基於可程式系統晶片的數位系統雛形測試框架

***陳顗文、楊榮林**

南臺科技大學電子工程系

*mb130102@stust.edu.tw

摘要

本研究針對數位系統設計教學，提出一低成本的數位電路測試框架，以協助學生有效地掌握電路設計和測試技巧，同時解決昂貴測試設備需求和專業培訓的挑戰。本研究的兩項主要工作，首先將測試框架和測試電路，燒錄至同一個可程式系統晶片（PSoC）中，減少對外部測試設備的依賴。其次使用常見的數位系統設計方法，在真實的教學環境中評估測試框架的實際可用性。這些工作項目已在PYNQ-Z2平台上完成實際測試，處理器端的程式設計採用了Python，而數位系統的雛形和測試框架則使用Verilog完成電路描述，最後透過Xilinx Vivado進行了高階合成。測試框架使用EMIO和四相通訊協定，實現了寫入測試資料、執行測試、讀取結果和完成測試的連續處理流程。在實際測試中，選擇了乘法器、計數器狀態機，以及4位元SPI通訊模組作為驗證電路，以確保測試框架的驗證功能。透過實驗，成功處理了與時脈相關的錯誤偵測，驗證了測試框架的精確狀態轉變識別能力，並展示了其在除錯數位I/O通訊方面的功能。本研究成功建立了基於PSoC Zynq 7000系列開發板的數位系統測試框架，減少對外部訊號產生與量測設備的依賴，提供一個低成本的數位系統測試平台，以滿足各種數位電路教學和設計的需求。

關鍵詞：數位系統、電路設計、電路測試、可程式系統晶片、四相通訊

Testing Framework Using Programmable System-on-Chip (PSoC) for Digital System Prototyping

***Yi-Wen Chen, Jung-Lin Yang**

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

Abstract

This research centers on educating students in digital system design and introduces a low-cost digital testing framework. A user-friendly and accessible testing framework based on PSoC Zynq 7000 series development board is developed. Our aim is to support students in mastering circuit design and testing skills efficiently, eliminating the obstacles associated with costly equipment demands and specialized training. The study targets two primary objectives. Firstly, both the testing framework and the circuits under test are incorporated into the same PSoC chip, aiming to minimize reliance on expensive external devices. Secondly, common digital system design methods are used to evaluate the practical viability of the testing framework within real-world digital system design education contexts. These work tasks have been subjected to practical testing on the PYNQ-Z2 platform. The program design on the processor side utilizes Python, while the digital system's prototype and testing framework are constructed using Verilog for circuit description. Ultimately, the high-level synthesis is conducted through Xilinx Vivado. The testing framework incorporates EMIO and a quadrature communication protocol, enabling a continuous processing flow encompassing test data writing, test execution, result reading, and test completion. During practical testing, a multiplier, a counter state machine, and a 4-bit SPI communication module are selected as validation circuits to ensure the verification function of the testing framework. Through

Received: Aug. 22, 2023; first revised: Oct. 27, 2023; second revised: Dec. 13, 2023; accepted: Jan. 2024.
Corresponding author: Y.-W. Chen, Department of Electronic Engineering, Southern Taiwan University of Science and Technology, Tainan 710301, Taiwan.

experimentation, clock-related error detection is effectively handled, the testing framework's ability to identify precise state change is validated, and its exceptional performance in debugging digital I/O communication is demonstrated. This study has successfully established a digital system testing framework using the PSoC Zynq 7000 series development board. This reduces the reliance for external signal generating and measuring equipment, resulting in a cost-efficient educational platform. This platform caters to a range of digital circuit education and design needs.

Keywords: Digital systems, Circuit design, Circuit testing, PSoC, 4-phase protocol

壹、研究背景

隨著電路越來越複雜，即使將所有狀態進行測試，還是有可能無法檢測到某些故障，透過 BIST 設計自我檢測，並找出其故障，使電路測試更容易、更快、更高效、更便宜，也是目前最為常見的測試方式。在 BIST 中分為記憶體 BIST（MBIST）與邏輯 BIST（LBIST）兩種，MBIST 主要用於測試設備的內部記憶體，而 LBIST 為內置電路，作為封裝後測試晶片的結構完整性，而整體架構可拆為即測試碼型發生器（TPG）、底層測試電路（CUT）和輸出回應分析器（ORA），架構如圖 1，TPG 為資料產生器，通常使用 LFSR 來設計，有效提高資料產生的速度，與降低 BIST 的使用資源，CUT 則為被測電路，最後 ORA 會將 CUT 輸出的結果進行分析[1]。除了 BIST 的架構設計外，故障覆蓋率也是在電路測試中重要的指標，使電路測試能得到更全面的檢測，由 Arbab Alamgir 等人所提出的論文中指出，將傳統測試模式生成架構中的第一級與第二級的 LFSR，改為三位元 LFSR 與五位元 LFSR，經過測試後較比傳統的故障覆蓋率增加了 85.06%[2]。此外在 CUT 中插入量測點也能有效提高故障覆蓋率，插入點分為控制端與 CUT 內部節點，在控制端可加入 AND Gate 將訊號拉出，而 CUT 內部節點則需加入 XOR Gate 將訊號拉出[3]。在測試電路的方法中，不只在 BIST 中可以看到多篇論文，對於提升資料產生的速度與故障覆蓋率增的研究，在於 ATPG 的測試方法也是如此[4–5]。

