應用系統層級設計方法在 SoC 平臺架構之探討

陳振隆 王勝德
國立台彎大學電機工程學系
sdwang@ntu.edu.tw

摘要

隨著 IC 製程的進步,今日已可以將上億個電晶體置入到一個晶片中，也使得今日系統晶片的設計日趨複雜。系統層級的設計方法使得系統開發人員可以提高設計的抽象層次，在設計階段能快速的進行實驗，收集數據加以分析與改良系統架構，進而降低系統單晶片開發的複雜度。本篇論文使用了 CoWare 的系統層級開發工具 ConvergenSC，並採用 SystemC 硬體描述語言。在這篇論文中以交易層級模組化方法設計各個硬體元件建立抽象模型，並使用這些工具在高抽象層次建構三個交易層級的虛擬原型系統架構，藉由模擬出來的數據分析各個系統架構的效能，並加以探討在各個系統架構的優劣點。

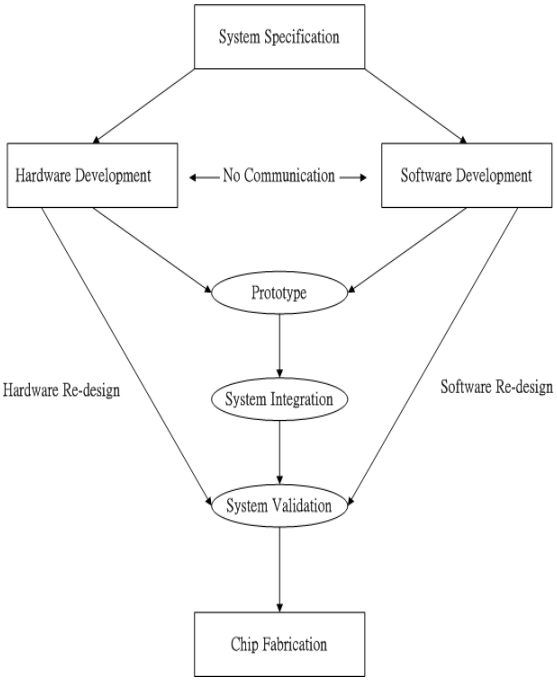
關鍵詞：電子系統層級；系統單晶片；架構探討。

1.序論

當代的晶片中有著龐大的電晶體數與強大的設計功能，但其複雜度也遠超過我們頭腦所能處理的範圍。傳統的主流設計方法是以 RTL(Register Transfer Level)這個階層為主來設計晶片的規格和驗證的流程。這種設計方法主要是讓設計的工程師，根據所得到的硬體上的規格，以 HDL(Hardware Description Language)精確地描述一個硬體 IP(intellectual property)的內部架構，以 HDL 模擬驗證操作，以邏輯合成軟體得到電路佈局圖(netlist)，此時邏輯合成軟體會考量到面積、操作頻率等限制，基於特定的半導體連接技術而構成電路。這種精確的描述方式確保了電路的可製造性，但也產生了一些問題。系統開發人員設計整個晶片系統(SOC)，必須先花費冗長的時間開發各個硬體 IP，例如 MPU(Main Processing Unit)、On-Chip Memory、系統匯流排、中斷控制器、數位訊號處理器(Digital Signal Processor)等電路元件，開發各個 IP 元件，並且驗證這些 IP 是否符合規格，接著整合這些開發完成之 IP 成為單一系統後，加入軟體進行系統模擬驗證。這些過程耗時與系統架構不易修改、模擬過程緩慢且不易收集分析數據。因此設計人員往往無法在硬體實際製造出來前，撰寫測試程式，以確認所有的時序與功能都符合預期的規畫。

一個專為特定應用設計的系統有著一個包括處理器的硬體架構，部分的功能在處理器上執行，而其他功能則用特殊設計的硬體來執行，這就是所謂的嵌入式系統，因此當面臨著這種龐大的系統設計問題,需要一個有系統的、分層的設計方法。系統層級設計方法則帶來了這種新的設計理念，由於在架構設計階段所應該考量的是改良軟硬體架構，而不是針對單一元件來做改良，因此將硬體描述的抽象層級由 RTL 提升到 Transaction Level，使用 Transaction Level 抽象層級描述的硬體模型不需考量硬體電路內部的設計細節，只要正確的定義電路的行為即可，因此尊循此原則，用 Transaction Level 描述的硬體元件就可以比 RTL Level 描述的元件更易製作，並且更快速的進行模擬。下面圖 1

為傳統的設計流程和以系統層級方法的設計流程做一個比較。傳統的設計流程在有限時間內要決定最終系統架構，且在設計的早期要做到效能評估與頻寬使用量計算等等是需要花費非常高的成本。如何找到最佳的效能、記憶體與匯流排最適合的架構，使用系統層級方法可在設計的早期解決這些問題。軟體與硬體的分割、IP 的重複利用在系統層級方法的高模擬速度與系統可變性下可找到最好的使用方式。在這些優點下，本篇論文將採用此種方法，並使用目前在電子系統層級(electrical system level，簡稱 ESL) 設計工具為主流的 Coware[7]的 ConvergenSC，利用它當作整合我們所設計的 Transaction Level 元件的 SoC 平臺，並證明它確實能帶來一些與傳統設計上不同的優點。並且撰寫應用程式，放入此 Transaction Level 的 SoC 平臺，進行模擬，以驗證 SoC 的設計是否正確無誤。



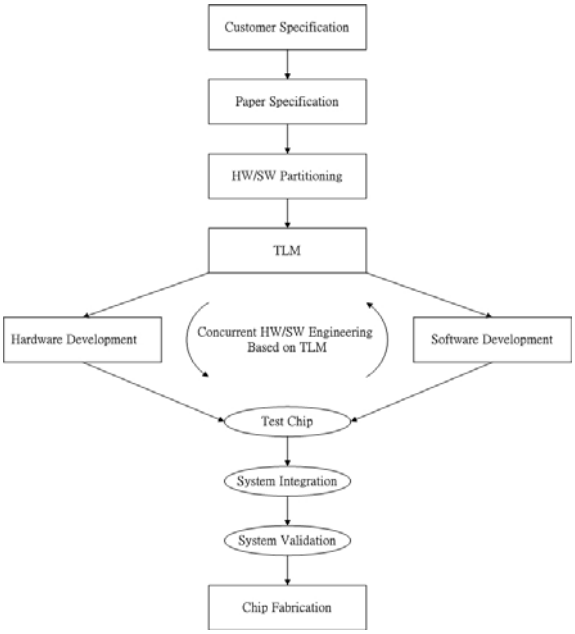


圖 1 傳統設計流程和系統層級方法的設計流程

本論文的研究目標是利用此嶄新的設計理念，使用 ConvergenSC 當作 SoC 模擬平臺，在 Transaction Level 這個抽象層次描述硬體 IP。設計觀念以軟、硬體分割，軟、硬體共同設計方法來建構整個 SoC 系統。論文計畫使用 JPEG 壓縮程式，將 JPEG 壓縮程式中，最佔資源的部分，描述成硬體 IP 加入到系統內取代之。論文中並將建構三種架構，將利用 ConvergenSC 的 profiling 工具取得相關數據，加以分析架構上的不同對軟硬體整合上效能的差異性，並找出最佳化的一個架構。

