УДК 621.37

DOI: 10.20535/2617-0965.eae.228485

Алгебраїчна характеристика класу матричних перетворень та її апаратна реалізація

Кудлай С. В., ORCID 0000-0003-3972-405X
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського», ROR 00syn5v21
Київ, Україна

***Анотація*—У даній роботі на основі примітивних програмних алгебр виводиться алгебраїчна характеристика класу матричних перетворень методом ізоморфних відображень на алгебраїчну характеристику классу векторних перетворень. В роботі також описано апаратну реалізацію прискорювача матричних операцій на основі отриманих даних. Актуальність роботи обумовлена тим, що сьогодні відбувається стрімке проникнення комп'ютерних технологій в усі сфери життєдіяльності соціуму і, як наслідок, кількість даних, які потрібно опрацювати за одиницю часу постійно зростає. Велика кількість задач, пов'язаних зі значними об'ємами складних обчислень вирішується методами, що грунтуються на матричних операціях. Тож дослідження матричних обчислень та їх прискорення є досить важливою задачею. У даній роботі в якості внеску в даному напрямку пропонується дослідження класу матричних перетворень за допомогою сигнатурних операцій примітивної програмної алгебри – багатомісної суперпозиції, галуження й циклування, що представляють собою уточнення найбільш поширених управлінських структур у більшості високорівневих мов програмування, а також ізоморфного відображення. Сигнатурні операції примітивної програмної алгебри в поєднанні з базовими частково-рекурсивними матричними функціями та предикатами дозволяють реалізувати множину усіх частково-рекурсивних матричних функцій та предикатів. Отримано результат про базис матричної примітивної програмної алгебри. Ізоморфізм забезпечує відтворення частково-рекурсивних функцій та предикатів для матричних перетворень як образів частково-рекурсивних векторних функцій та предикатів. Повноту алгебраїчної системи матричних перетворень забезпечено завдяки наявним результатам по виведенню повноти алгебраїчної системи для векторних перетворень. Створена іменна модель матричних даних, оптимізована під використання в розробці апаратної реалізації. В апаратній реалізації забезпечена підтримка сигнатурних операцій примітивної програмної алгебри та реалізована апаратна підтримка ізоморфного відображення. Реалізована апаратна підтримка функцій суми, множення та транспонування матриць, а також предиката рівності двох матриць. Підтримка сигнатурних операцій примітивної програмної алгебри забезпечується проектуванням контролюючої частини матричного обчислювального апарату на основі архітектури RISC. В основі апаратної підтримки ізоморфізму лежать лічильники, вони дозволяють інтуїтивно реалізувати циклування у функціях ізоморфних відображень. Швидке виконання векторних операцій забезпечується принципом комп'ютерних обчислень SIMD.**

Ключові слова — матричні обчислення; ізоморфізм; алгебраїчна характеристика; обробка матриць; комп'ютерна архітектура; матричний процесор.

I. Вступ

Потреба в швидкому опрацюванні великих об'ємів даних з кожним роком стає дедалі більшою. Загалом значущість обробки великих об'ємів структурованої інформації, зокрема і матричних обчислень важко переоцінити. Тому апаратна реалізація таких базових функцій здатна істотно збільшити швидкодію систем обробки великих об'ємів структурованих даних, що, з огляду на вище зазначене, є вкрай актуальним.

II. Позначення та визначення

Для дослідження класу обчислюваних матричних функцій та предикатів було використано метод ізоморфних відображень [1], а саме побудовано ізоморфізм між примітивними програмними алгебра ми (ППА) векторних та матричних чр-функцій та чр-предикатів. Під вектором у якості структури даних розуміється кортеж чисел <a1, … as>, де ai, i = 1, 2,…, s – елементи кортежа. Множину векторів натуральних чисел довжиною s позначатимемо:

$$N^s \equiv \underbrace{N \times ... \times N}_{s}$$

Множина всіх можливих векторів позначається як:

$$N^* \equiv \bigcup_i N^i \ , \ i \in N \cup \{0\}$$

При цьому $N^0 \equiv \{\Lambda\}$, Λ – кортеж нульової довжини або так званий пустий кортеж.

