可重組系統之通用型硬體內文儲存設計

李宗演 曾筱君 胡哲郝 蔡加春
國立台北科技大學 國立台北科技大學 國立台北科技大學 南華大學
tylee@ntut.edu.tw t7418016@ntut.edu.tw s4419010@ntut.edu.tw chun@mail.nhu.edu.tw

摘要

目前動態可重組 FPGA 系統架構中，在多重任務模組做硬體切換時，大部份的研究只針對單一版本硬體內文儲存做處理，而難以廣泛應用。因此本論文提出一個通用型硬體內文儲存方法，其中建立模組資料庫，並以選擇性 Frame 回讀機制記錄使用的硬體內文資料，所使用的內文儲存位元格式，可以廣泛儲存適用於 Xilinx Virtex-2、Virtex-4 和 Virtex-5 硬體架構，可以減少儲存模組時所重覆記錄的 Frame 位址和位元索引參數。整體而言，以通用型的硬體內文儲存方法，可降低 55.51%的記憶空間及 57.37%的指令空間。

關鍵字：動態部份可重組、硬體內文、FPGA、回讀

Abstract

Nowadays, the ability of dynamic reconfigurable in FPGA architecture has gain its importance while switching on hardware modules. But for recent researches on reconfigurable architecture, most of them aims at one specific version. It's inconvenient by adding additional hardware circuits to apply on all the versions. Therefore, we propose an adaptive hardware context-switching approach for reconfigurable systems. Accompany with the general bit saving format, it suits all types of Xilinx Virtex-2, Virtex-4 and Virtex-5. Which can reduces the saving number of saved frame address and bit-index. Overall, it reduces 55.51% in memory space and 57.37% in instruction space, by using proposed the adaptive hardware context saving method.

Keywords: dynamic self reconfigurable system, hardware context, FPGA, readback

---

本論文之部份成果由國科會計畫編號 NSC 98-2221-E-027-071 所贊助。

1. 簡介

隨著現今科技技術的進步，半導體製程技術上的精益求精下，FPGA（Field Programmable Gate Array）的應用及電路設計更趨於廣泛，在 Xilinx 公司所開發的一系列 Virtex FPGA 中，Virtex-Ⅱ及以上系列的產品均能夠在系統執行時間(Run-Time)內，在不暫停或停止全面裝置的執行前提下，針對使用者的要求或任務的高優先權處理硬體之切換，此功能稱為動態部分可重組(Dynamic Partial Reconfiguration, DPR)。但任務移入移出後，其重新載入整體任務及硬體模組資訊，會造成多餘的延遲和浪費記憶體空間。文獻[1]為了達到部分重新配置的能力，在狀態暫存器外加一個讀/寫介面，和一個溝通基本結構（Communication Infrastructure），均是做為任務的狀態讀取與復原，而導致需要在既有的硬體架構上外加硬體電路，不僅受限於 Xilinx Virtex 各版本的既有硬體電路，也難以擴充應用在不同版本上，造成實現上的不易以及成本的增加。

又由於 Xilinx 發展的多個 Virtex 系列，雖然各個版本的可重組能力均有其特色，但其可重組硬體架構均有所差異，在目前的文獻中大部分僅針對某一特定版本進行可重組能力的分析與改進。

因此，本論文提出一個通用型內文儲存架構，首先提出內文儲存及恢復的方式為選擇性 Frame 回讀機制，在任務重新載入硬體模組時，僅針對已存資料的暫存器讀取整體模組資訊，避免讀取資訊耗費過多回讀的時間；其次，整合不同 Virtex 系列，將 Virtex-2, Virtex-4

和 Virtex-5 的可重組硬體架構資訊建立模組資料庫，當使用者在選用 FPGA 各型號時，方便及時找出相對應的內文儲存架構，在特定規劃的可重組區域進行模組可重組功能執行，以快速的載入內文架構資料庫中，可以節省重新計算架構資料所耗費的時間及儲存的記憶體。

2. 相關文獻探討

在目前 FPGA 的發展技術中，在執行時間的同時可以針對各個不同區塊執行動態及可重組的功能，並在不影響或不暫停全部裝置的前提下，更新部分執行區塊的執行資訊，以提供更快速和更彈性的可重組能力。在基於執行時間中的可重組區域（RTR, Run-Time Reconfiguration），以一個欲設計的模組區塊做為基本的可重組區塊，並針對此區塊 HDL 程式碼區分為靜態的和動態兩種區塊，兩者的資訊藉由匯流排巨集（Bus marcros）來溝通，並用 Frame 位址指示位於 FPGA 內固定的位置。在文獻[2]中則是在 Virtex-5 的架構下定義其執行時間中的可重組區塊（RTR Grid），定義區塊內框架（Framework）的的建立流程，結合配置（Place）模組的方法，分別以中心為環狀（Circular）的方式配置模組和底部由上（Bottom-up）的方式配置模組，比較繞線（Route）的情形。最後，分別在 RTR 的設計前提下，比較經過限制配置模組及未限制配置模組的兩個方法，發現未限制配置模組的方法，其繞線的延遲較大，用以做為在設計 FPGA 可重組區塊時，配置模組及最佳化繞線程度的設計考量。