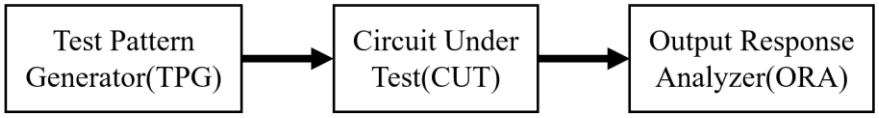

圖 1 *BIST 架構*

人工智慧（AI）作為現今科技發展中的關鍵技術之一，其在推論速度和服務吞吐量方面的需求逐漸攀升。使用現場可程式化邏輯閘陣列（FPGA）進行設計更是一項重要趨勢[6–8]，對比了 CPU、GPU 和 FPGA 在影像處理速度方面的性能，實驗數據顯示，FPGA 在較複雜的函數中相較於 CPU 和 GPU 能更快地獲得結果[9]。Zynq 7000 系列是 Xilinx 設計的一種可程式化系統單晶片（PSoC），在 Xilinx Zynq 7000 SoC 資料手冊[10]中有所介紹。該晶片的處理系統端（PS）採用 ARM® Cortex™-A9 作為核心，提供豐富的外部輸入輸出（I/O）介面和功能；而可程式化邏輯端（PL）則使用 28nm 製程。本論文中，使用 PYNQ-Z2 構建了數位系統測試平台，該開發板的核心是 Zynq 7020，其中主要硬體資源包含有：可程式邏輯單元（Programmable Logic Cells） 85000 個、查詢表單元（LUTs）53200 個，正反器（Flip-Flops） 106400 個、區塊隨機存取記憶體（Block RAM）4.9 Mb、及數位訊號處理單元（DSP Slices） 220 個。PYNQ-Z2 是當今許多研究者和初學者使用的開發板，不僅僅因為其搭載了 Zynq 7020 核心，在 PYNQ-Z2 Mouser 參考手冊中有詳細的技術解說[11]，該開發板搭載 512 MB 的 DDR3 內建記憶體，同時提供多種 I/O 介面，包括 USB、3.5mm 音源接頭、HDMI、Pmod、Raspberry Pi GPIO 介面、Arduino GPIO 介面以及基本的開關和顯示元件。在開發板中，PS 的 ARM 處理器時脈頻率為 650MHz，而 PL 的內建時脈

頻率為 125MHz。在軟體方面，官方提供的映像檔中，可以透過 Jupyter Notebook 編寫 Python 程式來開發 PS 與 PL 的軟硬體協同設計應用。就軟體執行速度而言 Python 並不出色，但在系統雛形及測試的開發階段，Python 能夠更迅速地實現原型設計，對於沒有程式經驗的初學者而言，Python 能夠更快速地幫助他們理解和吸收設計的概念。

貳、研究方法

雖然當今 IC 設計流程已穩定發展多年，但在驗證和測試方面，仍需耗費大量時間來確保其功能和效能，以避免在 IC 投產時出現，功能不如預期或者良率過低的問題。常見的測試驗證方法通常會使用到各種設備，如訊號產生器、檢測設備和微處理器等，這些設備被用來輔助進行測試，以生成必要的訊號。這種測試方法對於 IC 模組，的驗證準確度存在一定程度的疑慮，為同時確保驗證的準確性和效率，通常只能購買昂貴的自動化測試設備，然而這樣的配置需求，在一般的教學場域中很難實現。本研究的主要目的，是希望在 PSoC Zynq 7000 系列開發板上，建立數位系統測試平台輔助電路設計教學，幫助學生更迅速地掌握數位系統設計及測試的關鍵技術。為此需要達成以下兩點目標：1.實現設計模組載入 PSoC 晶片內部測試，減少對外部測試設備的需求；2.實踐驗證數位系統常見的設計範例，確認本文所提出的測試框架（testing framework），能有效應用於數位系統設計實務教學。第二章將分為兩節，分別探討數位系統雛形測試框架和模組規劃，以說明如何實現晶片內部測試。第一節將介紹所選用的開發板，並描述測試框架的架構和設計。第二節將詳細說明測試框架中的，評估模組（evaluation blocks）和記憶體的設計和規劃。第三章將選用數位系統設計實踐中常見的範例電路，驗證本文所提出的測試框架，是否適用於數位系統設計教學。

一、數位系統雛形測試框架

本研究選擇了 PYNQ-Z2 作為測試框架的構建平台，PYNQ-Z2 是基於 Zynq 7020 處理器的 PSoC 開發平台，該平台提供了 Jupyter Notebook 作為 PS 端軟體開發的工具，同時在 PL 端使用 Vivado 作為硬體設計環境。Vivado 中集成了多個智財模組（IP），使開發者能更輕鬆簡單地進行開發。本文提出的測試框架採用 EMIO 作為 PS 端和 PL 端之間的通訊橋樑，EMIO 的配置需要透過 PS 端來完成，它提供了 64 個輸入/輸出介面供開發者使用。當然，也有其他的通訊介面選擇。在 Ramagond 等人的研究中，對 Zynq 7000 中 PS 到 PL 的通訊協定進行了分析，並總結出了 AXI 系列通訊方式的優缺點[12]。目前，測試框架主要使用 EMIO，原因在於其配置簡單且連接方式較為直觀。本研究的目標是在 PYNQ-Z2 最高速度下測試電路，以提供有效的測試資訊供開發者參考。如果測試資料採用 EMIO 逐筆傳送，資料傳輸夾雜在測試過程，資料傳輸延遲一定會成為驗證效率的瓶頸。

因此本研究的測試框架（圖 2）是在 PS 端一次性生成所有測試資料，並將其儲存在 PL 端的記憶體中，等測試資料備齊後再啟動測試。在測試過程中，以即時的方式將擷取測試波形和測試結果，等待測試完成後，同樣以批次傳輸的方式回傳，以提供數位系統開發者進行分析。測試框架中，PL 端的輔助模組包括評估器、記憶體和待測數位系統。其中評估器用於管控 PS 端和 PL 端之間的資料傳輸以及啟動測試。為確保資料流動的正確性，評估器中設計了一個四相通訊協定（圖 3），用於規劃測試資料的寫入、測試、讀取資料以及完成測試等四種狀態，並通過要求（REQ）與回應（ACK）信號來實現通訊。

