УДК 004.75

В.В. ЯЦКІВ
Тернопільський національний економічний університет

МЕТОДИ ВИКОНАННЯ МОДУЛЯРНИХ ОПЕРАЦІЙ ТА ЇХ РЕАЛІЗАЦІЯ НА ПЛІС

В роботі розглянуто методи виконання модулярних операцій та їх реалізація на ПЛІС. Наведено результати експериментальних досліджень апаратних затрат та швидкодії виконання модулярних операцій для одиничних модулів та наборів модулів з різними діапазонами представлення чисел. Проведені дослідження показали, що найбільш ефективним, при реалізації на ПЛІС, є метод різниці квадратів. Ефективність виконання операції множення в значній мірі залежить від величини вибраних модулів та їх кількості.

Ключові слова: система залишкових класів, модулярні операції, програмовані логічні інтегральні схеми.

V.V. YATSKIV
Ternopil National Economic University, Ternopil, Ukraine

THE MODULAR OPERATIONS METHODS AND THEIR IMPLEMENTATION ON CPLD

The aim of the work is experimental investigation of hardware and time complexity of the basic modular operations implemented on CPLD.

The methods of modular operations and implementation on CPLD are examined in this work. The results of experimental research of hardware complexity and modular operation processing speed for single modules and the set of modules for different ranges of number representation are shown. All examined modular operation methods implemented in Verilog. These methods were verified in Quartus II 13.0.

The conducted investigations showed that the most effective is difference of squares method. The effectiveness of multiplication depends on the value of chosen modules and of their number. If the digital capacity of modules is 3-5 bits the hardware complexity will be two- three times less in comparison with hardware complexity of 8-10 bits digital capacity of modules.

Keywords: residue number system, modular operation, complex programmable logic device.

Вступ

Підвищення швидкодії апаратних засобів обробки даних можна досягти завдяки новим ефективним методам і алгоритмам обробки та новій технології виготовлення апаратних засобів, зокрема, використанню апаратних засобів з вищою тактовою частотою. Проблема підвищення швидкодії особливо актуальна при обробці мультимедійних даних, які характеризуються великими обсягами та чутливістю до затримки, а також при обробці даних в протоколах передавання телекомунікаційних мереж, зокрема, при застосуванні коректуючих кодів. Оскільки обробка даних часто пов'язана з виконанням великої кількості арифметичних операцій, то від ефективності реалізації останніх в значній мірі залежить ефективність обробки в цілому. Як відомо, виконання арифметичних операцій в системі залишкових класів (СЗК) має ряд переваг, які досягаються за рахунок можливості паралельного виконанням арифметичних операцій, малої (3–6 біт) розрядності залишків та незалежності залишків [1, 2].

Аналіз останніх досліджень та публікацій

Теоретичні основи системи залишкових класів закладені в [1] та розвинуті в [2]. В роботах [3, 4] приведено теоретичні основи побудови спецпроцесорів в базисі Крестенсона, який породжує систему залишкових класів. В роботі [5] розроблено метод індексного модулярного множення, однак не проведено його дослідження. В літературі достатньо уваги приділено методам виконання модулярних операцій, однак мало досліджено складність реалізації, зокрема, при реалізації модулярних операцій на ПЛІС [5]. Отже, проведений аналіз публікацій показав, що експериментальні дослідження складності реалізації методів виконання модулярних операцій на ПЛІС потребують додаткового дослідження.

Постановка задачі

Основною елементною базою для реалізації систем обробки даних на даний час є мікропроцесори, мікроконтролери, програмовані логічні інтегральні схеми (ПЛІС) та системи на кристалі (System-on-a-Chip, SoC). Застосування ПЛІС робить процес розробки більш гнучким, оскільки дає можливість використовувати готові IP ядра мікропроцесорів з необхідним набором периферійних контролерів. Ефективність виконання арифметичних операцій в СЗК на ПЛІС в значній мірі залежить також від методів та алгоритмів їх виконання. В роботі для оцінки ефективності виконання арифметичних операцій вибрано апаратні затрати (кількість логічних елементів) та час виконання відповідних операцій (максимальна затримка проходження сигналу).

