ОБЧИСЛЮВАЛЬНІ СИСТЕМИ КЛАСУ SIMD (ОКМД)

SIMD-системи були першими обчислювальними системами, що складаються з великого числа процесорів, і серед перших систем, де була досягнута продуктивність порядку GFLOPS. Згідно з класифікацією Флінна, до класу SIMD відносяться ОС, де множина елементів даних піддається паралельній, але однотипній обробці.

На рис. 1 показані приблизні характеристики продуктивності деяких типів обчислювальних систем класу SIMD.

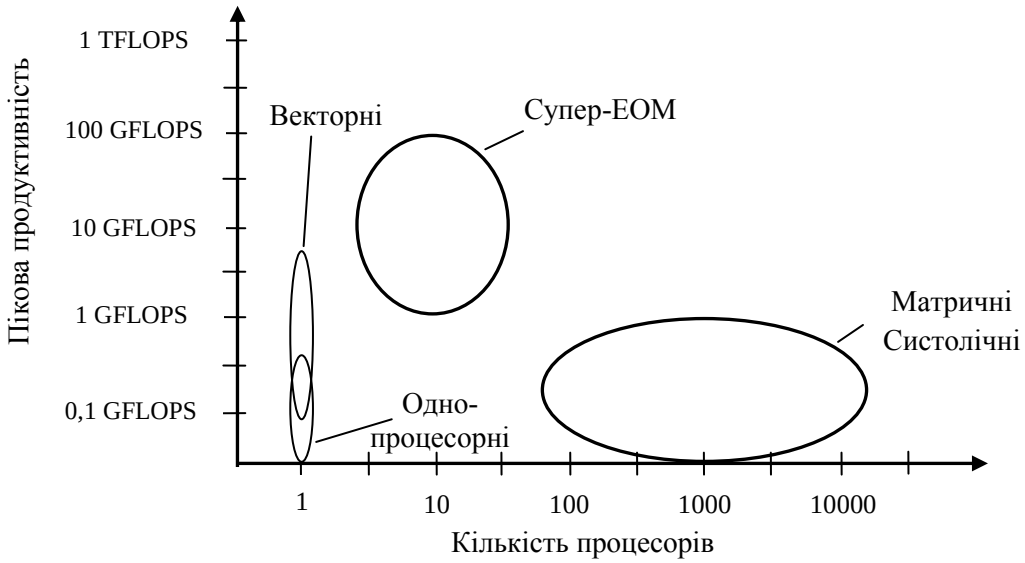

Рисунок 1 - Продуктивність SIMD-систем як функція їх типу та кількості процесорів

SIMD-системи багато в чому схожі на класичні фон-нейманівські ОМ: у них також є один пристрій управління, який забезпечує послідовне виконання команд програми. Відмінність стосується стадії виконання, коли загальна команда транслюється великій кількості процесорів (у простому випадку – АЛП), кожен з яких обробляє свої дані.

Раніше вже наголошувалася нечіткість класифікації Флінна, через що різні типи ОС можуть бути віднесені до того або іншого класу. Проте в теперішній час прийнято вважати, що клас SIMD складають векторні (векторно-конвеєрні), матричні, асоціативні, систолічні і VLIW-обчислювальні системи.

1 Векторні і векторно–конвеєрні комп'ютерні системи

Під час розв'язання на комп'ютері науково-технічних та інших задач часто зустрічається необхідність визначення значень однієї і тієї ж функції для групи даних, розташованих у комірках пам'яті (регістрах) з упорядкованими адресами (номерами). Такі набори даних називаються векторами.

Структурна схема векторного процесора показана на рис. 2.