文獻[3]則是在 Spartan 3 FPGA 的內部可規劃及可回讀（readback）架構下，利用內部 JCAP（JTAG Configuration Access Port）核心控制外部 JTAG 介面，並利用 JCAP 核心執行自我可重組的功能。但由於 Spartan 3 架構下執行部分可重組區塊，針對部分的區塊其時間（Clock）的統一度難以維持，因此在 Virtex-2 架構下，藉由內部規劃資訊存取埠（ICAP, Internal Configuration Access Port）介面，執行讀取修改後寫回（Read-Modify-Write-Back）。其缺點是在在 JTAG 的介面中以 CFG_OUT 的指令寫入後，資料暫存器（Data Register）必須以一個包含回讀指令的填補框架（pad frame）寫入，因此在可重組化的應用上缺乏靈活性。

在文獻[4]提出在即時執行系統（RTOS, Real-Time Operating System）下，為了減少可重組硬體在內文交換（context switching）和軟體中斷（software interrupt）所產生的多餘延遲（overhead），因此提出一個改善的 RTOS 硬體架構，針對 FPGA 在執行可重組過程時，中央處理器（CPU, Central Processing Unit）選擇最佳化暫存器的數目，用以在規劃區塊資訊時，避免過多相同的資訊，以減少暫存器的使用數目。為避免內文交換時執行任務的移出和移入，造成過多的延遲，因此重新初始化中央處理器的暫存器，並將暫存的最後狀態重新儲存回暫存器，在事先分析欲執行的任務內文，用以選擇最少數目的暫存器，將會大幅減少延遲的時間，但難以預測事先分析所耗費的時間；而對於軟體中斷過程，外加的硬體部分可以做為欲直接連接或中斷的轉換器，亦可在記憶體中找到預先儲存的資料值做為控制訊號，並應用於外加的硬體中。文獻[6]則是在針對 Virtex-2 XC2V100 提出硬體回存內文的方法，以分析其硬體結構後所精簡的位元儲存格式，以加速在硬體模組內可重組功能的切換時間。

3. FPGA 基本可重組硬體內文儲存架構

動態可重組 FPGA 架構與靜態重組架構 FPGA 的差異在於靜態配置僅能一次將全部的設計模組載入 FPGA 上，若是欲修改或增加功能至已載入的設計模組，則必須更改所有的硬

體電路設計而後重新載入 FPGA 中，造成整體任務執行上的不便，已不適用於現今多重任務分工的需求。因此利用現今 FPGA 的動態可重組架構實現可重組功能，可快速且重複使用硬體電路，在短時間內允許多個設計模組在 FPGA 裝置上同時執行，以減少裝置資源上的浪費及能源消耗。

在 FPGA 可重組化元件中，最小可定址的配置記憶體單元為 Frame，其包含可重組化的資訊，依照不同版本的可重組硬體架構被建立在各個 Column 中，Virtex 系列的版子可透過虛擬配置埠介面 JTAG（Joint Test Action Group）和外部並行介面 SelectMap，傳送重組後的資訊以利配置；其中 JTAG 利用單一位元序列（bit-serial）傳送重組化後資訊，SelctMap 則可以雙向載入連續或不連續的資料。另外一個內部重組資訊配置埠（Internal Configuration Access Port, ICAP），在第一次的初始化重組後即可使用，允許使用者在執行時間時，可以藉由處理重組化 IP，將設計的硬體模組 Bitsream 傳送給 HWICAP。HWICAP 提供介面，以利使用者可以經由 ICAP 讀取或寫入資料至重組記憶體中，並藉由每個 word 讀取或逐一分析單一重組化 Frame。

自我可重組 FPGA 系統架構可分為可重組區域（Reconfigurable Area）及固定區域（Fixed Area）兩個區域，如圖 1 所示。在固定區域中含有 ICAP、記憶體控制單元（Memory Controller）和中央控制程式（Central Controlling Agent），在此區域內中央控制程式保持運作，並透過 Bus Macro 和可重組區域連結，做為中央控制程式和模組間資料的傳遞；可重組區域內則切割為多個可重組功能模組，以便功能的切換或是新增模組。

3.1 內文儲存流程

首先，經由 VHDL（Very-High-Speed Integrated Circuit Hardware Description Language）描述硬體模組功能，經過 Xilinx 設計工具合成（Synthesis）、轉化（Translate）、對應（Mapping）、配置（Place）和繞線過程後，其程式化後的資料會轉化為相對應的位元資訊（Bitstream），使 FPGA 可以經由寫入 Bitstream 完成配置記憶體的規劃，而後所設計硬體模組即可載入至 FPGA 上應用。

