基於可程式化系統晶片的非同步電路雛型實現

***陳宥佑、楊榮林**

南臺科技大學電子工程系

*ma230109@stust.edu.tw

摘要

本論文提出一雛型化非同步電路(asynchronous circuits)及測試的新方法，該方法適用於初期設計想法探索及非同步電路設計教學。非同步 VLSI 的設計方法已發展多年且日趨成熟，學界及業界亦提供大量 EDA 工具，但對大多數位系統設計工程師而言，非同步電路設計仍是相當陌生的課題。造成此窘境的原因可歸納為以下兩點，在一般大專院校的數位系統設計課程，非同步電路雖然屬於進階課程，但常被視為非主流的數位系統設計，因此授課教師通常會略過此章節。此外，目前缺乏成熟的非同步電路實現技術及合成工具，無法如同步數位系統般可使用 FPGAs 快速雛型電路實現及實作驗證平台。就當前主流 VLSI 技術發展趨勢而論，短期內非同步電路應無機會取代同步電路(synchronous circuits)。但團隊始終相信，非同步設計將有助於解決同步電路設計上所遭遇的技術困境。因此重整先前所開發的基於 FPGAs 非同步電路實現技術，改用可程式化系統晶片(SoPC)來實現非同步電路及測試輔助系統，此一改良非同步系統實現方法，可降低測試輔助系統對待測電路(CUT)的干擾，並使測試向量的產生及供給更具彈性。以下以簡單的電路實驗來分析本文所提的新創方法，評估結果證明足以應付大部份非同步電路雛型及測試需求，而該方法亦可作非同步系統設計教學之快速雛型及電路驗證平台。

關鍵詞：非同步、硬體描述語言、可程式化系統晶片、交握、微處理器

A Novel Approach of Using SoPC to Prototype Asynchronous Circuits

You-You Chen*, Jung-Lin Yang

Department of Electronic Engineering, Southern Taiwan University of Science and Technology

Abstract

We proposed a novel approach of using SoPC to prototype asynchronous circuits with on-board diagnostics testing. The method can be used in the early design stage of exploring asynchronous systems, and it is also an affordable experimenting platform for teaching asynchronous circuits. Asynchronous VLSI design methods have been developed for years. Some open-source asynchronous EDA tools for logic-level synthesis are obtainable freely from the internet. However, asynchronous circuits are still mysterious for most of the circuit design engineers. The cause of this dilemma can be summarized as the following two reasons. In today's college course, the chapter of the asynchronous circuits is rarely taught or mentioned. Asynchronous VLSI is usually classified as an advanced topic and skipped by most teachers of the digital design classes. Also, lack of mature circuit-level synthesis tools, implementing asynchronous systems require much more design efforts than building synchronous counterparts. Judging from the near future trend of VLSI technology, there is not much chance that asynchronous circuits will become as popular as synchronous circuits. However, we believe asynchronous design can help to solve some serious challenges commonly seem in today's synchronous systems. Therefore, we proposed a simple method of using a low-end SoPC platform to prototype asynchronous circuits. In this thesis, we layout the

Received: Sep. 26, 2016; first revised: Feb. 10, 2017; second revised: Dec.29, 2017; accepted: March, 2018.
Corresponding author: Y.-Y. Chen, Department of Electronic Engineering, Southern Taiwan University of Science and Technology, Tainan 71005, Taiwan.

proposed method in detail with some sample designs. Although, we still not able to perform a state-level and real-time asynchronous verification for the test subjects. All the sample designs were passed by a series of high speed on-board functional tests. Based on the experimenting result, we have faith in the usefulness of the proposed method.

Keyword: Asynchronous, HDL, SoPC, Handshaking, Microprocessor

壹、前言

不論在業界或學界，同步系統的 EDA 工具一直不虞匱乏並推成出新，但缺乏如 FPGA 般的非同步系統實現及驗證平台，使在一般大學推廣非同步數位系統顯得窒礙難行。本文中以同步(synchronous)數位系統的電子設計自動化(EDA)工具為基礎，開發一系列適用於非同步(asynchronous)數位系統設計及合成的輔助工具。藉由對同步數位系統 VLSI 高階合成工具的了解及使用經驗後，盡可能找出適用於非同步電路設計的相關工具組，但因兩系統對訊號取樣的時序觀念極為不同，將不可能有完全適用的工具存在，所以藉由該計畫來扮演黏著劑的角色，將現有的 SoPC 合成工具與非同步控制器的邏輯合成工具做緊密結合，簡化非同步數位系統的設計流程，並加速非同步功能方塊的驗證及 SoPC 的電路雛形實現。

貳、背景研究

一、 非同步電路

非同步電路並不是一個全新的設計理念，它也不再是過去那個只存在於實驗室中的非成熟技術，昇陽的 Ebergen 博士已帶領著他的研究團隊，將此項技術使用在昇陽的 UltraSparc III i 微處理及外部記憶體的控制器上[1]。飛利浦的資深科學家 Ad Peeters 博士所帶領的研發團隊，也以非同步技術開發了一系列的省電型通訊專用微處理器，並將此項技術運用在傳呼機及智慧晶片卡等相關產品[1]。由 IBM 研究學者 Dharmendra S. Modha 博士所發表的腦直覺式處理器(brain-inspired processor)更是為同步與非同步電路混合設計寫下另一里程碑[2]。這些成功的案例，足以顯示出非同步設計在未來的高階數位電路設計應用將扮演著舉足輕重的角色。

非同步系統的運作原理非常近似事件導向(event-driven)程式設計觀念。運算元件間以訊息傳遞的方式來完成協同運算，各級運算元件只會在接到要求後才會開始動作，完成運算後並通知下一級前來取用結果，這中間的所有等待皆為不定時間，一切以「交握(handshaking)」的方式來導引運算的流程，這樣的工作模式非常自然且淺顯易懂。先不管電路是如何實現及所會面臨的技術困難，就以此一淺顯的工作原理來探討非同步系統的設計瓶頸[3-5]。非同步的交握信號有非常嚴謹的運作模式，在非同步環境下，時間是連續性的，每一個訊號在任何時間點上的轉變皆有其特定的涵意。所以在控制邏輯的實現與最佳化時，須完全去除潛藏的訊號危障(hazards)，換言之，同步電路控制邏輯中可被忽略的危障，是決不允許出現在非同步系統中的，因此非同步控制邏輯電路會比對等的同步設計複雜並大上許多。

另一難處則是缺乏現成的非同步元件及模組，目前市面上可取得或購得的數位模組皆以同步系統為主，亦即要再短時間內完成一特定功能項的非同步系統是非常困難的。再加上一般學校對數位設計的教學，仍以訓練同步電路專才為主， 非同步電路設計人員的匱乏可見一般。設計及合成工具的不成熟也是一大阻力，雖自 80 年代末期既有不少的電腦科學家及工程師，投入開發自動化非同步電路設計工具，迄今絕大部分的工具，還是以合成及最佳化演算法為主軸，對設計者在使用的便利性上仍嫌不足，且使

