Homework 4: In-System Debugging Using ILA Core

資訊所 P76134082 陳冠言、醫資所 Q56134102 王宇軒

Part 1 : ILA Core 使用前設定及實作

本次實驗的目標是透過 Vivado 平台，實作並驗證 SPI Controller (FSM) 的運作，並準備使用 ILA Core (Integrated Logic Analyzer) 進行 In-System Debugging，觀察訊號波形與時序。SPI (Serial Peripheral Interface) 是一種同步串列通訊協議，主要特點包括：主設備 (Master) 與從設備 (Slave) 間的通訊、使用 SPI Clock (SCK) 控制數據傳輸時序、數據透過 MOSI (Master Out, Slave In) 及 MISO (Master In, Slave Out) 進行傳輸。

本作業的 SPI Controller 主要功能為透過 clk_10 產生 10MHz SPI Clock，並由 CE 控制輸出。FSM 設計特點如下：

1. 本設計以四個零填充 4 位元輸入，從而實現 8 位元傳輸
2. 狀態轉換發生在 10MHz 時脈的下降沿
3. 控制器提供完成訊號（done_send），該訊號保持高電平直到下一次操作
4. 重置機制允許初始化或錯誤恢復

此 SPI 控制器設計為實驗報告後續部分的系統內調試實驗奠定了基礎。

Part 2 : ILA Core 軟體設定實際操作

在 Vivado 中完成專案 synthesis 後，可以從左側選單存取「Set up Debug」選項，對於啟用在系統偵錯功能至關重要，如圖 1 所示。

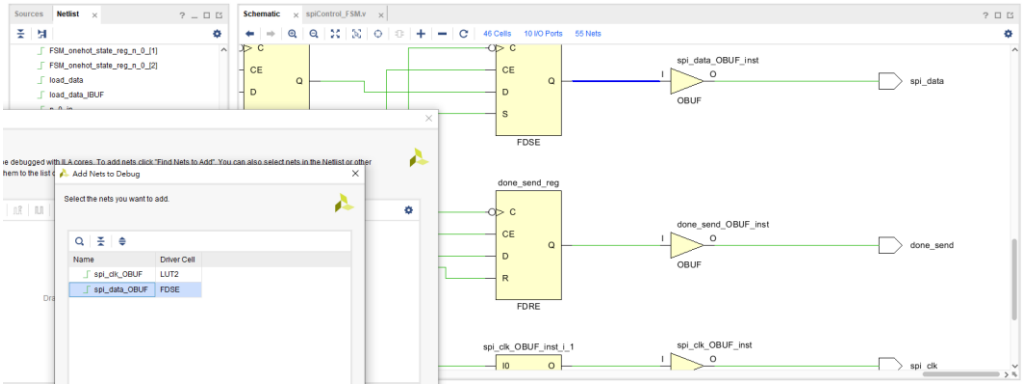

圖 1

設定 ILA 核心時，識別想要觀察的內部訊號是非常重要步驟。這些訊號不會直接被稱為輸出端口，而是緩衝區之前的線。例如，要觀察時脈訊號，就需要手動新增它，因為它是外部電路（如圖 2 所示）。

另外，樣本資料深度是指 ILA Core 可以捕獲的樣本數量。在這種情況下，深度我先設定為 2048，這意味著 ILA 最多可以儲存 2048 個資料樣本以供分析。此設定對於擷取足夠的資料來分析 SPI 控制器隨時間的行為，也可以根據不同的需求做調整（如圖 3 所示）。

