ВИСША АТЕСТАЦИОННА КОМИСИЯ

СПЕЦИАЛИЗИРАН НАУЧЕН СЪВЕТ ПО ИНФОРМАТИКА И ПРИЛОЖНА МАТЕМАТИКА

Стефка Стоянова Фиданова

СИНТЕЗ НА СИСТОЛИЧЕСКИ МАСИВИ

Научна специалност 01.01.12

АВТОРЕФЕРАТ

на дисертация за получаване на

"образователната и научна степен Доктор"

Научен ръководител:
проф. д.м.н. Райчо Лазаров
Научен консултант:
ст.н.с д-р. Христо Джиджев

Рецензенти:
1.проф. д.м.н. Тодор Гичев
2.доц. д-р Димитър Иванчев

София,1998

Дисертационният труд е обсъден и допуснат до защита на разширено заседание на секция "Паралелни алгоритми" при ЦЛПОИ - БАН.

Дисертационният труд съдържа 106 страници, 17 фигури, 58 литературни източници.

Защитата на дисертационния труд ще се състои на заседание на СНС по информатика и приложна математика при ВАК на 13.09.1999 от 13:30 часа в заседателната зала на Института по математика и информатика при БАН ул. "Акад. Георги Бончев" Бл.8. Материалите по защитата са на разположение на интересуващите се в библиотеката на Института по математика и информатика.

Автор: Стефка Стоянова Фиданова

Заглавие: Синтез на систолически масиви

Обща характеристика на дисертационния труд

Актуалност на проблема

Развитието на силициевата технология през последните няколко десетилетия доведе до значително увеличаване на плътността и намаляване на цената на електронните устройства. Възникването на високо интегрираните системи (VLSI), съдържащи стотици хиляди транзистори, оказва голямо въздействие върху архитектурата на съвременните изчислителни машини. така става възможно да се внедрят мощни компютърни структури върху един чип. Систолоческите масиви са интересна паралелна компютърна архитектура, много подходяща за директна реализация върху VLSI чип. Той се отличава с висока скорост на предаване на данни. Когато единица информация се вземе от паметта, тя се използва ефективно от всички клетки, преминавайки от клетка в клетка. Така се избягва класическият проблем с достъпа до паметта. Едно от направленията за изследване на систолическите масиви е свързано със задачата за създаване на методология за трансформация на алгоритмите върху систолически структури. Преминаването на данните през клетките води до редица проблеми като: активиране на клетките, подреждане на входните данни, синхронизация.

Идеята на Kung и Leiserson е систолическият масив да бъде периферно устройство на компютър с общо предназначение наречен Host. Ролята на този компютър (Host) е да подава входните данни за решаваната задача. Систолическият масив може да се използва за решаване както на една цяла задача, така и на разпаралелената част на една по-обща задача. Резултатът от паралелните изчисления се връща обратно на Host компютъра за получаване на крайното решение на задачата. Тази идея вече е реализирана дори в архитектурата на персонални компютри от последните поколения. Един от най-успешните продукти на германската фирма ISATEC, основана през 1989г. с цел разработване на паралелни компютри с ниска цена, е Systola 1024. Този продукт използва

стандартен персонален компютър като Host. Systola 1024 се състои от 1024 процесора разположени като правоъгълна мрежа. Предназначен е за матрични изчисления и при направените изпитания е установено, че ускорява работата на персоналния компютър от 10 до 60 пъти.

Систолическите масиви са от категорията SIMD (Single Instruction Multiple Data), от класификацията на Flynn [11] за паралелни архитектури. Клетките на систолическия масив работят синхронизирано. За един такт всяка клетка получава данни от съседните клетки, извършва някакви операции и изпраща данни на своите съседи. Така се получава поток от данни преминаващ на тласъци (систоли) през систолическия масив, подобно на сърцето изтласкващо кръвта на равни тласъци. Оттам идва и името систолически.

Систолическият масив се състои от изчислителни елементи с фиксиран размер на локалната памет ако има такава.

Чрез прилагане на нови технологии, цената на простите изчислителни елементи спада. Така цената на изчислителната система силно зависи от нейния дизайн. Чрез използването на регулярен и прост дизайн и използвайки VLSI технология може да се постигне намаляване на цената на архитектурата.

Важен фактор за постигане на висока скорост на изчислителната система е паралелизмът. При системите със специално предназначение, архитектурата зависи от алгоритъма, който се изпълнява.

Систолическият масив приема, данните от Host компютър, а след това му връща резултата. Намалява се броят на входно/изходните операции и обръщенията към външна памет.

Посочените свойства са основа за използване на систолическите масиви. Върху голям масив от данни.

Гореизложеното обосновава актуалността на темата на дисертацията.

Цел на дисертацията

Предмет на дисертационния труд е изследването на систолически

масиви, създаване на алгоритми за решаване на широко използвани в практиката задачи и разполагането им върху систолически масиви.

В съответствие с тази цел основните задачи в дисертацията са:

1. Изследване на задачи от типа на задачата за раницата:

- Реализация на задачата върху линеен систолически масив;
- Реализация на задачата върху двумерен систолически масив.

2. Разширяване на систолически масиви:

- Разширяване на систолически масив, съдържащ само един тип изчислителни елементи;
- Разширяване на систолически масив, съдържащ няколко типа изчислителни елементи.