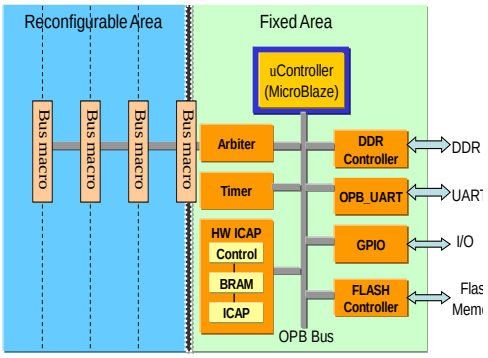

圖1 FPGA 自我可重組架構

當發生中斷或是有新的模組欲取代時，若硬體空間無法及時被釋放，必須切換可重組區域中的硬體模組，因此利用硬體內文動態切換的能力先儲存目前的硬體模組內文資料，等待中斷或新模組工作執行完成後，再回讀並繼續執行未完成的工作。利用 ICAP 介面讀取硬體模組內的資料並儲存，並在控制配置介面進行回讀（Readback），再藉由分析硬體模組內部暫存器和配置記憶體 Frame 位址的對應關係，回讀時不採用儲存完整內文資料，而是選擇性讀回 Frame 的位址，以減少記憶體使用空間。

在有限的可重組區域內，同時間有多個功能必須同時執行時，會依照優先權的高低執行任務，如圖 2 所示，為硬體內文儲存及恢復的流程。當有一優先權較高的任務要求執行時，會先判斷目前 FPGA 上是否有無可用的硬體模組空間，若是有可用空間，則直接載入並執行；若是目前沒有可用空間，則選擇一個硬體模組並中斷其原有的執行任務，在將原有任務由模組移除前，先判斷任務是否需要被回讀；若任務必須被回讀，則儲存原有任務的暫存器

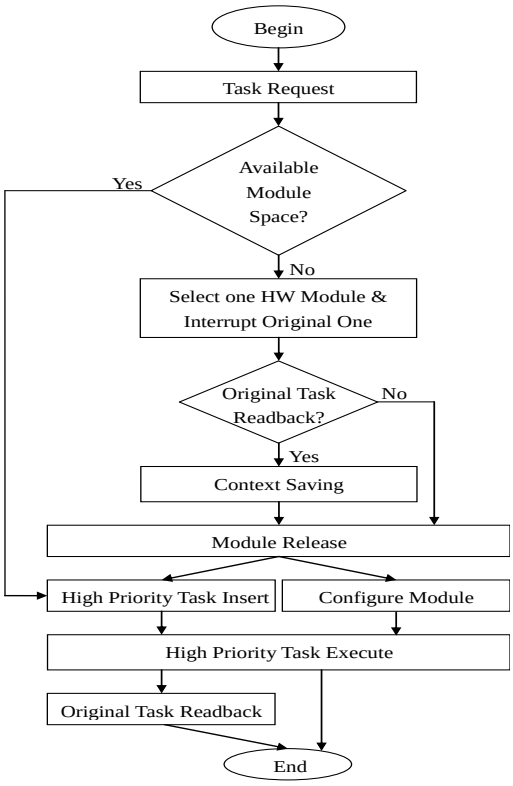

圖2 硬體內文儲存與回讀流程

位址參數和資料，而後釋出其硬體模組空間，使優先權較高的任務移入模組空間內執行，待其執行完成後，再將原先任務的暫存器位址參數從模組資料庫中讀出，將其載入至原硬體模組中執行完成；若是原有任務不需再回讀至硬體模組中，則目前的硬體模組空間可被釋出，而後針對模組做配置（Configuration），載入優先權較高任務執行，直到執行完成。

3.2 資料回讀流程

由於任務移入移出導致 Bitstream 的改變，當任務需要被重新載入至原先執行的硬體模組內，必須將存在暫存器內的資料讀出，稱為回讀（ReadBack），因此控制 ICAP 介面的配置，以預先設置並在回讀程序開始時傳送字元序列的控制指令，可減少回讀指令的重複設定次數及使用指令數目，降低執行複雜度，使載入時僅讀取更改的部分。

首先，送出指令以指示 FPGA 開始回讀，使 FPGA 執行配置記憶體；先使用一個同步字元以同步配置邏輯，而後為了讀取內部暫存器的目前狀態，將目前的值存入配置記憶體中；設置指令，從 FDRO 讀取配置資料及設定為輸出配置資料；設立暫存器，寫入開始讀取的位址；設定所讀的 Word 數量；寫入 NOOP，使封包緩衝器清除；開始讀出配置的資料，並設定指令以非同步 ICAP 介面，清除封包緩衝器。

3.3 通用型硬體內文儲存架構

由於 Xilinx 不同 Virtex 版本間的基本硬體架構差異性大，可重組區域的規劃及框架單位均不同，硬體模組因受限於架構的不同，而有不同的區域規劃方式進行重組功能，此外，ICAP 介面的位元長度亦不同，故當重組功能執行時，會因而產生不同架構的 Frame 位址和位元索引，因此提出一個整合介面，其內含模組資料庫便於使用者在應用 Xilinx 不同版本間的可重組功能時，可快速找出其相對應版本的 Frame 位址和位元索引關係。