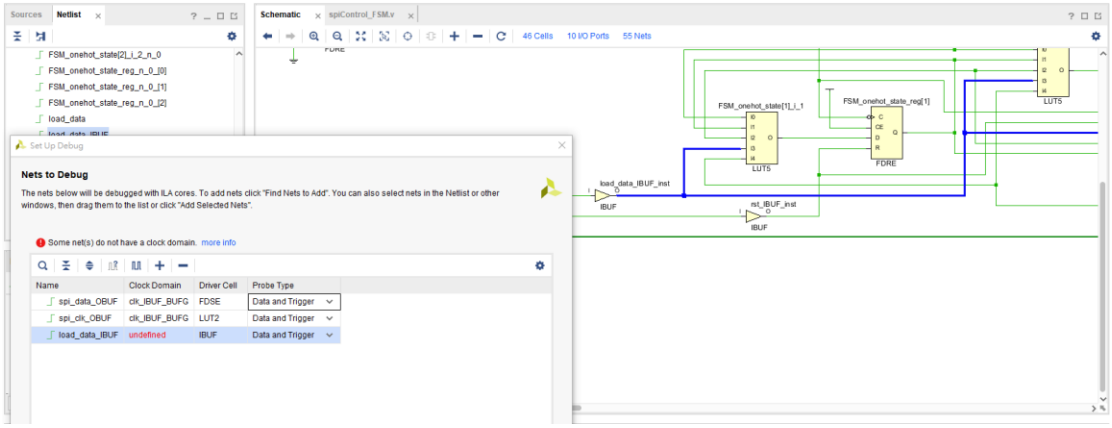

圖 2

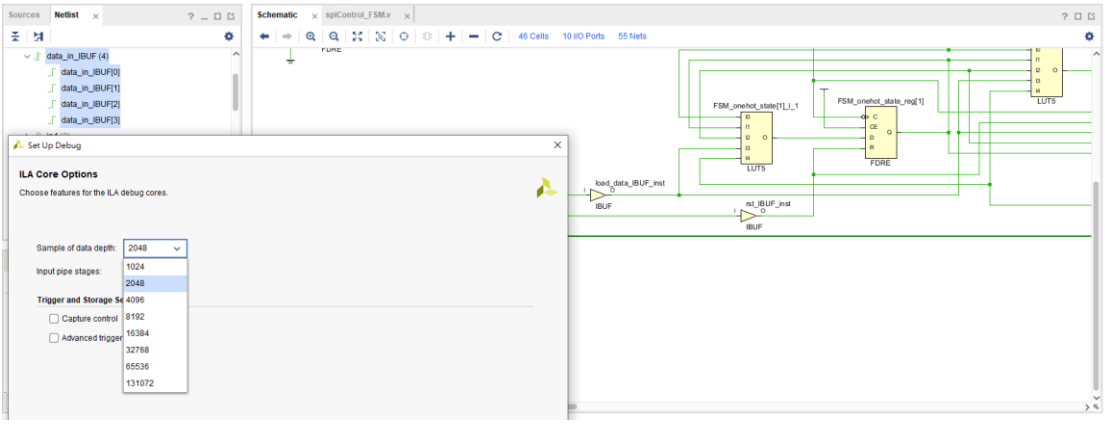

圖 3

Part 3 : ILA Core 硬體設定實際操作

在軟體中設定 ILA Core 後，Vivado 會產生一個包含偵錯參數的約束檔 (.xdc)。該檔案確保調試核心在硬體中正確實例化。此檔案配置 ILA 核心以捕捉時脈訊號、資料輸入和其他相關訊號，以便進行調試。另外，我也根據作業中的 PDF 檔案來建立外部控制 I/O，在 xdc 檔案可反映這些連接，確保 ILA Core 可以正確地與外部環境互動（如圖 4 所示）。

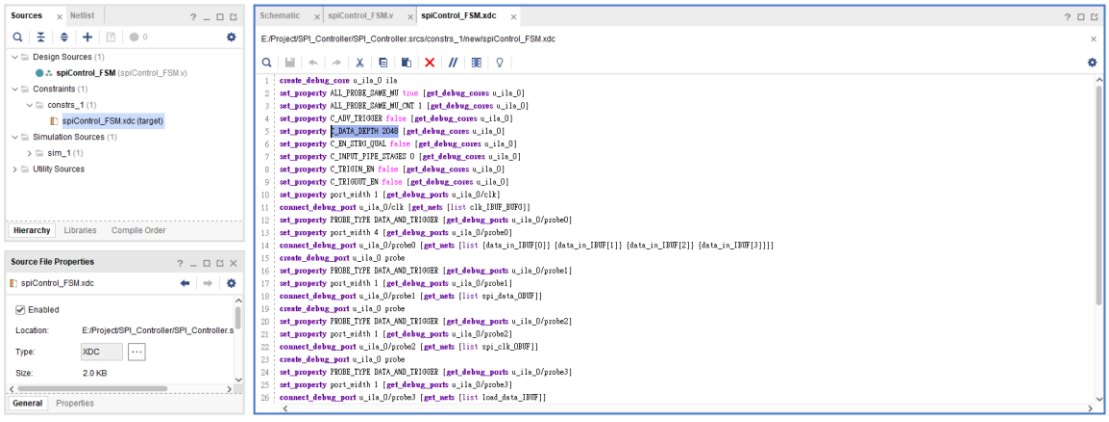

圖 4

Part 4 : ILA Core Debugging

在此部分，本作業設定了兩種操作模式：一種是 ILA 無需觸發即可連續擷取資料；另一種是僅在滿足特定觸發條件後才擷取資料。而 ILA 無需觸發即可連續擷取資料的部分如圖 5 所示。

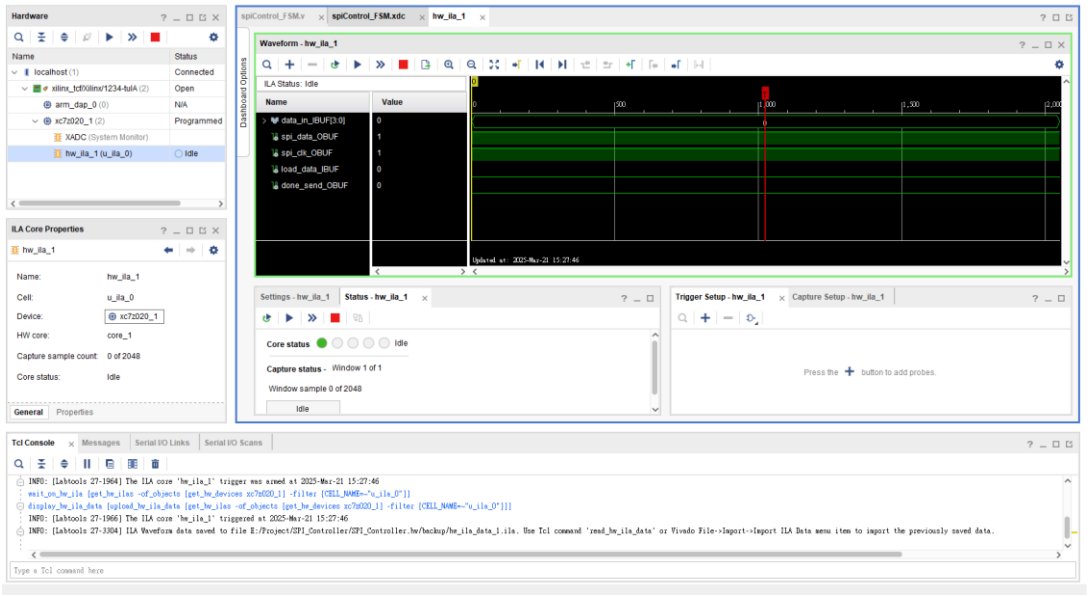

圖 5

而訊號觸發後執行的部分，主要會根據設定觸發點及觸發條件有不同的效果，圖 6 和圖 7 的部分就是本作業在設定觸發點及觸發條件。其中的參數 Window data depth (視窗數據深度)代表這次擷取總共會記錄 2048 個時脈週期的數據；而 Trigger position in window 代表觸發點在視窗中的位置，目前設為 1024，代表擷取數據時，觸發會出現在中間 (第 1024 個樣本位置)。這樣可以同時看到觸發點前後的波形，用來分析觸發事件發生前後的行為。

另外本作業也有嘗試不同位置的觸發點，如 **0**：觸發點在最開頭，只能觀察觸發後的行為、**2048**：觸發點在最末端，只能觀察觸發前的行為、**1024 (中間)**：可以同時看到觸發點發生前後的情況，通常是最有用的設定、**256**：觸發點在 1/8 的位置，並觀察它的行為（如圖 8 所示）。

在 Trigger Setup 視窗中，load_data_IBUF 被設為觸發條件，並且選擇了 R (0-to-1 transition)。這代表 R (0-to-1 transition) 是觸發條件，表示當 load_data 訊號從 0 變成 1 時，ILA 會開始擷取波形。這通常表示某個事件的開始，例如 load_data = 1 可能表示 SPI 傳輸開始，所以希望擷取從這個時刻開始的訊號變化。