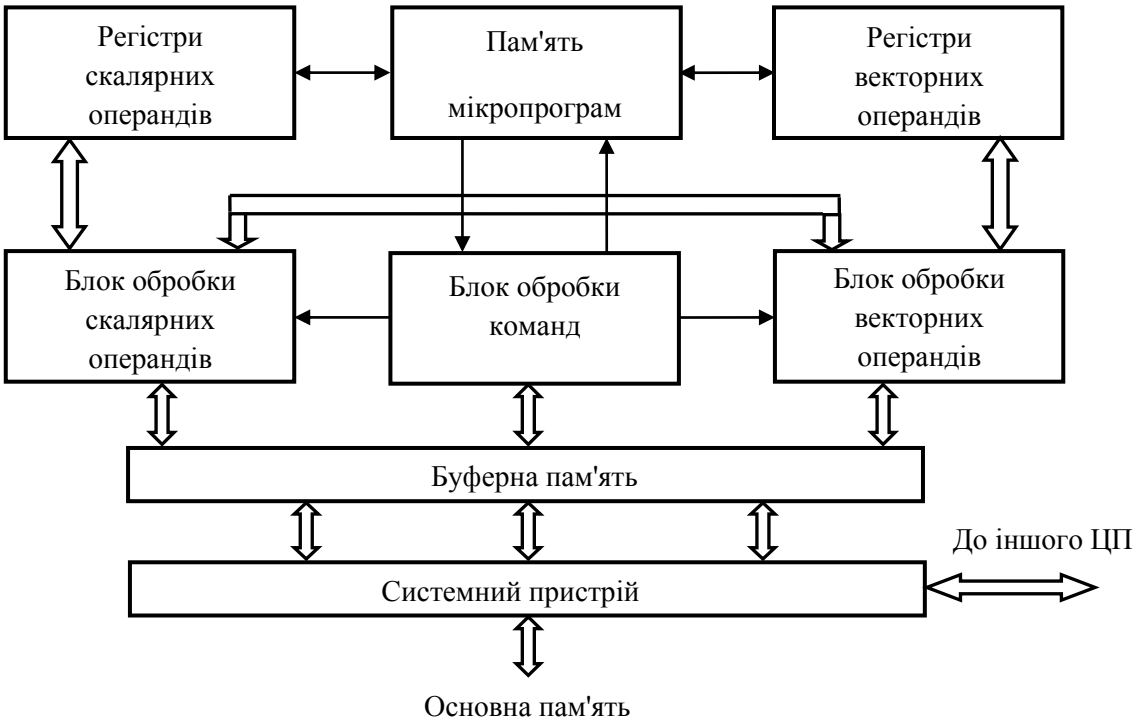

Рисунок 2 - Структурна схема векторного процесора

З метою підвищення продуктивності комп'ютера до складу його процесора разом зі скалярними засобами включають засоби *векторної обробки* даних або вводять спеціалізований векторний співпроцесор. Векторні засоби включають векторні команди, векторні регістри та іншу апаратуру для реалізації цих команд.

Засоби векторної обробки дозволяють за допомогою єдиної команди виконувати дії над усіма елементами масивів. Кожний елемент такого масиву (вектор) обробляється незалежно від інших елементів на конвеєрному операційному пристрої (ОП), який за рахунок спеціальної структурної організації може приймати в кожному такті пару елементів векторів-операндів і через деякий час, який називається *стартом конвеєра*, видавати відповідний елемент вектора-результату. Оскільки пари елементів векторів-операндів надходять у конвеєрний ОП в кожному такті, то через час, який визначає довжину часу старту конвеєра після запуску векторної операції, з виходу ОП кожного такту будуть видаватись елементи вектора-результату.

До апаратних векторних засобів відносяться:

- арифметичні конвеєри (кожний такий конвеєр може містити декілька незалежних спеціалізованих конвеєрних арифметичних пристроїв);
- векторні регістри, які зберігають векторну інформацію, що використовується для обробки в арифметичних конвеєрах;
- конвеєри завантаження-запису, які виконують обмін інформацією між векторними регістрами і оперативною пам'яттю ЕОМ.

Векторні засоби обробки інформації дозволяють збільшити продуктивність ЕОМ на кілька порядків. Збільшення продуктивності можна оцінити за допомогою коефіцієнта підвищення продуктивності

$$\mathrm{P} = \frac{t_{\mathrm{ср}}(1 + k_{\mathrm{в}})(m-1)}{\left(1 - k_{\mathrm{в}} t_{\mathrm{ср}}\right) + k_{\mathrm{в}}\left(t_{cm} + \frac{mt}{DC}\right)} \quad ,$$

де $t_{ср}$ – середній час виконання скалярних (не векторних) операцій;

$k_в$ – коефіцієнт векторизації задачі, який дорівнює відношенню числа векторних операцій до загального числа операцій, що виконуються;

$t_{ст}$ – середній час старту конвеєра;

t – довжина машинного такту;

D – число арифметичних конвеєрів;

C – число векторних команд в одному арифметичному конвеєрі, що виконуються паралельно;

m – кількість розрядів.

З приведеного співвідношення видно, що для збільшення продуктивності ЕОМ необхідно дотримуватись таких вимог:

- використовувати арифметичні конвеєри з мінімально можливою довжиною такту;
- збільшувати число арифметичних конвеєрів;
- збільшувати число векторних команд, що паралельно опрацьовуються в одному арифметичному конвеєрі.

Необхідна умова використання двох останніх вимог – збільшення пропускної здатності під час передачі інформації між арифметичними конвеєрами і векторними регістрами, а також підвищення продуктивності конвеєрів завантаження/запису під час операції обміну між векторними регістрами і оперативною пам'яттю.

У ході розв'язання реальних задач на продуктивність ЕОМ істотно впливають:

- швидкість обробки скалярних величин і параметри векторного пристрою. Залежності збільшення продуктивності ЕОМ від коефіцієнта векторизації при зміні параметрів векторних засобів обробки приведені на рис. 3;
- об'єм векторних регістрів. У разі недостатнього об'єму векторні регістри не можуть згладити нерівномірність інформаційного потоку, що обробляється, невизначено збільшується інтенсивність обміну між векторними арифметичними конвеєрами і оперативною пам'яттю.

