УДК 004.6

О.К. Тесленко, М.Ю. Бондарчук

РЕАЛІЗАЦІЯ ПІДСТАНОВОК ДОВІЛЬНОЇ РОЗРЯДНОСТІ НА БАЗІ КОМБІНОВАНИХ КАСКАДІВ КОНСТРУКТИВНИХ МОДУЛІВ

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ

Анотація. Швидкість перетворення і простота реалізації є одними з ключових факторів підстановок. У статті розглянуто реалізацію підстановки довільної розрядності в області комп’ютерної інженерії на одному із класів комбінаційних структур лінійної складності від кількості змінних – комбінованих каскадів конструктивних модулів. Використано той факт, що відображення, яке формує вказана лінійна структура, повністю збігається з відображенням відповідного скінченного автомата Мілі як прототипу конструктивного модуля каскаду. Це дозволило досліджувати властивості конструктивних модулів та каскаду в цілому у розрізі понять теорії цифрових автоматів. Реалізація підстановок довільної розрядності полягає у використанні зв’язаних пронумерованих однонаправлених автоматів для таблиці станів і використанні унікальних комбінацій без повторів для кожного рядку таблиці виходів. Метою реалізації даної підстановки є швидке перетворення даних великих об’ємів з можливістю застосування в кількох напрямках досліджень при простій реалізації на апаратному або програмному рівні. Виконано дослідження забезпечення бієктивності відображення та проведено аналіз еквівалентності відображень. Показано алгоритми формування автоматів для реалізації прямих та обернених підстановок, а також приклади формування таблиць переходів та виходів таких автоматів. Наведено приклади апаратної реалізації на програмованих логічних інтегральних схемах. Виконано оцінку об’єму таблиць переходів та виходів для апаратної та програмної реалізації. Виконано оцінку кількості унікальних бієктивних відображень. Проведено теоретичну оцінку швидкості бієктивних відображень при реалізації на програмованих логічних інтегральних схемах, а також при програмній реалізації згідно з сучасними показниками швидкості видів пам’яті обчислювальних пристроїв для кожного виду. Проведено порівняння швидкодії програмних реалізацій на базі комбінованого та одновимірного каскадів конструктивних модулів. Наведено експериментальну оцінку, а також проведено практичну перевірку швидкості перетворення за допомогою програмної реалізації. Запропоновано області застосування досліджених реалізацій підстановок довільної розрядності.

Ключові слова: функції підстановок; автомат Мілі; бієктивне відображення, програмовані логічні інтегральні схеми, каскади конструктивних модулів.

Abstract. The most crucial aspects of permutations are their speed and ease of implementation. This article examines the implementation of arbitrary bitness permutations in computer engineering using a particular class of combination structures with linear complexity, namely, combined cascades of structural units. The reflection formed by this linear structure is identical to that of the corresponding Mealy finite state machine, which allows for the exploration of the properties of structural units and the cascade in the context of the theory of digital automata. The purpose of this permutation is to convert large volumes of data using hardware or software quickly and simply that can be used in various research fields. The paper investigates the bijectivity and equivalence of the reflection and presents an algorithm for constructing finite-state machines for both direct and inverted permutations, along with examples of state and output table construction. The article also provides examples of hardware implementation using field-programmable gate arrays and estimates the size of state and output tables for software implementation. The theoretical speed of bijective reflection transformations is calculated for both field-programmable gate arrays and software implementation, and the paper compares the speed of software implementations based on combined and one-dimensional cascades of constructive units. The practical verification of processing speed is made with software implementation. Finally, the article proposes areas of application for this arbitrary bitness permutation.

Key words: permutation functions; Mealy machine; bijective reflection, field-programmable gate arrays, cascades of structural units.

DOI: https://doi.org/10.31649/1999-9941-2023-57-2-63-77.

Вступ

Підстановки (перестановки) розглядаються як функції однієї змінної, які забезпечують бієктивне відображення (взаємно однозначну відповідність) вхідних даних у вихідні. Підстановки застосовуються як при розгляді теоретичних питань в різних розділах математики (наприклад, теорія скінчених груп, скінчених полів, комбінаторика та ін.), так і в практичних розробках (наприклад в криптографічних перетвореннях). Не дивлячись на значні результати досліджень підстановок в математиці [[1]], в комп’ютерній інженерії реалізація підстановок досліджена в меншій мірі. В математиці прийнято характеризувати підстановки порядком – кількістю елементів вхідної (вихідної) множини. В комп’ютерній інженерії підстановки можна характеризувати розрядністю b вхідних бінарних даних. В такому випадку порядок підстановки дорівнює 2^b. Метою роботи є створення та дослідження програмних і частково апаратних засобів для реалізації підстановок довільної розрядності. Під довільною розрядністю мається на увазі можливість реалізації підстановки для будь-якого скінченного значення b. Проблема полягає в оптимізації по витратах і часу перетворення при забезпеченні залежності значень розрядів результату від всіх розрядів вхідних даних. Використання таких реалізацій підстановок дає перспективу забезпечення ефективності при створенні різноманітних перетворень інформації в тій чи іншій критеріальній сукупності.

Актуальність

Методи апаратної реалізації підстановок обмеженої розрядності досліджувались в ряді робіт [[2]][[3]][[4]], зокрема в [[2]] і [[4]] наведені результати реалізації на програмованих логічних інтегральних схемах (ПЛІС) як довільних 8-розрядних підстановок так і класу 8-розрядних підстановок зі спеціальними властивостями. Програмна реалізація довільних підстановок обмеженої розрядності проста та використовується у криптографічних перетвореннях [[5]][[6]]. В розглянутих роботах фактично досліджувався підхід «підстановки-реалізація», тобто формувався той чи інший клас підстановок, а потім визначалась реалізація підстановок цього класу. В [[7]] було запропоновано інший підхід до реалізації підстановок, який можна охарактеризувати як «реалізація-підстановки». Визначена комбінаційна структура лінійної складності від розрядності вхідних даних – одновимірні каскади конструктивних модулів (ОККМ). Наведена класифікація таких структур – визначені класи найпростіших, простих, складних, однонаправлених, двонаправлених та регулярних каскадів.

Очевидно, що на ОККМ теоретично можна реалізовувати підстановки довільної розрядності. При цьому виникають наступні задачі: які повинні бути конструктивні модулі (КМ) та скільки і які підстановки можна реалізувати на ОККМ відповідного класу. В загальному випадку ці задачі не вирішені.

В [[8]] показано, що на найпростіших однонаправлених регулярних ОККМ шляхом зміни КМ можна реалізувати 48 різних підстановок довільної розрядності (степені 2^b) ($b > 1$). В [[9]] показано, що на найпростіших двонаправлених регулярних ОККМ можна реалізувати 850 різних підстановок степені 2^b ($b > 1$). Значний приріст кількості різних підстановок при незначному ускладненні ОККМ дозволяє прогнозувати ефективність подальших ускладнень.

В [[10]] розглядається реалізація підстановок довільної розрядності в одному із класів лінійних структур з використанням одновимірних ОККМ, а в [[11]] досліджується використання таких підстановок для криптографічних перетворень.

Мета

Метою даної роботи є дослідження підстановок довільної розрядності на базі комбінованих каскадів конструктивних модулів.

Задачі

1. Визначення комбінаційних схем для формування сигналів на виходах КМ та реалізації на ОККМ підстановок довільної розрядності (бієктивних відображень).
2. Визначення умов, при яких різні ОККМ реалізують однакові підстановки.
3. Визначення кількості різних підстановок, які можуть бути реалізовані на ОККМ вибраного класу.
4. Визначення обернених КМ, які б забезпечували реалізацію обернених підстановок при заданих прямих КМ.
5. Усунення недоліку незалежності молодших розрядів підстановок, які реалізуються на однонаправленому ОККМ, від старших.
6. Оцінка об'єму таблиць переходів та виходів для апаратної та програмної реалізації.
7. Оцінка теоретичної і практичної швидкості підстановок.
8. Порівняння швидкодії програмних реалізацій на базі комбінованого та одновимірного каскадів конструктивних модулів.

Термінологія

Нехай маємо скінченний вхідний алфавіт X розмірністю n: $\{x_1, x_2, \dots, x_n\}$, скінченний проміжний алфавіт Y розмірністю n: $\{y_1, y_2, \dots, y_n\}$, скінченний вихідний алфавіт Z розмірністю n: $\{z_1, z_2, \dots, z_n\}$, скінчену множину станів S_1 розмірністю m_1, скінчену множину станів S_2 розмірністю m_2, функцію переходів першого автомата $f_s(x, s): S_1 \times X \to S_1$, функцію переходів другого автомата $g_s(y, s): S_2 \times Y \to S_2$, функцію виходів першого автомата $f_o(x, s): S_1 \times X \to Y$, функцію виходів другого автомата $g_o(y, s): S_2 \times Y \to Z$ і початкові стани $s_{0_1} \in S_1$ та $s_{0_2} \in S_2$.

