УДК 004.382

В.С. Глухов
Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин

ОСОБЛИВОСТІ ВИКОНАННЯ ОПЕРАЦІЙ У ПРОСТИХ ПОЛЯХ ГАЛУА GF(P) У СУЧАСНИХ ЗАСОБАХ ЗАХИСТУ ІНФОРМАЦІЇ

Сучасні алгоритми захисту інформації вимагають виконання операцій $s=(e+dr) \bmod n$ у простих полях Галуа $GF(p)$, де n – просте, e, d, r – цілі багаторозрядні числа. Відомий метод Монтгомері, який полягає у виконанні вказаних операцій за модулем $N>n$, із зведенням всіх проміжних результатів r більших або рівних N за модулем n. N вибирають зручним для аналізу умови $r<N$. Недоліками методу є низька швидкодія довгих комбінаційних суматорів, необхідних для зведення за модулем n, невизначеність часу множення. У роботі обґрунтовуються можливості застосування послідовних суматорів замість методу Монтгомері і паралельних суматорів. Також на прикладі модульного помножувача проілюстрований новий метод вирішення проблеми *Hardware-Software Codesign*, який на відміну від методу програмної реконфігурації полягає у трансляції змінюваних параметрів від апаратної частини до програмної.

Ключові слова: прості поля Галуа, множення за модулем, додавання за модулем, метод Монтгомері, послідовний операційний пристрій, *Hardware-Software Codesign*.

Modern data protection algorithms require $s=(e+dr) \bmod n$ operations in the simple Galois fields $GF(p)$, where n – a simple, e, d, r – multidigital big numbers. A well-known method of Montgomery which is performing these operations by modulo $N>n$, with the correction of all intermediate results r greater than or equal N by modulo n. N is chosen convenient to analyze the conditions $r<N$. The disadvantage of this method is slow performance of bug adders, which are necessary for the addition by modulo n, the uncertainty of addition time. The paper substantiates the applicability of dedicated serial adders instead of Montgomery method and parallel adders. Also, new method for solving Hardware-Software Codesign problem is illustrated by example of a modular multiplier. Unlike of software reconfiguration method in new method parameters are transmitted from hardware to software.

Key words: A simple Galois field, modular multiplication, modular addition, Montgomery method, serial operational unit, *Hardware-Software Codesign*.

Вступ

Сучасні алгоритми захисту інформації вимагають виконання операцій $s=(e+dr) \bmod n$ у простих полях Галуа $GF(p)$, де n – просте, e, d, r – цілі багаторозрядні числа. Відомий метод Монтгомері, який полягає у виконанні вказаних операцій за модулем $N>n$, із зведенням всіх проміжних результатів r більших або рівних N за модулем n. N вибирають зручним для аналізу умови $r<N$. Недоліками методу є низька швидкодія довгих комбінаційних суматорів, необхідних для зведення за модулем, невизначеність часу множення. У роботі пропонується метод оцінки частоти виконання багаторазових приведень проміжного результату до модуля n залежно від вибраного модуля N. У роботі обґрунтовуються можливості застосування послідовних суматорів замість методу Монтгомері і паралельних суматорів. Також наводяться рекомендації з вибору модуля N і обґрунтовується відмова від застосування методу Монтгомері в сучасних засобах захисту інформації. Також на прикладі модульного помножувача проілюстрований новий метод вирішення проблеми *Hardware-Software Codesign*, який на відміну від методу програмної реконфігурації полягає у трансляції змінюваних параметрів від апаратної частини до програмної.

