以電子系統級為基礎所建構的雙核心系統晶片
Building a Dual-Core SOC Based on Electronic System Level

周志鴻
聖約翰科技大學電子工程系研究所
96n04010@student.sju.edu.tw

杜日富
聖約翰科技大學電子工程系研究所
tu@mail.sju.edu.tw

摘要

在本研究中，我們提出以電子系統級設計工具來建構一同質性雙核心(homogeneous dual core)系統晶片，且我們將傳統式雙核心架構內的共享式記憶體(Shared Memory)重新設計，以便做為雙核心之間在資料傳遞與訊息聯繫的高效能裝置。另外，本研究使用SystemC 語言來建構此改良型的共享式記憶體模型。最後，我們使用電子系統級設計工具整合出系統晶片的整體架構並利用模擬平台來模擬雙核心系統晶片的運作效能。

關鍵詞：雙核心、系統晶片、共享式記憶體、電子系統級、SystemC

Abstract

In this paper, We proposed electronic system level design tool to build a homogeneous dual core system on chip, and we had re-designed the shared memory of tradition dual core architecture, and the shared memory was a high performance device which would transmission data and contact message between dual cores. Besides, the paper would use SystemC language to build the shared memory model. Finally, we used electronic system level design tool to integrate the integral architecture of system on chip and we used the simulation bench to simulate the operation performance of the dual core system on chip.

Keywords: Dual Core, System On Chip, Shared Memory, Electronic System Level, SystemC.

1. 前言

隨著微處理器系統的演進，單核心的系統晶片所能增進的運算效率似乎已經達到了一個瓶頸，且由於VLSI 製程技術的發展快速，因此系統晶片在設計上已逐漸傾向多核心[1-2、5、8-10]的方向邁進，而對稱多處理器(Chip Multiprocessors，CMP）是多核心系統目前較為明顯的實現目標，但多核心在核心與核心之間的溝通上，各家晶片設計公司皆有各自不同的想法與各自實現的技術，對此，本研究的主要研究目標乃是提出以電子系統級設計工具來建構一同質性雙核心(homogeneous dual core)系統晶片(system on chip)，並將傳統型雙核心架構內的共享式記憶體(Shared Memory)重新設計，以便改善雙核心之間的通訊問題，此改良型共享式記憶體裝置可以讓雙核心透過有效的記憶體管理方式來快速的達成雙核心之間的通訊目的，以便增進整個系統的運作效能，且本研究也利用SystemC 語言來建構此裝置的行為模型。最後，我們使用電子系統級設計工具整合出系統晶片的整體架構並使用模擬平台搭配ARMulator 及MediaBench suite來進行雙核心系統晶片的運作與效能之評估模擬。本研究的貢獻如下所示：

(a) 利用現今產業界較為流行的電子系統級設計工具 – RealView SoC Designer 及ARMulator 來建構與模擬雙核心系統晶片。
(b) 將傳統型雙核心系統內的共享式記憶體重新設計且使用SystemC 語言來建構此共享式記憶體的電子系統級模型以作為雙核心之間在互聯通訊上的高效能裝置，進而降低並改善雙核心在核心互聯通訊上所造成的延遲時間以便提高雙核心系統晶片的運作效能。

