УДК 681.325

М. Т. Соломко, **Б. Б. Круліковський**
Національний університет водного господарства та природокористування, (м. Рівне),
кафедра обчислювальної техніки

ОПТИМІЗАЦІЯ ПЕРЕНЕСЕННЯ ПРИ ДОДАВАННІ ДВІЙКОВИХ ЧИСЕЛ У ТЕОРЕТИКО-ЧИСЛОВОМУ БАЗИСІ РАДЕМАХЕРА

Розглянуто математичні моделі обчислювальної схеми у вигляді орієнтованого ациклічного графу для побудови паралельних суматорів з паралельним способом перенесення. Продемонстровано зв'язок між обчислювальними кроками орієнтованого ациклічного графу та процесом перенесення одиниці у схемі багаторозрядного суматора, що дає змогу визначати оптимальну кількість перенесень у схемі багаторозрядного паралельного суматора з паралельним способом перенесення у теоретико–числовому базисі Радемахера. Процес додавання двійкових чисел у схемі суматора використовує алгоритм логарифмічного підсумовування.

Ключові слова: суматор, каскадна схема, напрямлений ациклічний граф, ТЧБ Радемахера.

TRANSFER OPTIMIZATION WHILE ADDING OF BINARY NUMBERS IN NUMBER-THEORETIC BASIS RADEMACHER

The mathematical model of computer circuit as directed acyclic graph for the construction of parallel adders with parallel transfer method. Demonstrated communication between computing steps directed acyclic graph and the process of transfer of units in the scheme multibit adder that can determine the optimal number of transfers in the scheme multibit parallel adder with parallel transfer method in theoretical and numerical basis Rademacher. The process of adding binary numbers in the adder circuit uses an algorithm logarithmic summation.

Key words: adder cascade scheme, directed acyclic graph, Rademacher TNB.

1. Вступ

Технологія обчислень у теоретико-числовому базисі (ТЧБ) Радемахера існує вже упродовж довгого часу; накопичено інформацію, створено математичні і програмні ресурси. Двійкова система є однією з найпростіших із позиційних систем числення і, як наслідок, електронна техніка, виготовлена на її основі, є найбільш надійною та універсальною.

Простіша система числення, ніж двійкова – унітарна, за якою у 60-ті роки було створено обчислювальну машину "Промінь". Але її до позиційних систем числення віднести важко, а швидкодія низька.

Діапазон додавання чисел у ТЧБ Радемахера становить:

$$x_D + y_D =< 2^k - 1, \qquad (1)$$

де k – розрядність числа. Кількість варіантів додавання для багаторозрядного суматора з повним суматором у першому розряді становить:

$$c = 2 \times 2^k \times 2^k, \tag{2}$$

де k – розрядність числа. Кількість варіантів додавання з напівсуматором у першому розряді багаторозрядного суматора дорівнює:

$$b = \sum_{n=0}^{2^k-1} 2^k - n \tag{3}$$

або

$$b = \sum_{n=1}^{2^k} n, \tag{4}$$

де k – розрядність числа.

Арифметичні пристрої з характеристиками (1)–(4) для низки прикладів не потребують додаткових потужностей апаратних засобів, тому їх собівартість утримується низькою, а це є основою для подальших розробок у цьому напрямі.

Методи арифметичних операцій реалізуються вентильними схемами з функціональних елементів у базисах, що складаються з функцій алгебри логіки. Мінімізація складності та глибини логічних схем є однією з центральних і практично важливих проблем у цій теорії. Сьогодні оптимізують схему певною мірою програмні засоби (наприклад, програма Logic Friday оптимізує схему у деяких базисах з асортименту доступних йому елементів), однак у загалом задачу оптимізації логічної схеми (зокрема схем суматорів) і сьогодні розв'язували емпірично. Тому актуальною є необхідність математичного дослідження процесу формування результату у схемі суматора, що дає можливість управляти схемою на етапі проектування.

2. Аналіз публікацій і окреслення проблеми

Двійкова система володіє багатьма незаперечними перевагами, проте вона має також низку суттєвих недоліків:

1. У двійковій системі можна подати у прямому коді тільки додатні числа. Для подання від'ємних чисел необхідно використовувати спеціальні коди – доповняльний та інверсний. Це ускладнює арифметичні структури комп'ютерів та знижує їх швидкодію.

2. Проблема нульової надлишковості. Всі комп'ютерні схеми та пристрої піддаються численним зовнішнім і внутрішнім впливам, наслідком чого є збої та відмови. Такі помилки у двійковій системі не можуть бути виявлені, оскільки всі двійкові кодові комбінації є дозволеними. Відсутність кодової надлишковості, яка б дозволяла виявляти помилки безпосередньо у процесі обчислень, є принциповим недоліком класичної двійкової системи числення.

3. Проблема "довгого перенесення". Максимальним час операції додавання стає тоді, коли перенесення, що виникло у першому розряді, проходить всі інші розряди (наприклад, при додаванні кодів 11 ... 11 і 00 ... 01), що є принциповим фактором зниження швидкодії комп'ютерів.

У роботі [1] зазначено, що головною перевагою мікропроцесорів Фібоначчі сьогодні є їх завадостійкість і можливість отримання достовірних даних на виході мікропроцесора.

Треба гадати, що перенесення в операціях додавання у ТЧБ Радемахера не належить до головної проблеми, підтвердженням чого можуть бути такі, наприклад, публікації: [2] – представлена структура суматора з вибором перенесення (Carry Select Adder) для здійснення паралельних обчислень у FPGA додатках. Продемонстровано оптимальні структури для реалізації FPGA. Результати моделювання використовують для перевірки теорії; у [3] наведено 180 нм КМОП-технологію процесу обчислення у схемі суматора за структурою Carry Select Adder на базі паралельного суматора з послідовним перенесенням. Запропонований підхід забезпечує прийнятний компроміс між собівартістю і продуктивністю

обчислень. Зазначимо, що суматор CLA (carry look ahead adder) займає більшу площу на кристалі, для процесу обчислення вимагає більшої потужності; у [4] розроблено математичну модель для оцінювання часових параметрів самосинхронізації перенесення для структури Carry Select Adder, наведено пілотні проекти зазначеного підходу; у [5] представлена 128-бітна структура Carry Select Adder для проектування системи HBIC. На основі модифікації 16, 32 та 64-бітних Carry Select Adder (CSLA) розроблено архитектуру, що має перспективу для зменшення площі суматора та затримки перенесення.

Зменшення апаратної складності електронного пристрою додавання за допомогою введення інформаційної надлишковості на підставі результатів моделювання суматора двійково-трійкової надлишкової системи числення розглянуто у [6].

