УДК 681.325

М. Т. Соломко[1], Б. Б. Круліковський[2], Я. М. Николайчук[3]
[1]Рівненський державний гуманітарний університет,
кафедра ІКТ та МВІ,
[2]Національний університет водного господарства та природокористування, м. Рівне,
кафедра обчислювальної техніки,
[3]Тернопільський національний економічний університет,
кафедра спеціалізованих комп'ютерних систем

ПАРАЛЕЛЬНИЙ СУМАТОР БЕЗ ПЕРЕНЕСЕННЯ НА ЛОГІЧНИХ ЕЛЕМЕНТАХ XAND

Розглянуто обчислення сигналів суми в паралельних схемах суматорів без перенесення на логічних елементах XAND. Подана структура логічного елемента XAND, синтаксис його функції, схема паралельного суматора без перенесення на логічних елементах XAND. Продемонстрована таблиця істинності для перевірки логіки схеми суматора.

Ключові слова: суматор, паралельний суматор без перенесення, додавання кодів, коди поля Ґалуа, логічний елемент XAND.

PARALLEL ADDER WITHOUT CARRY BASED ON XAND GATES

In the article the calculation of the amount of signals in parallel schemes of adders without transfering by the logical elements XAND. The composition of logic element XAND is represented and the syntax of its function. The scheme of parallel adder without transfering by the logic elements XAND is represented. Demonstrated a truth table to check logic schemes of adder.

Key words: adder, parallel adder without transfer, adding codes, Galois field codes, logic element XAND.

Вступ

Обробка інформації цифровими методами зводиться до послідовності дій (операцій) над числами. У цифровій апаратурі основним блоком, в якому безпосередньо виконується оброблення інформації, є процесорний пристрій.

Розвиток теорії універсальних та спеціалізованих процесорів тісно пов'язаний з відповідним розвитком двійкової системи числення, тобто теоретико-числового базису Радемахера [5]. Сучасні досягнення у створенні високопродуктивних процесорів ґрунтуються на розробленнях теорії паралельних обчислень.

Зростання вимог до швидкодії процесорів стимулюють дослідження у застосуванні інших, відмінних від базису Радемахера, теоретико-числових базисів (ТЧБ). Наприклад, відомі успішні застосування базису Крестенсона, що породжує систему числення залишкових класів (СЗК), базису Ґалуа, що породжує коди поля та систему числення Ґалуа, а також базису Уолша, що використовується для створення комунікаційних та сигнальних процесорів до застосування їх у комп'ютерних мережах [1, 3, 7, 8].

Перенесення одиниці до старшого розряду під час виконання операції додавання у базисі Радемахера призводить до зниження швидкодії встановлення сталих значень сигналів на виходах Si однорозрядних суматорів. Тривалість перехідного процесу в суматорі Тз.с залежить від розрядності чисел n та часу затримки сигналів на кожному розряді суматора tз.р.

$$Т_{з.с.} = n \times t_{з.р.}$$

Максимальним час операції додавання стає тоді, коли перенесення, що виникло у першому розряді, проходить всі інші розряди (наприклад, у разі складання кодів 11...11 і 00…01).

Суматори з наскрізним перенесенням є практичним пристроєм для реалізації додавання з порівняно невеликою довжиною слова. У більшості настільних комп'ютерів використовується довжина слова 32 біти, сервер для обробки даних зазвичай потребує 64 біти; високопродуктивні мейнфреми, суперкомп'ютери або мультимедійні процесори, подібні до Sony PlayStation 2, повинні забезпечувати обробку даних розрядністю до 128 бітів.

Аналіз публікацій і окреслення проблеми

Значний внесок у розвиток теорії арифметичних операцій та архітектури суматорів у базисі Ґалуа зробили українські вчені: В. П. Тарасенко, О. К. Тесленко, А. І. Роговенко, Я. М. Николайчук, О. М. Заставний, П. В. Гуменний [6, 11]. Теоретичні питання та проектування логіки суматорів у ТЧБ Радемахера розглянуто у роботах Н. Воробйова [2], Рабаї Жана М. [9].

Оптимізація схеми суматора проводиться або на рівні логічних елементів (наприклад, з використанням у ланцюгу перенесення найбільш швидкодіючих елементів, зокрема, логічний елемент І-АБО-НІ має менший час затримки, порівняно з логічним елементом І-АБО, якщо останній реалізується структурою І-АБО-НІ-НІ), або на рівні схеми (наприклад, застосовують структурні методи прискорення проходження сигналу перенесення). Доволі часто задача оптимізації логічної схеми упродовж усього існування ЕОМ розв'язувалася емпірично. Тепер оптимізацію схеми до певної міри вирішують програмні засоби, наприклад безкоштовна програма Logic Friday здійснює оптимізацію схеми у ряді базисів – з асортименту доступних елементів [12].

Продуктивність суматора лінійно залежить від величини його розрядності, тому, розробляючи швидкодіючі суматори, доцільно дослідити залежність швидкодії встановлення сталих значень сигналів на виходах Si однорозрядних суматорів від складності та глибини схеми суматора з метою вибору оптимальних рішень.

Метою роботи є побудова багаторозрядного паралельного суматора без перенесення на логічних елементах XAND.

1. Логічний елемент XAND

Таблиця істинності логічного елемента XAND «Виключаюче І» подібна до таблиці істинності логічного елемента І, за винятком одного випадку (табл. 1).

Таблиця 1

Таблиця істинності логічного елемента XAND

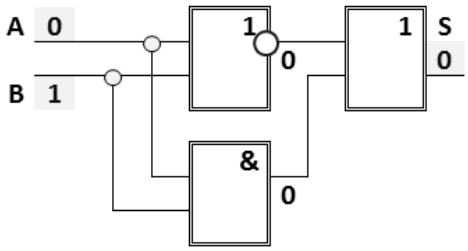
A	B	XAND
0	0	1
0	1	0
1	0	0
1	1	1

Табл. 1 реалізує також логічний елемент XNOR, скорочено NOR – виключаюче АБО-НІ (логічна операція) – заперечення альтернативної диз'юнкції.

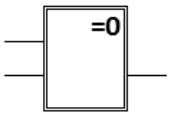
Структуру логічного елемента XAND і його умовне графічне зображення подано на рис. 1.

Синтаксис функції XAND:

$$A \text{ XAND } B = -A \vee B \wedge A \vee -B.$$



а



б

Рис. 1. Структура логічного елемента XAND (а)
і його умовне графічне зображення (б)

Операцію XAND позначимо значком – ⊞. Приклади операції XAND:

$$0 \boxplus 0 = 1$$
$$0 \boxplus 1 = 0$$
$$0 \boxplus 1 = 0$$
$$1 \boxplus 1 = 1$$

На логічних елементах XAND можлива побудова повного суматора (рис. 2).