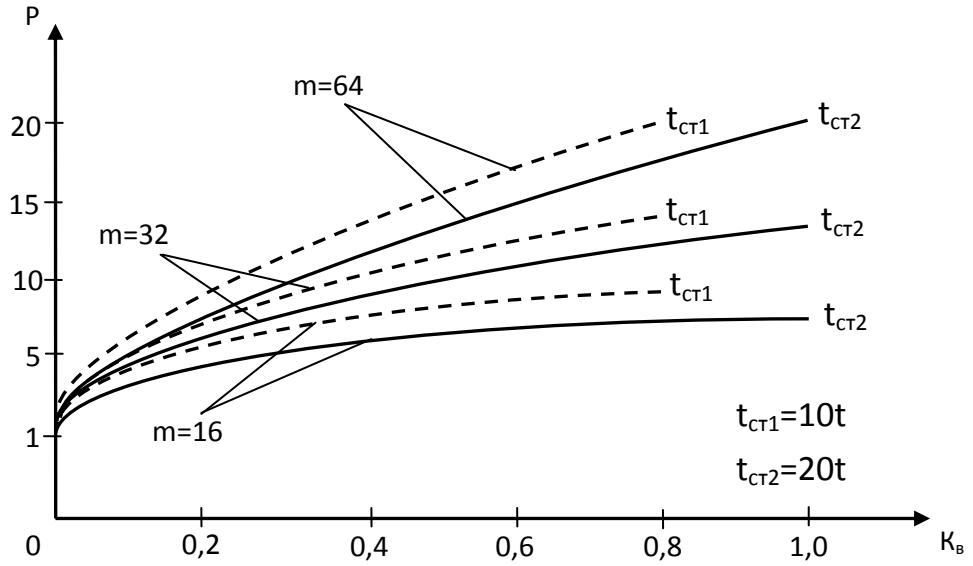

Рисунок 3 - Залежності P($k_в$, m)

Векторні команди дозволяють однією командою дати розпорядження на виконання деякої операції (векторної операції) над елементами вектора, наприклад, поелементне додавання векторів та організовують і управління обчислювальним циклом.

У машині, яка має векторні команди, виконуються також звичайні команди над одиничними даними (скалярні команди).

Наявність векторних команд сприяє підвищенню швидкодії процесора за рахунок зменшення витрат часу на організацію обчислювального циклу.

Звичайно, з метою досягнення підвищеної швидкодії виконання самих векторних операцій ставиться на конвеєр, до того ж може бути кілька арифметичних конвеєрів (ліній), а окремі пристрої конвеєрної лінії можуть, у свою чергу, містити конвеєри для виконання покладених на них підфункцій.

Векторні команди (звичайно їх порівняно мало) мають ряд особливостей. Їхній код операції не тільки задає операцію, яка підлягає виконанню, але і визначає необхідну конфігурацію активних частин конвеєра для даної операції. Команда вказує початок вектора (векторів) в пам'яті (або блоці регістрів), довжину вектора, розмір і тип елементів вектора (ціле число, число з плаваючою комою і т.д), визначає місце знаходження вектора-результату.

Прикладами конвеєрно-векторних процесорів можуть служити спеціалізовані на виконання векторних і матричних операцій співпроцесори, при цьому продуктивність машини під час виконання векторних і матричних операцій збільшується до 30 Мфлоп/с.

Конвеєрно-векторні ОС в теперішній час є одним з основних напрямків у разі утворення супер-ЕОМ для широкого кола задач. До ОС такого типу відносяться найшвидкодіючі супер-ЕОМ сімейства CRAY (США). На рис. 4 подана спрощена структура конвеєрно-векторного процесора супер-ЕОМ CRAY-1.

