CAD/VLSI Circuit Design

期末專題報告

2014 IC Design Contest Preliminary

研究所類標準元件數位電路設計

Serial Transmitter and Data Arrange Controller

學生：葉政勳

學號：7109064382

1.前言

此次專題為2014年ic設計競賽題目，用題目之系統架構設計硬體描述語言(verilog)，本次使用Vivado 2019.1作為simulation軟體，並在design compiler上進行90nm製程的合成。

2.簡介

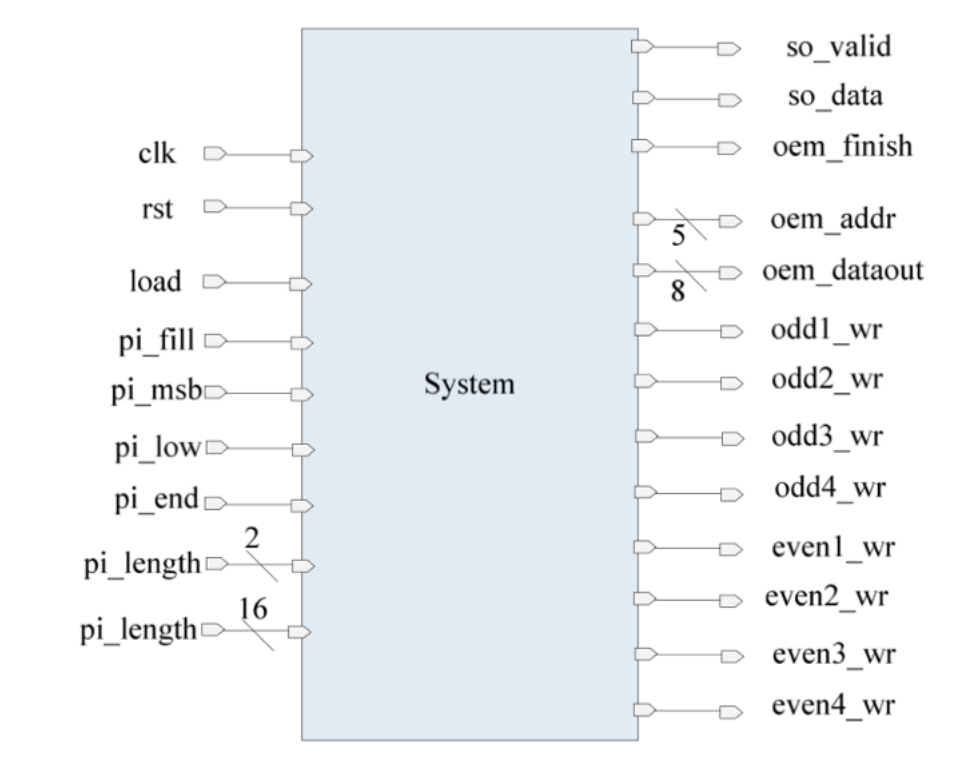
本系統包含一序列傳輸介面處理電路(Serial Transmitter Interface, STI)及一資料排列控制電路(Data Arrange Controller, DAC)。

3.目的

STI 電路動作為從並列埠進行資料輸入處理後由序列埠將處理完成之資料以序列輸出。DAC 電路之功能為將經 STI 電路處理完成後之序列資料進行排列後分別寫入指定記憶體。

4.方法與原理

4.1. 系統方塊圖



4.2. 輸入/輸出

信號名稱	輸入/輸出	位元寬度	說明
clk	input	1	系統提供的時脈信號。
reset	input	1	高位準非同步(active high asynchronous)之系統重置信號。 說明：此信號於系統啟動時送出。
load	input	1	系統提供的讀取控制訊號。 說明：訊號寬度持續一個時脈週期。當 load = 1 時且經時脈訊號正緣觸發時，表示並列資料輸入埠及序列控制訊號為有效。
pi_data	input	16	十六位元並列資料輸入埠。
pi_length	input	2	序列資料輸出長度設定信號。 說明：當此訊號呈現 2'b00 時，表示序列輸出為 8bits 資料輸出。 當此訊號呈現 2'b01 時，表示序列輸出為 16bits 資料輸出。 當此訊號呈現 2'b10 時，表示序列輸出為 24bits 資料輸出。 當此訊號呈現 2'b11 時，表示序列輸出為 32bits 資料輸出。
pi_fill	input	1	序列資料輸出模式設定信號。
pi_msb	input	1	序列輸出順序控制訊號。
pi_low	input	1	序列低位元輸出致能訊號。
so_data	output	1	序列資料輸出埠。
so_valid	output	1	序列資料輸出致能訊號。 說明 ： 當此信號為 1 時，表示 so_data 傳輸的資料被認為是有效的。

