УДК 004.382

Є.М. Мотрич
Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин

ПРОЦЕСОР ДЛЯ ВИКОНАННЯ ОПЕРАЦІЙ НАД ЕЛЕМЕНТАМИ СКІНЧЕННИХ ПОЛІВ

Проаналізовано алгоритми виконання операцій додавання, множення та ділення над елементами скінченних полів, на основі проведеного аналізу зроблено обґрунтований вибір алгоритмів для реалізації цих операцій. Для обраних алгоритмів розглянуто принципи побудови та наведено функціональні схеми вузлів, які їх реалізовують. Схеми конкретизовані для реалізації спеціалізованого 163-розрядного процесора.

Ключові слова: поля Галуа, скінченні поля, поліноміальний базис, спеціалізований процесор.

The analysis of the algorithms of accomplishment of operations of additions, multiplication and division on the elements of the finite fields is being carried out. The substantiated selection of the algorithms for the realization of the above-mentioned operations is made on the basis of the conducted analysis. For the particular algorithms, the principles of the mathematical construction are being considered and the functional diagrams of the computer components, which implement them, are resulted. The previously-mentioned diagrams are specified for the implementation of a specialized 163-bit processor.

Key words: Galois fields, finite fields, polynomial basis, specialized processor.

Вступ

Як відомо, всезагальна комп'ютеризація крім очевидних переваг має і численні проблеми, однією з найскладніших є інформаційна безпека. Одним із способів забезпечення інформаційної безпеки є цифровий підпис. В Україні діють два стандарти на цифровий підпис: міждержавний стандарт ГОСТ 34.310-95 та національний стандарт України ДСТУ 4145–2002. Останній стандартизує використання полів Галуа та еліптичних кривих в алгоритмах отримання та перевірки цифрового підпису. Згідно з ДСТУ 4145–2002 елементи поля Галуа можуть бути представлені у поліноміальному або нормальному базисі. У статті проаналізовані алгоритми виконання операцій додавання, множення та ділення над елементами скінченних полів, які представлені у поліноміальному базисі та зроблено вибір алгоритмів, які краще підходять для реалізації.

Аналіз публікацій

Математичною основою для оброблення цифрових підписів є операції над елементами скінченних полів та операції над точками еліптичних кривих, ці операції детально розглянуті в джерелах [1–3]. В Україні діють два стандарти на цифровий підпис: міждержавний стандарт ГОСТ 34.310-95 [4] та національний стандарт України ДСТУ 4145–2002 [5]. За стандартом [5] алгоритми роботи з цифровим підписом можна умовно поділити на три рівні (рис. 1).

Оскільки сьогодні в Україні практично всі реалізації є програмними [6], основними недоліками яких є недостатня стійкість до зламу, часто недостатня продуктивність, особливо при обробці інтенсивних потоків даних, то для збільшення надійності реалізації виникає необхідність у створенні апаратних засобів для виконання операцій над елементами скінченних полів. Однією з можливостей є реалізація на ПЛІС.

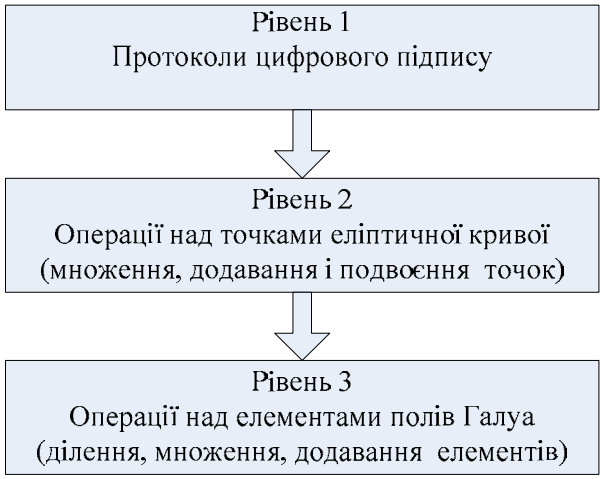

Рис. 1. Алгоритмічні рівні цифрового підпису за ДСТУ 4145-2002

Основною операцією під час оброблення елементів полів Галуа є множення. У роботі [7] описаний метод множення елементів скінченних полів, який не потребує розширення розрядної сітки. У роботі [8] проводиться аналіз апаратних і часових витрат на множення елементів поля Галуа $GF(2^n)$ і обчислення оберненого елемента поля Галуа $GF(2^n)$, однак детально не наведено принципів побудови схем, які виконують операції додавання, множення та ділення над елементами полів $GF(2^n)$. Це питання розглядається у цій статті.