用此類型工具的工程師，須具備相當的非同步電路設計訓練，絕非像同步系統的合成工具那麼簡便易學，更為非同步設計的入門增添了極高的門檻[6, 7]。

近年來已有為數不少的研究團隊，著手致力於混合式設計方法的開發，昇陽、IBM、Intel 及一些學術界的非同步研究團隊，相繼提出不同的解決方案，且都有不錯的成績。相信非同步設計，將有機會成為系統晶片(SoC)及嵌入式系統(embedded systems)設計的新寵兒。加上近年來 FPGA (field-programmable gate array)和 SoC (system-on-a-chip)設計的盛行，選擇從高階合成並以 SoPC 實作來切入該領域，希望藉由對非同步電路及合成工具的設計經驗，整合出一套同時試用於實作驗證與教學訓練的非同步系統設計流程[8]。

二、 GALS(Global Asynchronous Local Synchronous)物聯網應用

設計完善的非同數位系統早已被證實同時具有低消耗功率及極為優異的系統強健性[1][7-8]，不論是處在充滿雜訊亦是溫度及供給電源變化劇烈的工作環境，非同步數位電路都可以自我調適來因應環境參數的改變，加上跳脫傳統同步數位系統對總體時脈(global clock)訊號的依賴，不需額外設計時脈訊號及區域電源控制電路，就可以理所當然的擁有高效能及低功耗的表現，而這些正是目前物聯網(Internet of Things)應用迫切需要的數位電路屬性[9]。

非同步數位系統依輸入/控制訊號改變的限制又可分為輸出入模式(input output mode)及基本模式(fundamental mode)，如果系統允許輸入/控制訊號可隨意改變，且各訊號間的轉態沒有任何時間限制，該非同步系統類屬輸出入模式的設計，最早是由 David Muller 博士在 50 年代所提出，同時期 David Huffman 博士也提出了基本模式非同步系統，這種模式對環境訊號的改變有了較多的限制，基本模式只允許單一訊號的改變，且一旦有訊號改變必需等系統反應後且回復到穩態，才可允許下一個訊號轉態的時間限制[7, 8]。輸入模式依電路實現的延遲時間預設又可以區分為：延遲鈍化(delay insensitive, DI)、準延遲鈍化(quasi-delay insensitive, QDI)、 速度獨立(speed independent, SI)、 及 自我時序(self-timed, ST)，DI 是屬於最保守且強健性最高的電路，所有的連接線及邏輯閘都有無限的正時間延遲預設，這種設計既是所謂的「完善」非同步數位系統，然而也因為極為嚴苛的延遲時間預設，很難實做出具實用性的電路系統，QDI 及 SI 雖然在連接線的延遲預設上做了不同程度的讓步，但依然在最終電路的實做上極為綁手綁腳，因此我們選用了自我時序 ST 的實現方式，它的特性與 QDI 的設計極為相似，但在延遲時間的預設上加上了上下區間的限制，而這個限制區間會在電路合成的過程當中漸進式的更新，延遲預設上的區間限制可大幅增加電路合成時的優化空間， 也是最適合以可程式化邏輯(FPGA)作為實作平台的方式之一。

以交握協定(handshaking protocol)為基礎的非同步電路，絕對是解決目前同步電路在物聯網應用上所遭遇的種種困境，但礙於大型非同步電路設計、合成、及驗證上的困難度，我們並不建議單以非同步的方式設計物聯網應用相關的晶片及系統，但若改以同步電路為主體輔以非同步的交握介面，並將某些易受干擾亦是高耗電的區塊，改以非同步設計的方式來提高系統強健性及有效降低功耗，系統設計工程師將在效能、功耗、強健性間擁有較大的運作彈性，而 GALS (Global Asynchronous Local Synchronous)就是最典型的設計樣板[9]。

三、 微控制器

1947 年世界第一顆的電晶體於貝爾實驗室誕生，積體電路促使著人類的發展，從小型積體電路(SSI, Small-Scale Integration)、中型積體電路(MSI, Medium-Scale Integration)、大型積體電路(LSI, Large-Scale Integration)、超大型積體電路(VLSI, Very-Large-Scale Integration)…等等，微控制器在現今生活當中已是不可或缺的部分，因為製程的進步，使得晶片精簡縮小，可以從家電控制、生活娛樂、車用電子…等已

經足夠說明微控制器與生活的緊密度，這些微型電腦又可以依位元數分為各種複雜度的單晶片來因應各種應用，例如四位元常用於一般家電控制，八與十六位元可以不依賴作業系統來應用較複雜的運算處理，三十二位元通常搭配嵌入式系統且具網路功能可以做最複雜的運算。

做單晶片的開發，最早之前幾乎是此用組合語言，如今幾乎是使用 C 語言，因為有很多的單晶片內建了線上編譯設計(ISP)，改變了以往的開發模式，更加速了設計、製造、除錯的時間，大大的提高了工作效率，也有公司開發開源的編譯器包裝微控制器及免費好用的函式庫，降低微控制開發的門檻，即使不懂組合語言或 C 語言也可以輕鬆上手，所以使用微控制器已經不是那麼遙不可及。

四、 SoPC 平台

SoPC(system on programmable chip)是從 SoC 延伸出來的，都是將系統電路全部整合在一個單晶片上，與 SoC 不同的是加上了可程式化(programmable)的功能，其實就是一個完整的系統再加上 FPGA，透過本身系統的程式及 FPGA 可重複編譯的特性可以依需求去更改系統的架構及周邊，然而 FPGA 的廠家也有提供開發環境供使用者使用，如 Altera 的 SoPC Builder, 及 Xilinx 的 EDK 讓使用者可以快速的開發；一般 SoPC 平台常常被應用在需要複雜運算的影像處理系統平台上，然而研究團隊希望用 SoPC 來打造一個實作非同步及驗證非同步的平台。

參、非同步微控制器實作

由於本實驗室一直致力於非同步電路相關的研究，所以在設計非同步電路及驗證平台都有一些見解，因為非同步電路的合成及驗證的工具相對較少，一般測試非同步電路的方式是利用 XBM 來設計非同步的狀態機，並透過 FPGA 來合成非同步的控制器及模擬非同步測試訊號的模組，如圖 1，但是礙於測試訊號模組有時間函數及延遲而造成無法合成的問題，在驗證的過程當中，沒有辦法像同步電路一樣，可以即時的驗證、分析結果，只能以運算的結果來證明。