二、測試框架規劃及設計

在數位系統雛形測試框架的圖 2 中，評估器和記憶體是自行設計規劃的。評估器在此充當了 PS 端和 PL 端之間的通訊模組，運用狀態機實現資料寫入、測試、資料讀取以及重置等四個狀態，正如圖 4 所示。圖中的 Req 是輸入，而 Ack 則是輸出。當模組初始化時，它會進入 S0 狀態，此狀態為資料寫入狀態，在這個階段，PS 端將測試資料逐筆寫入測試記憶體中，寫入完成後，PS 端會將 Req 設定為高電

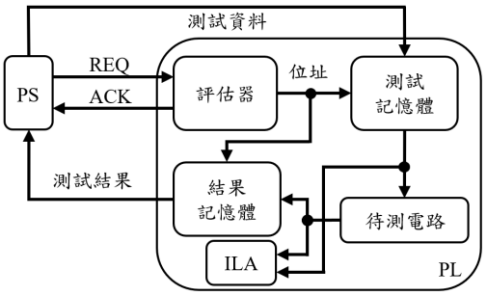

圖 2 *數位系統雛形測試框架*

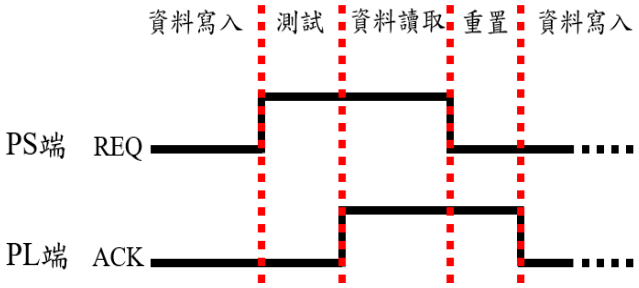

圖 3 *四相通訊協定規劃*

位，使系統進入 S1 狀態。S1 表示測試狀態，此時會讀取在 S0 狀態中寫入的測試資料，然後進行測試，測試結果將被儲存在結果記憶體中。每筆測試資料需要經過兩個時脈周期，分別是測試資料傳送和結果寫入，當測試完畢後，Ack 信號會被設定為高電位，使系統進入 S2 狀態，此狀態為讀取資料狀態。在 S2 狀態中，PS 端將從結果記憶體中讀取資料進行分析，分析完成後，PS 端將 Req 設定為低電位，進入 S3 狀態，此狀態為重置狀態。在 S3 狀態中，評估器將被重新設定，系統將回到 S0 狀態，準備進入下一個測試週期。整個架構中使用了記憶模組，包括測試記憶體和結果記憶體，正如圖 5 所示。這兩個記憶模組都被放置在同一個記憶元件內。為確保資料儲存的正確性，我們在記憶體中設置了三個致能腳位，分別是選擇（cs）、寫入（we） 和讀取（oe）。cs 確保讀寫操作針對正確的記憶體，we 控制記憶體何時寫入資料，oe 則控制記憶體何時讀取資料。此外，記憶體的大小可以根據待測電路的規模進行調整。例如，如果要測試四輸入的組合邏輯電路，需要測試所有可能的 4 個位元組合，那麼所需的記憶體大小就是 4 x 16，通過簡單的程式修改即可調整適用的記憶體大小。同樣地，可以透過修改評估器的控制腳位來增加記憶體的數量，從而提升測試框架的擴充性。

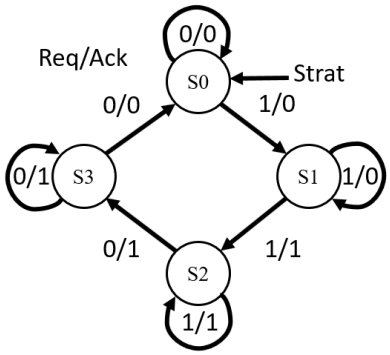

圖 4 *評估器狀態圖*

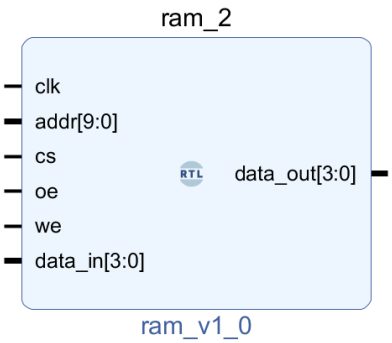

圖 5 *記憶體 IP 模組*

參、實驗與結果

近年來的數位系統設計實務課程中，雖然設計工具的使用方式有很大的改變，通常朝向提升設計便利性和電路規模。然而，值得關注的是，數位系統設計的基礎內容卻幾乎沒有太大的變革,這點可以從 Guoping Wang 與 Veer 所設計的課程規劃中得到確認[13–14]。在本研究中，我們選擇了這兩篇課程規劃中的一部分具代表性的電路設計，例如乘法器和狀態機，來驗證本文提出的測試框架。此外，我們還通過設計一個簡單的通訊模組，使用 4 位元 SPI 介面，用以確認測試框架能處理數位系統教學所需的電路規模。關於測試框架的應用與操作流程，大致可分為四個部分，如圖 6 所示。首先是數位電路設計階段，在這一步驟中，我們完成並模擬了待測的數位電路。接下來，是測試框架的導入階段，我們需要考慮

