行政院國家科學委員會專題研究計畫　成果報告

在多重資料記憶體模組 DSP 架構下變數分割與指令排程方法之探討

計畫類別：個別型計畫
計畫編號：NSC93-2213-E-009-077-
執行期間：93 年 08 月 01 日至 94 年 07 月 31 日
執行單位：國立交通大學資訊工程學系(所)

計畫主持人：陳正

計畫參與人員：李宜軒，柳文斌，李嘉淳，楊偉帆，蔡明憲，廖哲瑩

報告類型：精簡報告

處理方式：本計畫可公開查詢

中　華　民　國 94 年 10 月 28 日

行政院國家科學委員會補助專題研究計畫 **þ 成 果 報 告**
□期中進度報告

在多重資料記憶體模組DSP架構下變數分割與指令排程方法之探討

計畫類別：þ 個別型計畫　　□ 整合型計畫
計畫編號：NSC 93－2213－E－009－077－
執行期間：　　93 年 8 月 1 日至 94 年 7 月 31 日

計畫主持人：陳正
共同主持人：
計畫參與人員：李宜軒、柳文斌、李嘉淳、楊偉帆、蔡明憲、廖哲瑩

成果報告類型(依經費核定清單規定繳交)：þ精簡報告　□完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□ 涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：國立交通大學資訊工程學系

中　華　民　國　　94 年　10 月　27 日

在多重資料記憶體模組 DSP 架構下變數分割與指令排程方法之探討
A Study of Variable Partitioning and Instruction Scheduling Methods for DSP Architecture with Multiple Data-memory Modules

計畫編號：93-2213-E-009-077

一、中英文摘要

在數位訊號處理器的架構設計中，為了開發潛在的記憶體頻寬，時常採用多重資料記憶體模組架構；而要能夠確實得到此架構提供的優勢，則必須配合有效的編譯技術。因此在本計畫中，我們提出三個包含變數分割機制的指令排程法，並制定相關數學模組評估其初步效能。根據我們的分析，新提出的方法不僅在效率上優於既有方法，在效能上也能達到一定程度的改進。

關鍵詞：數位訊號處理器、多重資料記憶體模組、指令排程、迴圈平行化、記憶體存取

Abstract

In order to explore the potential of higher memory bandwidth, multiple on-chip memory modules are attractive for designing modern digital signal processors (DSP). To harvest the benefits provided by this architecture, effective compiler techniques are required. In this project, we propose three variable partitioning and instruction scheduling methods and design related analytic modules to evaluate their preliminary performances. Based on our analyses, the proposed methods are not only more efficient but also more effective compared with existed methods.

Keywords: *Digital Signal Processor* (*DSP*)、*Multiple Data-memory Modules*、*Instruction Scheduling*、*Loop Parallelization*、*Memory Access*

二、前言

由於多媒體通訊持續且快速的發展，需要對影像、音訊及圖形等做壓縮及傳輸的動作，陸續發展出許多新的訊號處理方法。這些訊號處理的過程需要大量且規則的運算，為了提高執行效率並顧及價格/效能 (cost/performance) 比例，針對不同應用程式設計的數位訊號處理器 (digital signal processor, DSP)，乃應運而生 [1-2]。近年來隨著各種嵌入式系統 (embedded system) 及系統晶片 (system-on-a-chip, SoC) 受到廣泛應用及重視，也間接使得數位訊號處理器的發無論在學界或業界，持續佔有一席之地 [3-4]。

大部分 DSP 應用程式都有重覆性 (iterative)，主要由迴圈組成，運算過程則以乘法和

加法為主 [1-2]。為了配合以上特性，DSP 通常包含特殊的乘法器，能在單一時間週期 (clock cycle) 內完成乘法指令；另外由於處理器和記憶體之間的速度間隔 (memory gap) 不斷擴大，也包含固定容量的晶片上記憶體 (on-chip memory)，以避免頻繁存取晶片外記憶體 (off-chip memory) 所產生的時間延遲。DSP 系統大多採用哈佛 (Harvard) 架構，將晶片上記憶體獨立成二個部分，分別儲存指令和資料 (instruction/data memory)；近年來為了進一步提升效能，發展出多重資料記憶體模組 (multiple data memory modules) 的架構，令多筆資料可以同時被存取，以便開發潛在的記憶體頻寬 (potential memory bandwidth) [5-6]。從 DSP 應用程式的特性來看，這種架構確實有助於提升效能；但根據實際硬體架構的設計，並非所有資料都可以被同時存取，必須顧及特定的資料平行存取條件 (parallel access conditions)，因此如何將資料適當的分割 (partition) 儲存在各個資料記憶體模組並安排存取順序，便成為能否真正提升執行效能的關鍵 [7-10]。

