使用 JAVA 與 UML 來實現可重組式運算系統的軟硬體設計

曾志豪、熊博安

國立中正大學資訊工程學系 嵌入式系統實驗室

一、簡介

可重組式運算系統（Reconfigurable Computing Systems）可在系統運作時，藉由軟體的控制來動態的變更其硬體電路，如此一來單一硬體晶片所提供的功能將更具有彈性，原本需要花費大量微處理器運算時間的運算工作，可大幅度地轉移至硬體來加速執行，使得整體系統的效能直逼 ASIC，同時具有使用微處理器的彈性。可重組式運算系統的硬體架構與一般使用微處理器系統的差異，在於多了可重覆規劃電路組態的晶片(例如：現場可規劃閘陣列, Field Programmable Gate Array, FPGA)，所以在設計可重組式運算系統時，除了要撰寫軟體程式碼，也需決定那些運算工作要由硬體來執行，並使用硬體描述語言（例如：Verilog HDL 與 VHDL）來實作硬體的部份。可重組式運算系統之所以尚未大量普及，其原因就是在於設計此一軟硬體混合系統的高複雜度與困難度，並缺乏成熟便利的開發環境與工具。同時具有軟硬體設計能力的人材難尋，一般的軟體工程師即使知道系統中那一部份的程式碼是效能瓶頸的所在，還需藉由硬體工程師來將該部份以硬體來實作，此外，還需處理複雜的軟硬體溝通介面問題。軟體設計人材一般多於硬體設計人材，若是開發出一套工具，使得單靠軟體工程師就可以設計出可重組式運算系統，將可使得可重組式運算系統的使用更加的便利與普及。

本文提出一種可重組式運算系統的設計流程，使得軟體工程師只要使用 UML 來建立系統的模型，並使用 JAVA 來撰寫整個系統，之後藉由本設計流程與相關工具的輔助，就可以自動的合成出所需硬體的 Verilog HDL 程式，以及軟硬體的介面。如此一來將可大幅減少設計可重組式運算系統的複雜度與困難度，使得一般的軟體工程師，也可以設計出簡易的可重組式運算系統。

二、可重組式運算系統設計流程

本文所提出的可重組式運算系統設計流程如圖 1 所示，本流程主要分為三大部份，分別是系統軟體模型建立與實作、軟體合成與硬體合成等三大部份。

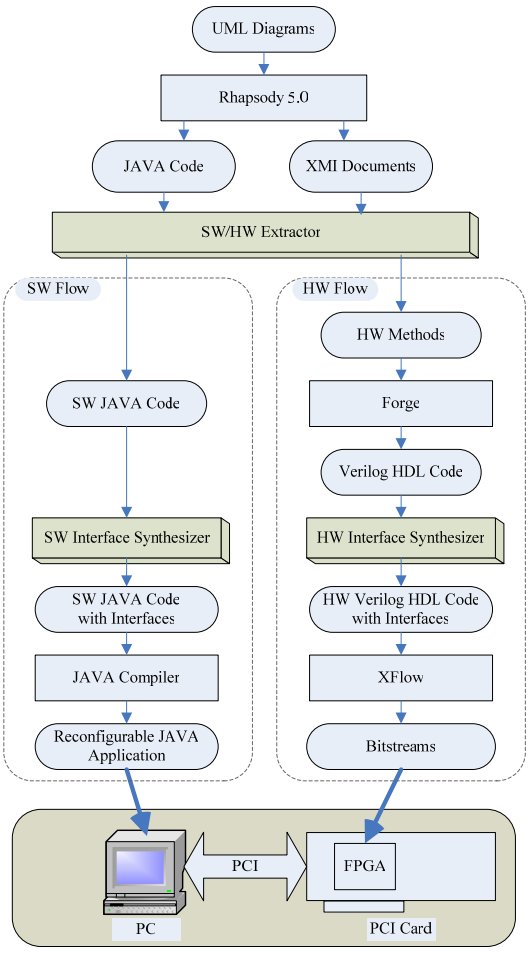

圖 1. 可重組式運算系統設計流程

甲、系統軟體模型建立與實作

系統軟體模型建立與實作，指的是使用者以統一塑模語言（Unified Modeling Language, UML）進行系統的分析與設計，也就是繪製出整個系統的軟體藍圖，之後再以 JAVA 語言來完成細部的程式設計，並“執行”整個系統模型，來進行測試與除錯的工作，最後藉由 Rhapsody 工具產生 JAVA 程式碼（Java Code）與模型資料（XMI Document）。