Метод на изследването

При изследването на алгоритмите и систолическите масиви използваме означения и понятия от теорията на графите. За описание на изображението на алгоритъма върху систолическия масив и получаване на алокираща функция прилагаме метода на покритията. При разширяването на хомогенни систолически масиви (състоят се от един тип изчислителни елементи), използваме замяната на един връх от графа описващ масива с граф, докато при хетерогенните систолически масиви (състоят се от повече от един тип изчислителни елементи) използваме разделяне на графа на хомогенни части.

Апробация

Части от дисертационната работа са докладвани на на семинара по Паралелни алгоритми в ЦЛПОИ - БАН. Някои от работите са използвани в договорите МУ02/92, МУ20/94, МУ02/96 с Националния фонд

за **Научни изследвания**. Част от публикуваните резултати са представени на специализирани международни конференции, както следва:

- Workshop of Parallel & Distributed Processing, Sofia, 1991;
- International Conference on Numerical Methods and applications, Sofia, 1992;
- Workshop of Parallel & Distributed Processing, Sofia, 1993;
- International Conference on Mathematical Modeling and Scientific Computation, Sozopol, 1993;
- Third International Conference on Numrerical Methods & Applications, Bankia, 1994;
- International Conference of Parallel Processing, Potsdam, Germany, 1994;
- International Conference on Parallel Processing and Applied Mathematics,Zakopane, Poland,1997.

Публикации

Основните резултати по дисертацията са публикувани в:

1. S. Fidanova, *The mapping of algorithms on linear systolic array*, Int. Conf. of Parallel and distributed processing, Sofia, 1991, pp.285-292.
2. V. Aleksandrov, S. Fidanova, *On the average exeqution time for a special class of non uniform recurrence equation on a linear systolic array*, Int. Conf. of Numerical Methods and applications, Sofia, 1992, pp.144-192.
3. S. Fidanova, *Recursive algorithms on fixed size systolic array*, Int. conf. of Parallel and distributed processing, Sofia, 1993, pp.360-368.
4. S. Fidanova,*Automatic derivation of expandable systolic algorithms for LU decompozition*, Int. conf. of Mathematical Modeling and Scientific Computation, Sozopol, 1993, pp.63-66.

5. V. Aleksandrov, S. Fidanova, *On the expected execution time for a class of non uniform recurrence equation mapped onto 1D regular array*, J. Parallel Algorithms and Applications, vol.1, 1994, pp. 303-314.

6. V. Aleksandrov, S. Fidanova, *Optimal 2D regular array for a special class of non uniform recurrence equations*, Tech. report, N 634, University of Newcastle, UK,1993, 17pp.

7. V. Aleksandrov, S. Fidanova,*Non uniform Recurrence equations on 2D regular arrays*, Advances in numerical methods and applications, Eds Iv. Dimov, Bl.Sendov, P. Vasilevski, World Scientific, 1994, pp.217-226.

8. S. Fidanova, V. Aleksandrov, G. Megson,*Mapping Knapsack Type Problems on 2D Regular Arrays: Two Case Studies*, Int. Conf. of Parallel Processing, Potsdam, Germany, 1994.

9. S. Fidanova, A. Goldman, *Parallel execution of irregular meshes into a systolic linear array*, Int.conf. on Parallel Processing and Applied Mathematics, Poland, 1997, pp.267-274

Резултатите от [2], [5], [6] и [7] са цитирани и използвани в книгата на G.M. Megson "Parallel Algorithms for Knapsack Type Problem", World Scientific Publ,1998. Резултата от [5] е цитиран в работата на A. Goldman и D. Trystram "An efficient parallel algorithm for solving the knapsack problem on the hypercube", In proceedings of 11th International Parallel Processing Symposium, Geneve, Suisse, 1997, organization (IEEE).

Структура и обем на дисертацията

Дисертационният труд съдържа увод оформен като първа глава и още три глави. Включва 106 страници текст, 17 фигури. Библиографията включва 58 литературни източника.

Кратко съдържание на дисертацията

Обикновено систолическите масиви са специализирани за изпълнението на конкретен алгоритъм. Цели се постигане на максимален па-

ралелизъм с минимум комуникации. Вземайки предвид модулярността, регулярността и локалните връзки, систолическите масиви са удобни за VLSI реализация.

Систолическите масиви са много удобни за решаването на някои задачи от линейната алгебра (конволюция, трансформация на Фурие, матрични изчисления приложими за решаване на линейни системи) [14, 16, 18, 21, 22]. Със систолически масиви могат да се решават не само числени, но и други сложни задачи като разшифроване на кодове, обработка на биологична информация и др. [1, 10, 25].

Едни от методите за синтез на систолически масиви са за решаване на хомогенни рекурентни уравнения [23, 24]:

$$u(i,j) = F(u(i-p,j), v(i,j-q)) \quad (i,j) \in D \; p,q = const.,$$

където $D-$ областта в която се изменят променливите на уравнението $u, v-$ функции, F- функция.

Едно от направленията за изследване на систолическите масиви е свързано със задачата за създаване на методология за автоматична трансформация на алгоритмите върху систолически масиви, които да реализират тези алгоритми. Методите за проектиране на паралелни архитектури, в частност методите за проектиране на систолически масиви съдържат изобразяване на множеството от операции извършвани от алгоритъма върху пространство/времева графика, описваща извършването на изчисленията във времето. Това изобразяване показва всеки елемент от множеството на изчисленията в кой изчислителен елемент ще бъде реализиран и в кой момент време.