DOI: 10.20535/2617-0965.eae.228485

Під матрицею (натуральних чисел) розміру $m\times n$, $(m,n=1,2,...)$ розуміється відображення $\{1,...,m\}\times\{1,...,n\}\to N$. Множину всіх матриць розміру $m\times n$ позначимо $\mathrm{M}_{m,n}$, а Λ – означає «порожню» матрицю, по аналогії з уведеним вище «порожнім» кортежем. Покладемо

$$\mathrm{M}\equiv\{\Lambda\}\cup\bigcup_{m,n=1}^{\infty}\mathrm{M}_{m,n}$$

множина всіх матриць. Матриці позначатимемо прописними буквами $A,B,C,...$, їх елементи – відповідними рядковими буквами з індексами. Наприклад, $A=(a_{ij})_{mn}$ – позначення матриці розмірності $m\times n$.

Сигнатуру ППА складають три типи операцій – суперпозиція $S^{n+1}(f^n,f_1^m,...,f_n^m)$, галуження $\Diamond(p,f_1,f_2)$ та циклування $*(p^m,f_1^m,...,f_n^m)$ багатомісних функцій, де n,m – арності відповідних чр-функцій $f^n,f_i^m|_{i=\overline{1,n}}$ та предикатів p^m, відповідно, над носіями N^* та M [2]. Дані операції є строгими уточненнями широко застосовуваних прийомів у більшості парадигм програмування. Це безпосередньо випливає з їх визначень, наведених у [2]. Примітивні програмні алгебри векторних чр-функцій та чр-предикатів і матричних чр-функцій та чр-предикатів (далі, також векторна та матрична ППА) позначатимемо $A^{чр}_{N^*}$ і $A^{чр}_{M}$.

III. ППА ЧР-ФУНКЦІЙ ТА ЧР-ПРЕДИКАТІВ НАД МАТРИЦЯМИ

У роботі [3] отримано результат про I^m_n-базис векторної ППА $A^{чр}_{N^*}$, згідно з яким система функцій та предикатів $\sigma_{N^*}\equiv\{\Pi,\circ,C_0,S,=_{N^*},I^n_m\}$ де $\circ$ – позначає собою конкатенацію кортежів. Через Π,C_0,S позначимо наступні векторні функції:

$$\Pi(\Lambda)=S(\Lambda)=\Lambda,\ C_0(u)=0,\ S(au)=(a+1)u,$$

Λ – «пустий» кортеж, тобто такий, що не має елементів, а I^m_n є базисом, тобто нескорочуваною системою породжуючих алгебри $A^{чр}_{N^*}$. Спираючись на цей результат та використовуючи згаданий метод ізоморфних відображень були розглянуті кодуюче та декодуюче відображення $\psi:N^*\to M$ та $\phi:M\to N^*$, відповідно, побудовані їх матричні моделі, тобто побудований ізоморфізм векторної ППА на матричну ППА. Це дозволило побудувати систему породжуючих $\sigma_M\equiv\{\Pi_M,\circ_M,C_{(0)_{11}},S_M,=_M,\psi,\chi,I^n_m\}$ ППА $A^{чр}_{M}$ як ізоморфних матричних образів наведених вище чр-функцій та чр-предикатів з $\sigma^{чр}_{N^*}$. А саме, Π_M – функція видалення першого стовпчика матриці:

$$\Pi_M((a_{ij})_{mn})=\begin{cases}(a_{ij+1})_{mn-1}, якщо\cdot n\ge 2\\ \Delta, інакше\end{cases}.$$