Постановка задачі

Визначити принципи побудови схем, які виконують операції додавання, множення та ділення над елементами скінченних полів $GF(2^n)$, для подальшої реалізації спеціалізованого 163-розрядного процесора. Обчислення повинні здійснюватись в поліноміальному базисі скінченного поля $GF(2^{163})$, заданого примітивним многочленом $x^{163} + x^7 + x^6 + x^3 + 1$ відповідно до стандарту ДСТУ 4145-2002.

Скінченні поля. Скінченні поля є математичним об'єктом. Поле – це множина F, на якій визначені дві алгебраїчні операції додавання та множення, які мають такі основні властивості [1]:

– для будь-яких елементів $a,b,c \in F$: $(a + b) + c = a + (b + c)$ та $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ – асоціативність додавання та множення ;

– для будь-яких елементів $a,b \in F$: $a + b = b + a$ та $a \cdot b = b \cdot a$ – комутативність додавання та множення;

– для будь-яких елементів $a,b,c \in F$: $a \cdot (b + c) = a \cdot b + a \cdot c$ – дистрибутивність множення відносно додавання.

Поле називається скінченним, якщо воно складається із скінченної кількості елементів і нескінченним – у противному разі. Поля із скінченним числом елементів q називають полями Галуа і позначають GF(q). Число елементів поля q називають порядком поля. Залежно від значення q розрізняють прості або розширені поля. Поле називають простим, якщо порядок поля q є простим числом, тобто, $q = p$, де p – просте число. Просте поле позначається GF(p). Поле називають розширеним, якщо порядок поля – кількість його елементів – є простим числом в натуральному степені, тобто, $q = p^n$, де p – просте число, n – натуральне число і $n > 1$. Таке поле позначається $GF(p^n)$. Розширене поле $GF(p^n)$ утворюють многочлени степеня не більше n з коефіцієнтами із простого поля GF(p) (поліноміальний базис), а операції додавання і множення виконуються за модулем примітивного многочлена $f(x)$ степеня n над полем GF(p), що утворює це поле.

У роботі необхідно виконувати операції в розширених полях характеристики 2, тобто, в полях вигляду $GF(2^n)$, або як їх ще називають – двійкових полях. Це зумовлено тим, що елементи таких полів зручно представляти в комп'ютері, оскільки будь-який елемент поля $GF(2^n)$ можна представити у вигляді двійкової послідовності розрядності n. Тому надалі розглядатимуться саме такі поля. Приклад двійкового поля Галуа : $GF(2^2) = \{0,1,x,x+1\} = \{00,01,10,11\}$.

Процесор. Спеціалізований процесор повинен складатись з операційного пристрою та пристрою керування ним (рис. 2) і повинен працювати так.

Операційний пристрій виконує операції додавання, множення елементів та ділення многочленів із залишком. Вузол, який виконує множення елементів поля $GF(2^n)$, реалізований на основі вузла, який виконує додавання елементів поля $GF(2^n)$. Своєю чергою ділення елементів реалізоване на основі помножувача та елемента, який виконує ділення многочленів із залишком. Відповідно операційний пристрій складається з арифметико-логічного пристрою та регістрового файла, для збереження проміжних результатів під час виконання алгоритму множення та ділення.

Пристрій керування керує роботою операційного пристрою, забезпечуючи виконання потрібної операції, подає потрібні сигнали керування в потрібні моменти часу. Виходи пристрою керування з'єднані з входами операційного пристрою, на який надходять сигнали керування для арифметико-логічного пристрою (АЛП), та сигнали запису/читання, встановлення, скиду, вибору кристала для регістрового файла. Пристрій керування пропонується реалізувати у вигляді цифрового автомата, робота якого може бути описана графом. Залежно від коду операції автомат формує послідовність сигналів керування для виконання заданої операції. Сигнали *старт* та *готовність виконання* синхронізують роботу процесора (рис. 2).