Предмет на дисертационният труд е изследването на систолическите масиви. Разработени са алгоритми за решаване на широко използвани в практиката задачи и разполагането им върху систолически масиви. За описание на алгоритмите и на хардуера се използват графи [6]. Ще използваме означенията и понятията въведени от Хр. Джиджев [9]

Ориентиран граф G се нарича наредена двойка (V, E) от множества, където елементите на V се наричат *върхове* на G, а елементите на E са

наредени двойки от върхове и се наричат *ребра* на G, т.е. E се съдържа в $V \times V$.

В дисертацията ще се използват само ориентирани графи и за краткост ще бъдат наричани само графи.

Ако $e = (v, w) \in E$, то v се нарича *начало*, а w - *край*. За върховете v и w ще се казва, че са *съседни*, а реброто e е *инцидентно* с тях. Броят на инцидентните с даден връх ребра се нарича *степен* на върха.

Път се нарича такава последователност от върхове $p = (v_0, v_1, \ldots, v_k)$, за която $(v_{i-1}, v_i) \in E$, $i = 1, ..., k$, числото k се нарича *дължина* на пътя. Ако $v_0 = v_k$, то p е *цикъл*.

За представяне на алгоритмите и систолическите масиви ще използваме графи с етикети и цветове на ребрата и цветове на върховете.

Поточен граф се нарича ацикличен граф с цветове на върховете и цветове и етикети на ребрата, удовлетворяващ условията:

1. Ребра с еднакъв етикет имат еднакъв цвят.

2. За всеки етикет s съществува път започващ и завършващ във върхове от степен 1 и съдържащ всички ребра с етикет s и само тях.

3. За всеки цвят c и за всеки връх v съществува най-много едно ребро с цвят c влизащо/излизащо от v.

4. За всяка двойка от върхове (v, w), всички пътища между v и w имат еднаква дължина.

Нека G е поточен граф. Ще въведем релация на частична наредба " $\leq$ ". За върховете $v, w \in G$ казваме, че $v \leq w$, ако съществува път от v до w. Минималните върхове по отношение на тази релация се наричат *входни*, а максималните - *изходни*. Всички входни и изходни върхове образуват множеството на *терминалните* върхове. Ребрата инцидентни с тях се наричат *терминални ребра* (входни и изходни).

Ще използваме следните означения, ако $v \in V(G)$ и $e = (v, w) \in E(G)$, то с $col(v)$ ще означаваме цвят на v, с $col(e)$ - цвят на e, а с $lab(e) = lab((v, w))$ - етикет на e.

Нека имаме алгоритъм за решаване на дадена задача. Ще построим граф $G = (V, E)$, по следния начин. На всяка операция от алгоритъ-

ма ще съпоставим връх от графа. Ако аргументът на една операция е получен, като резултат от изпълнението на друга операция, то съответните върхове ще свържем с ребро, започващо от върха откъдето се взема резултата. На входните и изходните данни съответстват терминални върхове. Ако аргументът на резултата е начална стойност или резултатът от операцията не се използва (т. е. изходни данни), то съответното ребро е терминално. Граф $G = (V, E)$ ще наричаме *граф на зависимостите.*

За изобразяване на систолическите масиви също ще използваме графи. Върховете ще съответстват на изчислителните елементи, а ребрата - на физическите връзки между тях. Този граф ще наричаме хардуерен граф.

Алокираща функция $alloc : G \to H$ ще наричаме изображение на графа на зависимостите G, за даден алгоритъм, върху хардуерния граф H изобразяващ даден систолически масив. При това изображение на всеки връх от граф G съответства точно един връх от H и в един връх от H могат да бъдат изобразени повече от един връх на G.

Реализацията на алгоритъма, изобразен чрез граф на зависимостите G, върху систолически масив, изобразен чрез граф H, ще разделим на времеви стъпки (моменти). В един момент в клетките на систолическия масив, изобразени чрез върховете на граф H, се извършват изчисления и предаване на данни към съседните клетки. Броят на моментите (стъпките) за получаване на резултата ще наричаме *време за реализация* на алгоритъма, върху систолическия масив (*времева сложност*).

Времева функция ще наричаме функция $t : V(G) \to N$, където $V(G)$ е множеството от върховете на G, а N е множеството на целите положителни числа.

Времева функция отговаряща на условията:

1.$t(v) \neq t(w)$, ако $alloc(v) = alloc(w)$, $v, w \in V(G)$;

2.$t(v) < t(w)$, ако от връх v излиза ребро, влизащо във връх w, $v, w \in V(G)$;

ще наричаме *валидна времева функция.*

Когато поточен граф (граф с цветове на върховете и цветове и етикети на ребрата), се използва за описание на даден алгоритъм (граф на зависимостите), върховете му съответстват на изчисленията, а ребрата на зависимостта между тях. Цветовете на ребрата съответстват на различните потоци данни, а етикетите на придвижването на една информационна единица. Цветовете на върховете съответстват на различните видове изчисления.

Когато поточният граф се използва за изобразяване на систолически масив (хардуерен граф), върховете съответстват на изчислителните елементи, а ребрата на физическата връзка между тях. Цветовете и етикетите на ребрата показват придвижването на информацията, а цветовете на върховете съответстват на различните видове изчислителни елементи.

Всеки поточен граф може да се разглежда като хардуерен граф или граф на зависимостите в зависимост от интерпретацията.

Преминаването на данните през клетките на систолическия масив води до проблеми като активиране на клетките и синхронизация на въвеждането на данните през крайните клетки на систолическия масив. Решаването на тези проблеми е важно за реализацията на дадения алгоритъм върху систолическия масив.

