УДК 004.383

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

ГЕНЕРАТОР ЯДЕР СЕКЦІОНОВАНИХ ПОМНОЖУВАЧІВ ЕЛЕМЕНТІВ ПОЛІВ ГАЛУА GF(2^M) ДЛЯ ОПТИМАЛЬНОГО НОРМАЛЬНОГО БАЗИСУ 2-ГО ТИПУ

Розглянуто генератор ядер (описів мовою *VHDL*) секціонованого помножувача елементів полів Галуа *GF*(2^m). Помножувач обробляє *m*-бітні елементи поля Галуа *GF*(2^m), представлені з використанням гауссівського нормального базису типу 2, і формує *m*-бітний добуток порціями по *n* бітів. Змінні *m* та *n* є параметрами, які може задавати користувач під час генерації ядра. Змінна *m* забезпечує формування помножувачів, які відповідають як стандарту ДСТУ 4145-2002 (*m*≤509), так і стандарту IEEE1363-2000 (*m*≤998). Помножувач містить вузол вбудованого контролю. Помножувач може використовуватися в процесорах оброблення цифрових підписів, які ґрунтуються на використанні еліптичних кривих. Наведено технічні характеристики згенерованих ядер (апаратні витрати і робоча частота).

Ключові слова: цифровий підпис, еліптичні криві, поле Галуа *GF*(2^m), гауссівський нормальний базис типу 2, помножувач, вбудований контроль, генератор ядер.

Scalable multiplier for Galois field GF(2^m) elements is examined. The multiplier uses type 2 Gaussian normal basis 521-bit Galois field GF(2^{521}) elements and forms 521-bit result by 16 bits portions. The multiplier forms the error flag in case error during the calculation. The multiplier is used in digital signature processors which are based on the use of elliptic curves.

Key words: Galois field GF(2^m), Gaussian normal base of type 2, multiplying, parity check, concurrent error detection, core generator.

Вступ

На сучасному етапі математичною основою цифрових підписів є еліптичні криві. За одним з варіантів реалізації цифрових підписів оброблення точок еліптичних кривих відбувається за правилами оброблення елементів полів Галуа *GF*(2^m). Розрядність елементів поля *m* може сягати тисячі бітів. Апаратна реалізація помножувача для таких полів вимагає понад мільйон транзисторів. Помножувачі можуть бути паралельними, послідовними та паралельно-послідовними – секціонованими. У роботах останніх років звертається увага на вбудовані методи виявлення помилок у роботі послідовних помножувачів за допомогою цифрового контролю на парність (зіставлення кількості 1 серед бітів операндів та результатів). Вбудованому контролю секціонованих помножувачів, які поєднують переваги паралельних (більша швидкодія) та послідовних (менші апаратні витрати), приділялася менша увага. Тому задача проектування секціонованих помножувачів елементів полів Галуа *GF*(2^m) з вузлами вбудованого контролю є важливою і актуальною. У роботі розглянуто генератор ядер (описів мовою *VHDL*) секціонованого помножувача елементів полів Галуа *GF*(2^m). Помножувач обробляє *m*-бітні елементи поля Галуа *GF*(2^m), представлені з використанням гауссівського нормального базису типу 2, і формує m-бітний добуток порціями по *n* бітів ($1 \le n \le m$). Змінні *m* та *n* є параметрами, які може задавати користувач під час генерації ядра. Змінна *m* забезпечує формування помножувачів, які відповідають як стандарту ДСТУ 4145-2002 (*m*≤509), так і стандарту IEEE1363-2000 (*m*≤998). Помножувач містить вузол вбудованого контролю.