即將成為官方標準的 UML 2.0 共包含十三種圖表，可依據結構與行為這兩種不同的觀點，將這些圖表分為結構類圖表（Structural Diagrams）與行為類圖表（Behavioral Diagrams）。我們選用結構類圖表中的類別圖（Class Diagrams）來描述整個系統的軟體架構，在行為類圖表中，我們則是選擇了狀態圖（State Machine Diagrams）與順序圖（Sequence Diagrams）來分別描述各個類別的狀態轉移與行為，以及類別之間的互動順序。

圖 2 是一個數位相機的類別圖實例，在類別圖中我們可以很清楚地表示出系統中所包含的類別（Class），以及類別之間的各種關係。圖中共包含 ccd、ccdpp、codec、cntrl 與 uat 等五個類別（Class），ccd 類別具有 rowIndex…等屬性（Attribute）以及 CcdInitialize…等操作（Operation），類別之間的箭頭代表了彼此間的關連（Association）。

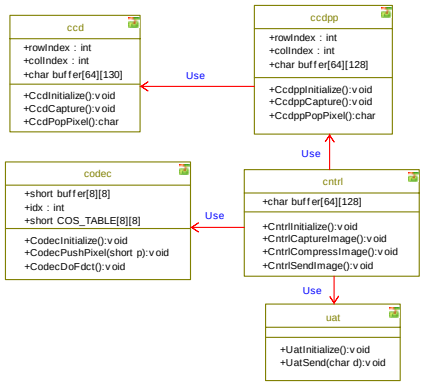

圖 2. 數位相機的類別圖實例

圖 3 是 cntrl 類別的狀態圖，從狀態圖可以看出一個類別被實體化成物件（Object）後，該物件會對那些事件有所動作與轉移至何種狀態。 cntrl 類別控制了整個數位相機的運作流程，由圖中可知當此類別被產生後會先呼叫 CntrlInitialize 函式進行初使化的動作，完畢後進入 Initialized 狀態，代表 cntrl 物件初使化完成。接著會再呼叫 CntrlCaptureImage 函式來捕捉一張影像，完畢後進入 ImageCaptured 狀態，代表影像捕捉完成。然後再呼叫 CntrlCompressImage 函式將影像進行 JPEG 壓縮，完畢後進入 ImageCompressed 狀態，代表影像壓縮完成。之後再呼叫 CntrlSendImage 函式將影像傳送到個人電腦，完畢後進入 ImageSent 狀態，代表影像傳送完成。最後則是進入以兩個同心圓所代表的最終狀態，表示物件已被刪除。

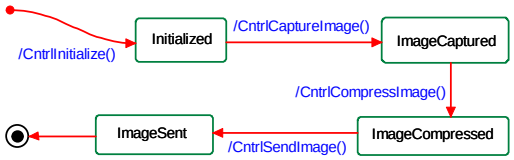

圖 3. cntrl 的狀態圖實例

圖 4 是一個數位相機的順序圖實例，從順序圖可以看出不同類別或物件之間的互動方式與時間順序。順序圖的時間順序是以由上而下的方式來表示時間的進行順序，圖中左上角有 ref 字樣的長條方塊是 UML 2.0 中所擴增的元素，表示該部份參考 Initialize_all 這張順序圖。

在順序圖中使用者可以自行決定要將那些方法(Method)或類別交由硬體來加速執行，也就是使用 UML 的 Stereotype（例如：<<HW>>）將要由硬體來實作的方法或類別標示出來。一般來講，都是將需要耗費大量微處理器計算時間的方法或類別實作成硬體，例如：資料壓縮（JPEG、MPEG）與加解密演算法（DES、RSA）。若是要將多個類別實作成硬體，但受限於硬體資源（FPGA 可用空間）無法一次將所有類別實作成硬體，則需分析順序圖中類別之間的活動

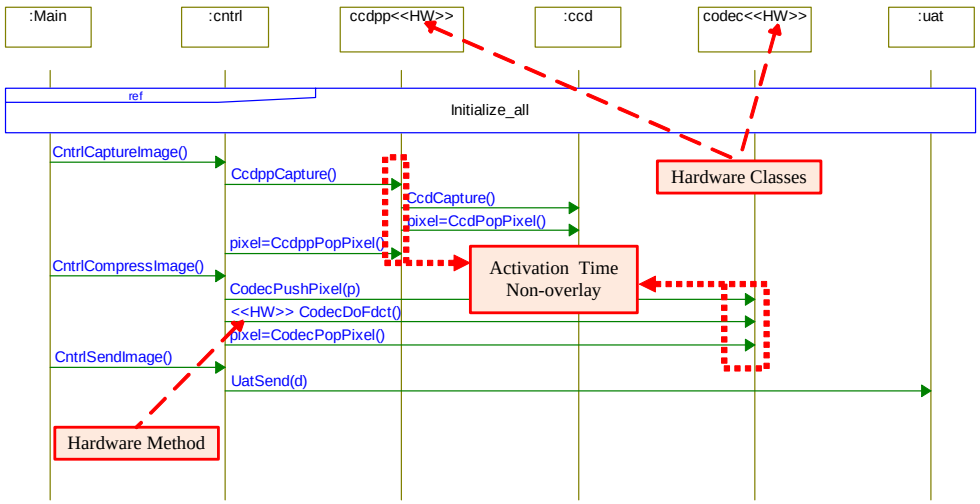

圖4. 數位相機的順序圖實例

時間（ Activation Time）是否有重疊（Overlay）。以圖中的 ccdpp 與 codec 類別為例，其生命周期沒有重疊，所以可以將兩者都實作成硬體，再依時間先後載入到 FPGA 上執行。

