УДК 519.6:512.972, 004.27

Жуков І.А., д.т.н.,
Клименко І.А., к.т.н.

ОБЧИСЛЮВАЛЬНА СИСТЕМА ДЛЯ РОЗВ'ЯЗКУ НЕЧІТКИХ СЛАР

Національний авіаційний університет

Запропонована обчислювальна системи для рішення нечітких СЛАР, яка належить до класу убудованих обчислювальних систем реалізованих за технологією система-на-кристалі. У склад системи входить апаратний співпроцесор призначений для рішення чітких СЛАР від великої кількості змінних методом Гауса, який реалізований на ПЛІС Cyclon II Altera. Експериментальні дослідження роботи співпроцесора показали значне збільшення швидкодії.

Вступ

До традиційних задач, де виникає необхідність рішення систем лінійних алгебраїчних рівнянь від великої кількості аргументів, належать задачі управління технічними системами. Системи таких рівнянь являють собою математичні моделі технічних систем або входять як складові частини в алгоритми рішення різноманітних більш складних задач [1].

Сучасні задачі управління більшим чином зв'язані з прийняттям рішень та необхідності врахування наявності складної інформаційної ситуації, особливістю якого є винятково висока апріорна невизначеність про властивості об'єкта і вплив навколишнього середовища, неможливість безпосереднього спостереження і виміру основних параметрів об'єкта, чи велика неточність і неповнота апріорної інформації. Це головна причина використання інтелектуальних технологій, зокрема, методологій рішення задач управляння в умовах невизначеності [2, 3, 4].

Сучасна проблема управління за умов невизначеності належить до таких проблем теорії управління, де розв'язок навіть окремих задач цієї проблеми має велике теоретичне та прикладне значення. Однією з таких задач є задача розв'язку систем лінійних алгебричних рівнянь (СЛАР) за умов невизначеності, зокрема, коли коефіцієнти матриці та вільні члени подаються інтервалами або нечіткими змінними (числами).

Для розв'язання інтервальних СЛАР запропоновано достатньо велику кількість методів [2], зокрема, існує так зване *інтервальне розширення* методу Гаусса [2, 3]. Найбільш повне дослідження прямих методів розв'язання систем лінійних та нелінійних алгебричних рівнянь наведено в роботі [4].

Таким чином метою роботи є розробка апаратних засобів для рішення систем лінійних алгебраїчних рівнянь методом Гауса та дослідження їх ефективності. В якості елементної бази для реалізації обчислень застосовуються програмовані логічні інтегральні схеми. Сучасний рівень розвитку мікроелектроніки обумовлює ефективність реалізації апаратних засобів на ПЛІС для рішення задач означеного класу і відповідність цих пристроїв вимогам високої продуктивності. Надвисока степінь інтеграції сучасних мікросхем ПЛІС та їх технічні показники дозволяють реалізувати обчислювальні пристрої для обробки великих масивів даних з високою степеню паралелізму і в той же час з високою швидкодією і низьким рівнем енергоспоживання [5].

Методології рішення систем лінійних алгебраїчних рівнянь за умов невизначеності

Одним з поширених способів розв'язання систем лінійних алгебричних рівнянь з нечіткими змінними є спосіб представлення вихідної (нечіткої) задачі у вигляді сукупності чітких задач [6]. Представлення нечіткої змінної, як тензора [6, 7] дає змогу формулювати чітку блочну СЛАР у вигляді, коли всі параметри (матриця коефіцієнтів, вільні члени) є матрицями.

В роботі [8] запропонована методологія розв'язання систем лінійних алгеб-

ричних рівнянь з нечіткими змінними з матричною формою представлення параметрів моделі, що полягає в тому, що формується блочна чітка СЛАР, розв'язок якої виконується стандартними методами (зокрема методом Гауса), з подальшим формуванням еквівалентного псевдорішення на підставі рівності слідів та норм. Алгоритм розв'язку нечітких СЛАР у тензорному базисі,незалежно від кількості α-рівнів, дає змогу, представити початкову задачу у вигляді однієї задачі з блочними параметрами, що істотно спрощує процедуру обчислень.

Проведені розрахунки модельних прикладів показали високу ефективність запропонованого алгоритму [8].

Математична постановка задачі

За результатами аналізу відомих способів рішення систем лінійних алгебраїчних рівнянь та існуючих програмних і апаратних засобів [1, 4], слід зазначити, що ітераційні методи застосовуються для рішення поставленої задачі, але за поганої обумовленості системи рівнянь, що характерна для математичних моделей технічних систем, швидкість збіжності ітерацій і точність отриманих результатів зазвичай низькі. Більш ефективними є прямі методи, зокрема метод Гауса.

Одною з найважливіших особливостей методів обчислення розв'язків СЛАР з подальшою реалізацією на ЕОМ є їх здатність до розпаралелювання. Ця особливість покладена в основу реалізації запропонованих високопродуктивних апаратних засобів.

Система лінійних алгебраїчних рівнянь від n змінних має наступний вигляд (1):

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n = b_3 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = b_n \end{cases}$$

Метод Гауса оснований на послідовному виключенні невідомих із системи алгебраїчних рівнянь. Метод складається з двох основних етапів: виключення невідомих (прямий хід алгоритму) і послідовне обчислення значень невідомих (зворотний хід алгоритму).

