Космічна наука і технологія. 1998. Т. 4, № 4. С. 36–43.

УДК 519.681.5 + 681.142.4

Організація обчислень, керованих потоками даних для бортової обробки інформації в реальному часі

Б. Я. Олексів

Державний науково-дослідний інститут інформаційної інфраструктури Національної академії наук України, Львів

Надійшла до редакції 13.04.98

Розглядаються проблеми організації обчислень в багатопроцесорних системах, керованих потоками даних. Описуються підсистема проектування мікропрограмних модулів, а також розроблена система програмування, яка включає мову високого рівня FPL, транслятор, систему налаштування та синтезу багатопроцесорних бортових систем.

ЗАГАЛЬНИЙ ОГЛЯД ПРОБЛЕМИ

Розробка інформаційних технологій багатопроцесорних систем конвеєрного типу функціонування та нейронних мереж, асоціативних і самоорганізовуваних систем є одним з важливих і перспективних напрямків розвитку сучасної інформатики. До числа першочергових задач, які розв'язують з допомогою таких систем, відносяться задачі аналізу і розпізнавання нечислових даних, дослідження і вивчення нервової системи людини, створення моделей функціонування і відтворення живої природи, штучного інтелекту, створення банків знань, експертних систем.

Разом з тим засоби традиційних неймановських і навіть паралельних систем не адекватні переліченим проблемам. Це зумовлено перш за все тим, що швидкодія окремих пристроїв в цих системах наближається до своєї природної межі, обумовленої швидкістю світла.

Проте розвиток технології надвеликих інтегральних схем (НВІС) сприяв значному зменшенню вартості обладнання, і можливо, скоро можна буде включати в конструкцію необхідну кількість елементів для розв'язку достатньо великих задач. Для повного використання переваг НВІС необхідно максимально використовувати апаратуру, що в ній міститься. Це, у свою чергу, стимулює створення нових методів та апаратних і програмних засобів моделювання і обробки інформації.

Отже, актуальність проблеми даного проекту визначається трьома взаємопов'язаними факторами:

- необхідністю розв'язання задач реального часу, які не розв'язуються, або незадовільно розв'язуються іншими засобами;
- актуальними потребами розвитку сучасних засобів обробки інформації, для яких властивий масовий паралелізм, мікромініатюризація та наявність елементів штучного інтелекту;
- потребами вивчення природних механізмів мислення.

Особливістю даного напряму робіт є те, що, з одного боку, для ефективного використання нових обчислювальних структур необхідним є як ґрунтовний теоретичний аналіз їх роботи, так і потужний інструментарій програмних та апаратних засобів. З іншого боку, актуальною є розробка нових засобів навчання самоорганізовуваних систем для побудови нейронних мереж на базі нових технологій.

На даний час для розвитку нових інформаційних технологій, моделювання та синтезу обчислювальних систем проводяться інтенсивні теоретичні і прикладні дослідження. В них значна увага надається теоретичним аспектам функціонування і моделювання обчислювальних структур та розпара-

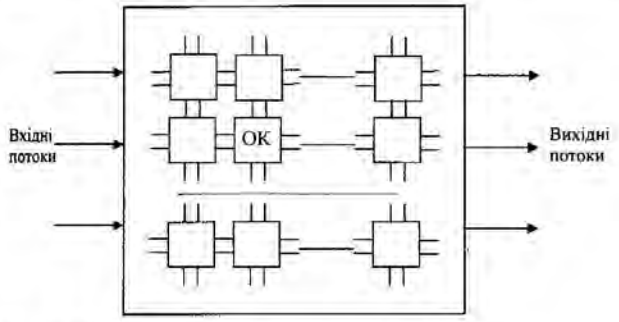

Рис. 1. Матриця однорідних процесорів ОК

лелювання алгоритмів на них. Ці підходи значною мірою лягли в основу концепції обчислювальних машин п'ятого покоління.

Існують також роботи, в яких розглядалися інформаційні технології синтезу засобів керування системними обчислювальними середовищами [2, 3]. Ці роботи завершилися створенням різних версій інструментарію програмних та апаратних засобів розв'язку задач великої розмірності. Проте загального підходу до налаштування та моделювання обчислювальних процесів на базі мультиконвеєрних систем (МКОС) типу однорідних обчислювальних середовищ (ООС) не розроблено. Зауважимо, що МКОС ООС мають багато спорідненого з нейроподібними мережами.