記憶體大小，然後將待測電路置入到測試框架中，並完成所有必要的連接。接著是硬體描述語言的高階合成階段，在這個階段中，我們使用 Vivado 來生成軟體端所需的檔案，包括描述硬體設計的位元流檔（bitstream）和硬體定義文件（hwh file）。最後一步是測試與分析階段，這個階段可以細分為：測試資料生成、執行測試以及測試資料結果蒐集與分析。在最後的階段，我們的操作平台是 PYNQ-Z2 端的 Jupyter Notebook，使用 Python 來生成測試資料、控制測試框架，並進行電路測試結果的分析。

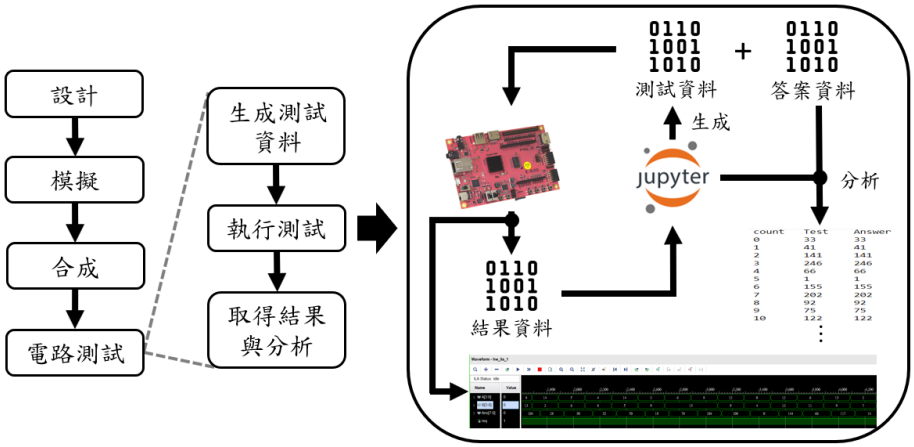

圖 6 *測試框架操作流程*

一、二進位乘法器

乘法器相較於一般的組合邏輯更為複雜，需要更多時間來進行運算。在本研究中，我們參考了移位加法乘法器（Shift-and-Add Multiplication）[15]，來實現圖 7 中的乘法器，以觀察其運算速度。圖中分為五個模組，其中狀態機作為乘法器的核心，來控制整體的運作，運算過程可以拆分為載入、判斷與結束，載入負責將輸入訊號寫入暫存器中，判斷會讀取右移暫存器的最低位元，當數值為 1 時將加法器的結果寫入暫存器中，而數值為 0 時則不寫入，判斷完後不管數值為多少皆要進行位移，在本實驗中為四位元乘法器，所以判斷會執行四次，最後結束時將結果輸出。在這個例子中，我們使用了 35 個 EMIO 輸入輸出接腳，記憶體的大小為 8 x 1000。內部時脈包括 3 個通道，分別是記憶體測試時脈、乘法器運作時脈以及內建整合邏輯分析（Integrated Logic Analyzer, ILA）的取樣時脈。ILA 的取樣資料長度為 65536，擷取的訊號包括 A 端輸入、B 端輸入、輸出結果、Req 信號以及內部運作訊號。合成後，PSoC 的資源佔用狀況如表 1 所示。

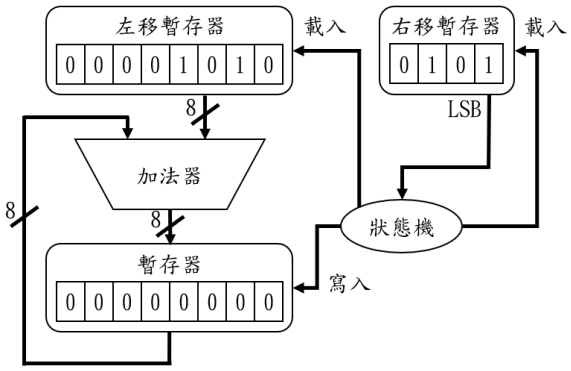

圖 7 *乘法器架構圖*

表 1 *資源佔用表*

資源	使用	可用	使用率%
LUT	1692	53200	3.18
LUTRAM	149	17400	0.86
FF	2457	106400	2.31
BRAM	57	140	40.71

在實驗中，乘法器狀態機內部包含了一個除頻器，並且每次運算需要經過 9 個不同的狀態。因此，若以 100MHz 的時脈運作，單次運算將需要約 360ns 的時間，在本次實驗中，我們將記憶體測試時脈設定為 2.7MHz，乘法器運作時脈設定為 100MHz，並且 ILA 的取樣時脈設定為 200MHz。圖 8 顯示了使用亂數產生的輸入訊號以及相對應的答案，而圖 9 則呈現了測試結果與實際答案之間的比較數據。從圖中

可以觀察到，錯誤的筆數高達 788 筆，這主要是因為乘法器內部並未設置在運算完成後停止運作，這導致部分運算會影響到接下來一筆資料的擷取結果。通過觀察擷取的波形圖 10，我們可以看到每筆資料雖然都在運算完成後才被擷取，但大部分的結果是在接近下一筆測試資料時才完成運算，這導致無法正確擷取當筆測試結果。透過這樣一個特意在時序上設計不足的實驗，設計者能利用本文所提出的測試框架，識別出電路錯誤運作的原因，並加以修正。

Number	B	A	Ans
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	4	14	56
6	5	14	70
7	9	10	90
8	10	5	50
9	1	14	14
10	0	7	0
11	6	1	6
⋮			
993	15	11	165
994	9	14	126
995	8	2	16
996	0	11	0
997	7	2	14
998	7	0	0
999	4	0	0