因此希望利用 SoPC 平台，為非同步電路打造一個驗證平台，架構如圖 2，圖中測試引擎(TE)的部分就是實作的 RAFSM，透過 PS 的 ARM 傳送測試碼給 TE，利用 PL 較快的時脈對待測電路(CUT)進行高速的運算，以達到趨近於即時(Real-time)的速度，並將運算結果傳回 PS 端驗證，以實現非同步驗證平台，雖然本文的新方法足以應付大部份非同步電路雛型實現及測試的需求，仍無法改變現階段 SoPC 的本質來呼應非同步設計的需求。但就教育及非同步設計的推廣而言，本文中的研究成果仍極具價值，不論是在非同步電路教學或研究上，該方法依然是可取得的非同步電路實現及測試最佳方案。

圖 1 非同步驗證之架構

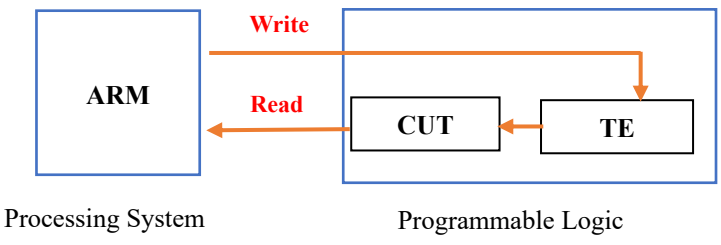


圖 2 非同步驗證架構

一、電路觸發條件

非同步電路動作原理如圖 3 所示，REQ 觸發由低態轉為高態時，CTRL 模組送出 R1_REQ 訊號給 R1 暫存器，則 R1 開始動作，運算結束後，將結果輸出，再傳回 R1_ACK 訊號給 CTRL 模組，告知 CTRL 模組可以進行下一次運算，當 CTRL 模組接收到 R1_ACK 訊號之後，隨即傳出 R2_REQ 訊號給 R2 暫存器，通知 R2 暫存器可以開始動作，以此類推交握方式當全部完成運算後，CTRL 模組則會輸出檢知訊號(Acknowledge, ACK)表示此系統完成所有動作。

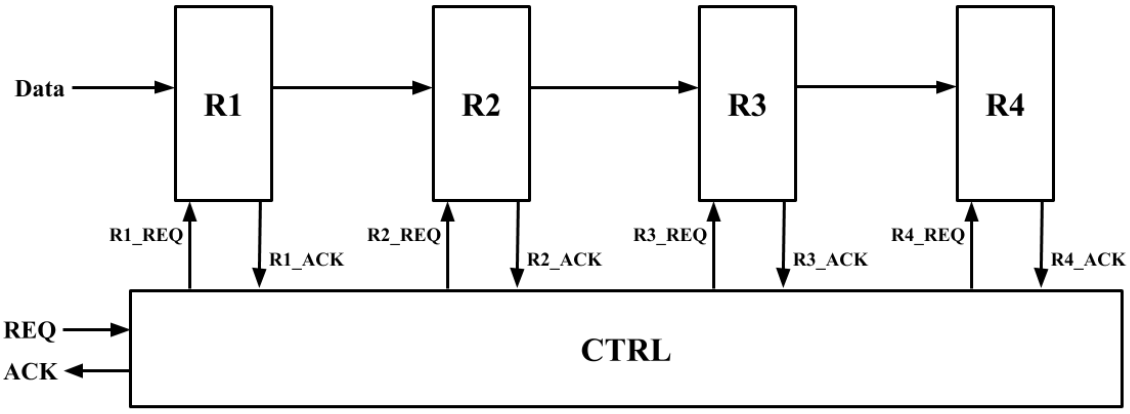


圖 3 電路交握示意圖

二、四相式資料包裹通訊協定

圖 5 為四相式資料包裹波形示意圖，REQ 及 ACK 訊號初始狀態皆設定為低態，模組不做任何動作，此時 Data 為可任意改變的無效資料。搭配來解釋，當 Sender 端傳送資料至 Receiver 端時，Sender 端將 Data 準備完成，此時 Data 為不可改變的資料，接著 REQ 訊號被設為高態，告知 Receiver 端可以開始接收資料。當 Receiver 端接收資料結束後會將 ACK 訊號轉變為高態，告知 Sender 端資料傳送已結束。Sender 端收到 Receiver 端 ACK 訊號由低態轉變為高態時，表示 Receiver 端接收完所有資料，並把 REQ 訊號設為低態。Receiver 端收到 Sender 端將 REQ 訊號由高態轉變為低態時，Receiver 端也將 ACK 訊號歸為低態，表示所有動作已結束。

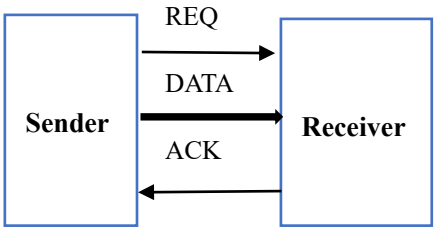


圖 4 資料包裹資訊協定架構

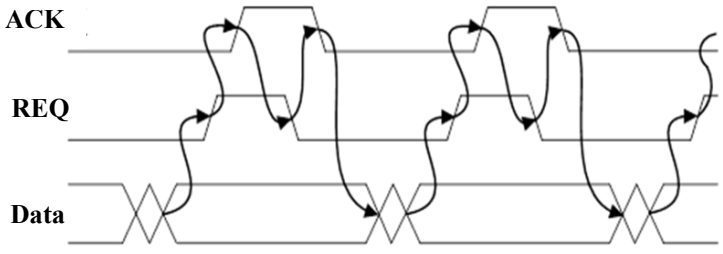


圖 5 四項式資料包裹通訊協定

三、BM/XBM 非同步狀態機

爆發模式(burst-mode, BM)狀態機[10]是常見的非同步有限狀態機(asynchronous FSMs, AFSMs)描述之一，藉由設定的條件來引導非同步系統訊號的交握與控制。在爆發模式狀態機中，預期狀態與指定輸出狀態之間是以“|”標籤符號隔開，當預期狀態條件滿足後，其輸出的變化將由指定輸出狀態來決定。“|”符號左邊的變數集合為預期狀態，又稱為輸入爆發(input bursts)；“|”符號右邊的變數集合為指定輸出狀態，又稱為輸出爆發(output bursts)。在輸入爆發及輸出爆發中所使用的變數描述語法相同，每一個變數將緊跟著“ ＋ ”或“ － ”來描述對該變數的預期狀態或指定輸出狀態。以“a＋”、“b＋”為例，該敘述若出現在輸入爆發變數集合中，所設計的爆發模式狀態機將等待訊號 a 與 b 由邏輯 0 轉變成 1。若該敘述若出現在輸出爆發變數集合中，意指所設計的爆發模式狀態機將會在預期輸入爆發條件滿足後，將訊號 a 與 b 由邏輯 0 轉變成 1。不同於同步系統的訊號輸出規則，非同步系統的控制訊號必須遵守單一轉態(monotonic change)規則，所以在先前的敘述中，以“由邏輯 0 轉變成 1”來描述輸入及輸出爆發，而非一般同步系統中常用的“輸出為 1”。乍看之下兩種敘述並無不同，但就非同步狀態機而言，“輸出為 1”不僅有“由邏輯 0 轉變成 1”的意涵，同時亦包含有“由邏輯 1 或未知狀態轉變成 1”的可能，這將使非同步狀態機無法正常運作，這也是為何危障(hazards)在所合成的非同步控制電路是絕不被允許的[11]。