Формулювання цілей

Метою роботи є експериментальне дослідження апаратної та часової складності методів виконання основних модулярних операцій при їх реалізації на ПЛІС.

Арифметичні операції в СЗК

В СЗК число A представляється у вигляді залишків від ділення на взаємно прості модулі p_i [1]:

$$A = (b_i, b_2, b_{3,} \dots, b_n),$$

де b_i – залишки, $b_i = A(\mathrm{mod}\, p_i)$ або $b_i = A - \left[\frac{A}{p_i}\right] \cdot p_i$, $[\bullet]$ – заокруглення до меншого цілого.

Розглянемо виконання основних арифметичних операцій в СЗК [1]

– додавання в СЗК

$$\begin{cases} c_i = a_i + b_i, & при\ \ a_i + b_i < p_i \\ c_i = a_i + b_i - p_i, & при\ \ a_i + b_i \geq p_i \end{cases}; \quad (1)$$

– віднімання в СЗК

$$\begin{cases} c_i = a_i - b_i, & при\ \ a_i - b_i \geq 0 \\ c_i = a_i - b_i + p_i, & при\ \ a_i - b_i < 0 \end{cases}; \quad (2)$$

– множення в СЗК

$$\begin{cases} c_i = a_i \times b_i, & при\ \ a_i \times b_i < p_i \\ c_i = (a_i \times b_i) \mathrm{mod}\, p_i, & при\ \ a_i \times b_i \geq p_i \end{cases}. \quad (3)$$

Використовуючи формули (1, 2) та оператори мови Verilog, арифметичні операції додавання та віднімання в СЗК можна записати різними способами, наприклад:

спосіб М.1: додавання – `assign c=(a+b)%p;`

віднімання – `assign c=(a-b)%p;`

спосіб М.2: додавання – `assign c=((a+b)<m)?(a+b):((a+b)-p);`

віднімання – `assign c=((a-b)>=0)?(a-b):((a-b)+p);`

де a, b – доданки; p – модуль.

Проведені дослідження розглянутих способів опису операції модулярного додавання мовою Verilog показали, що спосіб М2 є більш ефективним з точки зору апаратних затрат і часу виконання операції (рис.1, рис.2) відповідно.

В таблицях 1, 2 наведені результати досліджень апаратної складності та швидкодії виконання операцій додавання та віднімання в СЗК для різних наборів модулів та діапазонів представлення чисел, де спосіб 1 – опис на мові Verilog з використанням оператора отримання залишку %, спосіб 2 – опис на мові Verilog операцій додавання і віднімання на основі формул 1 і 2. При дослідженні використовувалась різна кількість модулів та їх значення.

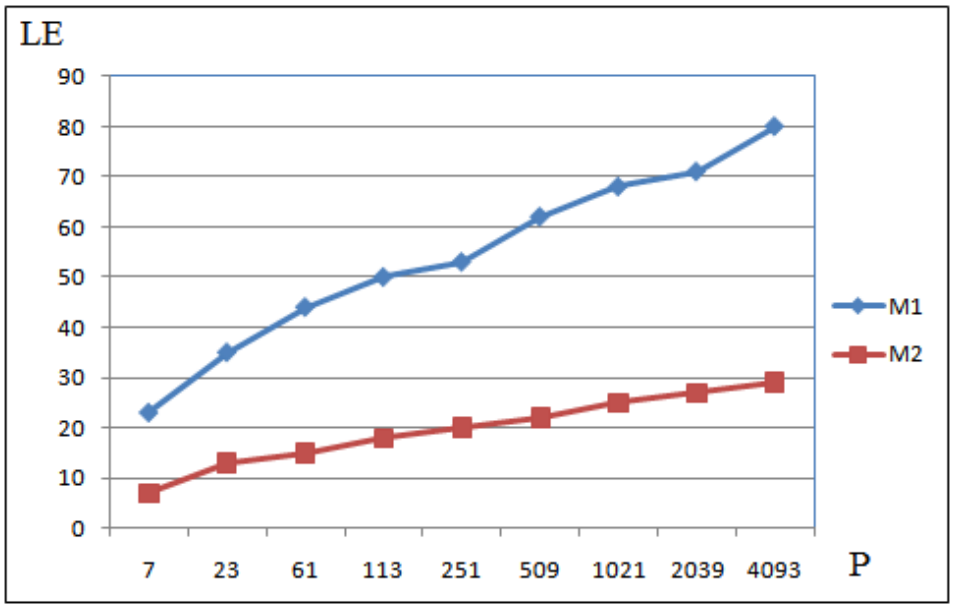

Рис.1. Залежність апаратних затрат від значення модуля при реалізації модулярного додавання способами М1 і М2