Нами розглядається МКОС, яка складається з матриць однорідних процесорів (ОК) [6] (рис. 1). Підключення джерел і приймачів потоків даних в таких системах реалізується через пристрої узгодження з використанням, при необхідності, буферних запам'ятовуючих пристроїв. В МКОС передача інформації між ОК здійснюється через регістрові комутаційні канали, просуванням за один такт на одну комірку середовища і передачею інформації одній або декільком сусіднім коміркам. Процеси налаштування і роботи матриці ОК розділені в часі. Спочатку формується потік команд для налаштування ООС, далі цей потік команд засилається в регістри команд (РК) кожної ОК. В результаті МКОС буде налаштована на виконання заданого алгоритму і підготовлена до роботи. Процес переналаштування обчислювального середовища на реалізацію іншого алгоритму полягає у формуванні нового потоку команд налаштування, який наново необхідно заслати в РК ОК. Після цього при поступанні потоків даних на бортову обчислювальну систему починається реалізація нового обчислювального алгоритму.

В даній роботі описується розв'язок задач, вирішення яких є необхідною умовою ефективної організації обчислювального процесу на бортових космічних системах потокового типу. Серед них описуються методи проектування модулів для ООС та інструментальні засоби програмування і автоматизованого налаштування бортових обчислювальних систем. В статті розвивається новий підхід до організації обчислень в багатопроцесорних системах, керованих потоками даних.

МЕТОДИ ПРОЕКТУВАННЯ МОДУЛІВ ДЛЯ ООС

Проектування мікропрограмних модулів. Автоматизований синтез мультиконвеєрних обчислювальних структур (МКВС) на однорідних обчислювальних середовищах (ООС) здійснюється при наявності базового набору мікропрограмних модулів (МПМ), налаштованих на виконання необхідних елементарних операцій. Через те, що обчислювальні комірки (ОК) [6] ООС працюють синхронно, то всі дані в МКВС мають однакову періодичність. Обчислювальний процес організований так, що дані поступають у вигляді потоків з фіксованим періодом і відділяються один від одного нульовим (буферним) розрядом.

Для обнулення буферного розряду та виділення необхідних розрядів операндів використовуються різні константи.

При ручному налаштуванні ООС ефективніше генерувати константи поза МПМ і подавати в модулі при необхідності окремим потоком. Тоді один раз згенерована константа використовується всіма МПМ при налаштуванні поля ООС. Але тоді всі МПМ повинні мати додаткові входи для констант, що суттєво ускладнює задачу автоматизованого налаштування ООС, бо із збільшенням числа вхідних потоків МПМ виникає проблема їх синхронізації. Автоматизована синхронізація потоків веде до значного збільшення неефективно використовуваних комірок поля ООС.

В даній статті розглядається генерація МПМ з внутрішньою константою. Задача генерації МПМ з внутрішньою константою повністю не вирішує проблеми синхронізації роботи різних МПМ, тому що необхідно, щоб потік операндів був узгоджений з фазою константи всередині МПМ. Але ця задача суттєво простіша, ніж синхронізація додаткового потоку зовнішніх констант. Таким чином, використання внутрішніх констант спрощує процес синхронізації в різних МПМ, ефективний при автоматизованому налаштуванні ООС.

Нижче описані алгоритми і наведені функціональні схеми базових МПМ з внутрішніми константами для 8-розрядних операндів. Розробка ана-