Під час прямого ходу виконується перетворення вихідної системи рівнянь до нижньої трикутної форми, де останнє рівняння вміщує лише одну невідому, що дозволить за зворотним ходом здійснити безпосереднє послідовне обчислення невідомих у порядку від n-ї до першої. За прямого ходу алгоритму виконується $(n-1)$ ітерація перетворювань. Після кожної k-ї ітерації з системи рівнянь видаляється чергова змінна x_k. При цьому перетворення здійснюються над рівняннями від $(k+1)$-го до n-го, перші k рівнянь залишаються незмінними.

Більш наглядне подання методу Гауса, дає матрична форма запису системи рівнянь, яка окрім того значно легше інтерпретується в апаратну модель реалізації, ніж формалізований запис методу. Система лінійних алгебраїчних рівнянь (1) у матричній формі має вигляд $A\vec{X} = \vec{B}$, де A – квадратна матриця порядку n, $\vec{X}$ – n-мірний вектор невідомих, $\vec{B}$ – n-мірний вектор вільних членів.

Для адаптації до апаратної реалізації в матрицю A додається стовбець з вільними членами, формуючи так звану розширену матрицю A, що є вихідною для обчислень. Після остаточних перетворень в останньому $(n+1)$ стовбці отримуємо результат рішення системи рівнянь. Розширена матриця СЛАР має наступний вигляд:

$$\left|\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} & b_2 \\ a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_3 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_n \end{array}\right| . \qquad (2)$$

Після виконання першої ітерації алгоритму формується перетворена матриця $A^{(1)}$:

$$\left|\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} & b_2^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & \dots & a_{3n}^{(1)} & b_3^{(1)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & a_{n2}^{(1)} & a_{n3}^{(1)} & \dots & a_{nn}^{(1)} & b_n^{(1)} \end{array}\right|.$$

Подальші перетворення відбуваються лише над виділеним фрагментом матриці $A^{(1)}$, результатом яких є перетворена матриця $A^{(2)}$:

$$\left|\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} & b_2^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \dots & a_{3n}^{(2)} & b_3^{(2)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & a_{n3}^{(2)} & \dots & a_{nn}^{(2)} & b_n^{(2)} \end{array}\right|.$$

Після проходження всіх ітерацій прямого ходу матриця (2), має вигляд:

$$\left|\begin{array}{ccccc|c} v_{11} & v_{12} & v_{13} & \dots & v_{1n} & b_1^{\sim} \\ 0 & v_{22} & v_{23} & \dots & v_{2n} & b_2^{\sim} \\ 0 & 0 & v_{33} & \dots & v_{3n} & b_3^{\sim} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & v_{nn} & b_n^{\sim} \end{array}\right|. \qquad (3)$$

де v_{ij} – елементи перетвореної верхньої трикутної матриці V, $b_n^{\sim}$ – відкоригований вектор вільних членів, при цьому $v_{nn} = a_{nn}^{(n-1)}$, $b_n^{\sim} = b_n^{(n-1)}$.

За зворотного ходу алгоритму проводимо аналогічні розрахунки у зворотному порядку. Так само виконується $(n-1)$ стадія перетворювань матриці (3), після яких формується діагональна матриця. Після кожної k-ї ітерації з системи рівнянь видаляється чергова змінна x_k. Номер ітерації k змінюється у зворотному порядку від n до двох, перетворення здійснюються над рівняннями від $(n-1)$-го до першого, при цьому останні $(n-k)$ рівнянь залишаються незмінними. Далі наведена діагональна матриця $V^{(1)}$:

$$\left|\begin{array}{ccccc|c} a_{11}^{(1)} & 0 & 0 & \dots & 0 & b_1^{(1)} \\ 0 & a_{22}^{(2)} & 0 & \dots & 0 & b_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \dots & 0 & b_3^{(3)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a_{nn} & b_n \end{array}\right|.$$

На наступному етапі діагональна матриця приводиться до одиничної матриці, для цього виконується ділення кожного рядка $i = \overline{1,n}$ на відповідний коефіцієнт $a_{ij}(i=j)$, що знаходиться на головній діагоналі в цьому рядку.

$$\left|\begin{array}{ccccc|c} 1 & 0 & 0 & \dots & 0 & b_1^{(1)}/a_{11}^{(1)} = x_1 \\ 0 & 1 & 0 & \dots & 0 & b_2^{(2)}/\ a_{22}^{(2)} = x_2 \\ 0 & 0 & 1 & \dots & 0 & b_3^{(3)}/\ a_{33}^{(3)} = x_3 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 & b_n/\ a_{nn} = x_n \end{array}\right|.$$

В правому стовбці отриманий вектор рішень системи лінійних рівнянь.