Огляд літератури, постановка проблеми

Математичною основою цифрових підписів є еліптичні криві та поля Галуа. Одним з варіантів представлення елементів поля Галуа *GF*(2^m) є гауссівський нормальний базис типу 2 [1]. Для цього

базису відомий послідовний помножувач Мессі–Омурі [2], паралельний помножувач та паралельно-послідовний помножувач (секціонований) [3, 4], особливості синтезу помножувальних матриць для них розглянуті у роботі [5]. Методи вбудованого контролю результатів множення у нормальних базисах полів Галуа $GF(2^m)$ відомі з [6]. Для збільшення надійності роботи послідовного помножувача, який працює у нормальному базисі типу 2 відповідно до стандарту [1], запропоновано просту схему вбудованого контролю [7]. В Україні діє стандарт цифрового підпису на основі еліптичних кривих [1], який визначає максимальну характеристику поля Галуа m=509, тоді як міжнародний стандарт [8] дає змогу працювати з більшими полями (m≤998). Водночас, задача автоматичної генерації *VHDL*-описів (ядер) секціонованих помножувачів з вузлами вбудованого контролю не розглядалася.

Оскільки секціонований помножувач поєднує переваги послідовного (менші апаратні витрати) та паралельного (більша швидкодія) помножувачів і має додатковий вузол вбудованого контролю, ця задача є актуальною і важливою.

Мета роботи

Метою роботи є розроблення генератора *VHDL*-описів (ядер) секціонованих помножувачів елементів полів Галуа $GF(2^m)$, представлених у гауссівському нормальному базисі типу 2, з вузлами їх вбудованого контролю. Помножувач повинен обробляти m-бітні елементи поля Галуа $GF(2^m)$, представлені з використанням гауссівського нормального базису типу 2, і формувати m-бітний добуток порціями по n бітів ($1 \le n \le m$). Змінні m та n є параметрами, які користувач може задавати перед генерацією ядра. Змінна m повинна забезпечувати формування помножувачів, які відповідають як стандарту ДСТУ 4145-2002 (m≤509), так і стандарту IEEE1363-2000 (m≤998).

Структурна схема процесу проектування пристроїв для роботи із цифровими підписами

Структурну схему процесу проектування пристроїв для роботи із цифровими підписами наведено на рис. 5. Генератор ядер, якому присвячена дана стаття, знаходиться на цій структурній схемі в елементі «Генерація описів окремих вузлів спецпроцесора на *HDL: HLL*» (*HDL* – мова опису апаратних засобів, *HLL* – мова програмування високого рівня).

Математичні основи вбудованого контролю секціонованих помножувачів для GF(2m)

Секціонований m-бітний помножувач формує m-бітний добуток R в $]m/n[=K$ секціях порціями по n бітів ($k = 0, 1, \ldots, K-1$):

$(r_{m-kn-1}, \ldots, r_0, r_{m-1}, \ldots, r_{m-n+r}) = (f(a_{m-kn}, \ldots, a_{m-1}, a_0, \ldots, a_{m-kn-1}; b_{m-kn}, \ldots, b_{m-1}, b_0, \ldots, b_{m-kn-1}), \ldots, f(a_{m-n+r+1}, \ldots, a_{m-1}, a_0, \ldots, a_{m-n+r}; b_{m-n+r+1}, \ldots, b_{m-1}, b_0, \ldots, b_{m-n+r}))$ [24]. (1)

Ознака помилки E_R у роботі послідовного помножувача Мессі–Омури для гауссівського нормального базису типу 2 формується за формулою (2):

$$E_R = \sum_{i=0}^{m-1} (a_i b_i \oplus r_i), \tag{2}$$

де a_i, b_i – біти елементів поля A та B, r_i – біти результату R.

Модифікація формули (2) $E_R = \sum_{k=0}^{K-1} \sum_{i=0}^{n-1} (a_{kn+i} b_{kn+i} \oplus r_{kn+i}) = \sum_{k=0}^{K-1} e_k$ дає змогу ввести вузол вбудованого контролю у кожну секцію помножувача. При цьому кожна секція разом з розрядами добутку формує часткову ознаку $e_k = \sum_{i=0}^{n-1} (a_{kn+i} b_{kn+i} \oplus r_{kn+i})$ помилки множення. Для отримання загальної ознаки множення необхідно провести додаткову згортку часткових ознак.