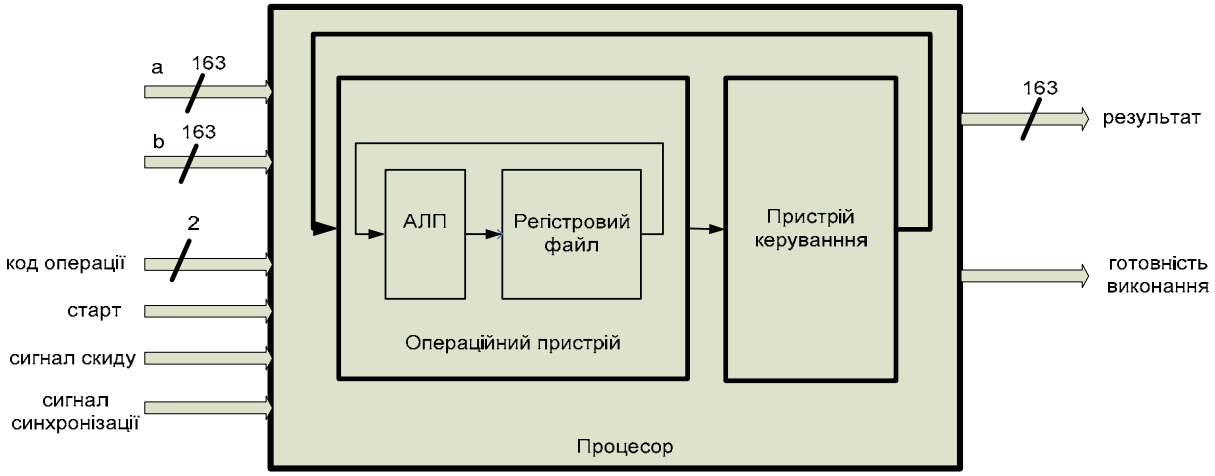

Рис. 2. Структурна схема процесора

Така структура процесора є стандартною: пристрій керування реалізовується у вигляді цифрового автомата, операційний пристрій реалізовується як комбінація арифметико-логічного пристрою та регістрового файла. Основна складність в реалізації спеціалізованого процесора припадає на арифметико-логічний пристрій (АЛП), який повинен виконувати операції додавання та множення елементів скінченних полів. Тобто основна складність припадає на реалізацію самих операцій, оскільки операції над елементами скінченних полів істотно відрізняються від звичайних операцій.

Операції над елементами скінченних полів та їх реалізація у процесорі. У розширеному полі $GF(2^n)$ визначені операції : додавання, віднімання, множення і ділення елементів. За стандартом [5] елементи поля можуть бути представлені у поліноміальному або нормальному базисі. У поліноміальному базисі елементи поля $GF(2^n)$ представляються як многочлени степеня $(n-1)$: $f(x) = a_{n-1}x^{n-1} + \ldots + a_1x + a_0$, де коефіцієнти $a_{n-1},\ldots,a_1,a_0$ – це 0 або 1. Додатково під час виконання операцій множення та ділення, крім двох операндів, використовується примітивний многочлен $f(x)$ степеня n, що утворює скінченне поле $GF(2^n)$, примітивний многочлен $f(x)$ потрібний при виникненні переповнення, для корекції проміжного результату.

У розширених двійкових полях Галуа віднімання дає той самий результат, що й додавання, і реалізується операцією XOR. Приклад : $(x^2 + 1) + (x^2 + x + 1) = 101 \oplus 111 = 010 = x$ (над $GF(2^3)$). Відповідна схема, яка реалізує додавання є простою, на її вхід надходять два операнди, над якими виконується операція XOR і результат подається на вихід.

Множення елементів виконується як множення звичайних многочленів із зведенням результату за модулем примітивного многочлена $f(x)$ при виникненні переповнення.