Архітектурні особливості обчислювальної системи

В роботі запропонована обчислювальна системи для рішення нечітких СЛАР, яка належить до класу убудованих обчислювальних систем реалізованих за технологією система-на-кристалі. Основний алгоритм обчислень виконується на програмному рівні. У склад системи входить апаратний співпроцессор призначений для високопродуктивного рішення чітких СЛАР від великої кількості змінних методом Гауса. З точки зору загальної архітектурної класифікації обчислювальних систем розроблювана система належить до класу систем із загальною шиною. До складу обчислювальної системи (рис. 1) входить центральний процесор (ЦП), співпроцесор для рішення СЛАУ (СП), загальна пам'ять (ЗП), контролер переривань (КПП), контролер прямого доступу до пам'яті (КПДП), зовнішні пристрої (ЗП). Обмін даними між процесором та співпроцесором відбувається

через загальну пам'ять. Центральний процесор, який є програмованим процесорним ядром [9], виконує функції управління в системі та виконання основного алгоритму цільової задачі. Співпроцесор використовується для обчислення коренів СЛАР.

Обчислювальна система належить до класу статичних реконфігурованих систем, які після реконфігурації потребують компіляції, але разом з цим система відповідає вимогам гнучкості проектування та швидкої адаптації до вимог вирішуваної задачі. В якості рішення задачі реконфігурації в системі відносно збільшення складності вирішуваних задач може бути змінено кількість зовнішніх пристроїв, об'єм загальної пам'яті, система команд центрального процесору [9], розрядність оброблюваних слів, обчислювальна потужність співпроцесора відносно кількості змінних СЛАР. Окрім того обчислювальна система розробляється з перспективою подальшого масштабування обчислювальної потужності за рахунок збільшення кількості програмованих процесорних ядер.

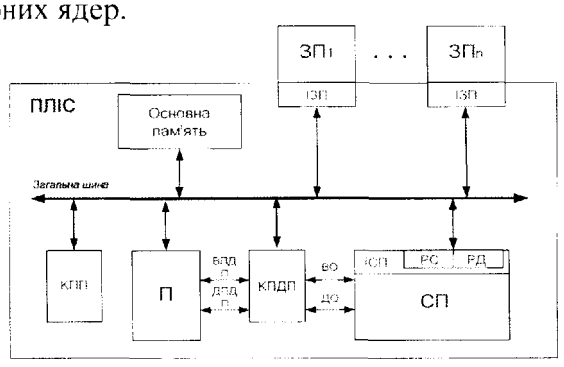

Рис. 1. Обчислювальна система на ПЛІС

В якості центрального процесора пропонується застосування програмованого процесорного ядра представленого авторами в роботі [9]. Процесор реалізує команди обміну даними з основною пам'яттю, зовнішніми пристроями та співпроцессором, арифметичні та логічні команди виконуються за один такт роботи системи. Процесорне ядро гнучке по відношенню до системи команд та розрядності оброблюваних слів. Як альтернатива, в якості програмованого процесорного ядра можливо використання процесорного ядра *Nios* II *Altera* в комплекті з загальною шиною *Avalon Altera*. Постачальник мікросхем ПЛІС пропонує спеціальні засоби проектування та розробки ситем-на-кристалі *SOPC Builder*, для автоматичної генерації ядра системи, шини і периферійних пристроїв, які можуть бути запропоновані для реалізації обчислювальної системи. Модулі процесорного ядра *Nios* II та шини *Avalon* включають засоби взаємодії з програмним середовищем *SOPC Builder*. Але запропонований підхід має певні проблеми з синхронізацією у мультипроцесорному режимі [10], тому для рішення поставленої задачі, яка має перспективу збільшення кількості процесорів, авторами пропонується власний механізм синхронізації на основі арбітражу системної шини [11]. На базі запропонованого підходу авторами планується рішення проблеми ефективної синхронізації в мультипроцесорній системі-на-кристалі.

Співпроцессор взаємодіє із системною шиною через інтерфейс співпроцесора (ІСП), в склад якого входить регістр даних (РД) і регістр стану (РС). Обмін даними між співпроцесором та загальною пам'яттю відбувається в режимі прямого доступу, який реалізує контролер прямого доступу до пам'яті (КПДП). Співпроцесор працює в пасивному режимі. За закінчення виконання чергової задачі співпроцесор формує сигнал вимоги обміну (ВО) до КПДП, який здійснює обмін управляючою інформацією з процесором (сигнали ВПДП, ДПДП – вимога/дозвіл ПДП) та процедуру підключення до системної магістралі і передачу масиву даних у пам'ять. Співпроцесор представлений в адресному просторі процесора двома адресами – адресою регістру стану, та адресою регістру даних.

Розробка архітектури співпроцесора на ПЛІС

Проект розроблений для ПЛІС сімейства *Cyclon* II фірми *Altera*. Пристрій синтезований на мові *Verilog*. Сімейство мікросхем *FPGA Cyclone* II актуальне на ринку ПЛІС, завдяки великій кількості логічних комірок, продуктивності, невисокій вартості та широкому набору вбудованих функцій. Основні характеристики сімейства [12]: кількість логічних ко-

мірок – від 4608 до 68416; убудована пам'ять об'ємом до 1,1 Мбіт, до 70 убудованих помножувачів, широкий набір стандартів вводу/виводу.