2. 相關研究與背景知識

在本節，將針對雙核心系統晶片在架構設計與所需使用到的模擬工具作一個概括性的介紹。在2.1 小節將對傳統型雙核心系統的硬體架構作一個簡要地介紹並提出本研究的一

些改良方針；在2.2 小節則對本研究所使用到的電子系統級設計工具作一個概略性地介紹。

2.1 傳統型雙核心系統的硬體架構介紹

文獻[7]使用了市面上一套較為古老的電子系統級工具 – SimpleScalar [22]來架構出如圖1 的傳統式雙核心系統。

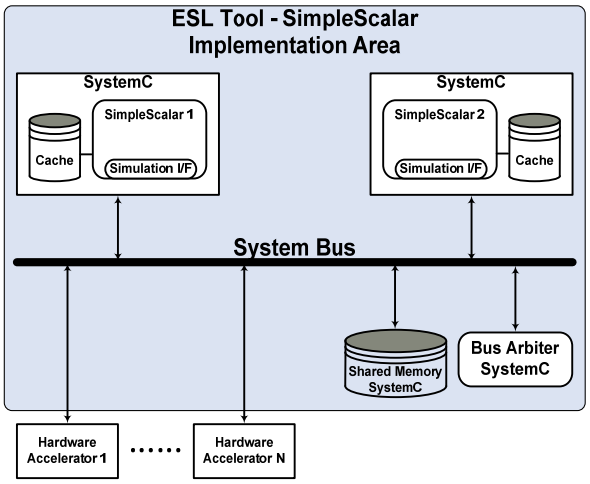

圖1 傳統型的雙核心系統之硬體架構

由圖1 的硬體架構顯示此共享式記憶體(Shared Memory)主要是用來做為核心與核心之間在資料傳遞與訊息聯繫上的暫存裝置，但架構中的共享式記憶體在雙核心互相進行通訊時卻一定得透過系統匯流排來進行聯繫，這樣的通訊機制往往會因系統匯流排[3]本身的運作速度而花費太多的機械週期(大約10個機械週期)，進而使整個系統的運作效能大幅度地降低，因此，本研究將此種架構重新改良與設計。首先，我們將共享式記憶體搬離系統匯流排，接著將此共享式記憶體加以改良後，並使用SystemC 語言來建構此改良型共享式記憶體的行為模型，並將它整合在雙核心之間，使系統晶片在雙核心之間的互聯通訊上，可利用此改良型共享式記憶體來提升雙核心系統的運作效能，對於改良型共享式記憶體的設計與運作原理，我們會在第3 節作一個詳細的說明。

2.2 電子系統級設計工具的介紹

在2.2.1 小節對電子系統級作了一個簡要的介紹；在2.2.2 小節對SystemC 語言作了一個概略性地介紹；在2.2.3 小節對ARM 公司所發展的RealView SoC Designer 作一個簡要地介紹；在2.2.4 小節對ARM 公司所發展的ARMulator 作一個簡要地介紹；在2.2.5 小節對本研究所使用的效能評估程式作一個概略性地介紹。

2.2.1 電子系統級的介紹

電子系統級（Electronic System Level, ESL）[4、6]是以抽象方式來描述系統設計，由於設計的速度快，因此能有充裕的時間分析設計內容，且功能足以提供虛擬原型，因此ESL 逐漸成為設計流程的基礎步驟，因為它不僅能應用在設計初期與系統架構規劃階段，也可以支援整個硬體與軟體互動設計的流程。就系統而言，ESL 功能其中的一項重要設計流程優勢就是提供一個初期設計平台，能用來建置元件軟體。也可以讓硬體和軟體協同驗證的流程更為容易。在這個層級的設計抽象規劃方面，最重要的關鍵就是模擬。利用模型來模擬系統功能，其速度比暫存器轉移層級(Register Transaction Level，RTL)快速許多，而模擬時序的精準度也有一定的準確程度。因此本論文所建構之雙核心系統晶片便是基於ESL 層級來進行效能評估，取得各參數的模擬數據，進而將雙核心系統晶片的設計達到最佳化的處理。

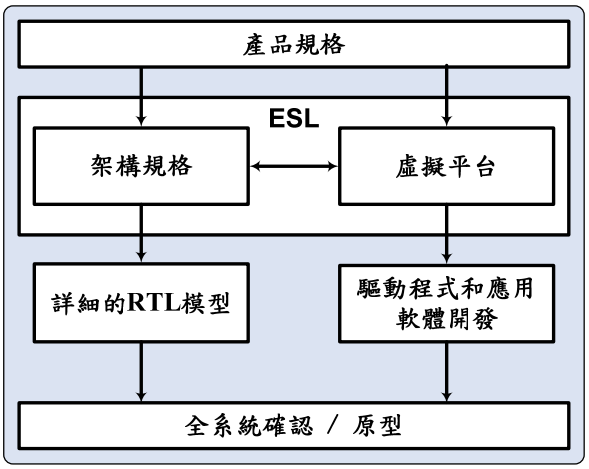

圖 2 ESL 的整體設計流程

圖 2 展示了 ESL 的整體設計流程[13]，ESL 適用於設計流程中的設計實現和功能驗證。它導入了一種稱為轉換程序層級模型(Transaction Level Model，TLM)的概念。一般而言，TLM 支援更高抽象級的設計，換言之，其實現細節更少，使系統匯編、變更和確認的速度更快。這些 TLM 模型可來自 IP 供應商，或用戶可利用 SystemC [14-15] 語言針對特定功能自行建造其行為模型，以便在功能驗證中

提供虛擬原型，且用抽象方法來描述 SoC 虛擬行為。具體來說，ESL 包含了基於平台的設計、各種功能裝置的建模、基於系統的模擬、軟硬體協同驗證、性能最佳化及基於系統的合成。

2.2.2 SystemC

SystemC [14-15]是一種被廣泛使用的系統級建模語言，透過該語言可實現測試平台的再使用，降低驗證成本，縮短驗證時間。SystemC 這種標準以 C++開發的資源開放式設計和驗證語言，為研究不同的系統結構，進行算法評估，軟、硬體任務劃分和軟體開發提供了有效的方法。SystemC 之所以能實現這些功能，原因就在於它簡化了轉換程序層級模型(Transaction Level Model，TLM)的開發。與暫存器傳輸級(RTL)模型比較而言，TLM 屬於系統硬體組件在更高等級上的抽象。RTL 模型中包含了比 TLM 模型更多的細節資訊(如單獨的時脈周期等)，而 TLM 則在結構級的組件上交換數據或執行事件。簡言之，TLM 所針對的應用是開發和驗證那些依賴於硬體的系統軟體部份。從這個角度來看，TLM 是更加容易讓使用者上手的新方法。本研究在系統內的各種週邊裝置皆是使用 SystemC 來加以描述。