本論文第一章說明研究動機與目標，並介紹整個論文的架構，第二章將介紹論文中應用到的方法和其相關研究與歷史，在第三章中開始說明在應用 ConvergenSC 與在 Transaction Level 層次實作上的整個流程與實作過程中應注意到的一些其他考量因素。第四章將對於實作完畢後取得的數據做一個分析，並提出改善的方法與建議。最後我們將說明此論文研究出來後得到的結論。

2.相關研究

2.1 電子系統層級設計

電子系統層級(Electronic System Level)設計是近幾年來在電子設計自動化(Electronic Design Automation)工業界中所產生的一個主流新名詞。它的論點主要在於提出一個新的思考方向將整個設計流程中各個設計部分應用高階的描述方法提高至更高的抽象化層次。在過去三十年的演變中，硬體設計的方法在抽象化層次中一直是朝向上提升的方向發展。從手工繪製電路圖到邏輯閘層次(Gate Level)設計接著暫存器傳輸級(Register Transfer Level)硬體描述語言的發明。在這期間整合系統如:手機、個人數位助理器(Personal Digital Assistant)等高度設計複雜度電子裝置的發明，開啟了軟硬體共同設計的裏念。越來越多的電子計算裝置混雜了高計算量的軟體和硬體，但這也提供了更高的彈性去劃分軟體和硬體各部分。在開發人員開始硬體邏輯閘的設計和嵌入式軟體人員撰寫軟體程式碼之前，架構的探索在系統設計流程中對於如何取得一個在效能和成本上的平衡扮演一個重要的角色。但現今整個系統設計的複雜度已增加至難以有效率的方法去設計它尤其是在硬體方面。因此描述層次的提升和開發相關電子設計自動化工具能夠幫助工程師在摩爾定律(Moore' law)的演變中，仍然能保持一個有效率的設計流程。

電子系統層級設計主要在探討下面幾點:

1.產品架構和規格的開發，包含了整合和設定現有的 IP。

2.產品的應用對映所需要的規格，包含了軟體與硬體的劃分以及處理器的最佳化。

3.產品投產前的為軟體開發用的虛疑硬體平臺的製作。

4.架構上硬體自動化的合成實現。

5.開發驗證硬體用的模型。

根據以上的幾點，我們探討近年來電子系統層級設計在產品產率和設計品質上的改善所研究出來的語言、工具、技術和方法。在研究電子系統層級的文獻[22]中，作者使用了一種叫 Gezel 的語言，它的核心語言為 C++ library，和本論文中使用的 SystemC[25]也是 C++ library 的集合，作者在這篇論文提出了一種電子系統層級設計流程並強調出它的抽象化與可重複使用性的優點。基於平臺設計是一個理念[29]，它的目標是系統的再使用與建造一個可以從設計者觀點掌握的系統。文獻[8]參考這樣的理念提出了一個分類方法，將目前電子系統層級設計相關的工具與方法做了不同層次的分類，以避免使用者發生運用工具過程中產生了不適當性。文獻[26]提供了電子系統層級設計系統階層驗證的模擬系統，可以快速的檢驗不同的硬體架構。這方面的研究比較不常見，通常是用於商用工具的開發，比較有名的為 MaxSim[2]及 ConvergenSC，這些工具均以 SystemC 以 Transaction-Level 的概念描述所設計的 IP，包括模擬處理器用的指令集模擬器(Instruction-Set Simulater,，簡稱 ISS)、記憶體、匯流排、快取、直接記憶體存取、ASIC 及各種 I/O 周邊硬體。軟硬體劃分法[20][9]，這是系統設計人員設計嵌入式硬體架構常用的方法，將應用系統中的各個函數根據經驗區分為硬體或軟體執行。以軟體執行時需選擇處理器，決定記憶體空間等；以硬體執行時需決定使用之 IP、製程、工作頻率、電壓及整體考量匯流排架構等等。文獻[3]將電子系統層級設計方法應用在雙核心單晶片設計，以 Transaction-Level 的層次模組化整個平臺，並比較使用二個 Inter-Process

Communication(IPC)機制 mailbox 和共用記憶體的不同點。文獻[28]則提供了一個專為多種處理器單晶片模擬的平臺 CASSE，這個工具能在設計流程的早期提供一個快速模組化和分析的能力，它並以一個 MPEG4 decoder 的應用當作範例。UML(Unified Modeling Language)在電子系統層級設計的相關方法[12]，這篇文獻探討 UML 關於電子系統層級設計的基本原則、相關工具以及如何將 UML 轉換至 SystemC。

2.2 SystemC

現在單晶片要成功實現，由於奈米製程的進步和應用面需求的影響，設計者將要面對龐大系統規格設計與硬體驗證的諸多問題，過去的做法都用硬體描述語言來作晶片硬體部分的設計，而現在的整個設計流程中則強調 IP 整合與重複使用，因此近年來許多電子系統層級設計工具都採用 SystemC 這種新一代的系統與硬體模型建立語言來開發單晶片，以簡化設計流程並做軟硬體共同設計。SystemC 是 1999 年 9 月由業界知名科技公司如:arm、Motorola 等與 EDA 廠商組成的 OSCI 組(Open SystemC Initiative)組織所訂定的一種系統設計與驗證語言，它是一種 C++ library 的強化。由於 C 和 C++這類程式語言均是硬體和軟體工程師最主要使用的，C++的物件導向語法滿足 SoC 設計過程中需整合 IP、物件或是軟體的需求。用這種方法制訂的目的是想在軟硬體共同設計過程中減少溝通不良的情況，原因在於軟體和硬體工程師均使用相同 C 和 C++這類的程式語言當作開發工具，這樣可以簡化設計流程，提高設計的效率。設計者在系統層次、行為層次或 RTL 層次使用 SystemC 類別函式庫描述系統元件，這些類別函式庫提供二個重要的目的，第一它提供了針對硬體規格需求的物件，如並行(Concurrent)、階層式模組(Hierarchical Modules)、埠(Port)和時脈(Clock)等等，也新增許多支援硬體觀念的 C++類別，包含有各種硬體描述的資料型別，訊號(signals)、事件(events)等硬體行為溝通模式，支援時間、中斷與例外處理等系統模型需求。第二它包含了一個模擬核心用來排班工作行程。使用者將以 SystemC 開發系統並也使用它去撰寫相關的測試程式(Testbench)，SystemC 可使用標準的 C++ complier 如：GNU[10]的 GCC 和除錯工具如：GNU 的 GDB 做為研發工具，也可在 Microsoft[16]的 Visual C++環境下運行。此外 SystemC 也提供了模擬結果以波形表示訊號值軌跡功能，支援的軌跡檔案(Trace File)格式有 VCD、WIF、ISDB 等可使用在標準的波形展示工具。