Для реалізації обчислювача з паралельним виконанням обчислень пропонується застосування конвеєрної архітектури. Конвеєрна реалізація алгоритму рішення СЛАР вирішує проблеми паралельної реалізації алгоритму пов'язані з обмеженням внутрішніх ресурсів ПЛІС, а саме каналів передачі даних та кількістю виводів мікросхеми. Для передачі m n-розрядних слів даних для m обчислювачів знадобиться nxm ліній передачі даних,що дозволить реалізувати на ПЛІС рішення СЛАР з вельми невеликої кількістю невідомих. Крім того навіть перспектива здійснити одночасно перетворення на паралельних обчислювачах всіх елементів рядку матриці не дасть суттєвого виграшу у часі за причини необхідності збереження отриманих даних у пам'яті. Для цього знадобиться кількість тактів, що дорівнює кількості елементів рядку отриманих в обчислювачах. Конвеєрна реалізація дозволяє в кожному такті видавати черговий перетворений елемент рядку з подальшим збереженням його у пам'яті.

Обчислювач складається з $(n+1)$ блоків (де n – кількість невідомих СЛАУ), в кожному з яких обробляється один елемент рядка матриці. За один прохід алгоритму в обчислювачі обробляється один рядок матриці, після чого формується перетворений рядок із видаленою черговою змінною. Під ітерацією алгоритму розуміють $(n-1)$ проходів, в результаті чого формується перетворена матриця із видаленою відповідною змінною у всіх рядках. Під час прямого ходу алгоритму виконується $(n-1)$ ітерація. Аналогічним чином здійснюється зворотний хід алгоритму. За один прохід видаляється чергова невідома з рядку, за одну ітерацію – видаляється відповідна невідома з усіх рядків, зворотний хід містить $(n-1)$ ітерацію і реалізується на тих самих $(n+1)$ обчислювальних блоках.

Співпроцесор містить наступні функціональні вузли: *Count_block*[j] – обчислювальні блоки, де $j=\overline{1,(n+1)}$; *Buf_rows* – буферна пам'ять ведених рядків; *Buf_master* – буфер провідного рядка; *Form_const* – блок формування константи; *Count_block* – блок лічильників; *Form_adress* – формувач адреси; *Reg_adress* – регістр покажчик адреси; *Sort_rows* – блок сортування та вибору провідного рядку, *Det_block* – блок обчислення визначника, .

Узагальнена структурна схема пристрою представлена на рис. 2. Головний алгоритм обчислень – на рис. 3.

Перетворення матриці відбувається за наступних етапів:

I. *Прямий хід алгоритму*, під час якого виконуються відповідні ітерації, які складаються з проходів алгоритму. Результатом є верхня трикутна матриця.

I.*а*. *Ітерація*. На початку ітерації виконується процедура пошуку провідного рядку, який завантажується в буфер провідного рядку. Ведені рядки у порядку від $(k+1)$-го до n-го завантажують ся в буферну пам'ять ведених рядків. Результат кожної ітерації зберігається в основній пам'яті.

I.*б*. *Прохід алгоритму*, у вихідному стані із буферної пам'яті рядків в обчислювальні блоки завантажується черговий для обробки рядок i, провідний рядок знаходиться в буфері провідного рядку. Результат проходу завантажується в буферну пам'ять.

I.*б.а*. *Обчислення константи*. На кожному проході p (де – $p=\overline{k,(n-1)}$) видаляється черговий елемент рядка i, для чого здійснюється обчислення константи m_p: $m_p = a_{ik}/a_{kk}$ (де a_{kk} – провідний елемент, a_{ik} – відповідний елемент поточного рядку). Константа після обчислення подається на входи всіх обчислювальних блоків конвеєру для подальшого перетворення рядка.

I.*а.б*. *Перетворення рядка*, під час чого елементи провідного рядка помножуються на косфіцієнт m_p, та віднімаються від перетворюваного рядку. Прохід алгоритму виконується на конвеєрі обчислювальних блоків *Count_block*[j], результат зберігається в буферній пам'яті.

II. *Обчислення визначника*, для обчислення визначника користуємось властивістю: визначник трикутної матриці дорівнює добутку діагональних елементів. Якщо визначник дорівнює нулю, рівняння коренів не мають, але мають нескінченну кількість рішень.

III. *Зворотній хід алгоритму*. Під час зворотного ходу алгоритму здійснюються аналогічні прямому ходу дії, але лічильники рядків і стовпців підраховують в зворотному напрямку, тобто $a_{i,j}$ ... $a_{i,j}$, b_i, де $i = \overline{n,(n-k)}$, $j = \overline{n,1}$, формуючи тим самим такі індекси, що елементи матриці будуть завантажені в буфери і оброблені у зворотному порядку.

IV. *Обчислення коренів* СЛАР.

Наведемо опис структури та принципу функціонування основних блоків обчислювача, структурна схема якого і основний алгоритм функціонування зображені на рис. 2 та рис. 3 відповідно.

Основна пам'ять з адресним доступом (*RAM*), застосовується для зберігання вихідної матриці, перетворених після кожної ітерації фрагментів матриці і остаточного результату обчислення.