我們要使用的是由 Yun 博士將 BM 狀態機中改良的延伸爆發模式(extended burst-mode, XBM)狀態機。為了使此狀態機更具彈性，他以 BM 為基礎加入指定狀態轉移路徑(directed don’t care)及準位訊號(level signals)的概念，除此之外 XBM 完全與 BM 相容，所以在使用上較沒負擔。

四、設計自我時序資料路徑

此部分根據交握訊號描述中的自我時序資料流路徑描述，轉譯出如圖 6 為自我時序資料流路徑，並將各個自我時序資料路徑之功能元件一一以應對延遲資料包裹來實現，由於完整設計資料流路徑過於繁雜，在此將會以簡單的例子說明。

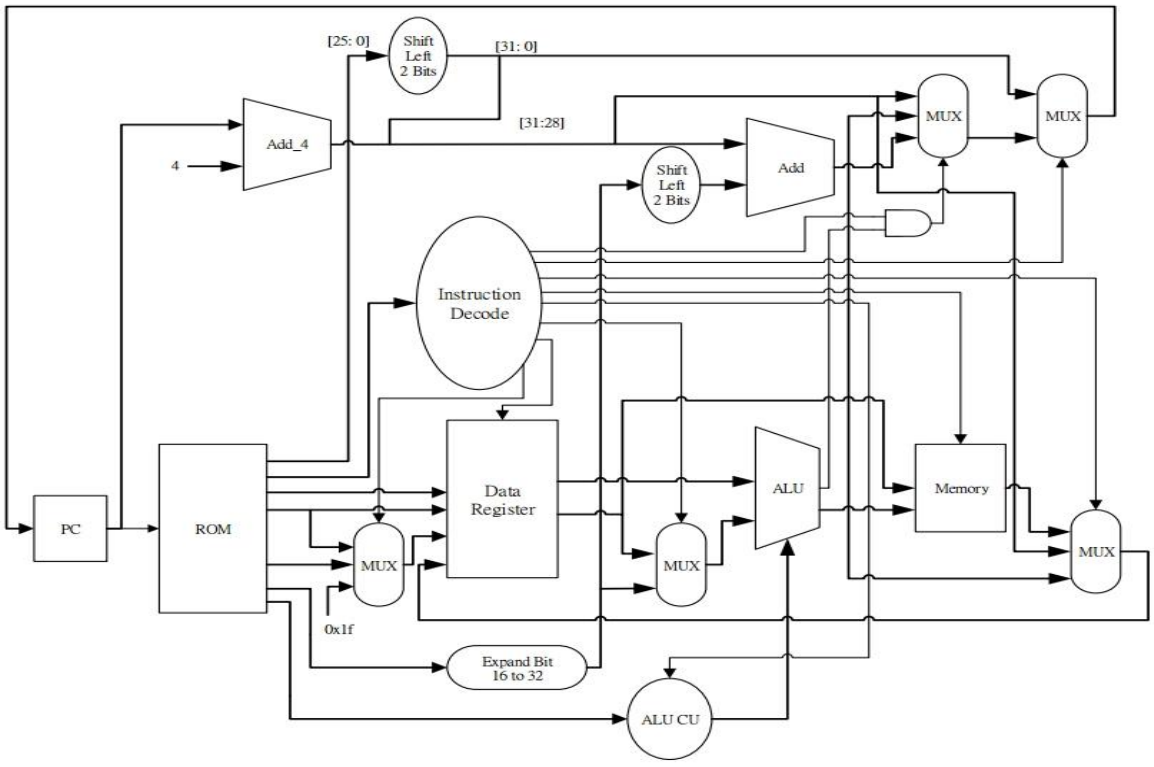


圖 6 MIPS 資料路徑

利用 verilog 實作了可合成元件後，接下來就可對這個電路進行功能驗證及靜態分析，功能驗證主要是確認電路的運算符合行為模式的動作，靜態分析則是分析電路的運算時間，找出電路最差的運算時間，這個最差情況將作為研究的延遲依據。

找出電路最差狀況時間後，將這個數據在電路中加入對應的延遲，把電路包裹成自我時序的資料路徑，由功能運算電路加上延遲元件組成自我時序的資料路徑元件，如圖 7。

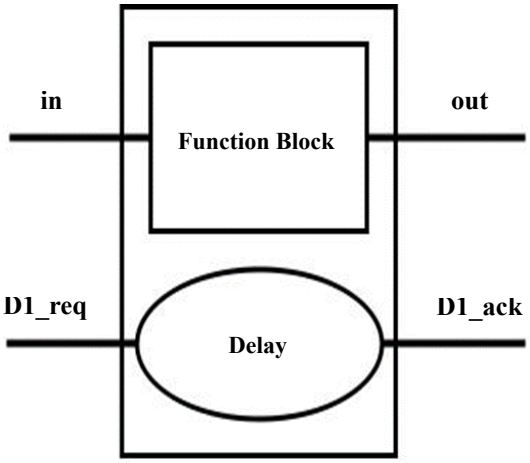


圖 7 自我時序的資料路徑元件

在包裹的過程中，是使用四項資料包裹的方式運作，因此在延遲元件的設計上要特別注意，不可太快太慢，太快會導致電路運算錯誤甚至發生不可預期的結果，反之太慢雖然可以確保電路正常運作且不會發生錯誤，但是在速度上會變慢，因此在設計上會以靜態時序分析最慢的時間作為標準，進行延遲元件的實現。

五、實作與驗證

(一)指令集

本文採用 MIPS R2000 當作微控制器之範例，先介紹其指令集，它可分為三種指令集，分別為 R-type、I-type、J-type，以下是微控制器指令碼之介紹：

R-type

Opcode	Rs	Rt	Rd	Shamt	Funct
6bits	5bits	5bits	5bits	5bits	6bits

I-type

Opcode	Rs	Rt	Address
6bits	5bits	5bits	16bits

J-type

Opcode	Address
6bits	26bits

- Opcode:指令之基本運算，一般稱為「運算碼」
- Rs: 第一個來源運算元之暫存器
- Rt: 第二個來源運算元之暫存器
- Rd: 目標運算元之暫存器,將存放運算結果
- Shamt: 移位量(shift amount)
- Funct: 功能。指令在 op 欄位中的各種選項,亦稱「功能碼」
- Address: 暫存器之位址

圖 8 指令格式

表 1 指令範例

R-type

指令	指令格式						說明
	opcode	rs	rt	rd	shamt	funct	把 rs,rt 的值做加法運算並
ADD	000000	10010	01000	10001	00000	100000	將結果寫入 rd