2.3 交易層級模組化

交易層級模組化(Transaction Level Modeling，簡稱TLM)，其主要精神在將硬體的抽象層次提高，降低系統單晶片開發的複雜度。使用TLM的技術[21]，每個元件的內部關於資料通訊的細節以通道模組化，並在通道上使用特定的介面函數，因此所用通訊的細節在TLM的技術中是被隱藏的，如此在這種高抽象層次上模擬的速度比傳統的RTL層次快，使得系統設計人員可以在架構設計的早期階段進行實驗，收集數據找出問題點，並可以容易的修改系統架構。設計系統第一個需要的動作是改良軟硬體架構[24]，所以我們需要硬體的模擬模組，而且它必須符合系統規格要求的功能，也能夠載入且執行軟體開

發人員所開發的應用程式。以TLM製作的元件為(Transaction Level Component，簡稱TLC)，而使用TLC所建構之虛擬架構稱之為 (Transaction Level Architecture，簡稱TLA)。TLC的內容包含了描述電路行為的狀態機(Finite State Machine，簡稱FSM)，FSM在每一個外部觸發發生時，例如時脈邊緣的上升或下降改變，即會根據內部儲存的資料或是外部的輸入，來決定下一個狀態機的狀態與輸出的資料。TLC在每一個外部的輸入時脈進行一次運算，判斷是否需要改變狀態、內部資料或改變輸出埠的資訊，由於每一個時脈循環計算一次電路狀態的行為，這種描述方法時使得TLC保證了它的時脈循環準確性。在TLA方面目前有二套成熟ESL工具可供執行與模擬，如前所提的ARM的MaxSim與Coware的ConvergenSC，在ConvergenSC中負責建立與修改TLA的工具稱之為Platform Creator，提供了改變匯流排架構，修改TLC元件參數，修改記憶體對映圖(Memory Mapping)等等。當TLA建立後，接著的工作就是要根據所要分析的問題，撰寫對應的軟體應用程式，然後利用架構模擬器(TLA Simulator)進行模擬，System Verifier為ConvergenSC上所提供的工具，它的主要工作在於將應用程式載入TLA進行模擬，並且在模擬過程中進行資料剖析(Profiling)的動作，以做為系統設計人員改進架構時重要的參考資料。近幾年來有很多應用TLM技術的研究發表，文獻[14]在硬體部分應用SystemC與TLM技術，軟體部分則使用ISS，在這種Cosimulation的方法下，它提出了四個架構，它們分別使用了四種內部程式通訊(Interprocess communication，簡稱IPC)的機制，實驗數據說明瞭這四種模擬架構下的效能比較，並藉此探討出一個最可行的Cosimulation的方法。在匯流排(bus)的TLM技術上，文獻[17]用TLM應用在AMBA[4]匯流排的模擬。由於SystemC可將硬體描述在不同的抽象層次上，文獻[15]則應用在ABMA上的TLM技術加以改良提出一個新的抽象描述的層次，名稱為Cycle Count Accurate at Transaction Boundaries，簡稱為CCATB。它和以Bus Cycle Accurate層次描述的模型，利用匯流排協議(protocol)和仲裁機制(arbitration)的設定不同，做一個匯流排動作速度與次數的探討。在能源消耗的量測方法方面，文獻[21]針對不同的硬體元件提出了不同的量測模型，文獻[18]針對AMBA AHB Bus提出了一個能源消耗的量測模型，將不同的計算能源消耗公式應用在匯流排不同的仲裁行為上。這方面的量測缺點為由於抽象層次越高的關係，量測出來的數據誤差將比在較低描述層次的誤差範圍大，如：RTL。文獻[27][19]提出在Transaction Level上匯流排架構最佳化的探討，利用多重匯流排(Multilayer bus)架構和不同的仲裁機制並將硬體IP放置在適當的位置，以求得一個資料傳輸效能最好的架構。

3.方法與實作

本論文的架構將參考文獻[5]所提的架構利用 ConvergenSC 這個工具建構出來加以分析， 論文中所提的架構著重在 AMBA-Based 的系統平臺，實驗中我們將建構出三種架構。在軟體方面將使用 ADS (ARM Developer Suite)編譯出符合 ARM CPU 格式的程式，論文將以[11][1] JPEG-6B 壓縮程式碼當作實際在架構上執行的多媒體應用程式。

3.1 軟體方面的實作

首先將 JPEG-6B 壓縮程式碼以 GPROF 工具來手動分割程式碼，由於離散餘弦轉換(Discrete Cosine Transform，簡稱 DCT)運算在整個程式碼中佔最多資源，因此將它實作成硬體取代原來在軟體 DCT 運算部分。RGB TO YCbCr 轉換的部分內有浮點運算，由於 ARM 處理器並不支援浮點運算(Floating Point Math)。所有的浮點運算都是在浮點運算模擬器上進行，因此特別緩慢。需要浮點運算的函式，常要耗費數千個循環才能執行，為了減少模擬所花費的時間，也將它實作硬體加入到硬體 DCT Device。整個 JPEG-6B 程式執行流程圖如圖 2 所示，將 DCT 部分以及 RGB TO YCbCr 部分以 SystemC 描述成一個硬體加速器 DCT Device 以加快處理速度。

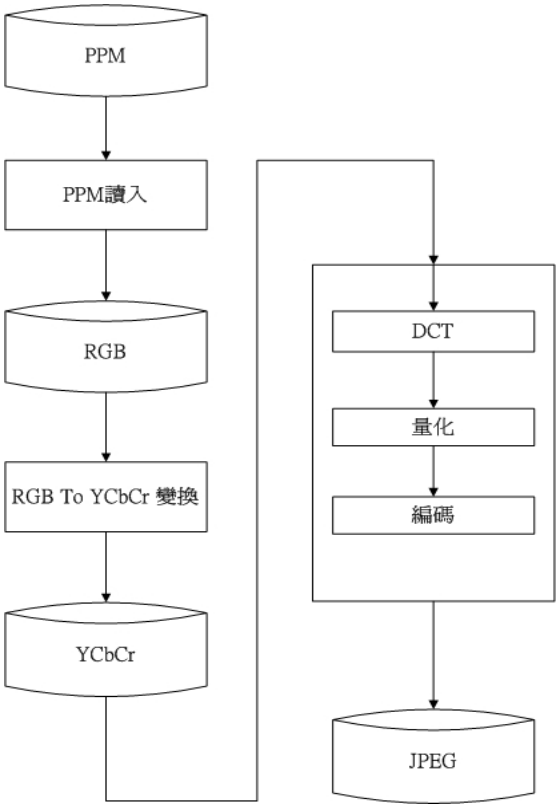


圖 2 JPEG-6B 程式執行流程圖

3.2 硬體方面的實作

一個典型的嵌入式系統如圖 3 所示，至少有一個處理器、數個記憶體、功能單元、I/O 裝置等。這些元件可透過一條匯流排加以連接，這些可分為主(master)元件和副(slave)元件，主元件至少有一個主埠(master port)，副主件至少有一個副埠(slave port)，主元件利用主埠傳輸一個讀取(read)或寫入(write)的動作到匯流排上，匯流排再根據要傳遞的位址對映至副元件，將要求的動作透過副埠使副元件得以接收，諸如此類的動作稱之匯流排交易動作(bus transactions)。而副元件本身沒有要求匯流排交易的權限，每個元件都有其位址在系統上的對映如圖 3 所示。

