УДК 681.3

А.О. Мельник, А. М. Сало
Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин

ОРГАНІЗАЦІЯ РЕГІСТРОВИХ ФАЙЛІВ ПРОГРАМОВАНИХ ПРОЦЕСОРІВ

Проаналізовано відомі структури регістрових файлів програмованих процесорів. Запропоновано структуру регістрового файла на базі черги з програмованою затримкою. Основні типи регістрових файлів реалізовано на базі ПЛІС сімейства VirtexE фірми Xilinx та проведено їх порівняльний аналіз.

Register file known structures analysis for programmable processors has been carried out. Queue based register structure with programmable latency had been proposed. Main register file types had been developed on the base of Xilinx's VirtexE FPGA set.

Вступ

Задачі обробки даних, такі як стиск і розпакування відео, синтез зображень, виділення зображень, обробка звуку характеризуються великою обчислювальною складністю. З ускладненням задач обчислювальна складність теж зростатиме. Більшість алгоритмів обробки даних мають природним паралелізм [2], який можна використати паралельним вмиканням операційних пристроїв. При цьому для забезпечення ефективного паралельного доступу до даних необхідно створювати багатопортові регістрові файли. Регістровий файл – це набір програмно-доступних регістрів. У сучасних програмованих процесорах регістровий файл займає одне з центральних місць. Його використовують для локального збереження операндів, результатів, адрес даних, а також для ефективного обміну між операційними пристроями та пам'яттю.

Розглянемо деякі історичні аспекти розвитку високопродуктивних комп'ютерів з погляду організації регістрових файлів процесора. Програмно–доступні регістри використовують з початку 1960-х років [1]. В 1964 році фірма IBM розробила серію універсальних комп'ютерів IBM/360 для наукових та комерційних розрахунків, в процесорах яких було використано регістровий файл, що містив 16 32-розрядних цілочисельних регістрів та 16 64-бітних регістрів з рухомою комою. Також в 1964 р. було виготовлено перший суперкомп'ютер для наукових розрахунків CDC 6600, процесор якого мав регістровий файл, який містив 24 регістри. Завантаження до семи з восьми адресних регістрів призводили до важливих побічних ефектів. Завантаження до регістрів A [1–5] призводило до того, що дані за цією адресою автоматично пересилалися в регістри даних X [1–5] відповідно. Аналогічно, завантаження адреси до A6 чи A7 призводило до зберігання даних з X6 чи X7 за цією адресою. Це давало змогу ефективно кодувати векторні операції, оскільки операції завантаження і зберігання не було потрібно явно визначати. CDC 6600 міг виконувати до 10 операцій паралельно.

У 1977 році було розроблено перший векторний суперкомп'ютер Cray-1. Його регістровий файл мав ієрархічну структуру, що складалася з п'яти типів регістрів. Загалом регістровий файл було поділено на файл основних регістрів і файл другорядних чи фонових регістрів. Кількість основних регістрів менша, що дає їм змогу швидше працювати, тоді як кількість другорядних регістрів більша, проте вони повільніші. Довготривалі значення розміщувалися в другорядному регістровому файлі і переміщалися в основні регістрові файли при використанні. Cray-1 загалом містить 656 регістрів даних і адрес (з векторними регістрами, але не враховуючи регістри керування). На особливу увагу заслуговує група з восьми векторних регістрів. Кожний такий регістр може містити 64-елементний вектор з рухомою комою. В одній 16-розрядній команді можна

додати, відняти чи перемножити два вектори. Вважається, що машина Cray-1 є найпродуктивнішою з класу однопроцесорних систем.

Отже, з розглянутих прикладів бачимо, що для організації високопродуктивних обчислень важливо не тільки мати велику кількість регістрів, але і забезпечити швидкий доступ до даних за рахунок ефективної організації регістрового файла.

Огляд літературних джерел

З аналізу публікацій [4–10, 12, 13] на рис. 1 запропоновано класифікацію типів регістрових файлів програмованих процесорів. У вказаних роботах в основному розглядається статична організація збереження даних. Тобто, при записі і читанні даних вказується безпосередня адреса регістру, яка є незмінною під час виконання програми. Такі принципи в основному використовують в універсальних процесорах. В роботі [14] вказано вимоги до регістрової пам'яті для потокових задач. Тут наголошено на тому, що ефективнішим буде динамічне збереження даних через те, що більшість задач інтенсивної обробки даних вимагають від регістрового файла прийняти дані, перевпорядкувати їх і з певною затримкою видати на обробку обчислювальним модулям. В роботі [3] в основу регістрового файла покладено динамічний принцип. Але реалізація такого регістрового файла має істотні недоліки, оскільки черги мають фіксовану затримку.