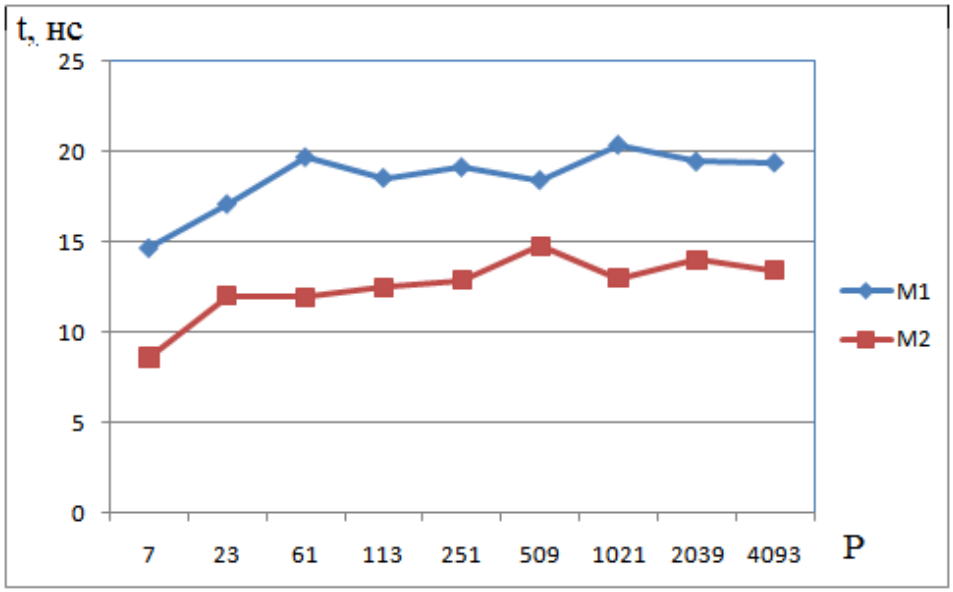

Рис.2. Залежність часу затримки від значення модуля при реалізації модулярного додавання методами М1 і М2

Таблиця 1

Параметри виконання операції додавання в СЗК

Діапазон представлення даних: 2^n	Значення модулів	Кількість логічних елементів		Час затримки, нс	
		Спосіб 1	Спосіб 2	Спосіб 1	Спосіб 2
n=8	3, 7, 13	64	24	16,437	8,889
n=16	239, 277	115	42	20,072	13,340
	29, 43, 47	120	43	18,610	11,864
	3, 7, 11, 13, 23	128	48	17,768	11,517
n=24	4091, 4111	160	61	19,779	14,164
	229, 239, 307	168	62	19,156	12,908
	7, 23, 29, 59, 61	181	64	17,371	12,379

Таблиця 2

Параметри виконання операції віднімання в СЗК

Діапазон представлення даних 2^n	Значення модулів	Кількість логічних елементів		Час затримки, нс	
		Спосіб 1	Спосіб 2	Спосіб 1	Спосіб 2
8	3, 7, 13	48	23	14,916	8,614
16	239, 277	83	36	20,436	10,71
	29, 43, 47	78	37	17,038	9,912
	3, 7, 11, 13, 23	131	44	18,549	11,416
24	4091, 4111	118	52	20,612	11,931
	229, 239, 307	138	53	18,704	12,193
	7, 23, 29, 59, 61	164	56	23,256	11,716

Як видно з таблиць 1, 2, опис на мові Verilog на основі формул (1, 2) способом 2 забезпечує зменшення кількості логічних елементів приблизно в 2,5 – 3 рази та підвищення швидкодії майже в 2 рази.