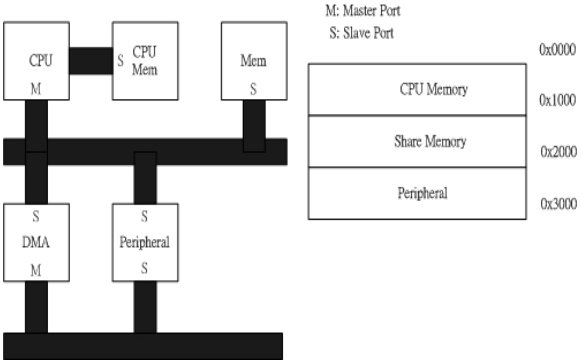


圖 3 典型的嵌入式系統架構圖

硬體所有的 IP Block 都以 SystemC 描述在 Transaction Level，實作中將採用文獻[6]所提出的 Transaction Level Modeling 的準則。TLM 對於 TLC 的描述準則可為四個部分。

1. 資料結構(Data structure)包含了暫存器(Register)、狀態暫存器(State Register)、變數(Variable)和常數(Constant)。暫存器包含了來源(Source Address)、目的(Target Address)和控制(Control)等資訊，而狀態暫存器含有目前元件的狀態資訊，這些暫存器都為一個 TLC 必須具備的。

2. 狀態機(Finite State Machine，簡稱 FSM)，採用米麗機(Mealy Machine)的方式實現，它的方式為輸出由目前的狀態和輸

入決定，而相關資訊和狀態都在資料結構內取得，輸入資訊則透過副埠取得。

3.埠(Port)分為主埠和副埠，主埠取得 FSM 的輸出，來決定匯流排交易的動作，而副埠則被主元件控制使用，在副元件上做讀入以及寫出的動作。

4.中斷(Interrupt)的功能，不一定每個 TLC 都有設計中斷的功能。中斷輸入將使得 TLC 進入中斷服務程式(Interrupt Service Routine)。中斷輸出則根據 TLC 內部的 FSM 決定何時提出中斷要求。

所有 TLC 型式如下圖 4 可分為下列三種：

1.主元件(A master component)，只有一個主埠，它只能允許中斷輸入，沒有中斷輸出，如：中央處理器就是這類元件。

2.可定址的副元件(An addressable slave component)，只有一個副埠，可以採用中斷輸入或中斷輸出，一般週邊元件都用此型態設計。

3.可定址的元件(An addressable Component)，這類元件有主埠、副埠以及中斷輸入、中斷輸出設計，通常應用在如：直接記憶存取(Direct Memory Access，簡稱 DMA)這類元件的設計上。

3.3 架構整合實作

CovergenSC 分成三成部份分別為 System Verifier、System Designer 和 Advanced ystem Designer。System Verifier 提供了模擬和除錯的環境，而 System Designer 主要則是提供了分析的工具，Advanced System Designer 即是 Platform Creator 提供了平臺組合與設定的能力。在論文引用到了 ConvergenSC 所提供的 ARM/AMBA IP，實驗中將使用 ARM 9 family 中的 ARM 926 當成指令集模擬器模擬的對像。ConvergenSC 使用如下，首先使用 Platform Creator，繪製出所欲建立的平臺，將自行開發的 SystemC module 載入至平臺上，memory map 設定各個元件在平臺的位址，當平臺建構完畢，將做一個檢查(check)與輸出(export)的動作，並呼叫 SCSH Shell 使用 GCC 去編譯整個平臺，當編譯完畢即可呼叫出 ARM debugger 去執行在 ADS 工具下編譯成的軟體。ConvergenSC 提供了 AMBA IP 實驗中將主要使用 AMBA Library 中的 AHB 以及 APB 當作系統元件溝通的通訊匯流排，這些匯流排以 TLM 技術模組化，架構中的硬體元件以 SystemC 描述，行程中關於匯流排的控制部分將由匯流排的事件觸發。論文將利用 ConvergenSC 繪製三個架構圖，並加入所設計的硬體。

