行政院國家科學委員會專題研究計畫　成果報告

處理器陣列設計上最佳無相衝線性排程問題之研究

計畫類別：個別型計畫
計畫編號：NSC91-2213-E-009-080-
執行期間：91 年 08 月 01 日至 92 年 07 月 31 日
執行單位：國立交通大學資訊工程學系

計畫主持人：蔡中川

計畫參與人員：黃為霖、王志誠、張瓊文

報告類型：精簡報告

處理方式：本計畫可公開查詢

中　華　民　國 92 年 9 月 26 日

行政院國家科學委員會專題研究計畫成果報告

處理機陣列設計上最佳無相衝線性排程問題之研究

On the Optimal Conflict-Free Linear Scheduling Problem in the Design of Processor Arrays

計畫編號：NSC 91-2213-E-009-080-

執行期限：91 年 8 月 1 日至 92 年 7 月 31 日

主持人：蔡中川　交通大學資訊工程系

計畫參與人員：黃為霖、王志誠、張瓊文　交通大學資訊工程系

一、中文摘要

空時映射法為一種常用之處理機陣列設計方法。一正確之空時映射由不引起計算相衝與通道相衝之線性排程向量及空間映射矩陣所組成。在一指定之空間映射矩陣下，如何選擇一線性排程向量，以建構正確之空時映射並使得處理機陣列之執行時間為最短，是處理機陣列設計上的一個重要研究問題—我們稱此問題為最佳無相衝線性排程問題。過去已有幾位學者提出解此問題之方法，然而，他們的方法不是無法保證找出最佳解，就是僅適用於特定處理機陣列模式或特定形狀之計算定義域。在本計劃中，我們提出一新一致性方法以解決此排程問題。我們的方法除可找出最佳解外，也較過去方法適用於更多種之陣列模式及任意凸形之計算定義域。我們也製作了程式以檢驗新方法之正確性與執行效率。將此程式應用於各種實例之結果，顯示出新方法在大部份情況下具有較過去方法為佳之執行效率，且可設計出過去方法無法設計之處理機陣列。

關鍵詞：計算相衝、通道相衝、最佳線性排程、處理機陣列、空時映射

Abstract

Space-time mapping is frequently employed in designing parallel processor arrays. A correct space-time mapping consists of a linear scheduling vector and a space mapping matrix, which have no computational conflict and link conflict. Under a given space mapping matrix, finding a linear scheduling vector, which contributes a correct mapping and minimizes the total completion time, is an important problem — we call it the *optimal conflict-free linear scheduling problem*. Several methods for solving this problem have been presented in the literature. However, these methods either cannot find the optimal solution or only apply to specific array models and/or index sets. In this project, we propose a new unified method to solve this problem. Our method not only finds the optimal solution but also applies to various array models and arbitrary convex index sets. We have written a computer program to implement our method, and have applied it to several examples. The experimental results showed that our method has better performance in general and that our method can design processor arrays that cannot be designed by previous methods.

Keywords: computational conflict, link conflict, optimal linear schedule, processor array, space-time mapping

二、緣由與目的

平行處理機陣列(processor array)為一類由許多排列規律之處理機互相鄰近連接而成的特殊平行計算機器[9]。由於其互連結構的局部性與規律性，平行處理機陣列極適合採用 VLSI 與 FPG 技術實作之[2,9]。

在平行處理機陣列之設計上，一種常用的方法是空時映射法(space-time mapping) [9,12]，其主要觀念為利用線性排程向量(linear scheduling vector) $H \in Z^{1\times n}$ 及空間映射矩陣(space mapping matrix) $S \in Z^{k\times n}$ 將一規律演算法(regular algorithm)內各計算點(index point) $\vec{j} \in Z^n$ 直接映射至時間 $H\vec{j}$ 及處理機 $S\vec{j}$ 上執行。一有效(valid)之線性排程向量及空間映射矩陣必須避免計算相衝(computational conflict)與通道相衝(link conflict)以確保陣列之正確性[6,7,10,11,13,14]。在一指定之空間映射矩陣下，如何選擇一不引起計算相衝與通道相衝之線性排程向量，使得處理機陣列之執行時間(completion time)為最短，是平行處理機陣列研究領域中的一個重要問題—我們稱此問題為最佳無相衝線性排程問題(optimal conflict-free linear scheduling problem)。此處所謂執行時間意指第一個資料元素進入陣列與最後一個資料元素離開陣列所間隔之時間差。