Проблемът за активиране на клетките идва от преминаването на данните през всички клетки на масива. Въпросът е кои данни да се считат за "анонимни"; отделната клетка не знае мястото си в масива, нито кои данни се намират в паметта и. За това успоредно с данните се придвижва информация за контрол на масива, която показва дали клетката трябва да извършва изчисления или да е неактивна.

Проблемът за подреждане на данните е проблем за вход на данните, които не са записани предварително в локалната памет и е нужно да бъдат подредени по определен начин и подавани на систолическия масив (от Host компютъра) в определени моменти.

Дисертацията се състои от увод, оформен като първа глава и още

три глави.

Глава 2

Линейна реализация на задачи от типа на задачата за раницата

В глава втора е разгледан алгоритъм за решаване на нехомогенни рекурентни уравнения за задачи от типа на задачата за раницата и разполагането му върху линеен систолически масив.

Можем да формулираме задачата за раницата по следния начин: имаме раница с капацитет c, в която можем да сложим m типа предмети. Всеки предмет от тип i има тегло $f(i)$ и цена $g(i)$, където $f(i)$, $g(i)$, m и c са цели положителни числа. Трябва от предмет от тип i да поставим в раницата z_i броя така, че цената на предметите в нея да е максимална, без да надхвърляме капацитета и, т.е.:

$$\max\left\{\sum_{i=1}^{m} g(i)z_i : \sum_{i=1}^{m} f(i)z_i \leq c,\ z_i \geq 0 \text{ цяло число},\ i = 1,\ldots,m\right\}.$$

От този тип са задачата за раницата без ограничения (UKP), задачата за сумиране на подмножества (SSP), задача за рестото (CMP) и др. Задачите от типа на задачата за раницата имат голямо приложение в икономиката и производството, например при подреждане на товари в кораби, разкрояване, контролиране на бюджет. Тези задачи водят до следния ограничен клас от нехомогенни рекурентни уравнения:

$$u[i,j] = F(u[i-1,j], u[i-l, j-f(i)] + g(i)).$$

Уравнението е дефинирано за $0 < i \leq m$, $0 < j \leq c$ и следните гранични условия: $u[i,0] = 0$, $u[i,j] = 0$ при $j < 0$. При $l \in \{0,1\}$ и $u[0,j]$ е дадено. Целочислените функции $f(i)$ и $g(i)$ са известни предварително за $i = 1,2,\ldots,m$ и стойностите $u[m,j]$ трябва да бъдат изчислени. До горното уравнение водят следните задачи:

- UKP Ако F е max, $l = 0$, $u[0,j] = 0$ за $0 \leq j \leq c$ $1 \leq i \leq m$;
- CMP Ако F е min, $l = 0$, $g(i) = 1$, $u[0,0] = 0$, $u[0,j] = \infty$ за $1 \leq j \leq c$ $1 \leq i \leq m$;
- SSP Ако F е max, $l = 1$, $f(i) = g(i)$, $u[0,j] = 0$ за $0 \leq j \leq c$ $1 \leq i \leq m$.

Стремежът в тази глава е да се създаде линеен систолически масиви за горепосочената задача. Всяка клетка на систолическия масив има локална памет с фиксиран размер α регистъра, използвани за запазване на междинните изчисления. Числото α не зависи от параметрите на задачата.

Съществуват различни паралелни реализации на задачата за раницата. Някои основни резултати са дадени в следната таблица.

	клетки	**време**
Lin, Storer	p	$O\left(\left(\frac{mc}{p} + c^2\right) log(p)\right)$
Gruau, Andonov	p	$\frac{mc}{p}$
Teng	c^3	$O(log^2 mc)$
Benaini, Andonov	mf_{max}	$mf_{max} + c$
Quinton, Andonov	p	$\frac{mcf_{max}}{p} + p$
Fidanova	p	$\leq c\frac{\sum_{i=1}^{m}\left\lceil\frac{f(i)}{\alpha}\right\rceil}{p} + p$
Fidanova	$p = nq$	$\frac{mc\frac{f_{max}}{\alpha}}{p \min\left\{q, \frac{f_{min}}{\alpha}\right\}} + p$

Andonov и Benaini [2] са едни от първите, които разглеждат реализация на задачата за раницата върху систолически масив. В случая, когато няма ограничение за броя на използваните клетки, тяхната времева сложност е $mf_{max} + c$ върху mf_{max} изчислителни елемента с фиксирана памет 1 регистър, $f_{max} = \max_i f(i)$. Недостатък на тяхната реализация

е, че имат изчислителни елементи, които не извършват изчисления, а само предават данни. Lin и Storer [15] предлагат реализация на тази задача върху хиперкуб с p процесора с времева сложност $O((\frac{mc}{p}+c^2)log(p))$, но локалната памет на изчислителните елементи зависи от параметрите на задачата. Chen, Chern и Jang [7] описват линейна архитектура с p процесора с времева сложност $O\left(\frac{mc}{p}+m\right)$, но локалната памет на изчислителните елементи зависи от параметрите на задачата. Броят на клетките при някои реализации е твърде голям. Например алгоритъмът на Teng [26] изисква $M(c)$ изчислителни елемента и има времева сложност $O(log^2(mc))$. Където $M(n)$ е броят на клетките, необходими за умножение на две $n \times n$ матрици за време $O(log(n))$ (това число е между n^2 и n^3). В [4] Andonov и Quinton постигат времева сложност $O\left(\frac{mc/_{max}}{p}+p\right)$ върху p процесора. Това е обобщение на реализацията в [2], но когато броят на клетките е фиксиран. В [3] Andonov и Gruau реализират задачата върху двумерен правоъгълен масив, състоящ се от p изчислителни елемента, с времева сложност $O\left(\frac{mc}{p}\right)$, но при тях локалната памет на отделната клетка е достатъчно голяма, т.е. зависи от параметрите на задачата.