Рис. 2. Прийняті позначення

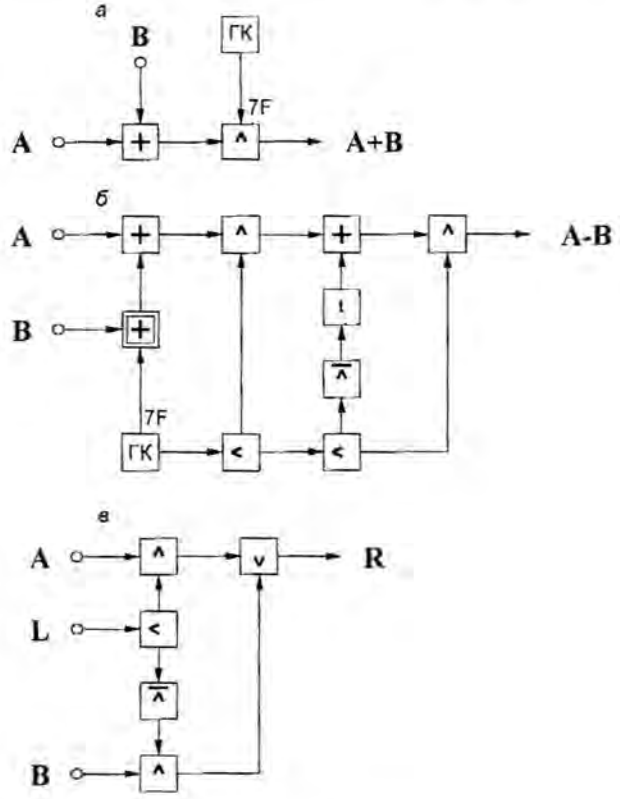

Рис. 3. Функціональні схеми МПМ-додавання (*а*), віднімання (*б*), вибору потоку (*в*)

логічних МПМ для операндів більшої розрядності принципових труднощів не викликає.

Алгоритми функціонування базових МПМ з внутрішніми константами. *Додавання чисел в модифікованому додатковому коді (МДК).* Числа A і B додаються порозрядно з переносом. Потім буферний розряд очищається.

Віднімання чисел в МДК. В алгоритмі віднімання двох чисел A і B використовується переведення числа B в додатковий код і додавання даного числа до числа A. Відомо, що $B_{дод} = B + 1$. В схемі реалізована еквівалентна формула $A - B = (A + B) + 1$. Після кожної операції алгебраїчного додавання буферний розряд очищають, щоб не було спотворень наступного двійкового числа.

Множення чисел в МДК. Реалізовано метод з використанням множення в прямому коді з переводом операндів в прямий код, а результату — в МДК. Через те, що при множенні n–розрядних операндів розрядність результату становить $2n$, то для того, щоб запобігти спотворенню результатів, вхідний потік штучно роздвоюється.

Після цього над кожним потоком проводяться наступні операції:

— перевід чисел із МДК в прямий, визначення знаку добутку, виділення абсолютної величини множників;

— множення в прямому коді. Всі розряди одного множника помножаються на фіксований розряд другого множника, одержані часткові добутки сумуються із зсувом. Сегмент містить стільки вертикальних стовбців, скільки двійкових розрядів має абсолютна величина множника;

— аналіз результатів множення на переповнення, переведення результатів в МДК і формування n-розрядного коду. При переповненні в МДК формуються різні знакові розряди, які ідентифікують некоректність результату.

Вихідний потік формується злиттям одержаних паралельних потоків.

Алгоритм роботи МПМ вибору потоку. МПМ вибору потоку використовується для реалізації обчислень умовних виразів. Модуль має три вхідних потоки: (A, B) — операнди, L — логічний. «1» в логічному потоці означає, що відповідний розряд результуючого потоку R містить розряд операнда A, в протилежному випадку — операнда B. Таким чином, МПМ реалізує логічну функцію $R = (A \cap B) \cup (B, L)$.

Приклади функціональних схем базових МПМ. Умовні позначення, прийняті в наведених нижче функціональних схемах, показані на рис. 2. На рис. 3 і 4 зображені функціональні схеми МПМ, необхідні для синтезу на ООС нейронних елементів.

Імітатор ООС. Імітація роботи ООС дозволяє програмними засобами перевірити правильність функціонування автоматизованої системи налагодження ООС. Імітатор ООС використовується для