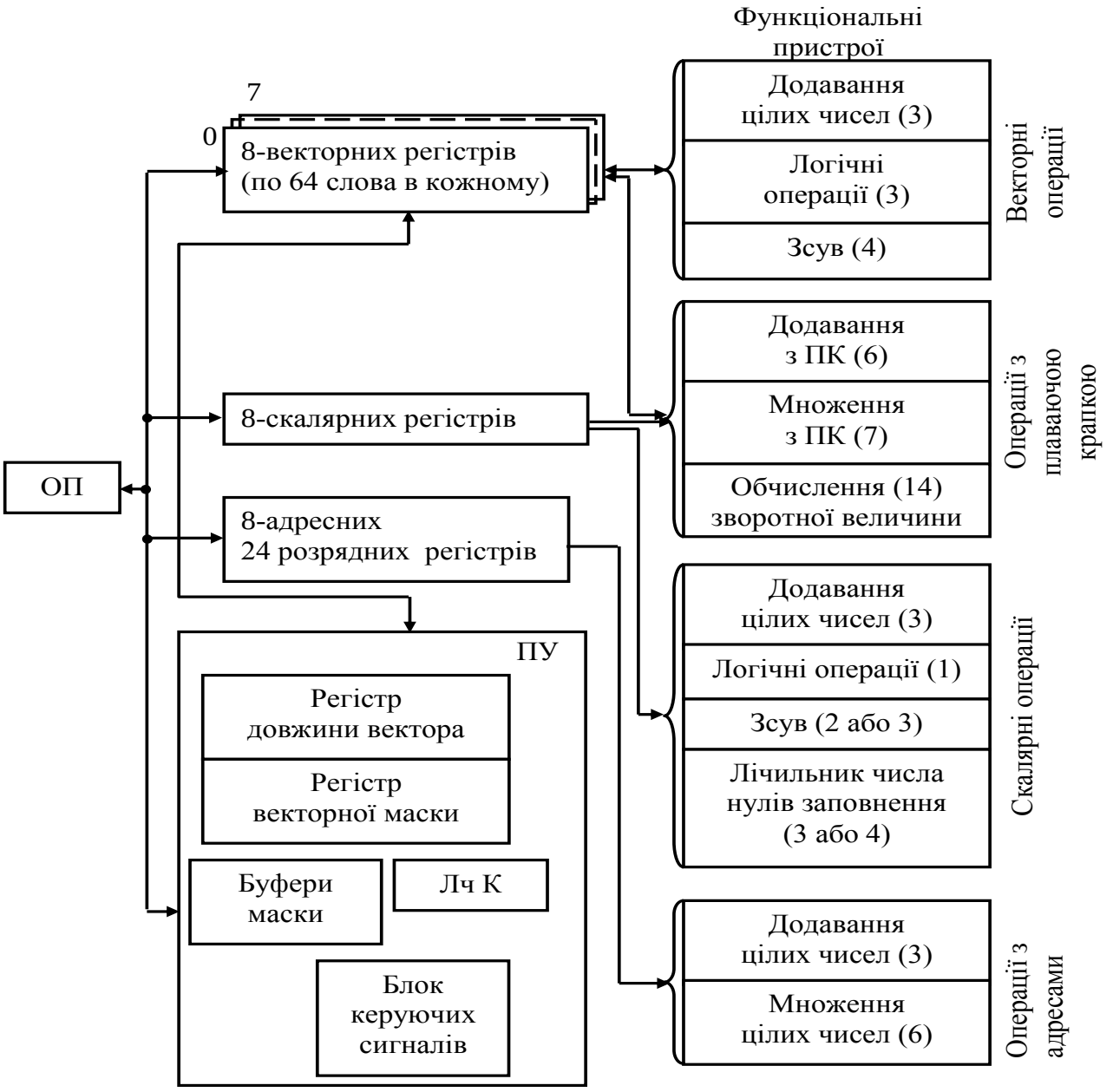

Рисунок 4 - Структура конвеєрно-векторної супер-ЕОМ (CRAY – 1)

Оскільки програми розв'язання науково-технічних задач, як правило, вимагають виконання як векторних, так і скалярних операцій, процесор має командні й апаратурні засоби для операцій двох типів. При цьому застосовані спеціальні заходи на зменшення часу виконання скалярних операцій, щоб останні вагомо не знижували загальну продуктивність системи.

Команди мають формат 16 або 32 розряди. Шістнадцятирозрядні команди основних, у тому числі векторних операцій мають 7-розрядне поле коду операції і три 3-розрядних поля для номера регістрів операндів і результату. В 32-розрядній команді під адресу основної пам’яті або безпосередній операнд відведено 22 розряди.

В системі реалізовані конвеєр команд і багаточисельні конвеєри для різних арифметичних і логічних операцій під час скалярного і векторного опрацюван-ня 64-розрядних слів з плаваючою комою (порядок - 15, мантиса – 49 розрядів), 64 – або 24 – розрядних слів з фіксованою комою, 22 – розрядних адрес.

Ці операції виконуються 12 конвеєризованими функціональними пристроями, які можуть працювати паралельно в часі, виконуючи різні операції. При цьому можливе утворення послідовних з'єднань конвеєрів, у тому числі так зване зачеплення векторних операцій, за яких отримані в ході виконання векторної команди результати через регістри (без використання пам'яті) подаються для використання наступною векторною командою. Конвеєри мають небагато робочих позицій (кількість вказана в дужках), з тим щоб зменшити втрати продуктивності, які пов'язані з періодом «розгону» конвеєра. Такт роботи конвеєра 12,5 нс.

Для підтримки конвеєрної і векторної обробки необхідні швидкі регістрові засоби, які звільняють від необхідності звернення до пам'яті під час виконання векторних операцій і забезпечують високий темп завантаження конвеєрів і інформаційні зв'язки між ними.

Процесор CRAY містить швидкі регістри (час звертання 6 нс): 8 векторних V – регістрів, кожний з яких може зберігати 64 64-розрядних слова; 8 24-розрядних адресних регістрів; 8 скалярних 64-розрядних S-регістрів.

Управляючий пристрій крім традиційних блоків містить чотири буфери команд (на 64 командних слова кожний) і спеціальні регістри: регістр довжини вектора і регістр векторної маски. Регістр довжини вектора фіксує довжину вектора S, який обробляється (завжди S<64). Регістр векторної маски, який містить 64 розряди, значенням свого і-го розряду визначає, із якого із регістрів, які беруть участь в операції, і-й елемент видається в регістр результату. По-іншому використовує регістр векторної маски команда перевірки елементів вектора, яка встановлює розряди вектора маски в 1, якщо відповідні елементи вектора задовольняють задану умову. Результат цієї команди може ефективно використовуватись іншими командами у ході опрацювання даних.

У пам'яті системи, яка має ємність від 1 до 4 М слів (слово 64 розряди + 9 контрольних), приблизно 16-кратне чергування адрес, за рахунок чого цикл звертання складає всього 50 нс.

В однопроцесорній ОС CRAY-1 досягнута продуктивність близько 80-130 Мфлоп/с. Для розгляду вибрана ця одна з перших конвеєрно-векторних супер-ЕОМ через її порівняну простоту і наочність відносно особливостей організації комбінацій конвеєрного і векторного опрацювання даних. У більш нових і більш потужних моделях фірми CRAY використовується від 2 до 8 процесорів, і продуктивність складає від 1000 до 2500 Мфлоп/с.