本實驗室多年來持續探討排程的研究議題，除了傳統多處理機上的迴圈平行化之外，近期對 DSP 架構的指令排程也有涉獵。在本計畫中我們根據既有的背景知識，針對多重資料記憶體模組的 DSP 架構，探討變數分割及指令排程的議題，希望藉由指令排程的技術充分開發潛在的記憶體頻寬，以期達到提升執效能的目的。

三、研究目的

無論是在何種系統架構下，排程技術的好壞往往均是影響整體效能的關鍵。對於多重資料記憶體模組 DSP 架構，完整的編譯過程應涵蓋以下步驟：uncompacted code generation、code compaction、memory bank assignment (or variable partition)、register assignment 及 memory offset assignment。這些步驟彼此相依但可獨立執行，因此有不少相關文獻針對其中數個步驟設計演算法。在本計畫中我們將重點放在變數分割 (variable partition) 機制的設計，再結合現有的指令排程法，以縮短執行時間為排程目標。

四、文獻探討

在介紹相關文獻之前，我們先大致描述 DSP 應用程式的表現方式。之前提過 DSP 應用程式以迴圈為主，而迴圈又是消耗執行時間最多的部分，所以大部分的 DSP 編譯技術都致力於開發迴圈指令的平行度來縮短執行時間。一個迴圈元素 (iteration) 通常用資料流程圖 (multi-dimensional data flow graph, MDFG) 表示，其中節點代表指令，有向邊代表資料相依性，邊上的數字則代表迴圈元素間的相依距離 (inter-iteration dependence) [7-10]。合理的排程結果必須在不違反資料相依性及系統資源限制下，執行所有資料流程圖中的節點各一次；另外為了進一步提升效能，常用 retiming 技巧重組迴圈元素，以開發迴圈元素間的平

行度 [11]。使用 retiming 技巧之後程式會分成 prologue、repetition 及 epilogue 三部分，有些文獻提出演算法避免儲存 prologue 及 epilogue 程式碼，但不在本計畫探討之內。

根據收集的相關文獻，做變數分割主要有使用 interference graph (IG) [12] 和 variable independence graph (VIG) [7] 二種方式，由於 [7] 已証明使用 VIG 結果較佳，以下我們僅介紹 VIG 的做法。VIG 根據代表應用程式的資料流程圖建構，其中節點代表應用程式存取的變數，無向邊代表二個變數有可能被同時存取，而邊上的數值則表示此二變數該被分割儲存在不同資料記憶體模組的加權值 (priority)。[7] 提出了三種複雜度不同的 VIG 建構方式，可以顯示不同的精確性，在此省略不做詳述。VIG 建構完成之後，依照資料記憶體模組個數切成數個獨立的分割 (partition)，並令各分割間相連邊上的加權值總和最大，產生初始的變數分割結果。得到變數分割結果之後，[7] 將它結合 rotation scheduling [13]，對所有資料流程圖內的指令做排程。結合後的方法名為 rotation scheduling with variable repartitioning (RSVR)，在各個 rotation phase 若是無法令迴圈元素長度縮短，會嘗試做適當的變數重新分割 (variable repartition)，改變某些變數儲存的資料憶體模組，令迴圈元素長度可以順利縮短。由此可知 RSVR 不僅 rotation 動作相當複雜，建構 VIG 的過程也不簡單，使得 RSVR 雖然整體看來效能不錯，但在時間複雜度上仍有改進空間。

五、研究方法

在多重資料記憶體模組 DSP 架構下要降低整體執行時間，顯然必須令所有存取的變數平均分散儲存在各個資料記憶體模組，才能將資料流程圖中的記憶體存取指令分散排程，不會集中在某些特定的資料記憶體模組。觀察 RSVR 的執行過程，我們認為因為它無法確定經由切割 VIG 所決定的變數儲存結果能否符合上述情形，所以在指令排程過程中需要藉由 variable repartition 來做彌補。有鑑於此，在本計畫中我們設計三種簡單有效的變數分割機制，同樣結合 rotation scheduling，提出三種用於多重資料記憶體模組 DSP 架構的指令排程法，以下分別簡介。