Модулярне множення

Модулярне множення, порівняно з додаванням і відніманням, є більш складною операцією, тому розглянемо детально методи його реалізації.

1. Виконується множення двох чисел, а потім за допомогою операції ділення знаходимо залишок у вигляді

$$r_i = (a_i \times b_i) \bmod p_i = (a_i \times b_i) - q \cdot p_i ,$$

де $q_i = \left\lfloor \frac{a_i \times b_i}{p_i} \right\rfloor$.

Даний метод є достатньо простий для реалізації засобами мови Verilog, але має значну затримку виконання.

2. Залишок знаходиться за допомогою операції послідовного віднімання модуля: поки $(a_i \times b_i) \geq p_i$,

зменшувати $(a_i \times b_i)$ на p_i.

Даний метод доцільно використовувати при малій розрядності модулів $n < 5$.

3. Операцію множення по модулю p_i можна реалізувати з використанням таблиці множення розміром p_i на p_i, що потребує збереження масиву даних розміром $p \times p$. Зменшення розміру масиву можна досягти використовуючи властивість симетрії вказаних таблиць, при цьому зростають апаратні затрати на реалізацію логічних переходів (розгортання скорочених таблиць). Отже, виконання модулярного множення на основі таблиць доцільно при розрядності модулів $n < 5$.

4. Метод індексного множення

Метод індексного множення ґрунтується на твердженні: якщо модуль p просте число, то будь-яке число $\{q_n\} = \{1, 2, \ldots, p-1\}$ може бути представлене у виді степеня первісного кореня g по модулю p: $q_n \equiv \left| g^{i_n} \right|_p$, де $\{i_n\} = \{0, 1, \ldots p-2\}$.

Отже, якщо $A = \left| g^{i_A} \right|_p$ і $B = \left| g^{j_B} \right|_p$, то їх добуток по модулю p дорівнює:

$$\left| A \cdot B \right|_p = \left| g^{\left| i_A + i_B \right|_{p-1}} \right|_p . \qquad (4)$$

Приклад. Задано числа $A = 3$, $B = 6$, треба знайти їх добуток по модулю 7.

Первісний корінь для модуля $p = 7$ дорівнює $g = 3$, відповідно відсортована таблиця індексів в порядку зростання має вигляд (табл.3):

Таблиця 3

Таблиця індексів для модуля 7

$q = \left\| g^i \right\|_7$	1	2	3	4	5	6
i	0	2	1	4	5	3

Перейдемо від чисел $A = 3$, $B = 6$ до їх індексів, отже $i_A = 1$, $i_B = 3$.

За формулою (4) знаходимо результат множення $\left| A \cdot B \right|_7 = \left| 3^{\left| 1+3 \right|_6} \right|_7 = 4$.

Знайти результат множення можна також використовуючи обернену таблицю, для цього необхідно додати індекси $i_A = 1$, $i_B = 3$ по модулю $p-1$ і за таблицею 3 зайти добуток. Оскільки $\left| 1+3 \right|_6 = 4$, згідно таблиці 4 $\left| A \cdot B \right|_7 = 4$.

Таблиця 4

Обернена таблиця індексів для модуля 7

i	0	1	2	3	4	5
$q = \left\| g^i \right\|_7$	1	3	2	6	4	5

Структурна схема пристрою множення на основі індексного методу наведена на рисунку 3.

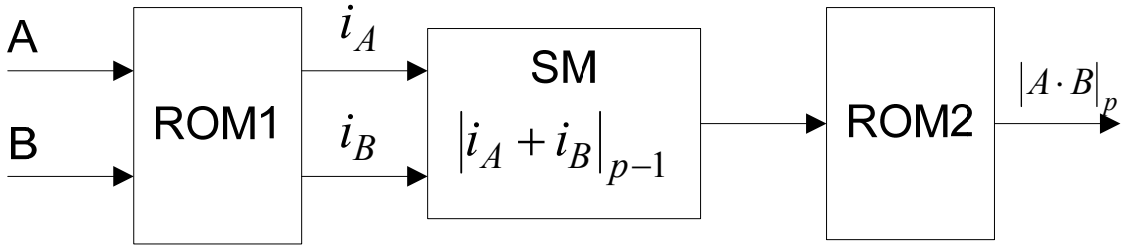

Рис.3. Структурна схема пристрою множення на основі індексного методу: ROM1 – пам'ять для зберігання таблиці залишки – індекси; ROM2 – пам'ять для зберігання оберненої таблиці індекси – залишки; SM – суматор по модулю $p-1$

Обсяг таблиць залежить від значення модуля і дорівнює $2(p-1)$ комірок.