在系統軟體模型建立與實作的階段中，我們採用 I-Logix 公司的 Rhapsody 5.0 作為 UML 的塑模工具，Rhapsody 是一個功能強大的 UML 塑模工具，除了支援 UML 2.0 外，其內建物件執行框架（Object Execution Framework, OXF），提供了事件產生、訊息傳遞與計數器等功能，搭配其程式碼產生（Code Generation）功能可以產生可執行的程式碼（C/C++、JAVA、Ada）。所以當我們在 Rhapsody 中繪製完成系統的類別圖、狀態圖與順序圖後，我們只要使用 JAVA 語言來實作系統的細部功能，之後藉由 Rhapsody 的程式碼自動產生功能，產生可執行的程式碼，最後再搭配 Rhapsody 的動畫（Animation）功能，讓我們可以單步執行程式碼，並在狀態圖與順序圖中觀察目前的執行過程，讓我們可以“執行”整個系統模型，使得系統的測試與除錯效率大幅度地提昇。

在系統軟體模型建立與實作階段中，我們主要使用 Rhapsody 來進行 UML 圖表的繪製、JAVA 程式的撰寫，以及系統的測試與除錯工作。在驗證整個系 r 統的正確性後，我們就可以產生第一階段的 JAVA 程式碼（Java Code），

以及藉由 Rhapsody 所提供的 XMI Toolkit 來產生包含 UML 圖形資料的 XMI Document。

SW/HW Extractor 是我們自行撰寫的程式，其目的是為了從 JAVA 程式碼（Java Code）中將要由硬體來實作的方法之程式碼分離出來。為了知道那些方法被標示為由硬體實作，SW/HW Extractor 需要讀取 XMI Document 中的 UML 圖形資料，XMI Document 是以 XML 的格式來記錄 UML 圖形資料，所以可以很容易的解讀其內容。SW/HW Extractor 處理後的輸出分別是 SW JAVA Code 與 HW Methods，整個可重組式運算系統的設計流程在這個階段之後，開始分支成軟體合成與硬體合成流程。

乙、軟體合成

軟體合成流程主要的目的就是合成出可以呼叫硬體方法，並可以在個人電腦上執行的 JAVA 應用程式。軟體合成流程的輸入來自於經由 SW/HW Extractor 處理後的 SW JAVA Code，將其輸入給 SW Interface Synthesizer 後，合成所需的軟硬體溝通介面，就可以產生具有呼叫硬體方法能力的 SW JAVA Code With Interfaces，最後再經由 JAVA Compiler（例如：JDK）編譯後，就成了可以在個人電腦上執行的可重組式

JAVA 應用程式（Reconfigurable JAVA Application）。

SW Interface Synthesizer 是我們自行撰寫的程式，其主要功能就是合成出可呼叫硬體方法並讀回回傳值的 SW JAVA Code With Interfaces。SW Interface Synthesizer 在執行時，會搜尋 SW JAVA Code 中的程式碼，以圖 5 為例，在搜尋到被標示為由硬體實作的方法後，將會對該程式碼進行下列的動作：

1. 在原軟體方法呼叫前，加入載入該硬體電路組態的函數呼叫。
2. 在原軟體方法呼叫後，加入傳遞參數給硬體方法的函數呼叫。
3. 接著加入讀取硬體方法回傳值的函數呼叫。

```
.
.
.
loadHwMethod("codecDoFdct.bit");   // Step1
//count = codecDoFdct("2D");       //Original software method
writeHwRegister(1, "2D");          //Step2
count = readHwRegister(2);         //Step3
.
.
.
```

圖 5. 軟體介面合成實例

在經由上述的處理後，SW Interface Synthesizer 所輸出的 SW JAVA Code With Interfaces 就是具有硬體方法溝通介面的 JAVA 程式碼。

丙、硬體合成

硬體合成流程主要的目的就是合成出可以接受JAVA應用程式呼叫，執行後回傳資料給JAVA應用程式的硬體方法電路組態。硬體合成流程的輸入來自於經由SW/HW Extractor處理後的HW Methods，將其輸入給Forge後，就可以將JAVA程式碼轉換成描述硬體電路設計的Verilog HDL Code。之後再經由HW Interface Synthesizer修改其介面，使得該硬體電路具有與軟體互動的能力，最後藉由XFlow工具自動地呼叫合成工具，產生可在FPGA上運作的硬體電路組態。