$\circ_M$ – т.з. горизонтальна конкатенація двох матриць з однаковою кількістю рядків: $(a_{ij})_{mn}\circ_M(b_{ij})_{mk}=(c_{ij})_{mn+k}$, де

$$c_{ij}=\begin{cases}a_{ij},1\le j\le n\\ b_{sj},n+1\le j\le n+k\end{cases}.$$

Крім того, $A\circ_M\Delta=\Delta\circ_M A=A$; $C_{(0)11}:C_{(0)11}(A)=(0)_{11}$ – співставлення будь-якій матриці A нульової матриці $(0)_{11}$; S_M – збільшення 1-го стовпчика матриці на одиницю: $S_M((a_{ij})_{mn})=(b_{ij})_{mn}$, де $b_{i1}=a_{i1}+1$, а $b_{ij}=a_{ij}$, для $i=1,...,m$, та $j=2,...,n$; $=_M$ – предикат по елементної рівності двох матриць:

$$(a_{ij})_{mn}=_M(b_{ij})_{mn}\equiv\begin{cases}True,\forall i=\overline{1,m};j=\overline{1,n}\ a_{ij}=a_{ij}\\ False, інакше\end{cases}.$$

Ψ – векторний образ матричної функції $F_M(\varsigma_1,...,\varsigma_n)$, якщо $F_{N^*}(\phi(A_1),...,\phi(A_n))\cong\phi(F_M(A_1,...,A_n))$ для будь-яких $A_1,...,A_n$, й аналогічно для предикатів$ χ – розширення відображення ψ^{-1}, $F_M(A_1,...,A_n)\cong\chi(H_M(\psi(A_1),...,\psi(A_n)))$. Матричні функції ψ та χ відіграють ролі кодуючої та декодуючої функцій відповідно.

З побудови $\sigma^{чр}_M$ безпосередньо випливає її нескорочуваність. Таким чином, справедливим є насупне твердження.

Твердження. $\sigma^{чр}_M$. – I^m_n-базис ППА $A^{чр}_M$.

IV. ПРИКЛАДИ РЕАЛІЗАЦІЇ ТРАДИЦІЙНИХ МАТРИЧНИХ ОПЕРАЦІЙ

Тут продемонструємо можливості ППА $A^{чр}_M$ на низці традиційних матричних операцій. Для зручності подальшого викладення, покажемо, що тут легко виразити також основні логічні операції. Почнемо з найважливіших серед них – константних предикатів тотожної істинності $P_T(x)$ та тотожної хиби $P_F(x)$:

$$P_T(x)=\Diamond(<x=x,T,F>)$$

та

$$P_F(x)=\Diamond(<C_{(0)_{11}}(x)=S_M(C_{(0)_{11}}(x)),T,F>).$$

Тоді, наприклад, логічну зв'язку «або» – $\vee$ для двох предикатів $P(x_1,..,x_n)|_{n=1,2,...}$ та $Q(y_1,..,y_m)|_{m=1,2,...}$ можна представити так:

DOI: 10.20535/2617-0965.eae.228485

$$P(x_1,..,x_n) \vee Q(x_1,..,x_n) =$$
$$= \Diamond\left(P(x_1,..,x_n), P_T(x_1), \Diamond\left(Q(x_1,..,x_n), P_T(x_1), P_F(x_1)\right)\right).$$

Для зв'язок $\neg$ та $\wedge$ представлення в ППА здійснюється аналогічно. Тепер перейдемо до матричних операцій. Задля компактності викладення матеріалу будемо використовувати тут термальну форму запису операцій та матриць, маючи на увазі при цьому їх іменні специфікації.

Розглянемо матричну функцію E, що для будь-якої матриці-рядка обирає її перший елемент, тобто $E([[a_{11},..,a_{1n}]]) = [a_{11}]$. Нехай,

$$F(x_1,x_2) = (x_1 \circ \Pi_M(x_2)) \underset{x_1}{*} (S_M(x_1)).$$

Тоді $E(x) = F(C_{(0)_{11}}(x), x)$.

Функція передслідування $P(x)$, де x – матриця-рядок і така, що $P([a_{11},..,a_{1n}]) = [a_{11}-1,..,a_{1n}]$. Нехай $F(x_1,x_2) = (S_M(x_1) \circ \Pi_M(x_2) \neq x_2) \underset{x_1}{*} (S_M(x_1))$. Тоді $P(x) = F(C_{(0)_{11}}(x), x)$, якщо $a_{11} \geq 1$ та $P(x) = \perp$, $a_{11} = 0$. Адже тоді для a_{11} не існує попереднього елементу.