Підвищити швидкодію виконання операції множення на основі індексного методу можна за рахунок розпаралелювання роботи суматора індексів. Це можливо зробити, якщо модуль $p-1$ можна

розбити на попарно взаємно прості множники $p-1=m_1\cdot m_2\cdot\ldots\cdot m_r$. Тоді операцію додавання можна розбити на r операцій додавання меншої розрядності. В цьому випадку індекси набудуть виду: ($|i|_{m_1}$, $|i|_{m_2}$,…., $|i|_{m_r}$), а додавання проводиться незалежно по кожному елементу вектора (рис.4) [2, 5].

Наприклад, для модуля 7 (попередній приклад) суматор по модулю 6 можна замінити двома суматорами по модулю 2 і по модулю 3.

В пам'яті ROM1 зберігаються залишки від індексів за відповідними модулями ($m_1, m_2, \ldots, m_r$).

5. Метод модулярного множення на основі різниці квадратів

Множення чисел можна реалізувати використавши формулу [2]:

$$A\times B=\frac{(A+B)^2}{4}-\frac{(A-B)^2}{4}, \tag{5}$$

для виконання операції множення по модулю p формулу (5) запишемо у вигляді

$$|A\times B|_p=\left|\frac{(A+B)^2}{4}-\frac{(A-B)^2}{4}\right|_p. \tag{6}$$

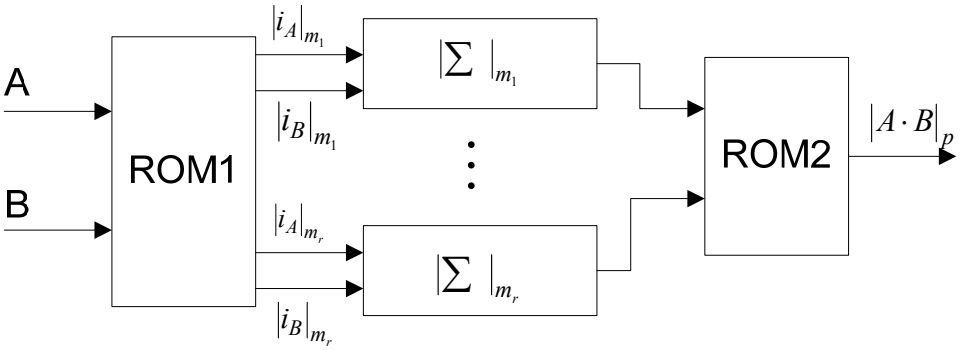

Рис. 4. Структурна схема пристрою множення на основі індексного методу з розпаралелюванням операції додавання

Операцію ділення на 4 замінимо операцією множення на обернений елемент:

$$|A\times B|_p=\left|\left|(A+B)^2\cdot\left|4^{-1}\right|_p\right|_p-\left|(A-B)^2\cdot\left|4^{-1}\right|_p\right|_p\right|_p,$$

де $\left|4^{-1}\right|_p$ – обернений елемент до $|4|_p$.

Тобто необхідно знайти таке x, щоб $|x\cdot 4|_p\equiv 1$ або $\left|4\cdot 4^{-1}\right|_p\equiv 1$. Наприклад, обернений елемент до 4 по модулю $p=7$ дорівнює 2, оскільки $|4\cdot 2|_7\equiv 1$.

На рис.5 наведена структурна схема пристрою множення на основі формули (6). При описі схеми на мові Verilog блоки 3 і 4 (рис.5) реалізовані у вигляді таблиць, при цьому обсяг таблиць залежить від значення модуля і дорівнює $2\cdot p-1$.