У [7] показано та експериментально доведено ефективне використання каскадної схеми обчислень для підтримки запитів динамічних даних у хмарі.

У [8] продемонстровано, що операція векторно-матричного перемноження, що застосовується для вирішення задач обробки, аналізу сигналів і зображень та розпізнавання образів, реалізується формуванням парних добутків та багатоопераднього підсумовування.

Модифікація методу обчислення оператора групового підсумовування для нейромереж реального часу на основі вертикального та багатоопераднього підходів розглянуто у [9].

Застосування нуклеїнових кислот (NA) для логічних вентилів та обчислень в галузі біотехнології та біомедицини розглянуто у [10]. Зокрема продемонстровано логічний вентиль, що використовує каскадну схему для реалізації логіки ДНК. Логічні пристрої на основі нуклеїнових кислот вперше були уведено у 1994 р. і з тих пір наука бачить появу нових логічних систем для імітації математичних функцій, діагностики захворювань і навіть таких, що імітують біологічні системи.

Молекулярні логічні схеми з множиною компонентів, що розміщені у декілька шарів, можуть бути синтезовані з використанням ДНК. Чи є глибина схеми, що є стандартною мірою часової складності обчислень в електронних цифрових схемах, правильною мірою часової складності для хімічної схеми розглянуто в [11]. З цією метою наведено кількісний аналіз того, як час обчислення пов’язаний з розміром схеми та її каскадною архітектурою. Результати зазначеної роботи можуть бути застосовані для синтезу продуктивних і надійніших молекулярних схем.

На відміну від [7–11], у цій роботі розглянуто застосування каскадної схеми для операції додавання двійкових чисел за допомогою паралельного суматора з паралельним способом перенесення у ТЧБ Радемахера. Використання математичної моделі, що обґрунтовує операцію додавання у схемі паралельного суматора з паралельним способом перенесення, додає наочності методу та дозволяє формування складності алгоритму обчислень.

3. Мета та задачі дослідження

Метою роботи є, використовуючи математичну модель обчислювальної схеми, встановити зв’язок між обчислювальними кроками орієнтованого ациклічного графу і процесом перенесення одиниці у схемі багаторозрядного суматора, і визначити оптимальну кількість перенесень у схемі багаторозрядного паралельного суматора з паралельним способом перенесення у теоретико-числовому базисі Радемахера.

Для досягнення поставленої мети необхідно розв’язувати такі задачі:

1. Виявити коректну відповідність в організації процесу додавання чисел за каскадною схемою і процесу додавання бітів однойменних розрядів двійкових чисел.

2. Встановити адекватність математичної моделі у вигляді орієнтованого ациклічного графу та процесу додавання двійкових чисел у паралельному суматорі з паралельним способом перенесення, призначити параметри опису характеристик математичної моделі.

3. Вивести логічні рівняння оптимізованого суматора, що будується з врахуванням кількості обчислювальних кроків відповідного орієнтованого ациклічного графу.

4. Встановити складність алгоритму обчислення сигналів суми і перенесення оптимізованого паралельного суматора з паралельним способом перенесення ТБЧ Радемахера.

5. Скласти протокол обчислень процесу додавання двійкових чисел та протестувати синтезований суматор на відповідність результатів додавання двійкових чисел та складеного протоколу.

4. Збереження перенесення

Для демонстрації перенесення при арифметичному додаванні багаторозрядних двійкових чисел використаємо табличну організацію запису процесу підсумовування (табл. 1). До верхніх рядків табл. 1 запишемо два 4-розрядні числа – A і B.

Таблиця 1

Процес збереження перенесення при додаванні двійкових чисел

Число – A			a_3	a_2	a_1	a_0
Число – B			b_3	b_2	b_1	b_0
№		Переповнення	4-й розряд	3-й розряд	2-й розряд	1-й розряд
1	1	-	$a_3 \oplus b_3 = S_0^3$	$a_2 \oplus b_2 = S_0^2$	$a_1 \oplus b_1 = S_0^1$	$a_0 \oplus b_0 = S_0$
	2	$a_3 \wedge b_3 = I_0^4$	$a_2 \wedge b_2 = I_0^3$	$a_1 \wedge b_1 = I_0^2$	$a_0 \wedge b_0 = I_0^1$	-
2	1	-	$S_0^3 \oplus I_0^3 = S_1^3$	$S_0^2 \oplus I_0^2 = S_1^2$	$S_0^1 \oplus I_0^1 = S_1$	-
	2	$S_0^3 \wedge I_0^3 = I_1^4$	$S_0^2 \wedge I_0^2 = I_1^3$	$S_0^1 \wedge I_0^1 = I_1^2$	-	-
3	1	-	$S_1^3 \oplus I_1^3 = S_2^3$	$S_1^2 \oplus I_1^2 = S_2$	-	-
	2	$S_1^3 \wedge I_1^3 = I_2^4$	$S_1^2 \wedge I_1^2 = I_2^3$	-	-	-
4	1	-	$S_2^3 \oplus I_2^3 = S_3$	-	-	-
	2	$S_2^3 \wedge I_2^3 = I_3^4$	-	-	-	-
		$S_4 = I_0^4 \vee I_1^4 \vee I_2^4 \vee I_3^4$	S_3	S_2	S_1	S_0

Однойменні розряди чисел записують у загальний для них стовпчик табл. 1. Таблична організація запису процесу підсумовування передбачає цикли обчислень для кожного розряду числа. Число циклів дорівнює кількості розрядів двійкових чисел. Отже, для розглянутого прикладу кількість обчислювальних циклів дорівнюватиме чотирьом. Кожен цикл у табл. 1 наведено двома рядками. У перший рядок записують результат побітного додавання, у другий рядок – перенесення.

Арифметична процедура додавання складається з двох операцій: XOR і логічного множення I. Підсумовують числа з молодшого розряду. Суму за mod 2 – $a_0 \oplus b_0 = S_0$ – молодших розрядів записують у той самий стовпчик у першому рядку першого циклу обчислень. Перенесення молодших розрядів (результат логічного множення I – $a_0 \wedge b_0 = I_0^1$) записується у другий рядок першого циклу обчислень, у стовпчик, зсунутий на один розряд ліворуч. Аналогічні дії у першому циклі обчислень проводять з другими розрядами чисел –

суму за mod 2 – $a_1 \oplus b_1 = S_0^1$ – других розрядів записують у тому самому стовпчику у першому рядку першого циклу обчислень, перенесення других розрядів (результат логічного множення І – $a_1 \wedge b_1 = I_0^2$) записується у другий рядок першого циклу обчислень, у стовпчик, зсунутий на один розряд ліворуч. Завершивши розглянуті дії додавання для третього і четвертого розрядів, обчислення у першому циклі завершують.