Транспонування матриці-рядка T_{str}: $T_{str}(x) = \chi(S_M(C_0(x) \circ \Pi_M(\psi(x))))$. Тобто, $T_{str}([a_{11},..,a_{1n}]) = \begin{bmatrix} a_{11} \\ ... \\ a_{1n} \end{bmatrix}$.

Функція E_M^T, що виділяє з матриці $x = \begin{bmatrix} a_{11}...a_{1n} \\ ... \\ a_{m1}...a_{mn} \end{bmatrix}$ перший рядок та транспонує його: $E_M^T(x) = \begin{bmatrix} a_{11} \\ ... \\ a_{1n} \end{bmatrix}$. Нехай

$$G(n,x_1,x_2) = (n \neq C_{(0)_{11}}) \underset{x_2,n,x_1}{*}$$
$$*(x_2 \circ E(x_1), P(n), (\Pi_M(x_1)))$$

де x_1, x_2 – матриці-рядки, а n – одноелементна матриця $\underbrace{S_M(...(S_M}_{n}(C_{(0)_{11}}))...)$. Тоді $E_n(n,x) = G(n,x,\Pi_M(n))$ виділяє перші n елементів матриці-рядка x: $E_n(3,[1,2,3,4,5]) = [1,2,3]$. Введемо тепер функцію $\hat{E}_M(x) = E_n(E_n(1,\psi(x)), \Pi_M(\psi(x)))$, що матриці-рядку $[a_{11},..,a_{1n}]$ ставить у відповідність матрицю-рядок $[a_{12},..,a_{1a_{11}}]$. Тобто, $\hat{E}_M([3,2,3,4,5]) = [2,3,4]$. Тепер, $E_M^T = \chi(S_M(C_{(0)_{11}}(x)) \circ_M (x))$. Таким чином, $E_M^T\left(\begin{bmatrix} a_{11}...a_{1n} \\ ... \\ a_{m1}...a_{mn} \end{bmatrix}\right) = \begin{bmatrix} a_{11} \\ ... \\ a_{1n} \end{bmatrix}$

Перейдемо до операції транспонування T_M. Для цього, спочатку побудуємо операцію видалення першого рядка матриці $D(x)$:

$$D(x) = \begin{bmatrix} a_{21}...a_{2n} \\ ... \\ a_{m1}...a_{mn} \end{bmatrix}, \text{ де } x = \begin{bmatrix} a_{11}...a_{1n} \\ ... \\ a_{m1}...a_{mn} \end{bmatrix}.$$

Візьмемо функцію $Q(n,x) = (n = C_{(0)_{11}}(x)) \underset{x,n}{*} (\Pi_M(x), P(n))$, що з матриці-рядка x видаляє перші n елементів. $Q(3,[3,2,3,4,5]) = [4,5]$. Використаємо її для побудови функції $\hat{D}(x) = E_M^T(x) \circ_M Q(E_M^T(x), \Pi_M(x))$:

$$\hat{D}([4,\underbrace{2,3,4,5}_{4},6]) = [4,6],$$

$$\hat{D}([2,\underbrace{2,}_{2}3,4,5,6]) = [2,4,5,6],$$

$$\hat{D}([5,\underbrace{2,3,4,5,6}_{5}]) = [5]$$

з матриць-рядків $[4,2,3,4,5,6]$, $[2,2,3,4,5,6]$ та $[5,2,3,4,5,6]$ було видалено починаючи з 2-го кількість елементів, вказана у першому елементі матриці-рядка.

Таким чином, $D(x) = \chi(\hat{D}(x))$.