另外，如果選擇： F (1-to-0 transition) → 當訊號從 1 變 0 觸發擷取(如圖 10、圖 11)。 B (both transitions) → 無論 0 → 1 或 1 → 0 變化都會觸發擷取(如圖 12、圖 13)。 X (don't care) → 這個訊號不會作為觸發條件。 N (no transitions) → 這個訊號完全不參與觸發判斷。這些操作都是很常見的數據擷取技巧，能夠讓自己清楚觀察 FPGA 內部邏輯在特定事件發生時的波形變化。

實作系列實驗觀察 spi_data/send 對於波型的貢獻，分別控制 data_in[0]，data_in[1]，data_in[2]，data_in[3]觀察波型的變動。

圖 14 : 無 data_in

圖 15 : data_in[0] = 1

圖 16 : data_in[1] = 2

圖 17 : data_in[2] = 3

圖 18 : data_in[3] = 4

圖 19 : data_in[0] + data_in[1] = 3

圖 20 : data_in[0] + data_in[1] + data_in[2] = 6

圖 21 : data_in[0] + data_in[1] + data_in[2] + daia_in[3] = 10

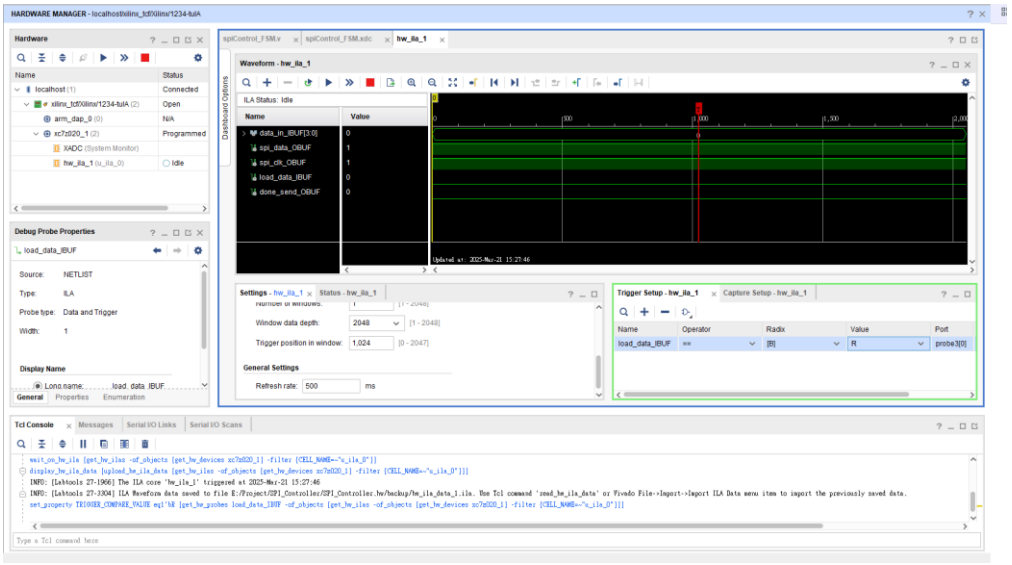

圖 6

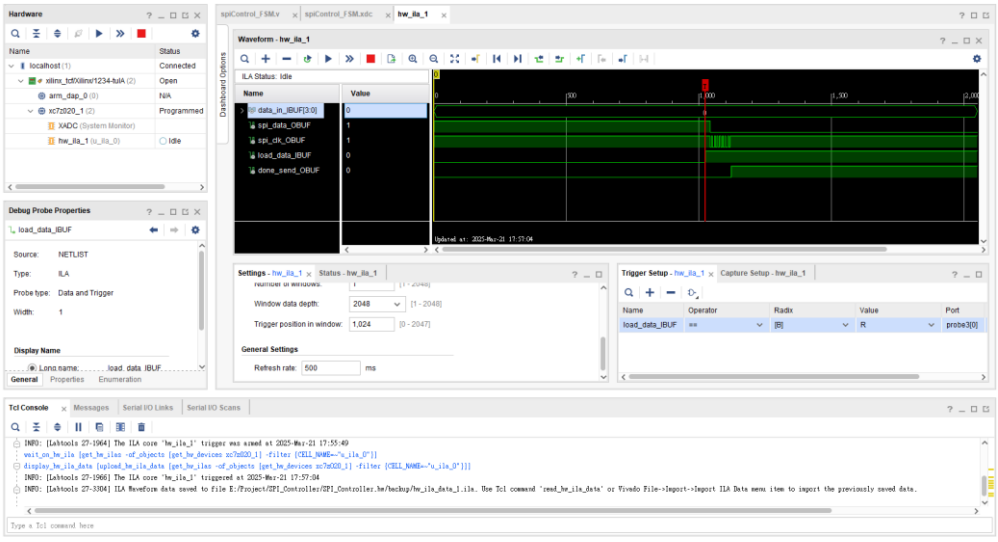

圖 7(R:0 → 1)

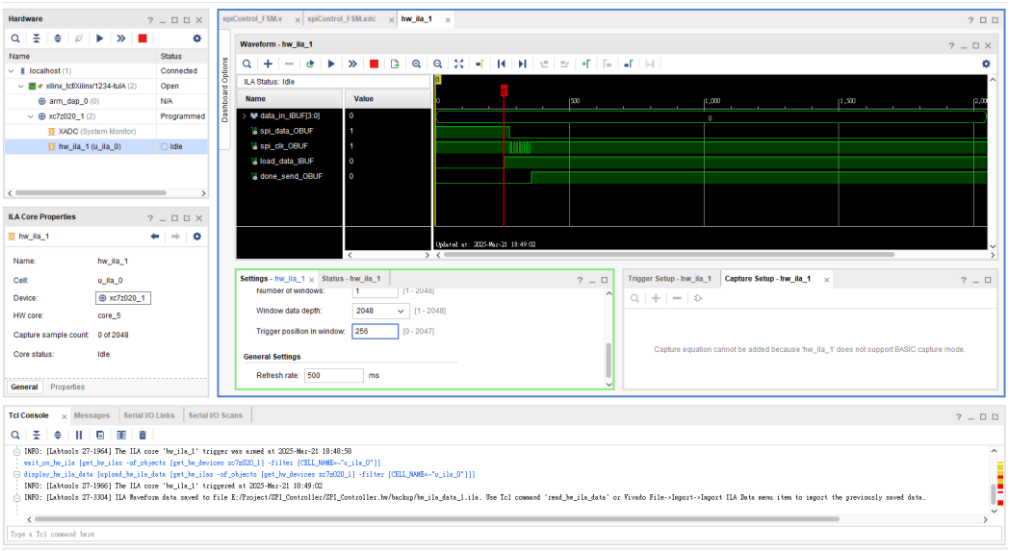

圖 8

(若將 cycle 的比例使放大即可查看到更細節的訊號 wave 如圖 9。)

