科技部補助專題研究計畫成果報告
期末報告

子計畫二：加寬匯流排考量下，多核心系統之記憶體層級架構設計(3/3)

計 畫 類 別 ： 整合型計畫
計 畫 編 號 ： MOST 103-2220-E-007-013-
執 行 期 間 ： 103 年 05 月 01 日至 104 年 04 月 30 日
執 行 單 位 ： 國立清華大學資訊工程學系（所）

計 畫 主 持 人 ： 黃婷婷

計畫參與人員 ： 碩士班研究生-兼任助理人員：梁澤凱
碩士班研究生-兼任助理人員：宋品萱
碩士班研究生-兼任助理人員：何培安
碩士班研究生-兼任助理人員：邱建邦
博士班研究生-兼任助理人員：許博揚
博士班研究生-兼任助理人員：陳衍昊

報 告 附 件 ： 出席國際會議研究心得報告及發表論文

處 理 方 式 ：
1.公開資訊：本計畫可公開查詢
2.「本研究」是否已有嚴重損及公共利益之發現：否
3.「本報告」是否建議提供政府單位施政參考：否

中 華 民 國 104 年 07 月 31 日

中 文 摘 要 ：

隨著製程技術的進步以及物理材質限制下，對單核處理器(Uni-processor)縮小製程，提升之效能已經遇到瓶頸，所增加之功率消耗也成為嚴重的負擔。透過 Gustafson ‘s law 得知，運用多核心處理器(Many-core) 能夠提升平行化工作的效能，藉此改善系統效能並且有效控制耗能需求。目前多核心架構下，記憶體 層級(Memory hierarchy)設計中，晶片內快取記憶體架構主要分為私有式快取記憶體(Private cache)與分 享式快取記憶體(Shared cache)。私有式快取記憶體主要概念為將快取記憶體分成好幾個區塊，每個區 塊都只由特定處理器作存取動作。而分享式快取記憶體則能讓所有處理器存取。現有記憶體階級設計 下，私有式快取記憶體通常應用於需要快速存取的裝置，例如與處理器頻繁傳輸資料的 L1 或 L2 快取 記憶體。而分享式快取記憶體由於容量使用效益較好，通常應用在最後層快取記憶體上(Last level cache)，確保應用程式的工作資料量(workload)能夠盡量保存在晶片記憶體內(On-chip memory)，不需要 存取速度較慢的晶片外部記憶體 (off-chip memory) 。晶片外部記憶體通常為動態隨機存取記憶體 (DRAM)設計，多核心處理器能透過多個記憶體控制器(Memory controller)同時地存取資料。許多文獻針對晶片內快取記憶體(On-chip cache memory)以及晶片外動態隨機存取記憶體(Off-chip DRAM memory)作相關研究，針對不同層記憶體架構，提出不同技術改善多核心系統的執行效率、功 率消耗、服務平衡(Quality of service)…等等目的。我們在這個計畫提出三個創新技術：1. 考量執行緒關鍵性的動態快取記憶體重整機制；2. 免壓實之資料壓縮式快取記憶體機制；3. 多執行緒下減少記憶體記憶庫衝突的資料管理機制。目的分別為改良動態快取記憶體機制來進一步降低快取記憶體的耗能、設計新的免壓實式的資料壓縮快取記憶體來增加執行效率以及降低耗能、設計新的控制技術來降低平成程式中再主記憶體內記憶庫衝突。

中文關鍵詞： 動態快取記憶體、壓縮快取記憶體、資料搬移、多執行緒應用程式、多核心系統、記憶體記憶庫衝突。

英 文 摘 要 ：

With the advances of VLSI design technology, the performance of
Uni-processors has reach its limitation. The increasing power dissipation also becomes a great burden of the design. Through Gustafson’ s law, many-cores system can improve the system performance and reduce energy consumption. For memory hierarchy design of multi-cores system, cache inside the chip

can be divided into two categories, one is private cache and the other is shared cache. The main idea of private cache is to divide the cache into several partitions, each partition can be accessed by a specific processor. On the other hand, shared cache can be accessed by all processors. Under modern design, private caches are often used as first level caches which need to access the cache frequently, such as L1 and L2 caches. Share caches often have large capacities, such as last level caches. This can avoid the system from accessing on-chip or off-chip memory. A multi-core system may have multiple memory controllers.
Many researches have focused on the topics about on-chip cache memorys and off-chip memrys and present their new methods to improve performance, energy consumption and quality of service…etc. In this project, we proposed three different new technologies: 1. Thread-criticality aware dynamic cache reconfiguration in multi-core system ； 2. Compaction-free compressed cache for high performance multi-core system ； 3. Dynamic Data Migration to Eliminate Bank-level Interference for Data Parallel Applications in Multicore Systems. The objectives of these methods are to improve the energy efficiency of reconfigurable cache by predicting the criticalities of threads, to design new compaction-free compressed cache to improve performance and decrease the energy consumption, to design new dynamic data migration technique to eliminate memory contention inside the main memory.

英文關鍵詞： reconfigurable cache、compressed cache、data migration、multi-threaded applications、multi-cores system、memory contention.

一、中英文摘要:

英文摘要：

With the advances of VLSI design technology, the performance of Uni-processors has reach its limitation. The increasing power dissipation also becomes a great burden of the design. Through Gustafson’s law, many-cores system can improve the system performance and reduce energy consumption. For memory hierarchy design of multi-cores system, cache inside the chip can be divided into two categories, one is private cache and the other is shared cache. The main idea of private cache is to divide the cache into several partitions, each partition can be accessed by a specific processor. On the other hand, shared cache can be accessed by all processors. Under modern design, private caches are often used as first level caches which need to access the cache frequently, such as L1 and L2 caches. Share caches often have large capacities, such as last level caches. This can avoid the system from accessing on-chip or off-chip memory. A multi-core system may have multiple memory controllers.

Many researches have focused on the topics about on-chip cache memorys and off-chip memrys and present their new methods to improve performance, energy consumption and quality of service…etc. In this project, we proposed three different new technologies: 1. **Thread-criticality aware dynamic cache reconfiguration in multi-core system ; 2. Compaction-free compressed cache for high performance multi-core system ; 3. Dynamic Data Migration to Eliminate Bank-level Interference for Data Parallel Applications in Multicore Systems.** The objectives of these methods are to improve the energy efficiency of reconfigurable cache by predicting the criticalities of threads, to design new compaction-free compressed cache to improve performance and decrease the energy consumption, to design new dynamic data migration technique to eliminate memory contention inside the main memory.

英文關鍵詞：reconfigurable cache、compressed cache、data migration、multi-threaded applications、multi-cores system、memory contention.

中文摘要：隨著製程技術的進步以及物理材質限制下，對單核處理器(Uni-processor)縮小製程，提升之效能已經遇到瓶頸,所增加之功率消耗也成為嚴重的負擔。透過 Gustafson's law 得知,運用多核心處理器(Many-core) 能夠提升平行化工作的效能，藉此改善系統效能並且有效控制耗能需求。目前多核心架構下，記憶體 層級(Memory hierarchy)設計中，晶片內快取記憶體架構主要分為私有式快取記憶體(Private cache)與分 享式快取記憶體(Shared cache)。私有式快取記憶體主要概念為將快取記憶體分成好幾個區塊，每個區 塊都只由特定處理器作存取動作。而分享式快取記憶體則能讓所有處理器存取。現有記憶體階級設計 下，私有式快取記憶體通常應用於需要快速存取的裝置，例如與處理器頻繁傳輸資料的 L1 或 L2 快取 記憶體。而分享式快取記憶體由於容量使用效益較好，通常應用在最後層快取記憶體上(Last level cache)，確保應用程式的工作資料量(workload)能夠盡量保存在晶片記憶體內(On-chip memory)，不需要 存取速度較慢的晶片外部記憶體 (off-chip memory)。晶片外部記憶體通常為動態隨機存取記憶體(DRAM)設計，多核心處理器能透過多個記憶體控制器(Memory controller)同時地存取資料。

許多文獻針對晶片內快取記憶體(On-chip cache memory)以及晶片外動態隨機存取記憶體(Off-chip DRAM memory)作相關研究，針對不同層記憶體架構，提出不同技術改善多核心系統的執行效率、功 率消耗、服務平衡(Quality of service)…等等目的。我們在這個計畫提出三個創新技術：**1. 考量執行緒關鍵性的動態快取記憶體重整機制**；**2. 免壓實之資料壓縮式快取記憶體機制**；3. **多執行緒下減少記憶體記憶庫衝突的資料管理機制**。目的分別為改良動態快取記憶體機制來進一步降低快取記憶體的耗能、設計新的免壓實式的資料壓縮快取記憶體來增加執行效率以及降低耗能、設計新的控制技術來降低平成程式中再主記憶體內記憶庫衝突。

關鍵詞 **：** 動態快取記憶體、壓縮快取記憶體、資料搬移、多執行緒應用程式、多核心系統、記憶體記憶庫衝突。

二、研究計畫之背景及目的

(1) Architectural Full System Simulator (第一年**)**：

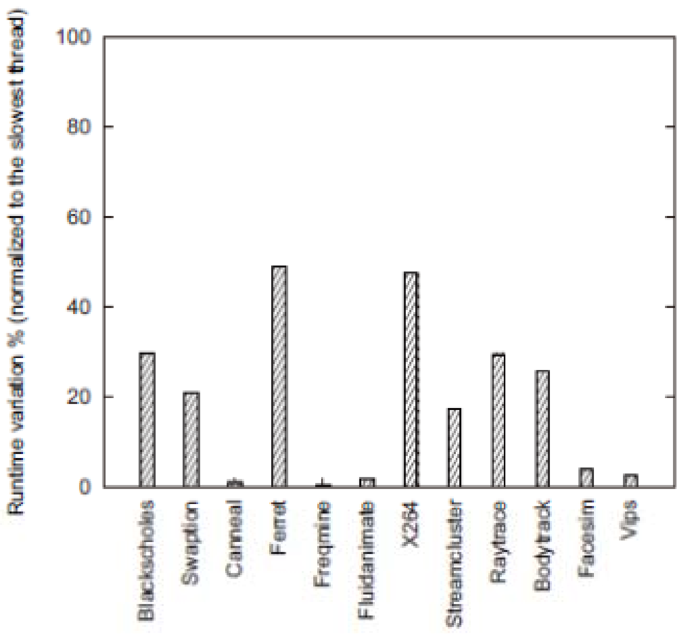
我們架構並熟悉 GEMS 和 Simics 系統，在考量 Wide I/O DRAM memory 架構下，針對記憶體層級 設計(Memory hierarchy design)和記憶體控制器 (DRAM controller)作相關模擬研究。