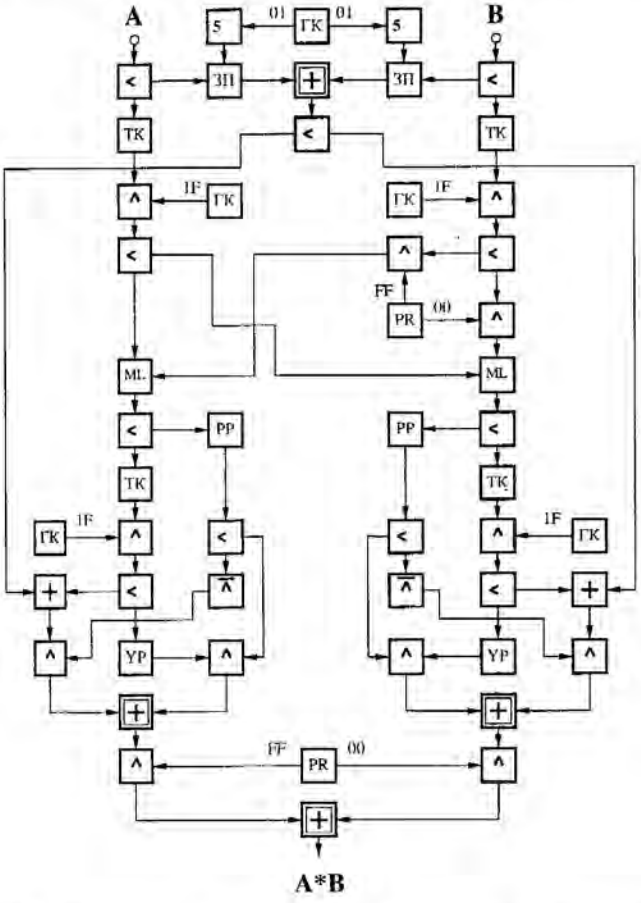

Рис. 4. Функціональна схема МПМ-множення: ТК — перевід мантиси числа з модифікованого доповнюючого коду в прямий і навпаки, PR — перемикач для розпаралелювання, ML — множення 8-розрядних чисел в прямому коді, PP — аналіз результату на наявність переповнення, YP — формування результату множення при виникненні переповнення

відладки МПМ, налаштованих на виконання фундаментальних операцій (арифметичні операції, порівняння, селекція потоків і т. д.).

Імітатор працює з полем ОК заданої розмірності HxL. Кожна ОК середовища попередньо налаштовується на виконання певних операцій, які виконуються при подачі зовнішнього імпульсу, і в процесі роботи не перенаштовуються.

Функціонування ВК визначається регістром команд $R_{i,\ j}$ (i = 1, ..., H; j = 1, ..., L) розміром 2 байти. Внутрішній стан кожної комірки характеризується байтом стану S. Нехай S_k означає k-й біт байта S (0-й біт — молодший). Тоді S_k (k = 0, ..., 3) – біти вихідної інформації, де містяться результати операцій ОК, які поступають на кожну з чотирьох сусідніх комірок; S_4 — біт затримки результату операції; S_5 — біт затримки транзиту; S_6 — використовується для запису «1» при виконанні коміркою операції «запис одиниці», або для запам'ятовування «1» переносу при виконанні операції додавання.

Біт S_7 не використовується.

Стан і регістр команд ОК змінюються дискретно при поступанні тактового імпульсу. З урахуванням описаних вище позначень функціонування ООС можна описати формулою, яка визначає залежність стану комірки в момент T від стану в момент $T-1$.

$$(R[i, j]; S[i, j]) = f(R[i, j]; S_0[i, j + 1], S_1[i + 1, j], S_2[i, j + 1], S_3[i - 1, j], S_4[i, j], S_5[i, j], S_6[i, j]).$$

Стан поля ООС в кожний момент може бути визначений на основі попереднього стану і зовнішньої інформації, яка подається на вхідні комірки ООС. На кожному такті з заданих вхідних комірок можна одержати вихідну інформацію, яка є результатом роботи алгоритму, реалізованого на даній ООС.

Алгоритм роботи імітатора. Алгоритм складається з наступних кроків.

1. Зчитування параметрів ООС з зовнішнього файлу.
2. Формування потоку інформації, яка поступає на входи МПМ.
3. Ввід координат потоку інформації, ознаки напряму виходу потоку з комірки, початкового такту виводу інформаційного потоку.
4. Обнулення початкових станів комірки.
5. Обчислення числа тактів, необхідного для завершення імітації.
6. Визначення нових станів комірок поля ООС в залежності від їх стану на попередньому такті.
7. Зчитування інформації на виходах МПМ і в контрольних точках.
8. Якщо задача імітації не закінчена, то переходимо до кроку 6.
9. Вивід інформації на виходах МПМ і в контрольних точках.

Вихідна інформація для роботи алгоритму імітації, яка вводиться в діалоговому режимі, містить ім'я файлу опису МПМ, розрядність оброблюваної інформації, число і координати контрольних точок.

Дані в файлі опису МПМ мають таку структуру:

— ідентифікатор МПМ;
— розміри матриці МПМ;
— число входів і виходів МПМ;
— регістри команд ОК;
— координати входів і виходів МПМ.

Інформація, яка поступає на входи МПМ і зчитується з їх виходів і в контрольних точках, може

бути представлена послідовністю символів «0» і «1» або в десятковій системі числення, в залежності від призначення імітатора.