I-type

指令	指令格式				說明
	opcode	rs	rt	address	讀取(rs+有號擴充立即數)的記憶體位置，並將資料存入 rt
LW	100011	10011	01000	0000000000100000	

J-type

指令	指令格式		說明
	opcode	address	
JUMP	000010	00000000000000100111000100	跳躍至 10000 的位址

(二) 電路合併

9

為團隊設計的非同步微控制器電路，非同步具有易模組化特性。此外，本文也將 ALU 模組結構化，形成非同步中還有非同步。而後將 ALU 中運算元分成 log(邏輯)、mul(乘法)、div(除法)等驅動模組，利用 to_choose_alu 模組，告知控制器選擇驅動模組。

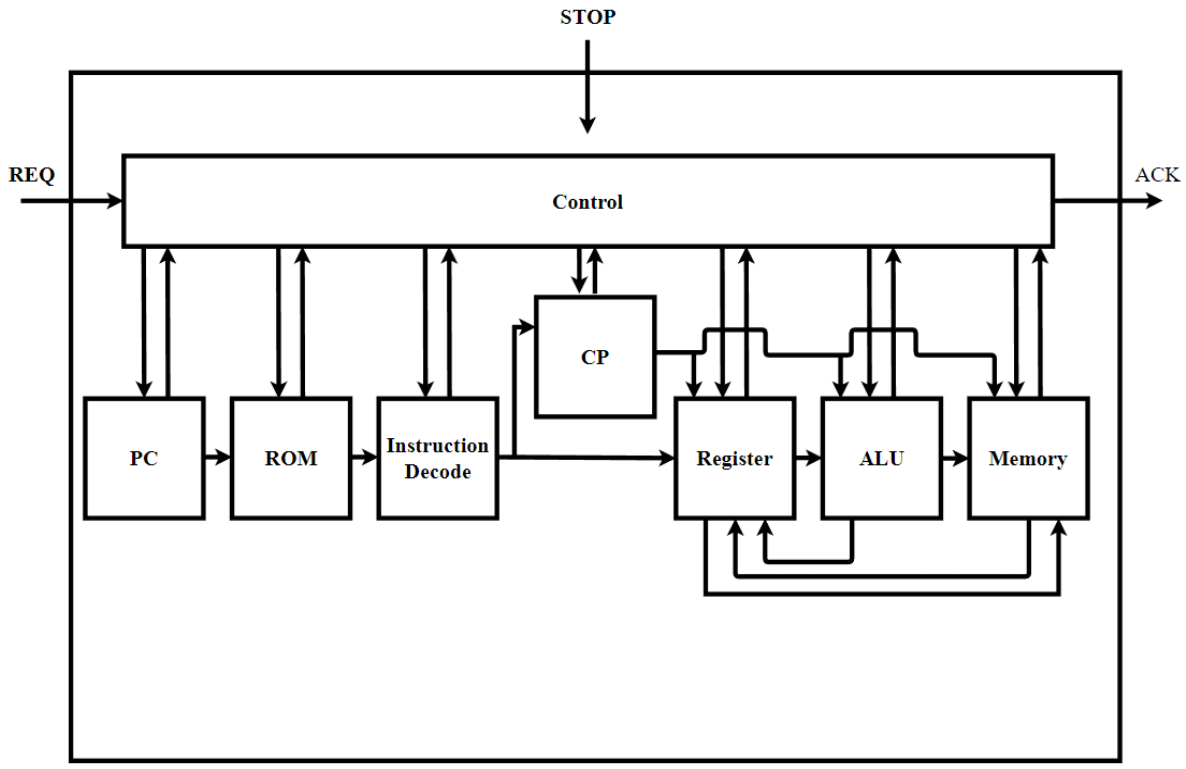


圖 9 非同步微控制器電路架構

(三) 電路驗證

1. 環境介紹

首先要介紹設計環境，設計之作業系統使用 Ubuntu 9.04 版，程式設計驗證使用 XILINX ISE 11.2 版合成工具，晶片使用 SPARTAN XC3S400-4PQ208。

2. 測試方法

由於非同步微控制器指令繁多，單純執行一個指令或少數指令並不能有效表示運算能力，因此以帶入演算法的方式，本文以執行費式數列演算法實際讓微控制器進行功能性驗證。

3. 模擬與驗證

第一個部分是以費式數列作為觀察功耗情形的演算法，測試模組有三個部分個別為：同步控制器、非同步控制器、非同步控制器及非同步 ALU 改良版，從費式數列當例子來觀測其結果，在同步與非同步的比較這之間大概減少了 3%多的功耗，證明作此改進是有降低功耗的，之後也做了其他的演算法來測試也是有降低功耗的效果，然而又進一步的對 ALU 進行了改進，將它做成了非同步的 ALU，發現又降低了 10%多的功耗，雖然這是在做費式數列的情況下所觀察的情形，也有可能因為運作的演算法不同而功耗會改變，但整體來看，此改進是有降低功耗的效果。

第二部分是將程式燒錄至實驗版中，利用版上的輸出端(七段顯示器)，來觀察其結果。

表 2 同步控制器執行費式數列功耗表

Total Quiescent Power	0.09853(W)
Total Dynamic Power	0.02195(W)
Total Power	0.12049(W)
Junction Temp	27.3 (degrees C)

表 3 非同步控制器執行費式數列功耗表

Total Quiescent Power	0.09837(W)
Total Dynamic Power	0.01815(W)
Total Power	0.11652(W)
Junction Temp	27.1 (degrees C)

圖 10 驗證燒錄合成結果

表 4 ALU 改良版之非同步微控制器執行費式數列的功耗表

Total Quiescent Power	0.09845(W)
Total Dynamic Power	0.00081(W)
Total Power	0.09925(W)
Junction Temp	27.2 (degrees C)

肆、非同步驗證平台模擬與分析

在介紹流程之前先說明驗證平台的開發環境，開發環境是使用 Xilinx 公司的 SOPC 平台 Zedboard，接下來會使用這個工具來實作非同步電路驗證的工具，Zedboard 開發版裡面有兩個部分，一個部分是 Processing System(PS)，此部分他是一顆 ARM 的晶片，另一個部分是 Programmable System(PL)，他是一塊 FPGA Zynq 的晶片，希望透過這塊開發版的優勢來作為實作的驗證平台。一直以來本實驗室一直致力於非同步電路相關之研究，根據之前的研究，非同步電路都是實作在 FPGA 平台下驗證，以圖 11 來說明，先將要驗證的演算法直接寫在 ROM 裡(淺色方塊)，再進行合成，但是這樣會因為修改測試的演算法而更改合成時所有的電路繞線，或者在合成過程中因為繞線問題導致不能合成，此時就必須修改演算法的