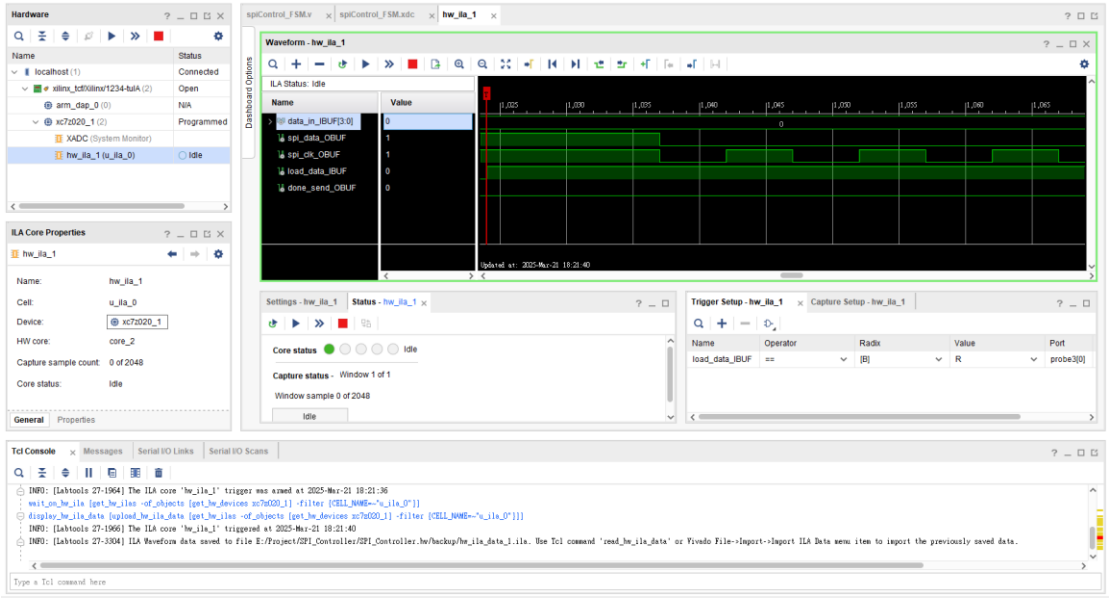

圖 9

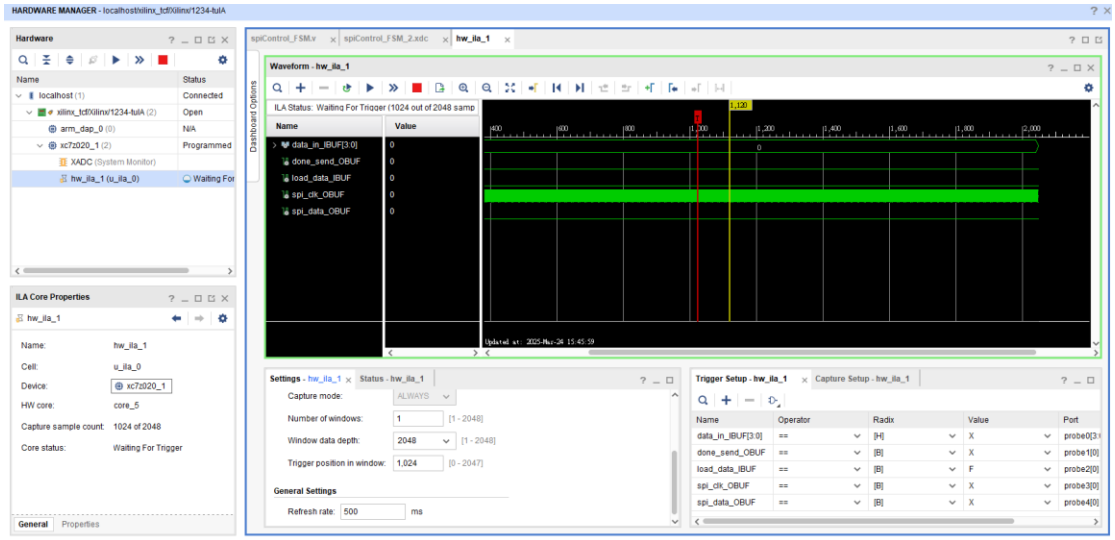

圖 10(F:0 → 1)

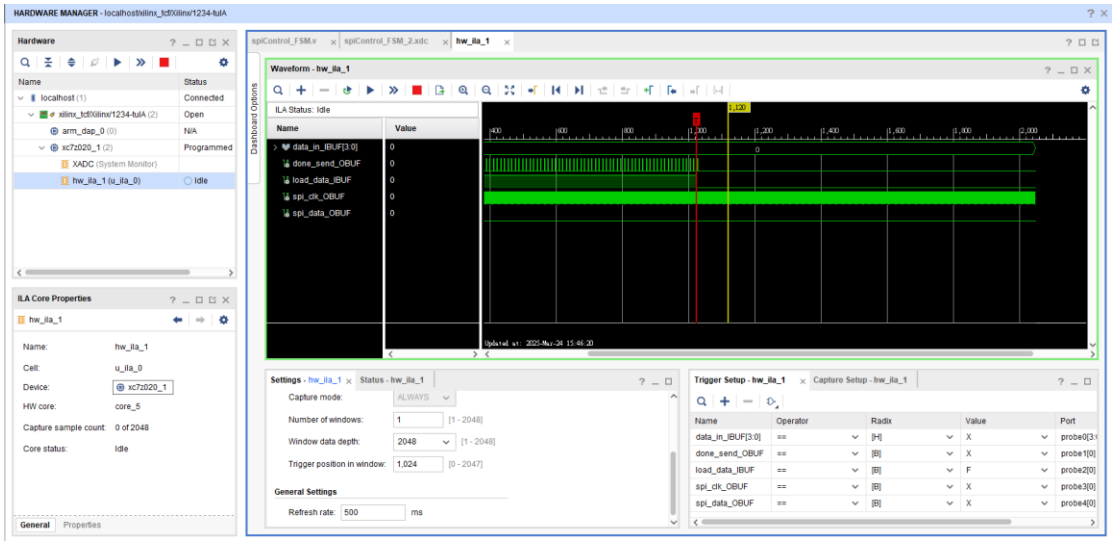

圖 11(F:1 → 0)