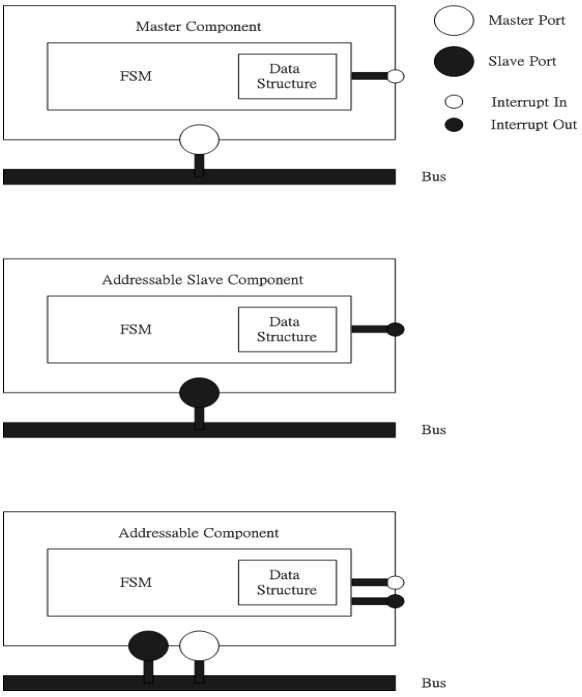


圖 4 三種型態的 TLC 元件

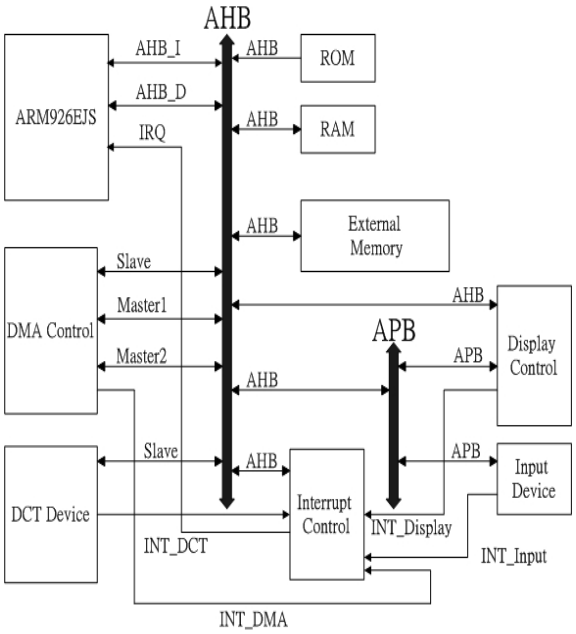


圖 5 第一個系統架構

第一個架構如圖 5 包含了一個 AHB (Advanced High-performance Bus)和一個 APB(Advanced Peripheral Bus)匯流排，所有的 AHB 埠包含了初始(initiator)埠和目標(target)埠都連接到這個 AHB 匯流排。而二個 APB 目標埠連接到 APB 滙流排，最後 APB 匯流排透過 AHB-APB 橋接器連接到 AHB 匯流排。第二個架構如下圖 6 所示，包含了一個多重匯流排(在 ConvergenSC 設計裏為輸入階(input stage)和輸出階(output stage)所構成的矩陣)，兩個 AHB 匯流排、一個 AHB-Lite 匯流排以及一個 APB 匯流排，另外 AHB-Lite 與 APB 之間則需要一個 APB 對 AHB 轉換連接的橋接器。第三個架構如下圖 7 所示有一個 AHB 和一個 APB 以及一個多重匯流排和第二個架構不同處在於它有五個輸入階，而第二個架構則使用到了三個輸入階。除了上述建構的三種架構外，在實驗過程中將嘗試在第一個架構把 DCT Device 移除，由於移除了 DCT Device，在軟體部分將完整執行 JPEG-6B 壓縮程式碼。移除了 DCT Device 的第一個架構如下圖 8 所示。藉由軟硬體的重新劃分，實驗過程中將比較 DCT Device 硬體加速器所帶來的效能改善。

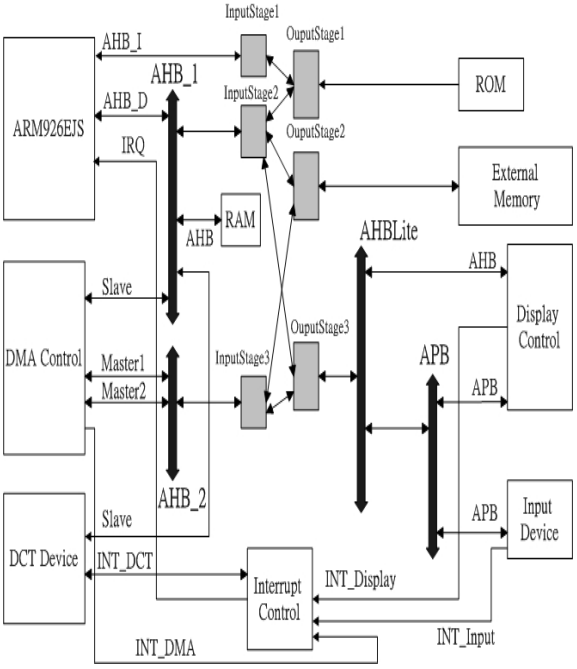


圖 6 第二個系統架構

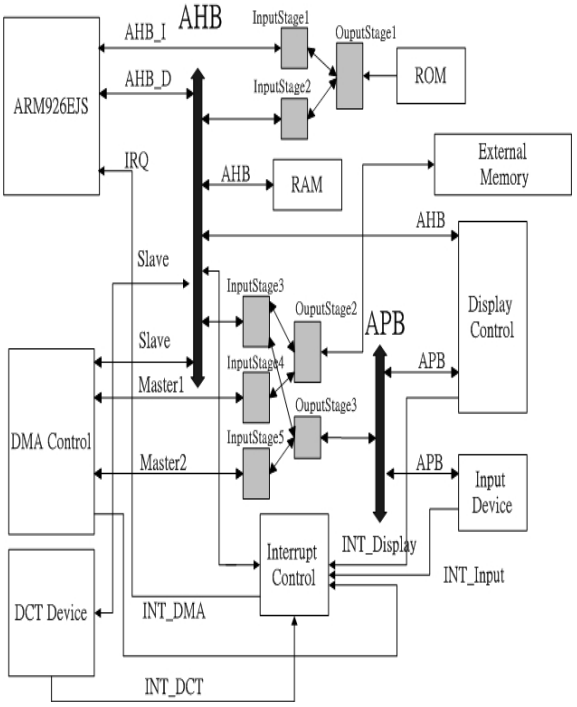


圖 7 第三個系統架構

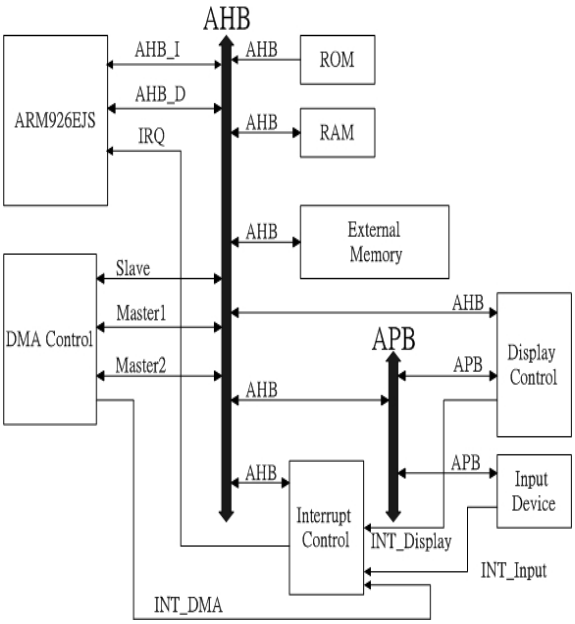


圖 8 第一個系統架構(移除了 DCT Device)

4.實驗結果分析

本論文實驗主要分成三個架構去執行 JPEG-6B 壓縮程式，實驗將使用四張大小不同的圖檔去做為來源的原始圖檔資料，原始圖檔的資料格式為[23]PPM(Portable Pixel Map)，為一個簡單的非壓縮性圖檔格式，用來作為 JPEG-6B 程式的輸入檔。整個架構的執行過程如下：

1.首先 ARM926 從 ROM 當中讀取啟動程式碼進行啟動動作。

2.接著從軟體當中執行關於各個硬體 IP 的初始化動作。

3.圖檔從 Input Device 讀取輸入，透過 DMA Device 傳送至 External Memory。

4.同時間，ARM926 從 External Memory 讀取資料，進行資料解析的動作，並儲存至 RAM 當中。

5.ARM926 從 RAM 當中把資料傳送至 DCT Device 執行硬體加速運算，並將結果回存至 RAM 當中。

6.ARM926 執行量化編碼的動作，並將結果傳送至 Display Device。

7.Display Device 執行寫檔動作，將 JPEG 圖檔存至磁碟中。

當以上三個架構實驗完畢後，實驗將把第一個架構的 DCT Device 移除，將 DCT Device 運算部分改由軟體執行，並各別輸入四個圖檔做運算。實驗完畢後將利用 ConvergenSC 提供的分析(Profiling)工具比較所有架構上效能的差異性。

