支援N階乘指令的處理器實作

Implementation of Processor with N! Computation Instruction

指導教授：林寬仁 博士　　學生：戴瑮、嚴嘉俊

輔仁大學 電機工程學系 大學部專題生

摘要

處理器，是現代電子產業中不可忽視的一環，在未來的科技發展中，也將隨著技術所需相依的發展下去，阿里巴巴在近期的報導中表示「AI設計發展中，IC設計扮演著『頭腦』的角色，在整個產業中極為重要，而處理器則是IC設計中最主要的基本架構，他甚至為了發展AI，跑去投資IC設計的公司，讓他們製作他們所需的電子產品的專用Processor等相關開發Project 」可見處理器在現代科技中，它的基礎性與重要性。

處理器相當複雜，主要是以邏輯閘為梁柱，架構出晶片所需的「運算」邏輯結構，成為電子相關產品運作的核心。它主要的功能主體是運算，使晶片達到某樣所需的目的或訊號指令的下達，所以它的運作速度與用途，會因為它所包含的架構、指令、運算方式等等的不同，產生不同的功能、速度或成本的處理器。

系統架構

·專題主要由硬體程式語言構成，由多個指令組合成一個 RISC 包含相當數量的邏輯閘，構成最基本的處理器程式，並且都使用16位元的訊號。其中再使用指令去完成所謂的N階乘的運算，再重新撰寫成獨立的函數，使其增添新的指令。

ADD	Adds the contents of source with destination register and store into destination register.
READ	Fetch a memory word from specified memory location by address load into specified destination register.
WRITE	Write the contents into specified location memory from source register.
BR	Jump the activity flow by loading the program counter with the word at the location specified by the second byte of the instruction.
MUL	Multiple the contents of source with destination register and store into destination register which is simulation from logic gate multiplier.
GRT	Return specified word from determine source g

圖一、指令基本列表

1. *Single byte instruction*

Opcode								Source				Destination			
0	0	0	0	1	1	1	1	1	0	0	1	0	1	0	1

2. *Two byte instruction*

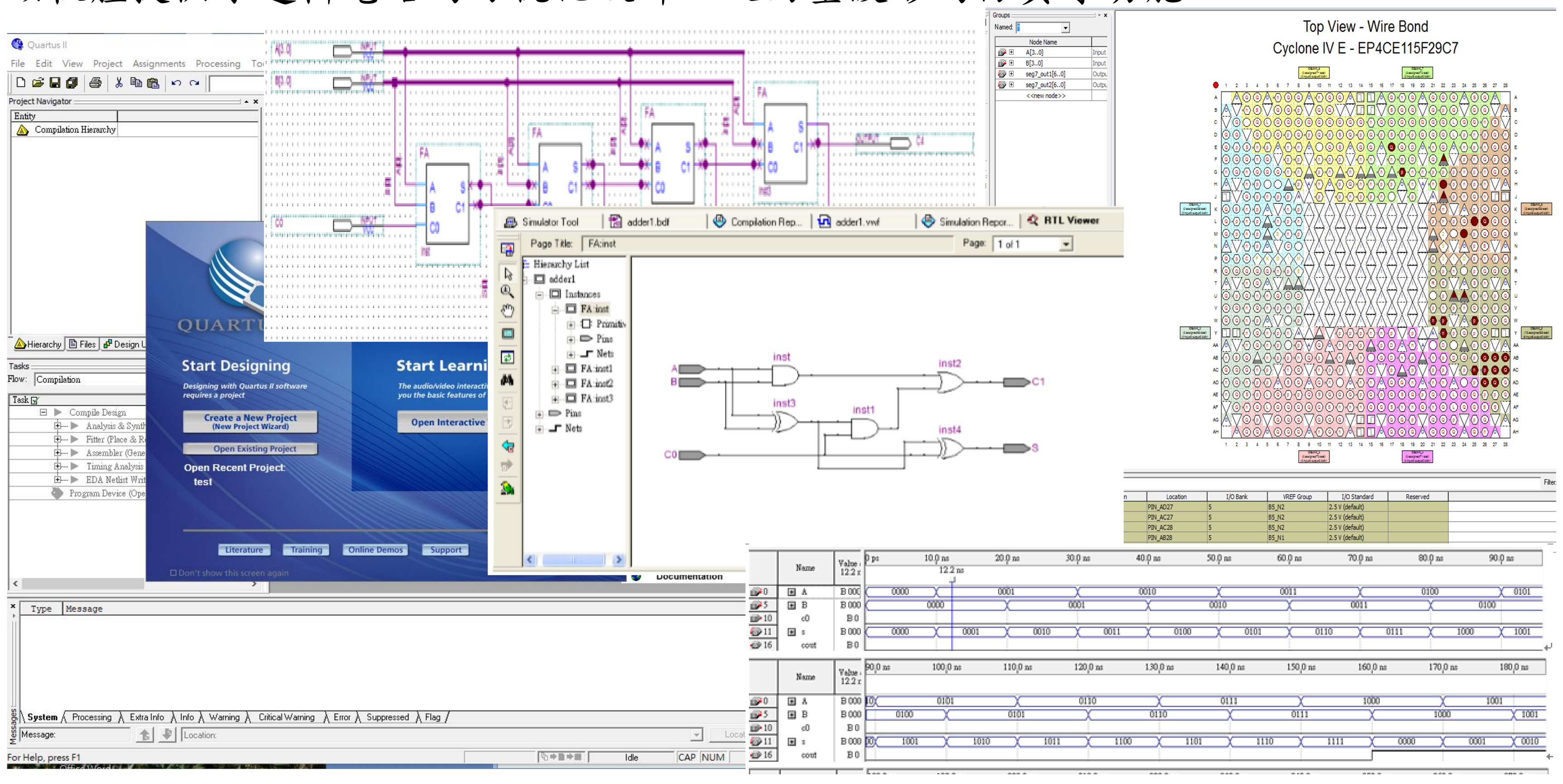
Opcode								Source				Destination			
0	0	0	0	1	1	1	1	1	0	0	1	x	x	x	x
Address															
1	0	1	0	0	1	1	1	0	0	0	0	0	1	1	1

X: don't care bit

圖二、16位元訊號

開發環境

Altera Quartus II 是由美國矽谷的一家可程式邏輯裝置和可反覆配置複查數位電路的製造公司Altera（中文名：阿爾特拉）的一家未上市公司於1984年推出的產品。它是一種可程式邏輯裝置電子設計自動化的開發軟體。可以識別常用的Verilog或VHDL高級硬體描述語言表述，或讀取指定格式的線路圖；進而完成邏輯仿真、功能驗證、邏輯綜合等任務，對器件的進行編程，即將設計項目轉換到實際的硬體。該軟體提供了邏輯電路的可視化設計以及向量波形的仿真等功能。



圖三、Quartus II程式介面

FPGA DE2-115 開發板是由友晶科技公司與Altera公司合作所開發的實驗室的硬體程式驗證平台，主要能夠執行Stratix、Arria和Cyclone三大系列程式，以及MAX系列程式功能等，除了承襲 DE2 系列豐富多樣的周邊應用介面外，還新增了支援高速 Gigabit Ethernet (GbE) 的介面。並提供一個 High-Speed Mezzanine Card (HSMC)介面，可連接 HSMC daughter cards 來擴充周邊應用，另外也可透過 HSMC cable 連結多片 DE2-115 開發平台，來實現大型 ASIC prototype 的開發驗證。涵蓋的系統相當多，相當適合用於教育開發的驗證平台。

圖四、驗證平台
FPGA DE2-115

ALTERA now part of Intel

圖五、開發公司之商標

實作方法與成果

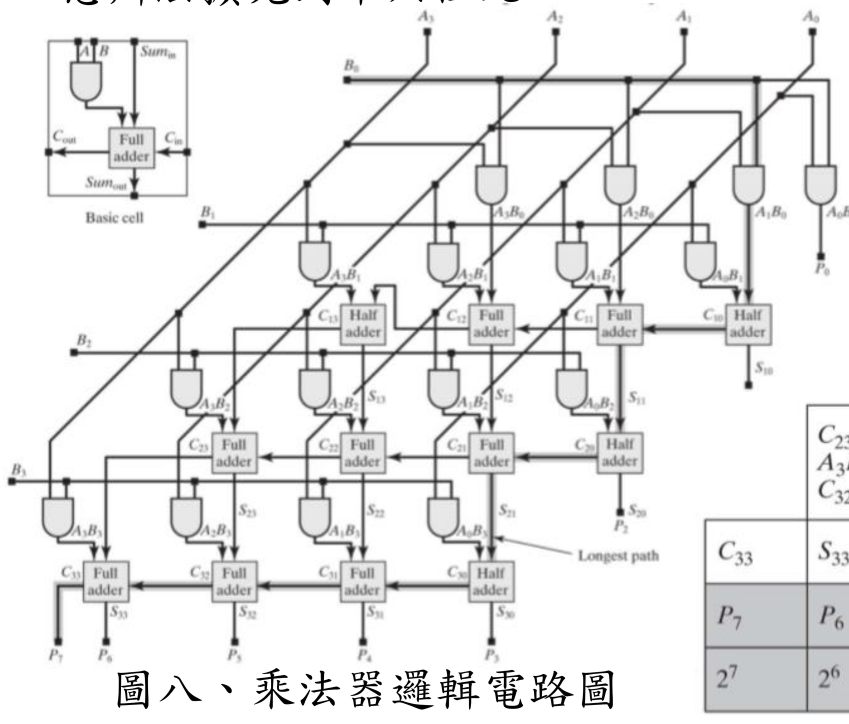
1. 經由課程所學，理解並使用Verilog HDL撰寫一個處理器（RISC）的程式碼，因為所學為八位元的處理器，所以我們想先將其程式的執行寬度擴大為十六位元，使程式的資料大小變寬。（概念如圖二）

Instruction	Instruction word		
	Opcode	Source	Destination
NOP	0000_0000	xx	xx
ADD	0000_0001	src	dest
SUB	0000_0010	src	dest
AND	0000_0011	src	dest
NOT	0000_0100	src	dest
READ	0000_0101	xx	dest
WRITE	0000_0110	src	xx
BR	0000_0111	xx	xx
BRZ	0000_1000	xx	xx
MUL*	0000_1001	src	dest
GRT	0000_1010	src	dest
HALT	1111_1111	xx	xx

圖六、程式指令之編號

圖七、處理器程式

2. 理解數位系統設計的課堂上所學的乘法器的概念，並撰寫一個乘法器（Multiplier），並想辦法擴充到十六位元。



圖八、乘法器邏輯電路圖

圖九、乘法步驟圖

				A3 B3	A2 B2	A1 B1	A0 B0	Multiplicand Multiplier
			A_3B_1 C_{12}	A_3B_0 A_2B_1 C_{11}	A_2B_0 A_1B_1 C_{10}	A_1B_0 A_0B_1	A_0B_0 S_{00}	Partial product 0 Partial product 1 1st row carries
		C_{13} A_3B_2 C_{22}	S_{13} A_2B_2 C_{21}	S_{12} A_1B_2 C_{20}	S_{11} A_0B_2	S_{10}		1st row sums Partial product 2 2nd row carries
	C_{23} A_3B_3 C_{32}	S_{23} A_2B_3 C_{31}	S_{22} A_1B_3 C_{30}	S_{21} A_0B_3	S_{20}			2nd row rums Partial product 3 3rd row carries
C_{33}	S_{33}	S_{32}	S_{31}	S_{30}				3rd row sums
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	Final product
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Weight

3. 透過我們第一步驟與第二步驟所寫的預設程式指令模擬N階乘。所以我們要先建構出N階乘的架構。

用數學表示：

$$n! = 1 \times 2 \times 3 \times \cdots \times n = \prod_{k=1}^{n} k$$

遞歸方式定義：

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \times (n-1)! & \text{if } n > 0 \end{cases}$$

圖十、階乘的概念

階乘有無限多個，表列前100個如下：

0!	= 1
1!	= 1
2!	= 2
3!	= 6
4!	= 24
5!	= 120
6!	= 720
7!	= 5040
8!	= 40320
9!	= 362880
10!	= 3628800
11!	= 39916800
12!	= 479001600
13!	= 6227020800
14!	= 87178291200
15!	= 1307674368000
16!	= 20922789888000
17!	= 355687428096000
18!	= 6402373705728000
19!	= 121645100408832000
20!	= 2432902008176640000
21!	= 51090942171709440000
22!	= 1124000727777607680000
23!	= 25852016738884976640000
24!	= 620448401733239439360000

圖十一、階乘結果資料表

4. 最後再利用第三步驟的邏輯於程式內利用預設指令撰寫，獨立撰寫出一個專門用來計算N階乘的程式指令。（這些動作都會燒錄到FPGA開發板進行驗證。）

結論

1.硬體程式相較於軟體程式，較不適合使用迴圈、除法等等方法撰寫（再製作晶片上，會造成不少的成本。），特別是在要寫專門用來撰寫可能實踐成硬體或晶片的程式的時候，所以再限制上會使其較難書寫，最後常常都會寫得較為冗長，且繁複。（p.s.但是越是詳細，速度越快。）

2.可透過外部程式 De1-SoC 將組合語言轉成C語言，再由ARM讀取程式中的DATA輸出到其他的軟體平台顯示或使用。

3.熟悉程式撰寫後，我們能繼續在處理器中加入新的指令或是新的演算法，使其更多樣化。

4.可以進行各種程式的撰寫以及各種程式若使用不同的寫法，或是不同的演算法，所可能造成不同的耗時、耗記憶體或者製作晶片的成本差別，甚至是不同電路的耗電程度，都可以做更深入的研究。

2017 輔仁大學電機工程學系 大學部專題成果展