ІНСТРУМЕНТАЛЬНІ ЗАСОБИ ПРОГРАМУВАННЯ І НАЛАШТУВАННЯ ООС

Розпаралелювання. Нехай $A = (a_1, a_2, a_3, \ldots, a_n)$ множина операцій алгоритму, який складається з послідовно виконуваних дій, які не містять циклів. Внутрішня структура такого алгоритму повністю описується його інформаційним графом $G(A, \Gamma)$ [5], де A – скінченна множина елементів, що відповідають операціям вихідної програми, які називаються вершинами, а Γ – бінарне відношення, задане на A. При цьому $a_i \in \Gamma a_j$ ($a_1, \ldots, a_n \in A$) в тому випадку, коли результат операції a_j є аргументом операції a_i. Це означає, що існує дуга напрямлена з вершини a_j до вершини a_i.

Розпаралелювання, при такому поданні алгоритму, є задачею розбиття вершин інформаційного графу $G(A, \Gamma)$ на множини Ω, які задовольняють умови

$$\Omega_i \neq \phi,\ \Omega_i \cup \Omega_j = \phi\ (i \neq j),\ \cup_i \Omega_i = A, \qquad (1)$$

$$\Omega_i \subseteq \Gamma\Omega_j\ (i > j). \qquad (2)$$

Умова (2) означає, що для кожної операції $a_i \in \Omega_s$ знайдеться операція $a_j \in \Omega_{s1}$, $s > 1$ і при цьому $a_i \in \Gamma a_j$, тобто вершина інформаційного графу $G(A, \Gamma)$, яка належить ярусу s і залежить хоча б від однієї вершини ярусу $s - 1$.

В умовах (1), (2) Ω_0 – множина операцій, яка залежить тільки від вихідних даних. Результати виконання нульового ярусу є вихідними даними для операцій першого ярусу і т. д. Такий запис алгоритму називається ярусно-паралельною формою Поспєлова [7].

Нами розглядається багатопроцесорна обчислювальна система, що складається з матриць однорідних процесорів [6]. Для програмування таких систем нами була розроблена мова програмування високого рівня FPL [4].

На етапі програмування багатопроцесорної системи засобами мови FPL описуються потоки трьох типів: вхідні потоки даних (EntryFlow), внутрішні потоки (FlowData, Buffer) і вихідні потоки (LogoutFlow). Це основні потоки системи. На етапі трансляції і розпаралелювання FPL-програми основні потоки деталізуються і доповнюються проміжними потоками. До них відносяться зовнішні розщеплені потоки, потоки – результати операцій попереднього рівня ярусно-паралельної форми (ЯПФ), потоки – результати операцій утворення нового потоку в системі («вентиль») і потоки – результати вибору гілки умовного оператора («селектор»). Номери (адреси) основних потоків формуються при трансляції FPL-програми, а проміжних – на етапі побудови ЯПФ. Проміжні потоки даних характеризуються тими самими атрибутами, що і основні потоки і відрізняються від них тільки обмеженим «часом життя» в системі. Побудова ЯПФ здійснюється з дотриманням принципу неперервності потоків: проміжний потік, що виник на даному рівні ЯПФ, може брати участь в операціях, починаючи з її наступного рівня. Виникнувши, проміжний потік існує в системі в одинарному або в розщепленому вигляді до того часу, поки він і всі його гілки, якщо вони є, не візьмуть участі у відповідних операціях на наступних рівнях. Після цього проміжний потік зникає.

Розглянемо детальніше атрибути потоків даних, побудову графу потоків даних та розпаралелювання FPL-програми. Граф потоків даних FPL-програми в ЕОМ представляється масивом записів Attribute. Характеристики рівня ЯПФ описуються списком Base. Кількість елементів у цьому списку дорівнює кількості рівнів ЯПФ. Кожен елемент списку містить номер рівня *nl*, довжину рівня *ll* та кількість розгалужень *br* на ньому. Кожен рівень ЯПФ FPL-програми складається з *ll* записів масиву Attribute. У свою чергу, кожен з цих записів містить всі необхідні для реалізації паралельних обчислень атрибути потоків даних. Серед них такі:

NumF – номер (адреса) потоку;

Opr – операція, в якій бере участь потік з номером *NumF*;

Tf – тип потоку в ярусно-паралельній формі. Для реалізації обчислень методом потоків даних в ЯПФ розрізняються сім типів потоків, які описуються перелічуваним типом *Rang*;

Li – номер гілки потоку на даному рівні. Для нерозщепленого потоку номер гілки дорівнює нулю;

Aj – адреса переходу. Цей атрибут використовується у вузлах операції для реалізації в ЯПФ умовних виразів. Для потоків даних цей параметр дорівнює нулю;

Mir – уявність потоку. В моделі паралельних обчислень, яка описується в даній статті можуть існувати потоки дійсні («true») та уявні («false»). Потоки даних EntryFlow, Buffer і проміжні потоки завжди є дійсними. Потоки даних FlowData і LogoutFlow на початку побудови ЯПФ є уявними і стають дійсними в результаті операції утворення потоку.