2.2.3 RealView SoC Designer

RealView SoC Designer [16-19]乃是由 ARM [23]公司所開發的一套圖形介面的電子系統級工具軟體，用來整合硬體與軟體共同開發之虛擬平台環境，此套 ESL 工具包含了元件的模型建造、電路的繪製與系統的時脈模擬，可以配合 ARM 所提供的 Model Library 及自行開發的模型來建構出電子系統級的設計流程。此外，該項工具也支援廣泛使用的 SystemC 語言，藉由 RealView SoC Designer 建立的虛擬平台，可以讓不同的軟、硬體設計工具搭配組合，進行共同模擬驗證，以加速系統單晶片之開發與研究。此開發工具包含：

(a) SoC Designer Canvas：用來設計 SoC Designer 系統。可以整合不同的系統模組、設定不同的系統環境等。

(b) SoC Designer Simulator：用來模擬 SoC Designer 系統。提供系統驗證、及不同層次的除錯之支援。

RealView SoC Designer 在開發流程的初期就能確認系統架構。本研究整個系統的模型建造、電路繪製及架構模擬皆是使用 RealView SoC Designer 搭配 ARMulator 及 MediaBench suite 來完成。

2.2.4 ARMulator

本研究也使用ARM公司所開發的另一套圖形介面的電子輔助設計軟體 - ARMulator [20-21] 用來做為軟體開發上的整合服務發展平台，我們利用此軟體中所提供之armcc 與 armlink 進行編譯與鏈結效能評估程式（benchmark programs）。而效能評估程式的程式軌跡則是利用軟體中所提供之ARMSD Tracer （armsd）來加以取得。

2.2.5 MediaBench suite

本研究所使用的效能評估程式是 MediaBench suite [11-12]，此套程式是著重於多媒體與通訊系統的效能評估程式，其所包含的評估程式大多以影像、聲音與通訊編解碼程式為主，表1 為MediaBench suite內各種效能評估程式的應用說明。

表 1 效能評估程式的項目與說明

效能評估程式	說明
g271dec	G.721 聲音壓縮解碼程式
g271enc	G.721 聲音壓縮編碼程式
jpeg	JPEG 解碼程式
cjpeg	JPEG 編碼程式
mpeg2dec	MPEG2 解碼程式
mpeg2enc	MPEG2 編碼程式
pegdec	公開金鑰的解密程式
pegenc	公開金鑰的加密程式

3. 改良型共享式記憶體的設計

在本節，針對本研究所提出之改良型共享式記憶體在硬體架構的設計與運作原理作一個詳細地介紹。

改良型共享式記憶體其主要的功能與特色如下：

(a) 只可做資料的存取，不可做指令的提取。
(b) 改良型共享式記憶體不屬於快取記憶體。
(c) 可做為雙核心之間快速的互聯裝置。
(d) 內部有 512K-Byte 的靜態式記憶體(SRAM)，且此靜態式記憶體以被劃分成

四個容量相等(128K-byte)的區塊，分別為Bank 1 ～ Bank 4。Bank 1 與Bank 2 是Core 1 與 Core 2 各別專屬的資料存取裝置。Bank 3 只可讓Core 1 單獨做資料的存入， 且只可讓Core 2 單獨做資料的取出。而Bank 4 只可讓Core 2 單獨做資料的存入，且只可讓Core 1 單獨做資料的取出。Bank3 與Bank 4 主要是做為雙核心在互聯通訊上的資料暫存空間，且必需透過內部專屬的仲裁器(Arbiter)做Bank 1 ～ Bank 4 在資料存取上的管理。

(e) 靜態式記憶體的位址線寬度為18-bit。
(f) 靜態式記憶體的資料線寬度為16-bit。
(g) 資料存取的延遲時間為1～2個機械週期。
(h) 此改良型共享式記憶體擁有兩個獨立式的對外通訊埠，因此可個別與兩個核心做連結