Целта е да се намерят съответно времева и алокираща функции. Алокиращата функция изобразява няколко върха от графа на зависимостите в един и същи връх на хардуерния граф, чрез който изобразяваме линеен систолически масив. Неотрицателната времева функция показва в кой момент време се извършва дадено изчисление. Тя трябва да бъде такава, че данните необходими за дадено изчисление да са получени преди извършване на изчислението.

Ще предполагаме, че разполагаме с толкова изчислителни елемента, колкото са ни необходими. Конструирането на линеен систолически масив може да се раздели на два етапа:

1. Директно изобразяване на графа на зависимостите върху линеен хардуерен граф състоящ се от m нетерминални върха. Така локалната памет на клетка с номер i от линейния систолически масив трябва да има $f(i)$ регистъра, т.е. зависи от параметрите на задачата. Такъв

систолически масив се нарича *немодулярен*.

2. За да бъде получен *модулярен* систолически масив (т.е. локалната памет да не зависи от параметрите на задачата) с фиксирана памет α регистъра, клетка с номер i се заменя с $\left\lceil \frac{f(i)}{\alpha} \right\rceil$ клетки ако $f(i) \leq \frac{c}{2}$ и с $\left\lceil \frac{c-f(i)+1}{\alpha} \right\rceil$, ако $f(i) > \frac{c}{2}$, по дължина на масива. С w_i ще бележим броя на клетките заменили клетка i.

Така алокиращата функция ще бъде:

$$alloc(i,j) = \sum_{s=1}^{i-1} w_s + k,$$

където $k = \begin{cases} \left\lceil \frac{c-f(i)+1}{\alpha} \right\rceil & \text{за } f(i) > \frac{c}{2}, \quad f(i) \geq j > c - f(i) \\ \left\lceil \frac{j \bmod f(i)+1}{\alpha} \right\rceil & \text{в останалите случаи} \end{cases}$,

а (i,j) е връх от графа на зависимостите, а дясната страна на равенството е номер на връх от хардуерния граф.

Разглеждаме следната времева функция:

$$t(i,j) = \begin{cases} j+k & i = 1 \\ \sum_{s=1}^{i-1} w_s + j + k & 2 \leq i \leq m \\ \sum_{s=1}^{i-1} w_s + j & i = m+1 \end{cases},$$

където $k = \begin{cases} \left\lceil \frac{c-f(i)+1}{\alpha} \right\rceil & \text{за } f(i) > \frac{c}{2}, \quad f(i) \geq j > c - f(i) \\ \left\lceil \frac{j \bmod f(i)+1}{\alpha} \right\rceil & \text{в останалите случаи} \end{cases}$.

Тогава времевата сложност на алгоритъма ще бъде:
$T = \sum_{i=1}^{m} w_i + c \leq \sum_{i=1}^{m} \left\lceil \frac{f(i)}{\alpha} \right\rceil + c$ върху $\sum_{i=1}^{m} w_i$ процесора.

Описали сме функционирането на отделната клетка на систолическия масив и контролните символи, които са необходими за правилното и функциониране. Използваме един управляващ символ, който показва съответно след колко клетки се намира клетката, със съответните

данни, в която ще се извършват изчисления. В случая, когато $\alpha = 1$ и всички стойности на $f(i)$ са равни, за $i = 1, \ldots, m$, нашият резултат се изравнява с резултата на Andonov и Benaini [2]. В останалите случаи получаваме по-добър резултат както за броя на изчислителните елементи, така и за времевата сложност на реализацията.

Разгледан е и случаят когато линейният масив е с фиксиран брой изчислителни елементи, p. Тогава времевата сложност на алгоритъма е:

$$t = c\left\lceil \frac{\sum_{i=1}^{m} w_i}{p} \right\rceil + p \leq c\frac{\sum_{i=1}^{m} \left\lceil \frac{f(i)}{\alpha} \right\rceil}{p} + p.$$

Когато всички стойности на $f(i)$ са равни, $i = 1, \ldots, m$, и $\alpha = 1$, нашият резултат се изравнява с резултата получен от Andonov и Quinton [4]. В останалите случаи получаваме по-добър резултат.

Глава 3

Двумерна реализация на задачи от типа на задачата за раницата

В трета глава предлагаме реализация на горе описаното нехомогенно рекурентно уравнение върху двумерен правоъгълен систолически масив. Двумерният правоъгълен систолически масив, състоящ се от $n \times q$ изчислителни елемента, може да се представи, чрез граф $G(V, E)$. На всеки връх от графа се съпоставя наредена двойка числа (i, j), $i = 1, \ldots, n$, $j = 1, \ldots, q$. Между два върха $v = (i, j)$ и $w = (i', j')$ има ребро само, ако $|i - i'| + |j - j'| = 1$. Подобно на предходния случай, локалната памет на отделната клетка на двумерния правоъгълен систолически масив е с фиксиран размер α регистъра, независещ от параметрите на задачата, предназначени за запазване на междинните изчисления. Предполагаме, че разполагаме с толкова изчислителни елемента, колкото са ни необходими. Конструирането на двумерен правоъгълен систолически масив може да се раздели на два етапа:

1. Директно изобразяване на графа на зависимостите върху линеен хардуерен граф, състоящ се от m нетерминални върха. Така ни е не-

обходима локална памет $f(i)$ регистъра в клетка i. Това означава, че паметта зависи от параметрите на задачата и получаваме немодулярен масив.

