УДК 004.942

Я. О. Каліновський[1], Ю. Є. Боярінова[1],
В. П. Тарасенко[2], Я. М. Клятченко[2]
[1]Інститут проблем реєстрації інформації НАН України
вул. М. Шпака, 2, 03113 Київ, Україна
[2]Національний технічний університет України «КПІ»
Проспект Перемоги, 37, 03056 Київ, Україна

Алгоритмічно-структурні та схемотехнічні особливості апаратної реалізації операцій з кватерніонами у функціональних процесорах

Показано принципову можливість створення багатопроцесорних систем на основі програмованих інтегральних мікросхем з уже реально досягнутим рівнем інтеграції для прямої апаратної реалізації повної системи операцій над кватерніонами.

Ключові слова: *гіперкомплексні числові системи, кватерніон, операція множення, функціональний процесор, багатопроцесорна обробка інформації, ПЛІС, SIMD-архітектура.*

Вступ

У тематичному відношенні представлена стаття є розвитком роботи [1], де отримано оцінки часу програмного виконання операції множення двох чотирикомпонентних гіперкомплексних чисел (кватерніонів) [2] і запропоновано концепцію апаратних структур для прискорення цієї операції на основі векторних обчислень одним процесорним елементом (ПЕ) за рахунок введення групової операції з автономним блоком управління, який забезпечує певну структуризацію даних у пам'яті [3].

Постановка задачі

Метою даної роботи є реалізація іншої концепції стосовно операцій над кватерніонами, що дозволяє повністю використати всі їхні потенційні можливості для досягнення максимальної продуктивності обчислень у цій гіперкомплексній числовій системі (ГЧС) — на основі структур із багатопроцесорною організацією обчислень на базі функціонально-орієнтованих процесорів. Такі дослідження мають вирішальне значення для створення комп'ютерних засобів з прямою апаратною реалізацією операцій над гіперкомплексними числами [1, 2].

Структурні особливості реалізації на програмованих логічних інтегральних середовищах

Як показано в [1] одним із способів підвищення продуктивності операційних засобів для обробки гіперкомплексних чисел є застосування SIMD-процесорів. Комп'ютерні системи з архітектурою SIMD (Single Instruction Multiple Data) [3] є ефективними засобами обробки даних з багатократною їх структуризацією в процесі обчислень. SIMD-системи містять набір однакових процесорів, але всі вони працюють з різними даними, розміщеними в пам'яті. Тому спершу коротко розглянемо основні аспекти створення операційних засобів з SIMD-архітектурою, що орієнтовані на використання сучасних ПЛІС (програмованих логічних інтегральних середовищ) типу FPGA фірми Xilinx та основні компоненти цієї архітектури.

У загальному випадку структура такої SIMD-системи відповідає схемі на рис. 1, а її основними компонентами є контролер (блок управління), деяке число ПЕ з пам'яттю та мережа міжз'єднань. Для SIMD-систем з певною кількістю ПЕ, які реалізовані на ПЛІС, головними факторами, що впливають на їхню конфігурацію є модель доступу до оперативної пам'яті та її тип. Для структури на рис. 1 може бути ефективною модель розподіленої пам'яті, що дозволяє масштабувати систему за кількістю ПЕ та за топологією мережі міжз'єднань. У такому разі в систему досить нескладно додавати або видаляти з неї ПЕ без втручання в пам'ять. У розподіленій пам'яті багатопроцесорної системи кожен ПЕ має свій власний адресний простір. Обмін інформацією виконується за принципом «точка-точка». Кожен ПЕ з блоком пам'яті можна розглядати як вузол багатопроцесорної системи. Недоліком розподіленої пам'яті є збільшення числа інструкцій, які необхідні для переміщення даних між ПЕ.

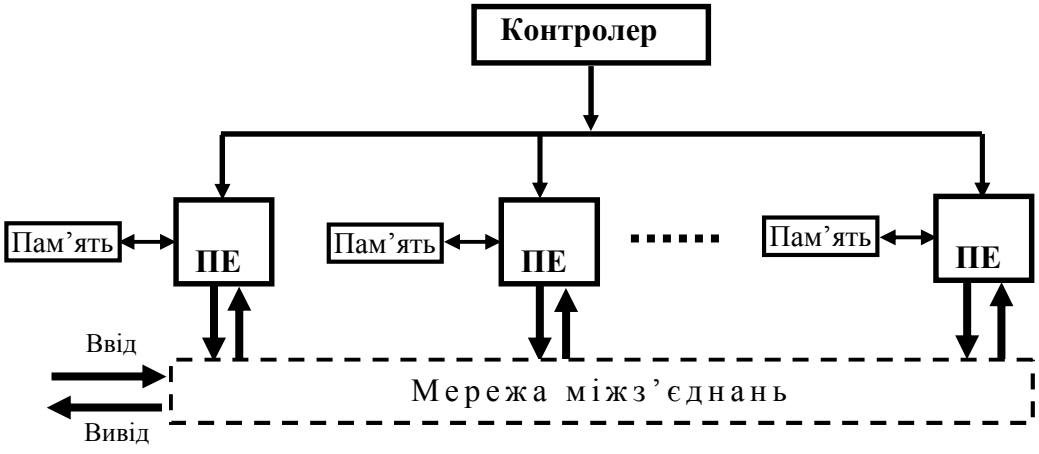

Рис. 1. Базові елементи SIMD-системи

Іншим фактором, що суттєво впливає на конфігурацію обчислювальної SIMD-системи, є вибір схемотехнічної реалізації оперативної пам'яті ПЕ, тобто її тип. Сумарний об'єм пам'яті, що реалізується безпосередньо на ПЛІС, обмежується фізичними ресурсами мікросхеми і залежить від сімейства та типу кристалу ПЛІС. Звичайно у складі ПЛІС є такі типи пам'яті: тригерна, розподілена, блочна, зовнішня.