Аналіз публікацій і окреслення проблеми

Сучасні алгоритми захисту інформації [1] вимагають виконання модульних операцій з багаторозрядними числами: $s=(e+dr) \bmod n$, де n – просте, e, d, r – цілі багаторозрядні числа. Розрядність чисел дорівнює 163 біта і більше. Відомий метод Монтгомері [2], який полягає у виконанні вказаних операцій за модулем $N>n$, із зведенням всіх проміжних результатів r, більших або рівних N за модулем n. N вибирають зручним для аналізу умови $r<N$. Недоліками методу є низька швидкодія довгих комбінаційних схем підсумовування, необхідного для зведення за модулем n, невизначеність часу обчислення – для різних проміжних результатів можливо 0, 1 або 2 зведень за модулем n при множенні на один біт множника), а також можливість використання інших методів виконання модульних операцій обговорюються в роботі [7]. У роботі [3] обґрунтовується можливість використання послідовних операційних пристроїв у сучасних засобах захисту інформації, один з яких описаний у роботі [4]. Особливості виконання операцій у простих полях Галуа наведені в роботі [5]. Використання залежного від операндів часу виконання операцій для зламу сучасних засобів інформації описаний у роботі [6]. У роботах [8, 9] наведені тестові приклади для перевіряння роботи засобів захисту інформації. Підходи до вирішення проблеми *Hardware-Software Codesign* описані у роботі [10]. Актуальною є задача прискорення обчислень за методом Монтгомері, стабілізації часу обчислень або обґрунтування можливості застосування іншого методу, а також розв'язання вказаних задач з врахуванням проблеми *Hardware-Software Codesign*.

Постановка завдання

Метою роботи є визначення впливу модуля N у методі Монтгомері на кількість багаторазових зведень за модулем n, на час обчислень. Також метою є визначення можливості застосування інших методів при виконанні модульних обчислень і визначення методу вирішення проблеми *Hardware-Software Codesign*, яка виникає при зміні апаратної частини (схем спеціалізованих обчислювачів) комп'ютерних систем.

Короткий виклад розв'язання поставленої задачі

Однією з операцій оброблення електронних цифрових підписів [1] є додавання та множення за модулем n ($s = (e + dr) \bmod n$). Складність виконання цих операцій визначається великою розрядністю операндів та необхідністю порівняння проміжних результатів з модулем n [5]. Одним з методів прискорення таких операцій є метод Монтгомері, який полягає в переході від модуля n до модуля $N=2^m$. Якщо проміжний результат більший за N, він зводиться за модулем n (коректується). Для визначення кількості корекцій (додаткових операцій зведення за модулем n) використовується геометричне представлення (рис. 1) проміжних результатів (z) залежно від проміжних операндів (x, y) при виконанні модульних операцій за методом Монтгомері [2]. Таким способом визначена кількість корекцій залежно від співвідношення N та n (рис. 2).

Модульне множення розглядається як послідовність додавань і зведень за модулем n. На рис. 1 показана множина (у вигляді точок, що лежать у площині ромба *0AHB*) можливих результатів додавання двох операндів, кожний з яких під час множення може набувати значень [0, N-1].

Діапазон можливих значень суми – [0, 2N-2]. Ромб поділений на три частини. Трикутник *0AB* зображує результати, для яких не виконується зведення за модулем n, оскільки вони менше N. Трапеція *ACDB* зображує результати, для яких виконується одне зведення за модулем n, оскільки вони лежать в діапазоні [N, N+n-1]. Трикутник *CDH* зображує результати, для яких виконується два зведення за модулем n, оскільки вони лежать в діапазоні [N+n, 2N-2]. Для $N >> 1$ загальна кількість результатів додавання подано площею ромба *0AHB*: $S = \sqrt{3}N^2$. Кількість результатів додавання, для яких необхідно одне або два зведення за модулем, подано площею трикутника *AHB*: $S_{12} = \sqrt{3}N^2/2$. Частка результатів, для яких необхідно одне або два зведення за модулем $p_{1\partial} = S_{12}/S = 1/2$. Кількість результатів додавання, для яких необхідно два зведення за модулем, представляється площею трикутника *CHD*: $S_2 = \sqrt{3}(N-n)^2/2$. Частка результатів, для яких необхідно два зведення за модулем $p_{2\partial} = S_2/S = ((1-n/N)^2/2$. Для випадку $n \approx M/2$, який використовується на практиці [1], $p_{2\partial} \approx 1/8$.