Реалізація секціонованого помножувача із вбудованим контролем

Схема вузла формування часткових ознак (рис. 6) та спосіб її під'єднання до послідовного помножувача (рис. 7) відомі з [7]. До складу вузла формування часткових ознак входять двовходові елементи І та виключне АБО (*XOR*), а також лічильний тригер (*T*-тригер), на якому накопичується ознака під час виконання множення.

Послідовний помножувач Мессі–Омурі (рис. 7) складається з двох регістрів циклічного зсуву операндів *RGA* і *RGB* та помножувальної матриці *M*.

Секціонований помножувач з вузлами вбудованого контролю наведено на рис. 8. Помножувач (рис. 8) також складається з двох регістрів циклічного зсуву операндів *RGA* і *RGB*, 16-ти секцій *F*1, …, *F*16, які працюють за формулою (1), вузла згортки часткових ознак помилки *Error Convolution* та регістрового файла результату множення *Out RG File*. Також до складу секціонованого помножувача входить вузол керування *Control Unit*, який блокує формування *i*-х розрядів добутку, коли $i < 0$, та їхніх часткових ознак помилок.

Функціональну схему вузла згортки часткових ознак наведено на рис. 9. Для прикладу 515-бітного помножувача вузол складається з 4-х каскадів елементів виключне АБО (*XOR*), розділених конвеєрними регістрами *Pipeline A, B, C*.