У випадку, коли автомати розглядаються як прототипи КМ, вхідний та вихідний алфавіти збігаються, значення n, m_1 та m_2 є степенями двійки, а функції виходів і переходів реалізуються відповідними комбінаційними схемами. Будемо вважати, що функції виходів та переходів повністю визначені. Будемо також вважати, що автомати є приведеними, тобто вони не містять недосяжних станів і для кожного з автоматів жодні два його стани не еквівалентні один одному [[12]]. Крім того, на бокових входах перших конструктивних модулів ОККМ задаються початкові значення: для самого правого КМ першого автомата і для самого лівого КМ другого автомата (рис. 1). Ці значення відповідають початковим станам автоматів.

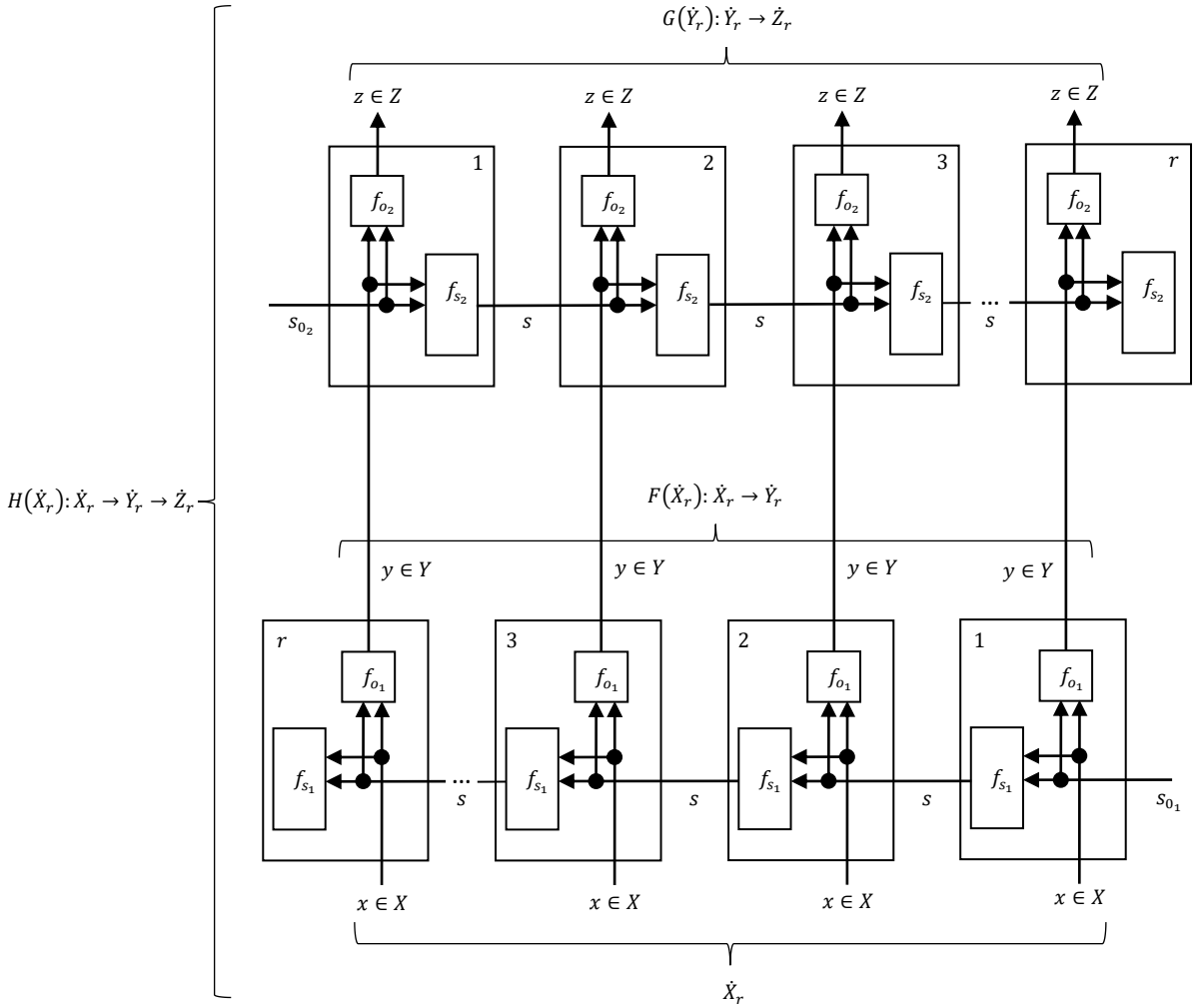

Рисунок 1 – Комбінований ОККМ для реалізації підстановки $H(\dot{X}_r)$

Позначимо $\dot{X}_r$ множину всіх послідовностей довжиною r із елементів вхідного алфавіту, $\dot{Y}_r$ – проміжного алфавіту, а $\dot{Z}_r$ – вихідного. Комбінований ОККМ виконує відображення множини $\dot{X}_r$ в множину $\dot{Z}_r$ ОККМ ($\dot{X}_r \to \dot{Y}_r \to \dot{Z}_r$). Тобто $F(\dot{X}_r)$ виконує відображення $\dot{X}_r \to \dot{Y}_r$, $G(\dot{Y}_r)$ виконує відображення $\dot{Y}_r \to \dot{Z}_r$, а $H(\dot{X}_r) = G\left(F(\dot{X}_r)\right)$, причому не варто плутати $H(\dot{X}_r)$ з $F(\dot{X}_r)$, оскільки $F(\dot{X}_r)$ позначає функцію відображення першого ОККМ, а $H(\dot{X}_r)$ – комбінованого.

Забезпечення бієктивності відображення

Реалізація комбінованим ОККМ підстановки означає реалізацію бієктивних відображень ($\dot{X}_r \to \dot{Y}_r \to \dot{Z}_r$) при будь-якому значенні r, тобто різним послідовностям $\dot{X}_r$ відповідають різні послідовності $\dot{Y}_r$, різним послідовностям $\dot{Y}_r$ відповідають різні послідовності $\dot{Z}_r$ і, таким чином, різним послідовностям $\dot{X}_r$ відповідають різні послідовності $\dot{Z}_r$.

Для усунення двозначностей, підстановки (бієктивні відображення) довільної розрядності, які реалізуються автоматами (або комбінованим ОККМ) в подальшому будемо позначати $H(\dot{X}_r)$.

Дві послідовності символів вважаються різними, якщо вони відрізняються принаймні одним символом. Якщо автоматне відображення не є бієктивним при деякому значенні r, то воно буде не бієктивним при будь-яких $r_1 > r$.

Станом з втратами називають стан s_l такий, для якого існують $x_1, x_2 \in X$, де $x_1 \neq x_2$ і $f_o(s_l, x_1) = f_o(s_l, x_2)$ або існують $y_1, y_2 \in Y$, де $y_1 \neq y_2$ і $g_o(s_l, y_1) = g_o(s_l, y_2)$. З урахуванням цього наводяться наступні теореми.

Теорема 1. Автоматне відображення бієктивне на множині всіх послідовностей вхідних символів довжиною r тоді і тільки тоді, коли автомат не містить станів з втратами, які досягаються з початкового стану за r кроків [[13]][[14]].

Теорема 2. Для того, щоб ОККМ реалізував підстановку $F(\dot{X}_r)$ необхідно і достатньо, щоб при будь-якому сигналі на бокових входах КМ, на первинних (не бокових) виходах комбінаційна схема реалізувала будь-яку підстановку значень сигналів на первинних входах.

Теорема 3. Для того, щоб комбінований ОККМ реалізував підстановку $H(\dot{X}_r)$ необхідно і достатньо, щоб при будь-яких сигналах на бокових входах КМ, на первинних (не бокових) виходах кожна комбінаційна схема реалізувала будь-яку підстановку значень сигналів на первинних входах.

Розрядність підстановок $F(\dot{X}_r)$, $G(\dot{Y}_r)$ і $H(\dot{X}_r)$ дорівнює $b = r\,log\,n$. Степінь підстановки при цьому дорівнює n^r.

Аналіз еквівалентності

В [[10]] розглядався аналіз еквівалентності автоматів, що використовуються і було визначено, що в класі одновимірних ОККМ, існує $m!$ (кількість можливих перестановок рядків таблиці станів) ОККМ, які реалізують одну і ту ж підстановку $F(\dot{X}_r)$. Тоді, аналогічно, для комбінованих ОККМ може існувати $m_1!\,m_2!$ еквівалентних ОККМ, які реалізують одну і ту ж підстановку $H(\dot{X}_r)$. Це важливо враховувати при оцінках кількості різних підстановок.