2. За да получим модулярен систолически масив с фиксиран размер на локалната памет α регистъра, заменяме клетка i с $\left\lceil \frac{f_{max}}{\alpha} \right\rceil$ клетки, $f_{max} = \max_i f(i)$, така че първата клетка от клетките заменящи клетка i се свързва с първата клетка от клетките заменящи клетка $i+1$, съответно втората клетка от клетките заменящи клетка i се свързва с втората клетка от клетките заменящи клетка $i+1$, $i = 1, \ldots, m-1$. Така получаваме двумерен правоъгълен систолически масив ,на който всяка клетка има фиксирана памет α регистъра.

В този случай алокиращата функция ще бъде:

$$alloc(i,j) = (i,k),$$

където $k = \begin{cases} j \bmod \left\lceil \frac{f(i)}{\alpha} \right\rceil & j \bmod \left\lceil \frac{f(i)}{\alpha} \right\rceil \neq 0 \\ \left\lceil \frac{f(i)}{\alpha} \right\rceil & j \bmod \left\lceil \frac{f(i)}{\alpha} \right\rceil = 0 \end{cases}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, c.$

Разглеждаме следната времева функция:

$$t(i,j) = (i-1)\left\lceil \frac{f_{max}}{\alpha} \right\rceil + \left\lceil \frac{j}{\left\lceil \frac{f_{min}}{\alpha} \right\rceil} \right\rceil.$$

Получаваме, че времевата сложност на алгоритъма е:

$$T = m\left\lceil \frac{f_{max}}{\alpha} \right\rceil + \left\lceil \frac{c}{\left\lceil \frac{f_{min}}{\alpha} \right\rceil} \right\rceil$$

върху $p = m \times \left\lceil \frac{f_{max}}{\alpha} \right\rceil$ изчислителни елемента.

Описали сме функционирането на отделната клетка на систолическия масив и контролните символи, които са необходими за правилното и функциониране. Използваните управляващи символи са като в случая на линеен систолически масив.

Разглеждаме и случая когато двумерният правоъгълен систолически масив е с фиксиран брой, $n \times q$, изчислителни елемента. Получихме, че времевата сложност на цялата задача е:

$$T = \left\lceil \frac{m}{n} \right\rceil \left\lceil \frac{\left\lceil \frac{f_{max}}{\alpha} \right\rceil}{q} \right\rceil \left\lceil \frac{c}{\min\{q, \left\lceil \frac{f_{min}}{\alpha} \right\rceil\}} \right\rceil + nq$$

Нека разгледаме случая, когато $\alpha = 1$, n дели m, q дели f_{max}. Тогава получаваме, че времевата сложност на алгоритъма реализиран върху двумерен правоъгълен систолически масив е:

$$T = \frac{mcf_{max}}{nq \min\{q, f_{min}\}} + nq$$

и е по-добра от резултата на Andonov и Quinton. Когато $q = 1$ или $f_{min} = 1$ нашият резултат се изравнява с резултата в [4].

Нека с T_1 бележим времевата сложност на алгоритъма за решаване на задачи от типа на задачата за раницата върху линеен систолически масив, а с T_2 отбележим времевата сложност на задачата, но върху двумерен правоъгълен систолически масив. Нека и двата систолически масива бъдат с еднакъв брой изчислителни елементи. Коя от двете реализации е по-добра зависи от това, коя от стойностите $\sum_{i=1}^{m} f(i)$ и $\frac{mf_{max}}{\min\{q, \frac{f_{min}}{\alpha}\}}$ е по-малка. Тогава при стойности на $f(i)$ близки до f_{min}, $i = 1, \ldots, m$ имаме $T_1 < T_2$. Обратно, ако стойностите на $f(i)$ са близки до стойностите на f_{max} то $T_1 > T_2$. Когато $q = 1$, т.е. двумерният масив стане линеен $T_1 \leq T_2$.

Така, ако разполагаме с някаква архитектура, например хиперкуб, можем да преценим, как е по-добре да реализираме задачата. В едни случаи е по-добре да използваме реализацията върху линеен систолически масив, а в други случаи по-подходящо е да използваме реализацията върху двумерен правоъгълен систолически масив.

В предишните глави показахме реализацията на един конкретен алгоритъм върху линеен и двумерен правоъгълен систолически масиви. До тук ставаше дума за един сравнително прост масив, съдържащ само

един тип изчислителни елементи. При по-сложни систолически масиви и особено когато съдържат повече от един тип изчислителни елементи, задачата за реализацията на отделен алгоритъм става изключително трудна. Поради това много актуален е въпросът за автоматизация на процеса на проектиране.

Глава 4

Разширими систолически масиви

Изследователите използват своя опит и интуиция за конструиране на различни систолически масиви, докато въпросът за тяхната оптималност и коректност се изследва след това. Друго направление свързано със задачата за конструиране на систолически масиви е да се открие методология за автоматично трансформиране на алгоритмите върху систолически масиви, които да реализират тези алгоритми коректно. Тази задача се решава по различен начин; трансформации върху системи рекурентни уравнения [8, 12, 20] и оптимизиране [17], геометрични трансформации [13] и др.