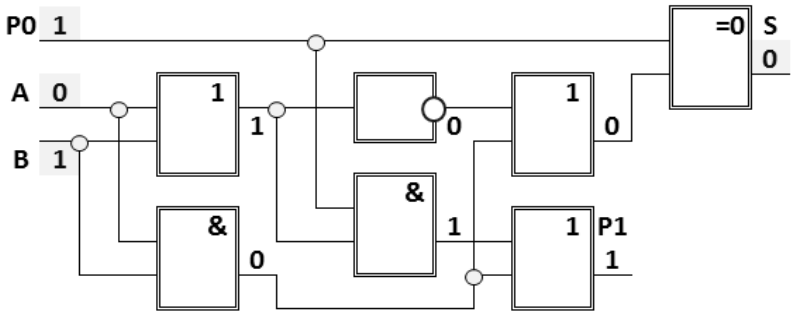


Рис. 2. Повний суматор на логічних елементах XAND

Логічні функції сигналів суми S і перенесення P_1 для схеми на рис. 2 у ДДНФ мають вигляд:

$$S = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc$$

$$P_1 = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc .$$

Мінімальна форма:

$$S = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc$$

$$P_1 = ab + ac + bc$$

або

$$S = c(\bar{a}\bar{b} + ab) + \bar{c}(a\bar{b} + \bar{a}b)$$

$$P_1 = c(b + a) + ab .$$

Таблиця 2

Таблиця істинності повного суматора

Вхід			Вихід	
A	B	P0	S	P1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Вихідні сигнали схеми на рис. 2 задовольняють таблицю істинності повного суматора (табл. 2).

Суматор на рис. 2 складається з десяти логічних елементів, час обчислення перенесення P_1 – 3dt, глибина схеми – шість елементів.

2. Коди Ґалуа

Коди Ґалуа отримують за допомогою рекурсії, де наступний елемент поля у кільці $GF\left(\begin{smallmatrix} m \\ n \end{smallmatrix}\right)$ подається за допомогою логічного виразу відповідної операції над попередніми елементами. Кожна з 2^{n-1} n-розрядна ненульова кодова комбінація послідовності Ґалуа є результатом циклічного зсуву вихідного ненульового кодового фрагмента з ключем $x_j = x_{i-n} \oplus x_{i-1}$, вони мають однакову вагу, що характеризує їх як еквідистантні або симплексні.

Наприклад, у полі Ґалуа $GF\left(\begin{smallmatrix} 4 \\ 2 \end{smallmatrix}\right)$ із породжуючим вектором *10011* і ключем $x_j = x_{i-4} \oplus x_{i-1}$ рекурсивна послідовність кодових елементів матиме вигляд:

$$1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0 \qquad (1)$$

Послідовність (1) подає чотирирозрядні коди Ґалуа, зсунуті один відносно одного на один біт, які мають значення: $1111_G = 0_{10}$, $1110_G = 1_{10}$, $1101_G = 2_{10}$, $1010_G = 3_{10}$, $0101_G = 4_{10}$ і т. д. [4]. Складена у такий спосіб система кодів повинна мати властивості кільця, що дає, зокрема, утворення початкового коду (1111) в разі операції зсуву останнього коду (0111) цієї системи на один біт.

Для заданої початкової послідовності, наприклад, x_1, x_2, x_3, x_4 з ключем $x_j = x_{i-4} \oplus x_{i-1}$ усі інші значення бітів послідовності (1) можна отримати через початкові:

$$x_5 = x_1 \oplus x_4 \qquad x_6 = x_2 \oplus x_5 = x_1 \oplus x_2 \oplus x_4$$
$$x_7 = x_3 \oplus x_6 = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \qquad x_8 = x_4 \oplus x_7 = x_1 \oplus x_2 \oplus x_3$$
$$x_9 = x_5 \oplus x_8 = x_2 \oplus x_3 \oplus x_4 \qquad x_{10} = x_6 \oplus x_9 = x_1 \oplus x_3$$
$$x_{11} = x_7 \oplus x_{10} = x_2 \oplus x_4 \qquad x_{12} = x_8 \oplus x_{11} = x_1 \oplus x_3 \oplus x_4 \qquad (2)$$
$$x_{13} = x_9 \oplus x_{12} = x_1 \oplus x_2 \qquad x_{14} = x_{10} \oplus x_{13} = x_2 \oplus x_3$$
$$x_{15} = x_{11} \oplus x_{14} = x_3 \oplus x_4 \qquad x_{16} = x_{12} \oplus x_{15} = x_1 \oplus x_3 \oplus x_4 \oplus x_3 \oplus x_4 = x_1.$$
$$x_{17} = x_{13} \oplus x_{16} = x_1 \oplus x_2 \oplus x_1 = x_2. \qquad x_{18} = x_{14} \oplus x_{17} = x_2 \oplus x_3 \oplus x_2 = x_3.$$
$$x_{19} = x_{15} \oplus x_{18} = x_3 \oplus x_4 \oplus x_3 = x_4. \qquad x_{20} = x_{16} \oplus x_{19} = x_1 \oplus x_4.$$

Під час виконання процедури додавання двох кодів *A(x)* і *D(x)* над кодом *A(x)* здійснюються дії, визначені залежностями (2) коду *D(x)*. Залежності коду *D(x)* задають своєрідну програмну процедуру (вектор) над кодом *A(x)* для обчислення розрядів суми *C(x)* (див. п. 4).

3. Кодування чисел за допомогою логічної операції XAND

Операція XAND дає змогу будувати коди, починаючи з нульового початкового фрагмента рекурентної послідовності. Отже, кожна з $2^n - 1$ n-розрядна кодова комбінація послідовності буде результатом циклічного зсуву, починаючи з вихідного нульового кодового фрагмента з ключем $x_j = x_{i-n} \boxplus x_{i-1}$.

Наприклад, для нульового вихідного фрагмента – 0000 і ключа $x_j = x_{i-4} \boxplus x_{i-1}$ рекурсивна послідовність кодових елементів матиме вигляд:

$$0\,0\,0\,0\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,1 \qquad (3)$$

Послідовність (3) подає чотирирозрядні коди, зсунуті один відносно одного на один біт, які мають значення: $0000_{XAND} = 0_{10}$, $0001_{XAND} = 1_{10}$, $0010_{XAND} = 2_{10}$, $0101_{XAND} = 3_{10}$, $1010_{XAND} = 4_{10}$ і т. д. Складена у такий спосіб система кодів повинна володіти властивостями кільця, що дає, зокрема, утворення початкового коду (0000) за операції зсуву останнього коду (1000) системи на один біт.

Аналогічно послідовності Ґалуа (1), для заданої початкової послідовності, наприклад, x_1, x_2, x_3, x_4 з ключем $x_j = x_{i-4} \boxplus x_{i-1}$ усі інші значення бітів послідовності (3) можна отримати через початкові:

$$x_5 = x_1 \boxplus x_4 \qquad x_6 = x_2 \boxplus x_5 = x_1 \boxplus x_2 \boxplus x_4$$
$$x_7 = x_3 \boxplus x_6 = x_1 \boxplus x_2 \boxplus x_3 \boxplus x_4 \qquad x_8 = x_4 \boxplus x_7 = x_1 \boxplus x_2 \boxplus x_3 \boxplus 1$$

Поява одиниці у виразі для x_8 зумовлена операцією $\boxplus$ над змінними x_4.

$$x_4 \boxplus x_4 = 1$$

Твердження 1. Операція $\boxplus$ одиниці до попередньої змінної не змінює значення попередньої змінної.