整體介面如圖 3 所示。一開始，使用者先選擇欲使用的 Xilinx Virtex 版本，根據其版本的可重組硬體架構，規畫任務模組的可重組區域，不同的模組在經過部分可重組過程後，會產生硬體模組的 Partial Bitstream 以及邏輯配置檔案（Logic Allocation File, .ll 檔），其包含 CLB、Slices、FF、IOB、BRAM 等參數，藉由邏輯配置檔的產生，可分析 Frame 位址和暫存器位元索引的特性，建立模組資料庫。

在建立資料庫時，不儲存所有的 Frame 位址和位元索引參數，減少儲存資料使用的記憶體；模組資料庫一旦建立後，便不需在每一次的模組可重組過程中，重新分析 Partial Bitstream 和邏輯配置檔。尤其是針對回讀模組資料時，將 Partial Bitstream 載入至 FPGA 內部配置記憶體中時，經由模組資料庫中已存的

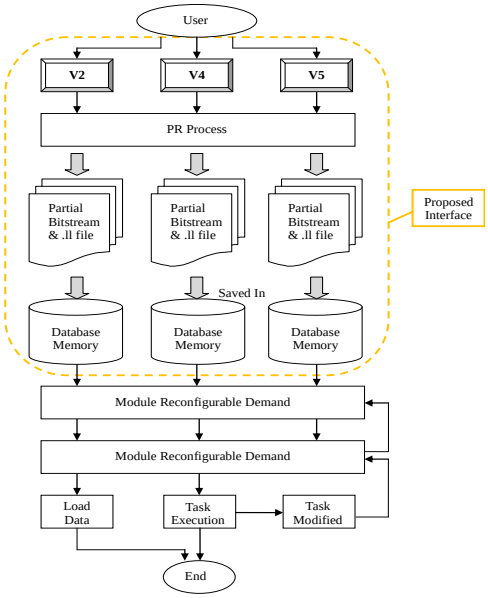

圖3 通用型內文儲存架構之設計

Frame 位址和位元索引，以讀取暫存器內的資料，以重新將資料配置到硬體模組上。因此，不論是硬體模組目前需要進行重組化，或任務切換須藉由回讀讀取硬體內部資料，或新增修改內文資料至硬體模組上，均可以藉由架構資料庫的建立後，快速計算其儲存內文資料的暫存器位址，以快速載入暫存器資料至 FPGA 硬體模組上。

3.4 Virtex-4 XC4VSX35 的內文儲存架構

在 FPGA 的動態可重組系統中，由於 Virtex 各版本間的硬體架構不同，導致其模組資料庫中儲存的硬體模組 Bitstream，和內部暫存器中 Frame 位址和位元索引參數之間的關係，將會因硬體架構的不同而有所不同，以下將針對 Virtex-4 XC4VSX35 的硬體可重組架構分析，以建立適合 Virtex-4 各型號之模組資料庫。

Virtex-4 XC4VSX35 可重組之硬體架構示意圖如圖 4，硬體模組可分為六個區塊，在 XC4VSX35 FPGA 中共有 96 個 Row，40 個 Column，每一個 CLB 由 4 個 Slice 組合，每個 Slice 包含一個 XQ 暫存器和一個 YQ 暫存器；暫存器的資料以一定規則分布在 Frame 的特定位置中。如表 1 和表 2 所示，每 1 個 Column 重覆一組 Frame 位址，另外，每 16 個 Row 重複一組位元索引位址，依 Virtex-4 XC4VSX35 硬體架構可分為上半部和下半部，由 R96 到 R49 為上半部，上半部的暫存器位元索引位址以相差 40 的方式逐一遞減，如表 1 所示；R48 到 R0 則為下半部，下半部的位元索引位址則以相差 40 的方式逐一遞增，如表 2 所示；由於每 16 個 Row 形成一個區塊，每個區塊會重複一組位元索引位址，由上而下共 6 個，上半部和下半部各 3 個，分別依序標記區塊為 Block 0 到 Block 2。綜合以上的規則，歸納出以下 4 個式子計算 XQ 和 YQ 暫存器的位元索引位址的公式。其中式(1)和式(2)為區塊上半部份的位元索引位址計算式，式(3)和式(4)則為區塊下半部份位元索引位址計算式，在式(1)到式(4)中，X_{oe} 是表示 X Column 的佔用情況，指示目前使用的 X Column 為偶數行或奇數行；Q_{xy} 則表示 Slice 中暫存器 XQ 或 YQ 的使用。針對上半部的位元索引位址，當使用偶數的 X Column 時，如 X0，其 $X_{oe} = 0$；當使用奇數的 X Column 時，如 X1，其 $X_{oe} = 1$；當使用暫存器為 XQ 時，其 $Q_{xy} = 28$，當使用暫存器為 YQ 時，其 $Q_{xy} = 0$。而針對下半部的位元索引位址，當使用偶數的