圖 8 *測試時脈 2.7MHz-測試與答案資料*

Number	Test	Result	
0	0	0	
1	0	0	
2	0	0	
3	0	0	
4	0	0	
5	56	56	
6	70	70	
7	90	90	
8	55	50	Error
9	50	14	Error
10	14	0	Error
11	0	6	Error
⋮			
993	12	165	Error
994	165	126	Error
995	16	16	
996	0	0	
997	14	14	
998	0	0	
999	0	0	

Error number 788

圖 9 *測試時脈 2.7MHz-測試結果與答案資料比對結果*

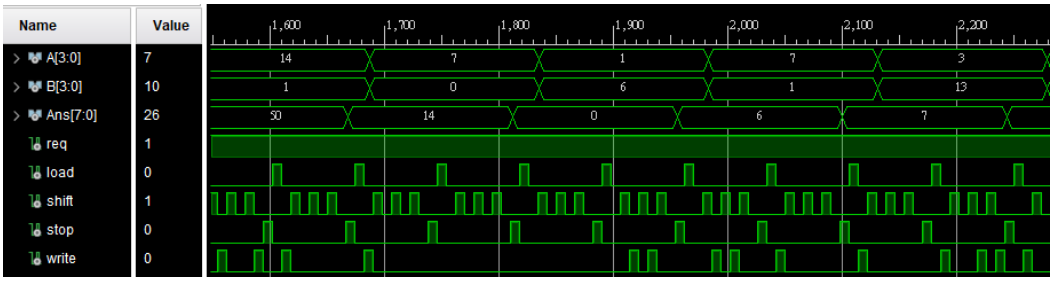

圖 10 *測試時脈 2.7MHz-測試訊號擷取*

在經過多次測試後，我們將記憶體測試時脈降低至 1.4MHz，這樣的調整讓結果記憶體能夠順利擷取到正確的結果。同時，乘法器運作時脈設定為 100MHz，而 ILA 的取樣時脈則設定為 200MHz。圖 11 呈現了本次測試的資料，而圖 12 則顯示了測試結果與答案資料進行比對後的數據，從圖中可以清楚地看到結果欄與答案欄中的數字完全一致，錯誤筆數為 0 筆。在擷取波形圖 13 中，我們可以觀察到乘法器運算完成後，有足夠的時間供記憶體進行資料擷取。透過在 PS 端反覆修改時脈並進行測試，我們不僅節省了硬體合成所需的時間，同時也能更有效地找到模組的最佳運作時脈。

二、計數器狀態機

狀態機的輸出生成不僅受到狀態機的輸入端控制，同時也必須考慮當前的狀態，以便產生正確的輸出。在本研究中，我們設計了一個包含三個狀態的上下計數器狀態機，這些狀態分別是上數、下數和停止[16]。圖 14 展示了計數器的狀態圖，其中輸入端包括清除、致能和控制三個引腳，而輸出則為一個 4 位元的計數器。我們使用這個範例來進行驗證和分析。在這個範例中，我們使用了共計 26 支 EMIO 輸入輸出接腳，記憶體的大小為 8x1000。內部時脈包括 3 個通道，分別是記憶體測試時脈、狀態機運作時脈以及 ILA 的取樣時脈。ILA 的取樣資料長度為 65536，擷取的訊號包括清除、致能、控制、輸出結果以及 Req 信號。合成後，PSoC 的資源佔用狀況如表 2 所示。

Number	B	A	Ans
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	7	5	35
6	15	7	105
7	7	6	42
8	9	2	18
9	8	5	40
10	13	11	143
11	10	2	20
⋮			
993	15	3	45
994	13	6	78
995	4	4	16
996	13	5	65
997	7	2	14
998	5	0	0
999	3	8	24

圖 11 *測試時脈 1.4MHz-測試與答案資料*

Number	Test	Result
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	35	35
6	105	105
7	42	42
8	18	18
9	40	40
10	143	143
11	20	20
⋮		
993	45	45
994	78	78
995	16	16
996	65	65
997	14	14
998	0	0
999	24	24
Error number 0		

圖 12 *測試時脈 1.4MHz-測試結果與答案資料比對結果*

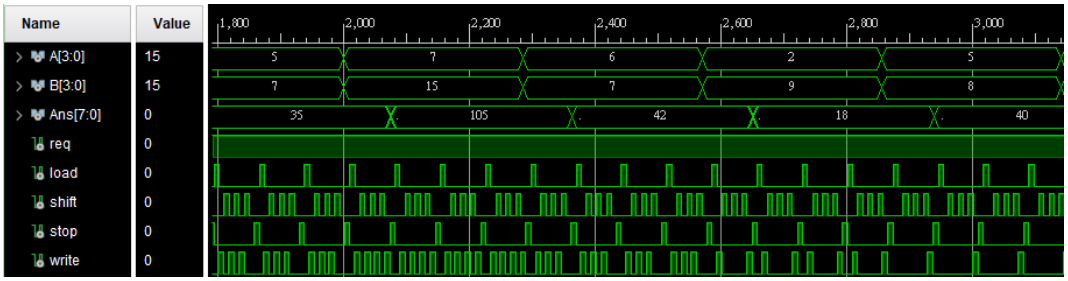

圖 13 *測試時脈 1.4MHz-測試訊號擷取*

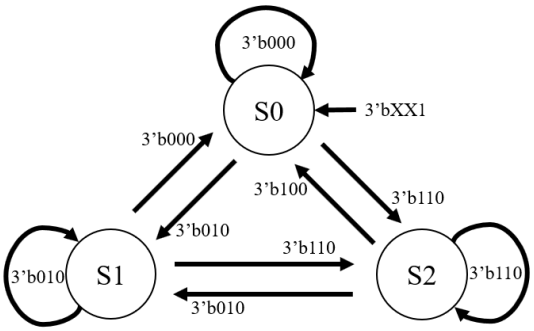

圖 14 *計數器狀態圖*

表 2 *資源佔用表*

資源	使用	可用	使用率%
LUT	1309	53200	2.46
LUTRAM	108	17400	0.62
FF	2065	106400	1.94
BRAM	17	140	12.14