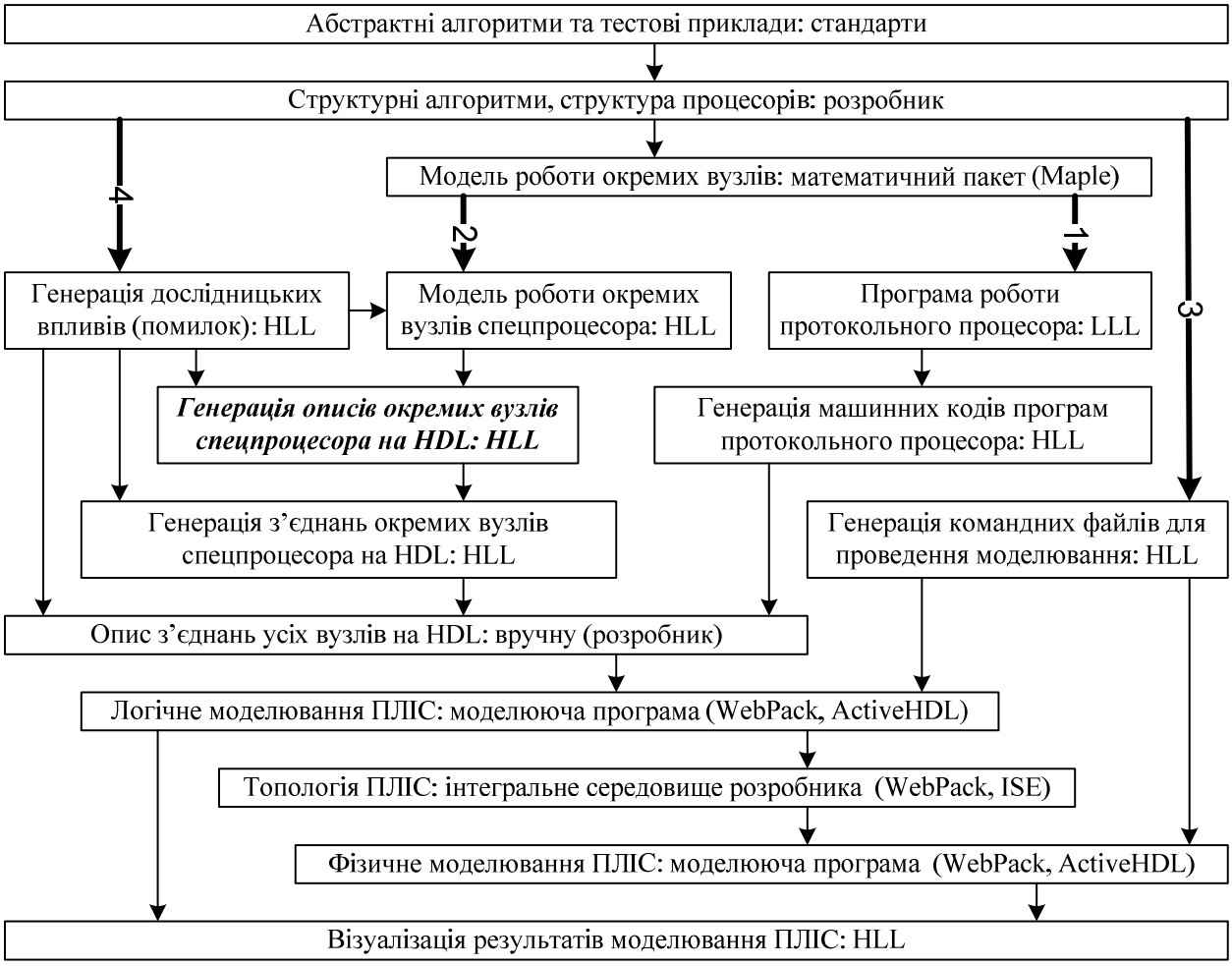

Рис. 5. Послідовність проектування описів функціональних вузлів

Послідовність роботи генератора

Основні етапи роботи генератора ядер:

введення параметрів *m* та *n*;

генерація утворювального полінома поля $GF(2^m)$ для оптимального нормального базису типу 2;

генерація помножувальної матриці;

генерація секцій помножувача;

генерація вузла вбудованого контролю кожної секції;

об'єднання секцій у єдиний вузол помножувача;

об'єднання секційних вузлів вбудованого контролю в єдиний вузол вбудованого контролю.