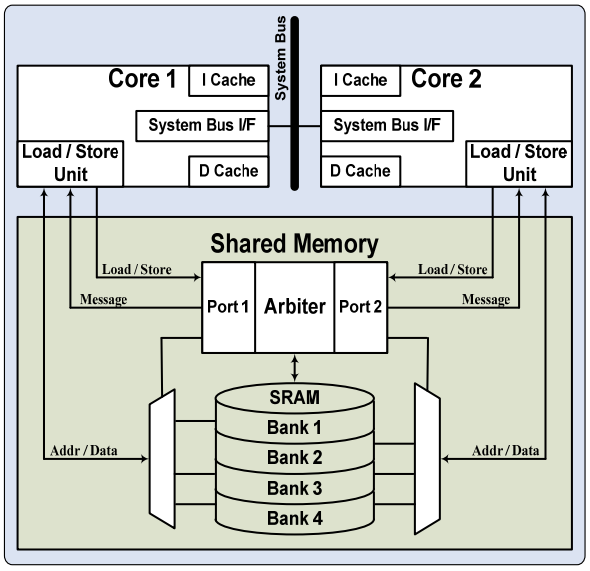

圖 3 改良型共享式記憶體的硬體架構

圖3 展示了本研究所設計之改良型共享式記憶體的硬體架構，我們將靜態式記憶體(SRAM)劃分成四個容量相等(128K-byte)的區塊，分別為Bank 1 ～ Bank 4，且利用各別獨立的通訊埠來分別與Core 1 及Core 2 進行資料的存取，此改良型共享式記憶體主要是做為雙核心之間的快速同步互聯機制，當系統中的任一核心(Core 1 或 Core 2)要對共享式記憶體進行資料存取時就必須使用到特殊的存入(Load)及取出(Store)指令，且改良型共享式記憶體中內含專屬的仲裁器(Arbiter)，其主要可用來：01.管理雙核心與共享式記憶體之間的存取時機，02. 作為雙核心在資料互相快速存取時的通道管制。另外，我們也提供了相關的互聯通訊程式來作雙核心之間的聯繫測試。在平時，Core 1 可對Bank 1 單獨進行資料的存入與取出，而Core 2 也可對Bank 2 單獨進行資料的存入與取出。當雙核心之間要進行通訊或資料交換時，就必需將雙核心之間所要溝通傳遞的訊息或資料透過Bank3 與Bank4 來做為暫存的媒介，此時雙核心與共享式記憶體之間的運作原理大致如下：(只說明將資料由Core 1 傳送至Core 2 的過程)

步驟一：由Core 1 送出Load / Store 命令旗標對仲裁器提出資料存入的要求。

步驟二:仲裁器會先判別Load / Store 命令旗標的數值後，接著仲裁器便發出相關的回應訊息(Message 旗標)給予Core 1 以便開始進行資料的存入行為。

步驟三：此時Core 1 開始對Bank 3 進行資料的存入動作，當Core 1 的存入動作完成後便發出旗標訊息(Load / Store 命令旗標)告知仲裁器動作已完成。

步驟四：仲裁器接著發出訊息(Message 旗標)告知Core 2 在Bank 3 中已有資料可被提取。

步驟五：此時Core 2 開始對Bank 3 進行資料的取出動作，當Core 2 的取出動作完成時便發出旗標訊息(Load / Store 命令旗標)告知仲裁器動作已完成。

以上的步驟同樣適用於Core 2 將資料存入Bank 4，而由Core 1 來加以取出。

表2 為雙核心用來與共享式記憶體進行通訊時所發出的Load / Store 命令旗標值。

表 2 Load / Store 的命令旗標值

Load / Store 命令旗標	說明
0000	Core 1 存入資料到 Bank 1
0001	Core 1 由 Bank 1 取出資料
0010	Core 2 存入資料到 Bank 2
0011	Core 2 由 Bank 2 取出資料
0100	Core 1 存入資料到 Bank 3
0101	Core 2 存入資料到 Bank 4
0110	資料存入結束
0111	資料取出結束

表3 為共享式記憶體內部的仲裁器用來與雙核心進行溝通時所發出的Message 旗標值。

表 3 Arbiter 發出的回應訊息

Message 旗標	說明
1000	允許資料存入
1001	允許資料取出
1010	通知 Core 2 可由 Bank 3 取出資料
1011	通知 Core 1 可由 Bank 4 取出資料
1100	通知 Bank 1 的空間已滿
1101	通知 Bank 2 的空間已滿
1110	通知 Bank 3 的空間已滿
1111	通知 Bank 4 的空間已滿

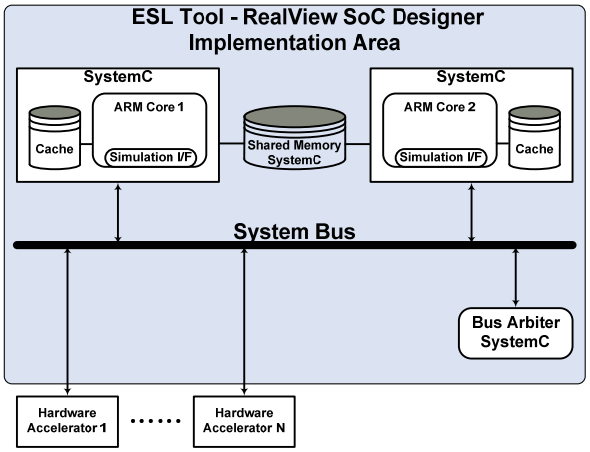

圖 4 本研究實現的雙核心硬體架構

圖4 展示了本研究所實現之雙核心系統晶片的硬體架構，我們已將原本的共享式記憶體搬離系統匯流排，且將它加以改良後再連接於雙核心之間以作為核心與核心之間快速的互聯裝置。