Тригерна пам'ять логічних комірок ПЛІС є найшвидшим і найбільш дефіцитним програмованим ресурсом. Саме завдяки використанню пам'яті цього типу можна досягти високих тактових частот функціонування системи на ПЛІС, до того ж усі регістрові пристрої (у тому числі й регістри процесора) реалізувати на них можна без особливих труднощів. Але через наявність тільки одного тригера в секції ПЛІС, використання пам'яті такого типу є марнотратним.

Розподілена пам'ять — пам'ять логічних комірок — має досить високу швидкодію і, в принципі, може об'єднуватися для утворення більш великих блоків. Однак на практиці для з'єднання дрібних фрагментів оперативної пам'яті потрібно багато трасувальних ліній, трасування стає заплутаним і, як наслідок, у ПЛІС зростають затримки сигналів. «Зручною» структурою на основі такої пам'яті є стек.

Блочна пам'ять сучасних ПЛІС, яку складають виділені блоки по 18 або 32 Кбіт, залежно від серії ПЛІС, може нарощуватися до об'ємів 67 680 Кбіт [4]. Завдяки великій кількості блоків, зручному інтерфейсу та високій швидкості обміну даними через них така пам'ять може застосовуватись як пам'ять для ПЕ тих типів, що описані в наступних розділах статті.

Процесорні елементи в архітектурі SIMD на ПЛІС керуються контролером. При виборі базового варіанта для контролера слід звернути увагу на такі його структурні характеристики як швидкість обчислення, зручність створення, тип і зручність використання внутрішньосистемної шини. Для ПЛІС фірми Xilinx звичайно можливі три варіанти організації контролера: hard-процесор (апаратний процесор) PowerPC, soft-процесор (програмне процесорне ядро) Microblaze™ [5] або реалізація абсолютно нового контролера «з нуля» (останній варіант далі не розглядається внаслідок складності та його високої вартості).

Апаратні процесори — мікропроцесорні ядра, які виконані у вигляді інтегрованих апаратних блоків ПЛІС. Головна перевага цих мікропроцесорних ядер (на прикладі сімейства PowerPC) — можливість функціонування з високими тактовими частотами і як наслідок — більш висока швидкодія порівняно з програмними мікропроцесорними ядрами. Недоліками апаратних ядер є невелика кількість кристалів, у яких вони застосовуються, та їхня висока вартість.

Hard-процесор PowerPC мав застосування переважно в ПЛІС ранніх серій, тому що для його розміщення не використовувались основні логічні ресурси ПЛІС. У сучасних же ПЛІС як основний процесор для контролерів найчастіше пропонується soft-процесор Microblaze™, оскільки його можна реалізувати на більшій кількості моделей ПЛІС Xilinx. Проблема економії ресурсів ПЛІС, що їх займає soft-процесор, і які б могли бути використані для реалізації ПЕ, на сьогоднішній день не актуальна. Наприклад, реалізація процесора Microblaze™ може займати залежно від параметрів настроювання приблизно від 3 % логічних ресурсів Spartan-6 і до 0,5 % — для кристалів Virtex 7-ї серії [6]. Використання Microblaze ™ в обчислювальній системі також обумовлює просту інтеграцію в проект однієї або декількох сучасних системних шин (інтерфейсів), що пропонуються фірмою Xilinx: LMB (Local Memory Bus), OPB (On-chip Peripheral Bus), PLB (Processor Local Bus), AXI (Advanced eXtensible Interface) або FSL (Fast Simplex Link) [7]. Також виробник пропонує готовий IP-модуль контролера на базі цього soft-процесора.

Від способу з'єднання операційних компонент в архітектурі SIMD залежить досягнення максимальної продуктивності і можливість реконфігурації мережі міжз'єднань. Операційні компоненти мають бути реалізовані таким чином, щоб кількість та довжина ліній зв'язку були зведені до мінімуму. Структура мережі міжз'єднань показана на рис. 2. Залежно від виду топології, яка використовується, комутуючі елементи маршрутизують міжпроцесорні потоки, визначають способи з'єднання ПЕ і т.д. Для управління цими елементами служить контролер мережі міжз'єднань. Залежно від інструкцій, що надходять шиною керування, він формує керуючі сигнали для кожного комутуючого елемента в системі, блока вводу/виводу та інших блоків. Блок вводу/виводу забезпечує взаємодію між елементами маршрутизації та периферійним пристроями. Як периферійний пристрій може використовуватися порт Ethernet, послідовний порт, зовнішня флеш-пам'ять або оперативна пам'ять. Структура блока вводу/виводу не залежить від виду топології мережі.