Буферна пам'ять ведених рядків (*Buf_rows*) організована за принципом *FIFO* сюди завантажуються фрагменти матриці на початку кожної ітерації (процедура *choose_rows*). Ведені рядки надходять на входи обчислювальних блоків безпосередньо з буферної пам'яті рядків. Елементи матриці послідовно обробляються на конвеєрі і записуються в буферну пам'ять знов, у порядку перший обрахований – перший записаний у пам'ять. Таким чином після чергового проходу алгоритму на виході буферної пам'яті готовий до перетворення перший елемент наступного рядку матриці, а після чергової ітерації – перетворений фрагмент матриці. Наприкінці кожної ітерації вміст буферної пам'яті, зберігається у основній пам'яті (процедура *load_memory*) за відповідними адресами рядок-стовпець, при цьому завантаження рядків розпочинається з рядку $i = (k+1)$, де k – номер ітерації, і зі стовпця $j = (k+1)$. Елементи ведених рядків відповідно матимуть індекси a_{ij}, де $i = \overline{k+1,n}$, $j = \overline{k,(n+1)}$.

Стовпці матриці, де $j < k$ та рядки, де $i < k+1$ в буфер ведених рядків не завантажуються та в обчислювачі не обробляються. Ці рядки вважаються вже перетвореними.