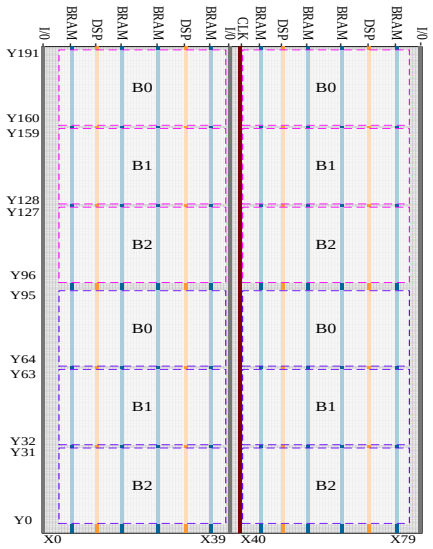

圖4 Virtex-4 XC4VSX35 硬體架構

表1 Virtex-4 XC4VSX35 架構上半部之 Frame 位址和位元索引參數

Frame		C1				C2			
		MJA=3				MJA=4			
		X0		X1		X2		X3	
Index(row)		XQ	YQ	XQ	YQ	XQ	YQ	XQ	YQ
		00008054	00008054	00008054	00008054	00008094	00008094	00008094	00008094
R96	Y191	1278	1306	1277	1305	1278	1306	1277	1305
	Y190	1238	1266	1237	1265	1238	1266	1237	1265
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	Y176	678	706	677	705	678	706	677	705
	Y175	606	634	605	633	606	634	605	633
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
R81	Y161	46	74	45	73	46	74	45	73
	Y160	6	34	5	33	6	34	5	33

表2 Virtex-4 XC4VSX35 架構下半部之 Frame 位址和位元索引參數

Frame		C1				C2			
		MJA=3				MJA=4			
		X0		X1		X2		X3	
Index(row)		XQ	YQ	XQ	YQ	XQ	YQ	XQ	YQ
		00408054	00408054	00408054	00408054	00408094	00408094	00408094	00408094
R16	Y31	33	5	34	6	33	5	34	6
	Y30	73	45	47	46	73	45	74	46
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	Y16	633	605	634	606	633	605	634	606
	Y15	705	677	706	678	705	677	706	678
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
R1	Y1	1265	1237	1266	1238	1265	1237	1266	1238
	Y0	1305	1277	1306	1278	1305	1277	1306	1278

X Column 時，其 X_{oe} = 1；當使用奇數的 X Column 時，如 X1，其 X_{oe} = 0；當使用暫存器為 XQ 時，其 Q_{xy} = 0，當使用暫存器為 YQ 時，其 Q_{xy} = 28。

$$X_{top\_up} = 1306 - 40 \times Para_{top\_up} - X_{oe} - Q_{xy} \quad (1)$$
where, $Para_{top\_up} = 191 - (Y\_row + 32 \times w)$
w is an integer number of block, $0 \le w \le 2$

$$X_{top\_down} = 634 - 40 \times Para_{top\_down} - X_{oe} - Q_{xy} \quad (2)$$
where, $Para_{top\_down} = 175 - (Y\_row + 32 \times w)$

$$X_{bottom\_up} = 34 + 40 \times Para_{bottom\_up} - X_{oe} - Q_{xy} \quad (3)$$
where, $Para_{bottom\_up} = 95 - (Y\_row + 32 \times w)$

$$X_{bottom\_down} = 706 + 40 \times Para_{bottom\_down} - X_{oe} - Q_{xy} \quad (4)$$
where, $Para_{bottom\_down} = 79 - (Y\_row + 32 \times w)$

而在硬體模組佈局後時，針對其佔用的 Frame 和暫存器，先儲存 Frame 位址，再儲存此 Frame 位址下的位元索引參數(Y_row)，可以節省重複儲存及讀取相同 Frame 位址所耗費的記憶體空間，並以最少位元建立模組資料庫中所需的各項參數。

為了在模組資料庫中記錄適用 Virtex-4 各型號的 Frame 位置及位元索引位址，分別各以 13 個位元儲存計算 Frame 位址所需的參數和計算位元索引的位置，如表 3 所示。最高 1 個位元為 *Bit_share_Flag*，用以判斷目前儲存為 Frmae 位址格式或為位元索引位址格式。若是為 Frame 位址格式，則用 7 個位元儲存 *MJA*，其 *MJA* 可用以判斷目前 Frame 位址所佔的 Column 位置，2 個位元儲存 X_{oe}，以記錄在相同 Column 中對 Xeven 和 Xodd 的佔用情況，再以 1 個位元標記目前 Frame 位址位於可重組硬體架構的上半部或下半部，及以 *w* 為 2 個位元記錄區塊號碼；若是針對位元索引位址格式，則用 9 個位元記錄 *Y_row* 的位置，並用 1 個位元記錄 Frame 位址是位於區塊中的上半部或下半部，再用 2 個位元儲存 Q_{xy}，記錄 XQ 及 YQ 的使用情形，其每項通用性參數使用如表 4 所示。