Алгоритм формування КМ

Реалізація функції підстановки потребує генерації чотирьох таблиць: таблиці переходів таблиці $S_1 \times X \to S_1$, таблиці переходів $S_2 \times Y \to S_2$, таблиці виходів $S_1 \times X \to Y$ і таблиці виходів $S_2 \times Y \to Z$. В [[10]] було запропоновано генерувати таблицю виходів за допомогою генератора псевдовипадкових чисел таким чином, щоб кожен рядок містив лише унікальні значення, а таблицю переходів просто генерувати довільним чином. В [[11]] було уточнено і значно покращено алгоритм генерування таблиці переходів як таблиці зв’язаного пронумерованого однонаправленого графа (граф, з будь-якої вершини якого можна потрапити до всіх його інших вершин).

Алгоритми генерування таблиць переходів і виходів прямого відображення мовою C# є занадто громіздкими порівняно з [[10]] і тому є доступними за посиланнями [[15]][[16]]. Варто зауважити, що даний алгоритм не є найоптимальнішим і використовується лише для генерації тестових даних для проведення досліджень.

Розглянемо приклад автомата для $m_1 = 12$, $m_2 = 10$ і $n = 8$.

В комірці кожної таблиці виходів (табл. 1 і табл. 2) знаходиться один із символів вихідного алгоритму, а в кожній комірці кожної таблиці кожної функції переходів (табл. 3 і табл. 4) знаходиться один зі станів.

Таблиця 1 – Таблиця виходів автомата для реалізації $F(\dot{X}_r)$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
s_1	y_2	y_4	y_6	y_8	y_5	y_1	y_7	y_3
s_2	y_2	y_4	y_8	y_3	y_5	y_6	y_7	y_1
s_3	y_5	y_8	y_6	y_3	y_7	y_4	y_2	y_1
s_4	y_1	y_8	y_6	y_2	y_7	y_4	y_3	y_5
s_5	y_1	y_6	y_3	y_7	y_5	y_2	y_4	y_8
s_6	y_8	y_4	y_5	y_7	y_6	y_2	y_3	y_1
s_7	y_4	y_7	y_6	y_1	y_3	y_8	y_2	y_5
s_8	y_4	y_8	y_5	y_2	y_3	y_7	y_1	y_6
s_9	y_6	y_1	y_3	y_2	y_8	y_5	y_7	y_4
s_{10}	y_6	y_5	y_8	y_4	y_3	y_2	y_1	y_7
s_{11}	y_1	y_7	y_5	y_6	y_3	y_2	y_4	y_8
s_{12}	y_1	y_2	y_7	y_5	y_4	y_3	y_8	y_6

Нумерація рядків таблиць переходів та виходів здійснюється зверху вниз починаючи з умовного позначення (номера) першого стану і закінчуючи останнім. Нумерація стовпчиків таблиць переходів та виходів здійснюється зліва направо починаючи з першого символу вхідного алфавіту і закінчуючи останнім.

Таблиця 2 – Таблиця виходів автомата для реалізації $G(\vec{Y}_r)$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
s_1	z_5	z_7	z_4	z_1	z_8	z_2	z_3	z_6
s_2	z_5	z_7	z_4	z_6	z_3	z_2	z_8	z_1
s_3	z_6	z_8	z_2	z_3	z_7	z_1	z_4	z_5
s_4	z_6	z_3	z_5	z_2	z_1	z_4	z_7	z_8
s_5	z_6	z_2	z_8	z_5	z_1	z_4	z_7	z_3
s_6	z_7	z_2	z_3	z_4	z_1	z_8	z_5	z_6
s_7	z_1	z_4	z_5	z_3	z_7	z_2	z_6	z_8
s_8	z_3	z_7	z_2	z_4	z_6	z_5	z_1	z_8
s_9	z_5	z_2	z_4	z_3	z_8	z_7	z_6	z_1
s_{10}	z_7	z_1	z_2	z_3	z_6	z_8	z_5	z_4

Таблиця 3 – Таблиця переходів автомата для реалізації $F(\vec{X}_r)$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
s_1	s_6	s_2	s_{11}	s_{12}	s_4	s_8	s_{10}	s_6
s_2	s_{10}	s_3	s_9	s_{11}	s_9	s_1	s_3	s_3
s_3	s_6	s_{11}	s_7	s_{11}	s_{12}	s_4	s_{11}	s_2
s_4	s_9	s_3	s_8	s_7	s_9	s_5	s_7	s_8
s_5	s_4	s_1	s_1	s_6	s_2	s_1	s_8	s_5
s_6	s_2	s_7	s_8	s_4	s_{12}	s_5	s_9	s_{11}
s_7	s_{11}	s_{10}	s_{12}	s_{11}	s_2	s_5	s_4	s_8
s_8	s_{12}	s_{12}	s_1	s_6	s_7	s_2	s_9	s_7
s_9	s_5	s_3	s_{11}	s_5	s_{10}	s_{12}	s_6	s_{10}
s_{10}	s_{10}	s_{12}	s_{12}	s_{10}	s_8	s_9	s_9	s_{11}
s_{11}	s_{11}	s_8	s_{12}	s_3	s_9	s_9	s_5	s_5
s_{12}	s_{10}	s_7	s_8	s_3	s_1	s_{11}	s_9	s_3

Таблиця 4 – Таблиця переходів автомата для реалізації $G(\vec{Y}_r)$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
s_1	u_1	u_6	u_4	u_6	u_5	u_6	u_3	u_2
s_2	u_8	u_7	u_6	u_3	u_8	u_8	u_2	u_2
s_3	u_3	u_4	u_9	u_4	u_6	u_{10}	u_9	u_7
s_4	u_5	u_9	u_1	u_5	u_5	u_5	u_4	u_9
s_5	u_2	u_9	u_3	u_9	u_{10}	u_6	u_8	u_6
s_6	u_1	u_3	u_7	u_{10}	u_8	u_4	u_3	u_4
s_7	u_4	u_5	u_8	u_8	u_9	u_{10}	u_2	u_8
s_8	u_4	u_4	u_7	u_8	u_5	u_1	u_7	u_9
s_9	u_7	u_3	u_9	u_6	u_3	u_9	u_{10}	u_3
s_{10}	u_5	u_{10}	u_6	u_2	u_8	u_1	u_9	u_4

Програмна реалізація відображень

Алгоритм реалізації відображень $\vec{X}_r \to \vec{Y}_r \to \vec{Z}_r$ мовою C#:

```
var outputs = new List<int>();
var firstLevelOutputs = new Stack<int>();
foreach (int input in inputs)
{
       firstLevelOutputs.Push(_machine1.Transform(input));
}
while (firstLevelOutputs.Any())
{
```

```
        int input = firstLevelOutputs.Pop();
        outputs.Add(_machine2.Transform(input));
}
return outputs;
```

Очевидна простота реалізації при використанні матриць перетворень, які можуть зберігатися як в оперативній пам'яті, так і в кеш-пам'яті процесора та високій швидкодії прямого та оберненого перетворень.

Варто зауважити, що в загальному випадку може знадобитись переформатування одержаного результату в $\vec{Z}_r$.

Обернені відображення

В комп'ютерній інженерії застосування прямих підстановок в багатьох випадках супроводжується використання обернених підстановок. Обернене бієктивне відображення кожного з автоматів визначається як $F^{-1}(F(\vec{X}_r)) = \vec{X}_r$ і $G^{-1}(G(\vec{Z}_r)) = \vec{Y}_r$ відповідно.

Підстановки є симетричною групою відносно операції суперпозиції.

Прийнято таку операцію вважати множенням. Симетрична група не є Абелевою [[17]], тобто $D_1 D_2 \neq D_2 D_1$.

Якщо є підстановка D_1 то оберненою до неї є підстановка $D_2 = {D_1}^{-1}$, де $D_1 D_2 = E$, а E – тотожня підстановка (одиниця групи).

Нехай D_1 – підстановка першого ОККМ досліджуваної структури, а D_2 – підстановка другого ОККМ досліджуваної структури. Створимо $D_3 = {D_1}^{-1}$ та $D_4 = {D_2}^{-1}$. Створення обернених підстановок на лінійній регулярній структурі показано в [[10]]. Створимо дворівневу структуру, де на першому рівні буде D_4, а на другому – D_3. Тоді підстановка $D_4 D_3$ буде оберненою до $D_1 D_2$. Дійсно, маємо $D_1 D_2 D_4 D_3 = D_1 (D_2 D_2^{-1}) D_1^{-1} = E$.

Алгоритм формування автоматів для реалізації обернених підстановок

Алгоритм генерування таблиць переходів і виходів кожного оберненого автомата залишається безмін порівняно з [[10]], особливість полягає лише в порядку генерування обернених автоматів. Перший рівень оберненого комбінованого каскаду має бути оберненим другим рівнем прямого, а другий рівень оберненого комбінованого каскаду має бути оберненим першим рівнем прямого.