在實驗中，我們將時脈設定如下：記憶體測試時脈設定為 100MHz，狀態機運作時脈設定為 50MHz，而 ILA 的取樣時脈則設定為 200MHz。這些設定使得在測試狀態機時，記憶體測試時脈需為狀態機運作時脈的兩倍，以確保能夠擷取到每一筆變化的資料。圖 15 展示了使用亂數產生的資料，與乘法器範例不同的是，這次資料每十筆才會變化一次，確保能夠觀察到該筆資料的狀態變化。在圖中的答案欄，我們使用狀態的代碼表示，其中 0 代表停止，1 代表下數，2 代表上數。這樣的方式能夠用來分析測試結果是否正確。在圖 16 中，我們展示了分析的結果。在該圖中，Test 欄位表示測試資料，由高到低位元分別為 LW、EN 和 R。而 Result 欄位則是記憶體讀取到的數值，不是狀態本身。因此，透過當前數值與前一次數值的差值，我們能夠得知當前的狀態，通過這樣的分析，我們可以看到錯誤數為 0 筆。此外，從擷取波形圖 17 中，我們也可以確認波形與狀態機的結果是一致的。

Number	LW	EN	R	Ans
0	0	0	1	2
1	0	0	1	2
2	0	0	1	2
3	0	0	1	2
4	0	0	1	2
5	0	0	1	2
6	0	0	1	2
7	0	0	1	2
8	0	0	1	2
9	0	0	1	2
10	1	0	0	2
11	1	0	0	2
⋮				
993	0	0	0	2
994	0	0	0	2
995	0	0	0	2
996	0	0	0	2
997	0	0	0	2
998	0	0	0	2
999	0	0	0	2

圖 15 *測試與答案資料*

Number	Test	Result	Ans
0	1	0	2
1	1	0	2
2	1	0	2
3	1	0	2
4	1	0	2
5	1	0	2
6	1	0	2
7	1	0	2
8	1	0	2
9	1	0	2
10	4	0	2
11	4	0	2
⋮			
993	0	4	2
994	0	4	2
995	0	4	2
996	0	4	2
997	0	4	2
998	0	4	2
999	0	4	2

Error number 0

圖 16 *測試結果與答案資料比對結果*

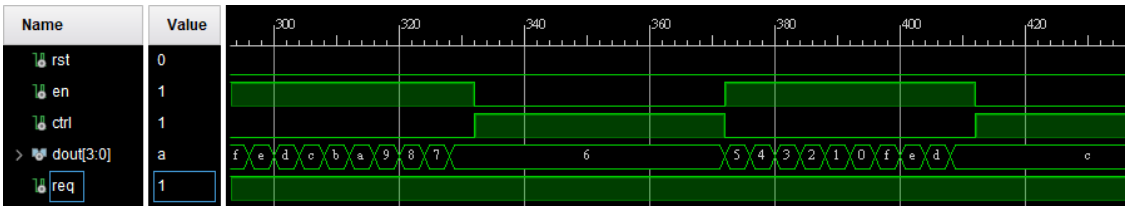

圖 17 *測試訊號擷取*

三、4 位元 SPI 發送/接收模組

SPI（Serial Peripheral Interface）通訊協定是現今廣泛應用於硬體設備的通訊方式之一，主要用於需要高速傳輸的模組，例如攝影機、顯示面板、SD 卡模組等。在本研究中，我們設計了兩個 SPI 模組[17]，分別為 4 位元的主控端和 4 位元的從屬端，使得這兩個模組能互相進行資料的發送和接收（圖 18）。這兩個模組的工作頻率被設計為 4MHz，我們使用這個範例來觀察 SPI 傳輸過程中各腳位的變化。這個範例使用了 37 支 EMIO 輸入輸出接腳，記憶體的大小為 8x1000，內部時脈包括 3 個通道，分別是記憶體測試時脈、SPI 模組運作時脈以及 ILA 的取樣時脈，ILA 的取樣資料長度為 65536，擷取的訊號包括主控端輸出、從屬端輸出、MOSI、MISO、SCLK、SS 以及 Req 信號，合成後，PSoC 的資源佔用狀況如表 3 所示。

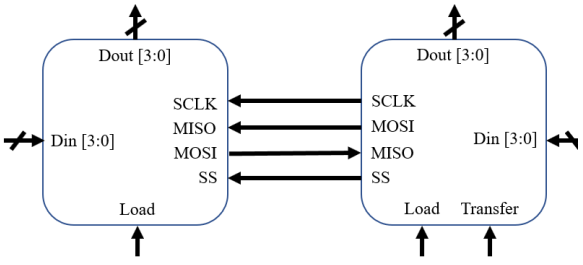

圖 18 *SPI 測試架構*

表 3 *資源佔用表*

資源	使用	可用	使用率%
LUT	1486	53200	2.79
LUTRAM	122	17400	0.70
FF	2258	106400	2.12
BRAM	27	140	19.29

在測試 SPI 通訊模組的過程中，我們將記憶體測試時脈設定為 0.5MHz，並將 SPI 模組運作時脈設定為 100MHz，ILA 取樣時脈設定為 50MHz。在這樣的設定下進行測試，SPI 模組的運作時脈是由 100MHz 經過除頻而來，以確保工作頻率能夠設定在 4MHz。在圖 19 中呈現了測試資料的情況，這些測試資料是由亂數產生，然後輸入到主控端與從屬端中。前 10 筆資料用於模組的初始化，初始化後，每 5 筆資料會進行一次變換，在這 5 筆資料中，第一筆用於載入資料，第二筆用於發送資料，接下來的三筆則是用