在過去文獻中，Lee 和 Kedem [11, p.72]最早提出求解最佳無相衝線性排程問題之方法。但是因他們的方法為啟發式(heuristic)方法，故無法保證能求得最佳解。在[13,15]中，Shang 和 Fortes 等人提出另一可保證求得最佳解之列舉方法。然而，此方法限制了規律演算法之計算定義域(index set)只能為矩形之形狀，且由於其不檢查通道相衝，故此方法僅適用於具直接 I/O (direct I/O)之直接處理機互連陣列模式(straight- connection array model) [6]。

在[4,5]中，Ganapathy 和 Wah 提出一個稱為一般參數法(General Parameter Method, GPM)之列舉方法來求解各種目標函數下之最佳處理機陣列。由於 GPM 法允許之目標函數包含執行時間，故其可被用於求解最佳無相衝線性排程問題。但是，因為 GPM 法假設了線性排程向量 H 之計算時間(computation time)必須為 $H\vec{d}_i$ ($\vec{d}_i$ 為資料相依向量)之單調(monotonic)函數[5, p.280]，故其實際上並不適用於任意形狀之計算定義域。

最近，Zimmermann 和 Achtziger [16]提出了一個線性規劃(linear programming)方法來近似求解最佳無相衝線性排程問題。此法並不能保證求得最佳解，但由於其僅需解數個與問題大小(problem size) N 無關之線性規劃程式，故當 N 增大時，此法將較上述各列舉方法有更佳之執行效率。為了能求得最佳解，Zimmermann 和 Achtziger 另外於[17]中提出了一個二次規劃(quadratic programming)方法來求解最佳無相衝線性排程問題。不過，此法需解限制式與變數數目皆與 N 有關之二次規劃程式，故其求解時間一般而言將隨問題大小增加而呈指數成長。與前述 Shang 和 Fortes 等人之方法相同，上述線性規劃與二次規劃方法皆沒有檢查通道相衝，因此它們也一樣僅適用於具直接 I/O 之直接處理機互連陣列模式。

由以上討論可知，過去的方法不是無法保證找出最佳解，就是僅適用於特定形狀之計算定義域或特定處理機陣列模式，故而我們認為最佳無相衝線性排程問題仍保留許多研究空間。在本計劃中，我們仔細研究此排程問題，並藉由推廣[4,5, 8,13,15]中各方法之原理，我們提出了一解此問題之新一致性列舉方法。新方法不但可保證找出最佳解，也允許規律演算法可以有任意凸形(convex)之計算定義域。而且，由於使用了[7]中之新通道相衝檢查條件，新方法也較過去各方法適

用於更多種之處理機陣列模式。此外，我們也進一步製作了電腦程式以具體檢驗新方法之正確性與執行效率。將此程式應用於各種實例之結果，顯示出新方法在大部份情況下具有較過去方法為佳之執行效率，且可設計出過去方法無法設計之處理機陣列。

三、結果與討論

由於篇幅限制，本節僅簡要描述本計劃之研究成果。至於更詳細之內容，我們將另外撰寫論文發表。

3.1 新列舉方法之基本原理

在本小節中，我們簡要說明新方法求解最佳無相衝線性排程問題之基本原理。對一線性排程向量 H，定義其計算時間(computation time)為

$$T_{\text{comp}}(H) = \max\{H(\vec{x}_1 - \vec{x}_2) \mid \vec{x}_1, \vec{x}_2 \in P\} + 1,$$

其中 P 為計算定義域之凸形殼(convex hull)；且令

$$T_{\text{c}}(H) = T_{\text{comp}}(H) + \text{data load and drain times}$$

為 H 之執行時間(completion time)。則我們可觀察最佳無相衝線性排程問題之最佳解 H_{opt} 對任意無相衝線性排程向量 H，皆有

$$T_{\text{comp}}(H_{\text{opt}}) \le T_{\text{c}}(H)$$

之關係。因此，若已知一無相衝線性排程向量 H，則藉由列舉所有計算時間小於 $T_{\text{c}}(H)$ 之排程向量，即可找出所求之最佳解 H_{opt}。

現在，我們接著說明如何列舉所有具有相同計算時間之排程向量。令

$$\text{Diff}(P) = \{\vec{x}_1 - \vec{x}_2 \mid \vec{x}_1, \vec{x}_2 \in P\}$$

為 P 之差異體(difference body)，

$$R^* = \{\vec{y} \in \text{Q}^n \mid \vec{y}^T \vec{x} \le 1 \text{ for all } \vec{x} \in \text{Diff}(P)\}$$