表3 適用於 Virtex-2、Virtex-4 和 Virtex-5 型號之 Frame 位址與位元索引參數

V2	V2-Frame Address and Bit-index														
	Frame_share_Flag		*MJA*							*MNA_flag*	*nFrame*				
	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	0	0	x	x	x	x	x	x	x	x	x	0	0	0	
	Frame_share_Flag		X_{oe}	*Y_row*											
	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	x	x	x	x	x	x	x	x	x	x	x	0	0	0	
V4	V4-Frame Address and Bit-index														
	Frame_share_Flag	*MJA*							*Xoe*		*Top/Bottom*	*w*			
	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	0	x	x	x	x	x	x	x	x	x	x	x	x	0	
	Bit_share_Flag	*Y_row*									*Up/Down*	Q_{xy}			
	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	1	x	x	x	x	x	x	x	x	x	x	x	x	0	
V5	V5-Frame Address and Bit-index														
	Frame_share_Flag	*MJA*							*Xoe*		*Up/Down*	*w*			
	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	0	x	x	x	x	x	x	x	x	x	x	x	x	x	
	Bit_share_Flag	*Y_row*								Q_R					
	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	1	x	x	x	x	x	x	x	x	x	x	x	x	0	

表4 通用性參數

Share_Flag	Description
00	The frame address
01	The frame bit index occupied on the first frame
10	The frame bit index occupied on the second frame
11	The frame bit index occupied on the both frames
0	The frame address
1	The bit index
X_{oe}	Description
0/1	The X Column is in the even/odd
00	The bit index occupied on the $X_{\_even}$ column
01	The bit index occupied on the $X_{\_odd}$ column
11	The bit index occupied on both $X_{\_even}$ and $X_{\_odd}$ column
MNA_flag	Description
0/1	MNA=2/1
nFrame	Description
0/1	Read 2 frames/Read 1 frame
Top/Bottom	Description
0/1	On the Top/Bottom of the reconfigurable hw module
Up/Down	Description
0/1	On the up/down part of the block
w	Description
xx/xxx	The block number
Q_{xy}	Description
00	The bit index occupied on register XQ
01	The bit index occupied on register YQ
11	The bit index occupied on both register XQ and YQ
Q_R	Description
xxxx	The used situation of register AQ、BQ、CQ、DQ

3.5 針對 Virtex-5 XC5VLX50T 的內文儲存架構

同理，分析 Virtex-5 XC5VLX50T 的可重組硬體架構，歸納其 Frame 位址及位元索引位址之間的關係，可得結果如表 5 所示。在表 5 中，每 1 個 CLB 由 8 個 Slice 組合，每個 Slice 包含 4 個暫存器，分別為 AQ、BQ、CQ 和 DQ；每 1 個 Column 會重複一組 Frame 位址，每 20 個 Row 重複一組位元索引位址，其形成一個區塊，硬體架構由上而下可以區分為 6 個區塊，分別標記區塊為 Block0 至 Block5，如圖 5 所示。其位元索引位址在每個區塊中以相差 64 的方式遞減，上下各 10 個 Row 依其規則區分為上半部及下半部，進而歸納出以下 2 式計算位元索引位址，其中式(5)計算區塊內上半部的位元索引位址，式(6)則是計算區塊內下半部的位元索引位址，式中的 X_{oe} 用以表示 X Column 的佔用情況，若為偶數 Column，則 X_{oe}＝0，若為奇數 Column，則 X_{oe}＝1；另外，Q_s 則用以表示暫存器 AQ、BQ、CQ 和 DQ 的佔用情況，若佔用暫存器 AQ，則 Q_s＝

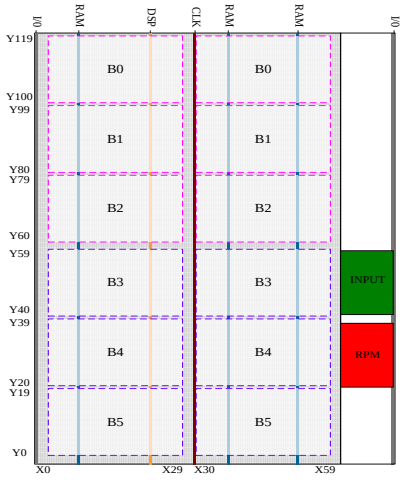

圖5 Virtex-5 XC5VLX50T 硬體架構

表5 Virtex-5 XC5VLX50T 區塊內之 Frame 位址和位元索引參數

Frame	C1							
	MJA=3							
	X0				X1			
Index(row)	AQ	BQ	CQ	DQ	AQ	BQ	CQ	DQ
	0001009F	0001009F	0001009F	0001009F	0001009F	0001009E	0001009F	0001009F
Y119	1251	1276	1288	1310	1250	1275	1287	1309
Y118	1187	1212	1224	1246	1186	1211	1223	1245
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Y110	675	700	712	734	674	699	711	733
Y109	579	604	616	638	578	603	615	637
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Y101	67	92	103	126	66	91	103	125
Y100	3	28	40	62	2	27	39	61