Розглянемо ще одну важливу допоміжну функцію $\hat{T}(x_1,x_2) = (x_2 \neq \Pi_M(C_{(0)_{11}}(x_2))) \underset{x_1,x_2}{*} (x_1 \circ E_M^T(x_2), D(x_2))$. Дана функція для двох підходящих матриць здійснює конкатенацію першої з них та транспонованої другої матриці. Наприклад,

$$\hat{T}\left(\begin{bmatrix} 1,1,3 \\ 2,2,7 \end{bmatrix}, \begin{bmatrix} 2,3 \\ 4,5 \\ 6,7 \end{bmatrix}\right) = \begin{bmatrix} 1,1,3 \\ 2,2,7 \end{bmatrix} \circ_M \begin{bmatrix} 2,4,6 \\ 3,5,7 \end{bmatrix} =$$
$$= \begin{bmatrix} 1,1,3,2,4,6 \\ 2,2,7,3,5,7 \end{bmatrix}.$$

Під кінець, $T(x) = \hat{T}(\Pi_M(C_{(0)_{11}}(x)), x)$ – операція транспонування матриці x.

Ці приклади свідчать про досить високу виразну силу системи породжуючих σ_M. При цьому, всі отримані функції можуть бути використані при побудові з них більш складних, більш важливих. Тобто ці, навіть проміжні, рішення поповнюють загальний «фонд» операцій над матрицями і можуть бути

DOI: 10.20535/2617-0965.eae.228485

використані стільки разів у побудовах, скільки це необхідно.

V. Іменна модель даних типу матриць

Для визначення структурних типів даних, зокрема матриць та маніпуляцій над їх елементами, була використана модель іменних множин [4]. У першому наближенні, іменна множина представляє собою скінчену множину пар виду *(M,x)*, де *M* – ім'я, зокрема, адреса зберігання значення в пам'яті комп'ютера, *x* – саме значення, або денотат імені *M*. При цьому, суттєвим для іменної моделі даних є дотримання принципу іменування, суть якого полягає в тому, що одне й те саме ім'я не може зустрічатися у іменній множині більше одного разу. Використання іменної моделі даних цілком відповідає фізико-технологічній інтерпретації іменної множини як ділянки пам'яті комп'ютера.

В нашому випадку для зберігання імен матриць-змінних використовується окрема від елементів матриці пам'ять, яка називається іменною пам'яттю. Обробка імен та елементів також відбуваються різними модулями. Тож дана іменна модель повинна при мінімальній кількості метаданих забезпечити повне відтворення матриці з пам'яті. Мінімальна кількість метаданих дозволить забезпечити простоту обробника імен, при цьому даний набір метаданих дозволяє коректно зчитувати елементи матриць та є оптимізованим під залізо.

Тут ім'я матриці містить кількість рядків, стовпчиків та адресу початкового елементу. Даний набір дозволяє повністю відтворити матрицю з пам'яті елементів без втрати розмірності. Слід зазначити, що кількість байт для зберігання одного елемента є фіксованою.

Отже іменна модель матриці в нашому випадку являє собою пару кортежів ($<A_0,m,n>,<a_{i},\dots a_{N}>$), де A_0 – адреса початкового елементу, m,n – кількість рядків та стовпчиків відповідно, $a_i,i=(1,2,\dots,N)$ – елементи матриці.

VI. Апаратна реалізація

В апаратній реалізації використано принципи логіко-предметної архітектури, в основі якої лежить розділення виконавчої та керуючої частин з їх прямим та зворотнім зв'язком.

На рисунку 1 показана блокова схема матричного прискорювача. Блок I/O Memory Controller розподіляє дані з входу пристрою до трьох блоків пам'яті. Блок пам'яті Instruction memory зберігає інструкції, які виконуються прискорювачем. Name model memory зберігає іменні моделі матриць. Matrix memory зберігає елементи матриці. Control unit послідовно обробляє інструкції та декодує їх в команди для calculate unit, а також саме цей блок перевіряє та обробляє іменні множини. Calculate unit в свою чергу виконує основні матричні операції з елементами матриць.

За порядок слідування кроків, індукованих програмою матричної обробки та обробки імен матриць відповідає контролюючий блок. Його конструкція базується на основі ядра MIPS, тож йому властиві особливості RISC архітектури [5] (див. 734 с.). Зокрема довжина однієї інструкції має фіксований розмір, а саме ядро конвеєрну основу.