Рис. 1 містить також геометричне трактування для результатів подвоєння множеного під час множення, це – лінія *0H*. Кількість корекцій при подвоєнні також може бути від 0 до 2. Загальна кількість результатів, кількість результатів з 1 або 2 корекціями, 2 корекціями відображаються відповідно відрізками 0H довжиною $L=\sqrt{6}N$, FH довжиною $L_{12}=\sqrt{6}N/2$, EH довжиною $L_2=\sqrt{6}(N-n)/2$. Частка результатів, для яких необхідно одне або два зведення за модулем $p_{1n}=L_{12}/L=1/2$, для яких необхідно два зведення за модулем $p_{2n}=L_2/L=((1-n/N)/2$. Для випадку $n\approx M/2$, який використовується на практиці [1], $p_{2n}\approx 1/4$.

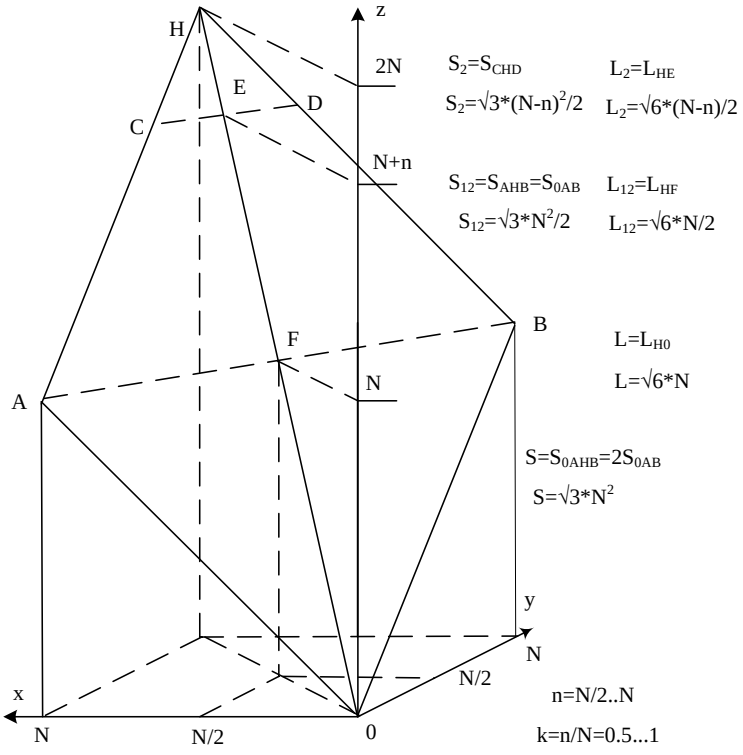

Рис. 1. Кількість корекцій під час підсумовування і підготовка множеного

При множенні на кожний розряд множника можуть виникати і корекції додавання, і корекції подвоєння. Кількість корекцій залежно від операндів може змінюватися від 0 до 4 на один розряд множника. Середня кількість корекцій на кожний біт множника при модульному множенні лежить в діапазоні 40/32 до 44/32 (рис. 2). Застосування методу Монтгомері в сучасних засобах криптографічного захисту інформації веде до нестабільного часу виконання множення і не завжди виправдане. Різна тривалість операцій залежно від операндів демаскує роботу пристрою і вимагає боротьби із цим явищем [6].

Зведення за модулем *n* для багаторозрядних чисел є повільною операцією, оскільки вимагає виконання багаторозрядного додавання. Використання суматорів з послідовним переносом збільшує затримку на комбінаційних схемах, тим самим зменшує тактову частоту роботи пристрою. Метод Монтгомері збільшує невизначеність часу виконання операцій.

Для збереження високої тактової частоти пропонується [7] використовувати послідовні спецпроцесори [3] і виконувати додавання при виконанні модульного множення за допомогою однорозрядних суматорів з накопиченням, які одночасно можуть формувати і накопичувати ознаку перевищення результатом граничного значення n (рис. 3). У разі виявлення такого перевищення здійснюється наступне зведення результату за модулем *n* з використанням цього ж однорозрядного суматора. Зведення за модулем *n* полягає у відніманні *n* від проміжного результату, що зводиться до додавання доповняльного коду числа *n* до проміжного результату.