4.1 系統架構探討

在論文中，實驗建構了三個架構，架構的改良以減少傳輸所需要花費的時間為主，在第一個架構中由於各個元件的輸入及輸出埠幾乎都透個同一條 AHB 匯流排在進行溝通，但當各元件的主埠在同一時間要求匯流排的使用權時，容易造成所謂的匯流排衝突(Bus contention)的情形，所謂的匯流排衝突代表當兩個以上的主埠同時間想利用同一條匯流排當作傳輸路徑時，這時就需要匯流排仲裁機制去決定優先權順序，這時只有一個主埠能取得使用權，而其他則進入等待狀況。在這樣的情況下，會造成同一時間只有單一元件進行動作，而其他元件則因為優先權較低，而進入閒置(idle)狀態中。而為了改善這個缺點，在第二個架構以及第三個架構中，使用了多重匯流排這個機制，以減少資料傳輸上所耗費的時間。多重匯流排在 ConvergenSC 設計裏為輸入階(input stage)和輸出階(output stage)所構成的匯流排矩陣(Bus Matrix)。多重匯流排是 ARM 公司為了考量在 AHB 系統中若有兩個主埠常需要去存取匯流排則系統的效能必定會下降。這種多層的 AHB 匯流排架構(Multi-layer AHB)其基本構想如下圖 9 所示。基本構想是兩個主埠(master) 走不同的匯流排去存取副埠(Slave)，若存取的

slave 不同，則兩個 master 可以同步的進行 transfer。若彼此存取同一個 slave 則由 slave 去判斷要先處理誰的 transfer。在第一個架構中，可以很清楚的判斷出那裏是最有可能發生匯流排衝突的地方。如下圖 10 所示。在圖 10 標示出了最有可能發生匯流排衝突的地方，因此在第二個架構使用了多重匯流排來降低匯流排衝突發生的可能性，但在第二個架構和第一個架構比起來需要多使用一個 AHB 以及 AHBLite 匯流排，因此在第三個架構的多種匯流排上多增加了二級的輸入階，這樣消除了在架構二上多需要二個 AHB 匯流排的考量，同時使的傳輸路徑更為簡化也降低了發生匯流排衝突的機率，並進一步使得架構的效能更為提升。

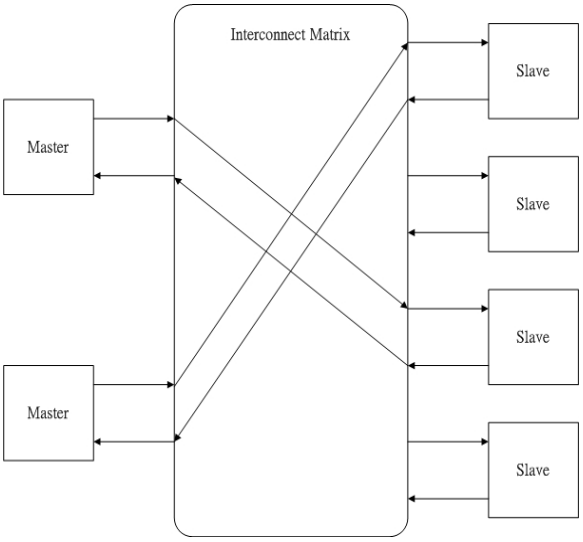


圖 9 Multi-layer AHB 基本架構圖

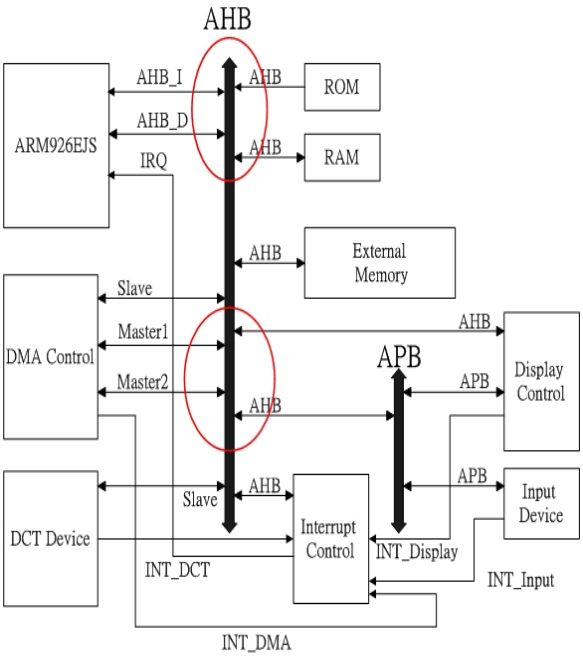


圖 10 第一個架構中匯流排衝突的發生點

4.2 匯流排衝突分析

實驗過程使用 ConvergenSC 提供的 Profiling 工具去量測匯流排的使用狀態。圖 11 為架構一的匯流排衝突狀態，此圖為量測在架構一 AHB 匯流排上匯流排的衝突狀態。它可以表示出匯流排中平均 master 等待的數目。由圖中可看到，整個執行的時間過程中，匯流排上等待的 master 的數目平均在 0.5 個以上，這是由於在架構一 ARM926EJS 上的 AHB_I 與 AHB_D 同時搶奪匯流排使用權造成的影響，且加上 DMA Device 同時競爭匯流排時，在某些時間點上更達到 1 左右。在架構一可以很明顯的看到造成系統效能的瓶頸所在在於下面二點：

1.ARM926EJS 上的指令與資料 master 造成的匯流排衝突。

2.ARM926EJS 與 DMA Device 造成的匯流排衝突。

架構二的匯流排衝突狀態如下圖 12，此圖為量測在架構二上 OuputStage1 上多重匯流排上匯流排上匯流排的衝突狀態。由於使用了多重匯流排排除了 DMA Device 的競爭影響和架構一的匯流排衝突狀態比較起來，整個 master 等待的數目有顯著的降低現像，平均大概在 0.3 左右，並且可以看到效能的改善使整個模擬時間縮短了不少。架構三的匯流排衝突如下圖 13，此圖為量測在架構三上 OuputStage1 上多重匯流排上匯

流排的衝突狀態。由於都使用到了多重匯流排的架構，因此和架構一比較起來都有顯著的改善。和架構二的匯流排衝突比較起來，master 等待的數目更低，平均都在 0.2 以下，且匯流排衝突發生的時間長度比架構二來的短，在資料傳輸的效能上有更一步的改善。