За послідовне виконання програми відповідає так званий програмний лічильник. З кожним тактом він збільшує своє значення для зчитування наступної інструкції. В кожній комірці програмної пам'яті міститься одна інструкція. Збільшення програмного лічильника на одиницю вилучає з пам'яті програми наступну інструкцію. Таким чином реалізується операція суперпозиції ППА. Операції гілкування та циклування реалізуються завдяки інструкціям типу «стрибок» та предикативною операцією.

Керуючий блок підтримує 4 типи операцій: читання з пам'яті іменних моделей (Read), «стрибок» (Jump), операції над матрицями (Exec) та операції з перериваннями (IR). Операції типу IR здатні як генерувати переривання, так призупиняти виконання програми для очікування надходження даних від зовнішнього пристрою.

В пам'яті матриця зберігається у вигляді витягнутого вектора. Даний вектор утворюється ітеративним приєднанням до першого стовпчика інших. Задля багатопоточної обробки даних в одній комірці пам'яті матриць зберігається декілька елементів матриці. Таким чином реалізується принцип виконання команд Single Instruction Multiple Data (SIMD). [5] (див. 1112 с.). Даний принцип дозволяє виконувати одноманітні дії над відповідними елементами двох кортежів за один такт. Далі розмір такого кортежу називатиметься довжиною вектора (k). Також слід зазначити, що для апаратної реалізації зручно, коли довжина вектора кратна двійці. Таке значення параметру дозволяє ефективно зчитувати пам'ять та позбутися деяких логічних елементів (множення на число кратному двійці тотожньо зсуву на кратність двійки).

При зберіганні в одній комірці декількох елементів матриці, сама матриця в пам'яті виглядає наступним чином:

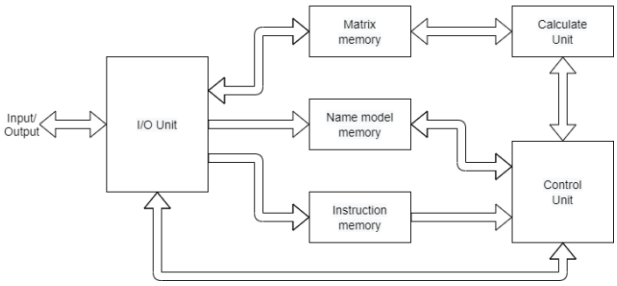

Рис. 1 – Схема матричного обчислювача.

{a[1][1], a[2][1], …, a[k][1]}
{a[k+1][1], a[k+2][1], …, a[2k][1]}
⋮
{… , a[m][1], X}
{a[1][2], a[2][2], …, a[k][2]}
{a[k+1][2], a[k+2][2], …, a[2k][2]}
⋮
{…, a[m][2], X}

Рис. 2 – Зберігання елементів матриці в пам'яті.

DOI: 10.20535/2617-0965.eae.228485

На рисунку 2 *X* позначає сміттєве значення, тобто яке не враховується при обчисленнях. Присутність такого значення в кінці стовпчику обумовлена тим, що кількість рядків може бути не кратною *k*. В іншому випадку сміттєвого значення не буде.

Виконавчий блок також має логіко-предметну архітектуру та поділений на дві частини: блок підтримки ізоморфізму (БПІ) та векторний обчислювач. Векторний обчислювач виходячи з назви опрацьовує операції з внутрішніми векторами. БПІ керує векторним обчислювачем, задаючи операції, які необхідно зробити над векторами; також він обчислює адреси внутрішніх векторів, які потрібно вилучити або записати до пам'яті.

БПІ на вхід приймає імена матриць-операндів, матриці для запису результату, саму операцію та статус векторного обчислювача. У якості виходу блока слугують команда векторному обчислювачу, адреси зчитування й запису даних та статус обчислень для контролюючого блоку. Імена матриць використовуються для розрахунку адрес зчитування та запису даних.