Рис. 1. Типи організацій регістрових файлів

Постановка задачі

Метою роботи є аналіз структур регістрових файлів програмованих процесорів, порівняння динамічної та статичної організації збереження даних в регістрових файлах, а також розробка та синтез на ПЛІС базових структур регістрових файлів та проведення їх порівняльного аналізу.

Класифікація типів регістрових файлів

Детально розглянемо структуру кожного з варіантів побудови регістрових файлів, запропонованих на рис. 1.

Інтегрований багатопортовий регістровий файл

Для архітектур процесорів з інтегрованим багатопортовим регістровим файлом необхідно, щоб кожний з N обчислювальних модулів мав доступ до будь-якого регістру r. На рис. 2 показано

класичну структуру інтегрованого регістрового файла з двома портами читання (rd1, rd2) та одним портом запису(wd), який містить 32 регістри (R0-R31). В регістр R0 заборонено запис, оскільки в ньому зберігається константа нуль. Кожний вихід регістра під'єднано до інформаційного входу мультиплексора. Адреси (rs1, rs2), що подаються на керівні входи мультиплексорів, визначають номери регістрів, які читаються. Для запису використовують демультиплексор, через який подають сигнал запису даних (we), що надходять з вхідної шини. Процес запису синхронізують тактовими імпульсами, що подаються на вхід clk. Збільшення кількості портів читання збільшує кількість вихідних шин та мультиплексорів. Збільшення кількості портів запису призводить до збільшення кількості вхідних шин даних та демультиплексорів.

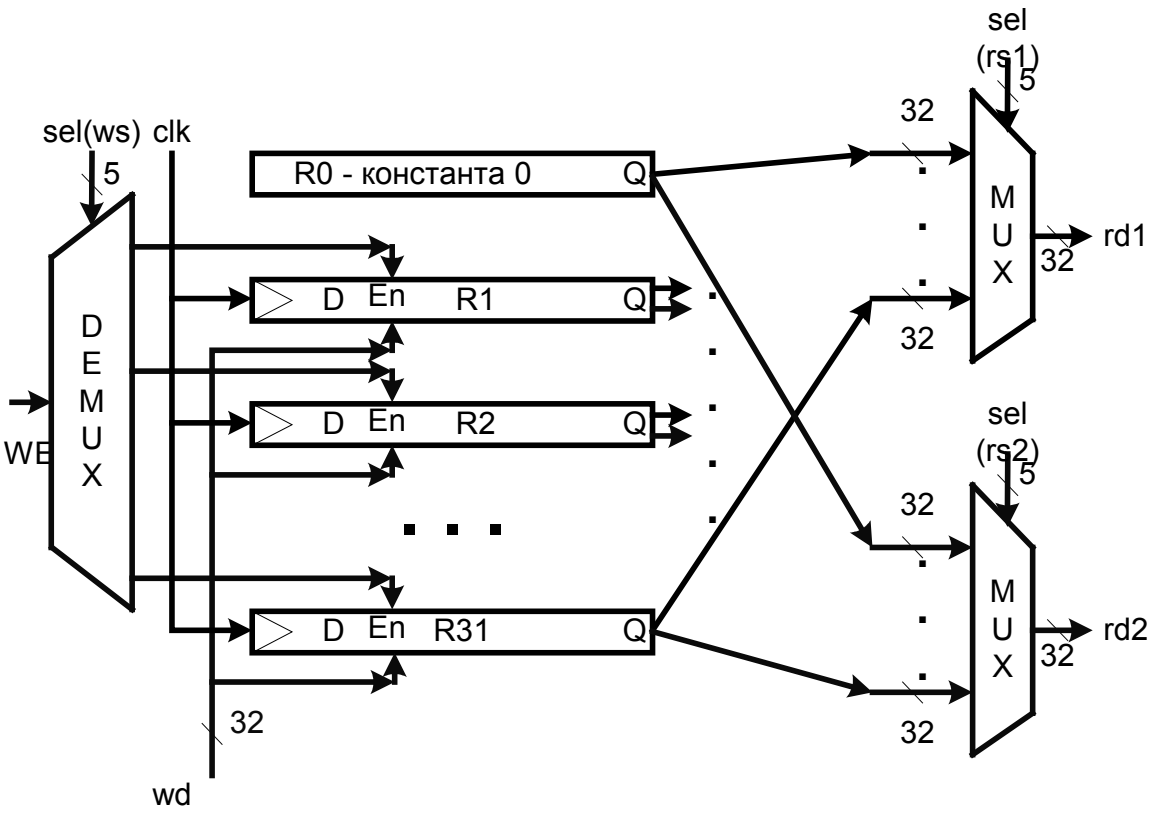

Рис. 2. Внутрішня організація інтегрованого регістрового файла

Площу кристала A для інтегрованого регістрового файла з p портами та r регістрами визначають за формулою [3]:

$$A = O(rp^2), \quad (1)$$

де K – деяке постійне значення.