Фрагмент коду генерування таблиць переходів і виходів одного оберненого автомата мовою C#:

```
InitialState = directMachine.InitialState;
for (int i = 0; i < M; i++)
{
    StateMatrix.Add(new List<int>());
    OutputMatrix.Add(new List<int>());

    for (int j = 0; j < N; j++)
    {
        for (int k = 0; k < N; k++)
        {
            if (directMachine.OutputMatrix[i][k] == j)
            {
                OutputMatrix[i].Add(k);
                StateMatrix[i].Add(directMachine.StateMatrix[i][k]);
                break;
            }
        }
    }
}
```

Тоді з використанням даного фрагменту обернений комбінований каскад мовою C# можна побудувати як

```
var machine1 = new ReversedMachine(directMachine2);
var machine2 = new ReversedMachine(directMachine1);
```

Повний алгоритм знаходиться за посиланням [[18]].

Алгоритм формування оберненого бієктивного відображення $\vec{Z}_r \to \vec{Y}_r \to \vec{X}_r$ повністю збігається з алгоритмом формування прямого, який наведено раніше. Різниця полягає у використанні відповідних таблиць.

Таблиця 5 – Таблиця виходів автомата для реалізації $G^{-1}(Z_r)$

	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
s_1	y_4	y_6	y_7	y_3	y_1	y_8	y_2	y_5
s_2	y_8	y_6	y_5	y_3	y_1	y_4	y_2	y_7
s_3	y_6	y_3	y_4	y_7	y_8	y_1	y_5	y_2
s_4	y_5	y_4	y_2	y_6	y_3	y_1	y_7	y_8
s_5	y_5	y_2	y_8	y_6	y_4	y_1	y_7	y_3
s_6	y_5	y_2	y_3	y_4	y_7	y_8	y_1	y_6
s_7	y_1	y_6	y_4	y_2	y_3	y_7	y_5	y_8
s_8	y_7	y_3	y_1	y_4	y_6	y_5	y_2	y_8
s_9	y_8	y_2	y_4	y_3	y_1	y_7	y_6	y_5
s_{10}	y_2	y_3	y_4	y_8	y_7	y_5	y_1	y_6

Таблиця 6 – Таблиця виходів автомата для реалізації $F^{-1}(Y_r)$

	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
s_1	x_6	x_1	x_8	x_2	x_5	x_3	x_7	x_4
s_2	x_8	x_1	x_4	x_2	x_5	x_6	x_7	x_3
s_3	x_8	x_7	x_4	x_6	x_1	x_3	x_5	x_2
s_4	x_1	x_4	x_7	x_6	x_8	x_3	x_5	x_2
s_5	x_1	x_6	x_3	x_7	x_5	x_2	x_4	x_8
s_6	x_8	x_6	x_7	x_2	x_3	x_5	x_4	1
s_7	x_4	x_7	x_5	x_1	x_8	x_3	x_2	x_6
s_8	x_7	x_4	x_5	x_1	x_3	x_8	x_6	x_2
s_9	x_2	x_4	x_3	x_8	x_6	x_1	x_7	x_5
s_{10}	x_7	x_6	x_5	x_4	x_2	x_1	x_8	x_3
s_{11}	x_1	x_6	x_5	x_7	x_3	x_4	x_2	x_8
s_{12}	x_1	x_2	x_6	x_5	x_4	x_8	x_3	x_7

Таблиця 7 – Таблиця переходів автомата для реалізації $G^{-1}(Z_r)$

	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
s_1	u_6	u_6	u_3	u_4	s_1	u_2	u_6	u_5
s_2	u_2	u_8	u_8	u_6	s_8	u_3	u_7	u_2
s_3	u_{10}	u_9	u_4	u_9	s_7	u_3	u_6	u_4
s_4	u_5	u_5	u_9	u_5	s_1	u_5	u_4	u_9
s_5	u_{10}	u_9	u_6	u_6	s_9	u_2	u_8	u_3
s_6	u_8	u_3	u_7	u_{10}	s_3	u_4	u_1	u_4
s_7	u_4	u_{10}	u_8	u_5	s_8	u_2	u_9	u_8
s_8	u_7	u_7	u_4	u_8	s_1	u_5	u_4	u_9
s_9	u_3	u_3	u_6	u_9	s_7	u_{10}	u_9	u_3
s_{10}	u_{10}	u_6	u_2	u_4	s_9	u_8	u_5	u_1

Апаратна реалізація на ПЛІС

Регулярний однонаправлений ОККМ є комбінаційним пристроєм, де кожний КМ реалізує $d + w_1$ та $d + w_2$ булевих функцій від $d + w_1$ та $d + w_2$ змінних, де $d = log\, n$, $w_1 = log\, m_1$ і $w_2 = log\, m_2$. В сучасних ПЛІС FPGA (англ. field-programmable gate array) [[19]] на одній таблиці пошуку (англ. lookup table або LUT) можна реалізувати будь-яку булеву функцію від 6 змінних. Якщо $d + w_1 < 7$ та $d + w_2 < 7$, то для реалізації одного КМ знадобиться $d + w_1$ та $d + w_2$ LUT, а для реалізації відображення ОККМ ($X_r \to Y_r \to Z_r$) потрібно $r(2d + w_1 + w_2)$ LUT. На одному LUT можуть бути реалізовані дві функції від 5 змінних. Тоді при $d + w_1 < 6$ та $d + w_2 < 6$ (наприклад $d = 2$, $w_1 = w_2 = 2$) КМ реалізується на чотирьох LUT, а байтові перетворення на 16 LUT. Схема на 8 входів реалізується за допомогою з’єднаних в

дві послідовності 8 КМ (по 4 КМ в кожній послідовності), кожний з яких має 2 входи та 2 виходи. Таким чином реалізується байтове відображення. Кількість різних байтових відображень дорівнює кількості різних КМ з врахуванням приведеності автоматів та їх еквівалентності. За оцінками згідно з (2) кількість складає понад 10^{22246} (табл. 10), але кількість різних 8-розрядних підстановок значно більша: $10^{78783} \approx 256!$ (цього значення немає в табл. 10, воно обчислене згідно формули програмним шляхом). Проте виникає задача належності підстановок, які реалізуються на восьми таких або інших КМ, заданим вимогам при тому чи іншому застосуванні, що є предметом подальших досліджень.

Таблиця 8 – Таблиця переходів автомата для реалізації $F^{-1}(\dot{Y}_r)$

	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
s_1	s_8	s_6	s_6	s_2	s_4	s_{11}	s_{10}	s_{12}
s_2	s_3	s_{10}	s_{11}	s_3	s_9	s_1	s_3	s_9
s_3	s_2	s_{11}	s_{11}	s_4	s_6	s_7	s_{12}	s_{11}
s_4	s_9	s_7	s_7	s_5	s_8	s_8	s_9	s_3
s_5	s_4	s_1	s_1	s_8	s_2	s_1	s_6	s_5
s_6	s_{11}	s_5	s_9	s_7	s_8	s_{12}	s_4	s_2
s_7	s_{11}	s_4	s_2	s_{11}	s_8	s_{12}	s_{10}	s_5
s_8	s_9	s_6	s_7	s_{12}	s_1	s_7	s_2	s_{12}
s_9	s_3	s_5	s_{11}	s_{10}	s_{12}	s_5	s_6	s_{10}
s_{10}	s_9	s_9	s_8	s_{10}	s_{12}	s_{10}	s_{11}	s_{12}
s_{11}	s_{11}	s_9	s_9	s_5	s_{12}	s_3	s_8	s_5
s_{12}	s_{10}	s_7	s_{11}	s_1	s_3	s_3	s_8	s_9

Позначимо t_{LUT} затримку сигналу на одному LUT. Тоді швидкість обчислення дорівнює $2rt_{LUT}$, де використано коефіцієнт 2, оскільки перетворення вхідної послідовності відбувається через дві комбінаційні схеми.

Приклад комбінованого ОККМ наведено на рис. 1.

Оцінки об'єму таблиць переходів та виходів КМ

Для алфавіту довжиною n необхідно $d = log\, n$ біт для кожного символу цього алфавіту. Розмірності таблиць переходів і виходів першого і другого автоматів складають $n \times m_1$ та $n \times m_2$ відповідно, а для збереження поточних станів необхідно $w_1 = log\, m_1$ і $w_2 = log\, m_2$ біт. Таким чином загальний об'єм пам'яті для досліджуваної функції складає

$$Q = n(m_1 + m_2)\log n + (nm_1 + 1)\log m_1 + (nm_2 + 1)\log m_2.$$

Об'єми пам'яті для зберігання таблиць одного із ОККМ надані в [[10]]. Бачимо, що для великих значень n, m_1 та m_2 необхідні терабайти інформації, що ускладнює апаратну реалізацію, але не є проблемою для сучасних програмних реалізацій.

