_i и b_i по модулю m_i , представляет собой совокупность из $n = [log_2(m_i - 1)+1]$ двоичных одноразрядных сумматоров (*ДОС*). При этом все ДОС объединены между собой связями, подобно связям позиционных двоичных сумматоров. Целью статьи является разработка алгоритма построения структуры сумматора двух остатков a_i и b_i , чисел *A* и *B,* для произвольного значения m_i модуля СОК. Этот процесс реализован, путем организации новых межразрядных связей ДОС, с использованием позиционного сумматора по модулю $M = 2^n–1$. Отмечено, что существуют специальные наборы модулей, которые применяются при обработке данных в СОК. Так, при выполнении операции модульного сложения остатков чисел, может использоваться один из *3-х* взаимно попарно простых чисел *(вида* $M = 2^n - 1$, $M = 2^n$ *или* $M = 2^n + 1$*).* Показано, что для синтеза сумматора по модулю m_i СОК, в структуре сумматора по модулю *M,* необходимо соответствующим образом сформировать дополнительные связи.

Ключевые слова: непозиционная система счисления; однобитовый сумматор; система счисления остатков; сумматор остатков; целочисленные арифметические операции.

Reviewer: Vyacheslav Kalashnikov, Doctor of Sciences (Physics and Mathematics), Full Prof., Department of Systems and Industrial Engineering, Campus Monterrey, Monterrey, Mexico.
E-mail: kalash@itesm.mx

Received: February 2021.

Authors:
Mykhailo Bagmut, graduate student of the Department of Security of Information Systems and Technologies, Kharkiv National University named after V.N. Karazin, Kharkiv, Ukraine.
E-mail: Mikhail56@ukr.net

Katerina Kuznetsova, student of the Department of Security of Information Systems and Technologies, Kharkiv National University named after V.N. Karazin, Kharkiv, Ukraine.
E-mail: kate7smith12@gmail.com

Ludmila Gorbachova, student of the Department of Security of Information Systems and Technologies, Kharkiv National University named after V.N. Karazin, Kharkiv, Ukraine.
E-mail: lusyag23@gmail.com

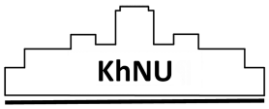
Algorithm for constructing the adder of residues of two numbers modulus.

Abstract. An urgent scientific and applied problem is the problem of constructing the adder structure, which is performed on logical elements with two stable states and operates according to an arbitrary modulo m_i. This type of adder is used both in the positional binary number system *(PNS)* and in the non-positional number system in residual classes *(RNS).* If the residuals a_i and b_i of numbers $A = (a_1||a_2||...||a_i||...||a_k)$ and $B = (b_1||b_2||...||b_i||...||b_k)$, represented in the RNS are given in a binary PNS, then the adder of two

residuals a_i and b_i modulo m_i is a set of $n = [log_2(m_i - 1)+1]$ binary one-bit adders (*BOBA*). Simultaneously, all BOBA are connected as positional binary adders. The purpose of the article is to develop an algorithm for constructing the adder structure of two residuals a_i and b_i of numbers *A* and *B* for an arbitrary modular value m_i of RNS. This process is realized by organizing new inter-bit connections of BOBA, using a positional adder modulo $M = 2^n - 1$. It is noted, that there are special sets of modules that are used when processing data in RNS. So, when performing the operation of modular addition of the remainders of numbers, one of three mutually pairwise primes (of the form $M = 2^n - 1$, $M = 2^n$ or $M = 2^n + 1$) can be used. It is shown that in order to synthesize an adder modulo m_i RNS, in the adder structure modulo *M*, it is necessary to appropriately form the additional connections.

Keywords: non-positional number system; single-bit adder; balance system; balance adder; integer arithmetic operations.

ISSN 2519-2310 DOI: 10.26565/2519-2310-2021-1



Наукове видання

КОМП'ЮТЕРНІ НАУКИ ТА КІБЕРБЕЗПЕКА

Випуск 1(19) 2021

Міжнародний електронний науково-теоретичний журнал

Англійською, українською, російською мовами

Комп'ютерне верстання – Федоренко В.В., Єсіна М.В.

61022, Харків, майдан Свободи, 6
Харківський національний університет імені В.Н. Каразіна

V. N. Karazin Kharkiv National University Publishing

CS&CS

2021