Для кожного обчислювального модуля необхідно n_r регістрів та $p_в+3$ портів (два порти читання операндів, один – на запис результату, а також $p_в$ зовнішніх портів). Отже:

$$A = O(N^* n_r^* ((p_в+3)^* N)^2). \quad (2)$$

З формули (2) випливає, що площа кристала для інтегрованого регістрового файла пропорційна кількості обчислювальних пристроїв N, піднесеної до куба.

Час доступу до багатопортового інтегрованого регістрового файла визначають за формулою [3]:

$$T = O(p^* r^{1/2}). \quad (3)$$

Підставивши значення p та r до формули (3), отримуємо:

$$T = O((p_e+3)^* N^* (n_r^* N)^{1/2}). \quad (4)$$

Отже, з формули (4) випливає, що затримка багатопортового інтегрованого регістрового файла залежить від кількості обчислювальних модулів N, піднесеної до степені 3/2.

Наведені дані показують, що на базі багатопортового інтегрованого регістрового файла практично неможливо реалізувати ефективну систему з високим коефіцієнтом паралелізму, оскільки для кількості обчислювальних пристроїв більше чотирьох стрімко зростає апаратна складність такого регістрового файла.

Існують методи зменшення розміру інтегрованого багатопортового регістрового файла шляхом дублювання інформації в різних комірках, що дає змогу для кожного порта читання мати свою копію даних. Однак, записувати дані необхідно в усі копії комірок для забезпечення цілісності даних. За великої кількості портів читання таке дублювання може дещо зменшити загальну площу регістрового файла.

Інтегрований регістровий файл ефективно використовувати в універсальних архітектурах з невеликими коефіцієнтами паралельності. Наприклад, регістрові файли IBM RS/6000 та SUN SuperSPARC містять 4 порти для читання і 2 порти для запису [5].

Розподілений регістровий файл

З використанням розподілених регістрових файлів можна зменшити площу кристала, що займає регістрова пам'ять процесора. Розподілення регістрового файла проходить шляхом поділу обчислювальних елементів на групи з локальним регістровим файлом. Відповідно, такі регістрові файли містять меншу кількість портів та регістрів, що зменшує їх апаратну складність. Локальні регістрові файли можуть бути повністю незалежними, тобто дані для певної групи обчислювальних елементів доступні тільки з конкретного регістрового файла. Такі розподілені файли згідно з класифікацією, наведеною вище, називають кластерними. Якщо будь-якій групі обчислювальних модулів доступні дані з будь-якого локального регістрового файла, то отримуємо інший крайній випадок організації розподілених регістрових файлів. Така структура називається розподіленими регістрами з керованою комутацією.

Будову розподіленого регістрового файла з керованою комутацією розглянуто в роботі [4]. Тут описано процесор з тригер-транспортною архітектурою (ТТА). Структура ТТА дуже проста (див. рис. 3). Її можна розглядати як набір функціональних модулів (ФМ) та локальних регістрових файлів (ЛРФ), пов'язаних мережею внутрішніх з'єднань. ТТА програмується за допомогою точного визначення необхідних передач даних шляхом подачі керівних сигналів на комутуючу мережу. Формування таких керівних сигналів виконує контролер комутуючої мережі(ККМ).

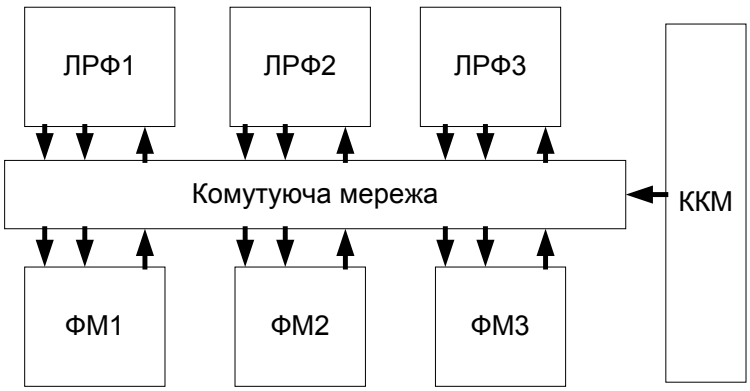

Рис. 3. Організація регістрового файла процесора з тригер-транспортною архітектурою