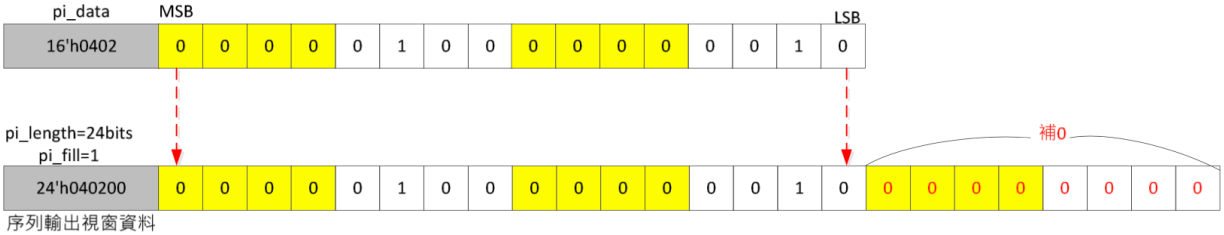
pi_end	input	1	並列資料輸入結束旗標。 說明：當此訊號為 1 時，表示測試樣本檔將不再向 STI_DAC 輸入任何資料； 當此訊號為 0 時，表示測試檔樣本仍可能會對 STI_DAC 進行資料輸入。
oem_finish	output	1	OM 及 EM 記憶體共用寫入完成指示訊號。 說明：當記憶體 ODD1_MEM~ODD4_MEM 及 EVEN1_MEM~EVEN4_MEM 完成資料寫入時，將 oem_finish 設定為 1，則測試樣本檔將開始進行驗證記憶體內容；預設值應設定為 0。
oem_addr	output	5	OM 及 EM 記憶體共用五位元位址埠。
oem_dataout	output	8	OM 及 EM 記憶體共用八位元資料埠。
odd1_wr	output	1	ODD1_MEM 記憶體資料寫入致能訊號。
even1_wr	output	1	EVEN1_MEM 記憶體資料寫入致能訊號。
odd2_wr	output	1	ODD2_MEM 記憶體資料寫入致能訊號。
even2_wr	output	1	EVEN2_MEM 記憶體資料寫入致能訊號。
odd3_wr	output	1	ODD3_MEM 記憶體資料寫入致能訊號。
even3_wr	output	1	EVEN3_MEM 記憶體資料寫入致能訊號。
odd4_wr	output	1	ODD4_MEM 記憶體資料寫入致能訊號。
even4_wr	output	1	EVEN4_MEM 記憶體資料寫入致能訊號。

4.3. STI系統功能

當 reset 結束後。每當 load = 1 且經時脈訊號正緣觸發時，表示 STI 輸入訊號為有效，STI 將依 據控制訊號(pi_length、pi_fill、pi_msb、pi_low)之設定將 pi_data 輸入訊號進行相對應之並列轉序列資料處理，處理完成後將 so_valid 拉成 1 表示有效資料輸出，並將處理完成之資料由 so_data 依 序送出。當 load = 0 時，表示輸入資料無效，STI 將不進行任何動作。

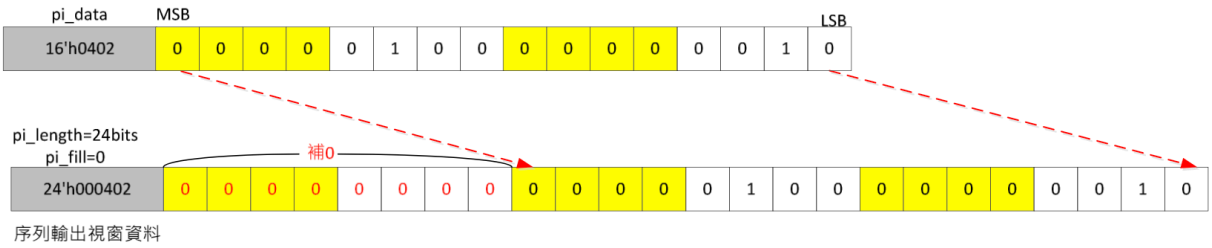
4.3.1.最高位元優先功能描述(pi_msb)