Другий цикл повторює обчислення першого циклу з тією різницею, що тут обчислення проводять з результатами, отриманими у першому циклі. Так, сума за mod 2 – $S_0^1 \oplus I_0^1 = S_1$ – результатів обчислень у першому циклі, других розрядів, записується у тому самому стовпчику у першому рядку другого циклу обчислень. Перенесення результатів обчислень у першому циклі, других розрядів (результат логічного множення І – $S_0^1 \wedge I_0^1 = I_1^2$), записується у другий рядок другого циклу обчислень, у стовпчик, зсунутий на один розряд ліворуч. Аналогічні дії у другому циклі обчислень проводять з третіми розрядами чисел – сума за mod 2 – $S_0^2 \oplus I_0^2 = S_1^2$ – третіх розрядів записується у той самий стовпчик у перший рядок другого циклу обчислень, перенесення третіх розрядів (результат логічного множення І – $S_0^2 \wedge I_0^2 = I_1^3$) записується у другий рядок другого циклу обчислень у стовпчик, зсунутий на один розряд ліворуч. Завершивши розглянуті дії додавання для четвертих розрядів, обчислення у другому циклі завершуються.

У третьому циклі використовують результати обчислень другого циклу, а в четвертому – результати обчислень третього циклу (табл. 1). У підсумку буде отримано суму – S_3, S_2, S_1, S_0 заданих двійкових чисел – a_4, a_3, a_2, a_1 і b_4, b_3, b_2, b_1.

Розглянута таблична організація процесу додавання двійкових чисел (табл. 1) збігається з методом додавання чисел зі збереженням перенесення (carry-save adder). Для такого методу додавання вирішення проблеми перенесення зводиться до використання надлишкового кодування результату, коли у підсумку у кожній позиції отримується значення, що перевищує основу системи числення.

Запис суми (5) для молодшого розряду у табл. 1 є надлишковий, оскільки для $a_0 = 1$ і $b_0 = 1$ сума S_0 подається як $S_0=10$ або $S_0=2$. Останнє позначення суми у двійковій системі є нетрадиційним, але результат, як і раніше, однозначний.

a_1	a_0
b_1	b_0
$2-$й розряд	$1-$й розряд
-	$a_0 \oplus b_0 = S_0$
$a_0 \wedge b_0 = I_0^1$	-

(5)

Аналогічна надлишковість запису присутня і для сум інших розрядів.

У розглянутому прикладі (табл. 1) не використовується перенесення з молодших у старші розряди – цей процес замінено розширенням набору значущих цифр у кожному розряді результату. Однак зазначену процедуру не можна робити безперервно, адже для того, щоб повернутися до стандартного представлення числа, необхідно буде виконати всі перенесення, що й проілюстровано у табл. 1.

Використання суматорів зі збереженням перенесення дозволяє суттєво прискорити послідовні додавання, які, наприклад, необхідні при виконанні операції множення.

Приклад 1. За допомогою табличної організації процесу додавання обчислити суму 4-розрядних двійкових чисел 0001 і 0111 (табл. 2):

Таблиця 2

Додавання двійкових чисел 0001 і 0111
за обчислювальною схемою,
що визначає таблична організація процесу

Число - А		0	0	0	1
Число - В		0	1	1	1
1	-	0	1	1	0
	0	0	0	1	-
2	-	0	1	0	-
	0	0	1	-	-
3	-	0	0	-	-
	0	1	-	-	-
4	-	1	-	-	-
	0	-	-	-	-
-	-	1	0	0	0

Приклад 2. За допомогою табличної організації процесу додавання обчислити суму 4-розрядних двійкових чисел 0011 і 0011(табл. 3):

Таблиця 3

Додавання двійкових чисел 0011 і 0011
за обчислювальною схемою,
що визначає таблична організація процесу

Число - А		0	0	1	1
Число - В		0	0	1	1
1	-	0	0	0	0
	0	0	1	1	-
2	-	0	1	1	-
	0	0	0	-	-
3	-	0	1	-	-
	0	0	-	-	-
4	-	0	-	-	-
	0	-	-	-	-
-	-	0	1	1	0

Використовуючи табл. 1, запишемо логічні рівняння 4-розрядного суматора з табличною організацією додавання двійкових чисел.

$$S_0 = a_0 \oplus b_0,$$

$$S_1 = (a_1 \oplus b_1) \oplus (a_0 \wedge b_0),$$

$$S_2 = ((a_2 \oplus b_2) \oplus (a_1 \wedge b_1)) \oplus ((a_1 \oplus b_1) \wedge (a_0 \wedge b_0)),$$

$$S_3 = (((a_3 \oplus b_3) \oplus (a_2 \wedge b_2)) \oplus ((a_2 \oplus b_2) \wedge (a_1 \wedge b_1))) \oplus$$

$$\oplus(((a_2 \oplus b_2) \oplus (a_1 \wedge b_1)) \wedge ((a_1 \oplus b_1) \wedge (a_0 \wedge b_0))).$$

У випадку, коли діапазон чисел, в якому працює суматор, буде фіксованим, можливість у нарощуванні розрядності схеми суматора відпадає. Тоді в останньому розряді суматора можна використати логічні елементи OR, що спростить структуру суматора. Рівняння для S_3 матиме вигляд.

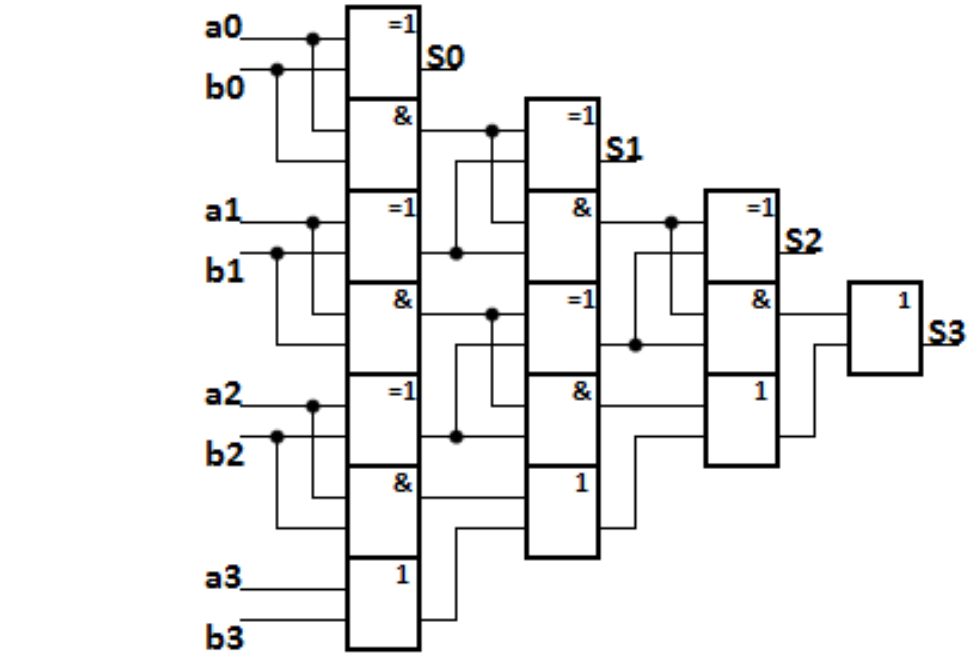
$$S_3 = (a_3 \vee b_3) \vee (a_2 \wedge b_2) \vee ((a_2 \oplus b_2) \wedge (a_1 \wedge b_1)) \vee$$