(2) Memory Hierarchy Design (第一、二、三年**)**：

我們主要針對多核心平台做了三種不同的記憶體架構的創新技術。

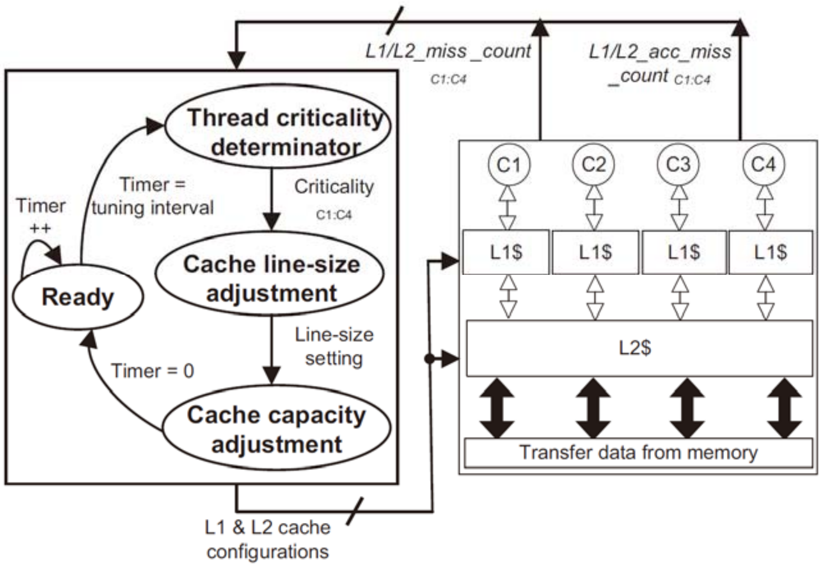
主要創新技術一： 考量執行緒關鍵性的動態快取記憶體重整機制

動態快取記憶體機制已被有效運用來降低快取記憶體的耗能。在這創新技術中，我們動態去預測多核心環境中執行緒的關鍵性(Thread criticality) ，並根據不同的關鍵性去對快取記憶體 Cache line size 與 Cache way 做動態調整。圖一為我們針對 multi-threaded applications 去測量各 thread 的 criticality 差異性。

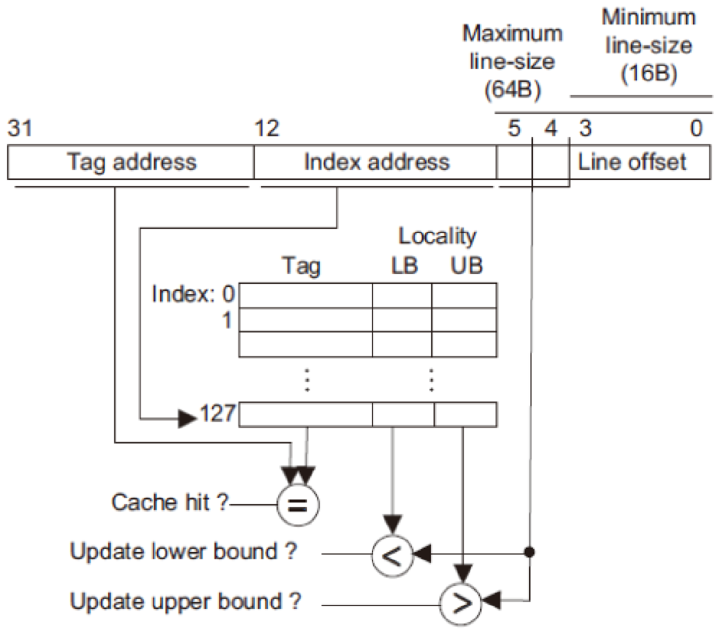


圖一 PARSEC benchmarks criticality 差異性

根據執行時不同 thread 的 criticality，我們設計的動態快取記憶體機制可以將 non-critical thread 的沒有使用到的 cache memory 來減少耗能，並且自動選擇最適合 cache line size 給 critical thread。圖二為我們提出的機制。機制中會有一硬體表來記錄 data locality 並在動態調整 cache line size 時使用，如圖三所示。


L1/L2_miss_count C1:C4
L1/L2_acc_miss_count C1:C4
Thread criticality determinator
Timer = tuning interval
Criticality C1:C4
Timer ++
Ready
Cache line-size adjustment
Line-size setting
Timer = 0
Cache capacity adjustment
C1
C2
C3
C4
L1$
L2$
Transfer data from memory
L1 & L2 cache configurations

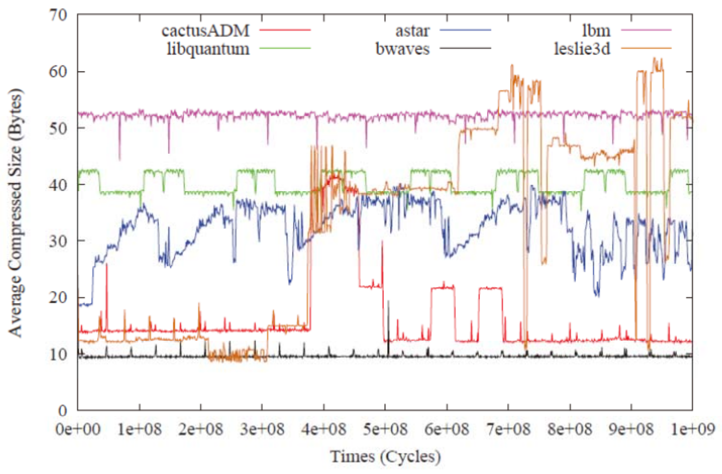
圖二 考量執行緒關鍵性的動態快取記憶體重整機制



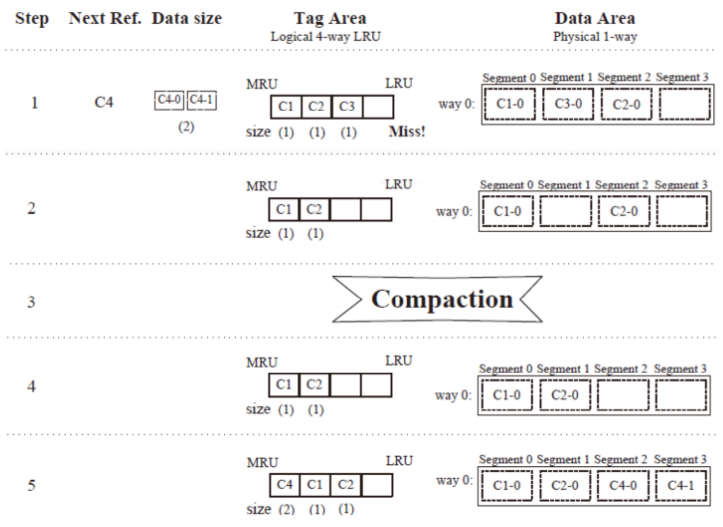
圖三 動態硬體表記錄 memory access bound

主要創新技術二：免壓實之資料壓縮式快取記憶體機制

資料壓縮式快取記憶體為目前為一有效的增加 Last level cache 的 capacity 的方法。但資料壓縮時，資料區塊大小可能有多種大小，如圖四所示。這導致快取記憶體為了容納這些大小不一的壓縮資料區塊時會資料破碎 fragmentation 的問題存在。所以一般為了解決這個問題，會使用壓實 compaction 這個方法如圖五所示。

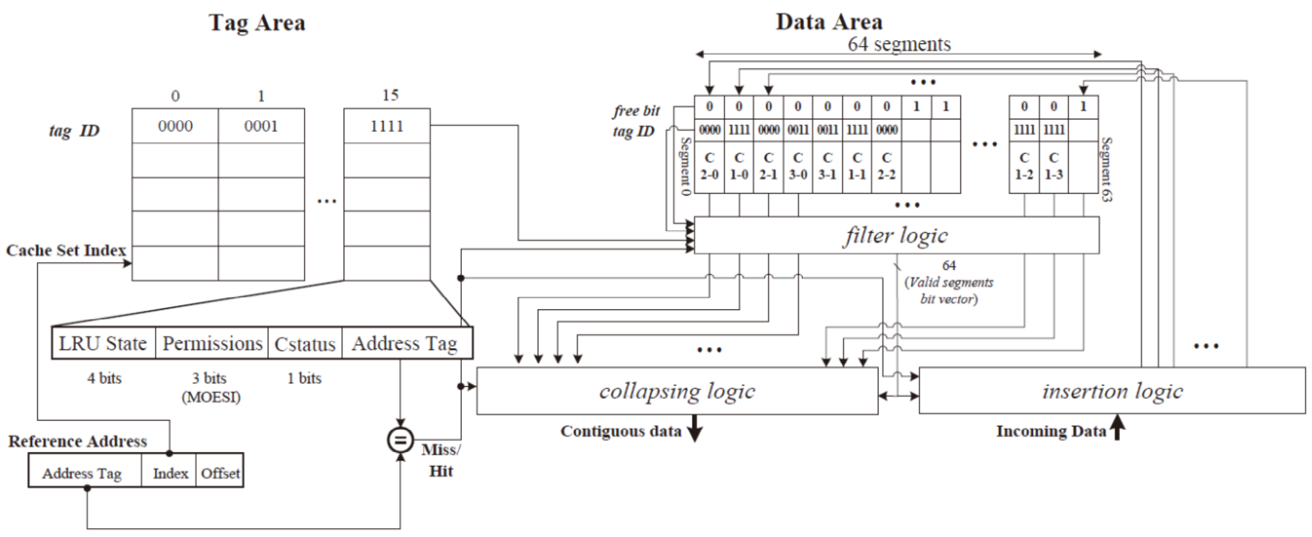


圖四 SPEC CPU 2006 中程式平行執行過程中 data compression size 變化



圖五 Compressed Cache 中 Compaction 的使用

圖六為我們新設計的 compaction-free architecture。新增的元件為 filter logic、collapsing logic 與 insertion logic，filter logic 用來挑選哪些資料區塊為目標資料區塊，collapsing logic 把這些目標資料區塊合併為一連續區塊，而 Insertion logic 則為把連續資料分散到不同小單位的資料區塊的元件。

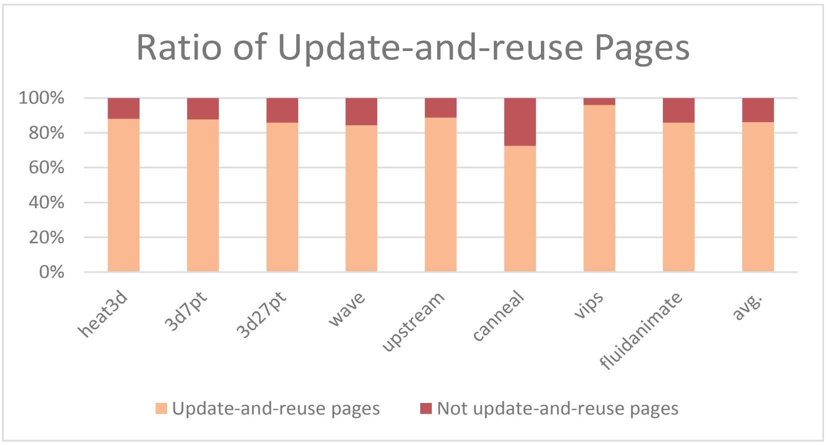


圖六 The overall architecture of our *compaction-free compressed cache*

主要創新技術三：多執行緒下減少記憶體記憶庫衝突的資料管理機制

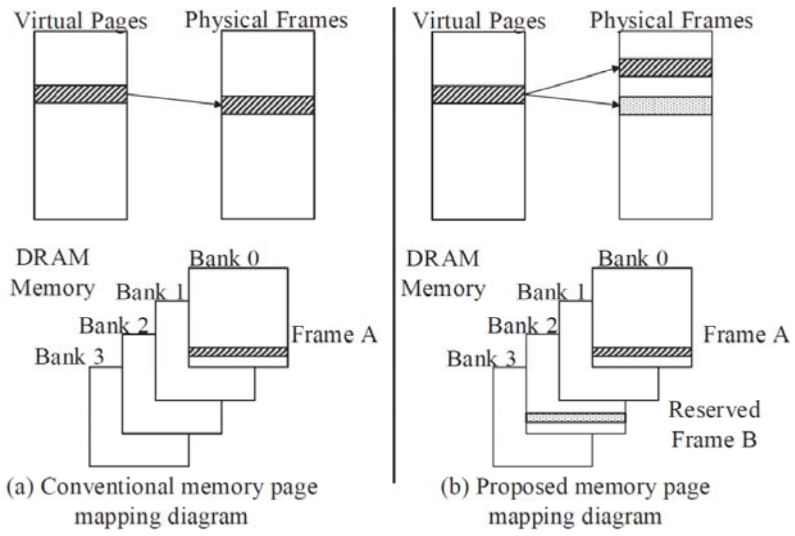
考量多執行緒平行程式中對於不同資料區塊做平行化的應用軟體，像是 Stencil applications，對於資 料的使用往往大於快取記憶體能容納的容量，其不同執行緒對記憶體的存取就會產生記憶庫衝突 (memory banks conflicts) 。先前的研究對於多核心架構上主記憶體記憶庫衝突的解決方法可分為單 純從作業系統上的記憶體內存頁分配 (page allocation)來解決[1]，單純從記憶體控制器的調度方式 (memory controller scheduling)上解決[2]，或者是由作業系統與記憶體控制器一起合作來解決問題， 像是 page migration 等等的做法[3]。然而，當多核心平台上執行的程式為多執行緒平行程式，尤其 是針對不同資料區塊做處理的平行程式時，由於使用的資料為每個執行緒所共享的，上述隔離干擾 的研究方法會無法奏效。所以我們根據多執行緒程式動態調整在主記 憶體中的資料位置，進而降低記憶體記憶庫衝突，提升系統效能。

為了動態減少記憶庫衝突的發生，我們發現在一般 data parallel applications 的記憶體存取的模式中存在一種 update-and-reuse 的模式，即更新過的資料有很大的機率會再被同一個 thread 讀取到，圖七為各種 data-parallel applications memory access 行為中，update-and-reuse pages 所占比例。

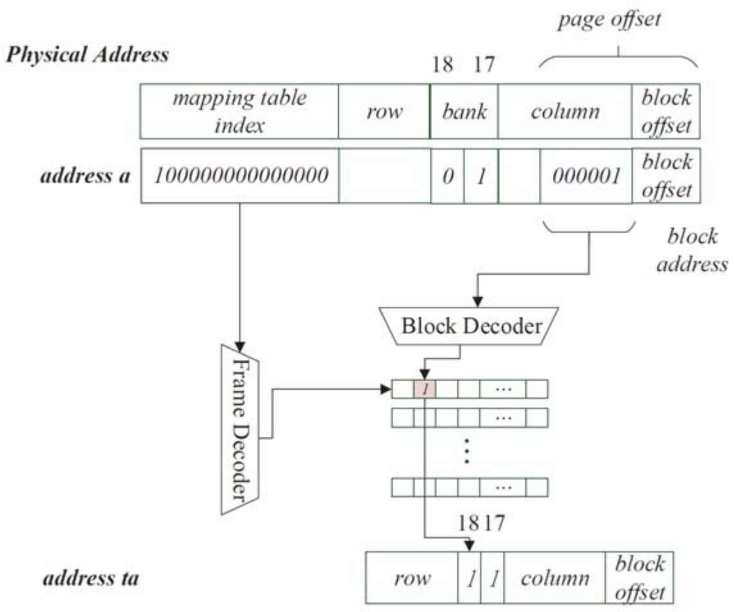


圖七 Update-and-reuse Pages Ratio in Data-parallel Multi-threaded Applications

由此發現我們提出了動態資料搬移的技術，主要分為更動作業系統與記憶體控制器的部分如圖八與圖九所示。



圖八 作業系統層級改動



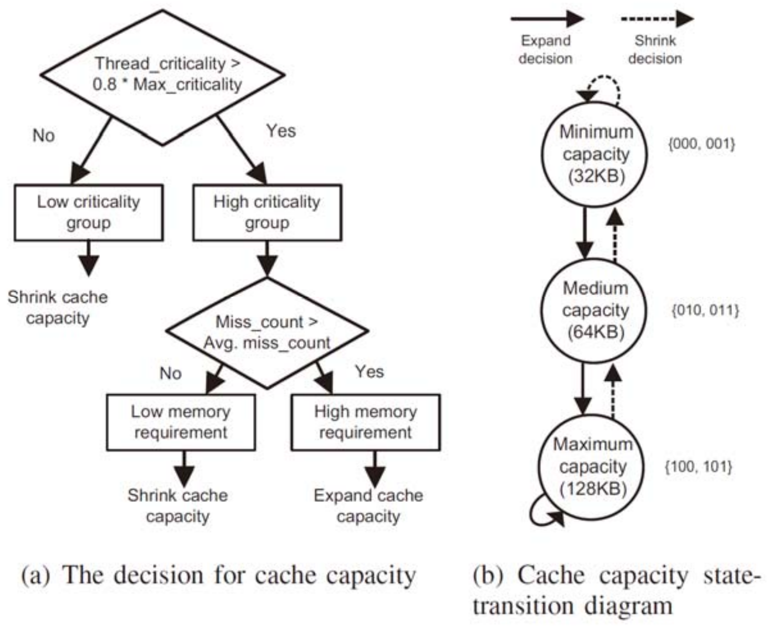
圖九 記憶體控制器改動

三、研究方法及進行步驟

對於三種不同的創新技術我們對其中對於快取記憶體、作業系統、以及記憶體控制器分別需要做哪些架構上的改變分別詳細介紹。

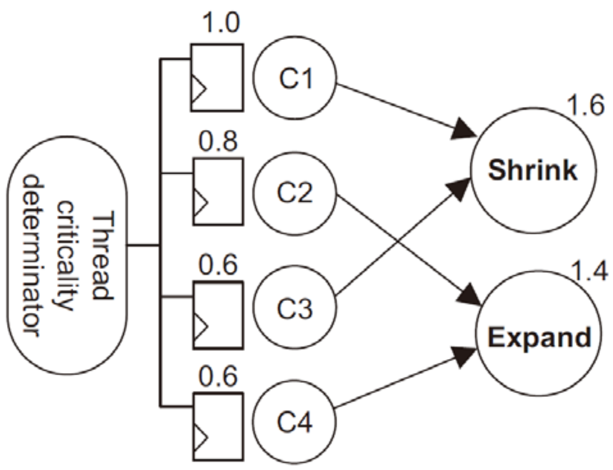
(a) 創新技術一 : 考量執行緒關鍵性的動態快取記憶體重整技術

主要分為 private L1 cache 和 shared L2 cache 的 capacity 調整技術。Private L1 cache 容量調整技術控制的流程圖如圖十所示。首先判斷 thread criticality 是否比最大的 thread criticality × 0.8 少，假如是的話就會被當成是 low criticality group 而被縮減 cache line size。若否，則在去比較 miss count 與 average miss count 的差異來進一步決定 cache line size。



圖十 Private L1 cache 動態快取記憶體重整控制依據

Shared L2 cache 的容量調整技術流程圖如圖十一所示。所有的 cores 都會算出對應的 criticality weight 與該 core 決定要 shrink 或是 expand 的決定，再利用投票表決的結果決定最後到底要增加或是減少容量。



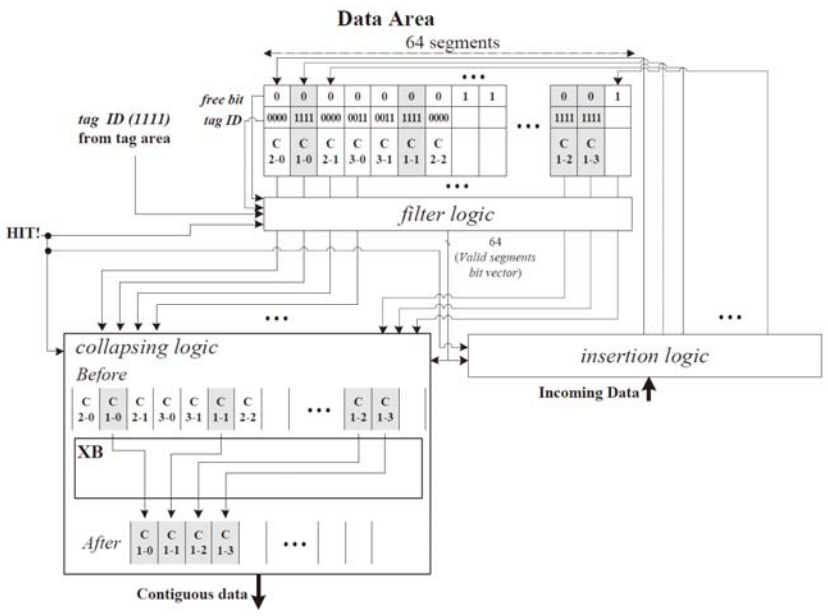
圖十一 Shared L2 cache 動態快取記憶體重整控制依據

(b) 創新技術二 : 免壓實之資料壓縮式快取記憶體機制

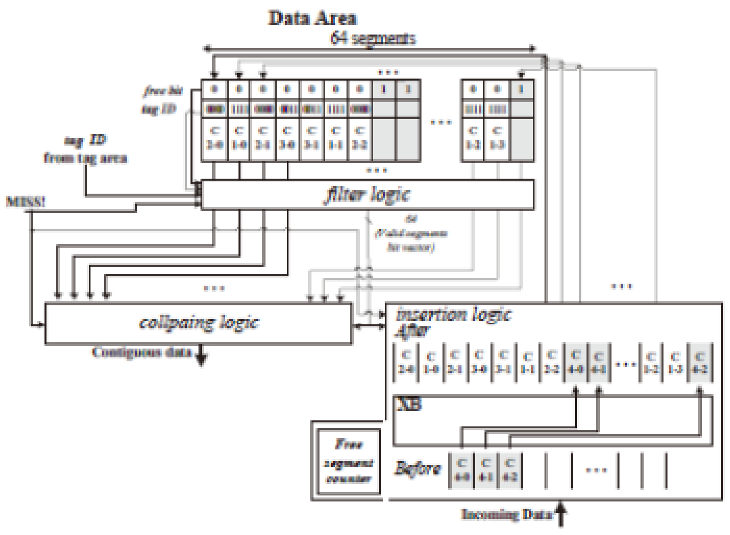
針對 compaction 所造成的 performance overhead，我們提出包含三種 circuits 的新架構來增加效能，分別為 filter logic、selection logic、跟 collapsing logic。

運作模式分為 cache hit (圖十二)與 cache miss (圖十三)的情況。在 cache hit 的情況下，tag ID 會先被比對並將資訊帶到 filter logic 中，filter logic 會將對應此 tag ID 的欄位都設為 1，再來 collapsing logic 會將這些欄位為 1 的資料搬移為連續資料輸出。Cache miss 的情況則是先由 filter logic 比對所有 free segment 的位置然後傳給 insertion logic，insertion logic 再將從記憶體搬來的資

料拆開到各 segment 之內。



圖十二 A cache hit example in our new cache design



圖十三 A cache miss example in our new cache design

(c) 創新技術三 ： 多執行緒下減少記憶體記憶庫衝突的資料管理機制

- 作業系統層級的改動

由於軟體端對於記憶體的使用只使用兩層 Time frame 的空間大小，所以整體記憶體使用量並不會很 大，我們修改作業系統使其在分配記憶體內存頁時，會把位於 free-page list 中除了決定記憶庫位址的 bits 之外的 bits 都一樣的內存頁都保留成保留內存頁(Reserved pages)，並將這些資訊傳遞給記憶體控制器做進一步的記憶庫衝突控制。

- 記憶體控制器的改動

記憶體控制器需要收到作業系統保留額外的內存頁資訊，並建立一個額外的硬體目錄以供查詢這些保 留內存頁，每一個硬體目錄單元紀錄著一份內存頁的實體記憶體位址，並指向第二層對應的硬體目錄，第二層硬體目錄中紀錄著在此內存頁中不同快取記憶體區塊的位置，由於作業系統已針對此內存頁額 外保留了各個記憶庫的內存頁，表示我們可以任意移動同一內存頁的不同快取記憶體區塊到其他保留 內存頁的相對位置。當最後一層快取記憶體要更新時，會將要寫回記憶體的快取記憶體區塊寫回記憶體。此時，根據其資 料來源的核心編號 (Core ID) ，我們可將資料寫到對應其編號的記憶庫保留內存頁面中，並在上述的硬 體目錄中紀錄此內存頁面實體位址以及第二層硬體目錄中紀錄搬移目的地的記憶庫編號。由於每個工 作單元存取的資料基本上是獨立的，此套架構會在程式執行時藉由更新資料搬移資料到對應核心編號 的記憶庫中，進而降低記憶庫衝突的發生率。

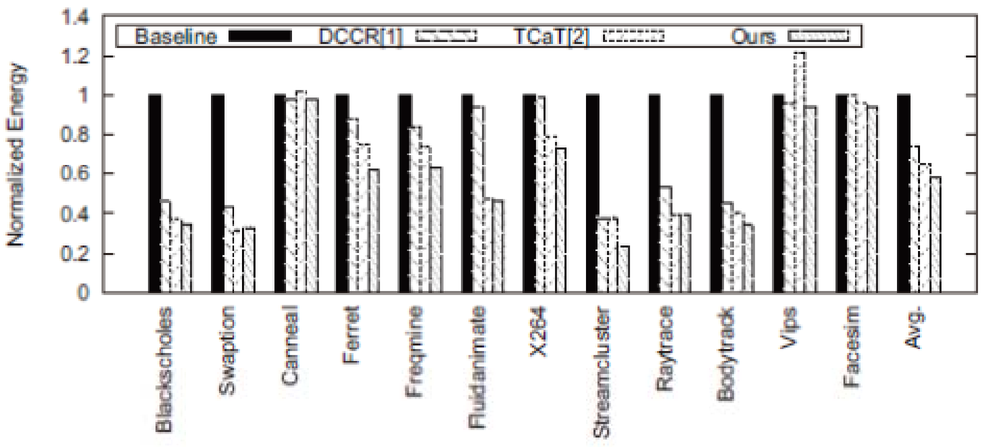
四、結果與討論

(1) Memory Hierarchy Design (第一、二、三年**)**：

研究結果分為三種不同創新技術的成果展示。

(a) 創新技術一 ： 考量執行緒關鍵性的動態快取記憶體重整技術

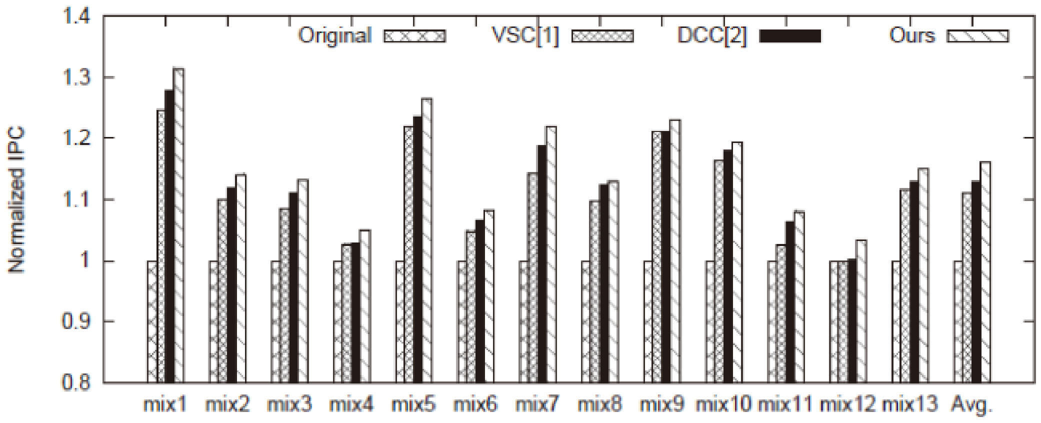
圖十四顯示我們的方法在耗能上比之前的方法更為有效減少耗能。(比 DCCR[3] 減少 16%耗能，比 TCaT[4]減少 8%耗能)



圖十四 整體快取記憶體耗能比較

(b) 創新技術二 ： 免壓實之資料壓縮式快取記憶體機制

圖十五為我們架構與之前架構的執行時間比較結果。跟 Baseline 相比有 16%效能上的提升，而與之前方法 VSC[6].與 DCC[7].相比各有 5%及 3%的提升。

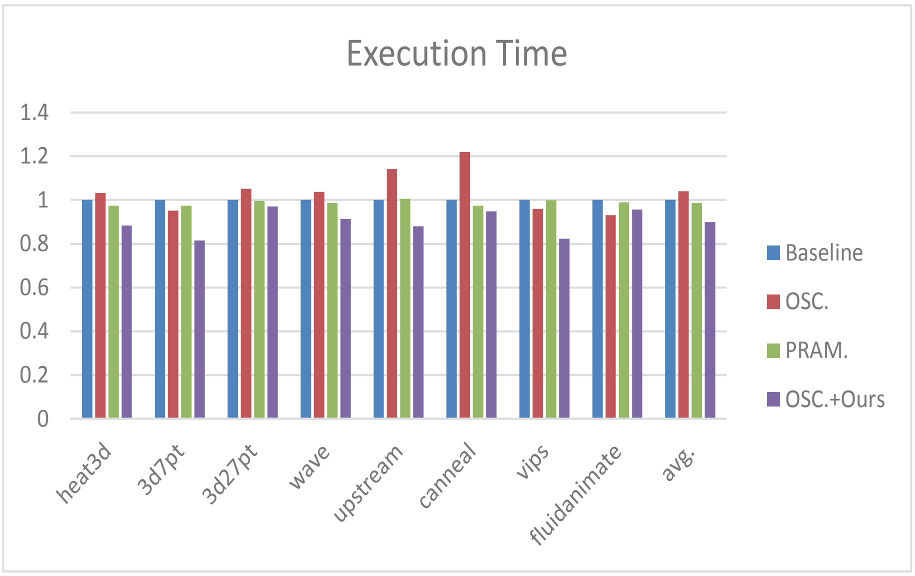


圖十五 Performance results by using different cache architectures

(c) 創新技術三 ： 多執行緒下減少記憶體記憶庫衝突的資料管理機制

我們實驗環境設定成十六核心、主記憶體有 八 個記憶庫的平台，透過我

們的軟硬體搭配的記憶庫衝突緩衝的方法來執行 Multi-threaded stencil applications。圖十五顯示我們利用 Pluto 自動產生 Heat 3D 程式的 Task dynamic scheduling (diamond-shape) 平行程式以及 PARSEC benchmarks 的實驗結果比較。其中 OSC.[1]為 OS page coloring 的方法，PRAM.[2]為 Multi-threaded memory controller scheduling 的方法。圖九可以看到我們提出的方法跟 OSC.相比有效提升了執行效率大約 13.2%，而與 PRAM.相比則提升了 9%



圖十五 各方法執行時間的比較

五、參考文獻

[1] L. Liu, Z. Cui, M. Xing, Y. Bao, M. Chen, C. Wu, “A Software Memory Partition Approach for Eliminating Bank-level Interference in MultiCore Systems,” in Proc. of the 21st international conference on Parallel architectures and compilation techniques, PACT’12, 2012, pp. 367-376.

[2] O. Mutlu, T. Moscibroda, “Parallelism-Aware Batch Scheduling : Enhancing both Performance and Fairness of Shared DRAM Systems,” in Proc. Of the 35th Annual International Symposium on Computer Architecture, ISCA08, 2012, pp. 63-74.

[3] M. Awasthi, D. W. Nellans, K. Sundan, R. Balasupramonian, A. Davis, “Handling the Problems and Opportunities Posed by Multiple Memory Controllers,” in Proc. of the 19st international conference on Parallel architectures and compilation techniques, PACT’10, 2010, pp. 319-330.

[4] Y.-T. Chen, J. Cong, H. Huang, B. Liu, C. Liu, M. Potkonjak, and G. Reinman, “Dynamically Reconfigurable Hybrid Cache: An Energy Efficient Last-level Cache Design,” in *Proc. Design, Automation, and Test in Europe, DATE’12*, 2012, pp. 45–50.

[5] A. Gordon-Ross, F. Vahid, and N. Dutt, “AutomAtic Tuning of Two Level Caches to Embedded Applications,” in *Proc. Design, Automation, and Test in Europe, DATE’04*, 2004, pp. 208 – 213.

[6] A. R. Alameldeen and D. A.Wood, “Adaptive Cache Compression for High-performance Processors,” in *Proc. the 31st Annual International Symposium on Computer Architecture (ISCA)*, 2004, pp. 212–223.

[7] S. Sardashti and D. A. Wood, “Decoupled Compressed Cache: Exploiting spatial Locality for Energy-optimized Compressed Cache,” in *Proc. The 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2013, pp. 62–73

科技部補助專題研究計畫出席國際學術會議心得報告

日期：___年___月___日

計畫編號	MOST　103－2220－E－007－013－		
計畫名稱	加寬匯流排多核心平台研究及其於三維圖形計算之應用-子計畫二：加寬匯流排考量下，多核心系統之記憶體層級架構設計 (3/3)		
出國人員姓名	黃婷婷	服務機構及職稱	國立清華大學資訊工程學系　教授
會議時間	103 年 05 月 29 日至 103 年 06 月 08 日	會議地點	美國舊金山
會議名稱	(中文)2014 DAC 會議 (英文) Design Automation Conference 2014		
發表題目	(中文) (英文) Time-Multiplexed Dual Role TSV for Fault Tolerance at Module Level		

一、參加會議經過

本次會議有兩個目的。第一個目的為在 DAC 會議的 work in progress session 中，報告研究成果。我們的論文主要是在探討 3D IC 中，在 module level，如何利用 test TSV 來擔任 fault tolerance TSV 角色。這個新的架構可以節省 TSV 的面積及 routing 的長度。第二個目的為參加劉炯朗校長得獎，在 DAC 的頒獎典禮。除此以外，亦見到許多相關領域教授，討論甚佳。

二、與會心得

DAC 會議是，EDA 領域最大的會議，全世界的學者都會參加。為了解 EDA 研究領域，提供未來研究方向的最重要會議。

三、發表論文全文或摘要

四、建議

五、攜回資料名稱及內容

六、其他

Time-Multiplexed Dual Role TSV for Fault Tolerance at Module Level

Abstract

In order to increase the yield of 3-D IC, fault-tolerance technique to recover failed TSV is essential. In this paper, an architecture of TSV recovery by using test elevator TSV is proposed. With the architecture, no dedicated redundant TSV is required to be inserted in advance. Hence, no extra area incurs. TSV assignment algorithm based on min-cost maximum-flow is proposed taking into consideration the locations of functional TSV as well as test TSV, so that the total Half-Perimeter Wire Length (HPWL) of a 3-D IC design is effectively reduced. Experimental results show that the total wirelength of 3-D IC testing is improved by 20% *in average compared to that of spare TSV approach.*

1 Introduction

Global interconnection has become a major bottleneck of performance and power for System on Chip (SoC) in single-chip integration. To tackle this problem, three-dimension Integrated Circuit (3-D IC) provides shorter interconnection by using Through-Silicon-Vias (TSVs) linking signals among multiple stacked dies [1].

TSV is a critical component in 3-D IC. Just like any other component, fabrication and bonding of TSV may fail [2,3]. A faulty TSV can be detrimental to the yield. To improve the yield, some recovery mechanism for faulty TSV is essential. To increase the yield of 3-D IC, a lot of methods to recover faulty TSV were proposed.

Among others, a simple but effective solution is to add redundant TSVs for faulty TSVs. The first shifting structure to shift a signal going through faulty TSV to redundant TSV was proposed by Hsieh et al. [10]. Then, following the same idea proposed by Hsieh, a built-in-self-repair (BISR) circuit for recovering faulty TSV in 3-D Random Access Memory using global redundant TSVs was proposed by Wu et al [7]. Xie et al. [?] introduce a yield-aware TSV defect searching and replacing strategy with built-in-self-test (BIST) design. Later on, Ye and Chakrabarty proposed spreading spare TSVs for faulty TSVs in considering physical locations [9]. Jiang et al. [?] propose a TSV repairing architecture to enhance the ability of recovering faulty TSV. Although these TSV recovery methods successfully recover faulty TSVs, they all need extra area to accommodate spare or redundant TSVs. Besides these recovery techniques using extra TSVs, TSV recovery without extra area was introduced by using sharing signal wires of pre-bond clock tree, reconfigurable routing hardware and fault-tolerant unit (TFU) [?,6,8]. However, these methods can be applied only to TSVs of clock tree and network-on-chip in 3-D IC.

On the other hand, 3-D IC testing has also been an active research area. Testing of SoC design in 3-D IC has its unique technical features that are different from testing in 2-D SoC design. Testing SoC design in 3-D includes pre-bond and post-bond testings where the former tests individual die before stacking and the latter tests the whole circuit after stacking. Lewis et al., proposed the first paper dedicated to testability of pre-bond 3-D IC [4]. Chi et al., presented an architecture and implementation for Design-for-Test (DfT) method for 2.5-D and 3-D pre- and post-bond testing [5, 16]. Marinissen et al., presented a 3-D DfT architecture for 3-D-SICs that allows pre-bond die testing and post-bond

Table 1: Comparison of the charaterisitcs of different TSV recovery techniques

Recovery Scheme	**Dedicated TSV**	**Design Type**
redundant TSV [?, 10]	yes	general
spare TSV [?, 9]	yes	general
redundant TSV [7]	yes	memory
clock wires for pre-bond test [?, 8]	no	clock tree
reconfigurable routing hardware [6]	no	Network-on-Chip
our *test* and redundant TSV	no	general

stacking testing.

In the above mentioned work related to testing 3-D IC, all DfT architectures are based on modular test approach. That is, its die-level and embedded core wrappers utilize IEEE STD 1500 [19]. In these test architectures, there are TSVs regarded as *test elevator TSVs* dedicated to transport test control and data signal up and down among dies during post-bond test. These *test TSVs* are used to transport test data **only** during post-pond testing and are **not used** during normal mode [11]. This important feature makes these *test elevator TSVs* redundant TSVs in normal mode. That is, by time-multiplexing, these TSVs have dual roles: as a *test elevator TSV* in scan mode and as a redundant TSV in normal mode.

In this paper, we propose a new architecture to recover faulty TSV by using *test elevator TSV*. Table 1 summarizes different TSV recovery techniques. In this table, we can see that the first three methods require extra area for redundant TSV while the last three methods do not need dedicated TSVs for recovering. However, the fourth can only be applied to clock signal and the fifth only to NoC. Our method is the only technique that uses **no dedicated TSV** and can be applied to **any TSV signal**.

The rest of this paper is organized as follows. In Sections 2 and 3, we introduce our architecture for TSV recovery and problem definition. Section 4 presents our algorithm for TSV recovery using *test elevator TSV*. Section 5 shows our experimental results. Finally, the conclusion of this paper is given in Section 6.

2 Preliminary

In this section, the architecture for TSV recovery and the fault model are described in Section 2.1. Then, in Section 2.2, TSV-block for TSV placement is presented.

Figure 1: Architecture of TSV recovery (a) normal mode and (b) recovery mode

Figure 2: 3-D IC design with TSV-blocks

2.1 Review on Architecture for TSV Recovery

The architecture of TSV recovery using redundant TSV is adopted [10] and shown in Figure 1(a). For each TSV, two configuration MUXs are added at two ends to shift the signal to neighboring TSV when one TSV fails. The TSVs are connected as a chain where the redundant TSV is placed at the end of the chain. When no TSV fails, all signals are transferred by original TSVs. When a TSV fails, the signal of the failed TSV needs to be shifted. This in term causes all signals between the failed TSV and the redundant TSV to be shifted. For example, let TSV_1 fail as shown in Figure 1(b). The selection inputs of configuration MUXs for TSV_2, TSV_3, and TSV_4 are set to 1. The signal paths after shifting are shown in Figure 1(b).

The recovery rate for a failed TSVs is analyzed based on probabilistic models. According to the result presented in Hiesh's work [10], assuming that single-fault model is used and the failure rate of a single TSV ranges between 10^{-4} to 10^{-5}, the recovery rate is 90% when the number of TSVs in a chain is no greater than 50, and 95% when the number of TSVs in a chain is no greater than 25.

We will adopt this architecture to recover failed TSV in this paper. In the following, the term TSV-chain will be used to refer to the structure of this redundant TSV architecture.

2.2 3-D IC Design with TSV-blocks

TSVs are not recommended to be placed arbitrarily on a plane. From the aspect of manufacturing, a regular placement of TSVs improves the exposure quality of the lithographic process and therefore improves the yield. Because of manufacturing and physical design issues, TSVs are suggested to form a regular placement in TSV-blocks. This 3-D IC design with TSV-blocks provides several beneficial properties [10, 17]. In real designs, TSV-blocks can be determined in floorplan stage. Inside each TSV block, TSVs can then be arranged in a grid-based structure to satisfy the pitch constraint. Moreover, TSV recovery architecture can be easily implemented by using shifted TSV-chain structure in 3-D IC design with TSV-blocks. Figure 2 shows fault-tolerant TSV-chains are constructed in 3-D IC with TSV-blocks where grey and black circles represent TSVs and redundant TSVs, respectively.

3 Test Architecture and Problem Definition

3.1 Test Architecture in 3-D IC

A conceptual overview of 3-D IC test architecture is shown in Figure 3. In this stacking integrated circuit, during 3-D post-bond scan mode, the test data of active devices on different dies are transported by using *test elevator TSVs*. In this figure, the grey arrows between dies represent *test elevator TSV* constructing test data paths in a 3-D IC DfT architecture where external I/Os at the bottom die are wrapped by IEEE 1149.1 Boundary Scan and intra-die is implemented based on IEEE 1500 core wrappers and TAMs (test access mechanisms). For 3-D IC test architecture with IEEE 1500 standard, each die has its own user-defined TAMs to support a parallel scan mode. Then, each core can have its own TAM-in/out ports consisting of a number of data and control lines for parallel test operations. Each core wrapper has a wrapper instruction register (WIR), which is used to store the instruction to be executed in its corresponding core. Furthermore, there are internal scan-chains for existing intra-die DfT in each die.

For SoC design, the IEEE 1500 standard provides a scalable architecture to test the embedded cores and interconnects between cores. The external test for testing core-to-core interconnects is prior to internal test in cores. In external testing, special instructions allow testing of core-to-core interconnects. After external testing, the test data can be loaded into WBRs (wrapper boundary registers) and then send to function input and internal scan-chain of corresponding core. Finally, functional test in core can be performed.

Similar to SoC design, IEEE 1500 standard can be applied to external and internal test at core-level in 3-D DfT architect (under discussion for standard). One difference is that in 3-D IC design, testing of TSV is usually performed before testing the whole design [7, 9, 12, 13, 20]. Before external and internal test of core-level design, whether a TSV is faulty is known.

3.2 TSV Redundancy Using Test Elevator TSV and MUX Configuration

Based on TSV-blocks design method with TSV redundancy and 3-D IC DfT architecture, our proposed architecture will use *test elevator TSV* as redundant TSV. A *test elevator TSV* acting as a redundant TSV in a TSV-chain is shown in Figure 4. In order to play two roles, the circuit of *test elevator TSV* (**redundant TSV**) is modified. A MUX, denoted as *DfT MUX*, is connected to source end and a

Figure 3: An overview of 3-D IC test architecture

Figure 4: Architecture of TSV recovery with *test elevator TSV* (a) normal mode without faulty TSV, (b) normal mode with faulty TSV, and (c) scan mode to transfer test data

de-MUX, denoted as *DfT de-MUX*, is connected to the destination end. During normal mode, the selection inputs of *DfT MUX* and *DfT de-MUX* are configured to 1 and the TSV is a redundant TSV, while during transferring test data of scan mode, the selection inputs of *DfT MUX* and *DfT de-MUX* are configured to 0 and the TSV is a *test elevator TSV*.

In normal mode, since *test elevator TSVs* do not transport test data, *test TSVs* can be regarded as redundant TSVs. These redundant TSVs can be used to recover failed TSV. If there is no faulty TSV in the TSV-chain, the data is transmitted as shown in bold line of Figure 4(a). If there is a faulty TSV in the TSV-chain, the last input (i.e., In_4 in Figure 4(b)) will be shifted passing through redundant TSV as shown in bold line of Figure 4(b).

In scan mode, when the TSV is used as a *test elevator TSV* to transfer test data, *DfT MUX* and *DfT de-MUX* are re-configured to 0 as shown in Figure 4(c).

As to test flow, Figure 5 shows our proposed method of testing in 3-D IC. Suppose there is no faulty TSV. In testing phase, test access is through external pins of the bottom die. Before transferring test data in die wrapper, the selection inputs of *DfT MUXs* and *DfT de-MUX* is configured to 0. Next, the test data is kept transmitted through *test elevator TSVs* in 3D-IC till all test data arrives their designated die. Then, test data is transferred to each core and run. Finally, test response is sent out for observation.

Suppose there is a faulty TSV on a TSV-chain, which can be found during TSV testing. Then, we need to re-configure the *DfT MUX* and *DfT de-MUX* first as shown in grey box. First, before running the test data, *DfT MUX* and *DfT de-MUX* are re-configured to 1 acting as a redundant TSV. Next, before sending the test response to bottom die, *DfT MUX* and *DfT de-MUX* are re-configured back to 0. Note that if no failed TSV is found, the second and the third configuring steps for *DfT MUX* and *DfT de-MUX*, i.e., grey boxes are removed in the flow.

Figure 5: Flow of testing in 3-D IC

3.3 Problem Formulation

Given a set of placed module blocks and a set of placed TSV blocks. Our object is to assign inter-die interconnections including function signals (functional TSVs) and test signals (*test elevator TSVs*) to TSV blocks so that

- the total wirelength of test nets are minimized

under the constraints that

- the total wirelength of functional nets remains the same as that without test nets, and
- there is sufficient *test elevator TSV* in each TSV block. This *test TSV* will also act as redundant TSV to recover the failed TSV in this TSV-block.

4 Algorithm

A floorplanning tool that allocates TSV-blocks to all inter-tier signals and places all blocks at the same time [18] is adopted in our design flow. This floorplanning tool outputs the locations of functional modules and TSV-blocks, and capacities of TSV-blocks. Given the output of the floorplanning tool, TSV assignment is to assign each TSV (functional TSV and *test elevator TSV*) to a TSV-block. In this section, first, that the number of redundant TSVs required in TSV-block is estimated and allocated in floorplanning stage is described in Section 4.1. Next, we define the cost function for an assignment of a TSV to a TSV-block in Section 4.2. Then, in Section 4.3, we model the assignment problem as a network flow optimization problem. Before we present the algorithm, for ease of explanation, a list of notations is defined in Table 2.

4.1 Estimation and Allocation of Required Redundant/Test TSVs

To guarantee that there are enough *test elevator TSVs* to ensure a given recovery rate for each TSV-block, the required number of redundant TSVs (i.e., *test elevator TSV*) required in a TSV-block, tb_j, needs to be calculated and allocated. Let N be the maximum number of functional TSVs allowed to be chained in a TSV-block, tb_j, for a given recovery rate.

$$\alpha(tb_j) = \frac{cap(tb_j)}{N} \tag{1}$$

Table 2: Notation List

e_i	the *i-th* net
te_k	the *k-th* net for 3-D IC testing
$tsv(e_i)$	the TSV used for e_i
$tsv(te_k)$	the TSV used for te_k
tb_j	the *j-th* TSV-block
$cap(tb_j)$	the capacity of tb_j

where $cap(tb_j)$, and N represent the number of TSVs that can be placed in a TSV-block, tb_j, and the maximum number of functional TSVs in the same TSV-chain mentioned in Section 2.1 and 2.2, respectively. Therefore, for each TSV-block has its number of required redundant TSVs, $\alpha(tb_j)$. Note that, if there are not enough *test elevator TSVs* to satisfy $\alpha(tb_j)$ redundant TSVs for a TSV-block, tb_j, the extra redundant TSVs are needed to be assigned in the TSV-block, tb_j, until the desired recovery rate are achieved.

4.2 Definition of Cost Function

In general, the priority of functional net is higher than that of *test net* because functional net affects circuit performance. However, a function TSV assignment may lead a test net to make a long detour if function TSV assignment does not consider the distribution of test nets. For example, suppose that a functional TSV can be assigned to two TSV-blocks, tb_a and tb_b where both have only one TSV capacity and a test net has only one tb_a inside its bounding box. In this case, if the functional TSV is assigned to tb_a, then the test net can not be assigned inside the bounding box of the net. Hence, a detour is required and the wirelength is increased. Therefore, assignment of functional TSV should take into consideration the distribution of test nets.

In order to take into consideration the distribution of test nets, we define a cost function for a functional TSV, $tsv(e_i)$, assigned to a TSV-block, tb_j, as

$$cost(tsv(e_i), tb_j) = \frac{\#tn}{cap(tb_j)} + \sum_{te_k \in TN} \frac{detour(te_k)}{detour_{max}} \tag{2}$$

where TN represents the set of test nets whose bounding box intersect the the center of the TSV-block, tb_j. $\#tn$ represents the size of TN, $cap(tb_j)$ represents the number of TSVs that can be placed in the TSV-block, tb_j. $detour(te_k)$ represents the shortest detour wirelength of test net, te_k, and $detour_{max}$ represents $max\{detour(te_k,),$ for all test nets$\}$.

Note that, for a functional TSV to be assigned, we consider only those TSV-blocks that are inside the bounding box of its net. As such in a TSV-block, the wirelength estimated by bounding box of the net will not change. Therefore, there is no term in the cost function defined in Equation (1) to reflect the wirelength of the functional net.

Next, we define the cost of assigning a *test elevator TSV*, $tsv(te_k)$, to a TSV-block tb_j

$$cost(tsv(te_k), tb_j) = HPWL(te_k, tb_j) \tag{3}$$

That is the half-perimeter wire length of the bounding box enclosing net te_k and TSV-block, tb_j.

We take Figure 6 (a) as an example to compute the cost function of assigning a functional TSV. In this figure, we are to compute the cost, $cost(tsv(e_1), tb_1)$, of assigning a functional TSV, $tsv(e_1)$, to a TSV-block, tb_1. Suppose that there are two test nets, te_1 and te_2, whose bounding boxes intersect the center of tb_1. The maximum detour wirelength equals 10 and the capacity of TSV-block tb_1 is 5. The shortest $detour(te_1)$ and $detour(te_2)$ are 3 and $2 + 6$, respectively. Then, The cost function is $cost(tsv(e_1), tb_1) = (\frac{2}{5}) + (\frac{3}{10} + \frac{8}{10}) = 1.5$.

4.3 Network-flow Optimization for TSV Assignment

Our assignment algorithm is performed tier by tier. At each tier, the assignment problem is modeled as a network flow optimization problem.

Now, we show how to model the problem. Assume we are to assign functional TSVs and *test elevator TSVs* at tier l. Let TSV_F and TSV_T denote the sets of TSVs for functional TSVs and *test elevator TSVs*, respectively, to be assigned, and TSV_{BK} represents the set of TSV-blocks at tier l.

The flow network to model the assignment problem is denoted by $G(V, E)$, where node set $V = \{(s \cup D_F), (D_T \cup C_F), (C_R \cup t)\}$ and edge set $E = \{(IE \cup FE), (TE \cup OE)\}$. s and t represent the source node and destination node of flow network. In this flow network, each node $n_{e_i} \in D_F$ corresponds to a functional TSV, $tsv(e_i)$, in TSV_F, each node $n_{te_k} \in D_T$ corresponds to a *test elevator TSV*, $tsv(te_k)$, in TSV_T, each node $n_{tb_j} \in C_F$ corresponds to a TSV-block, tb_j, in TSV_{BK}, and each node $n_{dtb_j} \in C_R$ corresponds to a duplication of TSV-block, tb_j. This duplication is to ensure that in each TSV-block, there are enough *test elevator TSVs* to act as redundant TSVs to recover failed TSVs. IE, FE, TE, and OE represent *incoming edge*, *functional assignment edge*, *test assignment edge*, and *outgoing edge*, respectively. *Incoming edge* is constructed from source to all nodes in D_F and D_T, and *outgoing edges* are constructed from all nodes in C_F and C_R to destination node, t.

There are two types of *assignment edges*, *functional assigned edges* FE and *test assigned edges* TE.

An *assignment edge*, $edge_{tsv(e_i),tb_j}$, in FE connecting a node n_{e_i} (i.e., $tsv(e_i)$) in D_F and a node n_{tb_j} (i.e., tb_j) in C_F represents a functional TSV, $tsv(e_i)$, is assigned to TSV-block, tb_j. Assignment edges in TE are further classified to two types. The first type is an *assignment edge*, $edge_{tsv(te_k),tb_j}$, connecting a node, n_{te_k} (i.e., $tsv(te_k)$), in D_T and a node, n_{tb_j} (i.e., tb_j), in C_F that represents a *test elevator TSV*, $tsv(te_k)$, is assigned to TSV-block, tb_j. The second type is an *assignment edge*, $edge_{tsv(te_k),dtb_j}$, connecting a node, n_{te_k} (i.e., $tsv(te_k)$), in D_T to a node, n_{dtb_j} (i.e., a duplication of TSV-block tb_j), that ensures there is an assignment of *test elevator TSVs* to TSV-block for recovery of failed TSV.

For a functional TSV of a net e_i, we construct edges only between the functional TSV and those TSV-blocks that are inside the bounding box of net e_i. For a *test elevator TSV* of a net te_k, we construct edges between the *test TSV* and all TSV-blocks and their duplications.

The capacity of *incoming edge* and *assignment edge* is set to 1. The capacity of an edge connecting a node n_{tb_j} (i.e., TSV-block, tb_j) and destination node t is set to $cap(tb_j) - \alpha_{tb_j}$. The capacity of an edge connecting the duplication node, n_{dtb_j}, and destination, t, is set to α_{tb_j}. This capacity will ensure that there are α_{tb_j} *test elevator TSVs* assigned to TSV-block, tb_j, for fault recovery.

As to the cost of an edge, for *incoming edges* and *outgoing edges*, they are set to 0. For *assignment edges*, the cost are assigned as we presented in Section 4.2. After we model the assignment problem as the network flow, we find the minimum cost maximum flow of the network.

Take Figure 6 as an example to demonstrate the modeling. Figure 6 (a) shows that there are two TSV-blocks, tb_1 and tb_2, two functional nets, e_1 and e_2, two test nets, te_1 and te_2. Four TSVs, $tsv(e_1)$ of e_1, $tsv(e_2)$ of e_2, $tsv(te_1)$ of te_1, and $tsv(te_2)$ of te_2 are to be assigned to TSV-blocks.

There are two TSV-blocks, tb_1 and tb_2, in the bounding box of net e_1. Hence, there are one edge from n_{e_1} to n_{tb_1} and one edge from n_{e_1} to n_{tb_2}. Similarly, since tb_2 is the only TSV-block in the bounding box of e_2, there is only one edge from n_{e_2} to n_{tb_2}. Since *test elevator TSV*, $tsv(te_1)$ ($tsv(te_2)$), can be assigned to any TSV-blocks, there are edges connecting n_{te_1} (n_{te_2}) to n_{tb_1}, n_{dtb_1}, n_{tb_2}, and n_{dtb_2}. Let the capacity of TSV-block, tb_1, be 5, and one redundant TSV is required to recovery faulty TSV in tb_1. Then, we have the capacity of $edge_{tb_1,t}$ being 4 and the capacity of $edge_{dtb_1,t}$ being 1. Since the cost of assigning $tsv(e_1)$ to tb_1 is $cost(tsv(e_1), tb_1) = 1.5$ shown in Section 4.2, we have cost $= 1.5$ and capacity $= 1$ on $edge_{n_{e_1},n_{tb_1}}$.

Figure 6: An example of flow network

Table 3: The ratio of Net degree of MCNC floorplan benchmark

degree	2	3	4	5
ratio	0.75	0.16	0.06	0.03

5 Experimental Results

Our experiment is developed on 3.0 GHz Linux environment with 16 GB memory. We implemented the proposed algorithm using C++/STL programming language and performed experiments on all ITC'02 SoC test benchmark [24] except $d281$, $h953$, $f2126$, $q12710$, and $a586710$. The reason to exclude these circuits is because they are small circuits with less than 15 blocks. Next, since the SoC test benchmark only has the testing information such as the number of scan-chain, the number of input and output, and the number of test pattern for developing SoC test scheduling, and has no physical information such as the width and height of blocks, connecting relations among blocks, and pin location in block, we need to generate the required physical information for 3-D IC floorplan. In our experiment, pitch of TSV-to-TSV is set to $10\mu m$ [21] including $5\mu m$ TSV diameter and $2.5\mu m$ keep-out-zone at both sides.

For generating connecting relations, we first profile the net degree data from MCNC floorplan benchmark [25] and compute the ratios of the number of nets with the same degree to the total number of nets. The ratios of net degree are reported in Table 3. Based on the ratios of net degree in this table, we establish connecting relations among blocks. Next, similar to a previous work [23], we determine the width and height of block according to its numbers of inputs and outputs. Finally, the pins are evenly placed at the boundary of its own block.

Next, given the test bandwidth, we use testing scheduling methods for 3-D IC testing proposed by Wu et al., [22] to determine communications among blocks.

In our experiment, we first compare two architectures for TSV recovery: *test elevator TSV* and spare TSV. Note that, a spare TSV is an extra TSV and *test elevator TSVs* can be regarded as a free redundant TSV. In order to achieve 90% recovery rate of TSV, **one redundant TSV per** 50 **functional TSVs is set** [10]. Table 4 shows the numbers of spare TSVs and *test elevator TSVs* for TSV recovery with different number of tiers, where $\#TSV_{func}$, $\#TSV_{spare}$, and $\#TSV_{test}$ represent the numbers of functional TSVs, spare TSVs and *test elevator TSVs*, respectively. W_{32}, W_{48}, and W_{64} are

Figure 7: The comparison for 3-D IC testing wirelengths by ours methodology and *greedy_assignment*

Table 4: The numbers of test elevator TSVs and spare TSVs

circuit name	2-tiers							4-tiers						
	$\#TSV_{func}$	$\#TSV_{spare}$			$\#TSV_{test}$			$\#TSV_{func}$	$\#TSV_{spare}$			$\#TSV_{test}$		
		W_{32}	W_{48}	W_{64}	W_{32}	W_{48}	W_{64}		W_{32}	W_{48}	W_{64}	W_{32}	W_{48}	W_{64}
g1023	1698	58	63	66	64	72	88	2412	82	83	96	102	104	116
p22810	1618	46	52	57	58	62	68	2287	71	75	93	84	108	108
p34392	912	31	35	38	44	54	54	1283	65	66	71	76	84	96
p93791	2724	60	60	62	62	66	72	3858	84	93	115	96	112	122
t512505	3028	62	67	74	64	78	84	4307	92	105	142	98	134	164
ratio		1.00	1.00	1.00	1.17	1.23	1.25		1.00	1.00	1.00	1.16	1.29	1.19

test bandwidths of 32, 48 and 64, respectively. Take the first row labeled *g1023* as an example. Entry "58" under column labeled $\#TSV_{spare}$ and column labeled W_{32} means to achieve 90% recovery rate (at least one redundant TSV per 50 functional TSVs), 58 spare TSVs are allocated for test bandwidth of 32 bits. Entry "64" under column labeled $\#TSV_{test}$ and column labeled W_{32} means that there are 64 free *test elevator TSVs* are available for fault recovery when test bandwidth is 32 bits. From this table, we can see that the number of *test TSVs* is enough to achieve 90% recovery rate of TSV.

Table 5 shows the testing wirelength in two and four tiers under different test bandwidths, where *#block*, *#nets*, WL_{spare} and WL_{ours} represent the number of blocks, the number of nets, the wirelength using spare TSV and the wirelength using *test elevator TSV*, respectively. The testing wirelength means the total wirelength for connecting all core wrappers in 3-D IC DfT architecture. From this table, the wirelength of ours method as compared to spare TSV-based method can be reduced by 20% to 24% in average.

In order to place spare TSV, extra area is needed. Table 6 shows the total area comparison, where $area_{spare}$ and $area_{ours}$ represent the chip area using spare TSV and *test elevator TSV*, respectively. It shows that using *test elevator TSV* as redundant TSV requires 3.4% and 4.1% in 2 and 4 tiers, respectively, less area in average when compared to spare TSV method.

The next experiment is to compare the wirelength for 3-D IC testing by our method and *greedy_assignment* method. *Greedy_assignment* method first assigns functional TSVs in TSV-blocks without detour, and then assigns a *test elevator TSV* one by one to a TSV-block that has the shortest distance from the test net to the TSV-block with free capacity. The wirelength of our method is 21.8% better than that of *greedy_assignment* as shown in Figure 7.

Table 5: Comparison for the wirelength of 3-D IC testing of spare TSV and test elevator TSV

circuit name	$\#blocks$	$\#nets$	2-tiers						4-tiers				
			WL_{spare} (μm)			WL_{ours} (μm)			WL_{spare} (μm)			WL_{ours} (μ	
			W_{32}	W_{48}	W_{64}	W_{32}	W_{48}	W_{64}	W_{32}	W_{48}	W_{64}	W_{32}	W_{48}
g1023	15	1603	25647	37458	51945	17456	25414	38542	21784	32745	42515	14784	22154
p22810	29	1518	42176	63215	84751	36541	52145	71548	34584	51488	68587	27545	43548
p34392	20	852	38451	58421	76541	27458	46218	62145	31530	48957	62985	24521	42158
p93791	33	2577	53125	78542	105478	42154	63258	80641	43563	66215	86258	38547	56487
t512505	32	2865	49851	72454	98745	43215	59214	83147	40878	62147	83541	31548	52147
ratio						0.76	0.78	0.82				0.78	0.82

Table 6: Comparison for the area of 3-D IC of spare TSV and test elevator TSV

circuit name	2-tiers						4-tiers					
	$area_{spare}$ (μm^2)			$area_{test}$ (μm^2)			$area_{spare}$ (μm^2)			$area_{test}$ (μm^2)		
	W_{32}	W_{48}	W_{64}	W_{32}	W_{48}	W_{64}	W_{32}	W_{48}	W_{64}	W_{32}	W_{48}	W_{64}
g1023	544644	550564	570694	529984	535824	559504	199809	196755	205450	190794	192897	201421
p22810	293764	306916	321489	274576	286225	297025	103684	109561	114244	98847	103041	106929
p34392	425104	444889	463362	412164	427716	454276	156025	158404	169744	148379	153978	163539
p93791	574564	589824	593812	559504	565504	582169	218089	212521	219024	201421	203581	209581
t512505	3239034	3283344	3449559	3175524	3319684	3381921	1166052	1218988	1241841	1143189	1195086	1217492
ratio	1.04	1.03	1.03	1.00	1.00	1.00	1.05	1.04	1.04	1.00	1.00	1.00

6 Conclusions

In this paper, we proposed an architecture of TSV recovery by using *test elevator TSV*. By our architecture, no extra area incurs. The wirelength for 3-D IC testing based on *test elevator TSV* as compared with that of spare TSV-based is improved by 20% to 24% in average. Furthermore, the wirelength of 3-D IC testing of our algorithm as compared to *greedy_assignment* is significantly better by 21.8%.

References

[1] C. S. Tan, Ronald J. Gutmann, and L. Rafael Reif, “Wafer Level 3-D ICs Process Technology,” *Springer* , 2008

[2] P. R. Morrow , M. J. Kobrinsky , S. Ramanathan , C.-M. Park , M. Harmes , V. Ramachandrarao , H.-M. Park , G. Kloster , S. List and S. Kim, “Wafer-level 3D Interconnects via Cu Bonding,” *Proc. AMC*, pp. 125-130, 2004

[3] P. Garrou, C. Bower, and P. Ramm, “Handbook of 3D Integration: Technology and Application of 3D Integrated Circuits,” Weinheim: XWILEY-VCH Verlag GmbH & Co. KGaA, vol. 1-2, 2008

[4] Dean L. Lewis and Hsien-Hsin S. Lee, “A Scan-Island Based Design Enabling Prebond Testability in Die Stacked Microprocessors,” *in Proceedings of IEEE International Test Conference (ITC)*, pp.1-8 Oct. 2007

[5] Chun-Chuan Chi, Marinissen, E.J., Goel, S.K., Cheng-Wen Wu, “Post-bond testing of 2.5D-SICs and 3D-SICs containing a passive silicon interposer base,” *in Proceedings of IEEE International Test Conference (ITC)*, pp.1-10, 20-22 Sept. 2011

[6] Igor Loi, Federico Angiolini, Shinobu Fujita, Subhasish Mitra, and Luca Benini, “Characterization and Implementation of Fault-Tolerant Vertical Links for 3-D Networks-on-Chip,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 30, No.1, Jan. 2011

[7] Cheng-Wen Wu, Shyue-Kun Lu and Jin-Fu Li, “On Test and Repair of 3D Random Access Memory,” *in Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp.744-749, 2012

[8] Heechun Park and Taewhan Kim, “Comprehensive technique for designing and synthesizing TSV Fault-tolerant 3D clock trees,” *International Conference on Computer-Aided Design (ICCAD)*, pp. 691-696, 2013

[9] Chiao-Ling Lung, Yu-Shih Su, Shih-Hsiu Huang, Yiyu Shi and Shih-Chieh Chang, “Through-Silicon Via Fault-Tolerant Clock Networks for 3-D ICs,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 7, pp. 1100-1109, 2013

[10] Fangming Ye and Krishnendu Chakrabarty, “TSV Open Defects in 3D Integrated Circuits: Characterization, Test, and Optimal Spare Allocation,” *in Proceedings of Design Automation Conference*, pp.1024-1030, 2012

[11] Ang-Chih Hsieh, TingTing Hwang, “TSV Redundancy: Architecture and Design Issues in 3-D IC,” *IEEE Transactions on Very Large Scale Integration Systems*, vol.20, no.4, pp.711-722, April 2012

[12] Jing Xie, Yu Wang, and Yuan Xie, “Yield-aware Time-efficient Testing and Self-fixing Design for TSV-based 3D ICs,” *in Proceedings of Asia and South Pacific Design Automation Conference*, pp. 738-743, Feb 2012

[13] Li Jiang, Qiang Xu, and Bill Eklow, “On Effective Through-Silicon Via Repair for 3-D-Stacked ICs,” *IEEE Transactions on Computer-Aided Design of Integration Circuits and Systems*, vol. 32, no.4, April 2013

[14] Fu-Wei Chen, Hui-Ling Ting,and TingTing Hwang, “Fault-tolerant TSV by Using Scan-chain TSV,” to appear in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2014

[15] D. L. Lewis and H. S. Lee, A Scanisland Based Design Enabling Pre-bond Testability in Die-stacked Microprocessors, *in Proceedings of IEEE International Test Conference (ITC)*, pp. 1-8, 2007

[16] Po-Yuan Chen, Cheng-Wen Wu and Ding-Ming Kwai, On-chip Testing of Blind and Open-sleeve TSVs for 3D IC before Bonding, in Proceedings of VLSI Test Symposium (VTS), pp.263-268, 2010

[17] Minki Cho, Chang Liu, Dae Hyun Kim, Sung Kyu Lim and Mukhopadhyay, S., “Pre-Bond and Post-Bond Test and Signal Recovery Structure to Characterize and Repair TSV Defect Induced Signal Degradation in 3-D System,” *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol.1, no.11, pp.1718-1712, Nov 2011

[18] Marinissen, E. J., Chun-Chuan Chi, Jouke Verbree, and Mario Konijnenburg, "3D DfT Architecture for Pre-bond and Post-bond Testing," *in Proceedings of IEEE International 3D Systems Integration Conference (3DIC)*, pp.1-8, Nov. 2010

[19] Chun-Chuan Chi, Marinissen, E.J., Goel, S.K., Cheng-Wen Wu, "DfT Architecture for 3D-SICs with Multiple Towers," *in Proceedings of European Test Symposium (ETS)*, pp.51-56, May 2011

[20] Johann Knechtel, Igor L. Markov, and Jens Lienig, "Assembling 2D Blocks into 3D Chips", *In Proceedings of ACM international Symposium on Physical Design (ISPD)*, ,pp 81-88, 2011

[21] Ming-Chao Tsai, Ting-Chi Wang, TingTing Hwang, "Through-Silicon Via Planning in 3-D Floorplanning, "*IEEE Transactions on Very Large Scale Integration Systems*", vol.19, no.8, pp.1448-1457, Aug. 2011

[22] Marinissen, E.J., Verbreee, J., and Konijnenburg, M. "A Structured and Scalable Test Access Architecture for TSV-based 3D Stacked ICs," *in Proceedings of VLSI Test Symposium (VTS)*, pp. 269-274, 2010

[23] B. Noia, S. Panth, K. Chakrabarty, and S. K. Lim, "Scan Test of Die Logic in 3D ICs using TSV Probing," *in Proceedings of IEEE International Test Conference (ITC)*, pp. 1-8, 2012

[24] Moongon Jung, Mitra J., Pan, D.Z. and Sung-Kyu Lim, "TSV Stress-aware Full-chip Mechanical Reliability Analysis and Optimization for 3-D IC," *IEEE Transactions on Computer-Aided Design of Intergrated Circuits and Systems*, vol.31, no.8, pp.1194-1207, 2012

[25] Xiaoxia Wu, Yibo Chen, Krishnendu Chakrabarty and Yuan Xie "Test-Access Mechanism Optimization for Core-Based Tree-Dimensional SoCs," *in Proceedings of International Conference on Computer Design*, pp. 212-218, 2008

[26] Li Jiang, Lin Huang and Qiang Xu, "Test Architecture Design and Optimization for Three-Dimensional SoCs," *in Proceedings of ACM/IEEE Design, Automation and Test in Europe (DATE)*, pp. 220-225, 2009

[27] ITC'02 SoC test benchmark
http://itc02socbenchm.pratt.duke.edu/

[28] MCNC floorplan benchmark
http://vlsicad.eecs.umich.edu/BK/BlockPacking/progress.html

科技部補助計畫衍生研發成果推廣資料表

日期:2015/07/31

科技部補助計畫	計畫名稱: 子計畫二：加寬匯流排考量下，多核心系統之記憶體層級架構設計(3/3)
	計畫主持人: 黃婷婷
	計畫編號: 103-2220-E-007-013- 學門領域: 智慧電子科技計畫-整合型學術研究計畫

無研發成果推廣資料

103年度專題研究計畫研究成果彙整表

計畫主持人：黃婷婷	計畫編號：103-2220-E-007-013-
計畫名稱：子計畫二：加寬匯流排考量下，多核心系統之記憶體層級架構設計(3/3)	

成果項目			量化			單位	備註（質化說明：如數個計畫共同成果、成果列為該期刊之封面故事...等）
			實際已達成數（被接受或已發表）	預期總達成數(含實際已達成數)	本計畫實際貢獻百分比		
國內	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	3	3	100%		
		專書	0	0	100%		
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（本國籍）	碩士生	4	4	100%	人次	
		博士生	2	2	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		
國外	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	1	1	100%		
		專書	0	0	100%	章/本	
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（外國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		

其他成果 (無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。)	無

	成果項目	量化	名稱或內容性質簡述
科教處計畫加填項目	測驗工具(含質性與量性)	0	
	課程/模組	0	
	電腦及網路系統或工具	0	
	教材	0	
	舉辦之活動/競賽	0	
	研討會/工作坊	0	
	電子報、網站	0	
	計畫成果推廣之參與（閱聽）人數	0	

科技部補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

   ■達成目標

   □未達成目標（請說明，以 100 字為限）

   　　□實驗失敗

   　　□因故實驗中斷

   　　□其他原因

   說明：

2. 研究成果在學術期刊發表或申請專利等情形：

   論文：■已發表 □未發表之文稿 □撰寫中 □無

   專利：□已獲得 □申請中 ■無

   技轉：□已技轉 □洽談中 ■無

   其他：（以 100 字為限）

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）（以 500 字為限）

   我們在這個計畫的第一年提出考量執行緒關鍵性的動態快取記憶體重整機制，該篇論文在 2013 年 11 月刊登在 ICCAD 上，第二年提出免壓實之資料壓縮式快取記憶體機制，並在 2014 年 11 月刊登在 ICCAD 上，第三年提出多執行緒下減少記憶體記憶庫衝突的資料管理機制，並刊登在 2015 年 6 月的 DAC WIP 上。

   DAC 和 ICCAD 是電子設計自動化領域頂尖國際研討會之一。研討會宗旨致力於電子設計自動化的技術突破與創新。為了維持會議的品質，每一篇被接受的論文，都需經過嚴謹的審查程序，ICCAD 2013、2014 年會議論文接受率為 26%和 25%，2014DAC 2015 年會議論文接受率僅約 22%。 WIP 接受率為 19%。