k= 0,50

Кількість корекцій при додаванні		$p_{ід}\backslash p_{ін}$	Кількість корекцій при підготовленні множеного: 2 рази $4L_2/L$	1 раз $4(L_{12}-L_2)/L$	0 разів $4L_0/L$		і - Кількість корекцій на 1 біт множника	p_i - частка (x32)	Кількість корекцій при множенні
			1,00	1,00	2,00		4	1	4
							3	4	12
2 рази	$8S_2/S$	1,00	1	1	2	$S_2/S=(1-k)^2/2$	2	9	18
1 раз	$8(S_{12}-S_2)/S$	3,00	3	3	6	$(S_{12}-S_2)/S=(1-(1-k)^2)/2$	1	10	10
0 разів	$8(S-S_{12})/S$	4,00	4	4	8	$(S-S_{12})/S=1/2$	0	8	0
			$L_2/L=(1-k)/2$	$(L_{12}-L_2)/L=k/2$	$L_0/L=1/2$	Кількість корекцій на 1 біт множника:			44 / 32

k= 1,00

Кількість корекцій при додаванні		$p_{ід}\backslash p_{ін}$	Кількість корекцій при підготовленні множеного: 2 рази $4L_2/L$	1 раз $4(L_{12}-L_2)/L$	0 разів $4L_0/L$		і - Кількість корекцій на 1 біт множника	p_i - частка (x32)	Кількість корекцій при множенні
			0,00	2,00	2,00		4	0	0
							3	4	12
2 рази	$8S_2/S$	0,00	0	0	0	$S_2/S=(1-k)^2/2$	2	8	16
1 раз	$8(S_{12}-S_2)/S$	4,00	4	4	8	$(S_{12}-S_2)/S=(1-(1-k)^2)/2$	1	12	12
0 разів	$8(S-S_{12})/S$	4,00	4	4	8	$(S-S_{12})/S=1/2$	0	8	0
			$L_2/L=(1-k)/2$	$(L_{12}-L_2)/L=k/2$	$L_0/L=1/2$	Кількість корекцій на 1 біт множника:			40 / 32