Крім потоків даних в ЯПФ містяться також вуз-

ли операцій. Вузли операцій описуються тими самими атрибутами, що і потоки даних. Параметри *NumF*, *Tf* і *Li* для вузлів операцій завжди дорівнюють нулю і *Mir* = *false*. В атрибут *Opr* вузла операцій записується номер операції, а в *Aj* — індекс елемента ЯПФ, де закінчується дія умовного виразу. Для арифметичних виразів атрибут *Aj* вузла операцій дорівнює нулю.

Ознакою кінця поточного рівня ЯПФ є запис

$$<NumF = 0,\ Opr = 0,\ Tr = 0,$$

$$Li = 0,\ Aj = 0,\ Mir = false>. \qquad (3)$$

Необхідною і достатньою умовою спонтанного виконання операцій (арифметичних, логічних, утворення потоку) є існування такої трійки елементів ЯПФ (два потоки даних і один вузол операції), для якої виконуються наступні умови:

$$<NumF_1 \neq 0,\ NumF_2 \neq 0,\ NumF_3 = 0>$$

$$<Opr_1 = Opr_1 = Opr_1 \neq 0> \qquad (4)$$

<Mir1 = true, Mir2 = true, Mir3 = false>.

Для спонтанної реалізації «селектор» необхідним і достатнім є існування на даному рівні подібної п'ятірки елементів. У цьому випадку до умови (4) додаються ще два елементи адресів закінчення на даному рівні ЯПФ гілок умовного виразу.

В запропонованій моделі обчислень в багатопроцесорній системі, керованій потоками даних, паралелізм, який міститься в програмі, реалізується ознаками (3) і (4) природним чином без явного виділення паралельних ділянок. Причому, в даній схемі досягається абсолютно максимальне розпаралелювання в сенсі В.О.Вальковського [1].

Налаштування. *Алгоритм налаштування лінійних ділянок програми.* Лінійна ділянка програми складається з декількох арифметичних виразів. При налаштуванні однорідного обчислювального середовища (ООС) [6] на реалізацію лінійних ділянок програми провадиться розпаралелювання операцій, які входять в різні арифметичні вирази даної ділянки. Перед налаштуванням ООС будується ЯПФ лінійних ділянок програми. На початку налаштування ООС вводяться всі глобальні змінні, які входять в арифметичні вирази.

На першому рівні провадиться налаштування всіх операцій, операндами яких є глобальні змінні, що подаються ззовні. На наступних рівнях виконуються операції, входами яких є глобальні змінні і результати операцій попередніх рівнів.

При налаштуванні ООС на обчислення лінійних виразів використовується така інформація: розрядність використовуваних мікропрограмних модулів, номер рівня налаштування, довжина рівня (кількість потоків на даному рівні), атрибути потоків змінних.

Налаштування ООС здійснюється за рівнями. Після введення інформації про даний рівень проводиться обробка його потоків і налаштування на виконання операцій на даному рівні.

На кожному рівні переглядаються потоки за виразами. Якщо у виразі є операція на першому рівні, то знаходимо кількість глобальних змінних, які беруть участь в цьому виразі і використовуються більше ніж один раз. Ці змінні розгалужуються для їх дальшого використання.

Якщо модуль використовує тільки зовнішні потоки, то він розміщується біля лівого краю плати ООС. Якщо в модулі використовуються потоки, які виходять з попереднього модуля, то модуль розміщується справа від попереднього. Принциповим є те, що вихід потоків з попереднього модуля розміщується зліва, а вхід в даний модуль — справа.

При перегляді номерів вхідних потоків другого і наступних рівнів перевіряємо їх відповідність з номерами вихідних потоків попереднього рівня. При відповідності номерів сусідніх рівнів вхідні потоки трасуються і синхронізуються.

Якщо вхідним потоком є результат виразу попереднього рівня, то цей потік трасується, розгалужується, якщо в цьому є необхідність (потік використовується більше одного разу). Розгалуження здійснюється шляхом використання транзиту з двома виходами. Один вихід використовується в даному МПМ, координати і такт другого виходу запам'ятовуються для подальшого використання на наступних рівнях.