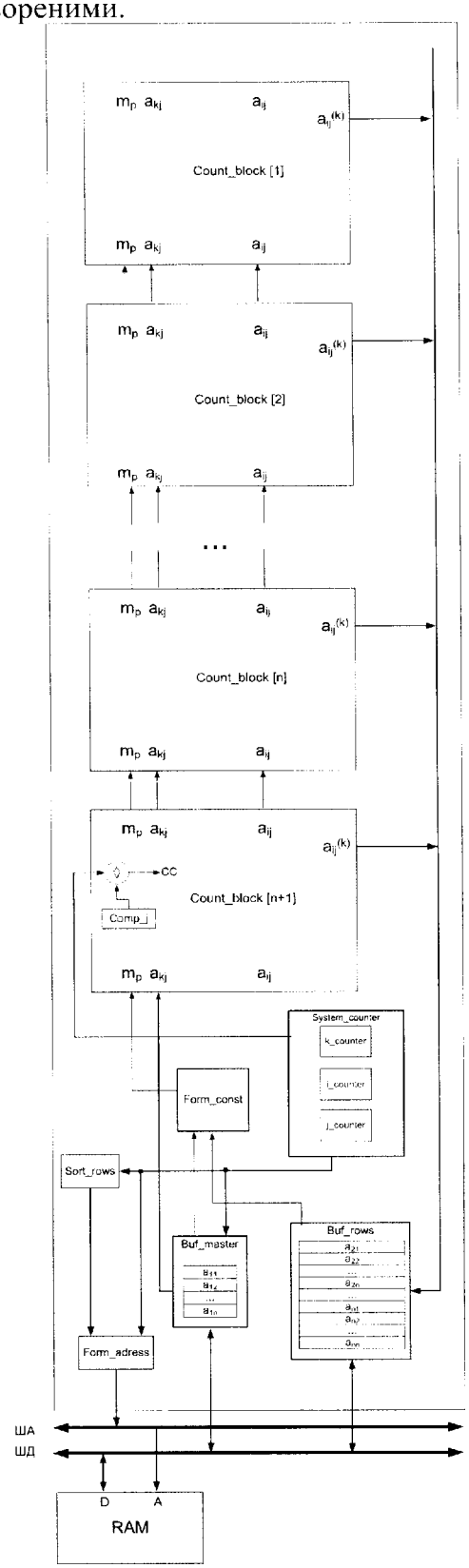

Рис. 2. Структурна схема обчислювача

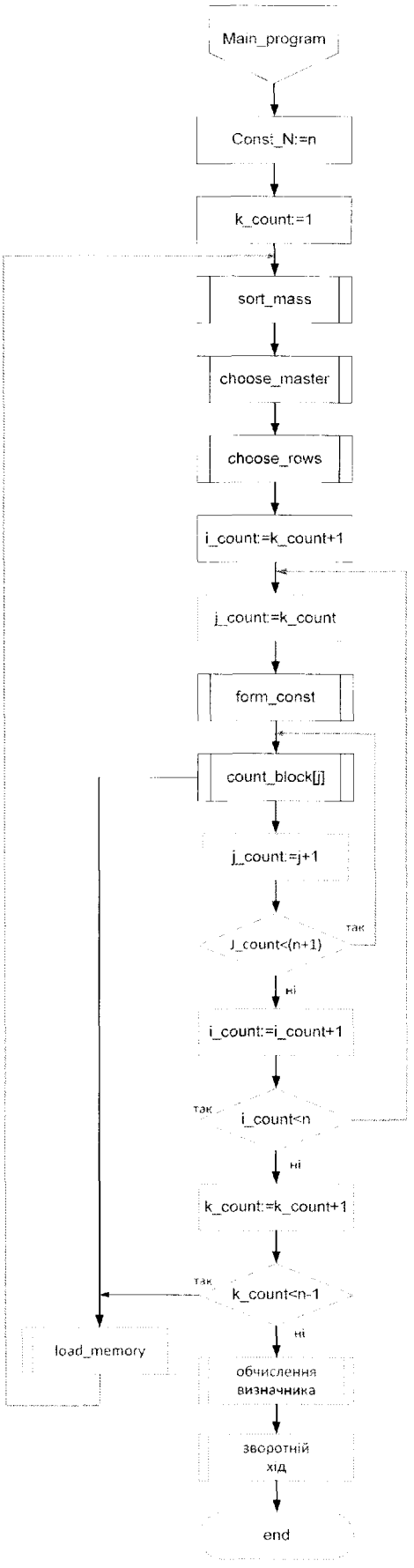

Рис. 3. Головний алгоритм обчислення

Буфер для зберігання провідного рядку (*Buf_master*) застосовується для зберігання провідного рядка на протязі чергової ітерації. Буфер організований у вигляді пам'яті з адресним доступом, кожна комірка якої є відповідним елементом провідного рядку. Вибором необхідного елементу, тобто формуванням адреси у буфері, управляє лічильник *j_count*, який встановлюється у значення *k* на початку кожної ітерації. Таким чином у блоці *Buf_master* знаходиться рядок $i = k$.

Наприкінці чергової ітерації вміст буфера завантажується в основну пам'ять (процедура *load_memory*) у рядок, номер якого дорівнює *k*.

Блок лічильників (*Count_block*). Для підраховування ітерацій і проходів алгоритму до складу системи входять лічильник ітерацій *k_count*. Для підраховування рядків і стовбців також застосовуються лічильники *i_count* ($i\_count = \overline{k+1,n}$) та *j_count* ($j\_count = \overline{k,n}$) відповідно, де *n* – кількість елементів матриці, *k* – номер ітерації ($k = \overline{1,(n-1)}$), *i* – номер рядку, *j* – номер стовпця.

Блок сортування та вибору провідного рядку (*Sort_rows*) містить набір регістрів і схему порівняння. Призначення блоку – сортування першого стовпця ($j = k$) відповідного перетворюваногофрагменту матриці з метою вибору найбільшого за модулем елементу стовпця (процедура *sort_mass*), обраний рядок буде вважатися провідним і завантажуватись у буфер провідного рядку (процедура *choose_master*). Елементи провідного рядка матимуть індекси a_{kj}, де $i = k$, $j = \overline{k,(n+1)}$. Елементи рядку де $j < k$ не розглядаються.

Блок формування константи (*Form_const*). В блоці формування константи виконується обчислення константи m_p для чергового проходу алгоритму (процедура *form_const*). На початку чергової ітерації лічильник *j_count* встановлюється у значення *k*. Аргументами для обчислення константи є перший член провідного рядка a_{kk} (адресний пошук в *Buf_master* за значенням лічильника *j_count*) і відповідний номеру проходу

член веденого рядка a_{ik}. (елемент на виході *Buf_rows*). Під час обчислення константи $m_p = a_{ik}/a_{kk}$ потрібно запобігти діленню на нуль $a_{kk} \neq 0$. Для цього на початку кожної ітерації виконується процедура вибору провідного рядку (*Sort_rows*), в результаті якої рядки матриці сортуються для отримання рядку в якому $a_{kk} = \max$, який вважатиметься провідним. Зважаючи на певні обмеження даної версії системи стосовно обробки чисел з плаваючою комою (на даному етапі досліджень співпроцесор обробляє лише мантиси чисел) процедура вибору провідного рядку завжди забезпечує умови $|a_{kk}| \geq |a_{ki}|$ і $(-1 \leq m_p \leq 1)$, тому для подальших обчислень в обчислювальні блоки передається лише мантиса константи зі знаком у доповнювальному коді. Блок формування константи складається з двох регістрів, в яких зберігаються аргументи для обчислення константи, і схеми ділення. На виході блоку отримуємо константу *m*.

Формувач адреси (*Form_adress*) застосовується для формування адреси елементу в основній пам'яті. Аргументами для формування адреси є вміст лічильників *i_count* і *j_count*. За індексами елементів розраховується адреса елементу в пам'яті. Адреса в основній пам'яті розраховується за формулами:

$$A(a_{ij}) = a_0 + j + (i-1) + \Delta_i;$$
$$A(b_i) = a_0 + (j-1) + (i-1) + \Delta_i;$$
$$\Delta_i = (i-1)n.$$

Обчислювальні блоки (*Count_block* [*j*], де $j = \overline{1,(n+1)}$) формують конвеєр довжиною $(n+1)$ ступеней. Структура обчислювального блоку представлена на рис. 4.

Номер обчислювального блоку відповідає номеру елементу рядку, таким чином на кожному блоці має бути обчислений визначений елемент рядку. Відповідність номеру блоку і елементу рядку визначає схема порівняння *comp_j*, сигнал з виходу якого управляє вхідними мультиплексорами. При співпаданні номера блоку і вмісту лічильника *j_count* виконуються відповідні обчислення, інакше, вхідні значення a_{kj} и $a_{(k-1)j}$, передаються в вихідні буфери *buf_reg*(a_{kj}) і *buf_reg*(a_{ij}) відповідно для передачі на наступний *count_block*. Також до складу блоку входять три регістри для збереження тимчасових результатів обчислення, помножувач і суматор.