$$\vee(((a_2 \oplus b_2) \oplus (a_1 \wedge b_1)) \wedge ((a_1 \oplus b_1) \wedge (a_0 \wedge b_0))). \quad (6)$$

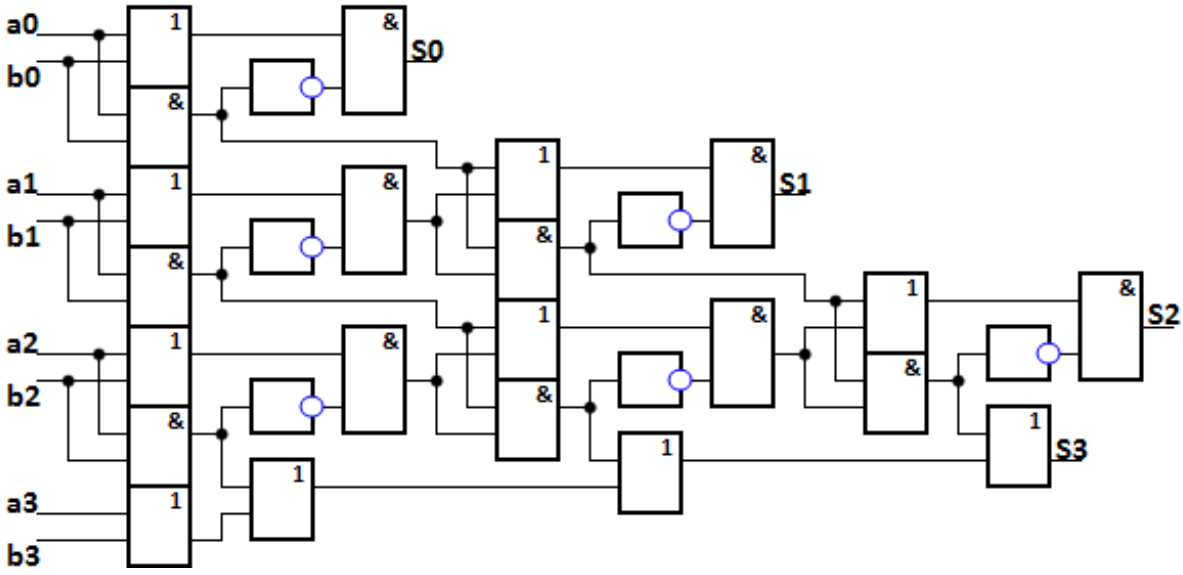
На рис. 1 подано логічну структуру 4-розрядного суматора, що реалізує табличну організацію процесу додавання двійкових чисел.

За класифікацією суматор з табличною організацією додавання двійкових чисел необхідно зарахувати до різновиду паралельних суматорів з паралельним способом перенесення. Особливістю логічної структури суматора на рис. 1 є відсутність ланцюгів для прискореного перенесення, що зменшує технологічні проблеми при збільшенні розрядності суматора.

Сигнал переповнення у схемі суматора на рис. 1 можна отримати за допомогою додаткової схеми з чотирьох логічних елементів.



а



б

Рис. 1. Логічна структура 4-розрядного суматора з табличною організацією процесу додавання двійкових чисел: а – схема суматора з умовним графічним позначенням логічного елемента XOR; б – схема суматора з одним із варіантів відкритої структури логічного елемента XOR

5. Модель паралельного додавання двійкових чисел ТЧБ Радемахера

Порозрядне додавання двійкових чисел за невеликими відмінностями подібне до алгоритму здвоювання за багатоопераднього підсумовування, коли одночасно додаються сусідні пари доданків, а потім їхні суми (табл. 4).

Таблиця 4

Алгоритм здвоєння ($n=2^3=8$)

Кроки	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
1	$e_1 + e_2$		$e_3 + e_4$		$e_5 + e_6$		$e_7 + e_8$	
2	$e_1 + e_2 + e_3 + e_4$				$e_5 + e_6 + e_7 + e_8$			
3	$e_1 + e_2 + e_3 + e_4 + e_5 + e_6 + e_7 + e_8$							

Якщо $n = 2^k$, де n – число доданків, то алгоритм здвоєння складається з k кроків (тактів): на першому кроці виконується n / 2 додавань, на другому – n / 4 , …, на останньому – одне додавання. Кількість кроків k визначають за формулою:

$$k = \log_2 n. \tag{7}$$

Такий варіант багатоопераднього додавання реалізується каскадною схемою (“пірамідою”) [12 – 15] (рис. 2). Існує також алгоритм рекурсивного підсумовування [14].

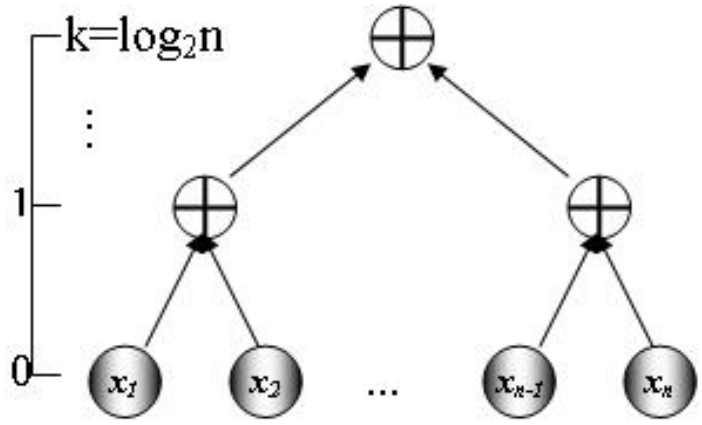


Рис. 2. Каскадна схема алгоритму багатоопераднього додавання (логарифмічне підсумовування)

Використовуючи процедуру багатоопераднього додавання за допомогою каскадної схеми, бачимо, що для процесу паралельного додавання двійкових чисел парами даних тут будуть біти однойменних розрядів, для кожної з яких обчислюється сума.