Рис. 2. Кількість корекцій

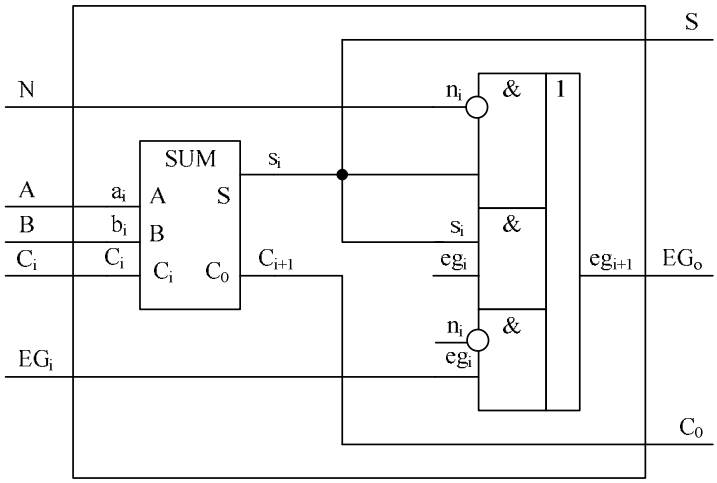

Рис. 3. Однорозрядний суматор FA_EG з вузлом порівняння

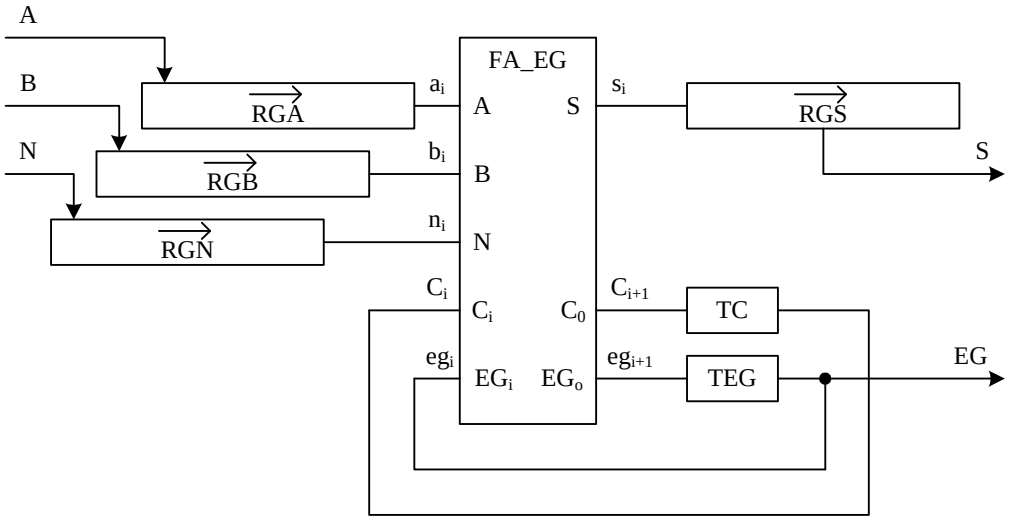

Рис. 4. Суматор за модулем на основі суматора з вузлом порівняння

Основним елементом багаторозрядного суматора за модулем n (рис. 4) є одноразрядний суматор з накопиченням суми і ознаки перевищення сумою заданого значення *n*, який керується контролером. Чергові розряди операндів подаються на суматор з регістрів зсуву. Результат також формується на регістрі зсуву:

Апаратні та часові характеристики модульного помножувача на основі послідовного суматора наведені нижче (таблиця 1, таблиця 2). Для порівняння у цих же таблицях наведені дані інших вузлів криптографічного процесора, до складу якого входить модульний помножувач [4].

Варіант вирішення проблеми Hardware-Software Codesign

Використання модульного помножувача у різноманітних спецпроцесорах призводить до необхідності зміни його параметрів (наприклад, величини числа *n*) і врахування цих змін програмним забезпеченням спецпроцесора (так звана проблема *Hardware-Software Codesign – HSC*). Ця проблема виникає під час проектування апаратно-програмних комп'ютерних засобів, коли одночасно проектуються і апаратні засоби комп'ютера, і його програмні засоби. Часто зміни в апаратних засобах призводять до змін у програмах, що є небажаним.

Таблиця 1

Характеристики вузлів шифропроцесора

Вузол	Призначення вузла	*Min* період тактової частоти, нс	К-сть слайсів	К-сть блоків блочної пам'яті	К-сть вентилів
zak	Проект у цілому	29.548	4,192	166	2,810,452
Zak/ECProc	СП оброблення точок ЕК відповідно до [1]	24.769	2,447	165	2,748,260
Zak/ECProc/cpu173n	173-бітний процесор для нормального базису *GF*(2^{173}) в складі *ECProc*	27.925	2,443	165	2,748,739
Zak/ECProc/cpu173n/fsm_ALUn	АЛУ в складі *cpu173n*	20.106	707	11	194,577
Zak/ECProc/cpu173n/ /CS_MUL_mod	Вузол модульного множення в складі *ECProc*	25.395	1,141	134	2,217,254
Zak/ECProc/cpu173n/ /CS_MUL_mod/Add_Ser	Послідовний багаторозрядний суматор в складі *CS_MUL_mod*	11.289	382	1	24,379
Zak/ECProc/cpu173n/ /CS_MUL_mod/Add_Ser/fa_eg	Однобітний суматор в складі *Add_Ser*	2.904	2	0	18
Zak/ECProc/cpu173n/procN	Протокольний процесор в складі *cpu173n*	26.256	451	2	39,196
Zak/Buf_R300vsAtmega	Контролер генератора випадкових кодів в складі *Zak*	6.176	124	2	35,098
Zak/cript1	СП зашифрування та розшифрування відповідно до [8] у складі *Zak*	12.831	333	0	13,640
Zak/key_gen	Генератор ключів під час гешування відповідно до [9] у складі *Zak*	5.196	536	0	8,936
Zak/interleaver	Перемішувач під час гешування відповідно до [[9] у складі *Zak*	10.522	424	0	7,915

Примітка: Тип ПЛІС – xcv3200e-7fg1156

Одним з відомих варіантів вирішення цієї проблеми є програмно реконфігуровані системи (рис. 5, а), які складаються з протокольної програмної частини (універсального (мікро)контролера з відповідним програмним забезпеченням) і спеціалізованої апаратної частини (реалізується на ПЛІС). Параметр, що змінюється (p_i), визначає режим роботи програмного забезпечення

Software(p_i), обробляється ним і передається в спеціалізовану частину як вказівник на один з режимів роботи її апаратури. При цьому в програмній і апаратній частинах повинні бути враховані усі можливі режими їхньої роботи *Software(p_i)* та *Hardware(p_i)* відповідно.