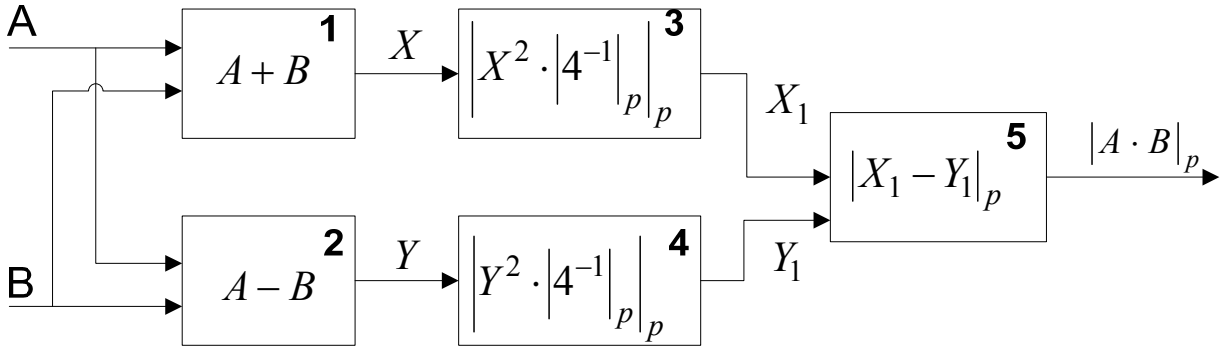

Рис. 5. Структурна схема пристрою множення на основі методу різниці квадратів

Проведені дослідження апаратної складності виконання операції модулярного множення показали, що при розрядності модуля менше 7 біт більш ефективними є метод індексного множення та метод різниці квадратів (рис.6).

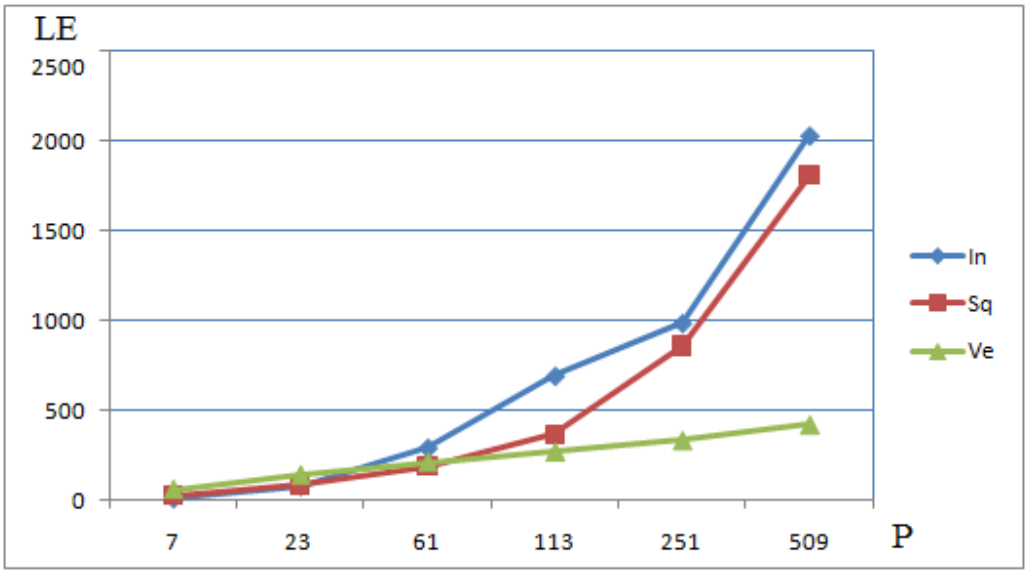

Рис. 6. Залежність апаратних затрат (кількість логічних елементів LE) від значення модуля при реалізації модулярного множення методами: In – індексне множення, Sq – множення на основі різниці квадратів, Ve – використання операторів мови Verilog

Однак при зростанні розрядності модулів менше логічних елементів потребує метод з використання операторів мови Verilog (рис.6). Дослідження часу затримки виконання операції модулярного множення (рис.7) показали, що при малій розрядності модулів (3–4 розряди) мінімальний час затримки забезпечує метод індексного множення, а при розрядності 5 біт і більше кращий результат забезпечує метод різниці квадратів.