Для прикладу розглянемо дві операції $\boxplus$ одиниці:

$$0 \boxplus 1 = 0 \qquad (4)$$
$$1 \boxplus 1 = 1 \qquad (5)$$

Вирази (4)–(5) демонструють, що операція $\boxplus$ одиниці до попередньої змінної не змінює значення попередньої змінної, тому у виразі для x_8 та в інших подібних виразах одиницю можна опустити. У підсумку будуть отримані аналогічні залежності (2) для послідовності (3) за винятком логічної операції. Операція $\oplus$ буде замінена на операцію $\boxplus$.

Аналогічно кодам Ґалуа під час виконання процедури додавання двох кодів $A(x)_{XAND}$ і $D(x)_{XAND}$ над кодом $A(x)_{XAND}$ здійснюються дії, визначені залежностями (6) для коду $D(x)_{XAND}$. Залежності коду $D(x)_{XAND}$ задають своєрідну програмну процедуру (вектор) над кодом $A(x)_{XAND}$ для обчислення розрядів суми $C(x)_{XAND}$ (див. п. 4).

Враховуючи вирази для бітів (6), послідовність (3) перепишемо так:

$$x_1, x_2, x_3, x_4, x_1 \boxplus x_4, x_1 \boxplus x_2 \boxplus x_4, \ldots x_1 \boxplus x_2, x_2 \boxplus x_3, x_3 \boxplus x_4, x_{16}, x_{17}, x_{18}, x_{19} \qquad (7)$$

За допомогою зсуву на один біт на послідовності (7) можна отримати 2^n чотирирозрядні коди. Однак коди (x_1, x_2, x_3, x_4), (x_{16}, x_{17}, x_{18}, x_{19}) матимуть однакові значення бітів, тому різних кодів залишається $2^n - 1$. Застосування однакових кодів з різних позицій рекурентної послідовності (7) для виконання арифметичних операцій з багаторозрядними числами можливе за умови використання процедури перенесення.

4. Арифметична операція додавання кодів, утворених за допомогою операції XAND

Операція додавання кодів $A(x)$ і $D(x)$ полягає у рекурсивному зсуві на послідовності (3), починаючи з вихідної позиції коду $A(x)$, на кількість дискретних позицій, визначених десятковим еквівалентом коду $D(x)$. Отже, реалізація вказаної операції додавання зводиться до одночасного паралельного взаємно незалежного формування кожного біта результату обчислення як суми за операцією $\boxplus$ без необхідності виконання операцій міжрозрядних перенесень. Зазначена послідовність дій призводить до збільшення продуктивності обчислень пропорційно до розрядності слова даних порівняно з двійковою системою числення. Обчислення кожного результату операції здійснюється за один такт. Процес обчислення інваріантний розрядності слова даних.

Для виконання процедури додавання коди-доданки необхідно подати за допомогою залежностей (6).

Приклад 1. Залежності для коду $A(x)$ – 1010_{XAND} (4_{10}), коду $D(x)$ – 1001_{XAND} (6_{10}), коду суми $C(x)$ – 1011_{XAND} (10_{10}) подані на рис. 3.

$A(x) - 4_{10}$			$D(x) - 6_{10}$			$C(x) - 10_{10}$		
x_5	=	$x_1 \boxplus x_4$	x_7	=	$x_1 \boxplus x_2 \boxplus x_3 \boxplus x_4$	x_{11}	=	$x_2 \boxplus x_4$
x_6	=	$x_1 \boxplus x_2 \boxplus x_4$	x_8	=	$x_1 \boxplus x_2 \boxplus x_3$	x_{12}	=	$x_1 \boxplus x_3 \boxplus x_4$
x_7	=	$x_1 \boxplus x_2 \boxplus x_3 \boxplus x_4$	x_9	=	$x_2 \boxplus x_3 \boxplus x_4$	x_{13}	=	$x_1 \boxplus x_2$
x_8	=	$x_1 \boxplus x_2 \boxplus x_3$	x_{10}	=	$x_1 \boxplus x_3$	x_{14}	=	$x_2 \boxplus x_3$
а			*б*			*в*		

Рис. 3. Вирази для розрядів кодів:
а – $A(x) - 1010_{XAND}$ (4_{10}); б – $D(x) - 1001_{XAND}$ (6_{10});
в – $C(x) - 1011_{XAND}$ (10_{10}), подані залежностями (6)

Під час виконання процедури додавання над кодом $A(x)$ (рис. 3, *а*) здійснюються дії, визначені залежностями (координатами) коду $D(x)$ на рекурсивній послідовності (3), які відповідають значенню коду $D(x)$ (рис. 3, *б*). Залежності коду $D(x)$ (рис. 3, *б*) задають своєрідну програмну процедуру (вектор) над кодом $A(x)$ для обчислення кожного розряду суми $C(x)$ (рис. 3, *в*).

Зазначена програмна процедура (вектор) подана на рис. 4.

	1-й розряд	*2-й розряд*	*3-й розряд*	*4-й розряд*
$D'(x) =>$	$x_1 \boxplus x_2 \boxplus x_3 \boxplus x_4$	$x_1 \boxplus x_2 \boxplus x_3$	$x_2 \boxplus x_3 \boxplus x_4$	$x_1 \boxplus x_3$

Рис. 4. Програмна процедура (вектор) D'(x) над кодом A(x)

Згідно з програмою $D'(x)$ в обчисленні першого розряду суми $C(x)$ повинні взяти участь всі чотири залежності коду $A(x)$ (див. рис. 4), для обчислення суми $C(x)$ у другому розряді повинні взяти участь перші три залежності коду $A(x)$; для обчислення суми $C(x)$ у третьому розряді повинні взяти участь залежності другого, третього та четвертого розрядів коду $A(x)$; для обчислення суми $C(x)$ у четвертому розряді повинні взяти участь залежності першого та третього розрядів коду $A(x)$ (див. рис. 4).

Зазначену програмну процедуру $D'(x)$ над кодом $A(x)$ можна подати таблицею (табл. 3). У першому рядку табл. 3 записані залежності коду $A(x)$ (див. рис. 3, *а*) для першого розряду суми $C(x)$ згідно з програмною процедурою $D'(x)$ (рис. 4) для першого розряду. У другому рядку табл. 3 записані залежності коду $A(x)$ (див. рис. 3, *а*) для другого розряду суми $C(x)$ згідно з програмною процедурою $D'(x)$ (рис. 4) для другого розряду і т. д.