2 Матричні комп'ютерні системи

Призначення *матричних комп'ютерних систем* багато в чому схоже з призначенням векторних комп'ютерних систем – обробка великих масивів даних. В основі матричних систем лежить *матричний процесор*, який складається з регулярного масиву процесорних елементів (ПЕ). Організація системи такого типу, на перший погляд, достатньо проста. Вона має загальний управляючий пристрій, який генерує потік команд, та велику кількість ПЕ, які функціонують паралельно і обробляють кожний свій потік даних. Проте на практиці, щоб забезпечити достатню ефективність системи у ході розв'язання широкого кола задач, необхідно організовувати зв'язки між процесорними елементами так, щоб найбільш повно завантажити процесори роботою. Саме характер зв'язків між ПЕ і визначає різні властивості системи.

Матричний процесор інтегрує множину ідентичних функціональних блоків (ФБ), які логічно об'єднуються в матрицю і працюють у SIMD-стилі. Матриця процесорних елементів може бути конструктивно реалізована на одному кристалі або на декількох. Важливим є принцип функціонування – ФБ логічно скомпоновані у матрицю і функціонують синхронно, тобто існує тільки один потік команд для усіх блоків.

Структуру матричної комп'ютерної системи можна подати у виді, який показаний на рис 5.

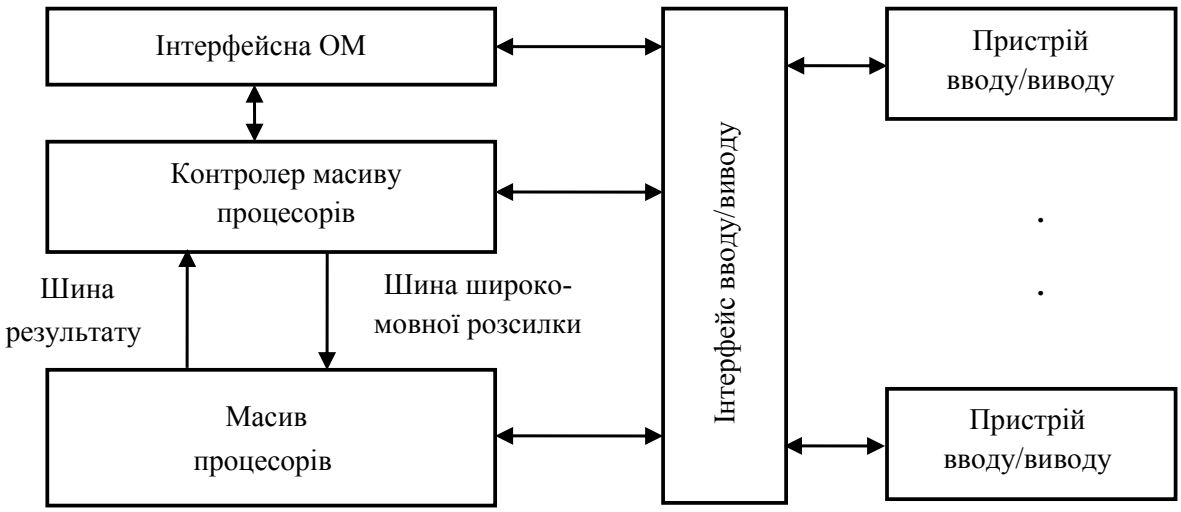

Рисунок 5 - Узагальнена модель матричної SIMD-системи

Власне паралельна обробка множинних елементів даних виконується *масивом процесорів* (МПр). Єдиний потік команд, який управляє обробкою даних у масиві процесорів, генерується *контролером масиву процесорів* (КМПр). КМПр виконує послідовний програмний код, реалізує операції умовного і безумовного переходів, транслює в МПр команди, дані та сигнали управління.

Команди обробляються процесорами в режимі жорсткої синхронізації. Сигнали управління використовуються для синхронізації команд і пересилок, а також для управління процесом обчислень, зокрема визначають, які процесори масиву повинні виконувати операцію, а які – ні. Команди, дані і сигнали управління передаються із КМПр у масив процесорів по *шині широкомовної розсилки*. Оскільки виконання операцій умовного переходу залежить від результатів обчислень, результати обробки даних у масиві транслюються в КМПр по *шині результату*.

Для забезпечення користувача зручним інтерфейсом під час створення та налаштування програм у склад подібних комп'ютерних систем звичайно включають *інтерфейсну обчислювальну машину* (ІОМ). Як таку обчислювальну машину використовують універсальну обчислювальну машину, на яку додатково покладається задача завантаження програм і даних в КМПр. Інтерфейсна обчислювальна машина функціонує під управлінням операційної системи, частіше це ОС UNIX. На ІОМ користувачі готують, компілюють і налаштовують власні програми. У процесі виконання програми спочатку завантажуються із інтерфейсної обчислювальної машини в контролер управління масивом процесорів, який виконує програму і розподіляє команди і дані по процесорних елементах масиву.

Крім того, завантаження програм і даних в КМПр може виконуватись і безпосередньо з *пристроїв вводу/виводу*, наприклад з магнітних дисків. Після завантаження КМПр приступає до виконання програми, транслюючи в МПр по широкомовній шині відповідні SIMD-команди.

Розглядаючи масив процесорів, слід враховувати, що для зберігання множинних наборів даних у ньому, крім множини процесорів, повинна бути присутньою і множина модулів пам'яті. Крім того, в масиві повинна бути реалізована мережа взаємозв'язків як між процесорами, так і між процесорами і модулями пам'яті. Таким чином, під терміном *масив процесорів* розуміють блок, який складається з процесорів, модулів пам'яті і мережі з'єднань.