Оцінка кількості бієктивних відображень

Очевидно, що кількість відображень будь-якої довжини визначається кількістю різних КМ і залежить від початкових станів, кількості різних функцій переходів та функцій виходів обох ОККМ. Таким чином кількість можливих підстановок $H(\dot{X}_r, \dot{Y}_r)$ складає

$$O(n, m_1, m_2) = L_{f_s} L_{f_o} m_1 L_{g_s} L_{g_o} m_2,$$

де L_{f_s} – кількість функцій переходів першого ОККМ, L_{f_o} – кількість функцій виходів першого ОККМ, m_1 – кількість початкових станів першого ОККМ, L_{g_s} – кількість функцій переходів другого ОККМ, L_{g_o} – кількість функцій виходів другого ОККМ, m_2 – кількість початкових станів другого ОККМ.

В [[10]] кількість функцій переходів представляла собою діапазон між кількістю можливих зв'язних графів з кількістю вершин m послідовності A001349 [[20]] та будь-якою комбінацією таблиці переходів без жодних обмежень. Кількість же функцій виходів L_{f_o} обчислювалась як розміщення рядків таблиці виходів, кожен з яких має $n!$ варіантів згідно з вимогами бієктивності на m можливих місць [[21]].

В [[11]] було уточнено обидві формули. Таким чином кількість різних функцій переходів звелась до кількості зв’язних орієнтованих (або ж пронумерованих) графів і обчислювалась за допомогою послідовності цілих чисел A001187 [[22]]. Також [[11]] уточнює кількість різних функцій виходів одного ОККМ як кількість розміщень із $n!$ по m.

Згідно [[22]], кількість функцій переходів одного ОККМ обчислюється за рекурентною формулою

$$C_m = \sum_{k=1}^{m-1} \binom{m-2}{k-1} (2^k - 1) C_k C_{m-k} \ \ (C_0 = 1; C_1 = 1) \tag{1}$$

Тоді кількість бієктивних відображень комбінованого ОККМ обчислюється як

$$O(n, m_1, m_2) = m_1 C_{m_1} \frac{(n!)!}{(n!-m_1)!} m_2 C_{m_2} \frac{(n!)!}{(n!-m_2)!} \tag{2}$$

Зауважимо, що кількість різних підстановок $H(X_r^*)$ може бути меншою через наявність еквівалентних автоматів згідно з теоремою 2.

Наближені межі кількості бієктивних відображень наведені в табл. 9.

Оскільки формула (2) вимагає довгої арифметики для обчислень значень (через астрономічні значення чисел), то наведені значення дещо завищені (мінімальна кількість біт, щоб описати отримане десяткове число довгої арифметики). Десяткові значення $O(n, m_1, m_2)$ наведені як порядки, а значення $O(n, m_1, m_2)$, *біт* – як мінімальна кількість біт, щоб вмістити отримане десяткове число довгої арифметики.

Таблиця 9 – Наближені межі кількості бієктивних відображень для деяких значень параметрів n, m_1 і m_2

n	m_1	m_2	$O(n, m_1, m_2)$	$O(n, m_1, m_2)$, біт
4	4	4	$O(10^{2246})$	676
4	4	8	$O(10^{15816})$	4761
4	4	16	$O(10^{113693})$	34225
16	32	8	$O(10^{1214760})$	365679
16	32	16	$O(10^{5307425})$	1597694
32	16	8	$O(10^{1214760})$	365679
32	16	16	$O(10^{5307425})$	1597694
32	64	64	$O(10^{694594961})$	209093918
32	64	128	$O(10^{3723094374})$	1120763083
64	32	16	$O(10^{30933430})$	9311890
64	32	32	$O(10^{135410596})$	40762651
64	32	64	$O(10^{694594961})$	209093918
64	32	128	$O(10^{3723094374})$	1120763083
128	64	32	$O(10^{736110658})$	221591388
128	64	64	$O(10^{3205388938})$	964918218
128	128	64	$O(10^{7036230806})$	2118116529
128	128	128	$O(10^{33089188983})$	9960838416
256	128	64	$O(10^{16576260841})$	4989951729
256	128	128	$O(10^{71608368118})$	21556266744
256	256	32	$O(10^{13088638863})$	3940072900
256	256	64	$O(10^{39051081615})$	11755546929
256	256	128	$O(10^{154967786826})$	46649952196
256	256	256	$O(10^{714433513610})$	215065917504

Табл. 9 показує, що вже для значень n, m_1 та m_2 більше 8 кількість бієктивних відображень набуває астрономічного значення, в першу чергу через астрономічне зростання значень послідовності A001187 [[22]].

Оцінки швидкості програмних реалізацій

Для перетворення одного символу вхідної послідовності необхідно виконати читання з області пам’яті по індексу в обох таблицях переходів, виконати читання з області пам’яті по індексу в обох таблицях виходів та змінити стани. Індекси визначаються значеннями поточних станів і символами вхідних алфавітів кожного автомату згідно визначень. Зміна кожного стану є записом в області пам’яті. Оскільки

не використовується жодних обчислень, то час прямого і оберненого перетворень прямо пропорційний швидкості пам'яті, яка використовується (рівні кеша процесора L1, L2, L3, L4, ОЗП – оперативний запам'ятовуючий пристрій, ПЗП – постійний запам'ятовуючий пристрій на базі SSD – твердотільного накопичувача або ж HDD – жорсткого магнітного диску, зовнішній пристрій тощо).

Позначимо швидкість виконання однієї операції з пам'яттю v (МБ/с). Іншими словами, v – це і є швидкість пам'яті.

Тоді час T перетворення вхідного повідомлення довжиною r символів вхідного алфавіту обчислюється як

$$T = \frac{r(4\log_2 n + 6\log_2 m_1 + 6\log_2 m_2)}{v},$$

де коефіцієнт 4 при n складається з двох операцій читання з таблиць переходів обох автоматів (звернення по індексу) та двох операцій читання з таблиць виходів обох автоматів (звернення по індексу); коефіцієнти 6 при m_1 та m_2 складаються з тих же операцій, що і коефіцієнт при n, а також з двох операцій читання поточних станів та двох операцій запису нових станів обох автоматів, m_1 та m_2 відповідно.

В табл. 10 наведено показники об'ємів пам'яті та максимальна швидкість видів пам'яті згідно з останніми дослідженнями та вимірами [[23]][[24]][[25]][[26]][[27]]. Зазвичай важко отримати одну єдину цифру для таких даних, оскільки апаратне забезпечення постійно покращується, а дані необхідно агрегувати з офіційних даних від виробників (таких як Intel), а також з реальних даних користувачів, оцінкою яких займаються електронні ресурси. Об'єм виду пам'яті необхідно враховувати при оцінці використання підстановки відповідно з табл. 9. Швидкості наведені для найбільш відомих і широко використовуваних видів пам'яті, хоча існує набагато більше проміжних і передових видів. Час читання і час запису можуть бути дещо різними для кожного виду пам'яті, але для узагальнення вважатимемо, що даний час є однаковим для обох операцій. Важливо зауважити, що швидкість пам'яті може змінюватися залежно від низки факторів, включаючи конкретний продукт, конфігурацію системи та тип робочого навантаження, яке використовується. Також варто зауважити, що не всі процесори мають L4 кеш.

Таблиця 10 – Швидкість та об'єм видів пам'яті

Вид пам'яті	Об'єм Q	Швидкість v
L1 кеш	128 КБ	700 ГБ/с
L2 кеш	1 МБ	200 ГБ/с
L3 кеш	6 МБ	100 ГБ/с
L4 кеш	128 МБ	40 ГБ/с
ОЗП	Гігабайти	10 ГБ/с
ПЗП SSD	Терабайти	500 МБ/с
ПЗП HDD	Терабайти	150 МБ/с

В табл. 11 наведено час перетворення за допомогою функції підстановки залежно від виду пам'яті та розрядності вхідного алфавіту і множини станів. Для випадків, коли максимальний об'єм пам'яті для реалізації підстановки на час написання статті перевищує вимоги згідно табл. 10, час позначено «*».

Через l позначено теоретичний сумарний об'єм пам'яті, до якого мають відбутися звернення при перетворенні повідомлення довжиною r.

Зазвичай сучасні алгоритми абстраговані від використання певних видів пам'яті і делегують визначення типу пам'яті для певної операції у певний момент часу операційній систем, або ж апаратному забезпеченню (особливо стосовно використання кешу процесора). Для видів пам'яті, на які алгоритми мають вплив (починаючи з оперативної пам'яті) зазвичай використовується комбінація кількох видів пам'яті, при чому чим меншим є пакет даних для обробки тим більш ймовірно, що буде використано пам'ять вищого рівня.