首先值得一提的是，在資料流程圖中定義的變數指的是陣列而不是單一變數。 [7, 12] 的方法都是將同一陣列的所有元素 (component) 存在同一個資料記憶體模組，但我們設計的三個變數分割機制都不遵循這個模式，而是將各個陣列元素分散儲存，以降低執行複雜度。我們提出的第一個指令排程法名為 rotation scheduling with unfolding (RSF)，它根據陣列元素的最右端指標 (rightmost index) 值決定變數儲存的資料記憶體模組，配合 loop unfolding 技術展開內層迴圈，再以 rotation scheduling 做指令排程。相較於 RSVR，RSF 避開了複雜的建構切割 VIG 以及 variable repartition 過程，執行效率明顯提升；更重要的是經過 loop unfolding 之後，不僅記憶體存取指令會平均分散排在各個資料記憶體模組，也能同時考慮相鄰迴圈元素的資料相依性，有助於縮短整體排程長度。根據評估 RSF 在效能效率

二方面都有不錯的表現，詳細結果留待下章節討論。

雖然 RSF 對大部分 DSP 應用程式都能達到理想排程結果，但也因為我們使用 loop unfolding 技術展開內層迴圈，若是相鄰迴圈元素的 critical path 在迴圈展開後互相串接 (cascade)，便會使排程長度無法如預期般經由 rotation scheduling 方法縮短。為了克服 RSF 的這個缺點，我們提出第二個名為 rotation scheduling with tiling (RST) 的指令排程法，它的執行流程與 RSF 類似，但改用陣列元素的最左端指標 (leftmost index) 值決定變數儲存的資料記憶體模組，配合 loop tiling 技術展開外層迴圈，再以 rotation scheduling 做指令排程。由此可見，RST 與 RSF 相同有不錯的執行效率，而 loop tiling 的結果也能令記憶體存取指令平均分散排程，不會形成排程長度的瓶頸。在下章節中我們會列出 RST 的評估結果並討論對於不同 DSP 應用程式它與 RSF 的適用情形。關於 RSF 和 RST 的相關研究成果已發表於會議論文 [8] 並整理成期刊論文投出 [9]。

既然 loop unfolding 之後可能會因為相鄰迴圈元素 critical path 串接導致 RSF 效能不佳，RST 也可能會有類似情形。根據觀察 RSF 和 RST 的排程結果，我們發現若是能先令相鄰迴圈元素彼此獨立，意即沒有資料相依關係，經由 loop unfolding 或是 loop tiling 展開之後，便能使 RSF 或 RST 達到最理想的排程結果。有鑑於此，我們結合 unimodular transformations 技術提出第三個名為 rotation scheduling with parallelization (RSP) 的指令排程法，以期達成上述目的。在 RSP 中我們設計較特別的變數分割機制，仍然根據陣列元素的指標，但沒有一般性的通則，僅針對 2~4 個資料記憶體模組的 DSP 架構設計；另外也提出一個簡單的演算法，使用 unimodular transformations 技術平行化內層迴圈，之後再套用 loop unfolding 技術及 rotation scheduling 做指令排程。相較於 RSF 和 RST，RSP 多了迴圈平行化的步驟，但這個動作已在傳統平行編譯技術的研究領域發展出完整的演算法，並不會對 RSP 造成太多執行上的負擔，因此整體來說 RSP 仍然相當有效率。至於效能方面，根據評估 RSP 與 RSF 和 RST 相差不遠，但因為迴圈平行化過程中可能造成些許額外負擔，使得 RSP 在某些 DSP 應用程式效能不如 RSF 及 RST。詳細評估結果留待下章節討論。關於 RSP 的相關研究成果已發表於會論文 [10]。

六、結果與討論

本計畫中提出的三個用於多重資料記憶體模組 DSP 架構的指令排程法，我們除了演算法的設計之外，也制定對應的數學模組計算所需執行時間，並選取數個表示 DSP 應用程式的資料流程圖來做初步測試評估。表格 1~2 列出在不同系統資源下，各個 DSP 應用程式使用包含 list scheduling、RSVR、RSF 和 RST 四種指令排程法得到的單一迴圈元素排程長度，表格 3 則加入 RSP 排程法並列出 retiming depth。需要注意的是本計畫提出的三個方法在排程前都經過 loop unfolding 或 loop tiling 將迴圈展開，單一迴圈元素的內容其實相當於數個