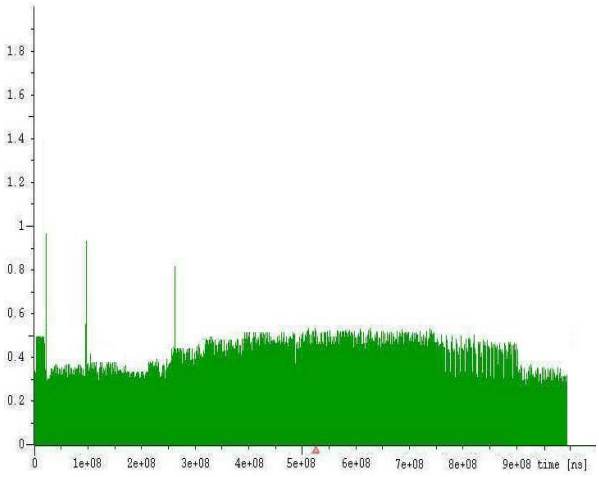


圖 11 架構一的匯流排衝突狀態

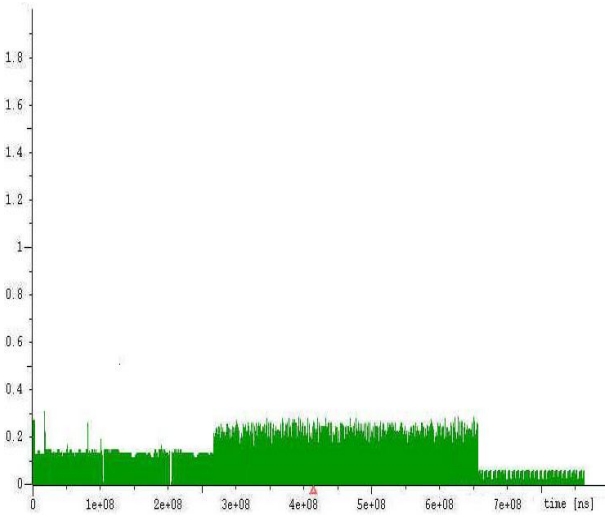


圖 12 架構二上 OuputStage1 的匯流排衝突狀態

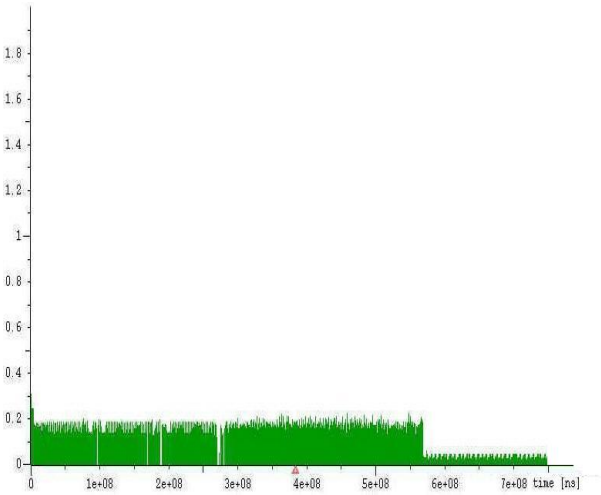


圖 13 架構三上 OuputStage1 的匯流排衝突狀態

4.3 效能分析

實驗結果數據從下面表 1 可以看到在架構一上加入 DCT Device 與沒有架入 DCT Device 的效能比較，由於硬體整體執行速度比 CPU 快很多，因此效能上有 50%以上的改良，且由於 RGB 轉換至 YCbCr 內有浮點運算因此隨著圖檔越大，ARM 將越需要更多時間去做浮點運算的處理，所以效能將隨著圖檔大小越大改良越多。

各圖的表示說明如下：

1. **架構一(SW)：** 表示沒有加入 DCT Device，全部的 JPEG 程式運算皆在 ARM CPU 上運行。
2. **架構一：** 加入 DCT Device。
3. **架構二：** 加入 DCT Device。
4. **架構三：** 加入 DCT Device。
5. 第一欄為輸入的圖檔大小。
6. 第二欄和第三欄為各別架構處理各圖檔所花費的時間，時間單位為(ns)。
7. 第四欄為效能差異性以百分比表示。

實驗數據如下表 2 可以看到由於使用多重匯流排來改善傳輸效能，因此整體效能也將隨著處理的圖檔資料越大越有往上增加的趨勢，這是由於當傳輸的圖檔越大

匯流排衝突發生的機率也將越往上提昇。藉著改善傳輸架構與將處理的資料加大，在數據上可以看到更明顯的效能差異性。實驗數據如下表 3 架構一與架構三的效能比較，由於多使用二級的輸入階，並且將架構 2 中的二個 AHB 匯流排取代之，除了節省一個 AHB 匯流排和一個 AHB_Lite 匯流排的使用，減少了這二個匯流排運行的動作時間，等於也縮短了資料的傳輸路徑，在效能上更為提昇。實驗數據如下表 4 與表 5 無 DCT Device 的架構一與架構三的效能比較，可以看到使用 DCT Device 再加上匯流排架構的傳輸改良，在效能的改良上已經達到接近 70%。

表 1 架構一沒有加入 DCT Device 與加上 DCT Device 的效能比較

圖檔大小	架構一(SW)	架構一	speed up
(80X60)	2.66125E+07	1.26570E+07	**52.44%**
(160X120)	1.02019E+08	4.84736E+07	**52.49%**
(320X240)	4.46420E+08	2.04346E+08	**54.23%**
(640X480)	1.81906E+09	8.03586E+08	**55.82%**

表 2 架構一與架構二的效能比較

圖檔大小	架構一	架構二	speed up
(80X60)	1.26570E+07	1.15038E+07	**9.11%**
(160X120)	4.84736E+07	4.28074E+07	**11.69%**
(320X240)	2.04346E+08	1.77070E+08	**13.35%**
(640X480)	8.03586E+08	6.79858E+08	**15.40%**

表 3 架構一與架構三的效能比較

圖檔大小	架構一	架構三	speed up
(80X60)	1.26570E+07	1.07685E+07	**14.92%**
(160X120)	4.84736E+07	4.01507E+07	**17.17%**
(320X240)	2.04346E+08	1.66183E+08	**18.68%**
(640X480)	8.03586E+08	6.36502E+08	**20.79%**

表 4 無 DCT Device 的架構一與架構二的效能比較

圖檔大小	架構一(SW)	架構二	speed up
(80X60)	2.66125E+07	1.15038E+07	**56.77%**
(160X120)	1.02019E+08	4.28074E+07	**58.04%**
(320X240)	4.46420E+08	1.77070E+08	**60.34%**
(640X480)	2.06870E+09	6.79858E+08	**67.14%**

表 5 無 DCT Device 的架構一與架構三的效能比較

圖檔大小	架構一(SW)	架構三	speed up
(80X60)	2.66125E+07	1.07685E+07	**59.54%**
(160X120)	1.02019E+08	4.01507E+07	**60.64%**
(320X240)	4.46420E+08	1.66183E+08	**62.77%**
(640X480)	2.06870E+09	6.36502E+08	**69.23%**