Потім, аналогічно до процедури багатоопераднього додавання, всі отримані суми однойменних розрядних пар двійкових чисел зі своєю специфікою також розбиваються на пари, і знову виконується додавання значень пар і т. д.

У підсумку значення старшого розряду суми двійкових чисел можна зіставити зі значенням загальної суми при багатоопераднoму додаванні. Крім зазначеної суми старшого розряду, у процесі паралельного додавання двійкових чисел використовуються також проміжні результати у вигляді значень сум попередніх розрядів двійкових чисел.

Обчислювальну схему паралельного додавання двійкових чисел можна визначити орієнтованим ациклічним графом (рис. 3), який являє собою бінарне дерево, де зокрема прийнятю такі параметри: k – кількість кроків у часі; ω – загальна кількість операцій алгоритму; τ – час виконання одного кроку; $T = \tau \cdot k$ – час виконання алгоритму; L – кількість типів операцій тощо.

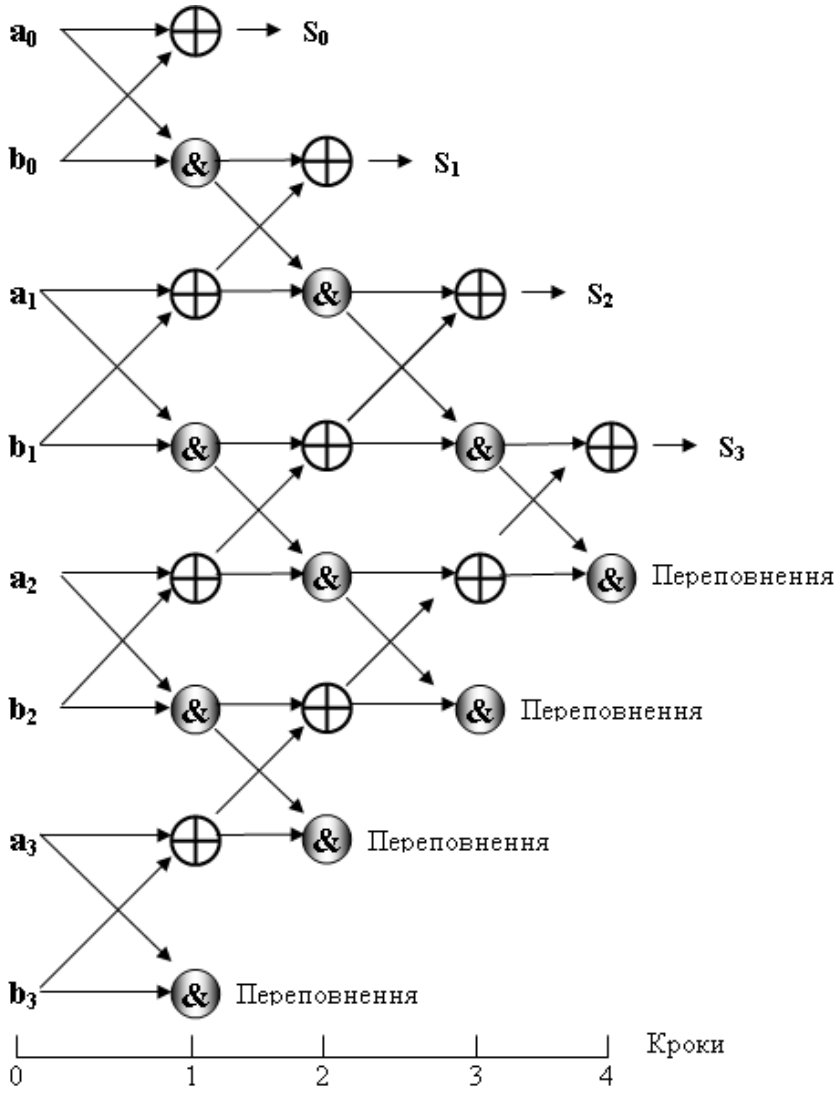


Рис. 3. Орієнтований ациклічний граф обчислювальної схеми паралельного додавання зі збереженням перенесення 4-розрядних двійкових чисел ТЧБ Радемахера

В обчислювальній схемі на рис. 3 використано дві логічні операції: AND і XOR, число обчислювальних кроків у ній дорівнює розрядності двійкових чисел. Наприклад, для паралельного додавання 4-розрядних двійкових чисел необхідно чотири кроки (рис. 3).

Табличну організацію обчислювальної схеми на рис. 3 подано у табл. 5. Верхній індекс параметра табл. 5 вказує на порядковий номер обчислювального кроку.

Таблиця 5

Таблична організація додавання 4-розрядних двійкових чисел за обчислювальною схемою, що визначає орієнтований ациклічний граф

№	a_3		a_2		a_1		a_0	
	b_3		b_2		b_1		b_0	
1	$a_3 \wedge b_3 = I_3^1$	$a_3 \oplus b_3 = S_3^1$	$a_2 \wedge b_2 = I_2^1$	$a_2 \oplus b_2 = S_2^1$	$a_1 \wedge b_1 = I_1^1$	$a_1 \oplus b_1 = S_1^1$	$a_0 \wedge b_0 = I_0^1$	$a_0 \oplus b_0 = S_0$
2		$S_3^1 \wedge I_2^1 = I_{23}^2$	$S_3^1 \oplus I_2^1 = S_{23}^2$	$S_2^1 \wedge I_1^1 = I_{12}^2$	$S_2^1 \oplus I_1^1 = S_{12}^2$	$S_1^1 \wedge I_0^1 = I_{01}^2$	$S_1^1 \oplus I_0^1 = S_1$	
3			$S_{23}^2 \wedge I_{12}^2 = I^3$	$S_{23}^2 \oplus I_{12}^2 = S^3$	$S_{12}^2 \wedge I_{01}^2 = I^3$	$S_{12}^2 \oplus I_{01}^2 = S_2$		
4				$S^3 \wedge I^3 = I^4$	$S^3 \oplus I^3 = S_3$			
					S_3	S_2	S_1	S_0

Приклад 3. За допомогою орієнтованого ациклічного графу обчислити суму 4-розрядних двійкових чисел 0001 і 0111 (табл. 6):

Таблиця 6

Процес додавання двійкових чисел 0001 і 0111 за обчислювальною схемою, що визначає орієнтований ациклічний граф

№	0		0		0		1	
	0		1		1		1	
1	0	0	0	1	0	1	1	0
2		0	0	0	1	1	0	
3			0	0	1	0		
4				0	1			
					1	0	0	0

У 4-му рядку табл. 6 праворуч записано значення старшого розряду суми двійкових чисел, ліворуч – значення переповнення. Процес додавання використовує чотири обчислювальні кроки.