Детально розглянемо будову розподіленого регістрового файла з керованою комутацією, показаного на рис. 4. Тут для кожного входу обчислювального модуля ОМ виділено окремий банк регістрів ЛРФ з одним портом для читання та одним портом для запису даних. Розмір кожного банку регістрів є сталим і не залежить від кількості обчислювальних модулів. Банки локальних регістрів об'єднуються комутуючою мережею. Щоб забезпечити з'єднання для М входів, необхідно $O(M^2)$ комутуючих елементів [17]. Отже, площа кристала, яку буде займати комутуюча

мережа, прямопропорційна N^2. Загальну площу кристала розподіленого регістрового файла з керованою комутацією визначають шляхом додавання площ банків регістрів ($A_{БР}$) та площі комутуючої мережі($A_{КМ}$):

$$A= N*A_{БР} + A_{КМ}=N*O(1/2*n_r*2^2)+O(N^2)= N*O(2*n_r)+O(N^2). \quad (5)$$

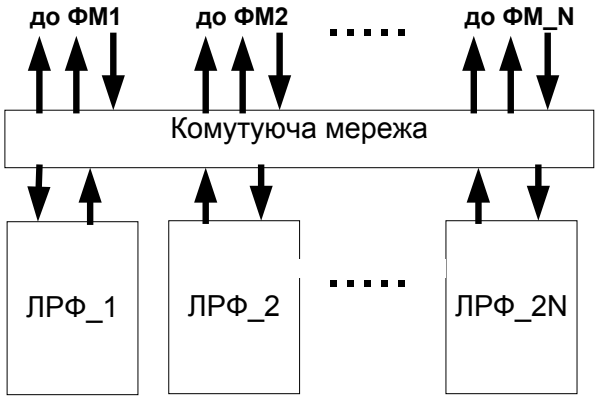

Рис. 4. Структура розподіленого регістрового файла з керованою комутацією

З формули (5) випливає, що площа кристала для розподіленого регістрового файла з керованою комутацією пропорційна кількості обчислювальних пристроїв N піднесеної до квадрата.

Час доступу до розподіленого регістрового файла з керованою комутацією складається з затримки на комутуючій мережі та з затримки звернення до банку локальних регістрів. Затримка до банку регістрів не залежить від кількості обчислювальних модулів і є сталою для заданої кількості регістрів. Затримка на комутуючій мережі прямо пропорційна кількості її входів [17]. Виходячи з вищесказаного, отримуємо формулу (6), яка показує, що час доступу до регістрового файла з керованою комутацією прямо пропорційний кількості обчислювальних модулів N.

$$T = O(2* n_r^{1/2})+O(N). \quad (6)$$

Кластерні регістрові файли в основному використовують в векторних та ОКМД процесорах [6, 7]. У таких процесорах обробка всіх n компонентів операндів задається однією командою для m обчислювальних елементів. Кожен з таких операндів міститься в локальному регістровому файлі для відповідного обчислювального модуля і немає необхідності в повноцінних зв'язках між локальними регістрами. На рис. 5 показано кластерну організацію регістрового файла процесора TMS320C64x. У цьому процесорі функціональні модулі поділені на підмножину A(L1, S1, M1, D1) та підмножину B(L2, S2, M2, D2). Модулі L(L1, L2) виконують арифметичні операції та операції порівняння. Модулі S(S1,S2) виконують арифметично-логічні операції та команди керування. Модулі M(M1, M2) дають змогу перемножувати 16-розрядні операнди, а модулі D(D1,D2) виконують арифметичні команди та виконують роль генераторів адрес. Кожна підмножина має свій регістровий файл. Для пересилання даних з регістрового файла однієї підмножини до функціональних модулів іншої підмножини використовують додаткові мультиплексори МП. Фактично можливість повноцінної пересилки обмежена, оскільки для цього в кожному з регістрових файлів виділяється тільки один порт. Саме з цієї причини організацію регістрового файла TMS320C64x можна назвати кластерною.

Для оцінки параметрів розподіленого регістрового файла з кластерною організацією потрібно ввести додаткову величину L. Вона визначає кількість кластерів, що утворюють регістровий файл. Відповідно, кількість обчислювальних модулів, які потрібно об'єднати комутуючими мережами в кластері, становить N/L. Підставивши в формули (5) та (6) значення N/L для розподіленого регістрового файла з кластерною організацією, отримуємо, що:

– площа кристала пропорційна $(N/L)^2$;

– час доступу пропорційний N/L.