В глава четвърта разглеждаме метод за синтез на систолически масиви чрез тяхното разширяване. Този подход е предложен от Хр. Джиджев [9, 8], но там са разгледани само систолически масиви с правоъгълна структура, състоящи се от един тип изчислителни елементи.

Нашият подход се основава на идеята да изследваме директно не алгоритъма и изчислителната архитектура, а структури, които са значително по-малки и все пак запазват основните свойства на изходната архитектура. Малкият размер на тези структури прави възможно разглеждането на голям брой кандидати за изобразяване. Така можем да намерим най-доброто решение по някакъв критерий.

Първо сме разгледали *хомогенни* систолически масиви, т.е. систолически масиви които можем да представим чрез хардуерни графи с върхове оцветени в един и същ цвят. Това съответства на алгоритми при които се извършват само един тип операции. Основно понятие при този метод са рекурсивните редици от графи.

Нека е даден един поточен граф G_0. Всеки връх от този граф ще заменим със същия граф, като ребрата свързващи два съседни върха в G_0 ще заменим с пътища от същият цвят. Върховете по тези пътища ще се наричат *допълнителни*. Доказали сме, че допълнителните върхове поставени за ребра от даден цвят, са допълнителни и за ребрата от останалите цветове. По този начин получаваме граф G_1, който ще наричаме *разширение* на G_0 с G_0. Ако всеки връх на G_1 заменим с граф G_0 получаваме граф G_2, който е разширение на G_1 с G_0 и т.н. получаваме редица от графи породена от граф G_0 която ще наричаме *рекурсивна редица*. Поточен граф, за който можем да построим рекурсивна редица ще наричаме разширим поточен граф. Намерено е необходимо и достатъчно условие граф G да бъде разширим поточен граф. Доказали сме, че ако $G_0, G_1, \ldots$ е рекурсивна редица от графи, породена от поточния граф G_0, то за всяко $i > 0$, G_i също е поточен граф.

Нека $G_0, G_1, \ldots$ и $H_0, H_1, \ldots$ са съответно рекурсивни редици от графи на зависимостите и хардуерни графи, като G_j е граф на зависимостите съответстващ на хардуерния граф H_j. $M_0 = (S_0, T_0)$ е изображение съставено от две изображения, S_0 е изображение на граф G_0 върху граф H_0, а T_0 е времева функция. Рекурсивно дефинираме редица от изображения $M_0, M_1, \ldots, M_j = (S_j, T_j)$. Доказали сме, че S_j е изображение на граф G_j върху граф H_j, а T_j е времева функция.

Разгледали сме и *хетерогенни* систолически масиви, т.е. систолически масиви състоящи се от няколко вида изчислителни елементи. Това означава, че както в програмния така и в хардуерния граф върховете са оцветени с повече от един цвят или това съответства на алгоритми, при които се извършват няколко типа операции. До сега подобен проблем е разглеждан само за хомогенни систолически масиви.

Хетерогенен поточен граф се разширява, като от него получим хомогенни графи. Нека G е поточен хетерогенен граф и нека върховете на G бъдат оцветени в s цвята. От граф G ще получим q на брой графа ($q \geq s$) $D_1, \ldots, D_q$ по следния начин:

1. D_i, $i \in [1, q]$ съдържа нетерминални върхове само от един и същи

цвят и свързващите ги ребра;

2. $\cup_{i=1}^{q} V_n(D_i) = V_n(G) \quad \cup_{i=1}^{q} E_n(D_n) \subset E_n(G)$

3. Ако $x \in V_n(D_i)$, $y \in V_n(D_j)$, $i \neq j$ и $(x,y) \in E_n(G)$ то в граф D_i от върха x излиза терминално ребро с цвят равен на $col(x,y)$, а в граф D_j във върха y влиза терминално ребро от същият цвят.

4. Ако $x \in V_n(D_j)$, $y \in V_n(D_i)$, $(x,y) \in E_n(G)$ и $col(x,y) = c$, тогава графите D_i и D_j имат равен брой терминални ребра от цвят c.

Доказали сме, че $D_1, \ldots, D_q$ са поточни графи. Ако $D_1, \ldots, D_q$ са разширими, то те се разширяват като хомогенни поточни графи, след което разширенията им отново се свързват. Така се получава разширение на хетерогенния граф G. Аналогично на хомогенния случай се построява рекурсивна редица от хетерогенни графи, породени от граф G. Доказано е, че необходимо и достатъчно условие поточният граф G да е разширим е хомогенните графи $D_1, \ldots D_q$ да бъдат разширими. Разгледана е коректността на алгоритъма за разширяване на хетерогенни поточни графи.

Нека е дадена рекурсивна редица породена от хетерогенния поточен граф G_0. Тогава всички графи от редицата също са хетерогенни поточни графи. Ако графите, на които се разделя хетерогенния поточен граф се състоят от един връх, то разширението на графа съвпада със самия граф.

Аналогично на хомогенния случай сме разгледали рекурсивни редици от графи на зависимостите и хардуерни графи и сме построили рекурсивна редица от изображения, така че i-тият член на редицата от изображения изобразява i-тият член на редицата от графи на зависимостите върху i-тият член на редицата от хардуерни графи.