4. 架構模擬與結果分析

在本節將介紹雙核心系統晶片的架構模擬與結果分析。我們首先會利用SystemC 語法來建構共享式記憶體的電子系統級模型，之後再利用RealView SoC Designer 搭配ARMulator及MediaBench suite 來進行雙核心系統晶片的運作與效能之評估模擬。在4.1 小節將介紹如何利用ESL 工具設計改良型共享式記憶體之ESL 模型；在4.2 小節將介紹雙核心系統晶片之模擬電路與模擬環境；在4.3 小節將對雙核心系統晶片在ESL 的模擬結果作一個說明與分析。

4.1. 改良型共享式記憶體之ESL 模型設計

本研究利用SystemC 的語法來實現改良型共享式記憶體的ESL 模型，進而達到時脈週期的準確度（Cycle Accurate）。

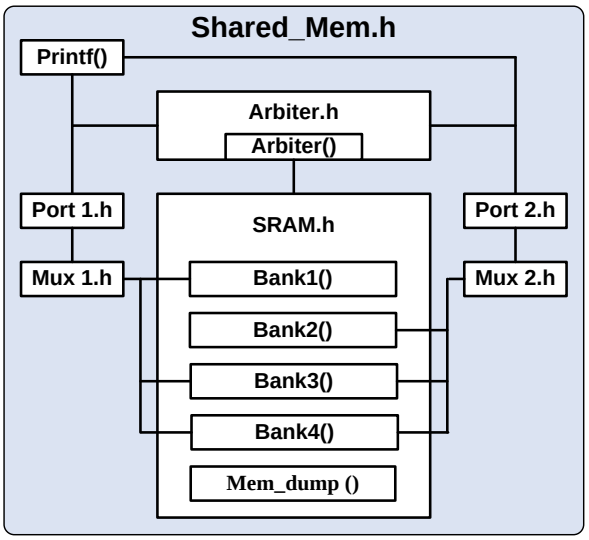

圖5 改良型共享式記憶體的SystemC 模型

圖5 展示了改良型共享式記憶體的SystemC 軟體模型，首先撰寫單位模組（Port 1.h、Port 2.h、Mux 1.h、Mux 2.h、Arbiter.h、SRAM.h），接著將各個小模組進行連接組合後即可組成整個ESL 模型(Shared_Mem.h)。另外，為了方便觀察暫存器內容值，加入將內容值以printf()或檔案的方式輸出。在資料存取的部分，由於記憶體模組(SRAM.h)必需提供雙核心進行資料的存取，因此在記憶體模組部分必需要有記憶體仲裁器來管理雙核心對四個記憶體區塊的讀寫動作，這個部分則由仲裁器模組內的Arbiter()函式完成。另外，為了方便觀察記憶體的內容值，在此模組也加入Mem_dump()函式將記憶體的內容值以printf()或檔案的方式輸出。由於本模型並沒有提供作業系統的函式呼叫，因此所執行的檔案必須是程式的軌跡檔案，並利用檔案方式輸入系統以進行模擬。關於模擬結果將在4.3 小節詳細說明。

4.2. 雙核心系統晶片之模擬電路與模擬環境

本研究中，我們使用ESL 設計工具架構出傳統型雙核心系統與改良型雙核心系統的模擬電路，在模擬電路方面，主要叫用了RealView SoC Designer 內含的處理器模型、Cache 模型、System Bus 模型及Arbiter 模

型，且將SystemC 語法所實現的改良型共享式記憶體ESL 模型整合進模擬電路中。之後再利用RealView SoC Designer 搭配ARMulator 及MediaBench suite 來進行雙核心系統晶片的運作與效能之評估模擬。

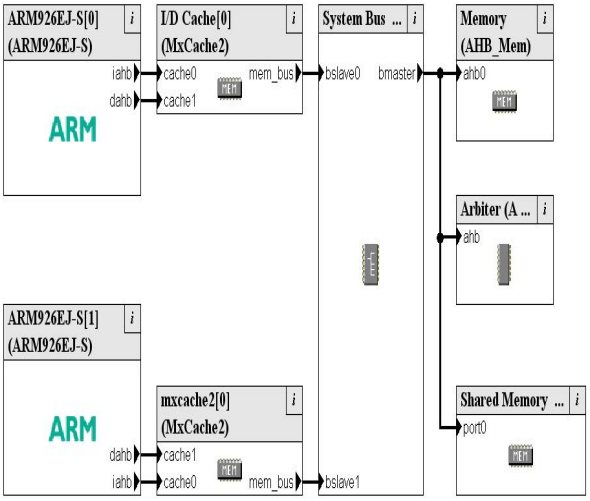

圖6 傳統型雙核心系統的架構模擬

圖6 展示了傳統型雙核心系統的架構模擬，我們叫用了兩顆ARM9 核心，在核心的下一級則分別連接了快取記憶體，而快取記憶體的下一級則連接到系統匯流排上，最後將共享式記憶體與系統主記憶體也連接到系統匯流排上。