為 Diff(P)之二元體(dual body)。Ke 和 Tsay [8]發現線性排程向量 H 與二元體 R^* 有下列關係：

定理 1 若 $T_{\text{comp}}(H) = t + 1$，則 H^T 爲多面體 $t \cdot R^* = \{t \cdot \vec{x} \mid \vec{x} \in R^*\}$ 任意面(facet)上之一整數點。

由此定理可知，我們可藉由列舉 $t \cdot R^*$ 所有面上之整數點來找出所有計算時間為 $t + 1$ 之線性排程向量。

根據上述討論，我們求解最佳無相衝線性排程問題之列舉方法可描述如下：

程序 1 (最佳無相衝線性排程向量之列舉)

輸入：空間映射矩陣 S；計算定義域 J；資料相依向量 $\vec{d}_1, ..., \vec{d}_m$；處理機陣列模式

參數(參數描述詳見[6])。

輸出：最佳無相衝線性排程向量 H_{opt}。

步驟 1：建構 R^*，並計算遞增量 Δ 及 $T_{\text{comp}}(H_{\text{opt}})$ 之下界(lower bound) t_{lb}。

步驟 2：令 $t = t_{\text{lb}} - 1$，$t_{\text{ub}} = \infty$。

步驟 3：列舉 $t \cdot R^*$ 所有面上之整數點，並根據 S, J, $\vec{d}_1,...,\vec{d}_m$ 及處理機陣列模式參數檢查是否存在有效排程向量。

步驟 4：若有效排程向量不存在，則跳至步驟 6；否則，令 H_t 為 $t \cdot R^*$ 面上具有最短執行時間之有效排程向量。

步驟 5：若 $T_c(H_t) < t_{\text{ub}}$，則令 $H_{\text{opt}} = H_t$，$t_{\text{ub}} = T_c(H_t)$。

步驟 6：令 $t = t + \Delta$。若 $t < t_{\text{ub}}$，則跳至步驟 3；否則，H_{opt} 為最佳無相衝線性排程問題之一解，輸出 H_{opt}。

有關二元體 R^* 之計算，可參考[8]。在以下各小節中，我們將進一步描述程序 1 各重要步驟之詳細內容。

3.2 遞增量 Δ 與 $T_{\text{comp}}(H_{\text{opt}})$ 下界之計算

遞增量 Δ 定義為一對任意線性排程向量 H，皆滿足

$$T_{\text{comp}}(H) = c\Delta + 1, \quad c \in \mathbb{Z}$$

之整數。由程序 1 可觀察，愈大之遞增量，愈可減少不必要 t 值之列舉。在本計劃中，我們證明遞增量可由以下定理計算：

定理 2 (遞增量 Δ 之計算) 令 $\vec{a}_1,...,\vec{a}_h \in \mathbb{Z}^n$ 爲滿足 $R^* = \{\vec{x} \in \mathbb{Q}^n \mid \vec{a}_1^T\vec{x} \le 1,...,\vec{a}_h^T\vec{x} \le 1\}$ 之向量。則

$$\Delta = \gcd(\gcd(\vec{a}_1),...,\gcd(\vec{a}_h))$$

爲一遞增量，其中 $\gcd(\vec{a}_i) = \gcd(a_{i1},...,a_{in})$ 表向量 $\vec{a}_i = [a_{i1},...,a_{in}]^T$ 各分量之最大公因數。

除較大之遞增量外，估計一個緊密(tight)的 $T_{\text{comp}}(H_{\text{opt}})$ 下界是另一個有效減少列舉不必要 t 值之方法。在本計劃中，我們針對大部份凸形計算定義域推導出一個適用的下界，如下：

定理 3 ($T_{\text{comp}}(H_{\text{opt}})$ 下界之計算) 令 $\vec{\alpha}_1,...,\vec{\alpha}_q \in \mathbb{Z}^n$ 爲滿足 $\text{Diff}(P) = \{\vec{x} \in \mathbb{Q}^n \mid \vec{\alpha}_1^T\vec{x} \le 1, ..., \vec{\alpha}_q^T\vec{x} \le 1\}$ 之列向量。若計算定義域 J 滿足

$$\text{Diff}(P) \cap \mathbb{Z}^n = \{\vec{x}_1 - \vec{x}_2 \mid \vec{x}_1, \vec{x}_2 \in J\},$$