Застосування

Запропоновані реалізації підстановок можуть застосовуватися, аналогічно до схеми з використанням однорівневого ОККМ в [[10]], в ущільненні даних [[28]], оптимізації комбінаційних схем [[8]][[29]], неалгоритмічній реалізації кодерів та декодерів завадостійкого кодування [[30]], а також для підвищення ефективності програмної реалізації алгоритмів [[31]]. Дослідження має широкий спектр застосування, та теоретично може використовуватись у будь-яких пристроях, які потребують, наприклад, високої швидкості шифрування та розшифрування, наприклад, в реальному часі.

Таблиця 11 – Теоретичний час перетворення за допомогою підстановки

r	n	m_1	m_2	l	T L1	T L2	T L3	T L4	T ОЗП	T SSD	T HDD
10^7	4	4	4	38 МБ	54 мкс	187 мкс	373 мкс	932 мкс	4 мс	75 мс	249 мс
10^{13}	4	4	4	36 ТБ	54 с	4 хв	7 хв	16 хв	2 год	21 год	3 днів
10^7	8	8	8	57 МБ	80 мкс	280 мкс	559 мкс	2 мс	6 мс	112 мс	373 мс
10^7	16	16	16	76 МБ	107 мкс	373 мкс	746 мкс	2 мс	8 мс	150 мс	497 мс
10^7	16	32	32	90 МБ	127 мкс	443 мкс	885 мкс	3 мс	9 мс	177 мс	590 мс
10^7	32	16	16	81 МБ	114 мкс	396 мкс	792 мкс	2 мс	8 мс	159 мс	528 мс
10^7	32	32	32	95 МБ	134 мкс	466 мкс	932 мкс	3 мс	10 мс	187 мс	621 мс
10^{13}	32	32	32	90 ТБ	3 хв	8 хв	16 хв	39 хв	3 год	52 год	8 днів
10^7	32	32	64	102 МБ	144 мкс	501 мкс	2 мс	3 мс	11 мс	201 мс	668 мс
10^7	32	64	64	109 МБ	154 мкс	536 мкс	2 мс	3 мс	11 мс	215 мс	715 мс
10^7	64	32	32	100 МБ	140 мкс	489 мкс	978 мкс	3 мс	10 мс	196 мс	652 мс
10^7	64	64	64	114 МБ	160 мкс	559 мкс	2 мс	3 мс	12 мс	224 мс	746 мс
10^{13}	64	64	64	109 ТБ	3 хв	10 хв	19 хв	47 хв	4 год	3 днів	9 днів
10^7	128	128	128	133 МБ	187 мкс	652 мкс	2 мс	4 мс	14 мс	261 мс	870 мс
10^7	128	256	256	147 МБ	207 мкс*	722 мкс	2 мс	4 мс	15 мс	289 мс	963 мс
10^7	256	256	256	152 МБ	213 мкс*	746 мкс	2 мс	4 мс	15 мс	299 мс	994 мс
10^8	256	256	256	1 ГБ	3 мс*	8 мс	15 мс	38 мс	150 мс	3 с	10 с
10^9	256	256	256	14 ГБ	22 мс*	75 мс	150 мс	373 мс	2 с	30 с	2 хв
10^{10}	256	256	256	149 ГБ	213 мс*	746 мс	2 с	4 с	15 с	5 хв	17 хв
10^{11}	256	256	256	1 ТБ	3 с*	8 с	15 с	38 с	3 хв	50 хв	3 год
10^{12}	256	256	256	14 ТБ	22 с*	2 хв	3 хв	7 хв	25 хв	9 год	28 год
10^{13}	256	256	256	145 ТБ	4 хв*	13 хв	25 хв	2 год	5 год	4 днів	12 днів

Порівняння результатів з однорівневим каскадом та існуючими аналогами

Ефективність запропонованих рішень випливає з наступних порівнянь з відомими результатами.

При апаратних реалізаціях, наприклад, байтових підстановок в нашому випадку достатньо $C = 8(d + w)/d$ LUT, в той же час, кращі реалізації байтових підстановок, запропоновані в [[4]] потребують 19 LUT. Якщо взяти $d = 4, w_1 = 2, w_2 = 2$ ($n = 16, m_1 = 4, m_2 = 4$), то маємо $C = 24$ LUT. Якщо ж використовувати можливість реалізації на LUT двох булевих функцій від 5 змінних, то при $d = 2$, $w_1 = 2, w_2 = 2$ ($n = 4, m_1 = 4, m_2 = 4$), маємо $C = 16$ LUT.

Щодо часових характеристик програмних реалізацій. В [[32]] розглядається реалізація підстановок великої розрядності з використанням простих перетворень в полях Галуа. Теоретичні розрахунки швидкості суттєво поступаються наведеним в табл. 7, оскільки в нашому випадку досягаються швидкості до мільйона Мбіт/с. Експериментальні результати в [[32]] відсутні.

В табл. 12 наведені результати проведених експериментів для деяких значень табл. 11 по формуванню бієктивних відображень для випадку $d = w_1 = w_2 = 8$ ($n = m_1 = m_2 = 256$). Позначимо обсяг даних, що перетворюються як q. Тоді значення q фактично відповідає кількості байт файлу (оскільки $d = 8$). Розмір вхідної послідовності позначено як l_q (значення наведено для читабельності). Тобто для експерименту перетворювалося q/r повідомлень довжиною r. T_o і ϑ_o – це час і швидкість перетворення вхідної послідовності однорівневим каскадом (згідно алгоритму [[10]]), а T_o і ϑ_o – це час і швидкість перетворення вхідної послідовності комбінованим каскадом. Експерименти проводились з використанням ОЗП DDR5 4800 МГц та процесора Intel Core i9-12900H. Варто зауважити, що виміри проводились алгоритмом мовою C# зі збереженням всіх даних у пам’яті. В такому випадку зазвичай використовується ОЗП, проте кінцевий вибір делегується операційній системі та апаратному забезпеченню, тому при виконанні програми додатково могла задіюватись кеш-пам’ять процесора, а, для великих об’ємів даних, – ПЗП SSD (хоча і малоймовірно оскільки об’єми досліджуваних даних не перевищували об’єм ОЗП пристрою, на якому виконувались тести).

Таблиця 12 – Час та швидкість перетворення за допомогою програмної реалізації підстановки

q	l_q	r	T_o	ϑ_o, Мбіт/с	$T_к$	$\vartheta_к$, Мбіт/с
10^7	9 МБ	1000	171 мс	55	581 мс	16
10^8	95 МБ	1000	2 с	68	7 с	15
10^9	953 МБ	1000	16 с	60	2 хв	15
10^{10}	9 ГБ	1000	3 хв	58	11 хв	15
10^{11}	93 ГБ	1000	24 хв	66	2 год	17

Бачимо, що реальний час перетворення є на порядок нижчим за теоретичний. По-перше, це можна пояснити тим, що дані табл. 11 є теоретичними і ідеалізованими для виконання на простих мікросхемах, метою яких є виключно робота згідно алгоритму, а реальний експеримент проводився в програмній емуляції на персональному комп'ютері. По-друге, в реальних умовах багатопотокових операційних систем при обробці файлів великої довжини складно отримати монопольний доступ до ресурсів системи, особливо до кеш-пам'яті.

Також варто зауважити, що результати є гіршими ніж для аналогічного експерименту в [[10]], оскільки використовувались значно більші об'єми даних, що відповідає більш реалістичному сценарію.

В [[5]] наведені експериментальні дані часу криптографічних перетворень за стандартом «Калина». Найкращі результати мають швидкість близько 2500 Мбіт/с. Дані по швидкості, які наведені в табл. 12, є гіршими за результати «Калини». Проте необхідно відзначити, що умови проведення експерименту в [[5]] фактично відповідали теоретичним розрахункам. Про це свідчить використання кеш-пам'яті, де знаходився лише один блок (від 16 до 64 байт) початкового повідомлення та шифрування лише цього блоку (фактично режим електронної кодової книги). В нашому випадку теоретично досягається швидкість більше мільйона Мбіт/с.

Необхідно зауважити, що в розглянутому способі реалізації підстановок забезпечується залежність всіх елементів вихідних значень від всіх елементів вхідних значень, а це не забезпечується в загальноприйнятих режимах криптографічних перетворень.

Окрім того, комбінований ОККМ дає значний приріст кількості можливих простих реалізацій (лінійної складності) підстановок довільної розрядності порівняно з однорівневим ОККМ.