Однак за такого множення двох *n*-розрядних елементів поля GF(2^n) потрібно розширення розрядної сітки ще на (*n-1)* розрядів, тому при реалізації пропонується використати метод Мастровіто (Mastrovito) [7], для реалізації якого не потрібне розширення розрядної сітки. Нехай дані два *n*-розрядні елементи *a* та *b*, що належать полю Галуа GF(2^n), яке задане примітивним многочленом *f(x)*, алгоритм множення за методом Мастровіто виглядає так:

Вхідні дані : елементи $A=(a_{n-1}\, a_{n-2}\ldots a_0)$ та $B=(b_{n-1}\, b_{n-2}\ldots b_0)$, що належить полю GF($2^m$), поле утворене примітивним многочленом $F(x)=(f_n\, f_{n-1}\, f_{n-2}\ldots f_0)$; $S=(s_n\, s_{n-1}\, s_{n-2}\ldots s_0)$

Вихідні дані : $C = A*B$

1. Встановити $S \leftarrow 0$
2. Для ***i*** від ***m*** до 1 виконати
   2.1. Встановити $S \leftarrow S + B*a_i$
   2.2. Встановити $S \leftarrow S(s_{n-1}\, s_{n-2}\ldots s_0)*2$
   2.3 Якщо $s_n=1$ то $S \leftarrow S + F$
3. Встановити $S \leftarrow S + B*a_0$
4. $C = S$

Приклад множення методом Мастровіто:

Виконаємо множення $(x^2 + x + 1)\cdot(x^2 + x)$ – елементів поля GF(2^3) утвореного примітивним многочленом [2] $f(x) = x^3 + x^2 + 1 = \mathbf{1101}_2$

```
   111  (a=a2a1a0)
 * 110  (b=b2b1b0)
   -------
   110   S= b*a2
  --------
  1100   S= S*2 – логічний зсув вліво
+ 1101   Виникло переповнення, проводиться корекція результату за модулем f(x)
  --------
   001   S= S mod f(x)
  +110   b*a1
  --------
   111  S= S ⊕ (b*a1)
  --------
  1110  S= S*2 – логічний зсув вліво
+ 1101  Виникло переповнення, проводиться корекція результату за модулем f(x)
  --------
   011   S= S mod f(x)
 + 110   b*a0, останнє множення, логічний зсув вліво не виконується
  --------
   101  S= S ⊕ (b*a0) – результат – x^2 + 1
```

Реалізацію помножувача пропонується виконати так (рис. 3).

На вхід *OP_X(162:0)* регістру *shtf_reg_x* (елемент U1) надходить перший операнд множення, якщо сигнал *Wr_Sh* дорівнює 1, то регістр записує значення, яке надійшло на вхід *OP_X(162:0)* у внутрішню змінну, якщо сигнал *Wr_Sh* дорівнює 0, то на вихід *A* подається старший біт внутрішньої змінної, після чого над внутрішньою змінною виконується логічний зсув вліво.

На вхід *Q_OP_Y(162:0)* регістру *ffd_y* (елемент U2) надходить другий операнд множення, якщо сигнал *R* дорівнює 1, то регістр подає на вихід *Q_OP_Y(162:0)* значення, яке надійшло на вхід *OP_Y(162:0)*, якщо сигнал *R* дорівнює 0, то на вихід *Q_OP_Y(162:0)* подаються всі нулі.

Елемент *Mastrovito* (U3) виконує множення другого операнду (вхід *b(162:0)*) на старший біт першого операнду (вхід *a*) і результат додає за модулем 2 з попереднім проміжним результатом (вхід *r(162:0)*) (кроки 2.1, 3 алгоритму множення), якщо вхідний сигнал *c* дорівнює 1, то додатково виконується корекція проміжного результату множення (вхід *r(162:0)*) за модулем примітивного многочлена *f(x)* (крок 2.3 алгоритму множення), відповідний результат подається на вихід *o(162:0)*.

На вхід *R_In(162:0)* регістру *shtf_r_reg* (елемент U4) надходить вихід *o(162:0)* елемента *Mastrovito* – проміжний результат множення, якщо сигнал *Sh* дорівнює 1, то регістр на вихід *Cr* подає старший біт значення, яке надійшло на вхід *R_In(162:0)*, після чого виконується його логічний зсув вліво і результат подається на вихід *R_Res(162:0)* (крок 2.2 алгоритму множення), якщо сигнал *Sh* дорівнює 0, то регістр на вихід *Cr* подає 0, а значення, яке надійшло на вхід *R_In(162:0)* подається на вихід *R_Res(162:0)*. Якщо сигнал *CLR* дорівнює 1, то на вихід *R_Res(162:0)* подаються всі нулі. Регістри U1, U2, U4 працюють за зростаючим фронтом сигналу синхронізації *CLK*, відповідні сигнали керування *Wr_Sh, R, Sh, CLR* надходять з пристрою керування.