於等待，以確保資料能夠成功轉換。圖中的結果展示了主控端與從屬端輸出的結果，這些結果是將從屬端的資料左移 4 位後，再加上主控端的數值所得到的。在發送後，主控端與從屬端的數值將會互換，圖 20 則呈現了分析的結果，我們通過將實際結果與預期答案進行比對，來判斷是否出現錯誤。從圖中可以看到，雖然沒有發現任何錯誤，但卻有 192 個警告，這些警告的產生位置都在 5 筆資料的第二筆，這說明了這些擷取到的數值都是在轉換過程中的數值。最後，圖 21 展示了擷取到的波形，可以清楚地看到各腳位的波形變化以及轉換的過程，從這些波形中可以確認主控端與從屬端的輸出確實已經完成了轉換。

Number	Master	Slave	Ans
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	1	7	113
11	1	7	23
⋮			
993	0	11	11
994	0	11	11
995	0	1	16
996	0	1	1
997	0	1	1
998	0	1	1
999	0	1	1

圖 19 *測試與答案資料*

Number	Test	Result	
0	0	0	
1	0	0	
2	0	0	
3	0	0	
4	0	0	
5	0	0	
6	0	0	
7	0	0	
8	0	0	
9	0	0	
10	113	113	
11	135	23	Warning
⋮			
993	11	11	
994	11	11	
995	16	16	
996	129	1	Warning
997	1	1	
998	1	1	
999	1	1	

Error number 0
Warning number 192

圖 20 *測試結果與答案資料比對結果*

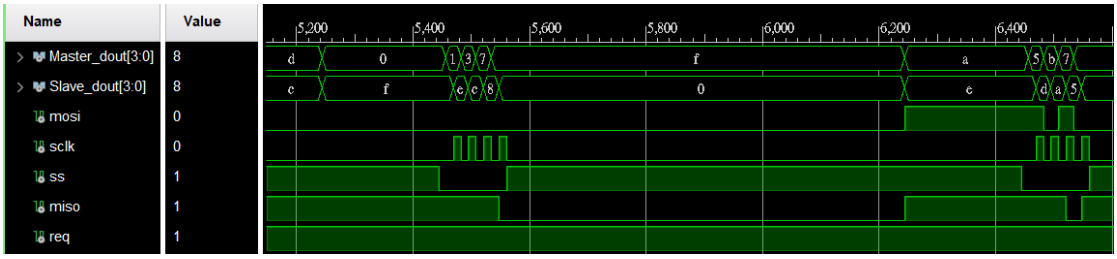

圖 21 *測試訊號擷取*

在本研究的實驗中，我們選擇了乘法器、狀態機和 SPI 通訊模組作為示範案例。首先，我們對乘法器進行了深入探討，乘法器在所有運算元件中具有相對複雜的設計，透過我們實驗中所建構的乘法器測試驗證，每次運算需時 360 奈秒，因此每秒大約能進行 277 萬次運算。在初始的測試中，我們成功找出故意植入的時脈過短錯誤，經過在 PS 端反覆修改測試時脈後，最終的測試結果顯示所有錯誤皆被排除。整個反覆的測試過程並不需要重新合成電路，透過 PS 端的資料分析和波形顯示，我們能精確地分析問題的來源。接下來簡單解析狀態機的實現與偵錯，乘法器中同樣包含了狀態機的設計，但我們將這部分獨立展示，將這個實驗結果專注於狀態機本身。透過我們的測試框架與待測狀態機互動，所收集到的測試結果能在 PS 端進行狀態分析，並進一步評估待測狀態機的準確性。最後，我們將焦點轉至作為本論文驗證用的微型專題之一：SPI 通訊模組，完成了主控端與從屬端的 4 位元模組，並在後續將 SPI 兩模組擴充至 8 位元後，分別與 Arduino Uno 連接，成功實現了與外部微控制器整合的應用。在所有的實驗中，我們充分運用了 PYNQ-Z2 內部的資源，省去了外部接線所帶來的困擾。透過 PS 端產生充足的測試資料，能在不更動測試模組的情況下，重複進行多次測試，而無需重新執行冗長的電路合成過程。以上的實驗結果充分展示了本文提出的測試框架，在數位系統設計實務教學、IC 設計前導課程教學、以及基礎數位電路雛形驗證方面的潛力。

肆、結論及未來展望

進階數位系統電路測試所需設備，通常價格昂貴且需要專業訓練，難以將其納入基礎課程教學場景中作為教學資源。為此，提出了一種低成本的測試框架，並在 PYNQ-Z2 上進行了框架的可用性測試，該測試框架並不局限於 PYNQ-Z2，同樣適用於其他 PSoC 實驗板且可輕鬆移植。本研究成功完成了兩個要務，首先整合了測試框架和測試電路，並燒錄至可程式化系統晶片（PSoC），以減少對外部測試設備的需求。其次，在實際的教學環境中，使用常見的數位系統設計方法，評估了測試框架的可用性和穩定性。在 PYNQ-Z2 平台上完成了實際測試，使用 Python 生成測試資料和結果分析，而測試電路和測試框架則是使用 Verilog 設計描述，透過 Xilinx Vivado 處理高階合成，將合成結果燒錄到實驗板上測試。此外，選擇了 EMIO 作為連接 PS 端和 PL 端的通訊橋樑，並通過四相通訊協定來實現測試資料的寫入、測試執行、結果讀取以及測試完成等四個循環狀態。