Таблиця 3

Обчислення над кодом $A(x)$, що визначені програмною процедурою $D'(x)$

		$x_5(A)$		$x_6(A)$		$x_7(A)$		$x_8(A)$		Сума кодів
x_7	=	$x_1 \boxplus x_4$	$\boxplus$	$x_1 \boxplus x_2 \boxplus x_4$	$\boxplus$	$x_1 \boxplus x_2 \boxplus x_3 \boxplus x_4$	$\boxplus$	$x_1 \boxplus x_2 \boxplus x_3$	=	$x_2 \boxplus x_4$
x_8	=	$x_1 \boxplus x_4$	$\boxplus$	$x_1 \boxplus x_2 \boxplus x_4$	$\boxplus$	$x_1 \boxplus x_2 \boxplus x_3 \boxplus x_4$			=	$x_1 \boxplus x_3 \boxplus x_4$
x_9	=			$x_1 \boxplus x_2 \boxplus x_4$	$\boxplus$	$x_1 \boxplus x_2 \boxplus x_3 \boxplus x_4$	$\boxplus$	$x_1 \boxplus x_2 \boxplus x_3$	=	$x_1 \boxplus x_2$
x_{10}	=	$x_1 \boxplus x_4$			$\boxplus$	$x_1 \boxplus x_2 \boxplus x_3 \boxplus x_4$			=	$x_2 \boxplus x_3$

Утворені одиниці під час обчислення коду суми $C(x)$ опускають (див. п. 3).

Індекси при x у крайній лівій колонці табл. 3 необхідно замінити відповідними індексами для суми $C(x)$ – $x_{11} = x_2 \boxplus x_4$, $x_{12} = x_1 \boxplus x_3 \boxplus x_4$, $x_{13} = x_1 \boxplus x_2$, $x_{14} = x_2 \boxplus x_3$.

Приклад 2. Залежності для коду $A(x)$ – 0001_{XAND} (1_{10}), коду $D(x)$ – 1100_{XAND} (13_{10}), коду суми $C(x)$ – 1000_{XAND} (14_{10}) подані на рис. 5.

$A(x) - 1_{10}$			$D(x) - 13_{10}$			$C(x) - 14_{10}$		
x_2	=	x_2	x_{14}	=	$x_2 \boxplus x_3$	x_{15}	=	$x_3 \boxplus x_4$
x_3	=	x_3	x_{15}	=	$x_3 \boxplus x_4$	x_{16}	=	x_1
x_4	=	x_4	x_{16}	=	x_1	x_{17}	=	x_2
x_5	=	$x_1 \boxplus x_4$	x_{17}	=	x_2	x_{18}	=	x_3
	а			*б*			*в*	

Рис. 5. Вирази для розрядів кодів:
а – $A(x) - 0001_{XAND}$ (1_{10}); б – $D(x) - 1100_{XAND}$ (13_{10});
в – $C(x) - 1000_{XAND}$ (14_{10}), подані залежностями (6)

Програмна процедура над кодом $A(x)$ – 0001 подана на рис. 6

	1-й розряд	*2-й розряд*	*3-й розряд*	*4-й розряд*
$D'(x)$ =>	$x_2 \boxplus x_3$	$x_2 \boxplus x_4$	x_1	x_2

Рис. 6. Програмна процедура D'(x) над кодом A(x) – 0001

Таблиця 4

Обчислення над кодом *A(x) – 0001*, визначені програмною процедурою *D'(x)* (рис. 6)

		$x_2(A)$		$x_3(A)$		$x_4(A)$		$x_5(A)$		Сума кодів
x_{14}	=			x_3	$\boxplus$	x_4			=	$x_3 \boxplus x_4$
x_{15}	=					x_4	$\boxplus$	$x_1 \boxplus x_4$	=	x_1
x_{16}	=	x_2							=	x_2
x_{17}	=			x_3					=	x_3

Утворені одиниці під час обчислення коду суми $C(x)$ опускають (див. п. 3). Індекси при x у крайній лівій колонці табл. 4 необхідно замінити відповідними індексами для суми $C(x)$ – $x_{15} = x_3 \boxplus x_4$, $x_{16} = x_1$, $x_{17} = x_2$, $x_{18} = x_3$.

Розглянута арифметична процедура додавання кодів, утворених за допомогою операції XAND, зберігається для всіх інших варіантів додавання. Для всіх можливих варіантів додавання чотирирозрядних кодів дані для коду $D(x)$ подано у табл. 5. Програмна процедура $D'(x)$ над будь-яким кодом $A(x)$, наприклад, для коду $D(x)$ – 0000 буде мати вигляд:

	1-й розряд	*2-й розряд*	*3-й розряд*	*4-й розряд*
$D'(x)$ =>	x_1	x_2	x_3	x_4

Таблиця 5

Програмна процедура *D'(x)* над кодом *A(x)*

№ з/п	Код XAND	Розряди кодів, утворених за допомогою операції XAND, виражені через перші змінні x_1,x_2,x_3, x_4 послідовності (3)			
	D(x)	*D'(x)*			
0	0000	x_1	x_2	x_3	x_4
1	0001	x_2	x_3	x_4	$x_1 \boxplus x_4$
2	0010	x_3	x_4	$x_1 \boxplus x_4$	$x_1 \boxplus x_2 \boxplus x_4$
3	0101	x_4	$x_1 \boxplus x_4$	$x_1 \boxplus x_2 \boxplus x_4$	$x_1 \boxplus x_2 \boxplus x_3 \boxplus x_4$
4	1010	$x_1 \boxplus x_4$	$x_1 \boxplus x_2 \boxplus x_4$	$x_1 \boxplus x_2 \boxplus x_3 \boxplus x_4$	$x_1 \boxplus x_2 \boxplus x_3$
5	0100	$x_1 \boxplus x_2 \boxplus x_4$	$x_1 \boxplus x_2 \boxplus x_3 \boxplus x_4$	$x_1 \boxplus x_2 \boxplus x_3$	$x_2 \boxplus x_3 \boxplus x_4$
6	1001	$x_1 \boxplus x_2 \boxplus x_3 \boxplus x_4$	$x_1 \boxplus x_2 \boxplus x_3$	$x_2 \boxplus x_3 \boxplus x_4$	$x_1 \boxplus x_3$
7	0011	$x_1 \boxplus x_2 \boxplus x_3$	$x_2 \boxplus x_3 \boxplus x_4$	$x_1 \boxplus x_3$	$x_2 \boxplus x_4$
8	0110	$x_2 \boxplus x_3 \boxplus x_4$	$x_1 \boxplus x_3$	$x_2 \boxplus x_4$	$x_1 \boxplus x_3 \boxplus x_4$
9	1101	$x_1 \boxplus x_3$	$x_2 \boxplus x_4$	$x_1 \boxplus x_3 \boxplus x_4$	$x_1 \boxplus x_2$
10	1011	$x_2 \boxplus x_4$	$x_1 \boxplus x_3 \boxplus x_4$	$x_1 \boxplus x_2$	$x_2 \boxplus x_3$
11	0111	$x_1 \boxplus x_3 \boxplus x_4$	$x_1 \boxplus x_2$	$x_2 \boxplus x_3$	$x_3 \boxplus x_4$
12	1110	$x_1 \boxplus x_2$	$x_2 \boxplus x_3$	$x_3 \boxplus x_4$	x_1
13	1100	$x_2 \boxplus x_3$	$x_3 \boxplus x_4$	x_1	x_2
14	1000	$x_3 \boxplus x_4$	x_1	x_2	x_3