則

$$\frac{\left(\left\lfloor 1/\max\left\{\left\|\tilde{\pi}_1\right\|_1,...,\left\|\tilde{\pi}_q\right\|_1\right\}\right\rfloor+1\right)^{n-k-1}}{\max\left\{\left\|\tilde{\pi}_1\right\|_1,...,\left\|\tilde{\pi}_q\right\|_1\right\}\prod_{i=1}^{k}\left\|S_i\right\|_1}+1$$

爲 $T_{\text{comp}}(H_{\text{opt}})$ 之一下界，其中 S_i 表空間映射矩陣 S 之第 i 列。

上述定理實際上為 Shang 和 Fortes 等人[15]所提出之下界的推廣。

3.3 線性排程向量之列舉

在本小節中，我們說明列舉 $t\cdot R^*$ 面上整數點之方法。為減少無效整數點之列舉，我們實際上只列舉滿足資料相依限制之整數點。定義 $t\cdot R^*$ 之第 i 面(facet)為 $F_i=\{\vec{x}\in t\cdot R^* \mid \vec{a}_i^{\,T}\vec{x}=t\}$ ($\vec{a}_i$ 之定義見定理 2)，則我們的方法可描述如下：

程序 2 (列舉 $t\cdot R^*$ 所有面上之整數點)

輸入： 資料相依向量 $\vec{d}_1,...,\vec{d}_m$ ；多面體 $t\cdot R^*$ 。

輸出： $t\cdot R^*$ 所有面上之整數點。

步驟 1： 令 $i=1$。

步驟 2： 應用[1]中之方法將點集合 $F_i'=\{H^T\in F_i \mid H\vec{d}_1\geq 1,...,H\vec{d}_m\geq 1\}$ 表示為

$$\{[h_1,...,h_n]^T \mid lb_j\leq h_j\leq ub_j,\ j=1,...,n\}$$

之形式，其中 lb_j, ub_j 分別為 $h_1,\ldots,h_{j-1}$ 之函數。

步驟 3： 執行下列巢狀迴圈(nested loops)以列舉 F_i' 中之整數點：

for h_1 **from** $\lceil lb_1\rceil$ **to** $\lfloor ub_1\rfloor$ **do**
…
for h_n **from** $\lceil lb_n\rceil$ **to** $\lfloor ub_n\rfloor$ **do**
輸出整數點 $[h_1,\ldots,h_n]^T$。
end for
…
end for

步驟 4： 令 $i=i+1$。若 $i\leq h$，跳至步驟 2；否則，所有整數點已列舉完畢，停止。

3.4 有效線性排程向量之檢查

在程序 1 之步驟 3 中，我們使用了[6]中之虛擬計算點(virtual node)方法來檢查線性排程向量之有效性(包含計算相衝與通道相衝之檢查)。由於此法適用於各種處理機陣列模式，故我們可正確地找出各種陣列模式下的有效線性排程向量。

3.5 實驗結果

為了具體檢驗程序 1 之正確性與執行效率，我們以 MAPLE 程式語言[3]實

作了此程序。表 1 與表 2 列出我們的方法分別應用於矩陣相乘(matrix multiplication) [9, pp.153–154]與 LU 分解(LU decomposition) [9, pp.157–158]演算法上之結果；其考慮的陣列模式為具直接 I/O 之直接處理機互連陣列模式，而執行平台為 Intel Pentium 166。

藉由比較表 1 與[17, Table 3]，我們可觀察到對於較大之問題大小，我們的列舉方法有較 Zimmermann 和 Achtziger [17]之二次規劃方法更短的求解時間。而藉由比較表 2 與[17, Table 4]，我們發現我們的方法可找出執行時間較 Zimmermann 和 Achtziger 之實驗結果更短的線性排程向量。由於篇幅限制，此處我們忽略與其它方法的比較結果。

四、計劃成果自評

我們已達到本計劃之主要預期目標。我們已提出了一求解最佳無相衝線性排程問題之新列舉方法，其不但可找出最佳解，也適用於更多種處理機陣列模式及任意凸形之計算定義域。我們也已製作了電腦程式以檢驗新方法之正確性與執行效率。我們的實驗結果顯示出新方法在大部份情況下具有更佳之執行效率，且可設計過去方法無法設計之處理機陣列。