在驗證測試實驗中，選擇了乘法器、計數器狀態機以及 4 位元 SPI 通訊模組作為驗證電路，以驗證測試框架的可用性和穩定性。透過這些實驗，成功地處理了與時脈相關的錯誤偵測，並在不需重複冗長高階合成的前提下，完成了這項除錯實驗工作。同時，也藉由計數器狀態以及反推的測試資料，確認了測試框架在狀態機狀態轉變的識別能力。實現了 4 位元 SPI 傳送/接收模組及相關的測試資料，展示了測試框架在數位 I/O 通訊的除錯功能。進一步將經過測試的 4 位元模組擴展成 8 位元，並連接到外部的 Arduino UNO 微控制器，展示了實際應用的可能性。透過這項研究，成功地建立了基於 PSoC Zynq 7000 系列開發板的數位系統測試框架，有效地減少了對外部訊號生成和量測設備的依賴。本研究成果提供了一個低成本的數位系統測試平台，能滿足多種數位電路教學和設計方面的需求。未來，將完成更多的實驗範例，將所有的操作過程及結果整理成數位系統及 IC 設計入門教材，也計劃將測試框架實際移植到不同平臺進行測試，並提供系統移植的參考指引。

參考文獻

[1] Ramya R., Madhura R. (2023). FPGA Implementation of optimized BIST architecture for testing of logic circuits. *SSRG - IJVSP, 7*(2), 36-42. https://doi.org/10.14445/23942584/IJVSP-V7I2P106.

[2] Alamgir, A., A'ain, A.K.B., Paraman, N., Sheikh, U., & Grout, I. (2020). A comparative analysis of LFSR cascading for hardware efficiency and high fault coverage in BIST applications. *2020 IEEE 29th Asian Test Symposium (ATS)* (pp. 1–5). Penang, Malaysia. https://doi.org/10.1109/ATS49688.2020.9301561.

[3] Kostin, S., Orasson, E., & Ubar, R. (2016). A tool set for teaching design-for-testability of digital circuits. *2016 11th European Workshop on Microelectronics Education (EWME)* (pp. 1–5), Southampton, UK. https://doi.org/10.1109/EWME.2016.7496466.

[4] Das, K.S., & Zala, A. (2020). Optimizing cell-aware ATPG pattern volume to keep test cost competitive. *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)* (pp. 1–6), Bangalore, India. https://doi.org/10.1109/CONECCT50063.2020.9198358.

[5] Renovell, M., Portal, J.M., Faure, P., Figureras, J., & Zorian, Y. (2001). A discussion on test pattern generation for FPGA—implemented circuits. *J. Electron. Test., 17*, 283–290. https://doi.org/10.1023/A:1012219513510.

[6] Wu, C.-B., Hsiao, Y.-K., & Chang, W.-H. (2022). Extensible and modularized processing unit design and

implementation for AI accelerator. *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (pp. 238–241). Incheon, Republic of Korea. https://doi.org/10.1109/AICAS54282.2022.9869949

[7] Perepelitsyn, A., Fesenko, H., Kasapien, Y., & Kharchenko, V. (2022). Technological stack for implementation of AI as a service based on hardware accelerators. *2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT)* (pp. 1–5), Athens, Greece. https://doi.org/10.1109/DESSERT58054.2022.10018615.

[8] Qin, H., Zeng, Y., Bai, J., & Kang, W. (2023). Searching tiny neural networks for deployment on embedded FPGA. *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (pp. 1–5). Hangzhou, China. https://doi.org/10.1109/AICAS57966.2023.10168571

[9] Qasaimeh, M., Zambreno, J., Jones, P.H., Denolf, K., Lo, J., & Vissers, K. (2019). Analyzing the energy-efficiency of vision kernels on embedded CPU, GPU and FPGA platforms. *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)* (pp. 336), San Diego, CA, USA. https://doi.org/10.1109/FCCM.2019.00077

[10] Xilinx (2018). *Zynq-7000 SoC data sheet: Overview.* https://docs.xilinx.com/v/u/en-US/ds190-Zynq-7000-Overview

[11] Mouser (2018). *PYNQ-Z2 reference manual v1.0.* https://www.mouser.com/datasheet/2/744/pynqz2_user_manual_v1_0-1525725.pdf

[12] Ramagond, S., Yellampalli, S., & Kanagasabapathi, C. (2017). A review and analysis of communication logic between PL and PS in ZYNQ AP SoC. *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)* (pp. 946–951), Bengaluru, India. https://doi.org/10.1109/SmartTechCon.2017.8358511.

[13] Wang, G. (2005). Teaching digital logic design using CAD tools in a teaching-oriented university. *2005 ASEE Annual Conference & Exposition: The Changing Landscape of Engineering and Technology Education in a Global World* (pp. 2005), Portland, OR, USA.

[14] Veer (2022). Digital system design pdf–Digital system design notes and study material PDF free download. https://btechgeeks.com/digital-system-design-notes/?fbclid=IwAR1wkCX7MQVLA85SggrVxcBsMjw2PCxmTcOvEalIVUiNCPXV2K9psE_3t-A

[15] Dempster, A., & Macleod, M. (1994). Constant integer multiplication using minimum adders. *IET Circuits Devices Syst., 141*(5), 407–413. https://doi.org/10.1049/ip-cds:19941191

[16] 林灶生、劉紹漢（2007）。**Verilog FPGA 晶片設計**，全華圖書股份有限公司。

[17] Aykenar, M.B., Soysal, G., & Efe, M. (2020). Design and implementation of a lightweight SPI master IP for low cost FPGAs. *2020 28th Signal Processing and Communications Applications Conference (SIU)* (pp. 1–4), Gaziantep, Turkey. https://doi.org/10.1109/SIU49456.2020.9302434