Обчислювальну схему паралельного додавання 4-розрядних двійкових чисел з логічним елементом OR в останньому розряді подано на рис. 4.

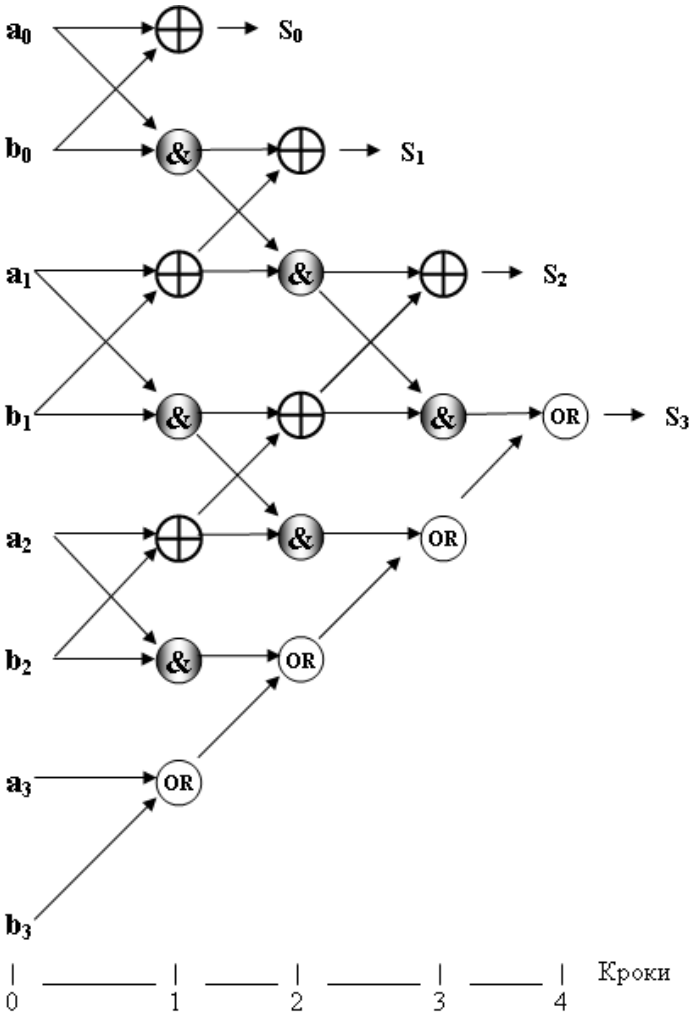


Рис. 4. Обчислювальна схема паралельного додавання зі збереженням перенесення 4-розрядних двійкових чисел з логічним елементом OR в останньому розряді

Табличну організацію обчислювальної схеми на рис. 4 подано у табл. 7.

Таблиця 7

Таблична організація додавання 4-розрядних двійкових чисел з обчислювальною схемою, що визначає орієнтований ациклічний граф з логічним елементом OR в останньому розряді

№	a_3	a_2		a_1		a_0	
	b_3	b_2		b_1		b_0	
1	$a_3 \vee b_3 = S_3^1$	$a_2 \wedge b_2 = I_2^1$	$a_2 \oplus b_2 = S_2^1$	$a_1 \wedge b_1 = I_1^1$	$a_1 \oplus b_1 = S_1^1$	$a_0 \wedge b_0 = I_0^1$	$a_0 \oplus b_0 = S_0$
2	$S_3^1 \vee I_2^1 = S_{23}^2$		$S_2^1 \wedge I_1^1 = I_{12}^2$	$S_2^1 \oplus I_1^1 = S_{12}^2$	$S_1^1 \wedge I_0^1 = I_{01}^2$	$S_1^1 \oplus I_0^1 = S_1$	
3		$S_{23}^2 \vee I_{12}^2 = S^3$		$S_{12}^2 \wedge I_{01}^2 = I^3$	$S_{12}^2 \oplus I_{01}^2 = S_2$		
4			$S^3 \vee I^3 = S_3$				
			S_3		S_2	S_1	S_0

Приклад 4. За допомогою орієнтованого ациклічного графу з логічним елементом OR в останньому розряді додати 4-розрядні двійкові числа 0011 і 0011 (табл. 8):

Таблиця 8

Процес додавання двійкових чисел 0011 і 0011 за обчислювальною схемою, що визначає орієнтований ациклічний граф з логічним елементом OR в останньому розряді

№	0	0		1		1	
	0	0		1		1	
1	0	0	0	1	0	1	0
2		0	0	1	0	1	
3			0	0	1		
4				0			
				0	1	1	0

У 4-му рядку табл. 8 записано значення старшого розряду суми двійкових чисел. У процесі додавання використано чотири обчислювальні кроки.

Використовуючи табл. 7, запишемо логічні рівняння 4-розрядного суматора, обчислювальну схему якого визначає орієнтований ациклічний граф з логічним елементом OR в останньому розряді.

$$S_0 = a_0 \oplus b_0,$$

$$S_1 = S_1^1 \oplus I_0^1 = (a_1 \oplus b_1) \oplus (a_0 \wedge b_0),$$

$$S_2 = S_{12}^2 \oplus I_{01}^2 = (S_2^1 \oplus I_1^1) \oplus (S_1^1 \wedge I_0^1) = ((a_2 \oplus b_2) \oplus (a_1 \wedge b_1)) \oplus ((a_1 \oplus b_1) \wedge (a_0 \wedge b_0)),$$

$$S_3 = S^3 \vee I^3 = (S_{23}^2 \vee I_{12}^2) \vee (S_{12}^2 \wedge I_{01}^2) = ((S_3^1 \vee I_2^1) \vee (S_2^1 \wedge I_1^1)) \vee ((S_2^1 \oplus I_1^1) \wedge (S_1^1 \wedge I_0^1)) =$$
$$= (((a_3 \vee b_3) \vee (a_2 \wedge b_2)) \vee ((a_2 \oplus b_2) \wedge (a_1 \wedge b_1))) \vee$$
$$\vee (((a_2 \oplus b_2) \oplus (a_1 \wedge b_1)) \wedge ((a_1 \oplus b_1) \wedge (a_0 \wedge b_0))).$$