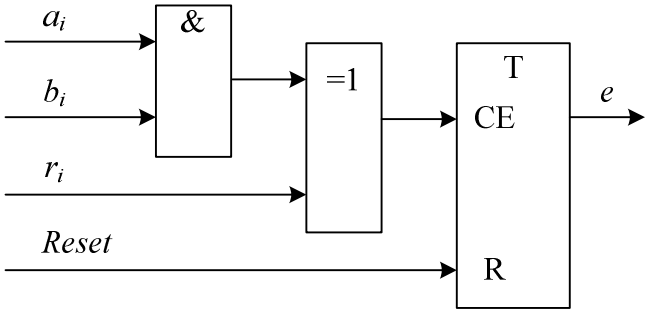

Рис. 6. Формувач часткових ознак помилки

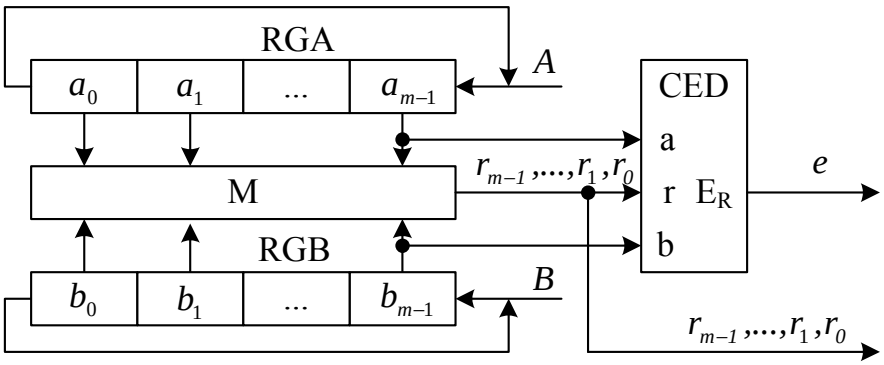

Рис. 7. Помножувач з вузлом виявлення помилок

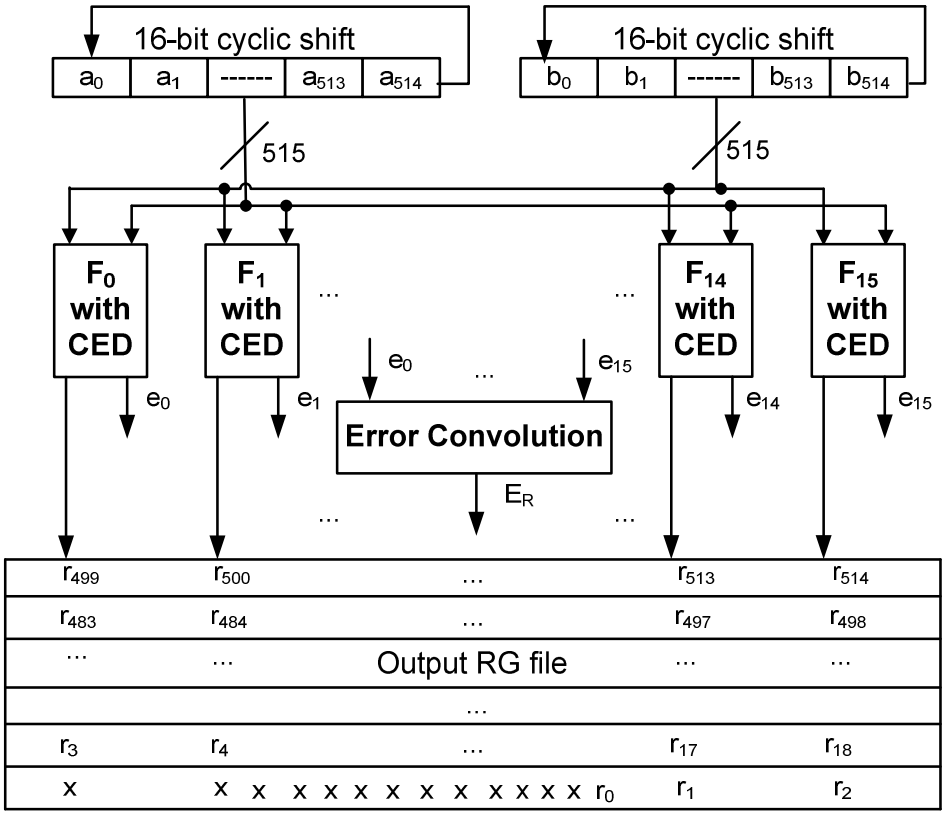

Рис. 8. Секціонований помножувач із вбудованим контролем

Особливості синтезу помножувальної матриці

Найскладнішою технологічною операцією під час синтезу *VHDL*-коду помножувальної матриці є знаходження оберненої матриці (*m* x *m*) [5]. Час знаходження оберненої матриці для різних полів наведено у табл. 1. Розрахунки для *m* = 515, 519, 530 проводилися на процесорі з тактовою частотою 2,6 ГГц, 2 ядра (працювало тільки одне ядро), об'єм оперативної пам'яті 4 Гбайти.