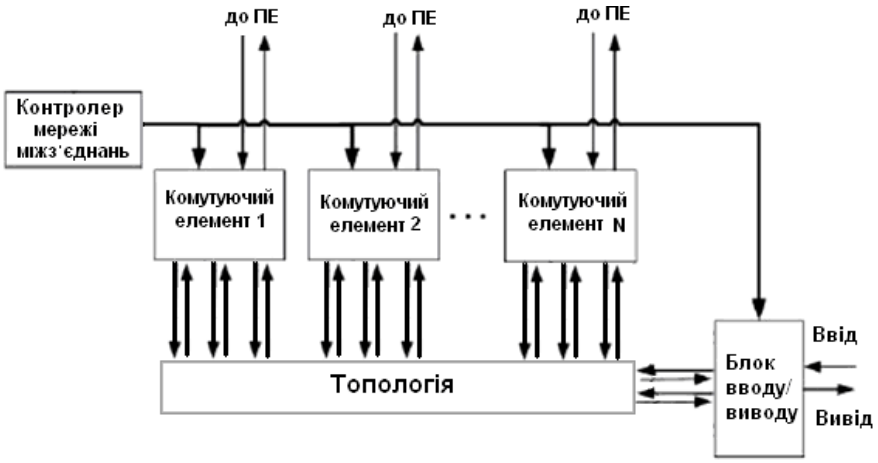

Рис. 2. Мережа міжз'єднань

Найважливішим чинником, що впливає на продуктивність багатопроцесорної системи, є спосіб з'єднання ПЕ. Процесорні елементи повинні «співпрацювати» та обмінюватися даними мережею. Можна виділити два різних класи топологій зв'язку: статична та динамічна. В статичній мережі вузли з'єднані один з іншим безпосередньо, і ці з'єднання не можуть бути змінені. На відміну від статичної мережі, динамічна топологія мережі може змінюватися під час функціонування обчислювальної системи, наприклад, шляхом реконфігурування ПЛІС.

Розглянемо основні типи топологій, що можуть утворювати ПЕ та міжпроцесорні лінії зв'язку в багатопроцесорній системі на базі ПЛІС Xilinx із шиною FSL, не беручи в розрахунок зв'язки, що потрібні для сполучення з контролером і деякими іншими блоками.

Топологія «повний граф» відповідає повністю зв'язаній мережі, в якій кожен вузол (ПЕ) зв'язаний з кожним іншим вузлом. Ця топологія зменшує час пересилання даних мережею, коли дані пересилаються безпосередньо від відправника до одержувача, але її основним недоліком є надзвичайно швидкий ріст кількості зв'язків, що зростає при збільшенні кількості вузлів (ПЕ). У випадку використан-

ня інтерфейсу FSL кількість ПЕ може дорівнювати 9, оскільки даний інтерфейс передбачає використання 8-х двонаправлених FSL-сполучень.

Кільцева топологія являє собою мережу, в якій кожен вузол (ПЕ) в мережі зв'язаний з наступним і попереднім вузлами в мережі, утворюючи кільце. Дані передаються від вузла до вузла, поки вони не досягнуть вузла призначення. При використанні інтерфейсу FSL не існує обмежень за кількістю ПЕ в системі тому, що кожний процесор ПЕ буде використовувати всього 2 інтерфейси FSL. Основним недоліком цієї топології є те, що пересилка даних між двома вузлами (ПЕ) може займати багато часу та використовувати ресурси інших транзитних процесорів.

Топологія типу «зірка» являє собою мережу, в якій кожен вузол зв'язаний з центральним вузлом. Слабким місцем цієї топології є те, що вихід з ладу центрального вузла призводить до непрацездатності всієї мережі. Інший недолік полягає в тому, що всі комунікації йдуть через центральний вузол, і тому у випадку інтенсивного обміну даними буде мати місце ефект «вузької горловини» в центральному вузлі. За цією топологією на основі шини FSL можна побудувати системи з 1-м керуючим процесором, як центральним вузлом (наприклад Microblaze™), і 8-ма ПЕ, як звичайними вузлами. Функціональні можливості системи можна розширити шляхом кількарівневого об'єднання різних підсистем із зіркоподібною топологією.

Як ПЕ в багатопроцесорній SIMD-системі для операцій з кватерніонами використовуються функціонально-орієнтовані процесори (ФОП), що реалізовані за допомогою програмованих ресурсів ПЛІС. Структура та особливості функціонування таких ПЕ докладно розглядаються в наступних розділах.

Передумови створення функціонально-орієнтованих процесорів для операцій з кватерніонами

Далі значною мірою використовуються основні поняття, визначення і математична символіка з роботи [1]. Крім того, при побудові ФОП будемо виходити з припущення, що в їхньому складі є достатня кількість регістрів для початкового розміщення компонент векторів $\boldsymbol{K}_1 = a_1 + ib_1 + jc_1 + kd_1$ та $\boldsymbol{K}_2 = a_2 + ib_2 + jc_2 + kd_2$ а також для запису чотирьох компонент вектора $\boldsymbol{K}_3 = \boldsymbol{K}_1 \times \boldsymbol{K}_2$, який формується покроково. Позначимо $\boldsymbol{K}_1 = a_1 + ib_1 + jc_1 + kd_1$, $\boldsymbol{K}_2 = a_2 + ib_2 + jc_2 + kd_2$, тоді, як показано в [1], добуток векторів $\boldsymbol{K}_1 \times \boldsymbol{K}_2$ можна записати наступним чином:

$$\begin{aligned} K_1 \times K_2 = K_3 = {} & \left(a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2\right) + i\left(a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2\right) + \\ & + j\left(a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2\right) + k\left(a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2\right). \end{aligned} \quad (1)$$