В основі розрахунку адрес лежать лічильники [5] (див. 649 с.). При розробці БПІ використовується декілька модифікацій лічильника, зокрема з заданим значенням переповнення та з заданим кроком. Лічильники при досягненні значення переповнення скидаються в нуль. Значення на виході лічильників лежать в межах [0; *N*-1], де *N* позначає значення переповнення, яке задається лічильнику. Заданий крок дозволяє змінити крок лічення, що звільняє від потреби в апаратних блоках множення. Лічильники зі сталим кроком лічення відповідають за стовпчики, а зі змінним за рядки. До адреси A_0 додається значення двох лічильників, вказуючи на певний елемент матриці. Таким чином відбувається кодування двомірної адреси в одномірну. Мультиплексори на вході суматорів дозволяють обрати різні лічильники, чим досягається гнучкість БПІ. За вибір лічильника, а також за дозвіл лічення відповідає спеціальна функціональна керуюча схема.

Завдяки властивим для матриці умовам для виконання операцій можна значно зекономити ресурси. Так, наприклад, для додавання/віднімання, або перевірки на рівність двох матриць вистачить лише два лічильника для рядка та стовпчика. Переповнення лічильника стовпчику рахується за формулою $ceil(m/k)$ (кількість комірок для зберігання одного стовпчика), що обумовлено зберіганням в одній комірці *k* елементів. Лічильник рядків має крок рівний переповненню лічильника стовпчиків. А значення переповнення обчислюється як $n \cdot ceil(m/k)$ (кількість комірок для зберігання елементів матриці).

В системи лічильників є 3 режими роботи, які визначаються операцією. До них належать: лінійний перелік елементів, множення двох матриць та транспонування. Керуючі сигнали кожного режиму задаються вищезазначеною керуючою схемою. Так, наприклад, при лінійному переліку лічильник рядків лічить лише в момент переповнення лічильника стопвчиків. При множенні матриць лічильник стовпчиків матриці *A* лічить в момент переповнення відповідного лічильнику стовпчиків. Лічильники матриці *B* працюють як і у випадку з лінійним проходженням, тільки лічення стовпчиків відбувається через кожні *k* тактів. В основі транспонування усієї матриці стоїть така ж дія для квадратної підматриці. Тож лічильники ітеративно формують дані під матриці до кінця самої матриці. Для відлічення тактів використовується лічильник тактів, який переповнюється кожні *k* тактів.

Схема обчислень відповідає вже безпосередньо за арифметичні дії. Поки наявні по елементні операції, множення матриць та транспонування. Також мається порівняння двох матриць на рівність, де порівнюються відповідні комірки до першої нерівності. В основі множення матриць лежить множення рядка матриці *A* на відповідний стовпчик матриці *B*. Дана операція за принципом виконання схожа на операцію згортки.

$$C_{(1,k)j} = \begin{pmatrix} A_{1(1,m)} \times B_{(1,k)j} \\ \vdots \\ A_{i(1,m)} \times B_{(1,k)j} \\ \vdots \\ A_{k(1,m)} \times B_{(1,k)j} \end{pmatrix} \quad (1)$$

де $C_{(1,k)j}$ – *j*-й стовпчик матриці C; $A_{i(1,m)}$ – *i*-й рядок матриці A; $B_{(1,k)j}$ – *j*-й стовпчик матриці B.

Поширеним апаратним рішенням для даної задачі є multiply-accumulate (MAC) модуль [6]. Як було сказано вище, транспонування відбувається покроковим транспонуванням підматриць розміром $k \times k$. В схемотехніці транспонування матриці відбувається за допомогою менеджменту провідників, а отже дозволяє інтуїтивно реалізувати транспонування.

Висновки

У роботі методом ізоморфних відображень побудована строга модель матричного типу даних з побудованою над ним ППА класу матричних чр-функцій та чр-предикатів та здійснена її апаратно-програмна реалізація у вигляді матричного перетворювача. Здійснено конструктивне уточнення поняття багатовимірної матриці як іменної множини спеціального виду. Уточнено змістовне поняття матричного перетворення у вигляді конструктивної (частково-рекурсивної) багатомісної маніпуляційної функції над матрицями. Проведено семантико-синтаксичне дослідження класу конструктивних маніпуляційних функцій над матрицями в рамках обраного інструментарію – примітивної програмної алгебри (ППА) частково-рекурсивних функцій та предикатів. Знайдено повну сукупність породжуючих класу частково-рекурсивних маніпуляційних функцій та предикатів над матрицями за допомогою методу ізоморфних відображень. Проведено апаратно-програмну реалізацію функцій та предикатів над матрицями з отриманої сукупності породжуючи.