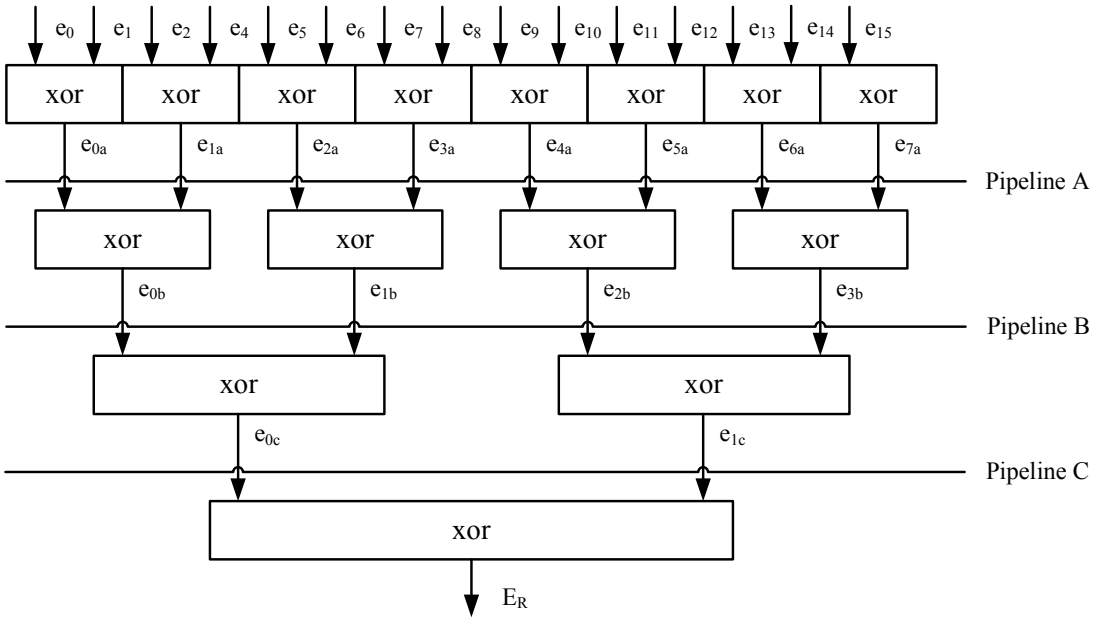

Рис. 9. Згортка часткових ознак помилок

\Таблиця 1

Час обчислення оберненої матриці

m	515	519	530	998
Час, год.	1 год. 35 хв.	1 год. 57 хв.	1 год. 39 хв.	50 год.

Логічна структура згенерованої помножувальної матриці наведена на рис. 10.

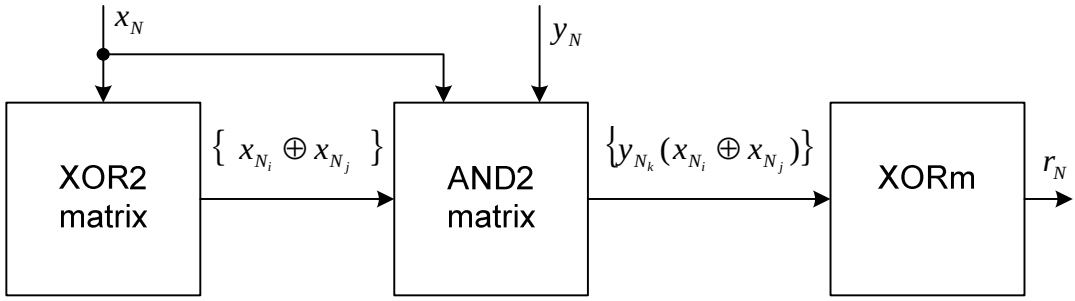

Рис. 10. Логічна структура помножувальної матриці

Можливі 2 варіанти синтезу множини елементів I матиці і формування з них результату множення. Послідовний варіант:

```
i:=1;
y(0) := '0';
L11: while i <= m loop
        y(i) := y(i-1) xor (c(i) and b(i)); i := i +1;
end loop L11;
x:=y(m);
```

Перевагою цього варіанта є простий опис з використання одного оператора *Loop* мови *VHDL* і одного параметра *m*. Недоліком є доволі тривалий час роботи засобів проектування топології кристала.