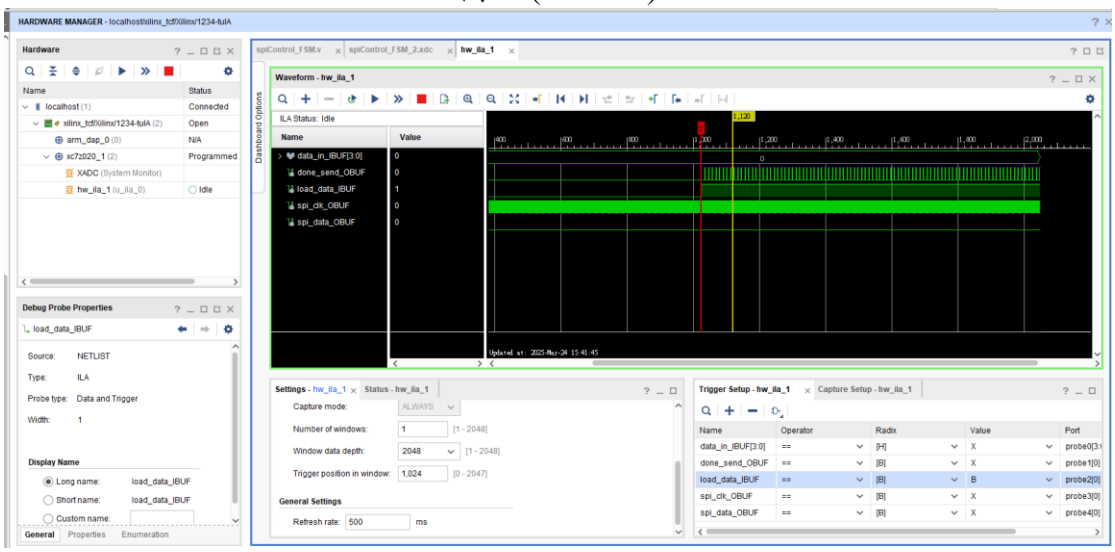

圖 12(B:0 → 1)

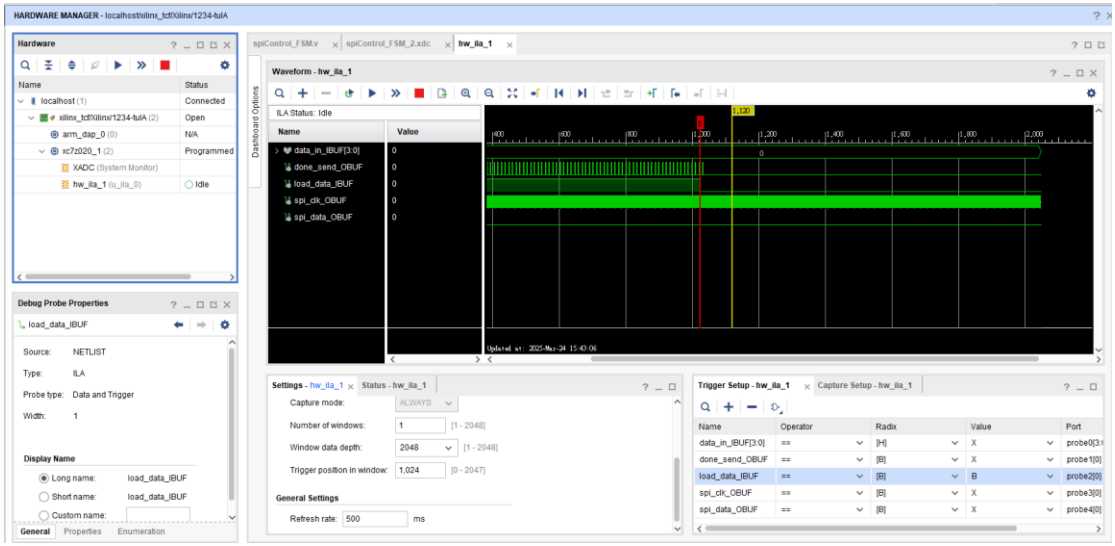

圖 13(B:1 → 0)