або

$$S_0 = a_0 \oplus b_0,$$
$$S_1 = (a_1 \oplus b_1) \oplus (a_0 \wedge b_0),$$
$$S_2 = ((a_2 \oplus b_2) \oplus (a_1 \wedge b_1)) \oplus ((a_1 \oplus b_1) \wedge (a_0 \wedge b_0)),$$
$$S_3 = (((a_3 \vee b_3) \vee (a_2 \wedge b_2)) \vee ((a_2 \oplus b_2) \wedge (a_1 \wedge b_1))) \vee$$
$$\vee(((a_2 \oplus b_2) \oplus (a_1 \wedge b_1)) \wedge ((a_1 \oplus b_1) \wedge (a_0 \wedge b_0))). \quad (8)$$

Оскільки рівняння (6) і (8) збігаються, логіка обчислень табл. 1 і табл. 7 однакова. Зазначимо, що в обчислювальному процесі табл. 7 використано обчислювальні кроки, а в обчислювальному процесі табл. 1 – процедуру перенесення

Твердження. Число обчислювальних кроків орієнтованого ациклічного графа, що моделює процес обчислення суматора, визначає оптимальне число перенесень у схемі багаторозрядного паралельного суматора з паралельним способом перенесення у ТЧБ Радемахера.

У випадку, коли синтезований суматор отримав більше число перенесень порівняно з числом обчислювальних кроків відповідного орієнтованого ациклічного графа, то такий суматор буде неоптимальним стосовно числа обчислювальних операцій. Зокрема, за критерієм орієнтованого ациклічного графу, неоптимальною є схема 8-bit Brent-Kung PPA [16], зазначену схему, однак, можна використати у системі суперсуматора.

Отже, число обчислювальних кроків визначає мінімально достатнє число перенесень у схемі суматора, що, своєю чергою, дає гіперпараметр для оптимізації структури суматора у процесі його синтезу. Логічні рівняння оптимізованого 4-розрядного суматора з числом перенесення – чотири – є, наприклад, такі:

$$S_0 = a_0 \oplus b_0,$$
$$S_1 = (a_1 \oplus b_1) \oplus (a_0 \wedge b_0),$$
$$S_2 = (a_2 \oplus b_2) \oplus ((a_1 \wedge b_1) \vee ((a_1 \vee b_1) \wedge (a_0 \wedge b_0))),$$
$$S_3 = (a_3 \vee b_3) \vee (a_2 \wedge b_2) \vee ((a_2 \vee b_2) \wedge (a_1 \wedge b_1)) \vee$$
$$\vee((a_2 \vee b_2) \wedge ((a_1 \vee b_1) \wedge (a_0 \wedge b_0))). \quad (9)$$

Час T виконання додавання двійкових чисел суматора становить:

$$T = \tau \cdot k,$$

де τ – час виконання одного кроку (перенесення), k – число кроків (перенесень). Причому оптимальний суматор (9) працюватиме швидше, оскільки містить менше операцій XOR порівняно зі схемою суматора на рис. 1.

Суматор на рис. 1 виконує 48 логічних операцій, з них XOR – 11, AND – 11, OR – 4. Оптимальний суматор (9) виконує 34 логічних операції, з них XOR – 5, AND – 10, OR – 9.

Враховуючи те, що логіка елемента XOR використовує чотири логічних операції (рис. 1. b), можна оцінити показник прискорення S роботи оптимального суматора:

$$S = T_1 / T_{опт.} = 59/39 = 1,5128 = 51,28\%,$$

де T_1, $T_{опт}$ – число логічних операцій неоптимального і оптимального суматорів відповідно.

6. Обговорення результатів моделювання процесу додавання двійкових чисел у паралельному суматорі з паралельним способом перенесення ациклічним графом з двома логічними операціями – AND і XOR

Результати досліджень у цій роботі демонструють те, що обчислювальні кроки орієнтованого ациклічного графу і перенесення одиниці з попередніх розрядів суматора являють собою один об’єкт. Обчислення, організовані каскадною схемою, мають логарифмічну складність, і оскільки ациклічний граф подає каскадну схему, то число обчислювальних кроків графу оптимізує (вказує на мінімально достатнє) число перенесень для операції додавання багаторозрядних двійкових чисел у схемі паралельного суматора з паралельним способом перенесення ТЧБ Радемахера. Цільовою функцією такого процесу оптимізації числа перенесень суматора є рівняння (7).

Зазначимо, що метод емпіричної побудови (зокрема і програмний) паралельного суматора з паралельним способом перенесення не гарантує отримання структури суматора з мінімальним числом перенесень і, як наслідок, синтезована у такий спосіб схема суматора може потребувати більшого числа обчислювальних операцій.

Зв'язок між числом обчислювальних кроків орієнтованого ациклічного графу і числом перенесень у схемі паралельного суматора з паралельним способом перенесення вказує на доцільність зіставлення структури суматора з відповідним орієнтованим ациклічним графом, а доцільність – це необхідність, тому корисність цього дослідження полягає у тому, що вони зумовлюють процес зіставлення структури суматора з відповідним орієнтованим ациклічним графом з метою встановлення оптимального числа перенесень для операції додавання двійкових чисел. Отже, результати дослідження можуть стати складовою технології проектування електронних схем суматорів, оскільки:

– дають зрозуміти, якою є структура суматора;

– вчать керувати схемою суматора на етапі проектування;

– дають можливість прогнозувати наслідки реалізації заданої структури суматора.

Перспективою подальшого розгляду паралельних суматорів з паралельним способом перенесення може бути виявлення умов зменшення обчислювальної складності за числом обчислювальних операцій у схемі суматора, наприклад, до $O(n-1)$.

Висновки

1. Встановлено, що обчислення сигналів суми і перенесення у паралельних суматорах з паралельним способом перенесення ТЧБ Радемахера можуть бути обґрунтовані математичною моделлю у вигляді напрямленого ациклічного графу, що являє собою бінарне дерево.

2. Виявлено, що показник ефективності орієнтованого ациклічного графу у вигляді числа обчислювальних кроків визначає оптимальне число перенесень у схемі багаторозрядного паралельного суматора з паралельним способом перенесення у ТЧБ Радемахера.

3. Встановлено, що число обчислювальних кроків для розглянутих моделей паралельних суматорів (моделі 4-розрядних суматорів) з паралельним способом перенесення дорівнює величині розрядності двійкових чисел n. Тобто складність алгоритму обчислення сигналів суми і перенесення паралельного суматора з паралельним способом перенесення у ТБЧ Радемахера становить $O(n)$ і є лінійною – число обчислювальних операцій алгоритму лінійно зростає зі збільшенням n.