Паралельний варіант (наведений приклад для *m*=515):

```
i:=0;
x := '0';
a0: while i <= 127 loop
d(i) := (c(4*i+1) and b(4*i+1)) xor
  (c(4*i+2) and b(4*i+2)) xor
  (c(4*i+3) and b(4*i+3)) xor
  (c(4*i+4) and b(4*i+4));
i := i +1;
end loop a0;
i:=128;
d(i) := (c(4*i+1) and b(4*i+1)) xor
  (c(4*i+2) and b(4*i+2)) xor
  (c(4*i+3) and b(4*i+3));
i:=0;
a1: while i <= 31 loop
e(i) := (d(4*i+0) and d(4*i+0)) xor
  (d(4*i+1) and d(4*i+1)) xor
  (d(4*i+2) and d(4*i+2)) xor
  (d(4*i+3) and d(4*i+3));
i := i +1;
end loop a1;
i:=32;
e(i) := (d(4*i+0) and d(4*i+0));
i:=0;
a2: while i <= 7 loop
f(i) := (e(4*i+0) and e(4*i+0)) xor
  (e(4*i+1) and e(4*i+1)) xor
  (e(4*i+2) and e(4*i+2)) xor
  (e(4*i+3) and e(4*i+3));
i := i +1;
end loop a2;
i:=8;
f(i) := (e(4*i+0) and e(4*i+0));
i:=0;
a3: while i <= 1 loop
g(i) := (f(4*i+0) and f(4*i+0)) xor
  (f(4*i+1) and f(4*i+1)) xor
  (f(4*i+2) and f(4*i+2)) xor
  (f(4*i+3) and f(4*i+3));
  i := i +1;
end loop a3;
g(i) := (f(4*i+0) and f(4*i+0));

i:=0;
a4: while i <= 0 loop
x := (g(4*i+0) and f(4*i+0)) xor
  (f(4*i+1) and f(4*i+1)) xor
  (f(4*i+2) and f(4*i+2));
i := i +1;
end loop a4;
-- x – result
```

Перевагою другого методу є швидша робота засобів проектування топології кристала. Недоліки – складний опис з використання декількох операторів *Loop* мови *VHDL* та їх параметрів, які додатково треба розраховувати із заданого значення параметра *m*.

Результати імплементації та перевіряння роботи помножувачів

Гауссівський нормальний базис типу 2 існує для полів Галуа $GF(2^m)$ для *m*=515, 519, 530, 531, 543, 545, 554, 558, 561, 575, 585, 593, 606, 611, 614, 615, 618, 629, 638, 639, 641, 645, 650, 651, 653, 659, 683, 686, 690, 713, 719, 723, 725, 726, 741, 743, 746, 749, 755, 761, 765, 771, 774, 779, 783, 785, 791, 803, 809, 810, 818, 831, 833, 834, 846, 866, 870, 873, 879, 891, 893, 911, 923, 930, 933, 935, 938, 939, 950, 953, 965, 974, 975, 986, 989, 993, 998 [8] (перераховано тільки поля, для яких $509<m\leq 998$).

Результати імплементації деяких ядер для ПЛІС *6vcx130tff1156-2* засобами *Xilinx WebPack12* наведено у табл. 2 та 3 (* – синтез попередньо проведений засобами *Synopsys Synplify*).

Таблиця 2

Імплементація помножувача для *m*=515 (без CED)

n	1	8	16*	16	32
Кількість слайсів (%)	688 (3%)	1771 (8%)	1026 (5%)	2307 (11%)	7018 (35%)
Період синхроімпульсів, нс	5.523	8,5	13,1	13,8	13,8
Час імплементації, хв.		5	50	304	Немає даних

Таблиця 3

Імплементація помножувача для *m*=519 (без CED)

n	1	8	16	32
Кількість слайсів (%)	675 (3%)	2248 (11%)	3240 (16%)	6,112 (30%)
Період синхроімпульсів, нс	6.319	13.198	12.490	14.576

Сумарна кількість входів операндів та виходів результату одного помножувача дорівнює 3*m* і для вищезгаданих полів перевищує кількість контактів сучасних ПЛІС. Для перевіряння роботи згенерованих помножувачів використовується спеціально створений *VHDL*-стенд (рис. 11), до складу якого разом з помножувачем входять вхідні *FIFO* із записом по 128 бітів і читанням по 1024 біти та вихідний *FIFO* із записом по 1024 біти і читанням по 128 бітів. Сумарна кількість входів операндів та виходів результату стенда дорівнює 256, що дає змогу реалізувати його на великому наборі сучасних ПЛІС.