Висновки

1. Отримані результати забезпечують бієктивне перетворення файлів будь-якої скінченої довжини. Кількість різних перетворень зростає швидше ніж експонента від параметрів n, m_1 та m_2 КМ і не залежить від розміру файлів. Наприклад, уже при $n = m_1 = m_2 = 8$ (розрядність даних КМ лише 3), кількість різних підстановок $H(X_r)$ оцінюється величиною, не меншою за 10^{50}.
2. Досліджена реалізація підстановок дозволяє забезпечити гарний показник швидкодії хоча й вимагає місця в пам'яті для таблиці переходів і виходів. Отримані результати показують, що у сенсі необхідного об'єму даних, таблиці виходів і переходів можуть бути вбудовані навіть у кеш-пам'ять процесора та забезпечувати перетворення з надвисокою швидкістю.

Список літератури

[1] M. Hazewinkel, *Encyclopedia of Mathematics*. Springer Science+Business Media B. V. Kluwer Academic Publishers, 2000, doi: 10.1007/978-94-015-1279-4.

[2] L. Peng, "The Generation of (n,n(n-1),n-1) Permutation Group Codes for Communication Systems", *IEEE Transactions on Communications*, vol. 67, no. 7, pp. 4535-4549, 2019, doi: 10.1109/TCOMM.2019.2902149.

[3] E. Boss, V. Grosso, T. Güneysu, G. Leander, A. Moradi and T. Schneider, "Strong 8-bit sboxes with ecient masking in hardware", *Journal of Cryptographic Engineering*. no. 7, pp. 149-165, doi: 10.1007/s13389-017-0156-7.

[4] Д. Фомин и Д. Трифонов, «Об аппаратной реализации одного класса байтовых подстановок», *Прикладная дискретная математика. Приложение*, № 12, с. 134–137, 2019, doi: 10.17223/2226308X/12/39.

[5] Р. Олійников, І. Горбенко, О. Казимиров, В. Руженцев та Ю. Горбенко, «Принципи побудови і основні властивості нового національного стандарту блокового шифрування України», *Захист інформації*, т. 17, № 2, с. 142-157, 2015, doi: 10.18372/2410-7840.17.8789.

[6] National Institute of Standards and Technology. (2001, Nov. 26). *Advanced Encryption Standard (AES)*. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.197.

[7] В. Тарасенко, О. Тесленко та О. Яновська, «Проблеми апаратної реалізації підстановок», *Наукові записки УНДІЗ*, № 2, с 52-58, 2007.

[8] В. Тарасенко, О. Тесленко та О. Яновська, «Властивості повних підстановок, які реалізуються найпростішим однонаправленим регулярним одновимірним каскадом конструктивних модулів», *Радіоелектронні і комп'ютерні системи*, № 6, с.123-128, 2010.

[9] В. Тарасенко, О. Тесленко та О. Яновська, «Можливості найпростіших двонаправлених регулярних одновимірних каскадів конструктивних модулів щодо реалізації різних повних підстановок», *Радіоелектронні і комп'ютерні системи*, № 7, с. 147-153, 2012.

[10] O. Teslenko and M. Bondarchuk, "Implementation of arbitrary bitness permutations in one of the classes of linear structures", *Herald of Advanced Information Technology*, vol. 3, no. 1, pp. 406-417, 2020, doi: 10.15276/hait01.2020.7.

[11] О. Тесленко та М. Бондарчук, «Використання реалізацій підстановок довільної розрядності для криптографічних перетворень», *Наукові вісті КПІ*, № 1-2, 2022, doi: 10.20535/kpisn.2022.1-2.263225.
[12] В. Глушков, *Синтез цифровых автоматов*. Київ, 1967.
[13] М. Карандашов, «Исследование биективных автоматных отображений на кольце вычетов по модулю 2^k», *Компьютерные науки и информационные технологии*, с. 148-152, 2014.
[14] A. Gill, *Introduction to the theory of finite-state machines*. McGraw-Hill electronic sciences series, 1962.
[15] М. Бондарчук «Алгоритм генерування таблиці переходів як таблиці зв'язаного пронумерованого однонаправленого графа». [Електронний ресурс]. Доступно: https://github.com/MaksymBondarchuk/Article-3-materials/blob/master/ConnectedGraphStateAlgorithm.cs. Дата звернення: 10.04.2023.
[16] М. Бондарчук «Алгоритм генерування таблиці виходів з унікальними значеннями в кожному рядку». [Електронний ресурс]. Доступно: https://github.com/MaksymBondarchuk/Article-3-materials/blob/master/RandomNonRepeatingRowsOutputAlgorithm.cs. Дата звернення: 10.04.2023.
[17] G. Frobenius and L. Stickelberger, "*Ueber Gruppen von vertauschbaren Elementen*", Journal für die reine und angewandte Mathematik, 1879, doi: 10.1515/crll.1879.86.217.
[18] М. Бондарчук «Алгоритм генерування обернених автоматів». [Електронний ресурс]. Доступно: https://github.com/MaksymBondarchuk/Article-3-materials/blob/master/ReversedMachine.cs. Дата звернення: 10.04.2023.
[19] Summary of Virtex-6 FPGA Features. Virtex-6 Family Overview. XILINX DS150 (v2.5). [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds150.pdf. Accessed on: 13.04.2023.
[20] Sequence A001349 – Number of connected graphs with n nodes. The On-Line Encyclopedia of Integer Sequences® (OEIS®). [Online]. Available: http://oeis.org/A001349. Accessed on: 13.04.2023.
[21] J. van Lint and R. Wilson, *A Course in Combinatorics*. Cambridge University Press, 1992.
[22] Sequence A001187 – Number of connected labeled graphs with n nodes. The On-Line Encyclopedia of Integer Sequences® (OEIS®). [Online]. Available: http://oeis.org/A001187. Accessed on: 13.04.2023.
[23] SiSoftware Official Live Ranker "SiSoftware Zone". [Online]. Available: https://ranker.sisoftware.co.uk/top_device_all.php?q=d6ebdb. Accessed on: 03.01.2023.
[24] Memory Performance in a Nutshell. [Online]. Available: https://www.intel.com/content/www/us/en/developer/articles/technical/memory-performance-in-a-nutshell.html. Accessed on: 10.04.2023.
[25] RAM UserBenchmarks – 115 Memory Kits Compared. [Online]. Available: https://ram.userbenchmark.com. Accessed on: 10.04.2023.
[26] SSD UserBenchmarks – 1071 Solid State Drives Compared. [Online]. Available: https://ssd.userbenchmark.com. Accessed on: 10.04.2023.
[27] HDD UserBenchmarks – 1015 Hard Drives Compared. [Online]. Available: https://hdd.userbenchmark.com. Accessed on: 10.04.2023.
[28] О. Яновська, В. Тарасенко та О. Тесленко, «Використання прямих та обернених підстановок довільної розрядності для підвищення ефективності існуючих засобів ущільнення даних», Тези доповідей Четвертої Міжнародної науково-практичної конференції «Методи та засоби кодування, захисту й ущільнення інформації, с. 223-226, Вінниця, 2013.
[29] В. Тарасенко, О. Тесленко та О. Яновська, «Аналіз впливу підстановок на декомпозицію булевих функцій», Вісник університету «Україна», № 8, с. 40-47, 2010.
[30] Я. Клятченко, В. Тарасенко, О. Тесленко та О. Яновська, «Використання підстановок для неалгоритмічної реалізації кодерів та декодерів завадостійкого кодування», *Сучасні комп'ютерні системи та мережі: розробка та використання (ACSN'2011)*, с.191-193, Львів, 2011.
[31] О. Мельникова та А. Маслєннікова, «Підстановки для підвищення ефективності програмної реалізації алгоритмів, які використовують знаково-цифрові представлення», *Математичне та комп'ютерне моделювання*, № 15, с. 126-132, Харків, 2017.
[32] А. Аборнев, «Нелинейные подстановки на векторном пространстве, рекурсивно-порождённые над кольцом Галуа характеристики 4», *Прикладная дискретная математика. Приложение*, № 7, с. 40–41, 2014.

Стаття надійшла: 29.04.2023.

References

[1] M. Hazewinkel, *Encyclopedia of Mathematics*. Springer Science+Business Media B. V. Kluwer Academic Publishers, 2000, doi: 10.1007/978-94-015-1279-4.

[2] L. Peng, "The Generation of (n,n(n-1),n-1) Permutation Group Codes for Communication Systems", *IEEE Transactions on Communications*, vol. 67, no. 7, pp. 4535-4549, 2019, doi: 10.1109/TCOMM.2019.2902149.

[3] E. Boss, V. Grosso, T. Güneysu, G. Leander, A. Moradi and T. Schneider, "Strong 8-bit sboxes with ecient masking in hardware", *Journal of Cryptographic Engineering*, no. 7, pp. 149-165, doi: 10.1007/s13389-017-0156-7.