Конвеєр має динамічну довжину залежно від довжини оброблюваного рядку. На першій ступені конвеєра обробляється n-й елемент рядка, перші елементи просуваються на наступні ступені конвеєра, таким чином перший елемент рядку буде оброблений на $(n-k)$-му такті та завантажений у *Buf_rows*.

Часова оцінка виконання розглянутих процедур полягає в завантаженні вихідних значень у регістри та сумарного часу встановлення лічильників – 3τ; вибору максимального елементу – $(n-k-1)\tau$; запису масиву елементів в буфер – $(n-k-1)(n-k)\tau$; де τ – один такт роботи системи. Загалом час потрібний на формування вхідного масиву для виконання однієї ітерації алгоритму дорівнює $((n-k)^2+2)\tau$, де k – номер ітерації.

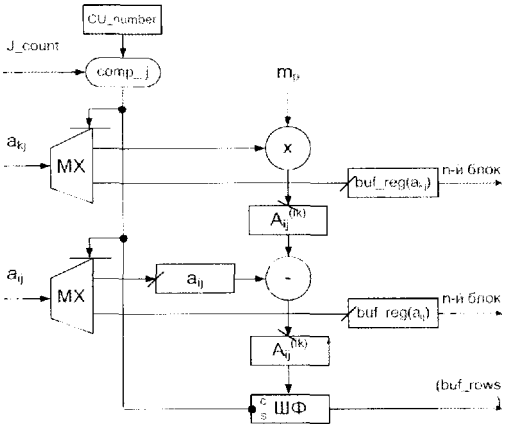

Рис. 4. Структура обчислювального блоку

Моделювання обчислювального блоку

На рис. 5 представлена часова діаграма роботи обчислювального блоку змодельованого в САПР *Quartus* II. Обчислювальний блок синтезований на базі бібліотечних модулів САПР *Quartus* II *Altera*. Час обчислення одного елементу

матриці дорівнює 7,2 нс. Модель виконана для обчислення СЛАР від трьох змінних. Сумарні витрати часу на обчислення системи рівнянь від трьох змінних.

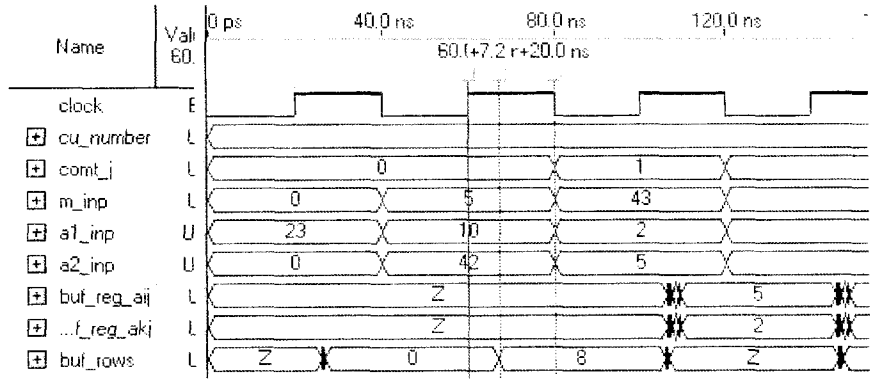

Рис. 5. Часова діаграма роботи обчислювального блоку

Теоретичні оцінки часу обчислення

Обчислення відбувається за $2(n-1)$ ітерацій алгоритму. Кожна ітерація складається з $(n-k)$ проходів, де k – номер ітерації. Нехай час на підготовку до кожної ітерації, який складається з сортування рядків t_{sort}, завантаження рядків у буфери пам'яті $(t_{master} + t_{rows})_{in}$, вивантаження рядків в основну пам'ять після здійснення перетворень $(t_{master} + t_{rows})_{out}$, дорівнює t_k. Час на виконання одного проходу алгоритму, який складається з часу обчислення константи t_m та часу виконання перетворень $t_{count} = (n+2-k)\tau$, дорівнює t_p. Тоді час виконання однієї ітерації алгоритму дорівнює $(n-k)t_k t_p$. Загальний час обчислень

$$t_{salr} = 2(n-1)(n+2-k)t_k t_p + \\ + t_{det} + t_{res},$$

де t_{det} та t_{res} час обчислення визначника матриці та час обчислення результату відповідно. Для моделі з трьома невідомими за теоретичними розрахунками, якщо прийняти час виконання операції вводу/виводу в пам'ять одного слова даних за 2τ, час виконання арифметичних операцій на бібліотечних модулях, згідно документації, за 2τ, отримаємо сумарний час обчислення коренів рівнянь $t_{salr} = 282\tau$. Якщо припустити, що такт системи дорівнює часу обчислень на одному обчислювальному блоці отримаємо $t_{salr} = 2030\text{нс} = 0.2030 \times 10^{-3}\text{с}$= загальний час обчислення.