При множенні на кожному кроці формуються чотири добутки компонент одного вектора на компоненти другого, і виконується додавання (віднімання) цих добутків до їхньої суми для утворення компонент вектора-результату $\boldsymbol{K}_3$, а саме:

$$\left.\begin{aligned} a_3 &= a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2 \\ b_3 &= a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2 \\ c_3 &= a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2 \\ d_3 &= a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2 \end{aligned}\right\}.$$

Для ФОП взагалі [8] структура операційного обладнання розробляється для оптимального (в певному сенсі) виконання списку заданих операцій. Далі будемо вважати, що для «кватерніонного» ФОП цей список складають чотири операції [1, 2]: додавання (віднімання), утворення спряженого кватерніона, множення та обчислення норми кватерніона. Крім того, вважаємо, що перед початком виконання будь-якої з цих операцій компоненти векторів $\boldsymbol{K}_1$ та $\boldsymbol{K}_2$ знаходяться у відповідних регістрах, і ще чотири регістри підготовлені до формування на них компонент вектора $\boldsymbol{K}_3$.

Далі розглядається варіант «горизонтального паралелізму» при обчисленні виразу (1), тобто з паралельним формуванням компонент всіх рядків.

Функціонально-орієнтовані процесори з горизонтальним паралелізмом

Горизонтальний паралелізм означає, що на першому кроці множення отримують добутки a_1a_2, a_1b_2, a_1c_2, a_1d_2 з відповідними знаками, на другому кроці — добутки b_1b_2, b_1a_2, b_1d_2, b_1c_2 і т.д. Кожен з таких добутків формують за допомогою матричного двійкового перемножувала [9, 10]. Це дозволяє отримувати добуток двійкових операндів за час, який приблизно дорівнює циклу звертання до пам'яті.

За вищевказаних припущень у ФОП кожна компонента вектора $\boldsymbol{K}_1$ почергово (на кожному кроці) управляє множенням, а вміст регістрів компонент вектора $\boldsymbol{K}_2$ після кожного кроку треба змінювати з тим, щоб забезпечити передбачену «горизонтальним паралелізмом» послідовність множень. Граф міжкрокових пересилок вмісту регістрів компонент вектора $\boldsymbol{K}_2$ для «горизонтального паралелізму» показаний на рис. 3, де Р-р є ідентифікатором регістра, а позначення 1τ,...,4τ вказують на те, що відповідна пересилка виконується після 1-го, …, 4-го кроку обчислень. Побудова ФОП, що реалізує всі основні операції з кватерніонами, в тому числі множення з «горизонтальним паралелізмом», пояснюється рис. 4. ФОП складається з чотирьох ідентичних модулів, позначених на рис. 4, як $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{d}$ всі зовнішні шини тут призначені для завантаження із пам'яті компонент кватерніонів і відправлення до пам'яті результатів операцій. Кожний модуль (рис. 5) містить матричний перемножувач МПМ1 та мультиплексор МХ2, який комутує операнди або добутки на вхід перетворювача ПР3. Цей перетворювач залежно від сигналів управління С (рис. 4) у процесі передачі виконує перетворення операнда в прямий або доповняльний код. Мультиплексор МХ4 комутує на вхід суматора СМ5 або операнди, що поступають з перетворювача ПР3, або компоненти вектора $\boldsymbol{K}_1$, що зберігаються в регістрах Р-р7 у кожному із модулів. Результат множення (додавання, віднімання і т.п.) записується на регістр Р-р6. Компоненти вектора $\boldsymbol{K}_2$ збе-

рігаються на регістрах Р-р8, а покроковий режим забезпечується за допомогою мультиплексора МХ9 та блока управління БУ10 (рис. 4).

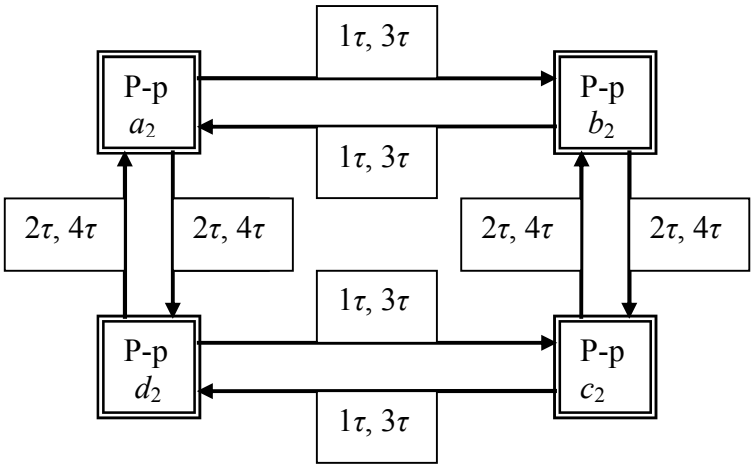

Рис. 3. Граф міжкрокових пересилок для «горизонтального паралелізму»