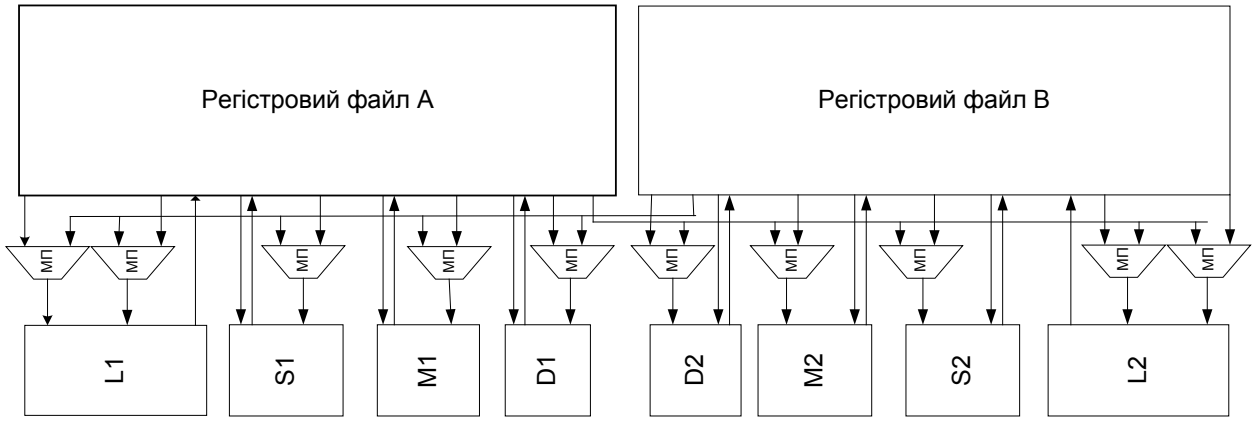

Рис. 5. Організація регістрового файла TMS320C64x

У розглянутих структурах розподілених регістрових файлів доступ до регістрів для обчислювальних модулів розмежовувався в просторі. Існує інша організація розподілених регістрових файлів, яка передбачає розподілений доступ до регістрів в часі. Така організація отримала назву регістрових вікон. В основному її використовують для зменшення довжини команди та споживаної потужності процесора, а також дає можливість нарощувати кількість регістрів без зміни формату команди. За апаратною та часовою складністю регістрові вікна повністю збігаються з розподіленим регістровим файлом з керованою комутацією. На рис. 6 показано віконну організацію регістрового файла мікроконтролера WIMS [12]. Тут регістровий файл поділено на дві рівнозначні частини (два вікна). До першої частини входять цілочисловий регістровий файл RF0 та регістровий файл адрес ARF0, до другої – цілочисловий регістровий файл RF1 та регістровий файл адрес ARF1. Команді в один момент часу дозволено доступ тільки до RF0 та ARF0 або до RF1 та ARF1. Для зміни активного вікна використовують спеціальну команду управління. Така організація регістрового файла дає змогу зменшити поле адреси регістру на один біт.

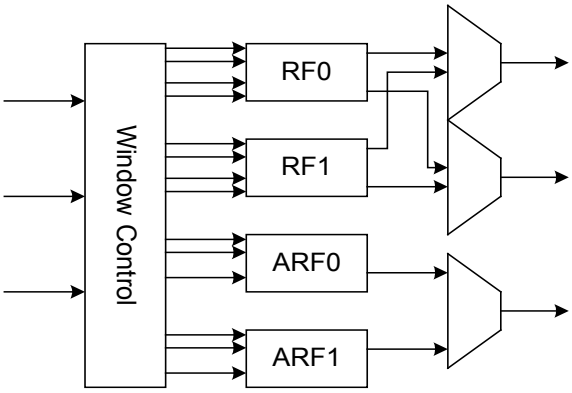

Рис. 6. Організація регістрового файла WIMS

У роботі [16] запропоновано віконну організацію, в якій 256 регістрів поділено на 8 рядків по 8 регістрів. Одна команда може адресувати тільки регістри з одного вибраного рядка. Перевагою такої структури є те, що відпадає необхідність додаткових команд перемикання вікон.

З наведених матеріалів бачимо, що організація регістрового файла у вигляді регістрових вікон є неефективною для систем з високим рівнем паралелізму. Вона дає змогу збільшити кількість регістрів, але не може організувати паралельний доступ до будь-якого регістру в один момент часу.