ПЕРЕЛІК ПОСИЛАНЬ

1. A. V. Horyelov, I. V. Red'ko, P. O. Yahanov, "Kompozytsiyni zasady prohramist·s'koyi diyal'nosti" *Visnyk Kyyivs'koho natsional'noho universytetu tekhnolohiy ta dyzaynu. Seriya "tekhnichni nauky"*. no. 3 (86), pp. 11-19, 2015.
2. Redko D. I., Redko I. V., Yahanov P. O., Zakharchenko T. L. "Compositional basis in programmer activity" *Systems research and information technology*, no. 4, pp. 83-96, 2015
3. Buy D. B., Red'ko Í. V. "Primitivnyye programmnyye algebry vychislimykh funktsiy" *Kibernetika*. no. 3. pp. 68-74, 1987
4. Redko I. V., Redko D. I., Zakharchenko T. L. *Kontseptolohichni osnovy proektuvannia [Conceptological foundations of designing]*. Kyiv:Comprint Publ, 2016.
5. Devid M. Kharris, Sara L. Kharris. *Tsifrovaya skhemotekhnika i arkhitektura komp'yutera, vtoroye izdaniye*. 2013. **ISBN** 978-0-12-394424-5
6. Joseph Yiu. The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors, third edition. 2014. 675 p. **ISBN**: 978-0-12-408082-9

Надійшла до редакції 04 квітня 2021 р.

DOI: 10.20535/2617-0965.eae.228485

UDC 321.37

Algebraic Characteristics of the Matrix Conversions Class and Its Hardware Implementation

S. V. Kudlai, ORCID 0000-0003-3972-405X
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", ROR 00syn5v21
Kyiv, Ukraine

***Abstract*—This paper derives the algebraic characteristic of the matrix transformations class by the method of isomorphic mappings on the algebraic characteristic of the class of vector transformations using the primitive program algebras. The paper also describes the hardware implementation of the matrix operations accelerator based on the obtained results. The urgency of the work is caused by the fact that today there is a rapid integration of computer technology in all spheres of society and, as a consequence, the amount of data that needs to be processed per unit time is constantly increasing. Many problems involving large amounts of complex computation are solved by methods based on matrix operations. Therefore, the study of matrix calculations and their acceleration is a very important task. In this paper, as a contribution in this direction, we propose a study of the matrix transformations class using signature operations of primitive program algebra such as multi place superposition, branching, cycling, which are refinements of the most common control structures in most high-level programming languages, and also isomorphic mapping. Signature operations of primitive program algebra in combination with basic partial-recursive matrix functions and predicates allow to realize the set of all partial-recursive matrix functions and predicates. Obtained the result on the basis of matrix primitive program algebra. Isomorphism provides the reproduction of partially recursive functions and predicates for matrix transformations as a map of partially recursive vector functions and predicates. The completeness of the algebraic system of matrix transformations is ensured due to the available results on the derivation of the algebraic system completeness for vector transformations. A name model of matrix data has been created and optimized for the development of hardware implementation. The hardware implementation provides support for signature operations of primitive software algebra and for isomorphic mapping. Hardware support for the functions of sum, multiplication and transposition of matrices, as well as the predicate of equality of two matrices is implemented. Support for signature operations of primitive software algebra is provided by the design of the control part of the matrix computer based on the RISC architecture. The hardware support of isomorphism is based on counters, they allow to intuitively implement cycling in the functions of isomorphic mappings. Fast execution of vector operations is provided by the principle of computer calculations SIMD.**

Keywords — matrix calculations; isomorphism; algebraic characteristic; matrix processing; computer architecture; matrix processor.