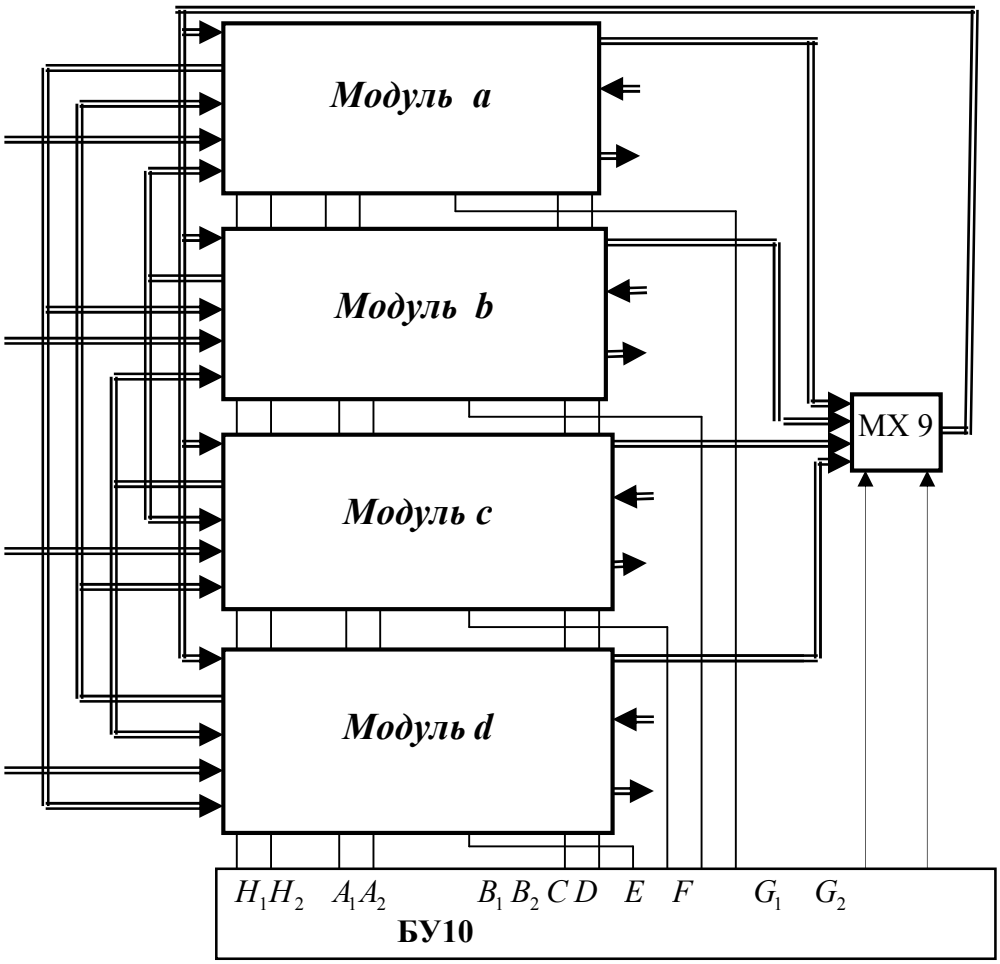

Рис. 4. ФОП з «горизонтальним паралелізмом»

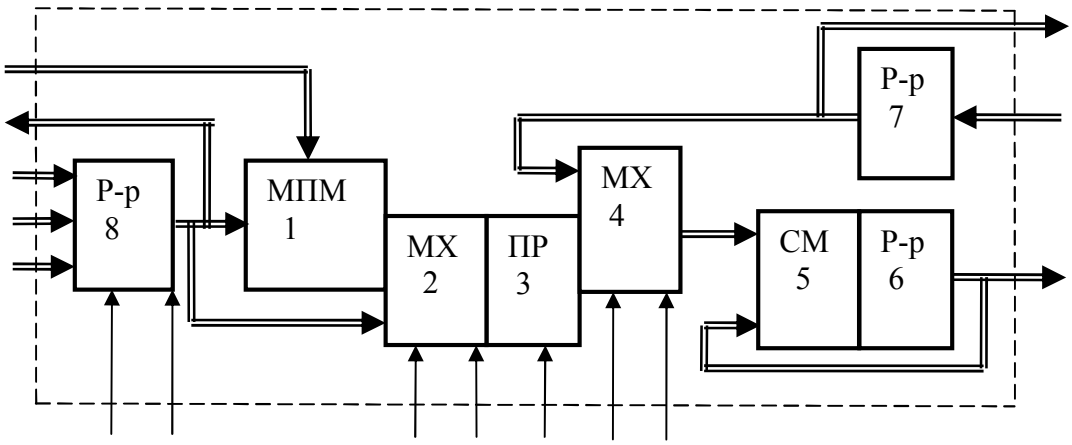

Рис. 5. Модуль ФОП з «горизонтальним паралелізмом»

Розглянемо детальніше призначення кожного із перелічених вузлів ФОП та особливості управління ними в процесі виконання операцій над векторними частинами кватерніонів $\boldsymbol{K}_1$ та $\boldsymbol{K}_2$. МПМ1 по суті є комбінаційними пристроями, які формують добутки двох операндів, поданих у двійковій позиційній однорідній системі числення. Їхня можлива реалізація вже вказана вище, а спеціального управління з боку БУ10 в процесі виконання операцій вони не потребують.