Оскільки арифметичні операції в електронних схемах виконують над фізичними сигналами, подамо коди, утворені за допомогою операції XAND – *A(x), D(x), C(x),* у бітах у бітовому представленні для кодів, утворених за допомогою операції XAND. Наявність у коді аргумента послідовності (3) позначається нулем, відсутність аргумента – одиницею. Наприклад, чотирирозрядний код $x_1 \boxplus x_4$ у бітовому представленні матиме вигляд – $0 \boxplus 1 \boxplus 1 \boxplus 0$, чотирирозрядний код $x_1 \boxplus x_2 \boxplus x_3 \boxplus x_4$ у бітовому представленні буде таким – $0 \boxplus 0 \boxplus 0 \boxplus 0$.

Розряди кодів *A(x) – 1010, D(x) – 1001, C(x) – 1011* прикладу 1 (рис. 3) у бітовому представленні матимуть вигляд

		A(x) – 4_{10}			*D(x)* – 6_{10}			*C(x)* – 10_{10}
x_5	=	$0 \boxplus 1 \boxplus 1 \boxplus 0$	x_7	=	$0 \boxplus 0 \boxplus 0 \boxplus 0$	x_{11}	=	$1 \boxplus 0 \boxplus 1 \boxplus 0$
x_6	=	$0 \boxplus 0 \boxplus 1 \boxplus 0$	x_8	=	$0 \boxplus 0 \boxplus 0 \boxplus 1$	x_{12}	=	$0 \boxplus 1 \boxplus 0 \boxplus 0$
x_7	=	$0 \boxplus 0 \boxplus 0 \boxplus 0$	x_9	=	$1 \boxplus 0 \boxplus 0 \boxplus 0$	x_{13}	=	$0 \boxplus 0 \boxplus 1 \boxplus 1$
x_8	=	$0 \boxplus 0 \boxplus 0 \boxplus 1$	x_{10}	=	$0 \boxplus 1 \boxplus 0 \boxplus 1$	x_{14}	=	$1 \boxplus 0 \boxplus 0 \boxplus 1$
		а			*б*			*в*

Рис. 7. Залежності кодів:
а – A(x); б – D(x); в – C(x) (рис. 3) у бітовому представленні

Над розрядами кодів *A(x), C(x)* (рис. 7) проведемо обчислення за операцією XAND. У підсумку отримаємо вирази для кодів *A(x), C(x)* у бітовому представленні (рис. 8). Код *D(x)* подає програмну процедуру *D'(x)* (вектор) над кодом *A(x)*.

	A(x) – 4_{10}			*D(x)* – 6_{10}			*C(x)* – 10_{10}
x_5	=	1	x_7	=	$0 \boxplus 0 \boxplus 0 \boxplus 0$	x_{11}	=	1
x_6	=	0	x_8	=	$0 \boxplus 0 \boxplus 0 \boxplus 1$	x_{12}	=	0
x_7	=	1	x_9	=	$1 \boxplus 0 \boxplus 0 \boxplus 0$	x_{13}	=	1
x_8	=	0	x_{10}	=	$0 \boxplus 1 \boxplus 0 \boxplus 1$	x_{14}	=	1
	а			*б*			*в*	

Рис. 8. Коди: а – A(x); б – D(x); в – C(x) у бітовому представленні

Програмна процедура *D'(x)* (рис. 9) над кодом *A(x)* (рис. 8) матиме вигляд:

	1-й розряд	*2-й розряд*	*3-й розряд*	*4-й розряд*
D'(x) =>	0⊞0⊞0⊞0	0⊞0⊞0⊞1	1⊞0⊞0⊞0	0⊞1⊞0⊞1

Рис. 9. Програмна процедура D'(x) над кодом A(x)

Програмну процедуру *D'(x)* (рис. 9) над кодом *A(x)* можна подати таблицею (табл. 6). У клітинках табл. 6 записані значення бітів, відповідних розрядам коду *A(x)*.

Таблиця 6

Обчислення над кодом *A(x) – 1010*, визначені програмною процедурою *D'(x)* (рис. 9)

		$x_5(A)$		$x_6(A)$		$x_7(A)$		$x_8(A)$		Сума кодів *C(x)*
x_7	=	1	⊞	0	⊞	1	⊞	0	=	1
x_8	=	1	⊞	0	⊞	1			=	0
x_9	=			0	⊞	1	⊞	0	=	1
x_{10}	=	1			⊞	1			=	1

Незаповнені клітинки табл. 6 відображають відсутність змінних у залежностях (6). Відсутні змінні у залежностях (6) для операції XAND за бітового представлення коду необхідно заповнити одиницями (табл. 7).

Таблиця 7

Обчислення над кодом *A(x) – 1010*, визначені програмною процедурою *D'(x)* (рис. 9) із заповненими клітинками табл. 6

		$x_5(A)$		$x_6(A)$		$x_7(A)$		$x_8(A)$		Сума кодів *C(x)*
x_7	=	1	⊞	0	⊞	1	⊞	0	=	1
x_8	=	1	⊞	0	⊞	1	⊞	1	=	0
x_9	=	1	⊞	0	⊞	1	⊞	0	=	1
x_{10}	=	1	⊞	1	⊞	1	⊞	1	=	1

Обчислення у табл. 7 запишемо за допомогою рівнянь (14).

$$\begin{aligned} x_7 &= 1_A \boxplus 0_A \boxplus 1_A \boxplus 0_A = \mathbf{1} \\ x_8 &= 1_A \boxplus 0_A \boxplus 1_A \boxplus 1_A = \mathbf{0} \\ x_9 &= 1_A \boxplus 0_A \boxplus 1_A \boxplus 0_A = \mathbf{1} \\ x_{10} &= 1_A \boxplus 1_A \boxplus 1_A \boxplus 1_A = \mathbf{1} \end{aligned} \qquad (14)$$

Отримані значення рівнянь $x_7=1$, $x_8=0$, $x_9=1$, $x_{10}=1$ відповідають розрядам коду суми – *C(x)*. Індекси при *x* необхідно замінити відповідними індексами для суми $C(x) – x_{11}=1, x_{12}=0, x_{13}=1, x_{14}=1$.

В обчисленнях (14) програмна процедура *D'(x)* над кодом *A(x)* існує неявно. Тому для іншого варіанта додавання кодів необхідно встановити нову програмну процедуру *D'(x)* над кодом *A(x)* і повторити розглянутий порядок додавання кодів.