原始迴圈元素，因此這三個方法實際上得到的排程長度，應該是表格中數值除以資料記憶體模組的個數。圖 1 我們套用由數學模組求出的公式，繪出各個應用程式在不同迴圈大小時，使用 RSVR、RSF/RST 及 RSP 等方法所得到的整體執行時間。由於 RSF 和 RST 可能會因為 critical path cascade 關係導致效能不佳，因此每個應用程式我們只繪出使用這二個方法結果較好的一種。由這些結果看來，list scheduling 最簡單但效能最差，因為它沒有使用 retiming 技術開發相鄰迴圈元素間的平行度；我們提出的三個方法得到的單一迴圈元素排程長度大致相同，通常也都優於 RSVR；另外從表格 3 可以看出 RSP 對應的 retiming depth 遠小於其他三種方法，這種情形使得 RSP 雖然單一迴圈元素的排程長度與 RSF 和 RST 類似，卻有助於縮短整個迴圈所需的執行時間，圖 1 可以明顯看出 RSP 的這個優勢。整體來說，本計畫中我們提出的三個方法不僅執行效率較 RSVR 為高，在效能上也有一定的改進；另外也因為它們將所有變數平均分散儲存，可以平衡各個資料記憶體模組儲存的陣列元素數量，雖然這個特性在此類架構中不見得必要。

最後我們討論在不同 DSP 應用程式中三種指令排程法的適用情形。根據資料流程圖中迴圈元素的相依距離，可以歸納出在何種條件下套用 loop unfolding 或 loop tiling 技術後會發生相鄰迴圈元素的 critical path cascade 情形，進而在條件成立時避免使用 RSF 或 RST。另外資料流程圖若是存在某些特定模式的相依距離，該對應迴圈在套用 loop tiling 技術之前必須先做適當的轉換，也會產生不佳的 RST 排程結果。至於 RSP，雖然它的效能不錯，但其迴圈平行化過程會產生相較於 RSF 和 RST 更多的額外負擔，因此我們建議它只在欲執行之迴圈元素個數較多時使用，才能抵消掉這些負擔。

七、計畫結果自評

在本計畫中我們針對多重資料記憶體模組 DSP 架構，提出三個包含變數分割機制的指令排程法，並制定對應的數學模組，選擇數個 DSP 應用程式做初步的效能評估。總體而言，本計畫大致達成以下幾點目標：

1. 提出 RSF 方法，設計簡單有效率的變數分割機制，結合 loop unfolding 技術及 rotation scheduling，同步提升既有排程法 RSVR 的執行效率及效能。
2. 提出 RST 方法，設計與 RSF 不同但同樣有效率的變數分割機制，結合 loop tiling 技術及 rotation scheduling，解決 RSF 可能產生的缺點。
3. 提出 RSP 方法，設計第三種較特別的變數分割機制，結合 unimodular transformations 和 loop unfolding 技術以及 rotation scheduling，同時避免 RSF 和 RST 可能產生的缺點，進一步提升執行效能。
4. 制定以上三種指令排程法對應的數學模組，並選擇數個表示 DSP 應用程式的資料流程圖，對提出的方法做初步效能評估。

由以上幾點可知，本計畫確實能將多重資料記憶體模組 DSP 架構上的變數分割及指令排程議題做詳細深入的探討，改善既有方法的效能，嘗試提出新的機制，並制定數學模組評估其初步效能。整體看來，我們提出的方法無論在效能效率上都能達到預期的結果，有效降低所需的排程時間及排程長度。這些研究成果均已發表於會議論文並整理成期刊論文投出，後續發展及其他相關排程議題我們也將持續研究。

參考文獻

[1] V. K. Madisetti, **VLSI Digital Signal Processors: An Introduction to Rapid Prototyping and Design Synthesis**, Butterworth-Heinemann, 1995.

[2] P. Lapsley, J. Bier, A. Shoham, and E. A. Lee, **DSP Processor Fundamentals: Architectures and Features**, Berkeley Design Technology, Inc. 1996.

[3] J. Eyre and J. Bier, "The Evolution of DSP Processors", *IEEE Signal Processing Magazine*, Vol. 17, Issue 2, pp. 43-51, March 2000.

[4] H. de Man, "System-on-Chip Design: Impact on Education and Research", *IEEE Design & Test of Computers*, Vol. 16, Issue 3, pp. 11-19, July-Sep. 1999.