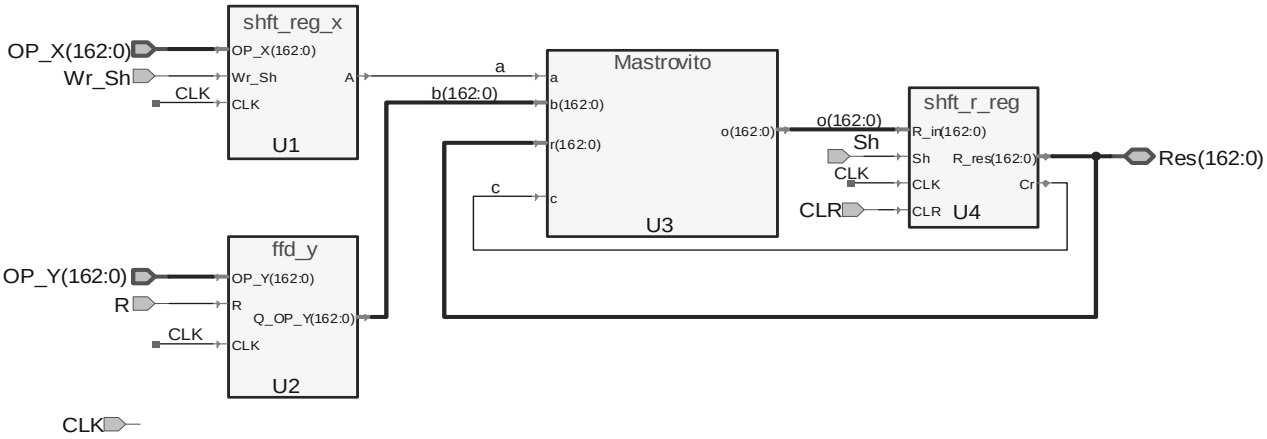

Рис. 3. Функціональна схема 163-розрядного помножувача за методом Мастровіто

Елемент *Mastrovito* пропонується реалізувати так (рис. 4). Перший елемент (U1) виконує додавання за модулем 2 проміжного результату множення (*r(162:0)*) і другого операнду множення (*b(162:0)*) (кроки 2.1, 3 алгоритму множення). Другий елемент (U2) виконує додавання за модулем 2 проміжного результату множення (*r(162:0)*) і примітивного многочлена *f(x)* (крок 2.3 алгоритму множення), код примітивного многочлена *f(x)* прописаний всередині другого елемента (U2). Третій елемент (U3) виконує додавання за модулем 2 значення з виходу другого елемента (U2) та другого операнду множення (*b(162:0)*) (крок 2.3 алгоритму множення). Мультиплексор (U4) залежно від старшого біту першого операнду множення *a*, та ознаки переповнення *c* вибирає, який з його входів *I0(162:0), I1(162:0), I2(162:0), I3(162:0)* подати на вихід *O(162:0)*.