У матричних SIMD-системах розповсюдження отримали два основних типи архітектурної організації масиву процесорних елементів.

У першому варіанті, який відомий як архітектура типу «*процесорний елемент − процесорний елемент*», *N* процесорних елементів (ПЕ) зв'язані між собою мережею з'єднань. Кожний ПЕ – це процесор з локальною пам'яттю. Процесорні елементи виконують команди, які отримують від КМПр по шині широкомовної розсилки, та обробляють дані як ті, що зберігаються у їх локальній пам'яті, так і ті, що потрапляють з КМПр. Обмін даними між процесорними елементами здійснюється по *мережі з'єднань*, в той час як *шина вводу/виводу* служить для обміну інформацією між ПЕ і пристроями вводу/виводу. Для трансляції результатів з окремих ПЕ в контролер масиву процесорів служить шина результату.

В багатьох алгоритмах дії по пересиланню інформації в більшості локальні, тобто виконуються між найближчими сусідами, тому дана архітектура, де кожний ПЕ зв'язаний тільки з сусідніми, достатньо ефективна.

Другий вид архітектури – «*процесор – пам'ять*». У такій архітектурі двонаправлена мережа з'єднань зв'язує *N* процесорів з *M* модулями пам'яті. КМП управляє процесорами через широкомовну шину. Обмін даними між процесорами здійснюється як через мережу, так і через модулі пам'яті. Пересилка даних між модулями пам'яті та пристроями вводу/виводу забезпечується шиною вводу/ виводу. Для передачі даних з певного модуля пам'яті в КМП служить шина результату.

Додаткову гнучкість під час роботи з матричною обчислювальною системою забезпечує механізм *маскування*, який дозволяє залучати до участі в операціях тільки певну підмножину процесорів з тих, які входять у масив. Маскування реалізується як на стадії компіляції, так і на етапі виконання, при цьому ті процесори, які виключаються шляхом встановлення в нуль відповідних бітів маски, протягом виконання команди простоюють.

3 Комп'ютерні системи з систолічною структурою

У фон-нейманівських комп'ютерах дані, які зчитуються з пам'яті, однократно обробляються в процесорному елементі, після чого знову повертаються в пам'ять (рис. 6, *а*). Автори ідеї систолічної матриці Кунг і Лейзерсон запропонували організувати обчислення так, щоб дані на своєму шляху від зчитування з пам'яті до повернення назад пропускалися через якамога більшу кількість ПЕ (рис. 6, *б*).

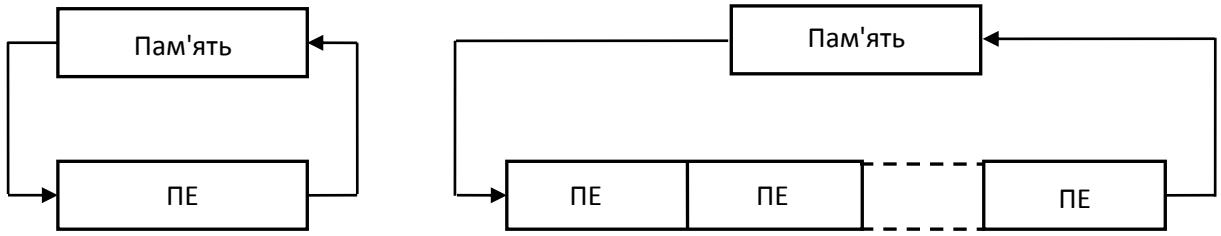

Рисунок 6 - Обробка даних у комп'ютерних системах:
а – фон-нейманівського типу; *б* – систолічної структури

Якщо порівняти положення пам'яті в комп'ютерній системі зі структурою живого організму, то за аналогією їй можна відвести роль серця, множині ПЕ – роль тканин, а потік даних розглядати як кров, що циркулює. Звідси і виходить назва *систолічна матриця* (систола – скорочення передсердя і шлуночків серця, при якому кров нагнітається в артерії). Систолічні структури ефективні під час виконання матричних обчислень, обробки сигналів, сортування даних і так далі.

Таким чином, *систолічна структура* – це однорідне обчислювальне середовище з процесорних елементів, яке суміщає в собі властивості конвеєрної та матричної обробки і володіє такими особливостями:

- обчислювальний процес у систолічних структурах являє собою неперервну і регулярну передачу даних від одного ПЕ до іншого без запам'ятовування проміжних результатів обчислення;
- кожний елемент вхідних даних вибирається з пам'яті однократно та використовується стільки разів, скільки необхідно за алгоритмом, введення даних здійснюється в крайні ПЕ матриці;
- ПЕ, які створюють систолічну структуру, однотипні і кожний з них може бути менш універсальним, ніж процесори звичайних багатопроцесорних систем;
- потоки даних та управляючих сигналів володіють регулярністю, що дозволяє об'єднувати ПЕ локальними зв'язками мінімальної довжини;
- алгоритми функціонування дозволяють суміщати паралелізм з конвеєрною обробкою даних;
- продуктивність матриці можна покращувати за рахунок додавання до неї певного числа ПЕ, причому коефіцієнт підвищення продуктивності при цьому є лінійним.