5.結論

在本論文中，實驗採用了 SystemC 硬體描述語言遵循 TLM 準則實作了所設計的硬體 IP，在 AMBA 匯流排上元件間的通訊也是模組化在交易抽象層級，因此僅需要使用所提供的高層級 API 函數，不需要瞭解內部實作上的細節，這提供了一個好處，高抽象層級便得系統開發人員可以更快速的進行模擬並且更容易的修改架構上的優劣點。論文中使用了電子系統層級工具 Coware 的 ConvergenSC 在交易層級建構軟硬體整合的虛擬平臺，論文中建立了三個架構。第一個架構為嵌入式系統基本型式的平臺，缺點在太多的硬體主埠競爭搶奪匯流排的使用權因此造成大量的匯流排衝突，這將造成各個元件間的傳輸效能降低。第二個架構在第一個架構改良上主要為 AHB 匯流排使用一個多重匯流排取代之，使用多重匯流排的好處在於允許多個主埠對於一個共用的副埠達成平行通

訊的效果。這可以使得匯流排衝突的比例降低，達成資料傳輸效能上的改善。第三個架構在匯流排上有更進一步的改良，將多重匯流排使用的階數升高，和第二個架構比起來節省了一個 AHB 匯流排和一個 AHBLite 匯流，減少了資料的傳輸路徑並且更加提升了傳輸效能。

在未來的工作上，由於每個元件放置的位置不同可能對效能上有不同的影響，這取決於系統上的應用如何設計，因此可以再更一步的去建構不同的架構，比較效能找出最好的架構。ConvergenSC 提供對於其他 EDA 工具的連結介面，因此可以更一步的在這些架構上進行面積與能量消耗的分析，也可以透過其他協力廠商開發的工具使用 HDL 在 RTL 層級進行模擬驗證。

誌謝：本研究為國科會計畫 NSC 95-2221-E-002 -100 -MY3 所支持。

6.參考文獻

[1]大村正之，深山正幸 原著，溫榮弘 編譯，"C/C++ VLSI 設計"， 全華科技圖書股份有限公司，2005.

[2]Axys Design, "http://www.axysdesign.com"

[3]A. P. Su, and R. Chen, "Applying ESL in A Dual-Core SoC Platform Designing," International SOC Conference, 2006 IEEE, 2006.

[4]AMBA,"http://www.arm.com/products/solutions/AMBAHomePage.html"

[5]ADS, "http://www.arm.com/products/DevTools/ADS.html"

[6]Alan P. Su, Chris Lennard, and Peter Grun, "A Finite State Machine Formalization for Transaction Level Modeling," Industrial Technology Research Institute, ARM Ltd.

[7]CoWare, Inc, "http://www.coware.com"

[8]D. Densmore, R. Passerone, and A. Sangiovanni-Vincentelli, "A Platform-Based Taxonomy for ESL Design," IEEE Design and Test of Computers, 2006.

[9]E. D. Lagnese, and D. E. Thomas , "Architectural Partitioning for System Level Design," Design Automation, 1989. 26th Conference on, 1989.

[10]GNU, "http://www.gnu.org/"

[11]JPEG-6B, "http://www.ijg.org/"

[12]Kurt Keutzer, Sharad Malik, Richard Newton, Jan Rabaey, and Alberto Sangiovanni-Vincentelli, "System-level design:orthogonalization of concerns and platform-based design," IEEE Trans. Computer-Aided Design, vol. 19, pp. 1523-1543, Dec. 2000.

[13]Lukai Cai, and Daniel Gajski, "Transaction Level Modeling: An Overview," Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, October 01-03, 2003.

[14]L. Benini, D. Bertozzi, D. Bruni, N. Drago, F. Fummi, and M. Poncino, "SystemC Co-simulation and Emulation of Multi-Processor SoC Designs," IEEE Computer, Vol. 36, No. 4, April 2003.

[15]L. Lee, H. Kim, P. Yang, S. Yoo, E.Y. Chung, K.M. Choi, J.T. Kong, and S.K. Eo, "PowerViP: Soc power estimation framework at transaction level ," Proceedings of the 2006 conference on Asia South Pacific, 2006.

[16]Microsoft, "http://www.microsoft.com/en/us/default.aspx"

[17]M. Caldari, M. Conti, M. Coppola, S. Curaba, L. Pieralisi, and C. Turchetti, “Transaction-Level Models for AMBA Bus Architecture Using SystemC 2.0,” Proc. DATE 2003.
[18]M. Caldari, M. Conti, M. Coppola, P. Crippa, S. Orcioni, L .Pieralisi, and C. Turchetti, “System-Level Power Analysis Methodology Applied to the AMBA AHB Bus,” Proc. Design Automation & Test Europe Conference, 2003.
[19]O. Ogawa, S. Bayon de Noyer, P. Chauvet, and K. Shinohara, “A practical approach for bus architecture optimization at transaction level,” Proceedings of the conference on Design, Automation and Test, 2003.
[20]P. Eles, Z. Peng, K. Kuchcinski, and A. Doboli, “System Level Hardware/Software Partitioning Based on Simulated Annealing and Tabu Search,” Design Automation for Embedded Systems, 1997.
[21]P. R. Panda , “SystemC-A modeling platform supporting multiple design abstractions,” System Synthesis, 2001. Proceedings. The 14th International Symposium on Volume, Issue, pp. 75-80. 2001.
[22]P. Schaumont, and I. Verbauwhede, “a component-based design environment for esl design,” IEEE Design & Test, 2006.
[23] PPM Format, “http://www.physics.emory.edu/~weeks/graphics/mkppm.html”
[24]Robert C. Chen, and Alan P. Su, “Construct A PAC PMP SoC Verification Platform Using ESL Design Methodology,” 66. 系統晶片 002 期, 2006.
[25]SystemC, “http://www.systemc.org/”
[26]T. Hollstein, J. Becker, and A. Kirschbaum, HiPART: A New Hierarchical Semi-Interactive HW/SW Partitioning Approach with Fast Debugging for Real-Time Embedded Systems, 6th International Workshop on Hardware/Software Codesign, 1998.
[27]T. Lei, Y. Yanhui, and W. Shaojun, “Optimizing SoC Platform Architecture for Multimedia Applications,” ASIC, 2005. ASICON 2005. 6th International Conference On, 2005.
[28]V. Reyes, W. Kruijtzer, T. Bautista, G. Alkadi, and A. Nunez, “A unified system-level modeling and simulation environment for MPSoC design: MPEG-4 decoder case study,” Proceedings of the conference on Design, automation and testing Europe: Proceedings, 2006.
[29]W. Mueller, A. Rosti, S. Bocchio, E. Riccobene, P. Scandurra, W. Dehaene, and Y. Vanderperren, “UML for ESL Design? Basic Principles, Tools, and Applications,” ICCAD'06, November 5-9, 2006.