Для автоматизації обчислень на рівні електронної схеми необхідно встановити явний зв'язок між бітами вектора *D'(x)* і бітами коду *A(x)*. Для операції XAND такий зв'язок встановлюється за допомогою логічної функції АБО. У разі додавання багаторозрядних кодів можливі чотири варіанти взаємодії:

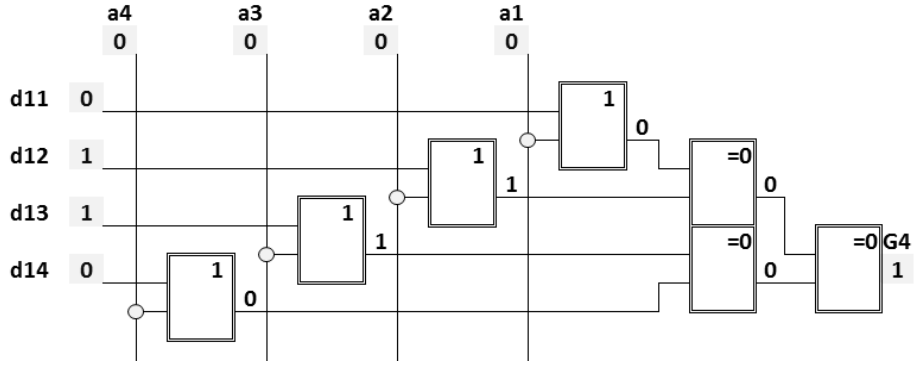
$$0_D \vee 0_A,$$
$$0_D \vee 1_A,$$
$$1_D \vee 0_A,$$
$$1_D \vee 1_A.$$

Тоді рівняння (14) подають рівняннями (15), у яких код А записується вихідними значеннями бітів – 1010:

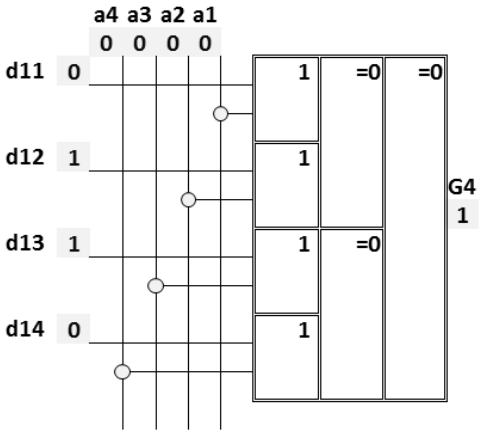
$$
\begin{aligned}
x_7 &= (0_D \vee 1_A) \boxplus (0_D \vee 0_A) \boxplus (0_D \vee 1_A) \boxplus (0_D \vee 0_A) = \mathbf{1} \\
x_8 &= (0_D \vee 1_A) \boxplus (0_D \vee 0_A) \boxplus (0_D \vee 1_A) \boxplus (1_D \vee 0_A) = \mathbf{0} \\
x_9 &= (1_D \vee 1_A) \boxplus (0_D \vee 0_A) \boxplus (0_D \vee 1_A) \boxplus (0_D \vee 0_A) = \mathbf{1} \\
x_{10} &= (0_D \vee 1A) \boxplus (1_D \vee 0_A) \boxplus (0_D \vee 1_A) \boxplus (1_D \vee 0_A) = \mathbf{1}
\end{aligned}
\qquad (15)
$$

Отримані значення рівнянь $x_7=1$, $x_8=0$, $x_9=1$, $x_{10}=1$ (15) відповідають розрядам коду суми – $C(x)$. Індекси при x необхідно замінити відповідними індексами для суми $C(x) – x_{11}=1, x_{12}=0, x_{13}=1, x_{14}=1$.

За рівняннями (15) синтезується комбінаційна схема суматора на логічних елементах XAND. Для першого розряду коду суми $C(x)$ x_{11} така схема матиме вигляд, як на рис. 10.



а



б

Рис. 10. Комбінаційна схема суматора для одного розряду суми C(x) на логічних елементах XAND (а); схема суматора для одного розряду суми, подана складною логікою (б)

Суматор на рис. 10 складається з 16 логічних елементів, глибина схеми – сім елементів. Аналогічні схеми будують і для інших розрядів коду суми $C(x) – x_{12}, x_{13}, x_{14}$.

Для всіх можливих варіантів додавання чотирирозрядного коду дані для коду $D(x)$ у бітовому представленні наведені у табл. 8. Програмна процедура $D'(x)$ над будь-яким кодом $A(x)$ у бітовому представленні, наприклад, для коду $D(x)$ – 0000 матиме вигляд:

	1-й розряд	*2-й розряд*	*3-й розряд*	*4-й розряд*
D'(x) =>	*0⊞1⊞1⊞1*	*1⊞0⊞1⊞1*	*1⊞1⊞0⊞1*	*1⊞1⊞1⊞0*

Таблиця 8

Програмна процедура *D'(x)* над кодом *A(x)* у бітовому поданні

№ з/п	Код XAND	Розряди кодів XAND, виражені через перші змінні (6) x_1, x_2, x_3, x_4 послідовності (3) у бітовому представленні			
		$d_{11}\, d_{12}\, d_{13}\, d_{14}$	$d_{21}\, d_{22}\, d_{23}\, d_{24}$	$d_{31}\, d_{32}\, d_{33}\, d_{34}$	$d_{41}\, d_{42}\, d_{43}\, d_{44}$
	D(x)	*D'(x)*			
0	0000	0⊞1⊞1⊞1	1⊞0⊞1⊞1	1⊞1⊞0⊞1	1⊞1⊞1⊞0
1	0001	1⊞0⊞1⊞1	1⊞1⊞0⊞1	1⊞1⊞1⊞0	0⊞1⊞1⊞0
2	0010	1⊞1⊞0⊞1	1⊞1⊞1⊞0	0⊞1⊞1⊞0	0⊞0⊞1⊞0
3	0101	1⊞1⊞1⊞0	0⊞1⊞1⊞0	0⊞0⊞1⊞0	0⊞0⊞0⊞0
4	1010	0⊞1⊞1⊞0	0⊞0⊞1⊞0	0⊞0⊞0⊞0	0⊞0⊞0⊞1
5	0100	0⊞0⊞1⊞0	0⊞0⊞0⊞0	0⊞0⊞0⊞1	1⊞0⊞0⊞0
6	1001	0⊞0⊞0⊞0	0⊞0⊞0⊞1	1⊞0⊞0⊞0	0⊞1⊞0⊞1
7	0011	0⊞0⊞0⊞1	1⊞0⊞0⊞0	0⊞1⊞0⊞1	1⊞0⊞1⊞0
8	0110	1⊞0⊞0⊞0	0⊞1⊞0⊞1	1⊞0⊞1⊞0	0⊞1⊞0⊞0
9	1101	0⊞1⊞0⊞1	1⊞0⊞1⊞0	0⊞1⊞0⊞0	0⊞0⊞1⊞1
10	1011	1⊞0⊞1⊞0	0⊞1⊞0⊞0	0⊞0⊞1⊞1	1⊞0⊞0⊞1
11	0111	0⊞1⊞0⊞0	0⊞0⊞1⊞1	1⊞0⊞0⊞1	1⊞1⊞0⊞0
12	1110	0⊞0⊞1⊞1	1⊞0⊞0⊞1	1⊞1⊞0⊞0	0⊞1⊞1⊞1
13	1100	1⊞0⊞0⊞1	1⊞1⊞0⊞0	0⊞1⊞1⊞1	1⊞0⊞1⊞1
14	1000	1⊞1⊞0⊞0	0⊞1⊞1⊞1	1⊞0⊞1⊞1	1⊞1⊞0⊞1