程式碼再重新燒錄。

因此本文希望不再將演算法一同寫在 ROM 裡，將 CPU 的架構固定，則可不用因為每次更改要測試的演算法，而在更改一次整體電路的繞線，利用 SoPC 平台之特性透過 PS 的 ARM 傳送 CPU 指令傳送或測試向量給 PL 裡所實作的非同步電路進行測試，就不會因為要驗證的演算法指令碼更改，而整個非同步電路在合成時一起更改，如此一來，就可以利用固定的硬體架構來測試各種演算法，如圖 12。

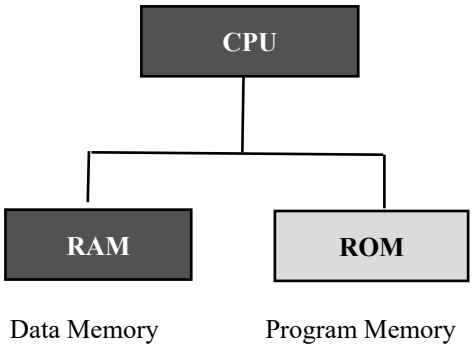


圖 11 微控制器合成架構

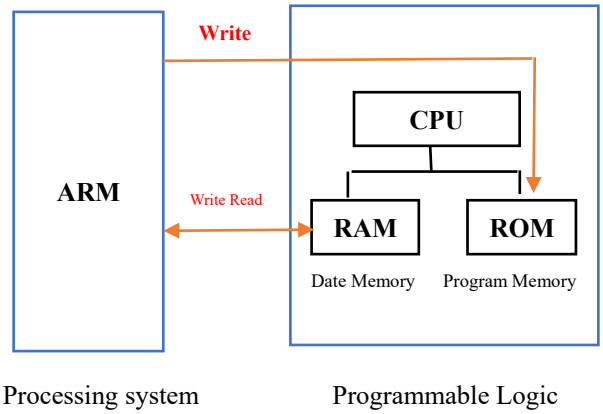


圖 12 測試模組規劃架構

一、SoPC 平台設計流程

因為是使用 Xilinx 公司的開發版，所以同時也是使用該公司的的工具 Vivado，因為它比之前的工具(ISE)在設計 SoPC 上提供更好的資源，更適合做 SoPC 開發，他也提供許多的 IP 供使用者使用；此平台常被用來實作影像分析、影像處理的嵌入式系統，透過 FPGA 合成高速的資料通道，讓整個系統可以達到即時(Real-time)，也有一些嵌入式比賽指定使用此平台，以下是依照官方文件與動手實作下完成的實際設計流程，如圖 13。

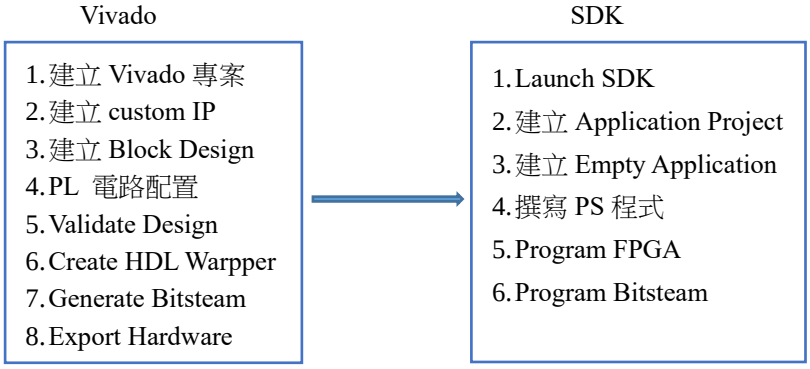


圖 13 Vivado 設計流程

二、CRC 電路

循環冗餘檢查碼(Cyclic Redundancy Check, CRC)，通常用來資料傳輸或資料儲存後可能會出現錯誤時，此時就可以透過此電路做資料檢測或校正資料。多筆資料經過此電路，透過輸出可以檢查資料在傳送中有沒有被更改或錯誤，因此被廣泛應用。

為了確保傳送資料的正確性與嚴謹性，在測試電路部分以 CRC 設計，利用此元件產生的資料序列做

為測試元件的資料輸入及後端的輸出結果作為簽名驗證元件，只要電路運算過程出錯，輸出的結果就會不同，藉由此方式驗證資料的正確性，在未來驗證非同步電路的時候，可以更嚴謹及完善。

首先，先構想電路的雛形，首先必須設計一個記憶體，來存放來自 PS 的多筆資料，暫存器再將資料傳給 CRC 電路檢測並且傳送結果至 PS 來驗證，如下圖 15。

利用 ISE 來模擬測試的結果，用 verilog 合成暫存器電路及 CRC 電路，inp 是輸入訊號依據表 5 所設計的指令碼，將要存放的資料放進所對應的位置裡，當 R_en 啟動時會將暫存器的資料依照時脈的正邊緣依序讀出，crc_en 是 CRC 電路啟動的時機，outp 則是經由 CRC 電路運算後所產生的結果，再以波型模擬來驗證結果如圖 14。

取得模擬結果後，再以 Vivado 來驗證，利用剛剛用 ISE 合成的電路，來設計新的 custom IP，再將設定好的 ZYNQ processing System(PS)加入，完成 Block Design，接著再設計 PS 端的 ARM code，將 ISE 所模擬的指令碼藉由 PS 端傳送給 PL 端，再將 PL 中的 CRC 電路運算結果透過 UART 來驗證結果是否與波型模擬後的結果一樣，是實驗的數據驗證想法是沒錯的，如圖 16

表 5 控制指令碼規劃表

Don't care	R_en	W_en	Address	Data
20 bits	1 bit	1 bit	5 bits	5 bits

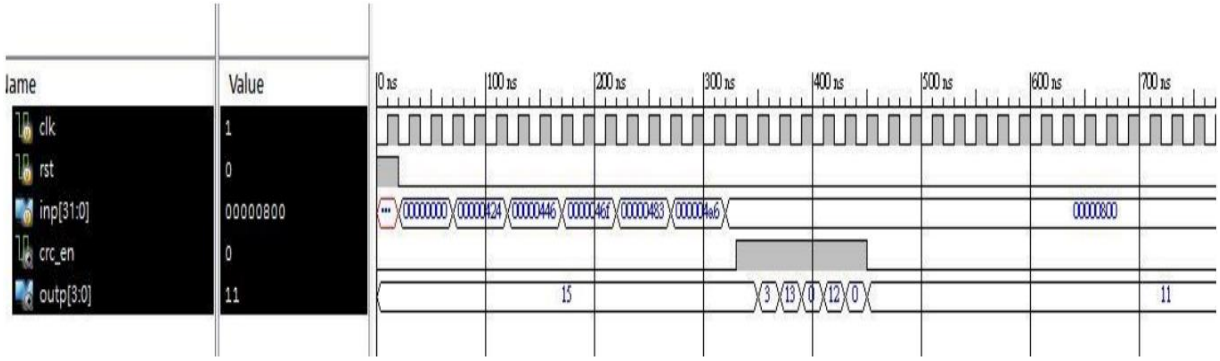
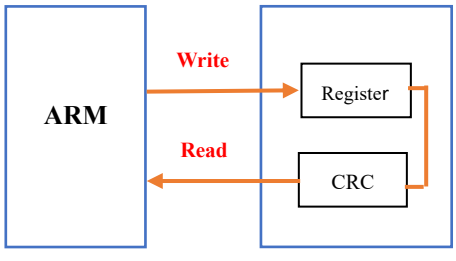