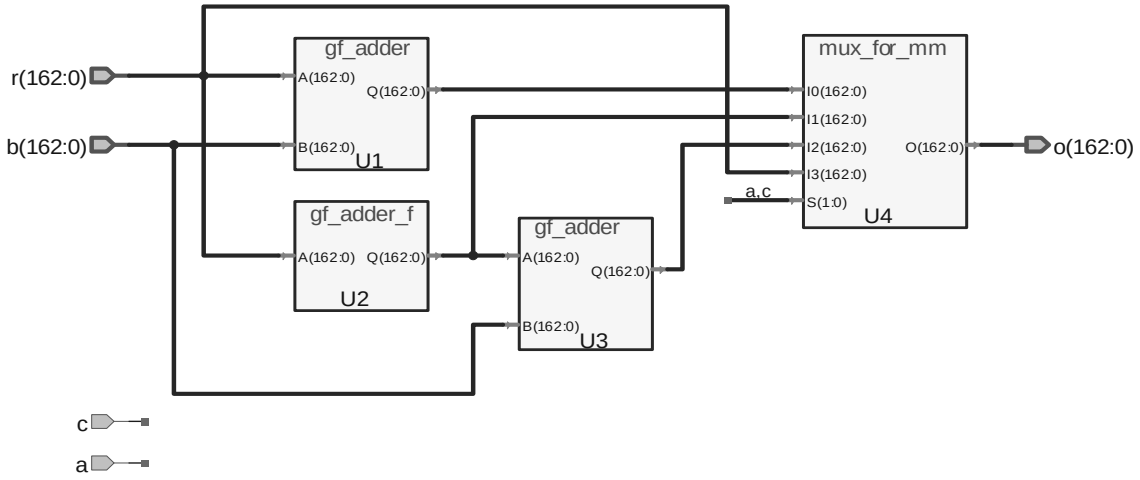

Рис. 4. Функціональна схема елемента Мастровіто

Ділення елементів виконується перетворенням його у множення $a/b = a \cdot b^{-1}$, де b^{-1} – обернений елемент до елемента *b*, причому $b \neq 0$, тобто, ділити на *0* не можна. Знаходження оберненого елемента є найтрудомісткішою операцією над елементами скінченних полів [8], для цієї операції використовується розширений алгоритм Евкліда [1, 5]. У стандарті [5] алгоритм знаходження оберненого елемента

описаний так: у поліноміальному базисі для знаходження оберненого елемента використовується узагальнений алгоритм Евкліда обчислення найбільшого спільного дільника (НСД) двох многочленів *f(t)* та *c(t)*. Цей алгоритм виражає НСД *(d(t))= a(t)f(t)+b(t)c(t)*, де *a(t) і b(t)* – деякі многочлени, що обчислюються під час виконання узагальненого алгоритму Евкліда. Цей алгоритм має такий вигляд:

Приймають *a(t)=1, d(t)=f(t), u(t)=0, v(t)= c(t).*

Якщо *v(t)=0*, то приймають *b(t)=(d(t)+f(t)a(t))/c(t)* та закінчують виконання алгоритму.

За допомогою ділення з залишком обчислюють *d(t))= q(t)v(t)+r(t)*, надалі обчислюють *w(t))= a(t) +u(t)q(t), a(t)=u(t), d(t)=v(t), u(t)=w(t), v(t)=r(t)* та переходять до кроку 2.

Якщо як *f(t)* взяти примітивний многочлен поля, а замість *c(t)* – многочлен, що зображає елемент поля, то *d(t)* є одиничним многочленом і наведене вище співвідношення за модулем примітивного многочлена перетворюється на співвідношення *b(t)c(t)=1 mod f(t)*, тобто, многочлен *b(t)* зображує елемент, обернений до *c(t)*.

Однак під час обчислення оберненого елемента *b(t)=(d(t)+f(t)a(t))/c(t)* для *n*-розрядного елемента поля потрібно розширення розрядної сітки ще на n розрядів (при множенні *f(t)a(t)*), що є нераціональним, тому у разі реалізації пропонується використати алгоритм ділення із стандарту [3], для реалізації якого не потрібне вказане розширення.

Вхідні дані : елемент α та $\beta \neq 0$, що належить полю $GF(2^n)$

Вихідні дані : $y = \alpha/\beta$

1. Встановити $r_0(t) \leftarrow p(t)$
2. Встановити $r_1(t) \leftarrow \beta$
3. Встановити $s_0(t) \leftarrow 0$
4. Встановити $s_1(t) \leftarrow \alpha$
5. Поки $r_1(t) \neq 0$
   - 5.1. Встановити $q(t) \leftarrow \lfloor r_0(t)/ r_1(t) \rfloor$
   - 5.2. Встановити $r_2(t) \leftarrow r_0(t) + q(t)r_1(t)$
   - 5.3. Встановити $s_2(t) \leftarrow s_0(t) + q(t)s_1(t)$
   - 5.4. Встановити $r_0(t) \leftarrow r_1(t)$
     Встановити $r_1(t) \leftarrow r_2(t)$
   - 5.5. Встановити $s_0(t) \leftarrow s_1(t)$
     Встановити $s_1(t) \leftarrow s_2(t)$
6. $y = s_0(t)$