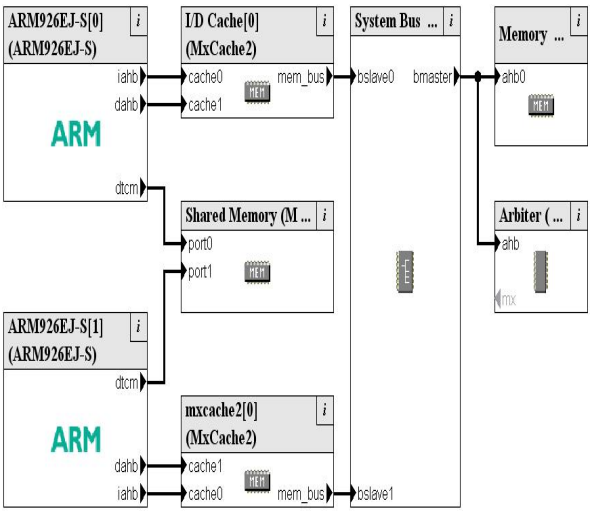

圖7 改良型雙核心系統的架構模擬

圖7 則展示了改良型雙核心系統的架構模擬，整體的架構與圖8 頗為相似，不同之處乃在於我們將改良型共享式記憶體連接於雙核心之間，當電路架構完成後，我們便利用圖10 的模擬環境來進行整個雙核心系統的模擬。

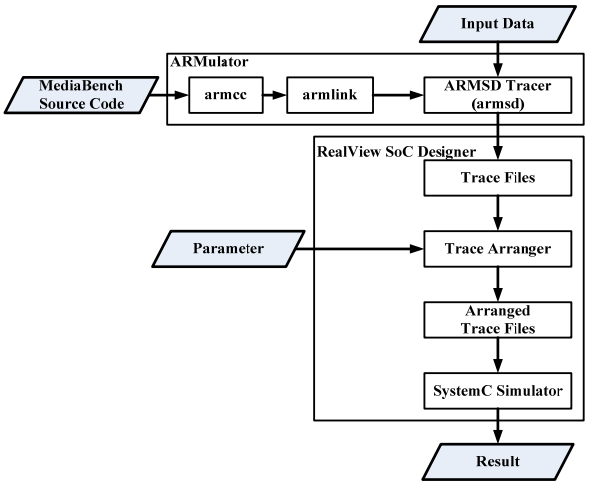

圖8 雙核心處理器的模擬電路圖

在模擬環境方面，我們採用類似文獻[4]的方法，唯一的差異點只在ESL 工具的廠牌不同而已，如圖8 所示。在評估雙核心系統的效能方面，採用程式軌跡導向模擬方式方法（trace-driven simulation methodology）進行模擬，利用ARMulator 中所提供之armcc 與armlink 進行效能評估程式（benchmark programs）的編譯與鏈結，而效能評估程式的程式軌跡則是利用ARMSD Tracer（armsd）取得。最後再利用RealView SoC Designer 來做整個雙核心系統晶片的運作模擬。