[4] D. Fomin and D. Trifonov, "Ob aparatnoy realisazii odnogo klassa baytovyh podstanovok". [About hardware implementation of one class of byte permutations]. *Prikladnaya diskretnaya matematika*, Appendix 12, pp. 134-137, 2019, doi: 10.17223/2226308X/12/39 (in Russian).

[5] R. Oliynykov, R. Gorbenko, I. Gorbenko, O. Kazymyrov, Y. Ruzhevcev and Y. Gorbenko, "Pryntsypy pobudovy i osnovni vlastyvosti novoho natsional'noho standartu blokovoho shyfruvannya ukrayiny" [Construction principles and basic properties of Ukraine's new national block cypher encryption standard]. *Zakhyst informatsiyi*, vol. 2, pp. 142-157, 2015, doi: 10.18372/2410-7840.17.8789 (in Ukrainian).

[6] National Institute of Standards and Technology. (2001, Nov. 26). *Advanced Encryption Standard (AES)*. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.197.

[7] V. Tarasenko, O. Teslenko and O. Yanovska, "Problemy aparatnoi realisacii pidstanovok". [Problems of hardware implementation of permutations]. *Naykovi zapysky UNDIZ*, no. 2, pp. 52-58, 2007 (in Ukrainian).

[8] V. Tarasenko, O. Teslenko and O. Yanovska, "Vlastyvosti povnykh pidstanovok, yaki realizuyut'sya nayprostishym odnonapravlenym rehulyarnym OKKM". [Properties of complete permutations implemented by the simplest unidirectional regular OCSU]. *Radioelektronni i komp'yuterni systemy*, no. 6, pp. 123-128, 2010 (in Ukrainian).

[9] V. Tarasenko, O. Teslenko and O. Yanovska, "Mozhlyvosti naiprostishym dvonapravlenyh reguliarnyh odnovymirnyh kaskadiv konstruktyvnyh moduliv schodo realizacii riznyh povnyh pidstanovok". [Features of the simplest bidirectional regular one-dimensional cascades of structural units for the implementation of various complete permutations]. *Radioelektronni i kompyuterni systemy*, no. 7 (59), pp. 147-153, 2012 (in Ukrainian).

[10] O. Teslenko and M. Bondarchuk, "Implementation of arbitrary bitness permutations in one of the classes of linear structures", *Herald of Advanced Information Technology*, vol. 3, no. 1, pp. 406-417, 2020, doi: 10.15276/hait01.2020.7.

[11] O. Teslenko and M. Bondarchuk, "Use of implementations of arbitrary bitness permutations for cryptographic transformations", *KPI Science News*, no. 1-2, 2022, doi: 10.20535/kpisn.2022.1-2.263225.

[12] V. Glushkov, "Syntes cyfrovyh avtomatov". [Synthesis of Digital Automata]. Kyiv, 1967 (in Russian).

[13] M. Karandashov, "Issledovanie biektivnykh avtomatnykh otobrazhenii na kol'tse vychetov po moduliu 2^k", [Research bijective automaton mappings on the ring of residues modulo 2^k]. *Computer Science and Information Technologies*, pp. 148-152, 2014 (in Russian).

[14] A. Gill, *Introduction to the theory of finite-state machines*. McGraw-Hill electronic sciences series, 1962.

[15] M. Bondarchuk "Connected graph state matrix algorithm". [Online]. Available: https://github.com/MaksymBondarchuk/Article-3-materials/blob/master/ConnectedGraphStateAlgorithm.cs. Accessed on: 10.04.2023.

[16] M. Bondarchuk "Random non-repeating rows output matrix algorithm"[Online]. Available: https://github.com/MaksymBondarchuk/Article-3-materials/blob/master/RandomNonRepeatingRowsOutputAlgorithm.cs. Accessed on: 10.04.2023.

[17] G. Frobenius and L. Stickelberger, "*Ueber Gruppen von vertauschbaren Elementen*". ["About groups of interchangeable elements"], *Journal for Pure and Applied Mathematics*, 1879, doi: 10.1515/crll.1879.86.217 (in German).

[18] M. Bondarchuk "Reversed machine algorithm". [Online]. Available: https://github.com/MaksymBondarchuk/Article-3-materials/blob/master/ReversedMachine.cs. Accessed on: 10.04.2023.

[19] Summary of Virtex-6 FPGA Features. Virtex-6 Family Overview. XILINX DS150 (v2.5). [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds150.pdf. Accessed on: 13.04.2023.

[20] Sequence A001349 – Number of connected graphs with n nodes. The On-Line Encyclopedia of Integer Sequences® (OEIS®). [Online]. Available: http://oeis.org/A001349. Accessed on: 13.04.2023.

[21] J. van Lint and R. Wilson, *A Course in Combinatorics*. Cambridge University Press, 1992.

[22] Sequence A001187 – Number of connected labeled graphs with n nodes. The On-Line Encyclopedia of Integer Sequences® (OEIS®). [Online]. Available: http://oeis.org/A001187. Accessed on: 13.04.2023.

[23] SiSoftware Official Live Ranker "SiSoftware Zone". [Online]. Available: https://ranker.sisoftware.co.uk/top_device_all.php?q=d6ebdb. Accessed on: 03.01.2023.
[24] Memory Performance in a Nutshell. [Online]. Available: https://www.intel.com/content/www/us/en/developer/articles/technical/memory-performance-in-a-nutshell.html. Accessed on: 10.04.2023.
[25] RAM UserBenchmarks – 115 Memory Kits Compared. [Online]. Available: https://ram.userbenchmark.com. Accessed on: 10.04.2023.
[26] SSD UserBenchmarks – 1071 Solid State Drives Compared. [Online]. Available: https://ssd.userbenchmark.com. Accessed on: 10.04.2023.
[27] HDD UserBenchmarks – 1015 Hard Drives Compared. [Online]. Available: https://hdd.userbenchmark.com. Accessed on: 10.04.2023.
[28] V. Tarasenko, O. Teslenko and O. Yanovska, “Vykorystannya pryamykh ta obernenykh pidstanovok dovil′noyi rozryadnosti dlya pidvyshchennya efektyvnosti isnuyuchykh zasobiv ushchil′nennya danykh”. [The use of direct and inverse arbitrary bit permutations to improve the performance of existing data comperssors]. *Fourth International Scientific and Practical Conference* “*Metody ta zasoby koduvannya, zakhystu y ushchil’nennya informatsiyi*”, Vinnytsia, Ukraine, pp. 223-226, 2013 (in Ukrainian).
[29] V. Tarasenko, O. Teslenko and O. Yanovska, “Analiz vplyvu pidstanovok na dekompozytsiyu bulevykh funktsiy”. [Analysis of the effect of permutations on the decomposition of boolean functions]. *Herald of University “Ukrayina” Informatyka, obchyslyuval’na tekhnika ta kibernetyka*, no. 8, pp. 40-47, 2010 (in Ukrainian).
[30] V. Tarasenko, O. Teslenko and O. Yanovska, “Vykorystannya pidstanovok dlya nealhorytmichnoyi realizatsiyi koderiv ta dekoderiv zavadostiykoho koduvannya” [Use of permutations for non-algorithmic implementation of encoders and decoders of fault-tolerant coding]. *International Scientific Conference «Suchasni komp’yuterni systemy ta merezhi: rozrobka ta vykorystannya»*, L’viv, Ukraine, pp. 191-193, 2011 (in Ukrainian).
[31] O. Mel’nykova and A. Maslyennikova, “Pidstanovky dlya pidvyshchennya efektyvnosti prohramnoyi realizatsiyi alhorytmiv, yaki vykorystovuyut’ znakovo-tsyfrovi predstavlennya”. [Permutations for increasing the efficiency of software implementation of algorithms that use character-to-digital representations]. “*Tekhnichni nauky” Series*, Kharkiv, Ukraine, vol. 15, pp. 126-132, 2017 (in Ukrainian).
[32] A. Abornev, “Nelineynyye podstanovki na vektornom prostranstve, rekursivno-porozhdonnyye nad kol’tsom Galua kharakteristiki 4” [Nonlinear permutations on a recursively generated vector space over a Galois ring of characteristic 4], *Prikladnaya diskretnaya matematika*, vol. 7, pp. 40-41, 2014 (in Russian).

Відомості про авторів

Тесленко Олександр Кирилович – кандидат технічних наук, старший науковий співробітник, доцент кафедри системного програмування і спеціалізованих комп’ютерних систем.

Бондарчук Максим Юрійович – аспірант кафедри системного програмування і спеціалізованих комп’ютерних систем.

O.K. Teslenko, M.Y. Bondarchuk

IMPLEMENTATION OF ARBITRARY BITNESS PERMUTATIONS BASED ON COMBINED CASCADES OF STRUCTURAL UNITS

National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv