具特殊指令支援的 ARM 處理器之電路設計

A circuit design for an ARM processor with special instructions features

范佐毅
逢甲大學電機系
nickfall.tw@yahoo.com.tw

王晝
逢甲大學電機系
lwang@plum.iecs.fcu.edu.tw

摘要

有鑑於嵌入式處理器在IA發展中的關鍵地位，考慮嵌入式系統的應用方向及設計限制，本研究立下“以最少量的硬體成本，擴充處理器能力以因應未來之應用趨勢”的設計精神，以應用於低階系統的 ARM7TDMI 為研究主體，展開研究與改良設計。我們由計算機結構的技術觀點出發，針對資料加密(Encryption)及多媒體(Multimedia)應用，提出了兩個快速查表指令及兩個 SIMD 式的乘加指令，透過處理器核心的電路設計及修改，最後並以 FPGA 的方式實作驗證，證明了這些附加指令不僅能對加密及多媒體資料處理產生有效的效能改善，且其硬體成本甚低，符合上述最少量硬體成本的設計原則。

關鍵詞：Embedded Processor；Data Encryption；SIMD；FPGA。

I. 前言

資訊家電(Information Appliances, IA)挾著其低價、方便使用、應用多樣化的優勢，目前已日漸取代 PC 的地位而成為資訊應用的主角[6]。有鑑於嵌入式處理器在 IA 發展中的關鍵地位，考慮嵌入式系統的應用方向及設計限制，純粹的 RISC 指令集應有特定的指令加強其處理應用程式的效能，不能凡事就靠加裝副處理器來解決。功能多卻體積龐大、耗電量大的嵌入式處理器，並不適合各項應用的需求。有鑑於以上的觀察，本研究方才立下“以最少量的硬體成本，擴充處理器能力以因應未來之應用趨勢”的設計精神，以應用於低階系統的 ARM7TDMI 為研究主體，展開研究與改良設計，其最終目的就在開發一個與 ARM7 完全相容，且在特定應用領域上可產生相當的效能改善，其硬體成本卻增加不多的 ARM7 系列處理器核心，以因應市場的需求。

我們由計算機結構的技術觀點出發，以 ARM 架構為研究標的，在指令集架構的層次上針對資料加密(Encryption)及多媒體(Multimedia)應用，提出效能改善之研究[5][10]。

1.1 資料加密的指令支援[5]

對加密演算法來說，查表不只可以達到迷惑(Confusion)的特性，更能達到程式最佳化的目的。根據統計[7]，查表的動作在一般加密演算法中佔有相當的比重(23%~72%)，因此如果能針對查表的動作提昇效能，便能對整個演算法的效能有著直接和重大的影響。

本研究針對查表運算之效能提昇提出了新的指令，以期能進一步的提昇加密演算法的效能。而此一新增指令：Load.extract，可將有效位元址的計算合併為一個指令：

```
ldr.nb   Rd,[Ra,Rb]
```

這個指令的動作除了將資料由實際記

憶體位址讀進 Rd 外，最重要的是還包括了有效位址的計算，將 Ra 的某個 Byte 取出，再進行移位，最後加上 Rb 的基底位址，以得到有效位址。

由於加解密演算法中 Load 的資料往往緊接著會作 XOR(eor)運算，如果能與 Load 一同運算，即能進一步提昇加密效能。這個指令的組語格式為：

ldr.nb.eor Rd,[Ra,Rb],Rc

I.2 對多媒體運算核心—乘加指令(MAC)之支援[10]

由於多媒體的資料型態大都是 8 bits pixel、16 bits audio data，在如同 ARM 架構的 32-bits 處理器上，利用 SIMD 的指令可以同時平行處理 4 個 8-bits(packed byte)或是 2 個 16-bits(packed word)的多媒體資料[12]。我們進一步分析多媒體應用程式中的演算法，發現主要都是在做乘 (Multiply)、加(Add) 或是 Multiply-Accumulation (MAC)的動作。

本研究即根據上述的觀察結果，選定將 SIMD Parallel MAC 指令，在不增加太多的硬體負擔下加入傳統 ARM RISC Processor Core 中，藉著增加 SIMD Parallel MAC 指令，使其處理多媒體應用程式的能力更加強，以因應未來嵌入式處理器對於多媒體處理能力的要求[9]，該指令的組語格式為：

PMLAV(cond)(s) Rd,Rm,Rs,Rn

此指令的動作為將 Rs 及 Rm 內各存放的兩筆 16-bits 多媒體資料相乘，而乘積再與 Rn 相加，運算完之結果存入 Rd 內，即完成 PMLAV 指令之動作。另一功能相近的指令為：

PMLA(cond)(s) Rd,Rm,Rs,Rn

其動作為將 Rs 存放的兩筆 16-bits 的多媒體資料與 Rm 存放的單筆 16-bits 的多媒體資料分別相乘，而乘積再與 Rn 相加，運算完之結果存入 Rd 內，即完成 PMLA 指令之動作。

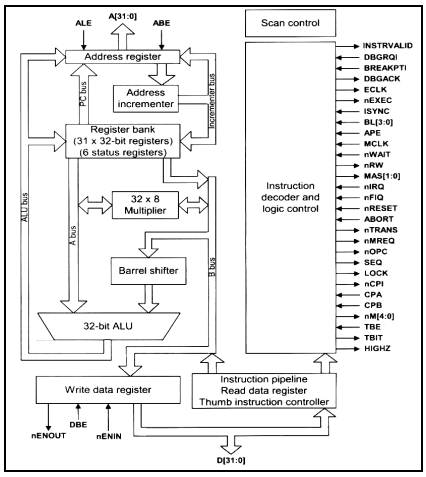
為驗證本研究所提出的改善效能指令，確實可在 ARM 原有的微架構下，經過稍微的修改而不須大量的硬體支援即可完成，而且這些指令改良對整個 Datapath 的 Execution Time 亦不致造成嚴重的 Delay。本研究開始著手於 ARM Datapath 之微架構電路設計，完成 ARM 架構 Gate Level 的電路描述與模擬，成為新加指令的驗證平台，具體驗證：

1) 新增指令的硬體成本(Gate Counts)。
2) 新增指令執行對 Execution Cycle Time 的影響。

我們以 ASIC(Application Specific Integrate Circuit)的方式，選擇 VHDL[6]為硬體描述語言完成 ARM Datapath 的 Gate Level 設計。以 Xilinx Foundation[5]為電路之開發平台，完成整個設計、模擬、合成到燒錄測試的研究程序，在本論文中，我們將首先介紹 ARM-7 的微架構，並於第三章說明上述加密及乘加指令的硬體改良設計，該設計之 FPGA 實作已完成，詳細的硬體評估數據則列入第四章中。

II. ARM-7 之微架構及主體設計

II.1 ARM-7 之硬體架構



〔圖 1〕 ARM-7 微架構

ARM7TDMI 處理器，其架構如〔圖 1〕所示，主要可分為以下七大部份

(1)暫存器組(Register Bank)

(2)乘法器(Multiplier)

(3)桶型位移暫存器(Barrel Shifter)

(4)算術邏輯運算單位(ALU)

(5) 位址暫存器(Address Register)與累加

器（Incrementer）

(6)資料暫存器(Data Register)

(7)指令解碼器（Instruction Decoder）

本研究在硬體實作階段，首先即完成上述七大部份的設計，再逐步修改增加指令有關的主要電路（如：移位器、乘法器），完成以下之設計。限於篇幅之限制，以下僅將處理器設計中較複雜且與本文相關的乘法器及 Control Unit，先做一概略性的介紹。

II.2 ARM-7 之乘法器

II.2.1 Modified Booth's algorithm

由於 ARM 的乘法器設計是採用 Booth's algorithm，考量 ARM 核心對計算速度之需求，我們以相關研究中所提出之改良式 Booth's algorithm 設計乘法器〔4〕。將乘數分成四個 8-bit，再做四次 8×33 的循環，其中多加的第 33 位元，代表有號或無號相乘。Booth's algorithm 改良為一次取出 3 bits 來判斷該執行何種動作，其演算法之對應表如〔表 1〕所示。因此，改良後之演算法其循環次數將減少一半。而每次循環所產生乘積如以下公式所示。

$R(i)=R(i-1)+ Booth(i*8+1)*2^1 + Booth(i*8+3)*2^3+ Booth(i*8+5)*2^5 +Booth(i*8+7)*2^7,$ $i=0,1,2,3$，

當 i＝0 時，R（i－1）＝ －A&B[0]，

當 i＝1,2,3 時，R（i－1）為上一循環計算出來的結果。

〔表 1〕 Booth's algorithm 對應表

Multiplier B[i+1]	B[i]	B[i-1]	Operation	Remarks
0	0	0	0	String of zeros
0	0	1	+A	End of 1's
0	1	0	+A	A single 1
0	1	1	2A	End of a 1's
1	0	0	-2A	Start of 1's
1	0	1	-A	End/start of 1's
1	1	0	-A	Start of 1's
1	1	1	0	String of 1's

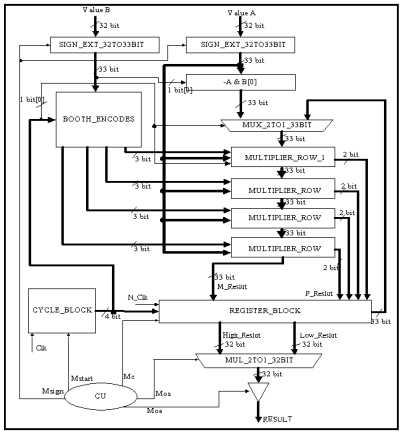
II.2.2 乘法器的主體架構

我們將乘法器的 Booth Algorithm 分成如〔圖 2〕所示的三大部份完成：

【1】Booth Encoder：此功能單元之主要動作是將輸入的 33-bits 乘數值分成四部分，並在每個循環將每部分訊號送至 Multiplier_Row 中。

【2】Multiplier_Row：此功能單元依據 Booth Encoder 所送出之訊號，將被乘數做相對應之處理。

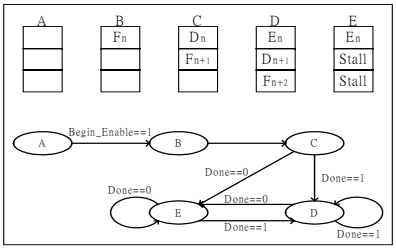
【3】Register Block：其功能為儲存每一次循環所產生之 8bits 之部分乘積（P_result），以及 33bits 部份乘積(M_result)。



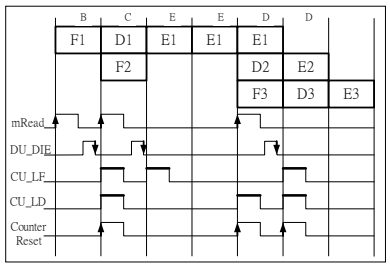
〔圖 2〕乘法器結構圖

II.3 ARM-7 的 Control Unit 電路

我們以標準的有限狀態機（Finite State Machine, FSM）的技術，配合其它如 Counter、Decoder 等元件完成整個共 47 條控制輸出的 C.U.。其中 Decode 電路的功能是將 FSM、Counter 以及 Instruction 的訊號整合化簡產生每個指令在其各個 Cycle 所需的控制訊號，其中控制訊號線(Done)，是用來判斷指令是否完成，即為指令執行到最後一個 Cycle 時訊號為 1，因此，FSM 可藉由此訊號作如〔圖 3〕的狀態轉換。



〔圖 3〕 Control Unit 狀態轉換圖



〔圖 4〕 ARM-7 Pipeline 的控制

〔圖 4〕為一指令片段的 pipeline，由圖中可歸納出五種狀態(如〔圖 3〕之上半部)，圖中之 STATE A 為開機前之狀態，開機後 Begin_Enable 為 1，因此跳往 STATE B，而後無條件的跳往 STATE C，之後便依據 Done 這條控制訊號，決定之後的狀態〈Done 在指令的最後一個 Cycle 為 1〉。最後由〔圖 3〕便可設計出 FSM 電路。

III 附加指令的電路修改及效能評估

本章將根據前述之新增指令，介紹如何加入原始的 ARM 處理器電路中。

III.1 查表指令

III.1.1 Load.extract 和 Load.ex.eor

我們首先提出一種新指令：Load.extract，這個指令能將索引的展開、索引 Scaled、以及記憶體的存取合併至單一指令內。即為將有效位址的計算合併為一個指令，其格式表示為：

ldr.nb Rd,[Rb,Ra]

這個指令的動作除了將資料由實際記憶體位址讀進 Rd 外，最重要的是還包括了有效位址的計算，將 Ra 的第 n 個 Byte 取出，再向左位移兩個位元，最後加上 Rb 的基底位址，以得到有效位址。

由於加解密演算法中，Load 的資料往往緊接著會作 XOR(eor)運算，考量 ARM 處理器之 DataPath 及 Pipeline 設計，我們發現該 XOR 動作可加入 ARM 原有的執行路徑且不影響整體微架構之設計，因此本研究提出了第二個名為 Load.ex.eor 之新增指令。這個新指令的組語格式為：

ldr.nb.eor Rd,[Rb,Ra],Rc

III.1.2 Shifter 電路改良

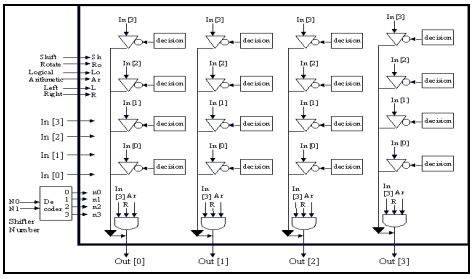
ARM 之指令集架構在處理查表動作時，主要由以三個指令完成有效位址的運算，前兩個指令主要為求出索引值，而後經由 Load 指令求出記憶體位址，以存取所需資料。考量本研究所提出新指令：Load.extract，這個指令在執行週期和資料路徑動作之安排上與一般 ARM 的 Load 指令相同，差異處在於計算有效位址時是採用 Extract 的概念，也就是取出 Ra 的某個 Byte 再做 Scaled。為支援該指令，其硬體電路的變動主要在指令解碼控制單元和移位器(Shifter)兩部分。由於新指令 load.4b、load.3b、load.2b 和 load.1b 各有不同的位移量，分別固定為向右位移 22-bits、14-bits、6-bits 和向左位移 2-bits，而這些位移量必須由控制單元決定，所以在控制單元內需新增解碼電路，主要功能為由指令的第 5、6 個位元計算相對應之位移量。此外由於索引值為 8-bits，因此其餘位元必須補零，所以也必須加上新的控制訊號，以決定是否補零，如此即完成指令 Load.extract 在硬體電路上之更動。

由於 ARM 之資料路徑的設計，在指令執行時，經由 shifter 作 extract 的動作取出索引值，再經由 ALU 計算有效位址，在執行週期之管線安排上相當自然，並不需增加執行週期數以達到新指令的功能。

ARM 採用 barrel shifter 之設計，使用一 cross-bar switch matrix。每一個輸入經由 switch 開關連結到一個輸出。而位移的動作可經由對角線之控制訊號對 switch 的開關來實現。為達到新增指令對 shifter 的動作要求，本研究又再加入一 SLe 控制訊號，目的為控制電路之補零動作，以完成新指令 Load.extract 中 extract 之動作。

〔圖 5〕表示 shifter 內部之電路設計，在此只用 4-bit shifter 表示，由於使用之設計工具之限制(因 Xilinx Foundation Series F4.2i 只能

做到 gate-level 電路設計)，所以我們使用三態緩衝器取代之 ARM 原本 NMOS 之設計，但其設計原理還是依照 ARM 之 cross-bar 的概念。



〔圖 5〕 Shifter 結構示意圖

在盡量不更改 ARM 的既有電路下，為了增加新增指令的功能，本研究在 barrel shifter 電路末端，增加一電路，使 out[9:2]保持原來輸出，其餘 out[31:10]和 out[1:0]則使其輸出為零。如此即可 shifter 具有 extract 的功能。

〔表 2〕列出 Xilinx 電路模擬之結果，我們使用之晶片族系為 XC400EX，晶片編號為 4036-HQ240，由表中可以發現，邏輯閘的增加非常有限(1.6%)，主要為那層閂鎖電路所造成，所以硬體電路增加之成本並不高，而平均延遲時間也只增加 1 ns。

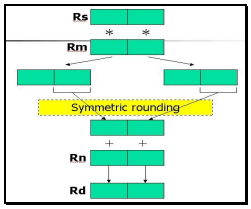
〔表 2〕電路模擬結果

	原本之 shifter	修改後之 shifter	增加比例
Gate number	1472	1496	1.6%
Average Delay Time	15.3ns	16.3ns	6.5%

III.2 SIMD Parallel MAC 指令

III.2.1.1PMLAV 和 PMLA

PMLAV 其動作概念如〔圖 6〕所示。而 PMLA 與 PMLAV 之差別只在 Rm 為兩筆相同的 16-bits 資料，

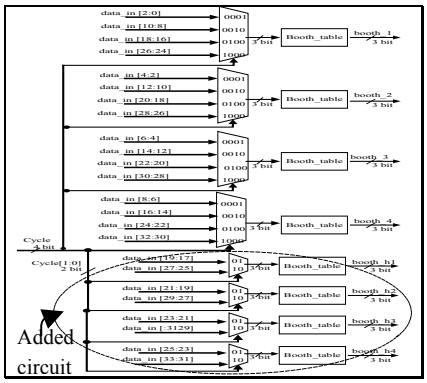


〔圖 6〕 PMLAV 動作概念

III 2.2 乘法器電路之修改

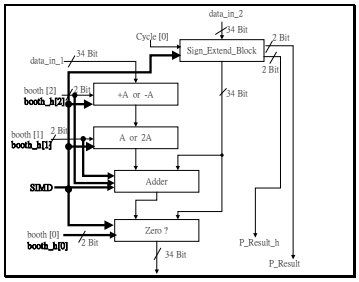
為使原乘法器亦能產生平行計算 2 組 16bits 乘 16bits 的乘加結果，本研究根據原乘法器電路架構作修改，以下分別說明各功能單元內部之修改重點及電路設計。

〔圖 8〕中虛線所圈示的是在 Booth Encoder 部分所需增加之硬體元件，由於 SIMD 為同時平行處理乘數之 low half-word 及 high half-word，所以只需要兩個循環即可完成運算。利用新增之四個 2 選 1 多工器，第一個循環除了送出欲處理比對之 low half-word 的前 9 bits ，同時也一併送出 high half-word 的前 9 bits，再分別經過 Booth_table 轉換後，輸出至下一階段 Multiplier row 1 至 Multiplier row 4 做處理，第二個循環則是送出後 9 bits。



〔圖 8〕 Booth Encoder

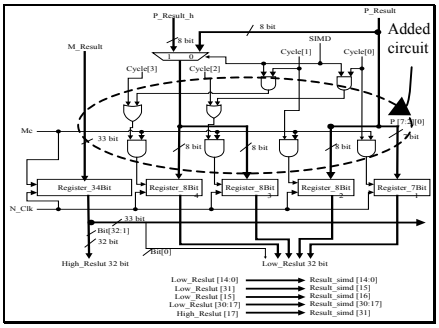
〔圖 9〕之粗體線路部分為配合新增指令所增加的線路：“+A_or_−A”以及“A_or_2A”兩個 Block，當 SIMD 訊號為 1 時，較高的 17Bit 將會由 Booth_h 的訊號來決定動作。在“Zero？Block”中較高之 17-bits 輸出，需由 SIMD 訊號來選擇要由 booth[0]、還是 booth_h[0]控制。



〔圖 9〕 Multiplier_Row_1

〔圖 10〕中虛線所圈示的為增加之部分，圖中 Low_Reslut 32 bit 即為經過四次循環處理後乘積之較低位元，High_Reslut 32 bit 即

為較高之位元。由於新增之平行乘加指令為一平行處理的 MLA 指令，而原始的 MLA 指令最後只取 64bits 乘積中之較低的 32bits，所以當執行 PMLAV 或 PMLA 指令時，也只需取各個乘積中較低之 16bits，而 Result_simd 即為執行 SIMD 指令後之兩筆 16bits 乘積。



〔圖 10〕 Register Block

改良後之乘法器，新增兩條控制訊號（SIMD、SV），執行 PMLAV 指令時 SIMD 為 1，SV 為 1；執行 PMLA 指令時 SIMD 為 1，SV 為 0；執行 MLA 指令時 SIMD 為 0，SV 為 0。

為了達到 16-bits 平行乘加，將 SIMD 之技巧加至 ARM 處理器核心中，對乘法器內部硬體架構所作之修改，增加之硬體成本不多（如〔表 3〕所示），且在執行 SIMD 16-bits 平行乘指令時只需兩個循環即可完成，花費時間亦只有傳統不具 SIMD 功能乘法器的一半。

〔表 3〕電路模擬結果

	原本之 multiplier	修改後之 multiplier	增加比例
Gate number	1472	1496	1.6%

IV 處理器晶片之實作及驗證

IV.1 處理器晶片實作

本研究之開發平台環境如〔表 4〕所示：

〔表 4〕開發平台環境

	配　　備
作業平台	**桌上型 PC**：Pentium-4 2.0G CPU 512MB DDR RAM 40GB HD
軟體	**作業系統**：Windows2000
	電路輔助設計軟體（EDA）：Xilinx Foundation 4.2i Xilinx ISE 5.1i ModelSim XE v5.6
實驗電路板	**FPGA 燒錄平台**：Xilinx Virtex-II FG456 Proto Board
	FPGA 晶片：Virtex-II xc2v250 可重複式燒錄晶片
	其他：JTAG Cable Line
測試設備	**邏輯分析儀**：DMATEK Pro-Open SERIES LH007

本研究所設計的 ARM 7 TDMI 相容指令集的處理器，使用 Xilinx Foundatuon，軟體內有完整的設計與模擬驗證流程，而設計步驟主要分為以下六個階段：

(1)架構設計、VHDL Code 設計

(2)功能上的驗證模擬（Function Simulation）

(3)電路合成（Synthesis）

(4)佈局與繞線（Place & Route）

(5)時序驗證模擬（Timing Simulation）

(6)晶片燒錄（Programming）

我們把設計的電路經由 EDA 輔助軟體的合成和時序分析後，將所得到的數據加以分析，並再經過不斷的改良，以期望得到各個元件的最大效能。以下是各個邏輯元件經過電路合成後所得到的數據。其中〔表 5〕為決定晶片面積的邏輯閘數目（Gate count of design）的統計，而〔表 6〕為各元件的延遲時間（Max. path delay）。

〔表 5〕主體單元的硬體成本

Decode 階段	Data Unit	Control Unit
Gate count of design	1184	4205
Additional JTAG gate	3264	7205
Slice Flip-Flops	64	112
Max. path delay	8.7 ns	32.3 ns

〔表 6〕各元件的硬體成本及延遲時間

Execution 階段	Multiply Register	Barrel Shifter	ALU	Register Bank	Address Unit
Gate count of design	6573	4749	3105	15242	787
Additional JTAG gate	4944	3744	5328	8640	4533
Slice Flip-Flops	66	0	0	1024	32
Max. path delay (ns)	32.5	17.1	21.4	10.3	14.5

最後，我們得到設計的晶片實際的 System Clock 最高可到達 24 MHz。

IV.2 晶片實體驗證

晶片實體驗證方面，由於驗證儀器的不同，我們把它分成三個部分討論。

- CPU 晶片部分：Xilinx Proto Board
- 輸入端部分：訊號產生器
- 輸出端驗證部分：邏輯分析儀

IV.2.1 CPU 晶片部分：（Xilinx Proto Board）

目前我們使用的實驗電路板為 Xilinx 公司原廠的 Xilinx Virtex-II FG456 Proto Board，搭配的 FPGA 晶片則是 Xilinx 原廠的 Virtex-II xc2v250 可重複式燒錄晶片：

IV.2.2 輸入端部分：（訊號產生器）

在輸入端部分，我們使用訊號產生器來輸入數位訊號，經過排線與實驗電路板上 FPGA 晶片的 input ports 連接，而給予晶片實際的硬體訊號，而不是經由 PC 軟體來產生模擬的硬體訊號，這樣會使我們的驗證更符合晶片真正的工作環境，更能避免訊號在經由 PC 輸出時產生時序上的誤差。

IV.2.3 輸出端驗證部分：（邏輯分析儀）

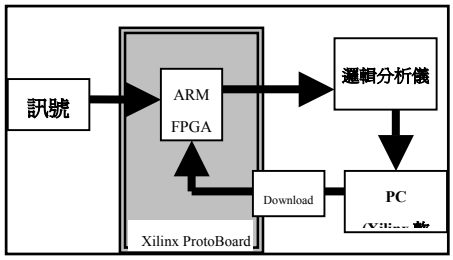
在輸出端部分，則是使用邏輯分析儀，把 FGPA 晶片的 output ports 接出來，連接到電腦上後，經由電腦分析其訊號波形。

因此，整個晶片實體驗證部分如〔圖 11〕所示，歸納其步驟如下：

（A）使用 Xilinx Foundation 軟體內附的 iMPACT 燒錄軟體，把製作好的電路程式經由 dowanload Cable 線下載到實驗電路板上。

（B）然後由訊號產生器，對照指令集格式表作為輸入端的輸入波形，一一的測試所有類別指令的輸入。

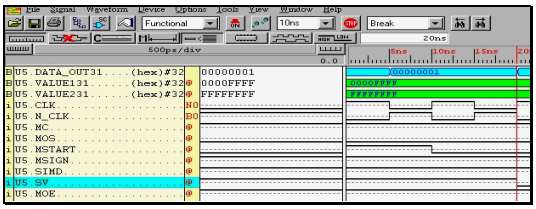
（C）最後由邏輯分析儀把訊號傳回電腦，與預期的輸出端波形做比對，驗證其正確性。



〔圖 11〕晶片實體驗證

IV.2.4 驗 證 結 果

經過上述步驟的實際執行測試，已證實本研究所完成的處理器核心電路能完全正確的執行，以下僅以 PMLAV 的執行為例，說明該指令的硬體運作結果：如〔圖 12〕所示，測試內容為 $0000FFFF_h$ 與 $FFFFFFFF_h$ 作平行相乘，即為 0000_h 乘 $FFFF_h$ 與 $FFFF_h$ 乘 $FFFF_h$，經過兩個 clock 的運算後，產生乘積 00000001_h。



〔圖 12〕PMLAV 指令測試

V 結論

本論文選定知名的嵌入式處理器ARM為研究標的，針對資料加密演算法及多媒體資料的處理特性所提出的兩個快速查表指令 Load.extract 和 Load.ex.eor及兩個SIMD式的乘加指令PMLAV 和 PMLA，由指令格式的定義至ARM處理器核心電路的設計，最後更以FPGA電路實作驗證了整體設計的正確性，並達成了“以最少量的硬體成本，擴充處理器能力以因應未來之應用趨勢”的設計精神。

我們也可由本研究過去針對資料加密及多媒體運算所提出支援指令之模擬統計中[5][10]，發現資料加密演算法的效能一般可增進17%~34%，而PMLAV / PMLA兩指令對多媒體應用程式之數據如表〔7〕所示，一般可加快10%~20%。

在資料加密驗算法中，我們針對各Benchmark進行靜態和動態的分析與統計，由靜態分析結果可以發現查表指令的確佔加密程式之一定比例，而且新指令也能有效取代原有查表動作。動態模擬結果更進一步說明了新指令對效能提昇的影響，加密執行效能有

12%(MARS)~34%(Rijndael)的提昇。

〔表 7 〕 整體效能提昇比例[10]

基準程式	整體效能提昇
dot	48%
color	10.62%
convolution	20.16%
composite	17.2%
edge detect	13.6%

目前本研究仍持續進行中，在下一年度的研究中，我們的目標將朝向使ARM處理器更加有效率並加入一般的加速技術，如將Pipeline增為五個Stage，同時開發適合的Branch Predication技術，以去除控制危障(Control Hazard)[11]，並且將推測執行(Predicated Execution)[12]及其技術加入到處理器架構設計中。

誌謝

本研究接受國科會計畫編號
NSC91-2213-E-035-020 之經費補助

參考文獻

[1] 林傳生, “使用 VHDL 電路設計語言之數位電路設計”, 九月, 2000.

[2] 鍾明政、吳金勇, “XILINX FPGA 數位邏輯設計”, 八月, 2000.

[3] Holmann E., Yoshida T., Yamada A., Mohri A., “A media processor for multimedia signal processing applications,” *IEEE Workshop on Signal Processing Systems*, *SIPS 97 - Design and Implementation*, pp. 86 –96, 1997.

[4] I. J. Huang, Y. L. Hung, “Cost-Effective Microarchitecture Optimization of the ARM7TDMI Microprocessor”, *International Computer Symposium*, pp.3, Taiwan, December 2000.

[5] Chia-Hua Liu, Larry Wang, Hong-Yang Hsu, "Enhancing the Performance of Encryption from the View of ISA," *14th National Information Security Conference 2002(ISC2002)*, pp. 265-271, Taiwan, May 2002.

[6] G. Martin and F. Schirrmeister, “A design chain for embedded systems,” *Computer*, Vol. 35, Issue 3, pp.100 –103 , March 2002.

[7] A. Murat Fiskiran and Ruby B. Lee, “Performance Impact of Addressing Modes on Encryption Algorithms,” *ICCD 2001*, pp. 542-545, 2001.

[8] D. A. Patterson and J. L. Hennessy, *Computer Architecture: A Quantitative Approach* Third edition, San Mateo, Calif.: Morgan Kaufmann, 2002.

[9] Talla D. and John L.K., “Execution characteristics of multimedia applications on a Pentium II processor” *Performance, Computing, and Communications Conference 2000*, pp. 516 –524, 2000.

[10] L. Wang, Leo Fang and H. Y. Hsu, “Equipping the SIMD MAC Operations into embedded RISC Processors,” *2002 International Computer Symposium (ICS2002)*, pp. 69 –76, Taiwan, 2002.

[11] P. H. Wang, H. Wang, R. M. Kling, K. Ramakrishnan, J. P. Shen, “Register Renaming and Scheduling for Dynamic Execution of Predicated Code”, *The Seventh International Symposium on High-Performance Computer Architecture (HPCA)*, pp.15-25, 2001.

[12] Chia-Lin Yang, Sano B., Lebeck A.R., “Exploiting parallelism in geometry processing with general purpose processors and floating-point SIMD instructions,” *IEEE Transactions on Computers*, vol. 49, pp.934 –946,9, Sept. 2000.