4.3. 模擬結果與分析

本研究在結果的呈現方式分為時序圖與統計圖兩種，而部份的模擬結果乃是利用ARMSD Tracer（armsd）取得程式軌跡後以統計圖的方式來加以呈現。

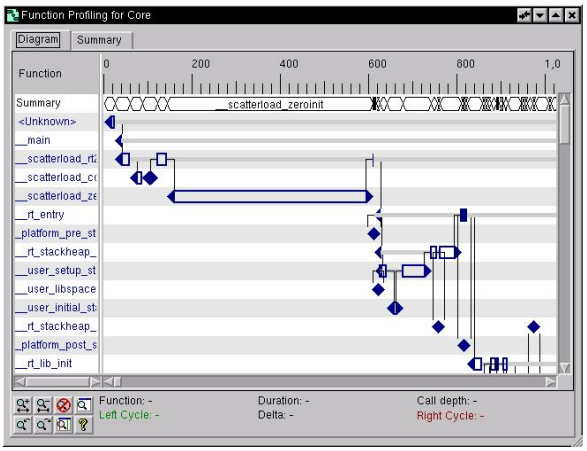

圖9 雙核心系統晶片的核心運作時序圖

圖9 展示了本研究所提出之改良型雙核心系統在核心運作時的時序情形。

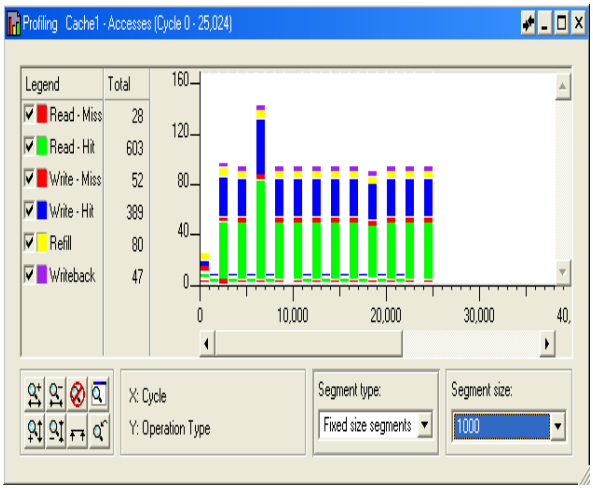

圖10 資料快取記憶體的運作存取統計圖

圖10 展示了雙核心內部的資料快取記憶體在資料讀取時的擊中率與遺失率之統計圖，由圖中可以發現，快取記憶體在做資料讀取與寫入時的擊中率頗高，而資料讀取時的遺失率很低，由此可發現本研究所架構的雙核心系統在運作時，快取記憶體的運行效率頗高。

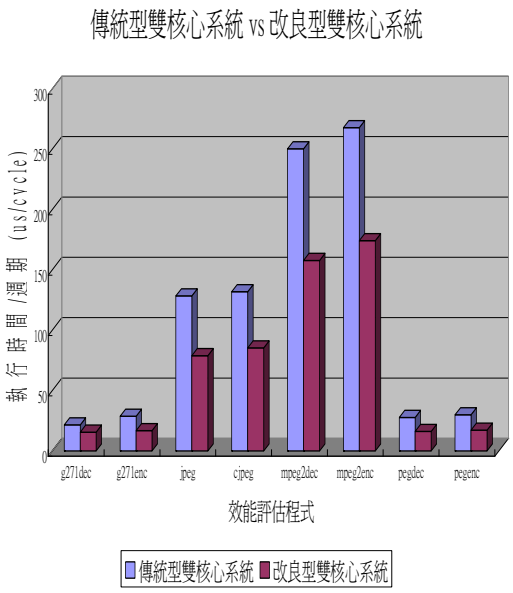

圖11 傳統型與改良型雙核心系統在程式執行的效能比較

圖11 展示了傳統型與改良型雙核心系統在執行效能評估程式後的結果比較，由圖中可以發現，改良型雙核心架構在執行各類效能評估程式時，其每週期的執行時間比傳統型雙核心系統來的短，顯示我們提出的雙核心架構確實有著較佳的效能。

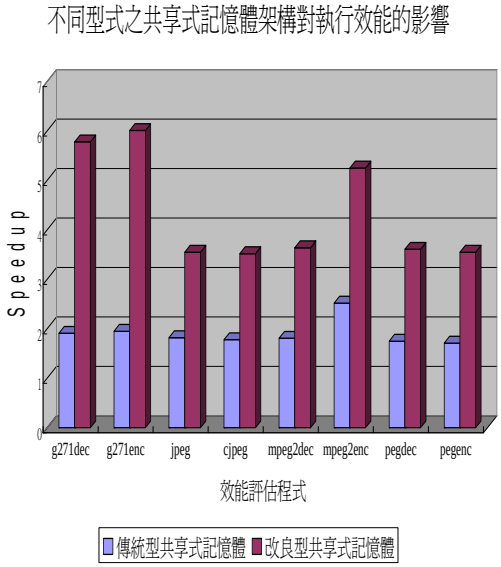

圖12 不同形式之共享式記憶體架構對執行效能的比較

圖12 展示了不同形式之共享式記憶體架構對執行效能的比較，由圖中可以發現，改良型共享式記憶體因其資料傳輸路徑不需經過系統匯流排，所以在雙核心系統執行效能評估程式上比傳統型共享式記憶體提昇了將近2～3的執行效能。

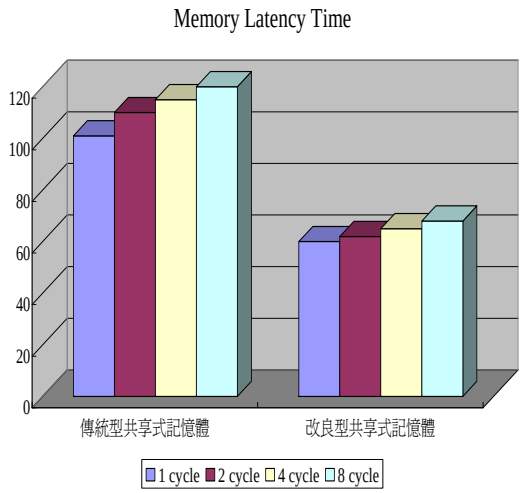

圖13 記憶體存取時間對雙核心架構之影響

圖13 展示了不同形式之共享式記憶體架構在資料傳輸時的延遲情況，由圖中可以發現，改良型共享式記憶體因其資料傳輸路徑不需經過系統匯流排，因此可大大地降低雙核心之間在進行資料互相傳輸存取上所需花費的延遲時間。