Таблиця 2

Час виконання криптографічних перетворень (Тип ПЛІС – xcv400e)

Вузол / Операція	$t_{операції}$, мкс	$F_{вузла}$, МГц	Тестовий приклад
CS_MUL_mod / Модульне множення	11543	16	*s=(e+dr)modp* [1, Додаток В.2] *p*=0800000000000000000000189B4E67606E3825BB2831 *e*=070516411E5D9886B8486ECE54A30E9403D103B95F90 *d*=049FF09C848613AEA23699F78C960D5174617311ADCC *r*=0477ECC260F390FB6D0AE4AE3B7A78120F8EC458EF9A *s*=0472EA56AE478F95F1EC9F628FF43857E168B50FB819
ECProc / Множення точки ЕК на константу	118186	16	*Q=-dP* [1, Додаток В.2]
cript1 / Шифрування 64-бітного блока	5	50	[9, с.7, с. 8]
Interleaver / Перемішування під час гешування	9	50	[9, с.10]
key_gen / Генерация ключей К1, …, К4 під час гешування	0,3 (на кожний ключ)	50	[9, с.10]

Для спецпроцесорів, до яких можна зарахувати шифропроцесор [4], пропонується інший варіант вирішення проблеми, при якому змінюваний параметр (p_i) спочатку враховується в апаратурі спеціалізованої частини, а вже з неї транслюється до мікроконтролера комп'ютерної системи, де враховується його програмним забезпеченням (рис. 5, б). Програмна частина як і у попередньому варіанті повинна враховувати усі можливі режими роботи (усі можливі значення параметра, що змінюється). Апаратна спеціалізована частина створюється заново і містить лише схему *Hardware(p_i)*, яка залежить тільки від одного заданого значення параметра. Для ПЛІС – це створення і завантаження нового проекту (апаратна реконфігурація). Конфігурація, яка враховує тільки одне значення параметра, є простішою в реалізації, вимагає менших апаратних витрат, є надійнішою і споживає меншу потужність.