Переглядаючи вирази, в яких є операції на даному рівні, розпізнаємо арифметичні операції. За кодом операції звертаємося до бібліотечних МПМ і налаштовуємо ООС на виконання цих модулів.

При налаштуванні другого і наступних рівнів інформаційні потоки можуть виявитися не в тому порядку, в якому вони необхідні при їх подачі в МПМ. Тому проводиться їх переміщення зверху вниз і знизу доверху. Це робиться за допомогою обчислювальної комірки з операцією «розширений транзит».

Після трасування потоків здійснюємо їх синхронізацію при вході в МПМ. Задача синхронізації полягає в вирівнюванні всіх потоків на потрібну різницю тактів, потрібну для подачі в МПМ.

При налаштуванні рівня знаходимо кількість виразів, налаштування яких закінчене на даному рівні, резервуємо для них траси. В них розгалужуємо

потоки результатів виразів для їхнього дальшого використання.

Потоки, які не використовуються в даному модулі, виводяться на ліву межу даного модуля без синхронізації. Канали зв'язку цих потоків проходять нижче від модуля, який розглядається.

Після закінчення налаштування ООС на обчислення лінійної ділянки програми запам'ятовуються координати і такти вихідних потоків, розміри поля ООС, номери стрічок подачі вхідних потоків, масив інформації налаштування обчислювальних комірок ООС.

Умовні вирази. При розв'язанні більшості задач виникає необхідність організації гілок обчислень для реалізації деяких частин алгоритму. Звертання до тієї чи іншої гілки програми здійснюється з допомогою команд умовного і безумовного переходів. При цьому, якщо виконується відповідна умова, обчислення йде по заданій гілці, а по інших гілках воно не відбувається. При налаштуванні ООС команді умовного переходу не можна поставити у відповідність визначений модуль. В цьому випадку для реалізації команди умовного переходу необхідно провести розкладку модулів для всіх гілок програми. Ця обставина знижує ефективність використання однорідних середовищ при реалізації в ній програми, в якій є умовні переходи.

Приклад. Застосування розробленого інструментарію для моделювання нейронного елементу засобами ООС.

Допускається, що нейронний елемент реалізує деяку функцію, яка «приймає рішення» відносно подальшого проходження сигналу на основі одержаних вхідних сигналів. Спектр можливих функцій досить широкий. Найпростіша порогова функція має вигляд

$$\sum_{i=1}^{m} x_i w_i \geq p \qquad (5)$$

і приймає значення 1 або 0, в залежності від того, перевищує значення зваженої суми величину порогу P, чи ні. В загальному випадку це може бути довільна монотонна дійсна функція, яку називають функцією стиску або функцією активації нейрона.

Традиційні схеми обчислень при навчанні і функціонуванні нейронних мереж та інших близьких до них структур не передбачають будь-якого фіксованого порядку обробки елементів. Тому обчислення задаються в непроцедурній формі. При програмній реалізації обчислення, як правило, проводяться в послідовному режимі. При апаратній реалізації витягується природний паралелізм, який має місце між перцептронами, що знаходяться на одному рівні. В запропонованих підходах та

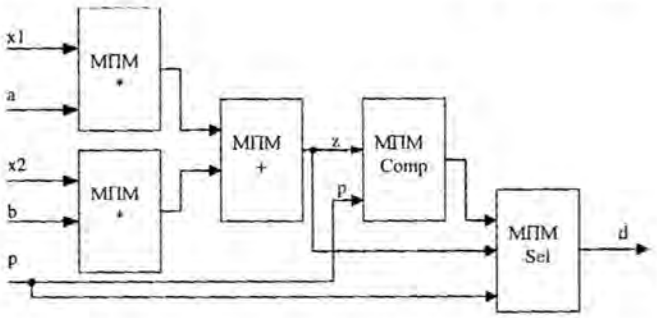

Рис. 5. Структурна схема синтезованого двовходового нейрона

розробленим інструментарієм обчислення в ООС організовуються методом потоків даних, так що паралелізм, як на внутрішньонейронному, так і на міжнейронному рівнях витягується природним чином, без явного вказування паралельних ділянок програми. Для збереження наглядності нижче наведена програма, розроблена автором мовою програмування для моделювання засобами ООС двовходового нейрона з вагами.

```
FPL – програми моделювання
нейронного елемента на ООС.
cap 8;
Entry a, b, x1, x2, p;
Flow z;
Logout d;
begin
    z := a*x1+b*x2;
    d := if z >= p then z else 0;
end.
```

Структурна схема синтезованого нейрона показана на рис. 5.