Ієрархічні регістрові файли

В універсальних комп'ютерах надзвичайно важливим є той факт, що звернення процесора до пам'яті завжди локалізовано в невеликому діапазоні змін її адрес. Саме він дає змогу застосовувати

ієрархічну систему пам'яті, аби розв'язати невідповідність швидкодій процесора та пам'яті з одним рівнем ієрархії. Однак існує множина алгоритмів з великими наборами вхідних даних, що характеризуються дуже низьким рівнем перевикористання. Для виконання таких алгоритмів кеш-пам'ять є зайвою ланкою. Щоб ефективніше реалізувати доступ до пам'яті, будують ієрархічні регістрові файли [8]. За допомогою таких регістрових структур можна перерозподілити центральний регістровий файл на великий регістровий файл з декількома портами для виконання операцій з пам'яттю і менший регістровий файл з великою кількістю портів для внутрішнього з'єднання арифметичних блоків. Цей перерозподіл зменшує складність ієрархічного регістрового файла порівняно з інтегрованим. Загальну площу кристала ієрархічного регістрового файла підраховують шляхом додавання площ регістрових файлів, що входять до його складу. Підставивши потрібні параметри до формули (2), отримуємо площі складових регістрових файлів.

У роботах [9, 10] запропоновано дворівневу модель регістрового файла для динамічних суперскалярних архітектур, яка дає змогу зменшити кількість регістрів та кількість портів. Така дворівнева організація зменшує кількість портів утричі, а також покращує час доступу до даних на 46%. Деякою мірою регістровий файл архітектури Sparc є також ієрархічним, оскільки містить безпосередньо регістрові вікна та асоціативний регістровий кеш [13].

Динамічна та статична організація збереження даних у регістрових файлах

У розглянутих структурах регістрових файлів використовують безпосередню адресацію регістрів. Доступ до даних у згаданих регістрових файлах відбувається шляхом задання адреси регістру безпосередньо в команді. Така статична організація збереження даних для виконання потокових алгоритмів неефективна. Наприклад, задачі роботи зі зображеннями, звуком, стиском даних в реальному масштабі часу вимагають від регістрового файла одночасного приймання вхідних масивів з декількох вхідних каналів, їхньої затримки на потрібну кількість тактів та видавання декількома вихідними каналами раніше прийнятих масивів з впорядкуванням за заданим законом. Для розв'язання таких задач доцільно використовувати динамічне збереження даних в регістрових файлах [13]. Основою апаратних рішень динамічних регістрових файлів є черги. Збереження даних для складних задач обробки інформації в реальному масштабі часу на основі черг за динамічним принципом має такі переваги:

– доступ до черги є простішим, ніж до статичного регістрового файла;

– збільшення регістрів в черзі не призводить до вагомого зростання апаратної складності регістрового файла;

– простота оптимізації структури регістрового файла під заданий алгоритм;

– необхідна менша кількість адресних ліній порівняно з типовими регістровими файлами;

– проблема розподілу регістрів вирішується автоматично, оскільки дані записуються не в конкретний регістр, а в чергу;

– можливість реалізації регістрового файла на основі модулів пам'яті.

Відомі такі статистичні дані: 40 % результатів виконання арифметичних операцій використовують на наступному такті виконання програми, а ще 45 % – через такт [4]. Такі дані ще раз переконують у ефективності реалізації регістрових файлів на базі черг.

У роботі [3] запропоновано будувати регістровий файл на основі черг за принципом FIFO. Недоліком такої організації є те, що для алгоритмів, в яких умови сумісності черг не виконуються, різко збільшується апаратна складність динамічного регістрового файла.

У цій роботі запропоновано організувати регістровий файл на базі черги з програмованою затримкою. Структуру такого регістрового файла, що належить до розподілених регістрових файлів з керованою комутацією та динамічним збереженням даних, наведено на рис. 7. Такий тип структури має найбільші можливості спеціалізації, що дає змогу виконувати алгоритми з максимальною продуктивністю.

До внутрішньої структури модуля черги належить двопортова адресна пам'ять. Один порт дає змогу читати дані, а інший – записувати дані до пам'яті. Додатково введено два елементи:

лічильник та суматор. Читання даних виконується аналогічно до пам'яті FIFO, тобто через кожний тактовий імпульс читається комірка за адресою, більшою на одиницю від попередньої. Адресу збільшує лічильник. Алгоритм просування даних у банку являє собою генерування сигналу читання для потрібної комірки пам'яті цього банку. Отже, кожна комірка є пронумерованою і фіксовано зберігає свій номер. Лічильник пам'яті генерує послідовні значення, що визначають номер комірки, яка повинна видавати дані назовні. Отже, як бачимо, система читання дуже проста – дані читаються тоді, коли підходить їхня черга. А весь алгоритм впорядкування та адресації полягає у записі: до якої комірки запишуться дані, тоді відповідно вони і виведуться назовні. Адреса запису формується за допомогою лічильника та суматора шляхом додавання значення затримки (адреса А) та значення лічильника. Двх – вхідна шина даних, Двих – вихідна шина даних.