Forge是Xilinx公司所提供的高階語言編譯器，可以將JAVA程式碼轉換成Verilog硬體描述語言，Forge並具有迴圈展開（Loop Unrolling）的能力，可使轉換出來的硬體電路效能最佳化，藉由Forge的輔助自動地將軟體轉成硬體，其優點是軟體工程師可以省去硬體電路設計的工作，然而其缺點就是並非所有JAVA的程式碼Forge都有辦法處理，這在使用上是特別需要注意的地方。

Forge轉換出的Verilog HDL Code還需經由我們所自行設計的HW Interface Synthesizer來修改其介面，使得軟硬體間可互相傳遞資料，主要需修改的地方如下：

1. 加入接受軟體呼叫該硬體方法時，參數傳遞所需的暫存器。
2. 加入回傳執行結果所需的暫存器。
3. 加入控制硬體電路執行所需的控制訊息與狀況訊號。

在經由上述的處理後，HW Interface Synthesizer所輸出的HW Verilog HDL Code With Interfaces就是具有可接受軟體呼叫的硬體電路設計。

Xilinx所提供的XFlow工具，可以自動的將HW Verilog HDL Code With Interfaces輸入給Xilinx ISE Foundation來合成出可燒錄至FPGA的硬體電路組態（Bitstreams），藉由這個工具的輔助，軟體工程師可以不用學習硬體開發環境，就可以合成出所需的硬體電路組態。

在硬體合成流程結束後，每個硬體方法都會有一個對映的硬體電路組態檔案（例：JPEG .bit），Reconfigurable JAVA Application 執行時，在呼叫該硬體電路前就會先經由函式呼叫，將所需的硬體電路組態檔案載入到 FPGA 上。

四、執行環境與平台

我們所採用的可重組式運算系統執行環境

與平台，是以個人電腦搭配一塊具有 PCI 介面的 FPGA 開發卡所組成。個人電腦上只要安裝 JAVA 虛擬機器（JAVA Virtual Machine, JVM）就可以執行本設計流程所產生的 JAVA 應用程式。PCI 介面的 FPGA 開發板我們是採用 Xilinx 公司的 XtremeDSP Development Kit，內含 Nallatech 公司的 BenONE PCI 主板與 BenADDA 子板，其組合後如圖 2 所示。在 BenADDA 子板上包含了一個三百萬邏輯閘的 Xilinx Virtex-II FPGA，可以用來載入並執行本設計流程所產生的硬體電路組態檔，此外 BenONE PCI 主板上也包含了一個 Xilinx Spartan-II FPGA，並預先載入 32-bit/33MHz PCI 與 USB 1.1 的韌體，讓使用者省去設計 PCI 溝通介面的時間。

圖 6. XtremeDSP Development Kit PCI 卡
（圖片來源：www.xilinx.co.jp）

以個人電腦搭配一塊具有 PCI 介面的 FPGA 開發卡所組成的可重組式運算系統執行環境，其優點是軟體應用程式是在個人電腦上執行，所以在微處理器、作業系統與 JVM 的選擇上較無限制，反之，若是在 FPGA 上執行軟體應用程式，則該 FPGA 需具有嵌入式微處理器（例如：ARM9、PowerPC、Nios 與 Microblaze），並且所採用的作業系統與 JVM 也需支援該微處理器。在缺點方面，由於此種架構中個人電腦上微處理器與 FPGA 的溝通是經由 PCI 介面，所以在傳輸速度上會比具有嵌入式微處理器的 FPGA 來的慢。

五、結論

可重組式運算系統的研究從 1960 年代至今以歷經了四十多年，之所以尚未被廣泛的應用除了硬體平台的限制外，另外一個因素就是缺乏成熟便利的開發環境與工具來克服設計可重組式運算系統的複雜度與困難度。JAVA 手機已隨處可見，JAVA 讓手幾的軟體功能更加的多樣化，隨著大容量、低耗電與低成本 FPGA 的研發，在未來手機將內含 FPGA，構成可重組式運算系統，使得手機的硬體功能也更加的強大與多樣化，屆時，便利的可重組式運算系統開發環境與工具將扮演關鍵性的角色。本文所提出的可重組式運算系統設計流程，希望能讓一般的軟體工程師，也可以設計出簡易的可重組式運算系統，使得可重組式運算系統廣泛地普及，讓人類更加享受科技所帶來的便利。

六、參考書目

F. Vahid and T. Givargis, Embedded System Design: A Unified Hardware/Software Introduction, John Wiley & Sons, 2002.