圖 14 ISE 測試 CRC 電路之波形



Processing System Programmable Logic

圖 15 CRC 電路驗證架構

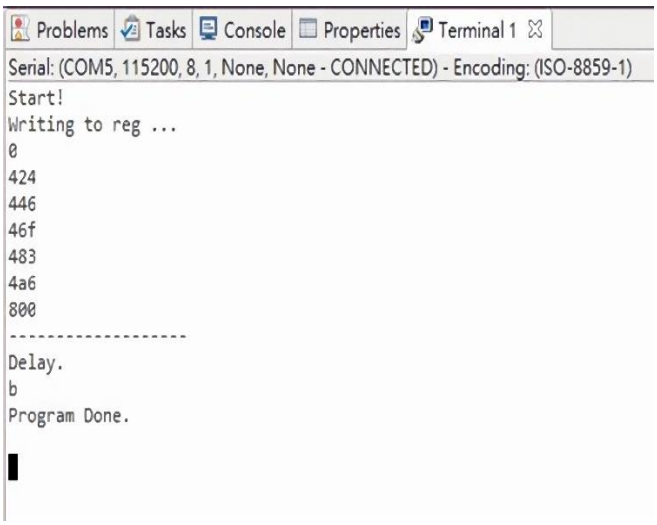


圖 16 用 Terminal 來觀察 CRC 電路相關資訊及結果

三、同步電路

有了前面的實驗基礎，接下來要測試基於 SoPC 平台的同步電路之驗證，先來介紹我所設計的電路，以下圖 18 組合邏輯的是所設計的同步電路圖，圖 17 同步電路之狀態圖，希望一次由 PS 傳送多筆資料來測試組合邏輯，並將每筆資料處理後，將得到的輸出儲存下來，當全部運算完成後，再將所以執行結果傳回 PS 來驗證。

設計完驗證架構後，也是先用 ISE 透過 verilog 先進行合成模擬，觀看波型模擬做初步的驗證，首先合成我們所設計的狀態機的電路，如圖 19，inp 是輸入訊號依據表 5 所設計的指令碼，將要存放的資料放進所對應的位置裡，當 R_en 啟動時會將暫存器的資料依照時脈的正邊緣依序讀出，En 是啟動狀態機電路啟動的時機，outp 則是經由狀態機電路運算後所產生的結果，取得模擬結果後，再以 Vivado 來驗證，利用剛剛用 ISE 合成的電路，並加上存放運算結果的記憶體，一同來設計新的 custom IP，再將設定好的 ZYNQ processing System(PS)加入，並規劃 GPIO 以方便測試驗證之結果，完成 Block Design，接著再設計 PS 端的 ARM code，將 ISE 所模擬的指令碼藉由 PS 端傳送給 PL 端，再將 PL 中的同步狀態機電路運算結果透過 GPIO 來驗證結果是否與波型模擬後的結果一樣，是實驗的數據驗證這樣的想法是正確的。