[5] W.-F. Lin, S. K. Reinhardt, and D. Burger, "Designing a Modern Memory Hierarchy with Hardware Prefetching", *IEEE Transactions on Computers*, Vol. 50, No. 11, pp. 1202-1218, Nov. 2001.

[6] R. Leupers and D. Kotte, "Variable Partitioning for Dual Memory Bank DSPs", *Proc. of International Conference on Acoustics, Speech, and Signal Processing*, Vol. 2, pp. 1121-1124, 2001.

[7] Q. Zhuge, B. Xiao, and E. H.-M. Sha, "Exploring Variable Partitioning for Dual Data-memory Bank Processors", *Proc. of 34th International Symposium on Microarchitecture*, pp. 45-52, Dec. 2001.

[8] Y.-H. Lee and C. Chen, "Efficient Variable Partitioning and Scheduling Methods of Multiple Memory Modules for DSP", *Proc. of 10th Workshop on Compiler Techniques for High-Performance Computing*, pp. 80-89, March 2004.

[9] Y.-H. Lee and C. Chen, "Efficient Variable Partitioning and Scheduling Methods of Multiple Memory Modules for DSP", submitted to *Journal of Architecture.*

[10] Y.-H. Lee and C. Chen, "An Effective Variable Partitioning and Scheduling Algorithm for DSP with Multiple Memory Modules", *Proc. of International Computer Symposium*, Dec. 2004.

[11] C. E. Leiserson and J. B. Saxe, "Retiming Synchronous Circuitry", *Algorithmica*, Vol. 6, No. 1, pp. 5-35, June 1991.

[12] M. A. R. Saghir, P. Chow, and C. G. Lee, "Exploiting Dual-memory Banks in Digital Signal Processors", *Proc. of 7th International Conference on Architecture Support for Programming Language and Operating Systems*, pp. 234-243, 1996.

[13] N. L. Passos and E. H.-M. Sha, "Scheduling of Uniform Multi-dimensional Systems under Resource Constraints", *IEEE Transactions on VLSI Systems*, Vol. 6, No. 4, pp. 719-730, Dec. 1998.

表 1. 實驗結果 (1 multiplier and 1 adder).

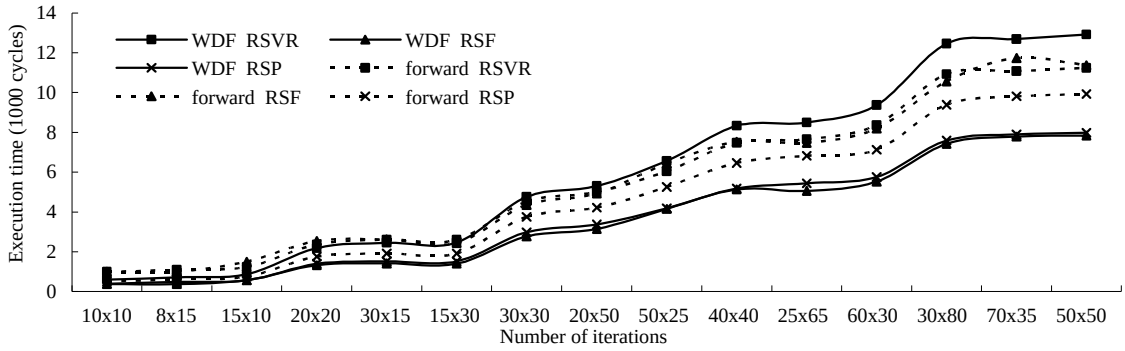
	2 memory modules				3 memory modules				4 memory modules			
	List	RSVR	RSF	RST	List	RSVR	RSF	RST	List	RSVR	RSF	RST
Wave Digital Filter	9	5	9	10	9	5	9	10	9	3	9	10
Forward-substitution	12	6	11	11	12	4	12	11	12	4	12	15
IIR 2D	26	24	40	40	20	14	40	40	20	11	50	43
Filter	13	6	10	10	13	5	12	11	13	4	13	12
Discrete Fourier Transform	16	14	28	28	15	10	31	28	15	7	29	29
THCS	7	4	8	8	7	4	10	8	7	2	10	8