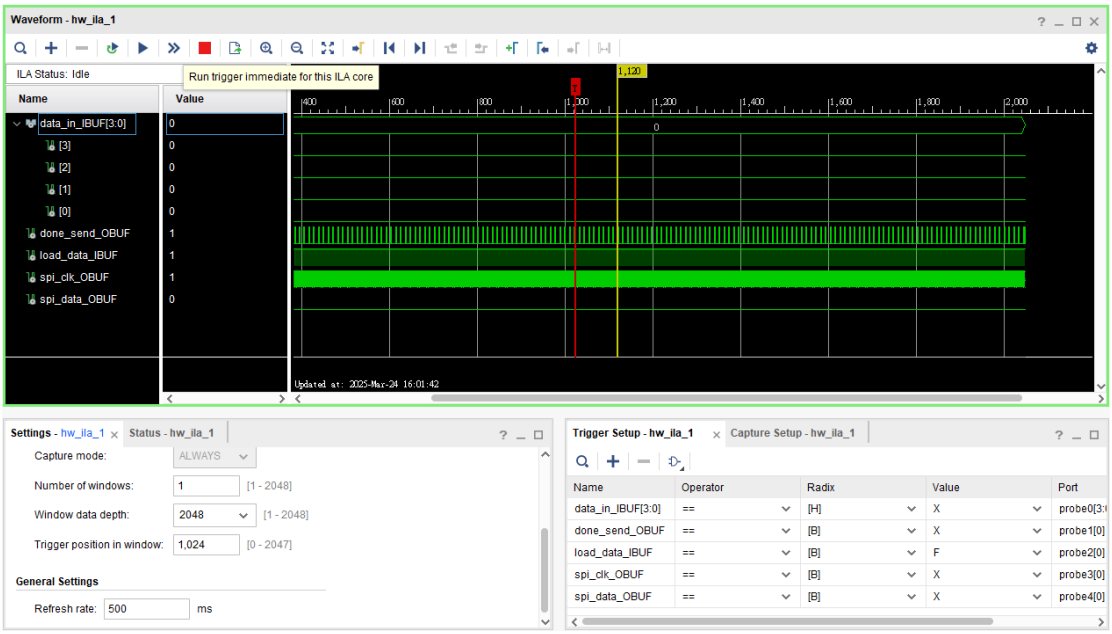

圖 14 : 無 data_in

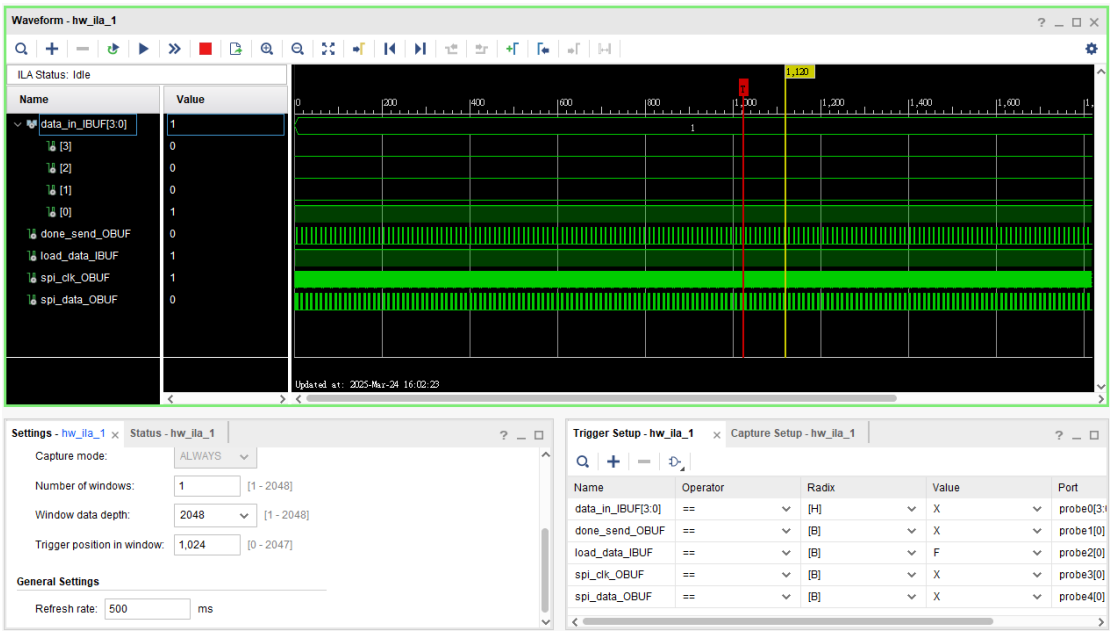

圖 15 : data_in[0] = 1

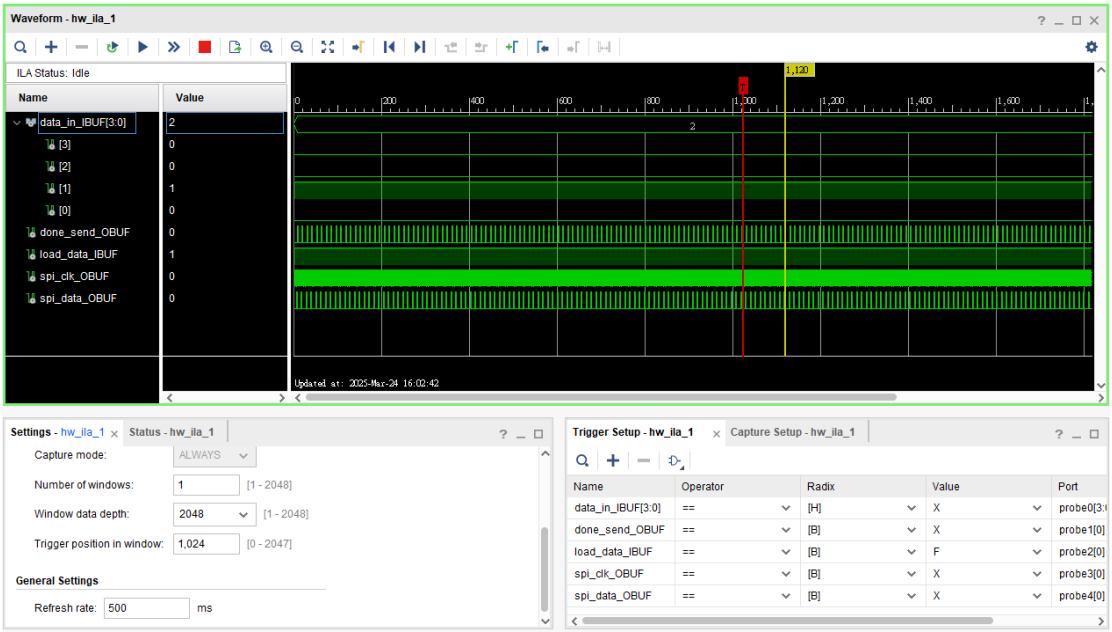

圖 16 : data_in[1] = 2

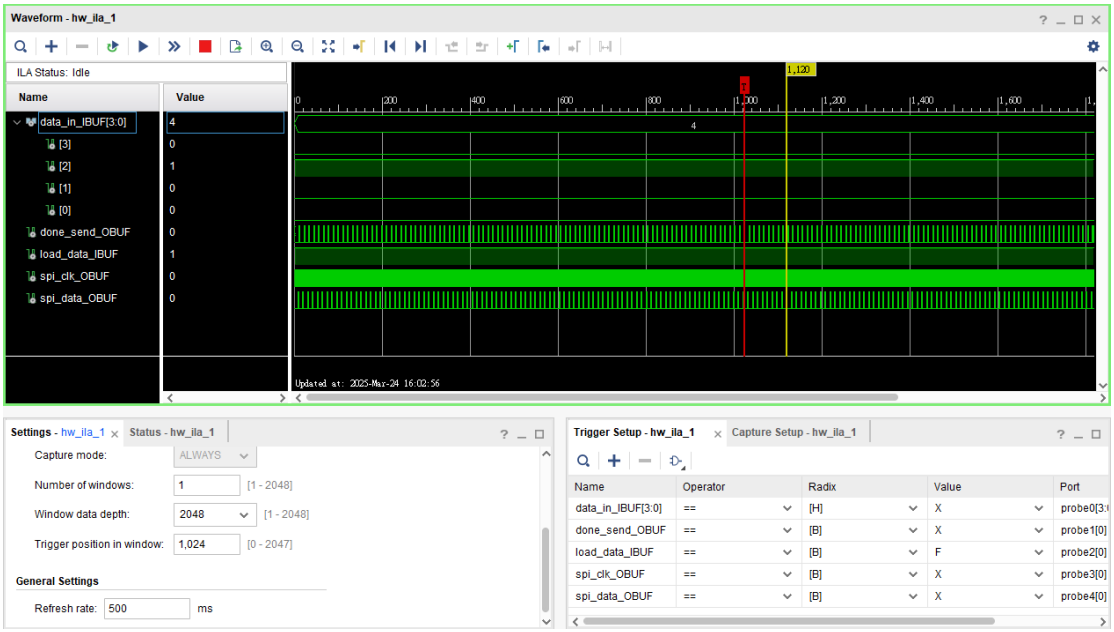

圖 17 : data_in[2] = 3

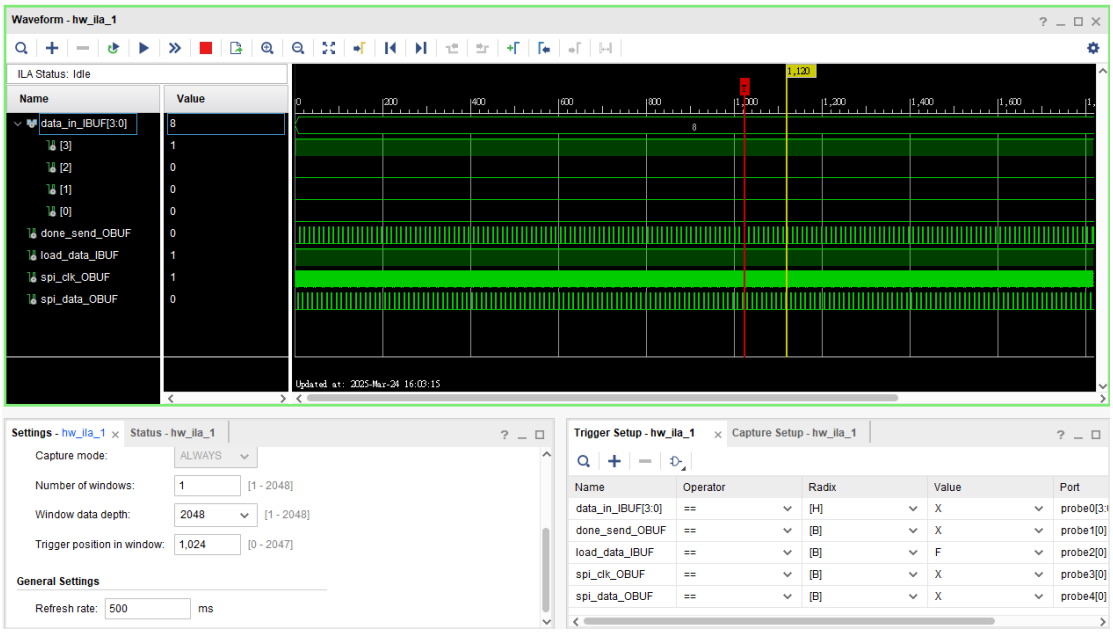

圖 18 : data_in[3] = 4

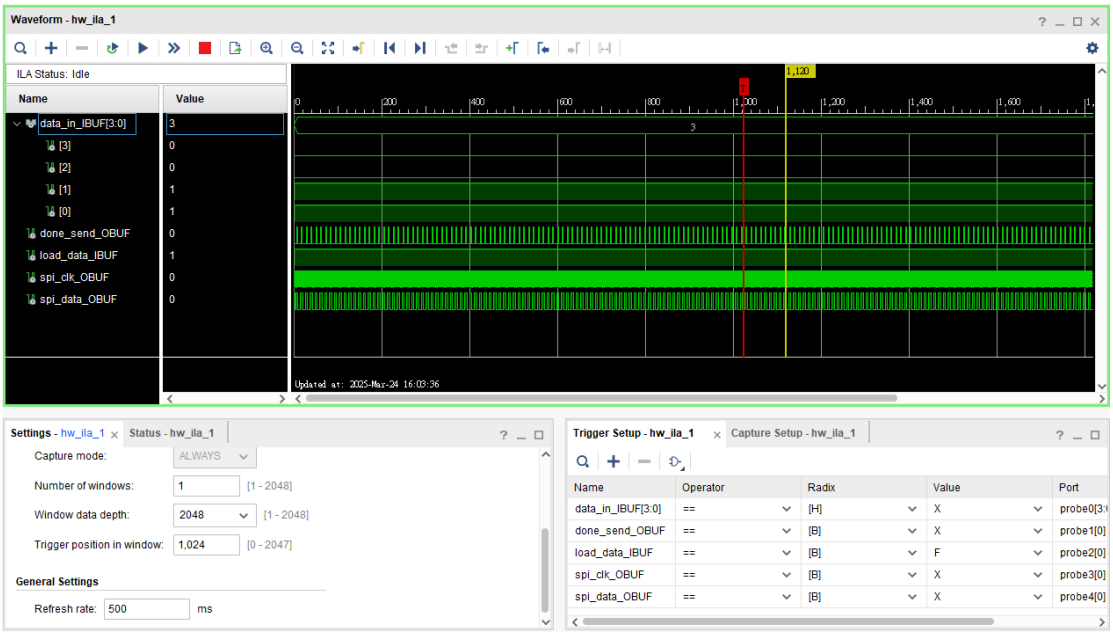

圖 19 : data_in[0] + data_in[1] = 3