Мультиплексори МХ2 (у кожному з модулів $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{d}$) можуть знаходитись у двох положеннях (напрямках) комутації. Положення 1 відповідає операціям множення і обчислення норми кватерніонів, положення 2 — операціям додавання, віднімання і спряження. Крім того, ці мультиплексори можуть бути встановлені в положення 3, за якого їхні виходи знаходяться в високоомному стані, що відповідає відключенню вихідного каналу мультиплексора від його вхідних каналів. Управління всіма мультиплексорами 2 єдине і здійснюється паралельно по колах A_1 та A_2. При цьому $A_1A_2 = 01$ задає перше положення МХ2; $A_1A_2 = 10$ — друге положення МХ2; $A_1A_2 = 00$ — третє (високоомне) положення. Управління формуванням сигналів A_1 та A_2 здійснює поле коду операції в команді.

Перетворювачі ПР3 необхідні для правильного врахування знаків компонент векторів $\boldsymbol{K}_1$ та $\boldsymbol{K}_2$ і передають операнди, що відповідають компонентам векторів або результатам множення, прямим або додатковим кодом. Управління роботою цих перетворювачів здійснюється наступним чином: а) знаками компонент вектора K_2 при додаванні — відніманні; б) відповідно з виразом (1) при множенні й обчисленні норми; в) інверсіями знаків компонент при обчисленні спряженого кватерніона. Кола управління *C*, *D*, *E*, *F* перетворювачами ПР3 в кожному з модулів $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{d}$ виконуються окремо, а для кодування сигналів у цих колах приймають 0 (передача прямим кодом) та 1 (передача додатковим кодом).

Мультиплексори МХ4, як і МХ2, можуть знаходитись у двох положеннях комутації: 1 — відповідає операціям додавання та віднімання (в цьому положенні здійснюється прийом першого операнда на блоки суматора СМ5 і регістра Р-р6); 2

— відповідає операціям додавання та віднімання (забезпечується прийом другого операнда на блоки суматора СМ5 і регістра Р-р6), множення, обчислення норми і спряженого кватерніона. Третє положення, як і в МХ2, є високоомним і відповідає відключенню МХ4 від вихідного каналу. Управління всіма мультиплексорами МХ4 єдине по колах B_1 і B_2 і здійснюється кодом операції. Кодування управляючих сигналів таке ж, як і для МХ 2.

Мультиплексор МХ9 забезпечує почергове (покрокове) підключення компонент вектора $\boldsymbol{K}_1$ до вхідного каналу матричних перемножувачів. Управління мультиплексором МХ9 здійснюється колами G_1, G_2, сигнали по яких задають одне з чотирьох можливих положень мультиплексора.

Регістри Р-р7 призначені лише для зберігання компонент вектора $\boldsymbol{K}_1$ і не мають будь-яких особливостей, пов'язаних з управлінням ними.

Регістри Р-р8 призначені для зберігання компонент вектора $\boldsymbol{K}_2$. Управління цими регістрами в процесі виконання операцій множення й обчислення норми здійснюється по колах H_1 та H_2. Одиничні сигнали по H_1 та H_2 подаються після закінчення кожного кроку виконання операції і забезпечують міжрегістрові обміни відповідно із графом на рис. 3. Внутрішня логіка регістрів реалізує окремі мікрооперації згідно з таблицею, де РЗІ — режим зберігання інформації.

H_1	H_2	Р-р a_2	Р-р b_2	Р-р c_2	Р-р d_2
0	0	РЗІ	РЗІ	РЗІ	РЗІ
0	1	Передача на Р-р b_2	Передача на Р-р a_2	Передача на Р-р d_2	Передача на Р-р c_2
1	0	Передача на Р-р d_2	Передача на Р-р c_2	Передача на Р-р b_2	Передача на Р-р a_2
1	1	Передача на МПМ	Передача на МПМ	Передача на МПМ	Передача на МПМ

Суматор СМ5 і регістр Р-р6 в кожному модулі призначені для додавання і накопичення суми операндів, один з яких зберігається на Р-р6, а інший поступає з виходу мультиплексора МХ4. Якихось особливостей по управлінню ця пара функціональних вузлів не має.

Розглянемо тепер роботу ФОП у процесі виконання кожної операції. Для додавання векторів $\boldsymbol{K}_1$ та $\boldsymbol{K}_2$ спочатку треба прийняти на Р-р7 компоненти першого вектора, що і здійснюється подачею сигналів управління $B_1B_2 = 01$. Потім за сигналами $A_1A_2 = 10$ і $B_1B_2 = 10$ відбудеться передача на блоки СМ5 і Р-р6 компонент другого вектора (з регістрів Р-р8 через мультиплексор МХ2, перетворювач ПР3 та мультиплексор МХ4). Сигнали H_1H_2 при цьому відкривають кола видачі вмісту регістрів Р-р8, тобто $H_1H_2 = 11$. Решта управляючих сигналів у процесі додавання не змінюються і дорівнюють нулю.

При відніманні вектора $\boldsymbol{K}_1$ від вектора $\boldsymbol{K}_2$, що вже прийнятий на Р-р7 так, як описано вище, крім сигналів $A_1A_2 = 10$ і $B_1B_2 = 10$, необхідно також подавати сиг-

нали $C = D = E = F = 1$. У результаті цього компоненти другого вектора будуть передані на суматор СМ5 додатковим кодом, що і приведе до утворення правильної їхньої різниці в регістрах Р-р6.