$$X_{up} = 1310 - Para_{up} - X_{oe} - Q_s$$
$$\text{where, } Para_{up} = 64 \times [119 - (Y\_row + 20 \times w)] \quad (5)$$
$$\text{w is an integer number of block, } 0 \le w \le 6$$

$$X_{down} = 638 - Para_{down} - X_{oe} - Q_s$$
$$\text{where, } Para_{down} = 64 \times [109 - (Y\_row + 20 \times w)] \quad (6)$$

59，佔用暫存器 BQ，則 Q_s＝34，佔用暫存器 CQ，則 Q_s＝22，佔用暫存器 AQ，則 Q_s=0。

為了儲存適用於 Virtex-5 所有型號的 Frame 位址和位元索引位置，因此歸納如表 3 的 V5 欄位所示，14 個位元儲存 Frame 位址及位元索引位置計算所需的參數。其中不同於 Virtex-4 儲存位元格式的參數為 Q_R，用於標記暫存器 AQ、BQ、CQ 和 DQ 的使用情形。

最後，改善文獻[6]中所提出針對 Virtex-2 XC2V1000 的 Frame 位址和位元索引參數儲存格式，將其修改為適用於 Virtex-2 各型號的儲存格式，並合併 3 個版本的位元儲存格式，以最大位元數 14 個位元儲存計算 Frame 位址及位元索引位址所需的各個參數，以此型式儲存邏輯配置位址於模組資料庫中。在通用型設計內，當使用者選擇所使用的版本及型號時，以此通用型的位元儲存格式，儲存任務執行時硬體模組佔用的 Frame 位址及暫存器，當任務模組需被回讀時，可以快速在模組資料庫中找到所儲存的 Frame 位址資料。

4. 實驗結果與分析

在本論文的實驗中，我們使用一個七段顯示之上數計數器做為可被動態切換的硬體模組，其計數的狀態由 0 計數至 F，共 16 種狀態，根據不同版本的可重組硬體架構規畫最佳化可重組區域，再藉由邏輯配置檔讀取在模組資料庫內的資料，以通用型設計中提出的通用型位元儲存格式記錄任務模組所佔用的 Frame 位址和暫存器，可減少重複記錄及讀取每 Frame 位址所耗費的記憶體，並針對各版本以及通用型版本所佔用的資料庫記憶空間及指令使用空間分析其效能。

首先，分析任務模組在各版本所使用的記憶使用空間，分別記錄各版本在七段顯示器任務執行下，所使用的 Frame 和暫存器，並以位元儲存格式記錄 Frame 位址和位元索引參數，如表 6 所示，其為通用於 Virtex-4 儲存七段顯示模組的邏輯配置位址，而表 7 所示為通用於

表6 Virtex-4 執行七段顯示上數計數器之模組資料庫

Memory Address	Memory Data (Context Data)	Description
0	00000011111000	Frame address (0x00400054)
1	10010101000110	Slice X0Y84 XQ,YQ
2	10010100110110	Slice X0Y83 XQ,YQ
3	10010100100110	Slice X0Y82 XQ,YQ
4	10010100010110	Slice X0Y81 XQ,YQ
5	10010100000110	Slice X0Y80 XQ,YQ
6	10010011111110	Slice X0Y79 XQ,YQ
7	10010011101110	Slice X0Y78 XQ,YQ
8	10010011011110	Slice X0Y77 XQ,YQ
9	10010011001110	Slice X0Y76 XQ,YQ
10	10010010111110	Slice X0Y75 XQ,YQ
11	10010010101110	Slice X0Y74 XQ,YQ
12	10010010011110	Slice X0Y73 XQ,YQ
13	10010010001110	Slice X0Y72 XQ,YQ
14	10010001111110	Slice X1Y71 XQ,YQ
15	10010001101010	Slice X0Y70 YQ

表7 Virtex-5 執行七段顯示上數計數器之模組資料庫

Memory Address	Memory Data (Context Data)	Description
0	00000011111000	Frame address (0x00001009F)
1	10110010010110	Slice X1Y100 AQ,BQ,CQ,DQ
2	10110010011110	Slice X0Y100 AQ,BQ,CQ,DQ
3	10110010100010	Slice X1Y101 AQ
4	10110010111110	Slice X0Y101 AQ,BQ,CQ,DQ
5	10110011011110	Slice X0Y102 AQ,BQ,CQ,DQ
6	10110011111110	Slice X0Y103 AQ,BQ,CQ,DQ
7	10110100011110	Slice X0Y104 AQ,BQ,CQ,DQ
8	10110100111110	Slice X0Y105 AQ,BQ,CQ,DQ
9	10110101001010	Slice X0Y106 AQ,BQ