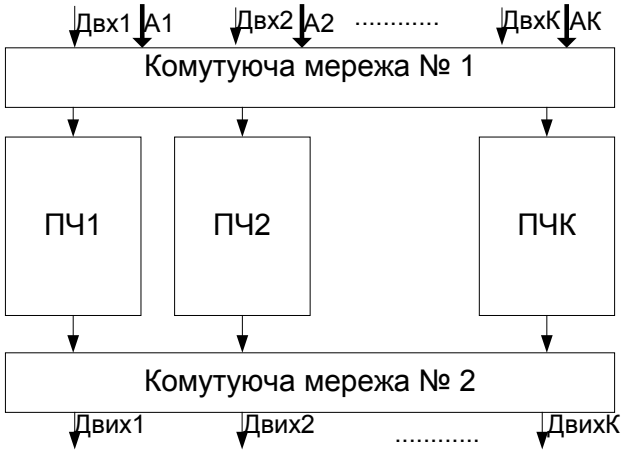

Рис. 7. Структура регістрового файла на базі черги

Синтез регістрових файлів на базі ПЛІС

Для практичного дослідження мовою VHDL описано регістрові файли з різними параметрами, а також проведено їхній синтез на базі ПЛІС серії VirtexE фірми Xilinx. Особливістю будови мікросхем цієї серії є наявність двопортової 4096-бітної пам'яті, яку можна використовувати як складовий елемент регістрового файла [18]. Кожному обчислювальному модулю виділялося вісім 8-розрядних регістрів і три порти (один для запису та два для читання). Прийнято, що кількість портів для вводу/виводу даних $p_в$ дорівнює нулю. Результати синтезу багатопортового інтегрованого регістрового файла для ПЛІС XCV300E наведено в таблиці. З результатів, наведених в таблиці, бачимо, що практично можливо використати тільки дві конфігурації інтегрованого регістрового файла для одного та двох обчислювальних модулів. Отже, реалізувати процесор, що містить багатопортовий інтегрований регістровий файл на ПЛІС з великим коефіцієнтом паралельності, неможливо.

Розглянемо реалізацію розподіленого регістрового файла. На кожний вхід обчислювального модуля виділяється два двопортові локальні регістрові файли, що складаються з чотирьох регістрів.

Результати синтезу багатопортового регістрового файла на базі ПЛІС XCV300E

Кількість обчислювальних модулів	Кількість регістрів	Кількість портів	Кількість Slice	Час доступу, нс
1	8	3	24 (3%)	7,2
2	16	6	225 (29%)	7,24
3	24	9	625 (81%)	9,23
4	32	12	1246(162%)	-

На рис. 8 наведено результати синтезу розподіленого регістрового файла з керованою комутацією. Тут можна побачити дві криві, які відповідають за статичний та динамічний принцип збереження даних. Площа кристала, що займає регістровий файл з динамічним збереженням даних, в середньому на 3 % більша, ніж регістрового файла з статичним збереженням даних. Це пояснюється наявністю додаткових елементів, що забезпечують можливість програмувати затримку. Важливим є той факт, що розподілений регістровий файл для ПЛІС XCV300E дає змогу під'єднати до п'яти обчислювальних модулів, що в 2.5 раза більше порівняно з інтегрованим багатопортовим регістровим файлом. Час доступу до розподіленого регістрового файла практично дорівнює часу доступу до інтегрованого багатопортового регістрового файла для N менше 5. Використовуючи формули (4) та (6) можна передбачити, що з більшенням N час затримки для інтегрованого багатопортового регістрового файла зростатиме швидше порівняно з часом затримки для розподіленого регістрового файла.

Рис. 8. Площа кристала розподіленого регістрового файла на базі ПЛІС XCV300E

Ще однією вагомою перевагою розподіленого регістрового файла є можливість оптимізації структури під задачу шляхом усунення непотрібних зв'язків.