Для обчислення спряженого вектора компоненти вихідного вектора, що знаходяться в Р-р8, необхідно передати на регістри Р-р6 зі зміною знака у всіх операндів, які відповідають уявним частинам. Тому такі операнди передаються через перетворювач ПР3, який формує їхні доповняльні коди. Операція буде виконана правильно за одночасної подачі сигналів $A_1A_2 = 10$, $B_1B_2 = 10$, $C = D = E = 1$, $F = 0$, $H_1H_2 = 11$. Відзначимо, що як і при відніманні, компоненти векторів, які мають від'ємні знаки (вказані у відповідних розрядах), будуть подані додатковим кодом.

При множенні векторів $\boldsymbol{K}_1$ та $\boldsymbol{K}_2$ з урахуванням «горизонтального паралелізму» на першому кроці мають бути отримані компоненти першого стовпчика виразу (1), для чого необхідно відтворити наступний набір управляючих сигналів: $A_1A_2 = 01$, $B_1B_2 = 10$, $C = D = E = F = 0$, $G_1G_2 = 00$, $H_1H_2 = 11$. Після цього за сигналами $H_1H_2 = 01$ слід виконати міжрегістрові обміни (рис. 1 і 2) і далі приступити до формування компонент другого стовпчика виразу (1). Відповідний набір управляючих сигналів має вигляд: $A_1A_2 = 01$, $B_1B_2 = 10$, $C = 0$, $D = 1$, $E = 0$, $F = 1$, $G_1G_2 = 01$, $H_1H_2 = 11$. Наступне оновлення вмісту регістрів Р-р8 відбудеться за сигналами $H_1H_2 = 10$. Компоненти третього стовпчика виразу (1) будуть отримані за сигналами $A_1A_2 = 01$, $B_1B_2 = 10$, $C = 1$, $D = 0$, $E = 0$, $F = 1$, $G_1G_2 = 10$, $H_1H_2 = 11$. Черговий міжрегістровий обмін — за сигналами $H_1H_2 = 01$, а компоненти четвертого стовпчика — за сигналами $A_1A_2 = 01$, $B_1B_2 = 10$, $C = D = 0$, $E = F = 1$, $G_1G_2 = 11$, $H_1H_2 = 11$. Останній міжрегістровий обмін (за сигналами $H_1H_2 = 10$) відновлює вихідний стан регістрів Р-р8. В узагальненому вигляді структура сигналів управління ФОП при реалізації перелічених вище операцій показана на циклограмі (рис. 6). Зауважимо, що час обчислення норми внаслідок «горизонтального паралелізму» тут не може бути меншим за час множення, хоча, по суті, при цьому функціонує лише модуль a (рис. 4). Обчислення норми виконується так само, як і множення, але початкове заповнення регістрів має відповідати спряженим кватерніонам. Слід лише враховувати ту обставину, що компоненти спряженого вектора, які мають від'ємний знак і подані додатковим кодом, слід перетворити і записати їх у прямому коді.

Час виконання кожної операції, як видно з рис. 6, складає: додавання (віднімання) — 2 такти, спряження — 1 такт, множення і обчислення норми — 4 кроки по 2 такти кожен. Якщо вважати, що тривалість такту того ж порядку, що і час t_0 звернення до пам'яті, то $t_{дод} \approx t_{від} \approx 4t_0$, $t_{спр} \approx 2t_0$, $t_{множ} \approx t_{норм} \approx 16t_0$. Отже, ФОП, який розглядається, підвищує швидкість виконання операції множення кватерніонів у 3,5…4 рази порівняно з кращими операційними структурами, що використовують блоки автономного управління [1], та в $n + 8$ разів порівняно з програмним способом множення [1, 2] (n — розрядність компонент векторів $\boldsymbol{K}_1$ та $\boldsymbol{K}_2$).

Дещо прискорити операції додавання (віднімання) за рахунок виключення підготовчого такту по заповненню регістрів Р-р6 можна шляхом внесення у ФОП незначних структурних змін, суть яких показана на рис. 7. Ці зміни відповідають перемиканню каналів, через які операнди подаються на суматори СМ5.