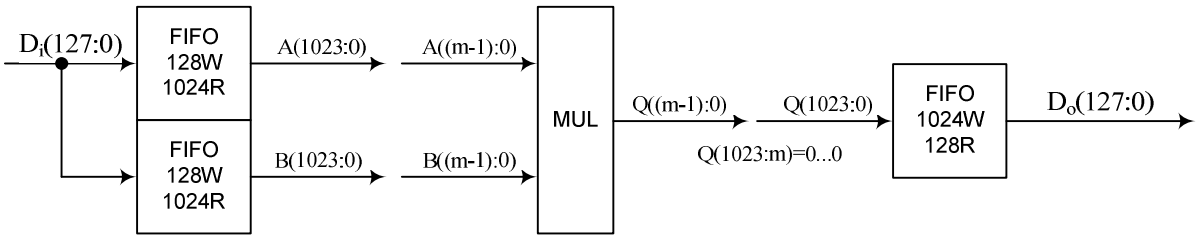

Рис. 11. Стенд для перевіряння помножувачів

Тестові приклади для перевіряння роботи помножувачів утворюють:

множення 1 х 1=1, при цьому у нормальному базисі 1 представляється кодом з *m* двійкових одиниць (1…1);

множення a х a = a^2, при цьому у нормальному базисі результат піднесення до квадрата операнда a дорівнює результату циклічного зсуву цього операнда a на один двійковий розряд (якщо $a=a_0a_1\ldots a_{m-1}$, $a^2=a_1\ldots a_{m-1}a_0$).

За результатами вказаних тестів згенеровані помножувачі працюють правильно.

Висновок

У роботі запропоновано генератор *VHDL*-описів (ядер) секціонованих помножувачів елементів полів Галуа $GF(2^m)$, представлених у гауссівському нормальному базисі типу 2, з вузлами їх вбудованого контролю. Помножувач обробляє m-бітні елементи поля Галуа $GF(2^m)$ і формує *m*-бітний добуток порціями по *n* бітів ($1\le n\le m$). Змінні *m* та *n* є параметрами, які користувач може задавати перед генерацією ядра. Змінна m забезпечує формування помножувачів, які відповідають як стандарту ДСТУ 4145-2002 ($m\le 509$), так і стандарту IEEE1363-2000 ($m\le 998$). У роботі наведені результати імплементації та тестування згенерованих помножувачів.

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. J. Omura and J. Massey. Computational method and apparatus for finite field arithmetic. U.S. Patent Number 4,587,627, May 1986. 3. Chanho Lee, Jeongho Lee. Design of an Elliptic Curve Cryptography Processor Using a Scalable Finite Field Multiplier in GF(2193). Journal of the Korean Physical Society, Vol. 44, No. 1, pp. 39–45. January 2004. 4. Elias Rodrigue. Design of an Elliptic Curve Cryptography Using A Finite Field Multiplier in GF(2521). Вісник Нац. ун-ту "Львівська політехніка" № 658 «Комп'ютерні системи та мережі». – 2009. – С. 144–149. 5. Глухов В.С. Особливості виконання операцій над матрицями в полях Галуа // Вісник Нац. ун-ту "Львівська політехніка" "Комп'ютерні системи проектування. Теорія і практика". Вип. 564. – 2006. – С.35–39. 6. Chiou-Yng Lee, Chin-Chin Chen, Erl-Huei Lu. Concurrent error detection in bit-serial normal basis of GF(2m). VLSI Test Technology Workshop. July 16-18, 2008, Tainan, Taiwan. 7. Глухов В.С. Вбудований контроль множення в гаусівському нормальному базисі типу 2 полів Галуа GF(2m). Науково-технічний журнал «Радіоелектронні і комп'ютерні системи 6(47). Національний аерокосмічний університет ім. М.Є. Жуковського «Харківський авіаційний інститут». – Харків: «ХАІ». – 2010. – С. 255–259. 8. IEEE Std 1363-2000 IEEE Standard Specifications for Public-Key Cryptography Sponsor Microprocessor and Microcomputer Standards Committee of the IEEE Computer Society. Approved 30 January 2000.