У скороченому записі вектора *D'(x)* знак операції опускають, наприклад:

	D(x)	*D'(x)*			
14	1000	1100	0111	1011	1101

5. Таблиця істинності паралельного суматора без перенесення на логічних елементах XAND

Діапазон додавання чисел паралельного суматора без перенесення на логічних елементах XAND становить:

$$x_{\mathrm{D}} + y_{\mathrm{D}} =< 2^k - 2,$$

де k – розрядність числа. Кількість варіантів додавання багаторозрядного паралельного суматора без перенесення на логічних елементах XAND становить:

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

або

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

де k – розрядність числа [10].

Обчисливши значення b, визначаємо кількість рядків таблиці істинності суматора. Таблиця істинності чотирирозрядного паралельного суматора без перенесення на логічних елементах XAND буде вміщувати 120 рядків (табл. 9).

Таблиця 9

Таблиця істинності чотирирозрядного паралельного суматора без перенесення на логічних елементах XAND

Коди XAND								Коефіцієнти dij, що відповідають коду DXAND																CXAND			
AXAND				DXAND				d11	d12	d13	d14	d21	d22	d23	d24	d32	d32	d33	d34	d41	d42	d43	d44				
a_1	a_2	a_3	a_4	d_1	d_2	d_3	d_4																	s_1	s_2	s_3	s_4
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**
0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0	0	0	0	1
0	0	0	0	0	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	0	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1	0
0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0
0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1
0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	1	1	0	1	0	0	0	1	1
0	0	0	0	0	1	1	0	1	0	0	0	0	1	0	1	1	0	1	0	0	1	0	0	0	1	1	0
0	0	0	0	1	1	0	1	0	1	0	1	1	0	1	0	0	1	0	0	0	0	1	1	1	1	0	1
0	0	0	0	1	0	1	1	1	0	1	0	0	1	0	0	0	0	1	1	1	0	0	1	1	0	1	1
0	0	0	0	0	1	1	1	0	1	0	0	0	0	1	1	1	0	0	1	1	1	0	0	0	1	1	1
0	0	0	0	1	1	1	0	0	0	1	1	1	0	0	1	1	1	0	0	0	1	1	1	1	1	1	0
0	0	0	0	1	1	0	0	1	0	0	1	1	1	0	0	0	1	1	1	1	0	1	1	1	1	0	0
0	0	0	0	1	0	0	0	1	1	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	0	0	0
0	0	0	1	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	0	0	0	1
0	0	0	1	0	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0
0	0	0	1	0	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	1	0	1
0	0	0	1	0	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0	0	1	0	1	0
0	0	0	1	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0
0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1
0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	0	1	1
0	0	0	1	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	1	1	0	1	0	0	1	1	0
0	0	0	1	0	1	1	0	1	0	0	0	0	1	0	1	1	0	1	0	0	1	0	0	1	1	0	1
0	0	0	1	1	1	0	1	0	1	0	1	1	0	1	0	0	1	0	0	0	0	1	1	1	0	1	1
0	0	0	1	1	0	1	1	1	0	1	0	0	1	0	0	0	0	1	1	1	0	0	1	0	1	1	1
0	0	0	1	0	1	1	1	0	1	0	0	0	0	1	1	1	0	0	1	1	0	0	0	1	1	1	0
0	0	0	1	1	1	1	0	0	0	1	1	1	0	0	1	1	1	0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	0	0	1	0	0	1	1	1	0	0	0	1	1	1	1	0	1	1	1	0	0	0
0	0	1	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	0	0	1	0
0	0	1	0	0	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0	0	1	0	1
0	0	1	0	0	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	1	0	1	0
0	0	1	0	0	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	1	0	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1
0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	1	1	0
0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	1	1	0	1	0	1	1	0	1
0	0	1	0	0	1	1	0	1	0	0	0	0	1	0	1	1	0	1	0	0	1	0	0	1	0	1	1
0	0	1	0	1	1	0	1	0	1	0	1	1	0	1	0	0	1	0	0	0	0	1	1	0	1	1	1
0	0	1	0	1	0	1	1	1	0	1	0	0	1	0	0	0	0	1	1	1	0	0	1	1	1	1	0
0	0	1	0	0	1	1	1	0	1	0	0	0	0	1	1	1	0	0	1	1	1	0	0	1	1	0	0
0	0	1	0	1	1	1	0	0	0	1	1	1	0	0	1	1	1	0	0	0	1	1	1	1	0	0	0
0	1	0	1	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	0	1
0	1	0	1	0	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0	1	0	1	0
0	1	0	1	0	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	1	0	0
0	1	0	1	0	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1
0	1	0	1	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1
0	1	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0
0	1	0	1	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	1	0	1
0	1	0	1	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	1	1	0	1	0	1	0	1	1
0	1	0	1	0	1	1	0	1	0	0	0	0	1	0	1	1	0	1	0	0	1	0	0	0	1	1	1
0	1	0	1	1	1	0	1	0	1	0	1	1	0	1	0	0	1	0	0	0	0	1	1	1	1	1	0
0	1	0	1	1	0	1	1	1	0	1	0	0	1	0	0	0	0	1	1	1	0	0	1	1	1	0	0
0	1	0	1	0	1	1	1	0	1	0	0	0	0	1	1	1	0	0	1	1	1	0	0	1	0	0	0
1	0	1	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	0	1	0
1	0	1	0	0	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0	0	1	0	0
1	0	1	0	0	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	1	0	0	1
1	0	1	0	0	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	1	1
1	0	1	0	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	1	0
1	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	1
1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	1	1
1	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	1	1	0	1	0	0	1	1	1
1	0	1	0	0	1	1	0	1	0	0	0	0	1	0	1	1	0	1	0	0	1	0	0	1	1	1	0