表8 Xilinx Virtex 不同版本任務模組硬體切換所佔用之記憶體空間

Xilinx 版本	所需記憶體	減少記憶體 (%)
分別使用 Virtex-2 、Virtex-4、Virtex-5 硬體內文儲存	535-bit (187-bit+208-bit+140-bit)	-
通用型硬體內文儲存	238-bit	55.51

表9 Xilinx Virtex 不同版本任務模組硬體切換所使用之指令空間

Xilinx 版本	所需指令空間	減少指令空間 (%)
分別使用 Virtex-2 、Virtex-4、Virtex-5 硬體內文儲存運算	373-Byte (81-Byte+133-Byte+159-Byte)	-
通用型硬體內文儲存運算	159-Byte	57.37

Virtex-5 儲存七段顯示模組的邏輯配置位址。

就分析資料庫記憶體而言，未採用通用格式儲存任務模組時，各版本任務模組所佔用的記憶空間如表 8 所示。而在模組資料庫中，以最多使用位元數 14 位元儲存通用型儲存格式，並記錄任務模組所佔用的暫存器位址後，如表 8 所示，Virtex-2 使用了 238-bit 記憶空間，Virtex-4 使用了 224-bit 記憶空間，Virtex-5 則使用了 140-bit 記憶空間，以最大記憶空間 238-bit 儲存適用於各版本的任務模組，則通用型硬體內文儲存方法比分別使用 Virtex-2 、Virtex-4、Virtex-5 硬體內文儲存方式可以減少 55.51%的記憶體。

而在分析各版本的指令空間使用，根據各版本在執行和切換任務模組時所佔用的指令空間分析，並取最多指令使用空間做為通用型的指令空間使用，分析結果如表 9 所示，在分析各自版本的指令空間時，Virtex-2 為 81-byte 指令空間，Virtex-4 為 133-byte 指令空間，而 Virtex-5 則為 159-byte 指令空間，若是未採用通用型儲存格式，則三個版本共需 373-byte 儲存所使用的指令空間，採用通用型儲存格式時，則取最大指令空間 159-byte 儲存適用於三個版本的指令空間。通用型硬體內文儲存格式可以比分別使用 Virtex-2 、Virtex-4、Virtex-5 硬體內文儲存格式減少 57.37%指令空間。

5. 結論

Xilinx Virtex 系列的可重組硬體架構其可重組能力及加快模組切換的速度均受限於特定型號或版本，而難以廣泛應用，因此本文提出通用型硬體內文儲存方法，可以應用於 Virtex-2、Virtex-4 和 Virtex-5 三個版本的任務模組所使用的 Frame 位址和位元索引參數，針對各版本的硬體架構特色，儲存任務模組所需的 Frame 位和位元索引參數，減少在模組資料庫記憶體空間，同時可以大幅減少在任務模組重新載入可重組區塊時，因硬體資料回讀而所耗費的重新計算 Frame 位址的時間。在記憶空間及指令空間的效能評估均獲得良好的效果。

參考文獻

[1] M. Koester, M. Porrmann and H. Kalte, "***Relocation and defragmentation for***

heterogeneous reconfigurable system," in *Proc. of the International Conference on Engineering of Reconfigurable Systems and Algorithms*, pp. 70-76, Jun. 26-29, 2006.

[2] J. Strunk, T. Volkman, K. Stephan, W. Rehm and H. Schick, "***Impact of run-time on design and speed – a case study based on a grid of run-time reconfigurable modules insides a FPGA,***" in *Proc. of the IEEE International Symposium on Parallel and Distributed Processing*, pp. 1-8, May 2009.

[3] K. Paulsson, U. Vierck, M. Hubner and J. Becker, "***Exploitation of the external JTAG interface for internally controlled configuration readback and self-reconfiguration of spartan 3 FPGAs,***" in *Proc. of the IEEE Computer Society Annual Symposium on VLSI*, pp. 304-309, April 2008.

[4] M. Song, S. H. Hong, and Y. Chung, "***Reducing the overhead of real-time operating system through reconfigurable hardware,***" in *Proc. of the 10th Euromicro Conference on Digital System Design Architectures*, pp. 311-316, August 2007.

[5] Trong-Yen Lee, Che-Cheng Hu, Li-Wen Lai, Chia-Chun Tsai and Rong-Shue Hsiao, "***Hardware context switching methodology for dynamically partially Reconfigurable Systems,***" in *the Proc. of National Computer Symposium*, pp. 300-310, December 20-21, 2007.

[6] Trong-Yen Lee, Che-Cheng Hu, Yang-Kun Huang and Chia-Chun Tsai, "***A New Approach of Hardware Context-Switch for Dynamic Partial Self-Reconfigurable Systems,***" in *the Proc. of the Conference on the 19th VLSI Design/CAD Symposium*, Aug. 5-8, 2008.