Висновки

У роботі проаналізовано можливі організації регістрових файлів програмованих процесорів. Наведено формули, що визначають апаратну та часову складність різних структур регістрових файлів. Показано обмеження, які спричиняє використання багатопортового регістрового файла. Запропоновано нову структуру регістрового файла на базі черги з програмованою затримкою, що може бути використано в програмованих процесорах. Проведено синтез структур регістрових файлів на базі ПЛІС сімейства VirtexE фірми Xilinx. Результати синтезу показали, що розподілений регістровий файл для ПЛІС XCV300E дає змогу під'єднати до п'яти обчислювальних модулів, що в 2.5 рази більше порівняно з інтегрованим багатопортовим регістровим файлом. Наведені показники доводять, що розподілені регістрові файли доцільно використовувати у процесорах з високим рівнем паралелізму.

1. David J. Kuck. The Structure of Computers and Computations. John Wiley & Sons, Pittsburgh, Pennsylvania, 1978. 2. Kung S.Y. VLSI Array Processors, Prentice Hall, 1988. 3. Fernandes M.M., Llosa J., Topham N. Using Queues for Register File Organization in VLIW Architectures. Technical Report ECS-CSG 29–97, Dept of Computer Science, University of Edinburgh, 1997. 4. Corporaal H. Microprocessor Architectures: From VLIW to Tta, John Wiley & Sons, Inc., New York, NY, 1997. 5. Blank G. and Krueger S. SuperSPARC: A fully integrated superscalar processor. In Hot Chips III. A Symposium on High-

Performance Chips // IEEE, August 1991. 6. CEVA: CEVA-X1620 Datasheet. CEVA, 2005. 7. Texas Instruments: TMS320C64x Technical Overview. 2005. – www.ti.com. 8. Rixner S., Dally W., Khailany B., Mattson P., Kapasi U. Register organization for media processing. International Symposium on High Performance Computer Architecture (HPCA). – 2000. – P. 375–386. 9. Balasubramonian R., Dwarkadas S., Albonesi D. Reducing the Complexity of the Register File in Dynamic Superscalar Processor. In Proceedings of the 34th International Symposium on Microarchitecture, December 2001. 10 Balasubramonian R., Dwarkadas S., and Albonesi D. Reducing the complexity of the register file in dynamic superscalar processors. In Proceedings of the International Symposium on Microarchitecture, Dec. 2001. 11. Russell R.M. The CRAY-1 computer system. Communications of the ACM, 21(1): 63–72, Jan. 1978. 12 Ravindran R., Senger R., Marsman E., Dasika G., Guthaus M., Mahlke S. and Brown R. Increasing the Number of Effective Registers in a Low-Power Processor Using a Windowed Register File. Proc. – 2003. 13. David L. Weaver and Tom Germond. The SPARC Architecture Manual, Version 9. Sparc International and PTR Prentice Hall, Englewood Cliffs, NJ, 1994. 14. Мельник А.О. Спеціалізовані системи реального часу : конспект лекцій. – Львів: Навч.видання, 1996. – 53 с. 15. Мельник А.О., Сало А.М. Методика проектування паралельного процесора на основі пам'яті з детермінованою вибіркою // Вісн. Нац. ун-ту "Львівська політехніка". – 2005. – № 546. – С. 96–101. 16. Hakenes R.. A novel low-power microprocessor architecture. www.iccd-conference.org/proceedings/2000/08010141.pdf. 17. Gregory W. A Comparison of Circuits for On-Chip Programmable Crossbar Switches // 10th NASA Symposium on VLSI Design, Albuquerque, NM, March 20–21, 2002. 18. www.xilinx.com

УДК 621.317

Р.С. Паньків
Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин

ЗМЕНШЕННЯ НАДЛИШКОВОСТІ ДАНИХ АНАЛОГО-ЦИФРОВОГО ПЕРЕТВОРЕННЯ ПОЛІГАРМОНІЧНИХ СИГНАЛІВ

Досліджено можливість використання рівномірного квантування за рівнем під час аналого-цифрового перетворення полігармонічних сигналів з метою зменшення загального об'єму даних, що формуються. Розглянуто основні принципи апаратної реалізації перетворення та подальшої обробки отриманих кодів. Наведено основні числові характеристики похибки, що виникає під час використання рівномірно квантованих миттєвих значень вхідних сигналів.

Possibility of the use a uniform quantization on a level at analog-digital conversion of polyharmonical signals with the purpose of diminishing of total volume of data, which are formed is explored in the article. Basic principles of hardware representation of transformation and subsequent treatment of the got codes are considered. Main numerical descriptions of error which arises up at the use of uniformly quantized instantaneous values of input signals are resulted.

Вступ

Сучасна промисловість пропонує широкий спектр різноманітних аналого-цифрових перетворювачів (АЦП) в інтегральному виконанні, які відрізняються принципами функціонування, технічними параметрами та сферою використання: АЦП паралельного унітарного кодування, інтегруючі