Параметри синтезованого нейрона: кількість рядків комірок ООС – 45, кількість стовпців – 20, ЯПФ синтезованого нейрона складається з чотирьох рівнів. Кількість активних потоків за рівнями, враховуючи розгалуження: перший рівень – 6 потоків, другий – 4, третій – 4, четвертий – 3 потоки.

Оптимізація при налаштуванні лінійних ділянок програми. При оптимізації процесу налаштування ООС на обчислення арифметичних виразів визначається номер рівня максимальної довжини L. На цьому рівні знаходимо операції, які виконуються не раніше як на L + 2-му рівні. Якщо такої операції на даному рівні немає, то шукаємо наступний номер рівня максимальної довжини. Якщо L є передостанім рівнем, то послідовно переносимо на наступний рівень операції, починаючи з останньої операції L-го рівня.

Після знаходження номера рівня з максималь-

ним числом операцій визначаємо можливість переносу операції з даного рівня на нижчі рівні до рівня, на якому використовується результат даної операції.

Переноси операцій з верхніх рівнів на нижчі починаються з останньої операції. Процес оптимізації побудови ЯПФ є ітераційним і, кількість ітерацій залежить від заданої ширини ЯПФ.

ВИСНОВКИ

В статті розроблено методи проектування модулів та інструментальні засоби програмування і розпаралелювання алгоритмів в багатопроцесорних системах на основі запропонованого опису властивостей потоків даних. Описано імітатор багатопроцесорних систем потокової дії. Встановлено принцип неперервності потоків даних, який застосовується для побудови та аналізу ЯПФ представлення алгоритму. Встановлено умови включення потоку в операцію на заданому рівні ЯПФ. Розпаралелювання алгоритмів, що при цьому досягається відповідає абсолютно-максимальному розпаралелюванню [1]. Описано розроблений алгоритм налаштування багатопроцесорних систем типу однорідних обчислювальних середовищ з елементами оптимізації, яку доцільно використовувати при обмеженій ширині налаштовуваного обчислювача. Розроблені засоби дозволяють автоматично синтезувати бортові спецпроцесори, проблемно-орієнтовані обчислювальні середовища в реальному часі та нейроподібні середовища. Результати роботи можуть стати основою також для створення спеціалізованих стендів програмування, налаштування і відлагодження бортових космічних систем.

1. Вальковский В. А. Распараллеливание алгоритмов и программ. Структурный подход. — М.: Радио и связь, 1989. — 176 с.
2. Голиченко Е. П., Костив Ю. В., Фурдило А. И., Шмойлов В. И. Программирование однородных вычислительных сред. Методические рекомендации для программистов. — Львов, 1989. — 51 с.
3. Грицык В. В., Слуцкий Ю. Н. Распараллеливание и настройка в сложных системах параллельной обработки информации. — Львов, 1981. — 78 с.
4. Олексів Б. Я. Розробка інформаційних технологій налаштування високопродуктивних обчислювальних систем на базі однорідних обчислювальних середовищ: Автореф. дис. ... канд. техн. наук. — Львів, 1994. — 23 с.
5. Параллельная обработка информации: В 3 т. — Киев: Наук. думка, 1985. — 280 с. — Т. 1: Распараллеливание алгоритмов обработки информации / Под ред. А. Н. Свенсона.
6. Параллельная обработка информации: В 3 т. — Киев: Наук. думка, 1986. — 288 с. — Т. 3: Вычислительные системы структуры и среды для решения задач большой размерности / Под ред. В. В. Грицыка.
7. Поспелов Д. А. Введение в теорию вычислительных систем. — М.: Сов. радио, 1972. — 280 с.

ORGANIZATION OF DATAFLOW-CONTROLLED CALCULATIONS FOR ONBOARD REAL-TIME INFORMATION PROCESSING

B. Ya. Oleksiv

Organization of dataflow-controlled calculations for onboard real-time information processing is discussed. A subsystem was developed for designing microprogram modules (MPM) for the homogeneous computing medium (GCM). This subsystem includes also an imitator of the GCM. A few samples of the MPM functional schemes are given. The method of paralleling algorithm for multiprocessor computing system was worked out on the basis of the dataflow properties. The conditions for the onset of dataflow at a definite level of the level-parallel form are determined. The adjusting algorithm of flow activity multiprocessor system based on the homogeneous computing medium with the elements of optimization was developed and realized in a program form.