Продовження табл. 9

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	0	1	0	1	1	0	1	0	1	0	1	1	0	1	0	0	1	0	0	0	0	1	1	1	1	0	0
1	0	1	0	1	0	1	1	1	0	1	0	0	1	0	0	0	0	1	1	1	0	0	1	1	0	0	0
0	1	0	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	0	0
0	1	0	0	0	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0	1	0	0	1
0	1	0	0	0	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	0	1	1
0	1	0	0	0	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	1	1	0
0	1	0	0	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	0	1
0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	1
0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	1	1	1
0	1	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	1	1	0	1	0	1	1	1	0
0	1	0	0	0	1	1	0	1	0	0	0	0	1	0	1	1	0	1	0	0	1	0	0	1	1	0	0
0	1	0	0	1	1	0	1	0	1	0	1	1	0	1	0	0	1	0	0	0	0	1	1	1	0	0	0
1	0	0	1	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	0	0	1
1	0	0	1	0	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0	0	0	1	1
1	0	0	1	0	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	1	1	0
1	0	0	1	0	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0	0	1	1	0	1
1	0	0	1	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1	1
1	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1
1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	1	1	0
1	0	0	1	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	1	1	0	1	0	1	1	0	0
1	0	0	1	0	1	1	0	1	0	0	0	0	1	0	1	1	0	1	0	0	1	0	0	1	0	0	0
0	0	1	1	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	0	0	1	1
0	0	1	1	0	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0	0	1	1	0
0	0	1	1	0	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	1	1	0	1
0	0	1	1	0	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0	0	1	0	1	1
0	0	1	1	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	1	1
0	0	1	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	1	0
0	0	1	1	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	1	0	0
0	0	1	1	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	1	1	0	1	0	1	0	0	0
0	1	1	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0
0	1	1	0	0	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0	1	1	0	1
0	1	1	0	0	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	1	0	1	1
0	1	1	0	0	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	1	1	1
0	1	1	0	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	0
0	1	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0
0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	0
1	1	0	1	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	0	1
1	1	0	1	0	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0	1	0	1	1
1	1	0	1	0	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	1	1	1
1	1	0	1	0	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0	0	1	1	1	0
1	1	0	1	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	0	0
1	1	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
1	0	1	1	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	0	1	1
1	0	1	1	0	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0	0	1	1	1
1	0	1	1	0	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	1	1	1	0
1	0	1	1	0	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0	0	1	1	0	0
1	0	1	1	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0
0	1	1	1	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	1
0	1	1	1	0	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0	1	1	1	0
0	1	1	1	0	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	1	1	0	0
0	1	1	1	0	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0	0	1	0	0	0
1	1	1	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	0
1	1	1	0	0	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0	1	1	0	0
1	1	1	0	0	0	1	0	1	1	0	1	1	1	1	0	0	1	1	0	0	0	1	0	1	0	0	0
1	1	0	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	0	0
1	1	0	0	0	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0	1	0	0	0
1	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	0	0	0

Висновки

Побудована схема чотирирозрядного паралельного суматора без перенесення на логічних елементах XAND. Принципи побудови чотирирозрядного суматора зберігаються для створення суматора більшої розрядності. Логіка схеми суматора перевірена його таблицею істинності. Для роботи розглянутого суматора необхідний пристрій для подавання на його входи сигналів проміжних коефіцієнтів d_{ji} логічного вектора для заданого коду. Одним із пристроїв, який зможе забезпечити функціональне з'єднання керуючого виходу дешифратора з рядком проміжних

коефіцієнтів d_{ji} логічного вектора для процедури додавання кодів, побудованих за допомогою операції XAND, є мультиплексор. Іншим рішенням може бути застосування пристрою пам'яті.

Діапазон додавання чисел паралельного суматора без перенесення на логічних елементах XAND становить:

$$x_D + y_D =< 2^k - 2,$$

де k – розрядність числа. Кількість варіантів додавання багаторозрядного паралельного суматора без перенесення на логічних елементах XAND дорівнює:

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

де k – розрядність числа.

Суматор на логічних елементах XAND дає змогу в процесі арифметичних операцій оперувати нульовим кодовим фрагментом.

1. Акушский И. Я. Машинная арифметика в остаточных классах [Текст] / И. Я. Акушский, Д. И. Юдицкий. – М.: Сов. радио, 1978. – 256. 2. Воробьев Н. Сумматоры: определения, классификация, уравнения, структуры и применение [Електронний ресурс]. – Режим доступу: http://www.chipinfo.ru/literature/chipnews/200002/37.html. – 10.04. 2015 р. – Загол. з екрана. 3. Круцкевич О. Д. Матричні системи числення [Текст] / О. Д. Круцкевич, Я. М. Николайчук // Вісник Хмельницького національного університету. – 2007. – № 3. – Т. 1. – С. 62–64. 4. Николайчук Я. М. Коди поля Галуа: теорія та застосування: монографія / Я. М. Николайчук – Тернопіль: ТзОВ Терно-Граф. – 2012. – 576 с. 5. Николайчук Я. М. Теорія джерел інформації: монографія / Ярослав Миколайович Николайчук. – 2-ге вид., виправлене – Тернопіль: ТзОВ Терно-Граф. – 2010. – 534 с. 6. Николайчук Я. М. Теоретичні засади та принципи побудови арифметико-логічного пристрою на основі вертикально-інформаційної технології [Текст] / Я. М. Николайчук О. М. Заставний, П. В. Гуменний // Вісник Хмельницького національного університету. – 2012. – № 2. – С. 190–196. 7. Николайчук Я. М. Теоретичні основи побудови та структура спецпроцесорів в базисі Крестенсона [Текст] / Я. М. Николайчук, О. І. Волинський, С. В. Кулина // Вісник Хмельницького національного університету. – Хмельницький. – 2007. – № 3. – Т. 1. – С. 85–90. 8. Николайчук Я. Н. Программные модели распараллеливания измерения, кодирования и передачи сообщений унитарным преобразованием СОК [Текст] / Я. Н. Николайчук, Б. М. Шевчук, А. А. Попов // Материалы VI Всесоюзной школы-семинара. – Львов, 1987. 9. Рабаи Жан М. Цифровые интегральные схемы. Методология проектирования / Жан М. Рабаи, Ананта Чандракасан, Николич Бориож; перев. с англ. – 2-е изд. – М.: ООО «И. Д. Вильямс», 2007. – 912 с. 10. Соломко М. Т. Оцінка часу переносу в суматорах з напівсуматором у першому розряді для ТЧБ Радемахера [Текст] / М. Т. Соломко, П. В. Ольшанський // Науковий вісник Чернівецького університету. Комп'ютерні системи та компоненти. – 2014. – Т. 5. – Вип. 2. – С. 114–123. 11. Тарасенко В. П. Частково-груповий перенос суматорів у скінченному полі GF(P) [Електронний ресурс] / В. П. Тарасенко, О. К. Тесленко, А. І. Роговенко // Вісник Національного університету "Львівська політехніка" "Комп'ютерні системи та мережі". – 2013. – № 773. – С. 118–125. – Режим доступу http://nbuv.gov.ua/jpdf VNULPKSM_2013_773_21.pdf – 10.04. 2015 р. – Загол. з екрана. 12. Logic Friday [Електронний ресурс]. – Режим доступу: http://sontrak.com/. – 10.04. 2015 р. – Загол. з екрана.