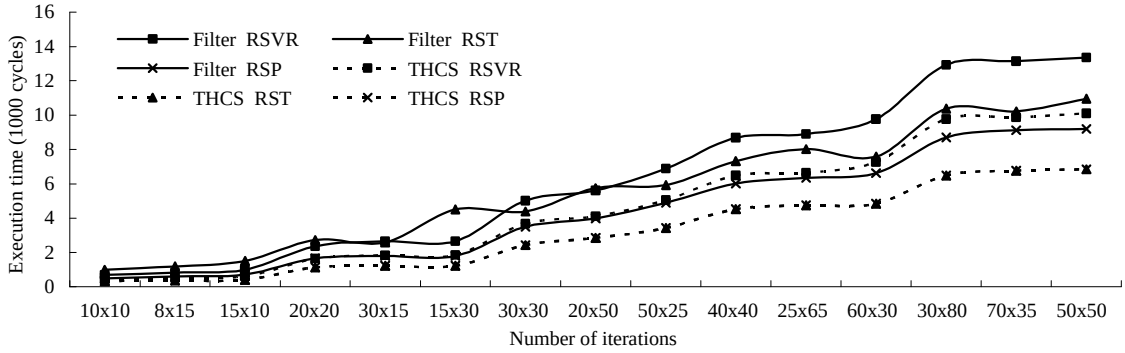
表 2. 實驗結果 (2 multipliers and 2 adders).

	2 memory modules				3 memory modules				4 memory modules			
	List	RSVR	RSF	RST	List	RSVR	RSF	RST	List	RSVR	RSF	RST
Wave Digital Filter	9	5	9	9	9	5	9	10	9	3	9	10
Forward-substitution	12	6	11	11	12	4	11	11	12	4	11	12
IIR 2D	26	24	40	40	20	14	40	40	20	11	40	40
Filter	13	6	10	10	13	5	11	10	13	4	11	10
Discrete Fourier Transform	16	14	28	28	14	10	28	28	13	8	29	28
THCS	6	4	8	8	6	4	8	8	6	2	8	8

表 3. 實驗結果 (2 multipliers and 2 adders) (length, d).

	2 memory modules				3 memory modules				4 memory modules			
	List	RSVR	RSF	RST	List	RSVR	RSF	RST	List	RSVR	RSF	RST
Wave Digital Filter	5,2	9, 1	9, 3	9, 1	5, 3	9, 1	10, 5	9, 1	3, 6	9, 2	10, 8	9, 1
Forward-substitution	6, 3	11, 3	11, 4	11, 1	4, 8	11, 5	11, 8	11, 2	4, 8	11, 9	12, 12	11, 2
IIR 2D	24, 1	40, 1	40, 1	40, 1	14, 1	40, 1	40, 1	40, 1	11, 1	40, 2	40, 3	40, 1
Filter	6, 2	10, 5	10, 3	10, 1	5, 3	11, 9	10, 6	10, 1	4, 3	11, 15	10, 7	10, 1
Discrete Fourier Transform	14, 1	28, 2	28, 0	28, 0	10, 1	28, 4	28, 1	28, 1	8, 3	29, 6	28, 1	28, 1
THCS	4, 2	8, 2	8, 0	8, 0	4, 2	8, 3	8, 1	8, 1	2, 4	8, 5	8, 1	8, 1





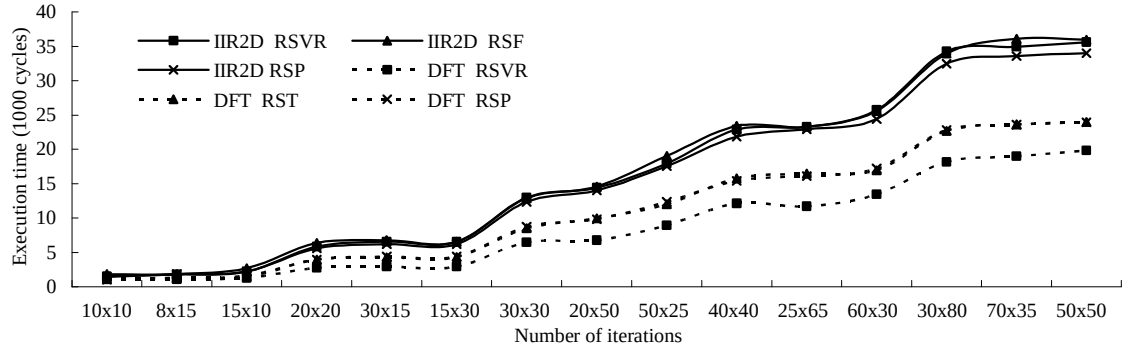


圖 1. 實驗結果 (2 multipliers, 2 adders, and 3 memory modules).