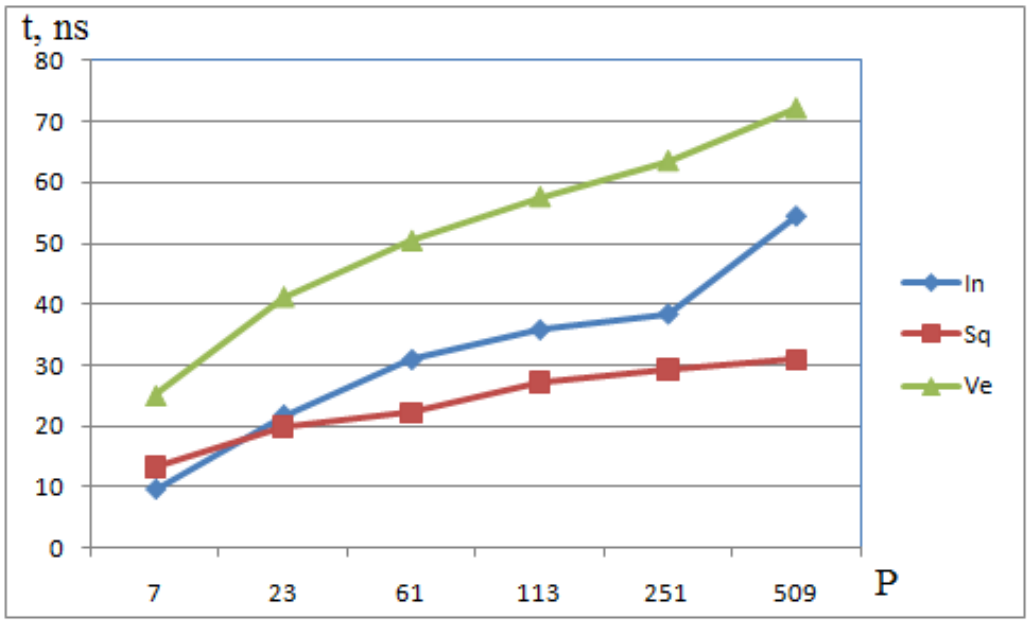

Рис. 7. Залежність часу затримки від значення модуля при реалізації модулярного множення методами: In – індексне множення, Sq – множення на основі різниці квадратів, Ve – використання операторів мови Verilog

Для дослідження ефективності виконання операції модулярного множення для різних наборів модулів та діапазону представлення вибрано три методи: метод 1 – використання операторів мови Verilog; метод 2 – індексне множення, метод 3 – різниця квадратів [3] та дві серії мікросхем: Max II та Cyclone II (табл.5).

Висновки

Проведені експериментальні дослідження показали, що найбільш ефективним є метод різниці квадратів. При розрядності модулів (3–5 біт) апаратні затрати в 2–3 рази менші, порівняно із апаратними затратами при розрядності модулів 8–10 біт (табл.5). Використання методу різниці квадратів (метод 3) підвищує швидкодію виконання операції множення в середньому на 40%. Ефективність виконання операції множення, зокрема, апаратні затрати, також залежать від величини вибраних модулів та їх кількості.

Отримані результати досліджень планується використати при розробці спецпроцесора кодування даних на основі модифікованих коректуючих кодів системи залишкових класів [6].

Подальшого підвищення ефективності виконання модулярних операцій можна досягти завдяки таким діям: 1) розробки нових методів отримання залишку числа та алгоритмів їх реалізації; 2) вибору спеціального набору модулів типу: $2^n-1, 2^n, 2^n+1$; 3) оптимізації коду на мові Verilog. Проведення попередніх досліджень показало, що пункти 2 і 3 дозволять підвищити ефективність виконання модулярних операцій приблизно на 10–20 %.