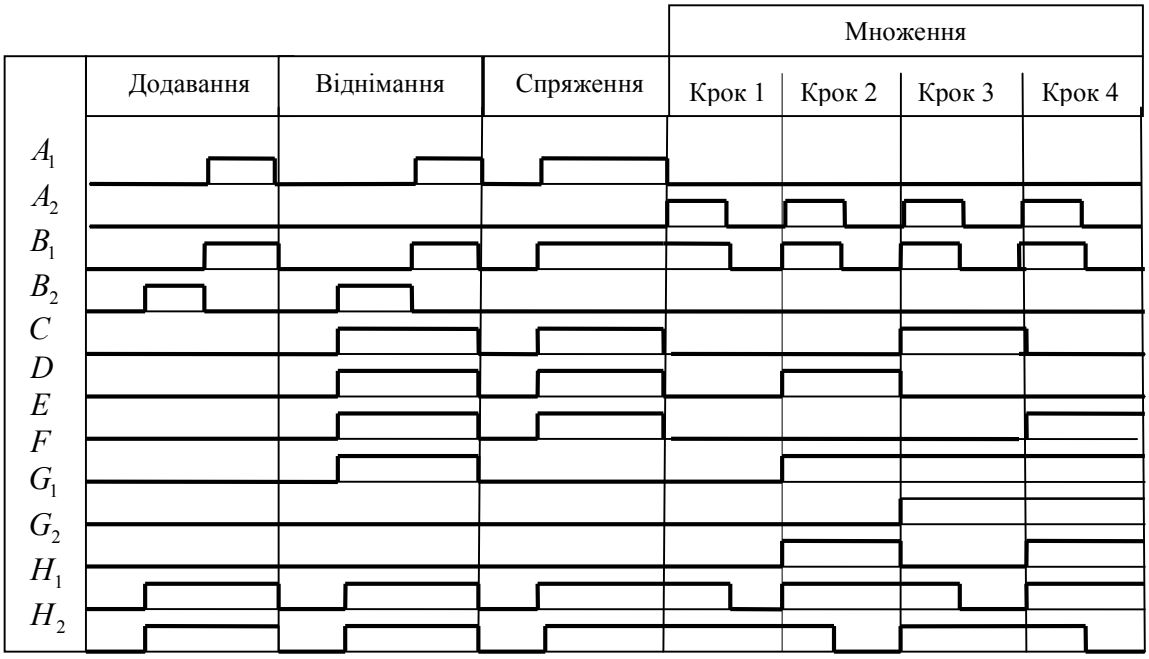

Рис. 6. Циклограма сигналів управління ФОП

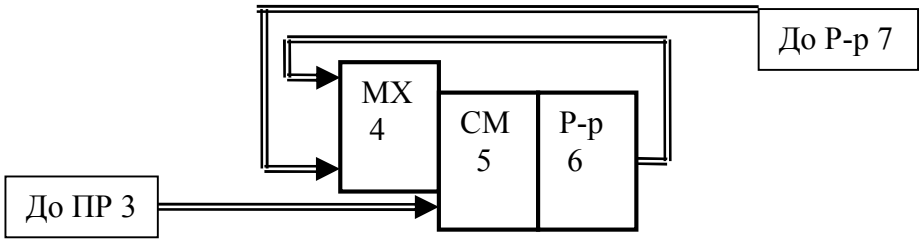

Рис. 7. Структурна зміна в ФОП для забезпечення підготовчого такту

Управління мультиплексором МХ4 у такому випадку здійснюється наступним чином: 1-е положення комутації — операції множення і обчислення норми; 2-е положення комутації — операції додавання (віднімання) та спряження.

Висновки

Вищевикладене показує принципову можливість створення багатопроцесорних систем на основі програмованих інтегральних мікросхем з реально вже досягнутим рівнем інтеграції для прямої апаратної реалізації повної системи операцій над кватерніонами.

Концепція багатопроцесорної обробки інформації на основі функціонально-орієнтованих процесорів на відміну від концепції однопроцесорної обробки стосовно організації виконання основних операцій над кватерніонами, реалізована шляхом використання сучасних програмованих інтегральних мікросхем, дозволяє досягти показників продуктивності, які не залежать від довжини окремих двійкових компонент гіперкомплексних чисел та є величинами одного порядку для різних операцій.

1. *Алгоритмічно-структурні* та схемотехнічні особливості апаратної реалізації множення кватерніонів як операції з автономним управлінням / Я.М. Клятченко, В.П. Тарасенко, Я.О. Каліновський, Ю.Є. Боярінова // Наукові записки Українського науково-дослідного Інституту зв'язку. — Вип.1 (21). — 2012. — С. 15–25.

2. *Синьков М.В.* Конечномерные гиперкомплексные числовые системы. Основы теории. Применения / М.В. Синьков, Ю.Е. Бояринова, Я.А. Калиновский. — К.: Изд-во ИПРИ НАНУ, 2010, — 389 с.

3. *Самофалов К.Г.* Структуры и организация функционирования ЭВМ и систем / К.Г. Самофалов, Г.М. Луцкий — К.: «Вища школа», 1978 — 392 с.

4. *Virtex-7* FPGA Family [Електронний ресурс]. — Режим доступу: http://www.xilinx.com/ publications/prod_mktg/Virtex7-Product-Table.pdf

5. *MicroBlaze* Soft Processor Core [Електронний ресурс]. — Режим доступу: http://www.xilinx.com/ tools/microblaze.htm

6. *All Programmable* FPGAs [Електронний ресурс]. — Режим доступу: http://www.xilinx.com/ products/silicon-devices/fpga/index.htm

7. *LogiCORE IP* Fast Simplex Link (FSL) V20 Bus (v2.11c) [Електронний ресурс]. — Режим доступу: http://www.xilinx.com/support/documentation/ip_documentation/fsl_v20.pdf

8. *Функционально-ориентированные* процессоры / [Водяхо А.И., Смирнов В.Б., Плюснин В.У., Пузанков Д.В.]: под ред. В.Б. Смолова. — Л.: Машиностроение, 1988. — 224 с.

9. *Компьютерная* схемотехника (краткий курс) / [Процюк Р.О., Корнейчук В.И., Кузьменко П.В., Тарасенко В.П.]. — К.: Изд-во «Корнейчук», 2006. — 433 с.

10. *Кривуля Г.Ф.* Схемотехника / Г.Ф. Кривуля, В.М. Рябенький, В.С. Буряк. — Харьков: Изд-во СМТ, 2007. — 250 с.

Надійшла до редакції 24.04.2013