5. 結論與未來工作

本研究提出單晶片雙核心系統晶片架構，並且將傳統式雙核心架構內的共享式記憶體重新設計，以便做為核心與核心之間在資料傳遞與訊息聯繫的高效能裝置。另外，本研究使用 SystemC 語言建構此改良型共享式記憶體的 ESL 模型，並使用電子系統級設計工具繪製出系統晶片的整體電路。最後利用模擬平台來模擬與分析整個雙核心系統晶片的運作效能。在未來，希望能將本研究所架構的雙核心系統晶片以暫存器轉移層級(Register Transaction Level，RTL)的方式實現於 FPGA 之中，以便進行實體層的效能驗證。

參考文獻

[1] 王建章，"雙處理器架構之SoC平台"，***國立成功大學電機工程學系碩士論文***，pp. 13-81，2004。

[2] 周育樑，"支援超純量之ARM9 指令集的雙核心架構"，***國立中山大學電機工程學系碩士論文***，pp. 12-60，2005。

[3] 陳育宏，"多層匯流排架構仲裁器之設計及實作"，***義守大學電子工程研究所碩士論文***，pp. 10-36，2007。

[4] 陳柏愷，"單晶片超多純量處理器之ESL 模型設計"，***國立中山大學電機工程學系碩士論文***，pp. 20-59，2007。

[5] 陳俊宏，"以軟核心為主的可重組多處理器系統"，***大同大學資訊工程研究所碩士論文***，pp. 12-31，2007。

[6] 陳紀綱、蘇培陞，"以系統層級設計方法建立PAC PMP SoC 驗證平台"，***工研院系統晶片科技中心技術期刊第四期***，pp. 66-77，2005。

[7] Rongrong Zhong, Yongxin Zhu, Weiwei Chen, Mingliang Lin, "An Inter-core Communication Enabled Multi-core Simulator Based on SimpleScalar," *Advanced Information Networking and Applications Workshops Conference, IEEE*, pp. 170-175, 2007.

[8] L Peng, JK Peir, TK Prakash, YK Chen, D Koppelman, "Memory Performance and Scalability of Intel's and AMD's Dual-Core Processors: A Case Study", *Performance, Computing, and Communications Conference, IPCCC, IEEE International 2007*, pp. 55-64, 2007.

[9] JC Chiu, YL Chou, PK Chen, "A Superscalar Dual-Core Architecture for ARM ISA", *Proceedings of the International Computer Symposium 2006*, pp. 21-26, 2006.

[10] Wen-Ting Zhang, Luo-Feng Geng, Duo-Li Zhang, Gao-Ming Du, Ming-Lun Gao,Wei Zhang,Ning Hou, Yi-Hua Tang, "Design of Heterogeneous MPSoC on FPGA," *VLSI Research Institute, Hefei University of Technology, Hefei 230009, P. R. China, Computer Design. Proceedings. International Conference, IEEE*, pp. 102-105, 2007.

[11] Bishop, TP Kelliher, MJ Irwin, "A detailed analysis of MediaBench," *Signal Processing Systems, 1999. SiPS 99. 1999 IEEE Workshop, 20-22*, pp. 448-455, 1999.

[12] C Lee, M Potkonjak, WH Mangione-Smith "MediaBench: a tool for evaluating and synthesizing multimedia and communications systems," *Micro-architecture, 1997. Proceedings. Thirtieth Annual IEEE/ACM International Symposium on Publication Date: 1-3*, pp. 330-335, 1997.

[13] *ARM RealView ESL API v2.0 Developer's Guide*, ARM Corp, 2007.

[14] Open SystemC Initiative, http://www.systemc.org/

[15] *IEEE Standard SystemC Language Reference Manual*, IEEE Std. 1666-2005, 2006.

[16] *RealView SoC Designer v7.0 SystemC Linking Guide*, ARM Corp, 2007.

[17] *RealView SoC Designer v7.0 Developer's Guide*, ARM Corp, 2007.

[18] *RealView SoC Designer v7.0 User Guide*, ARM Corp, 2007.

[19] *RealView SoC Designer v7.0 Standard Component Library Reference Manual*, ARM Corp, 2007.

[20] *RealView Developer Suite v2.2 AXD and armsd Debuggers Guide*, ARM Corp, 2004.

[21] *RealView ARMulator ISS v1.4 User Guide*, ARM Corp, 2004.

[22] SimpleScalar, http://www.simplescalar.com/

[23] ARM Corp., http://www.arm.com/