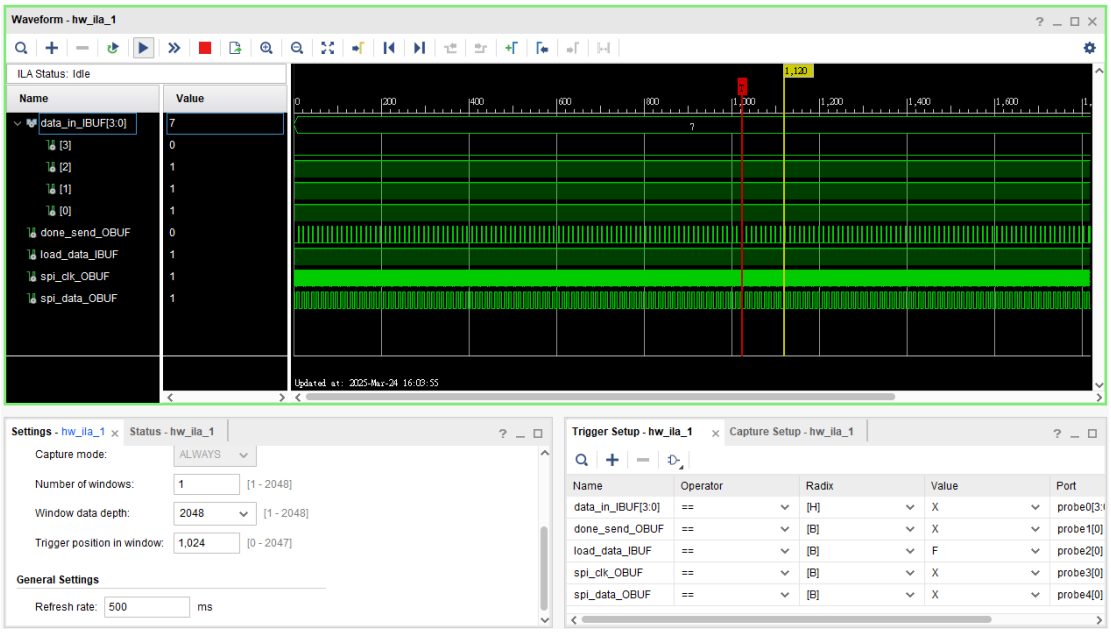

圖 20 : data_in[0] + data_in[1] + data_in[2] = 6

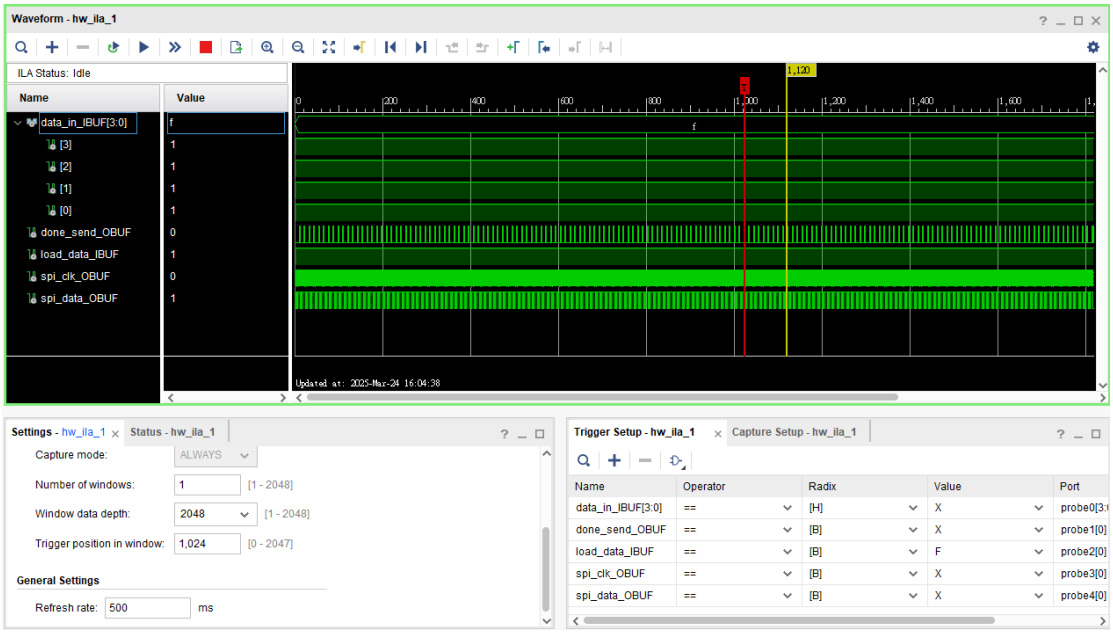

圖 21 : data_in[0] + data_in[1] + data_in[2] + daia_in[3] = 10

這次作業主要學習如何使用 Vivado ILA 來 Debug SPI Controller，並驗證其波形行為。通過不同的測試條件，可以發現 FPGA 設計中的潛在錯誤並進行修正，這對於實際的 FPGA 設計與開發是非常有幫助的。透過本次作業學習到的知識包括以下項目：

1. Vivado ILA 是非常有用的 In-System Debug 工具

它可以讓設計者即時觀察 FPGA 內部信號，而不需要使用外部邏輯分析儀。

可以設定觸發條件來捕捉特定事件，例如 load_data 上升沿。

2. SPI 設計需確保時序正確

spi_clk 需要準確產生 10MHz。

spi_data 需符合 SPI 標準的 MSB-first 傳輸順序。

done_send 需準確標記資料傳輸完成的時間點。

3. 不同觸發條件影響測試結果

使用 R(0-1 transition)、F(1-0 transition) 來觀察不同信號轉變時的行為，可以幫助找到錯誤。如果沒有設置正確的觸發條件，可能會錯過關鍵的波形變化。

4. 對 FPGA Debug 的重要性

透過 ILA，可以在不重新燒錄 FPGA 的情況下即時查看內部訊號，縮短 Debug 時間。透過調整 .xdc 設定和 I/O 連接，可以優化測試結果，提高開發效率。