4. Виявлено, що обчислюють сигнал суми і перенесення у схемі паралельного суматора з паралельним способом перенесення за алгоритмом логарифмічного додавання.

5. Встановлено, що логіка роботи оптимізованого суматора за допомогою розглянутої математичної моделі відповідає протоколу обчислень паралельного суматора з паралельним способом перенесення.

1. Борисенко А. А. Реплика по поводу микропроцессоров Фибоначчи [Электронный ресурс] / А. А. Борисенко // Академия Тринитаризма. – 01.09.2011. – Эл. № 77-6567, публ. 16805. – Режим доступа: \www/URL: http://www.trinitas.ru/rus/doc/0232/009a/02321223.htm 2. Sajesh Kumar, Mohamed Salih (2012) Efficient Carry Select Adder Design for FPGA. Procedia Engineering, 30, 449 – 456 http://www.sciencedirect.com/science/article/pii/S1877705812008946 3. Yogita Hiremath (2014) A Novel 8-bit Carry Select Adder using 180nm CMOS Process Technology. International Journal of Emerging Engineering Research and Technology, Volume 2, Issue 6, September, 187–194 http://www.ijeert.org/pdf/v2-i6/25.pdf 4. Balasubramanian, P., Jacob Prathap Raj, C., Anandi, S., Bhavanidevi, U., Mastorakis, N. E. (2013) Mathematical Modeling of Timing Attributes of Self-Timed Carry Select Adders. Recent Advances in Circuits, Systems, Telecommunications and Control, 228–243 http://www.wseas.us/e-library/conferences/2013/Paris/CCTC/CCTC-34.pdf 5. Chithra, M., Omkareswari, G. (2013) 128-bit Carry Select Adder Having Less Area And Delay. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 2, Issue 7, July 2013, 3112–3118

http://www.ijareeie.com/upload/2013/july/35E_128-BIT.pdf 6. Куницкая, С. Ю. Синтез устройства сложения в двоично-троичной избыточной системе счисления [Электронный ресурс] / С. Ю. Куницкая // Вісник ЧДТУ: Інформаційні технології, обчислювальна техніка і автоматика, 2015, № 1, С. 86–90. – Режим доступа: \www/URL: http://www.irbis-nbuv.gov.ua/cgi bin/irbis_nbuv/cgiirbis_64.exe?I21DBN=LINK&P21DBN=UJRN&Z21ID=&S21REF=10&S21CNR=20& S21STN=1&S21FMT=ASP_meta&C21COM=S&2_S21P03=FILA=&2_S21STR=Vchdtu_2015_1_16 7. Tang, Y., Liu, L., Tech, G., Tatemura, J., Hacigumus, H. (2015) KTV-Tree: Interactive Top-K Aggregation on Dynamic Large Dataset in the Cloud. IEEE 35th International Conference on Distributed Computing Systems Workshops, June 29 2015-July 2 2015, 136-142 https://pdfs.semanticscholar.org/ cb3e/ae43d0e3465cd52acf73de974bcc374e6665.pdf 8. Мартинюк, Т. Б. Аналіз операційного базису для нейромережевих інтелектуальних систем [електронний ресурс] / Т. Б. Мартинюк, А. В. Кожем'яко, Н. О. Денисюк, Т. Ю. Позднякова // Інформаційні технології та комп'ютерна інженерія. – 2015. – № 2. – С. 83–87. – \www/URL: http://www.google.com.ua/url?url=http://irbisnbuv.gov.ua/ cgibin/irbis_nbuv/cgiirbis_64.exe%3FC21COM%3D2%26I21DBN%3DUJRN%26P21DBN%3DUJRN%2 6IMAGE_FILE_DOWNLOAD%3D1%26Image_file_name%3DPDF/Itki_2015_2_15.pdf&rct=j&frm=1& q=&esrc=s&sa=U&ved=0ahUKEwj0pMKMxLjMAhWKIpoKHfS6BOgQFgg7MAg&usg=AFQjCNG8Mu VpB_g7LaFHXW8ZVNYKn3hc0A 9. Цмоць І. Модифікований метод та нвіс-структура пристрою групового підсумовування для нейроелемента [Електронний ресурс] / І. Цмоць, О. Скорохода, Б. Балич // Вісник Нац. ун-ту "Львівська політехніка". – 2012. – № 732 : Комп'ютерні науки та інформаційні технології. – С. 51–57. – Режим доступу: \WWW/URL: http://ena.lp.edu.ua:8080/ bitstream/ntb/14865/1/9_Tsmots_51_57_732.pdf 10. Wu C., Wan Sh., Hou W., Zhang L., Xu J., Cui Ch., Wang Y., Hu J., Tan W. (2015) A survey of advancements in nucleic acid-based logic gates and computing for applications in biotechnology and biomedicine. Chem. Commun., 2015, 51, 3723–3734 https://www.chem.ufl.edu/wp-content/uploads/sites/39/pubs/2015/A%20Survey%20of%20Advancements% 20in%20Nucleic%20Acid-based%20Logic%20Gates%20and%20Computing%20for%20Applications% 20in%20Biotechnology%20and%20Biomedicine.pdf 11. Seelig, Georg and Soloveichik, David (2009) Time-Complexity of Multilayered DNA Strand Displacement Circuits. In: DNA computing and molecular programming. Lecture Notes in Computer Science. No.5877. Springer, Berlin, pp. 144–153. http://www.dna.caltech.edu/Papers/CRN_circuit_complexity.pdf 12. Гамаюн В. П. О развитии многооперандных вычислительных структур [Текст] / В. П. Гамаюн // Управляющие системы и машины. – 1990. – № 4. – С. 31–33. 13. Гамаюн В. П. Автореф. дис. д-ра техн. наук: 05.13.13 / В. П. Гамаюн. – НАН України. Ін-т кібернетики ім. В. М. Глушкова. – К., 1999. – 33 с. 14. Мартинюк Т. Б., Методи та засоби паралельних перетворень векторних масивів даних [Монографія] / Т. Б. Мартинюк, В. В. Хомюк – Вінниця: "УНІВЕРСУМ-Вінниця", 2005. – 202 с. 15. Мартинюк Т. Б. Рекурсивні алгоритми багатооперандної обробки інформації [Монографія] / Т. Б. Мартинюк – Вінниця : "УНІВЕРСУМ-Вінниця", 2000. – 216 с. 16. Class ECE6332 Fall 12 Group-Fault-Tolerant Reconfigurable PPA. http://venividiwiki.ee.virginia.edu/mediawiki/index.php/ ClassECE6332Fall12Group-Fault-Tolerant_Reconfigurable_PPA