У теперішній час продуктивність систолічних процесорів перевищує 1000 млрд операцій /с.

Аналіз різних типів систолічних структур та тенденцій їх розвитку дозволяє класифікувати ці структури по декільком ознакам.

За ***ступенем гнучкості*** систолічні структури можуть бути згруповані так:

- спеціалізовані;
- алгоритмічно орієнтовані;
- програмовані.

Спеціалізовані структури орієнтовані на виконання певного алгоритму. Ця орієнтація відображається не тільки в конкретній геометрії систолічної структури, статичності зв'язків між ПЕ та кількості ПЕ, але і в виборі типу операції, що виконується усіма ПЕ. Прикладами є структури, які орієнтовані на рекурсивну фільтрацію, швидке перетворення Фур'є для заданої кількості точок, матричні перетворення.

Алгоритмічно орієнтовані системи володіють можливістю програмування або конфігурації зв'язків у систолічній матриці або самих ПЕ. Можливість програмування дозволяє виконувати на таких структурах деяку множину алгоритмів, які зводяться до однотипних операцій над векторами, матрицями та іншими числовими множинами.

У *програмованих систолічних структурах* є можливість програмування як самих ПЕ, так і конфігурації зв'язків між ними. При цьому ПЕ можуть володіти локальною пам'яттю програм. Команди, які зберігаються в пам'яті таких ПЕ, можуть змінювати і напрямок передачі операндів.

За ***розрядністю процесорних елементів*** систолічні структури діляться так:

- одноразрядні;
- багаторозрядні.

В одноразрядних матрицях ПЕ в кожний момент часу виконує операцію над одним двійковим розрядом, а в багаторозрядних – над словами фіксованої довжини.

За ***характером локально-просторових зв'язків*** систолічні структури бувають:

- одновимірні;
- двовимірні;
- тривимірні.

Вибір структури залежить від виду інформації, що обробляється. Одновимірні структури застосовують при обробці векторів, двовимірні – матриць, тривимірні – множин іншого типу.

У теперішній час розроблені систолічні матриці з різною геометрією звязків: лінійні, прямокутні, гексагональні, тривимірні та ін. (рис. 7).

Кожна конфігурація матриці найбільш пристосована для виконання певних функцій, наприклад лінійна матриця оптимальна для реалізації фільтрів в реальному масштабі часу; гексагональна – для виконання операцій обернення матриць, а також для операцій над матрицями спеціального типу. Найбільш універсальними і розповсюдженими є матриці з лінійною структурою.

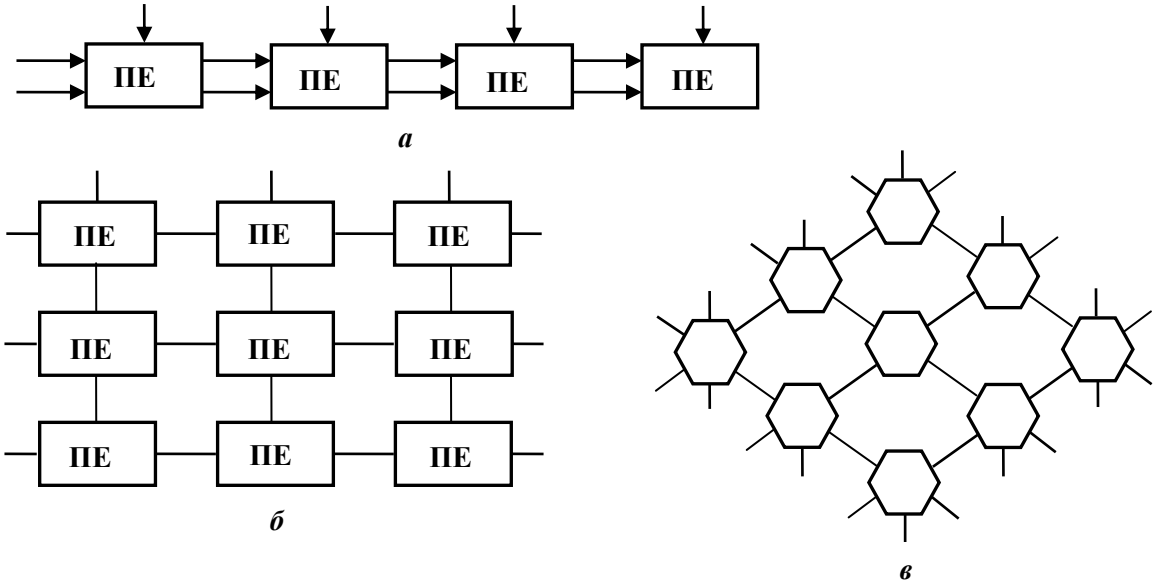

Рисунок 7 - Конфігурація систолічних матриць:
а – лінійна; *б* – прямокутна; *в* – гексагональна

Для розв'язання складних задач конфігурація систолічної структури може являти собою набір окремих матриць, складну мережу взаємозв'язаних матриць або оброблювану поверхню. Під *оброблюваною поверхнею* розуміють безкінечну прямокутну мережу ПЕ, де кожний ПЕ є з'єднаним зі своїми чотирма сусідами (або більшим числом ПЕ).