Як видно, операція ділення є послідовністю операції додавання, множення, та ділення многочленів із залишком (пункт 5.1). Тобто ділення повинно виконуватись пристроєм керування, який послідовно подаватиме сигнали керування операційному пристрою, для виконання операцій додавання, множення, та ділення многочленів із залишком. Для операції ділення пропонується реалізувати елемент, який виконує ділення многочленів із залишком, логіка роботи якого наведена на прикладі нижче:

$(x^3 + x^2 + 1)/(x^2 + 1) = 1101_2/0101_2$

Зсунутий вліво дільник		ділене	частка ↓	
L(4) **0101**0000				
L(3) 0**0101**000	>	0000**1101**	0	– ліва частина більша
L(2) 00**0101**00	>	0000**1101**	0	– ліва частина більша
L(1) 000**0101**0	<	0000**1101**	1	– ліва частина менша, виконується сума за модулем 2 лівої і правої частини, результат записується у колонку ділене
L(0) 0000**0101**	<	0000**0111**	1	– ліва частина менша, виконується сума за модулем 2 лівої і правої частини, результат записується у колонку ділене
		0000**0010**		– залишок

Результат : частка – $0011_2 = x + 1$, залишок – $0010_2 = x$.

Перевірка : $(x^2 + 1)* (x + 1) + x = (x^3 + x^2 + x + 1) + x = x^3 + x^2 + 1$.

Така логіка роботи повинна бути реалізована у пристрої, що виконує ділення многочленів із залишком.

Висновки

У роботі проаналізовано алгоритми виконання операцій над елементами скінченних полів, зроблено обгрунтований вибір алгоритмів для реалізації цих операцій. Для цих алгоритмів розглянуто принципи побудови та наведено функціональні схеми, які реалізовують відповідно до операції додавання, множення та ділення над елементами скінченних полів. На основі аналізу та вибору алгоритмів зроблено вибір методу реалізації спеціалізованого процесора.

У статті проводиться вибір методу реалізації спеціалізованого процесора, тому наступним напрямком роботи повинна бути сама реалізація процесора на ПЛІС. При виборі методу реалізації процесора було вибрано розрядність елементів поля 163 – це найменша розрядність, яку рекомендує ДСТУ4145-2002. Доцільною є така реалізація процесора, при якій користувач може задавати потрібну йому розрядність в діапазоні, передбаченому у ДСТУ4145-2002.

1. Кузнецов Г.В., Фомичов В.В, Сушко С.О., та ін. Математичні основи криптографії. – Дніпропетровськ: Нац. гірничий ун-т, 2004. – 391с. 2. Лидл Р. Нидеррайт Г. Конечные поля: В 2-х томах. Т. 1. Перевод на русский язык с дополнениями. – М.: Мир, 1988. – 430 с. 3. IEEE 1363-2000: Standard Specifications For Public Key Cryptography. 2000. The Institute of Electrical and Electronics Engineers, Inc. 3. 4. Межгосударственный стандарт ГОСТ 34.310-95. Информационная технология. Криптографическая защита информации процедура выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма. Межгосударственный совет по стандартизации, метрологии и сертификации. – Минск: Госстандарт Украины, с дополнениями, 1997. 5. ДСТУ 4145-2002. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевіряння. – К.: Держ. комітет України з питань техн. регулюв. та споживч. Політ., 2003. 6. Сайт державної служби спеціального зв'язку та захисту інформації України. Перелік засобів КЗІ, які мають експертний висновок за результатами державної експертизи у галузі КЗІ – http://www.dstszi.gov.ua/dstszi/control/uk/publish/article?art_id=72121 7. Mastrovito E.D. VLSI architectures for multiplication over finite field GF(2m). In T. Mora, editor, Applied Algebra, Algebraic Algorithms, and Error-Correctin-Codes, 6th International Conference, AAECC-6, Lecture Notes in Computer Science, No. 357. – Ð. 297–309, Rome, Italy, July 1988. New York, NY: Springer-Verlag. 8. Глухов В.С. Порівняння поліноміального та нормального базисів представлення елементів полів Галуа. // Вісник Нац. ун-ту "Львівська політехніка" "Комп'ютерні системи проектування. Теорія і практика". – Львів, 2007. – № 591. – С. 22–27.