Таблиця 5

Параметри виконання операції множення в СЗК

Діапазон даних 2^n та значення модулів	Кількість логічних елементів			Час затримки, нс		
	Метод 1	Метод 2	Метод 3	Метод 1	Метод 2	Метод 3
n=8: 3, 7, 13	188	109	78	33,94	12,28	17,1
n=16 239, 277	757	2574	2719	71,37	35,94	31,23
29, 43, 47	552	758	421	46,61	28,84	23,1
3, 7, 11, 13, 23	430	409	207	40,06	24,33	20,04
n=24: 4091, 4111	1493 (Max II)	*		105,85 (Max II)	–	
	1042 (Cyclone II)			68,28 (Cyclone II)		
229, 239, 307	1089 (Max II)	4015 (Cyclone II)	4342 (Cyclone II)	74,91 (Max II)	40,18 (Cyclone II)	33,96 (Cyclone II)
	731 (Cyclone II)			46,29 (Cyclone II)		
7, 23, 29, 59, 61	767 (Max II)	883 (Max II)	596 (Max II)	48,06 (Max II)	29,25 (Max II)	22,7 (Max II)
	516 (Cyclone II)	900 (Cyclone II)	596 (Cyclone II)	32,76 (Cyclone II)	25,28 (Cyclone II)	20,1 (Cyclone II)

* – недостатньо апаратних ресурсів у вибраних серіях мікросхем.

Література

1. Акушский И.Я. Машинная арифметика в остаточных класах / И.Я.Акушский, Д.И.Юдицкий. – М. : Сов. Радио, 1968. – 460 с.

2. Omondi A. Residue Number System: Theory and Implementation / A.Omond, B.Premkumar. Imperial College Press, 2007. – Vol. 2. – 296 p.

3. Николайчук Я.М. Теоретичні основи побудови та структура спецпроцесорів в базисі Крестенсона / Я.М. Николайчук, О.І Волинський, С.В.Кулина // Вісник Хмельницького національного університету. – 2007. – № 3. – Т. 1. – С. 85–90.

4. Яцків Н. Г. Спецпроцесор обробки даних на основі перетворення Крестенсона – Галуа / Н. Г. Яцків, Р. І.Король, В. В.Яцків, Т. Г.Федчишин // Вісник Технологічного університету Поділля. – 2003. – № 3, Т.1. – С. 105–108.

5. Radhakrishnan D. A fast RNS Galois field multiplier / D. Radhakrishnan, Y. Yuan // Circuits and Systems, 1990, IEEE International Symposium on. – IEEE. – 1990. – P. 2909–2912.

6. Яцків В.В. Модифіковані коректуючі коди системи залишкових класів та їх застосування / В.В. Яцків // Інформаційні технології та комп'ютерна інженерія. – 2013 – № 2. – С. 39–45.

References

1. Akushskiy I.YA. Mashinnaya arifmetika v ostatochnykh klassakh. / I.YA.Akushskiy, D.I. Yuditskiy – M.: Sov. Radio. -1968. – 460 s.

2. Omondi A. Residue Number System: Theory and Implementation / Omondi A., Premkumar B. Imperial College Press. – 2007. – vol. 2. – 296 p.

3. Nykolaychuk YA.M. Teoretychni osnovy pobudovy ta struktura spetsprotsesoriv v bazysi Krestensona / YA.M. Nykolaychuk, O.I Volynskyy, S.V.Kulyna // Visnyk Khmelnytskoho natsionalnoho universytetu. – 2007. - №3. – T1. – S. 85-90.

4. Yatskiv N. H. Spetsprotsesor obrobky danykh na osnovi peretvorennya Krestensona – Galua / N. G.Yatskiv, Korol R. I., Yatskiv V. V., Fedchyshyn T. H. // Visnyk Tekhnolohichnoho universytetu Podillya. – 2003. –№3, T1. – S. 105 – 108.

5. Radhakrishnan D. A fast RNS Galois field multiplier / D.Radhakrishnan, Y.Yuan // Circuits and Systems, 1990, IEEE International Symposium on. – IEEE, 1990. – P. 2909-2912.

6. Yatskiv V.V. Modyfikovani korektuyuchi kody systemy zalyshkovykh klasiv ta yikh zastosuvannya / V.V. Yatskiv // Informatsiyni tekhnolohiyi ta kompyuterna inzheneriya. – 2013. – №2. – S.39-45.

Рецензія/Peer review : 23.09.2014 р. Надрукована/Printed :29.11.2014 р.
Рецензент: д.т.н., професор Николайчук Я.М.