4 Обчислювальні системи з командними словами надвеликої довжини (VLIW)

VLIW (Very Long Instruction Word) – це набір команд, які організовані аналогічно організації горизонтальної мікрокоманди в мікропрограмному пристрої управління.

Ідея VLIW базується на тому, що задача ефективного планування паралельного виконання декількох команд покладається на «розумний» компілятор. Такий компілятор спочатку досліджує початкову програму з метою виявити всі команди, які можуть бути виконані одночасно, причому так, щоб це не приводило до виникнення конфліктів. У процесі аналізу компілятор може навіть частково імітувати виконання програми, яка розглядається. На наступному етапі компілятор намагається об'єднати такі команди в пакети, кожний з котрих розглядається як одна наддовга команда. Об'єднання декількох простих команд в одну наддовгу виконується за такими правилам:

- кількість простих команд, які об'єднуються в одну команду надвеликої довжини, дорівнює числу функціональних (виконавчих) блоків (ФБ);
- у наддовгу команду входять тільки прості команди, які виконуються різними ФБ, тобто забезпечується одночасне виконання усіх складових наддовгої команди.

Довжина наддовгої команди звичайно складає від 256 до 10 24 біт. Така *метакоманда* містить декілька полів (по числу простих команд, які її утворюють), кожне з яких описує операцію для конкретного функціонального блока. На рис. 8 показаний можливий формат наддовгої команди та взаємозв'язок між її полями і ФБ, які реалізують окремі операції.

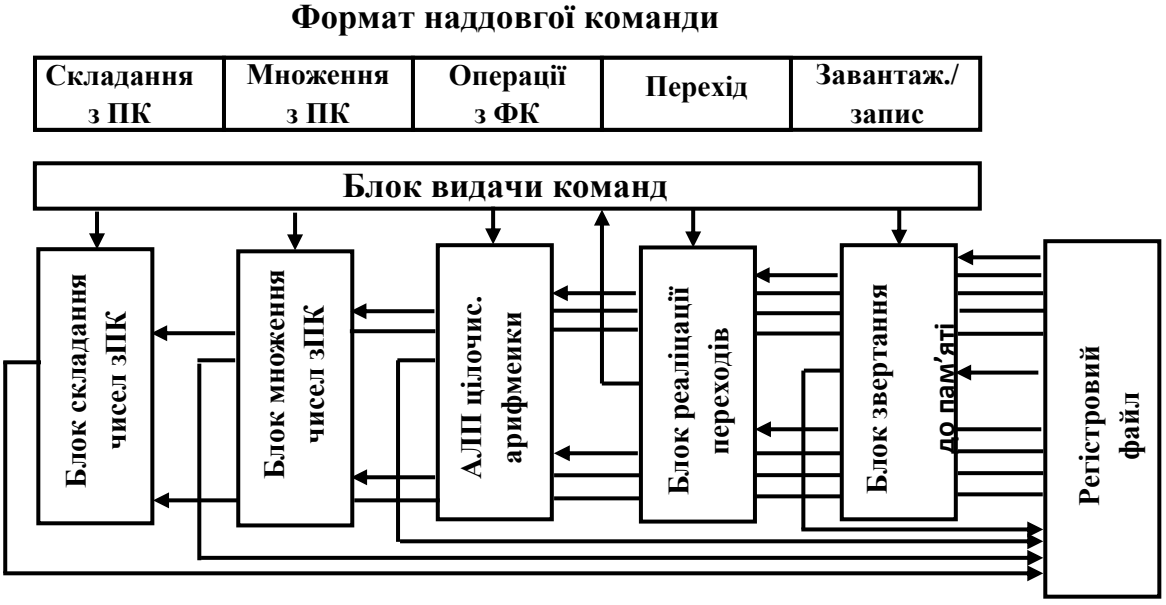

Рисунок 8 - Формат наддовгої команди та взаємозв'язок полів команди зі складовими блока виконання

Як видно з рисунка, кожне поле наддовгої команди відображається на свій функціональний блок, що дозволяє отримати максимальну віддачу від апаратури блока виконання команд.

VLIW-архітектуру можна розглядати як статичну суперскалярну архітектуру. Мається на увазі, що розпаралелювання коду здійснюється на етапі компіляції, а не динамічно в час виконання. Як уже було відмічено, у виконуваної наддовгої команди виключена можливість конфліктів, а це дозволяє значно спростити апаратуру VLIW-процесора та збільшити швидкодію.

Як прості команди, що створюють наддовгу, звичайно використовують команди RISC-типу, тому архітектуру VLIW іноді називають постRISC-архітектурою. Максимальне число полів у наддовгій команді дорівнює числу обчислювальних засобів і звичайно знаходиться в діапазоні від 3 до 20 . Усі обчислювальні засоби мають доступ до даних, які зберігаються у єдиному багатопортовому регістровому файлі. Відсутність складних апаратних механізмів, які є характерними для суперскалярних процесорів (передбачення переходів, позачергове виконання і ін.), дає значний виграш у швидкодії та можливість більш ефективно використовувати площину кристалу. Переважна більшість цифрових сигнальних процесорів та мультимедійних процесорів з продуктивністю більш 1 млрд операцій/с базується на VLIW-архітектурі. Серйозна проблема VLIW-архітектури – ускладнення регістрового файла та зв'язків цього файла з обчислювальними пристроями.