當 pi_msb 輸入為 1 時，表示 so_data 序列輸出由序列輸出緩衝資料的 MSB 開始，如圖三.(範例使 用 16 bits 說明)所示，若 pi_data 輸入為 16'h4020，當 pi_msb 輸入為 1 時，其 16 bits 序列輸出緩 衝資料由 so_data 依序輸出 0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0。



圖五、位元填滿模式資料格式（pi_fill=1）

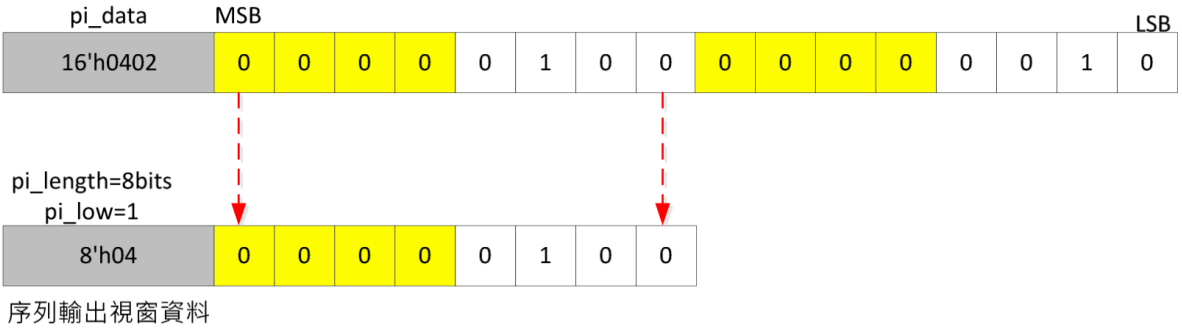
當 pi_fill 輸入為 0 時，表示 pi_data 並列資料與序列輸出緩衝資料為由 LSB 對齊，其餘位元都補 0。 如圖六.範例所示，pi_data 輸入 16’h0402 的資料時，pi_length=2’b10(24bits 資料輸出)、pi_fill=0 時，則序列輸出緩衝資料的資料 so_data 依序為 24’h000402。



圖六、位元填滿模式資料格式（pi_fill=0）

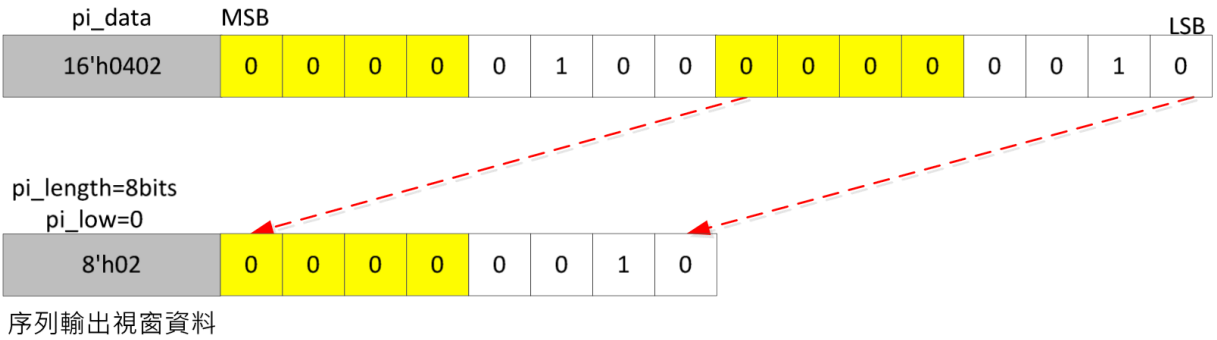
4.3.2.低位元組資料致能功能描述(pi_low)

pi_low 僅在 pi_length 設定為 8bits 序列資料輸出時有效。當 pi_low 輸入為 1 時，表示序列輸出緩 衝資料為 pi_data 並列資料的高位元組共 8bits。如圖七.範例所示，pi_data=16’h0402，pi_length=2’b00 (8bits)，pi_low = 1，則輸出 pi_data 的高位元組 8bits(8’h04)