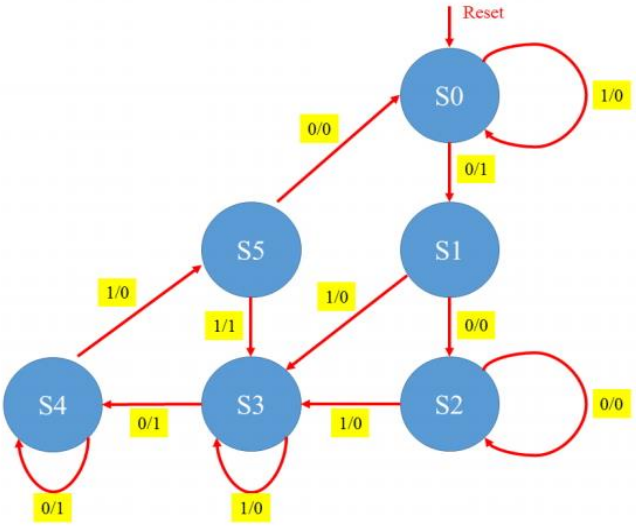


圖 17 同步狀態機狀態圖

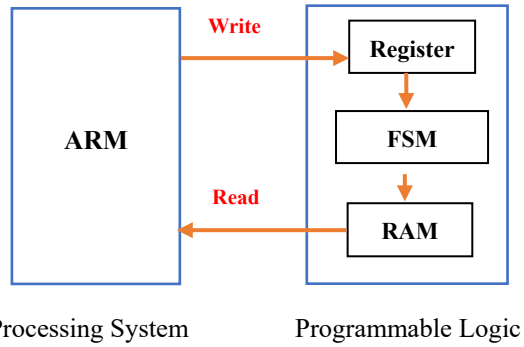


圖 18 同步電路驗證架構

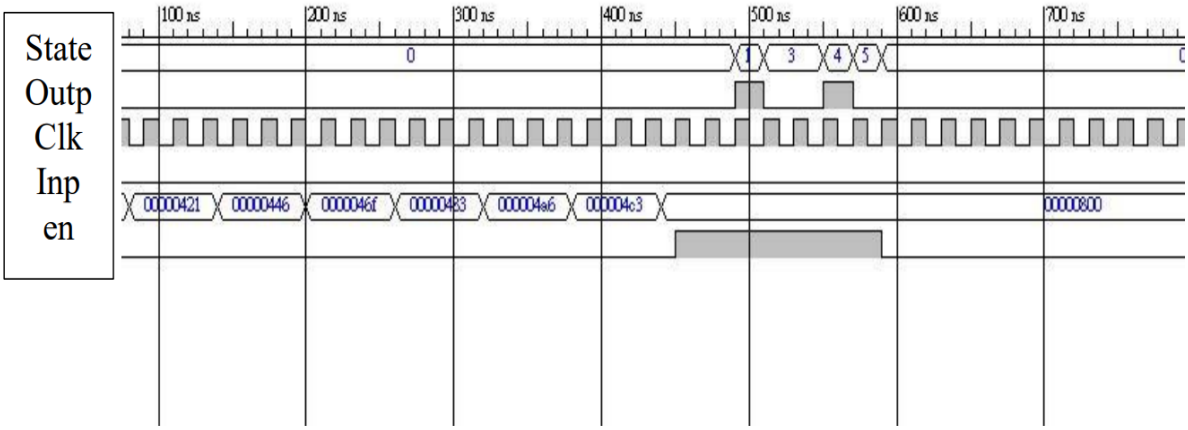


圖 19 ISE 測試同步電路驗證架構之波形圖

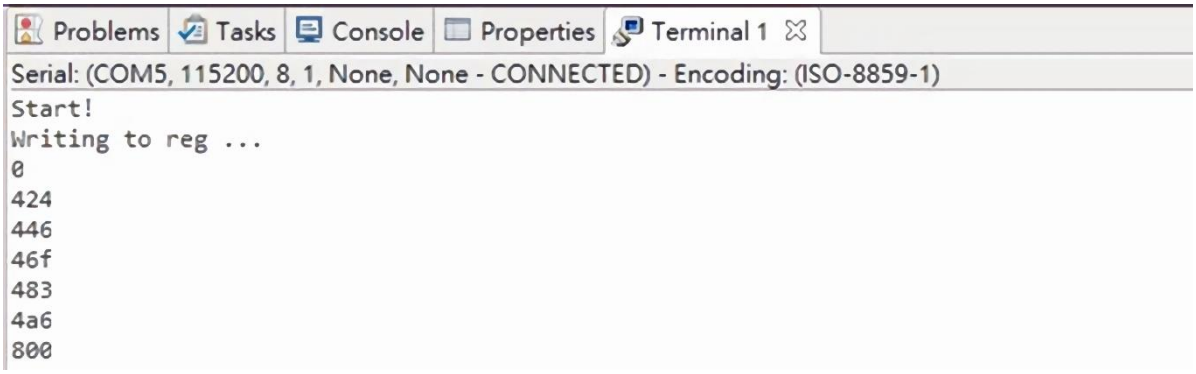
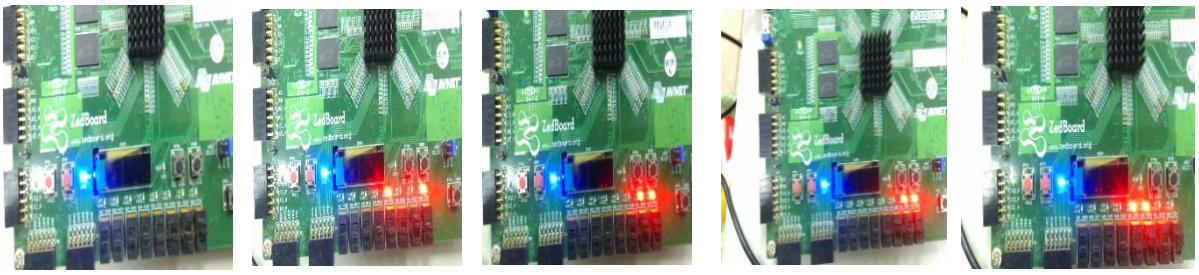


圖 20　用 Terminal 來觀察同步狀態機電路相關資訊



輸出：0 狀態：0	輸出：1 狀態：1	輸出：0 狀態：3	輸出：0 狀態：3	輸出：1 狀態：4

圖 21　同步狀態機電路實驗結果

伍、結論

本文提出一個基於可程式化系統晶片(SoPC)的非同步電路實現及測試方法，目前採用 Xilinx Zynq 系列 SoPC 做為評估該創新方法的實作平台，但本文所提之設計方法並不僅侷限於 Xilinx 的系統晶片。在 HDL 設計流程章節中，以一精簡版的 32 位元 MIPS 微處理器來說明整個非同步電路實作流程，實作過程包含非同步系統行為模式的描述，行為模式描述至交握模式硬體描述語言的轉換及功能驗證，可合成 RTL 自我時序資料路徑生成及非同步狀態機(AFSM)的萃取，接著再以 BM/XBM 邏輯合成工具來合成無危障(hazard free)的非同步控制器，最後將自我時序資料路徑與無危障非同步控制器整合置入 SoPC 晶片上的可程式化邏輯(PL, programmable logic)，在藉由微處理器系統(PS, processing system)來提供測試資料並監看待測電路的反應。就如同團隊所預期的結果，實驗顯示以此種方法實現的非同步電路，並無法像背景研究所指出的那般優異表現。這主要是因為實驗基準的不平等，一般市面上的 SoPC 實驗平台無疑是針對同步電路調校的，對非同步電路的實現而言，是一不甚友善的設計環境，雖然本文的新方法足以應付大部份非同步電路雛型實現及測試的需求，仍無法改變現階段 SoPC 的本質來呼應非同步設計的需求。但就教育及非同步設計的推廣而言，本文中的研究成果仍極具價值，不論是在非同步電路教學或研究上，該方法依然是當前可取得非同步電路實現及測試最佳方案。

參考文獻

[1] I. E. Sutherland and J. Ebergen. (2002). Computers without clocks. *Scientific American, 287,* 62-69.

[2] J. Kessels, T. Kramer, G. den Besten, A. Peeters, and V. Timm. (2000). Applying asynchronous circuits in

contactless smart cards. *ASYNC '00 Proceedings of the 6th International Symposium on Advanced Research in Asynchronous Circuits and Systems, 1,* 36-44.

[3] H. Park and T. Kim. (2016). *Synthesizing Asynchronous Circuits toward Practical Use,* 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Pittsburgh, PA, USA.

[4] M. Gibiluka, M. Moreira, and N. Calazans. (2015). *A Bundled-Data Asynchronous Circuit Synthesis Flow Using a Commercial EDA Framework,* 2015 Euromicro Conference on Digital System Design(DSD), Funchal, Portugal.

[5] K. N. Le-Huu, T. T. Nguyen, T. H. Bui, and A. V. Dinh-Duc. (2013). *Asynchronous circuit verification: From specification to circuit,* 2013 International Conference on Computing, Management and Telecommunications (ComManTel), Ho Chi Minh City, Vietnam

[6] K. Y. Yun and D. L. Dill. (1999). Automatic synthesis of extended burst-mode circuits. II. (Automatic synthesis). *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 18,* 118-132.

[7] J. Sparso and S. Furber. (2001). *Principles of Asynchronous Circuit Design: A Systems Perspective,* MA: Kluwer Academic Publishers.

[8] C. J. Myers. (2001). *Asynchronous Circuit Design,* NJ: John Wiley and Sons.

[9] E. Beigne, P. Vivet, Y. Thonnart, J. F. Christmann and F. Clermidy. (2016). Asynchronous circuit designs for the internet of everything: a methodology for ultralow-power circuits with GALS architecture. *IEEE Solid-State Circuits Magazine, 8*(4), 39-47.

[10] T. Curtinhas, L. A. Faria, D. Oliveira and O. Saotome. (2014). *A novel state assignment method for Extended Burst-Mode FSM design using genetic algorithm,* 2014 27th Symposium on Integrated Circuits and Systems Design (SBCCI), Aracaju, Brazil.

[11] D. A. Oliveira, N. Alles and L. A. Faria. (2012). *Synthesis by direct mapping of asynchronous extended burst-mode controllers using RS latch,* 2012 IEEE 3rd Latin American Symposium on Circuits and Systems (LASCAS), Playa del Carmen, Mexico.