Моделювання алгоритму на потоковому обчислювачі

Під час дослідження була розроблена програма для моделювання алгоритму обчислення СЛАР від трьох невідомих на потоковому обчислювачі [11]. Під час обчислювання реалізовано 55 команд. З них 8 – ділення, 15 – множення, 11 – віднімання, 13 команд розгалуження, 5 повторювачів входу і 3 команди виводу результатів. Обчислення відбувається на восьми процесорних блоках за 448 тактів. Приймаючи такт системи за 7,2 нс, отримаємо результат 3225.6 нс, що в 1.56 разів більш ніж апаратна реалізація алгоритму.

Висновки

Алгоритм розв'язку нечітких СЛАР у тензорному базисі дає змогу представити початкову задачу у вигляді однієї задачі з блочними параметрами, що істотно спрощує процедуру обчислень, на підставі цього авторами роботи [8] запропонована методологія розв'язку СЛАР з нечіткими змінними, що полягає в формуванні блочної чіткої СЛАР, розв'язок якої виконується стандартними методами. Для реалізації даного методу в статті запропонована обчислювальна система на базі ПЛІС.

Запропоновані обчислювальні засоби належать до нестандартної вузькоспеціалізованої цифрової апаратури, для реалізації якої на сьогодні використовують

технології ПЛІС. Програмовані логічні інтегральні схеми на ряду з порівняними технічними характеристиками з замовленими інтегральними мікросхемами, мають ряд переваг: розробка цифрових пристроїв будь-якої складності, аж до багатопроцесорних обчислювальних систем; висока швидкодія і низьке енергоспоживання; простота проектування, налагоджування та вдосконалювання; низькі затрати на виробництво; можливість динамічної реконфігурації архітектури відповідно вимогам вирішуваних задач [5].

Засоби для розв'язання блочних чітких СЛАР на базі ПЛІС дозволять вдосконалити логічну структуру розподіленої обчислювальної мережі та, за рахунок високої швидкодії елементної бази, підвищити продуктивність нечітких систем управління.

До складу обчислювальної системи входить конвеєрний співпроцесор для рішення чітких СЛАР методом Гауса. Результати моделювання роботи співпроцесора показали зменшення часу обчислення в 1,6 разів у порівнянні з відомою мультипроцесорною потоковою системою [11].

Перелік літератури

1. Тарасик В.П. Математическое моделирование технических систем: Учебник для вузов / В.П. Тарасик. – Мн. : ДизайнПРО, 2004. – 640 с.

2. Заде Л.А. Понятие лингвистической переменной и его применение к принятию приближенных решений / Л.А. Заде. – М. : Мир, 1976. – 165 с.

3. Zade L.A. Fuzzy logic, neural networks and soft computing / L.A. Zade // Communications of the ACM. – 1994. – Vol.37, № 3. – P. 77- 84.

4. Аверкин А.Н. Мягкие вычисления и измерения / А.Н. Аверкин, С.В. Прокопчина // Интеллектуальные системы. – 1997 – Т. 2, № 1-4. – С. 94-113.

5. Клименко І.А. Тенденції застосування сучасної елементної бази для побудови високопродуктивних обчислювальних систем // Проблеми інформатизації та управління: Зб.наук.пр. – К. : Вид-во нац. авіац. ун-ту «НАУ-друк», 2010.– Вип. 1(29). – С 90-103.

6. Минаев Ю.Н. Нечеткая математика на основе тензорных моделей неопределенности. Часть 2 – нечеткая математика в тензорном базисе / Ю.Н. Минаев, О.Ю. Филимонова // Электронное моделирование. – К. : ИПМЭ НАН Украины, 2008. – № 2, Т.30. – С. 4-21.

7. Крон Г. Тензорный анализ сетей / Г. Крон. – М. : Сов. радио, 1978. – 720 с.

8. Мінаєв Ю.М. Розв'язок нечітких систем лінійних алгебричних рівнянь / Ю.М. Мінаєв, О.Ю. Філімонова, Ю.І. Мінаєва, Є.О. Гончарова // Наукоємні технології: Науковий журнал. – К. : НАУ, 2009. – С. 45-67.

9. Жуков І.А. Модуль оперативної пам'яті для RISK процесора на ПЛІС / І.А. Жуков, І.А. Клименко // Проблеми інформатизації та управління: Зб.наук.пр. – К. : НАУ, 2009. – Вип.2 (28). – С. 50-54.

10. Dorta T. Reconfigurable Multiprocessor Systems: A Review / T. Dorta, J. Jiménez, J. L. Martín, U.Bidarte, A.Astarloa // International Journal of Reconfigurable Computing [Special issue on selected papers from ReconFig International conference on reconfigurable computing and FPGAs (ReconFig 2009)] – 2010. – Volume 2010. – 11 p.

11. Жабин В.И. Архитектура вычислительных систем реального времени / В.И. Жабин. – К. : ВЕК +, 2003. – 176 с.

12. Cyclone II Device Handbook [Електронний ресурс]. – Altera Corporation, 2008. – Режим доступу: http://www.altera.com/devices/fpga/cyclone2/cy2-index.jsp.