未來，我們將更深入研究更好的整數點列舉方法與封閉式(closed-form)計算相衝及通道相衝檢查條件[14]，以期進一步改進新方法之執行效率。而且，我們也將應用新方法於更多常見且重要之規律演算法上，以期設計出過去文獻尚未提出之處理機陣列。

五、參考文獻

[1] C. Ancourt and F. Irigoin. Scanning polyhedra with DO loops. In *Proc. ACM Symp. on Principles and Practice of Parallel Programming*, pp. 39–50, 1991.

[2] M. Bednara and J. Teich. Automatic synthesis of FPGA processor arrays from loop algorithms. *The Journal of Supercomputing*, 26:149–165, 2003.

[3] B. W. Char. *MAPLE V language reference manual*. Springer-Verlag, New York, 1991.

[4] K. N. Ganapathy. *Mapping regular recursive algorithms to fine-grained processor arrays*. Ph.D. dissertation, University of Illinois, Urbana- Champaign, 1994.

[5] K. N. Ganapathy and B. W. Wah. Optimal synthesis of algorithm-specific lower-dimensional processor arrays. *IEEE Trans. Parallel and Distri. Sys.*, 7(3):274–287, 1996.

[6] W. L. Huang. *A virtual node approach to checking link conflicts in the mapping of dependence graphs into processor arrays*. Master thesis, National Chiao Tung University, Taiwan, R.O.C., 2001.

[7] J. Y. Ke and J. C. Tsay. An approach to checking link conflicts in the mapping of uniform dependence algorithms into lower dimensional processor arrays. *IEEE Trans. Comput.*, 48(7):732–737, July 1999.

[8] J. Y. Ke and J. C. Tsay. Finding space-optimal linear array for uniform dependence algorithms with arbitrary convex index sets. *Journal of Information Science and Engineering*, 14:743–763, 1998.

[9] S. Y. Kung. *VLSI Array Processor*. Prentice-Hall Int. , Englewood Cliffs, NJ, 1988.

[10] P. Z. Lee and Z. M. Kedem. Synthesizing linear array algorithms from nested for loop algorithms. *IEEE Trans. Comput.*, C-37(12):1578–1598, December 1988.

[11] P. Z. Lee and Z. M. Kedem. Mapping nested loop algorithms into multidimensional systolic arrays. *IEEE Trans. Parallel and Distri. Sys.*, 1(1):64–76, January 1990.

[12] D. I. Moldovan. On the design of algorithms for VLSI arrays. *Proc. of the IEEE*, 71(1):113–120, January 1983.

[13] W. Shang and J. A. B. Fortes. On time mapping of uniform dependence algorithms into lower dimensional processor arrays. *IEEE Trans. Parallel and Distri. Sys.*, 3(3):350–363, May 1992.

[14] J. Xue. Closed-form mapping conditions for the synthesis of linear processor arrays. *Journal of VLSI Signal Processing*, (10):181–199, 1995.
[15] Z. Yang, W. Shang, and J. A. B. Fortes. Conflict-free scheduling of nested loop algorithms on lower dimensional processor arrays. In *Proc. 6th IEEE Int. Parallel Processing Symp.*, pp. 156–164, 1992.
[16] K.-H. Zimmermann and W. Achtziger. Conflict-free scheduling of nested loop Finding space-time transformations for uniform recurrences via branching parametric linear programming. *Journal of VLSI Signal Processing*, 15:259–274, 1997.
[17] K.-H. Zimmermann and W. Achtziger. On time optimal implementation of uniform recurrences onto array processors via quadratic programming. *Journal of VLSI Signal Processing*, 19:19–38, 1998.

表 1. 對矩陣相乘之實驗結果；$S = [0,0,1]$。

N	H_{opt}	$T_c(H_{opt})$	CPU time (sec)
15	[1,15,1]	239	79.7
25	[25,1,1]	649	205.2
27	[27,1,1]	755	252.7

表 2. 對 LU 分解之實驗結果；$S = [1,0,-1]$。

N	H_{opt}	$T_c(H_{opt})$	CPU time (sec)
4	[2,1,1]	13	15.4
8	[6,1,1]	57	33.3
12	[10,1,1]	133	92