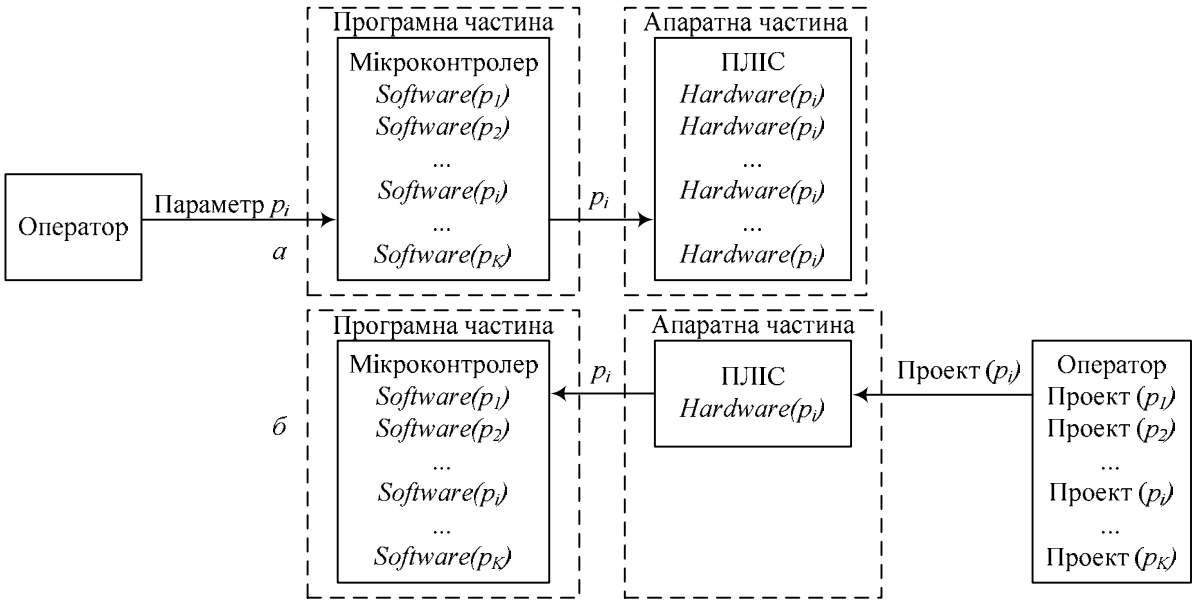

Рис. 5. Реконфігурована комп'ютерна система (а) та система, що забезпечує HSC

Модернізація комп'ютерної системи залежно від значення параметра p_i полягає у проектуванні та завантаженні нової конфігурації до ПЛІС за умови, що нове значення параметра було раніше враховане при розробленні програмного забезпечення.

Висновки

Показано, що при $N=2^m > n > 2^{m-1}$ зведення за модулем n при використанні методу Монтгомері виконується для 75% усіх проміжних результатів. При виконанні окремих множень на черговий біт множника кількість зведень за модулем може коливатися в межах 0…4. Отже, застосування методу Монтгомері в сучасних засобах захисту інформації веде до нестабільного часу виконання множення, час виконання залежить від операндів. Показано, що можливе виконання модульних операцій на послідовних швидкісних однорозрядних спеціалізованих процесорах без застосування методу Монтгомері. Також на прикладі модульного помножувача проілюстрований новий метод вирішення проблеми *Hardware-Software Codesign*, який на відміну від методу програмної реконфігурації полягає у трансляції змінюваних параметрів від апаратної частини до програмної.

1. ДСТУ 4145-2002. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевіряння. – К.: Державний комітет України з питань технічного регулювання та споживчої політики. 2003. 2. Montgomery P. L. Modular multiplication without trial division // Mathematics of Computation, vol. 44, no. 170, pp. 519–521, Apr. 1985. 3. Глухов В.С., Ногаль М.В. Спеціалізований однорозрядний процесор для захисту інформації в гарантоздатних системах // Наук.-техн. журн. "Радіоелектронні і комп'ютерні системи 5 (32). Національний аерокосмічний університет ім. М.Є. Жуковського "Харківський авіаційний інститут". – Харків: ХАІ. 2008. – С. 104–109. 4. Глухов В., Заіченко Н., Оліярник Б. Шифропроцесор для бортових інформаційно-керуючих систем. Наукові нотатки // Міжвузівський збірник (за напрямком "Інженерна механіка"), вип. 19 (січень 2007). Луцький державний технічний університет. – Луцьк. 2007. – С.33–43. 5. Ногаль М.В. Додавання і множення в полях Галуа // Вісник Нац. ун-ту "Львівська політехніка" "Комп`ютерні системи та мережі". № 603. – Львів, 2007. – С.105–111. 6. Глухов В.С., Еліас Р. Кодування станів керуючих автоматів у гарантоздатних системах // Наук.-техн. журн. "Радіоелектронні і комп'ютерні системи 5 (39). Національний аерокосмічний університет ім. М.Є. Жуковського "Харківський авіаційний інститут". – Харків: ХАІ, 2009. – С.91–95. 7. Глухов В. С., Элиас Р. Эффективность использования метода Монтгомери в современных средствах защиты информации // Матер. 12-ї міжнар. наук.-практ. конф. "Сучасні інформаційні та електронні технології". Одеса, 23–27 травня 2011 р. – С. 167. 8. ГОСТ 28147-89 Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. 9. Межгосударственный стандарт ГОСТ 34.311-95. Информационная технология. Криптографическая защита информации. Функция хэширования. Межгосударственный совет по стандартизации, метрологии и сертификации. – Минск: Госстандарт Украины, с дополнениями, 1997. 10. Schaumont, Patrick R. A Practical Introduction to Hardware/Software Codesign. 1st Edition, 2010, ISBN: 978-1-4419-5999-7. Springer. New York. Dordrecht. Heidelberg. London. September 2010. xviii+396.