Приноси на дисертацията

Резултатите в дисертационния труд и приносите в него могат да бъдат обобщени както следва:

- Конструирани са линеен и двумерен правоъгълен систолически

масиви за задачата за раницата и подобни на нея задачи. Клетките на тези масиви имат локална памет с фиксиран размер, независещ от параметрите на задачата. Когато няма ограничение за броя на използваните клетки е подобрена времевата сложност и броя на изчислителните елементи. Когато броят на клетките е фиксиран е подобрена времевата сложност.

- Обобщен е алгоритъмът на Хр. Джиджев за разширяване на хомогенни систолически масиви. Новият алгоритъм обхваща по-голям клас систолически масиви. Доказана е коректността на алгоритъма. Намерено е необходимо и достатъчно условие един хомогенен систолически масив да е разширим.

- Предложен е алгоритъм за разширяване на хетерогенни систолически масиви. Доказана е коректността на алгоритъма. Намерено е необходимо и достатъчно условие един хетерогенен систолически масив да бъде разширим.

Литература

[1] **Akl S.G., Calvert M., Stojmenovic I.,** Systolic generation of derangements, *Algorithms and parallel VLSI architectures II, Elsevier Sci. Pub.* , 1992, pp.59-70.

[2] **Andonov R.A., Benaini A.,** A linear Systolic Array for the 0/1 Knapsack Problem, *TR 90-12, LIP-IMAG, Ecol Normale Superieur de Lyon*, 1990.

[3] **Andonov R., Gruau A.,** A 2D toroidal systolic array for knapsack problem, *In Algorithms and Parallel VLSI Architectures II, Bonas, France*, June 1991.

[4] **Andonov R., Quinton P.,Rajopadhye S.,Wilde D.** A Shift-Register based systolic array for the general knapsack problem , *Parallel Procesing Letters, 5(2)*, 1995.

[5] **Andonov R., Rajopadhye S.,** Knapsack on VLSI:From Algorithm to Optimal Circuit, IEEE Transaction on Parallel and Distributed Systems, vol 8(6),1997

[6] **Bollobas B.**,Graph Theory,*Springer-Verlag,New York*,1979.

[7] **Chen G.H., Chern M.S., Jang J.H.,** Pipeline architectures for dynamic programming algorithms, *Parallel Computing 13*, 1990, pp.111-117.

[8] **Djidjev H. N.**, Highly Parrallel Computers, *North-Holand,* 1987, pp.157-174.

[9] **Djidjev, H. N.**, Design and Analysis of Systolic Implementations, *J. of New Gener. Comput. Syst. 1,* 1988, pp.7-20

[10] **S.Fidanova,**Linear array for spelling correction,*Concurency:Practise and Experience, Vol 7*,1997

[11] **Flynn M.J.,** Some computer organisation and their effectiveness, *IEEE Trans Computers 21,9*, 1972, pp.948-960.

[12] **Fortes J.A.B., Moldovan D.I.,** Parallelism detection and transformation techniques useful for VLSI algorithms, *J. Parallel and Distributed Computing,* 1985.

[13] **Kung H.T., Lin W.T.,** Elliptic problems slovers. *Academic Press,* 1984, pp.141-160.

[14] **Kung H.T., Lo S.C., Lewis P.S.,** Optimal systolic design for the transitive closure and the shortest path problems, *IEEE trans Comput. C-36(5)*, 1987, pp.603-614.

[15] **Lee J., Shragowitz E., Sahni S.,** A hypercube algorithm for the 1/0 knapsack problem, *J. of Parallel and Distributed Computing 5*, 1988, pp.483-456.

[16] **Lee P., Kedem Z.,** Synthezing linear array algorithms from nested for loop algorithms,*IEEE Trans Comput., C-37,* 1988, pp.1578-1598.

[17] **Li G.H., Wan B.W.,** The design of optimal systolic arrays, *IEEE Trans. Comp.34(1)*, 1985, pp.66-77.

[18] **Lisper B.,** Computing tranzitive closure on systolic arrays of fixed size, *Distributed Computing 5*, 1991, pp.133-144.

[19] **Moldovan D. I., Fortes J.A.B.,** Partitioning and Mapping of Algorithms into Fixed Size Systolic Arrays, *IEEEE Trans. on Computers, 35(1)*, 1986, pp.1-12.

[20] **Moldovan D. I.,** On the Analysis and Synthesis of VLSI Algorithms, *IEEE Trans. Comput. C-32*, 1982, pp.1121-1126.

[21] **Mongenet C., Clauss Ph., Perrin G.-R.**, Geometrical Tools to Map Systems of Affine Recurrence Equations on Regular Arrays, *Parallel Architectures and Languages Europe, PARLE91, LNC505, Springer Verlag Ed.*, 1995

[22] **Myoupo J.-F., Fabret A.-C.,** Designing modular linear systolic array using dependance graph partition, *IEEE Transaction on Parallel and Distr. Systems*, 1995.

[23] **Quinton P., Van Dongen V.,** The mapping of linear recurrence equations of regular arrays, *J. of VLSI Signal Processing 1*, 1989, pp.95-113.

[24] **Radjopadhye S., Fudjimoto R.,** Synthesing Systolic Arrays for Recurrence Equations, *Parallel Computing 14, North-Holand,* 1990,pp.163-189.

[25] **Stoimenovic I.**, An optimal algorithm for generating equivalence relations on linear array of processors, *BIT 30(3),* 1990,pp.424-436.

[26] **Teng S.,** Adaptive Parallel Algorithm for Integral Knapsack Problem, *J of Parallel and Distributed Computing, 8,*1990, pp.400-406.