圖七、低位元組資料模式資料格式（pi_low =1）

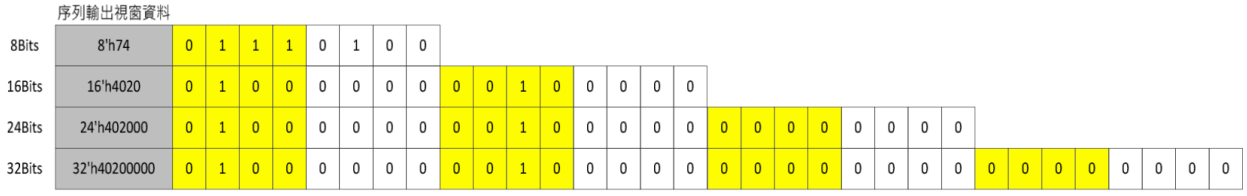
當 pi_low 輸入為 0 時，表示序列輸出緩衝資料為 pi_data 並列資料的低位元組共 8bits。如圖八.範 例所示，pi_data=16’h0402，pi_length=2’b00(8bits)，pi_low = 0，則輸出 pi_data 的低位元組 8bits (8’h02)。



圖八、低位元組資料模式資料格式（pi_low =0）

4.3.3.序列輸出訊號長度功能描述(pi_length)

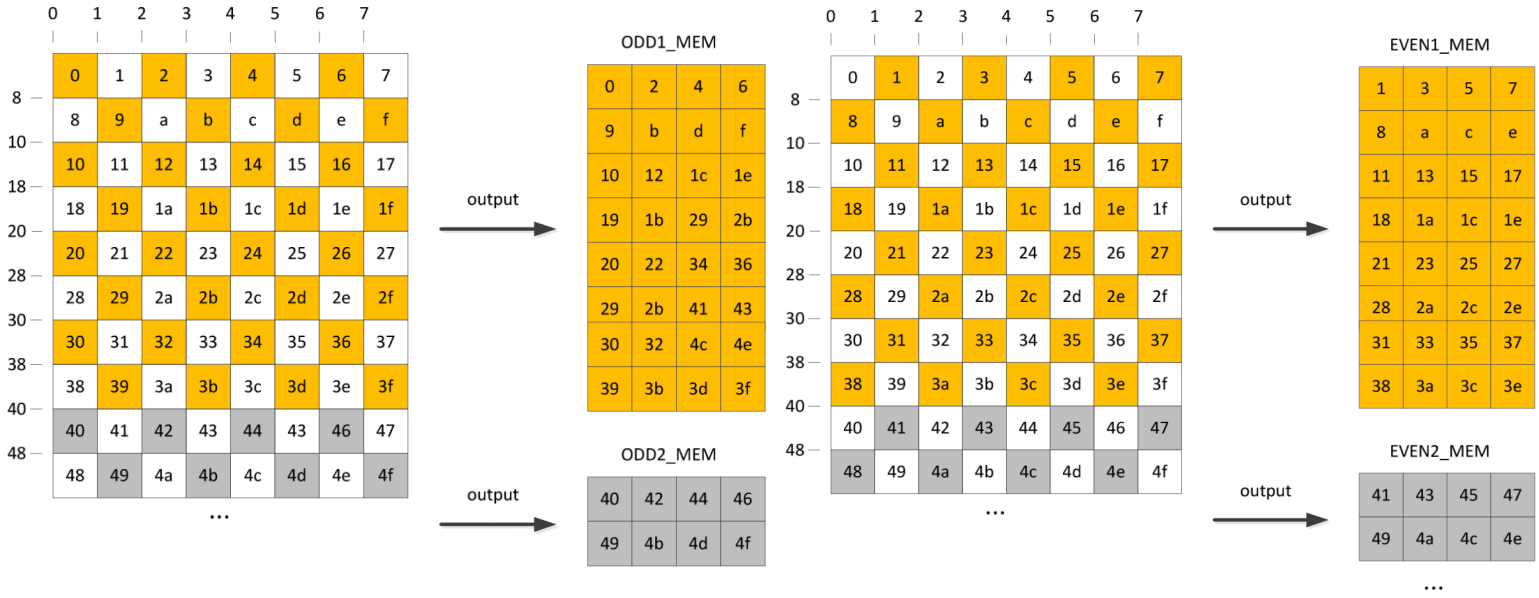
序列輸出資料格式共有 8 位元、16 位元、24 位元及 32 位元資料格式，如圖九所示，依 4.3.1~4.3.3 指令需求，利用 so_data 將序列資料輸出。



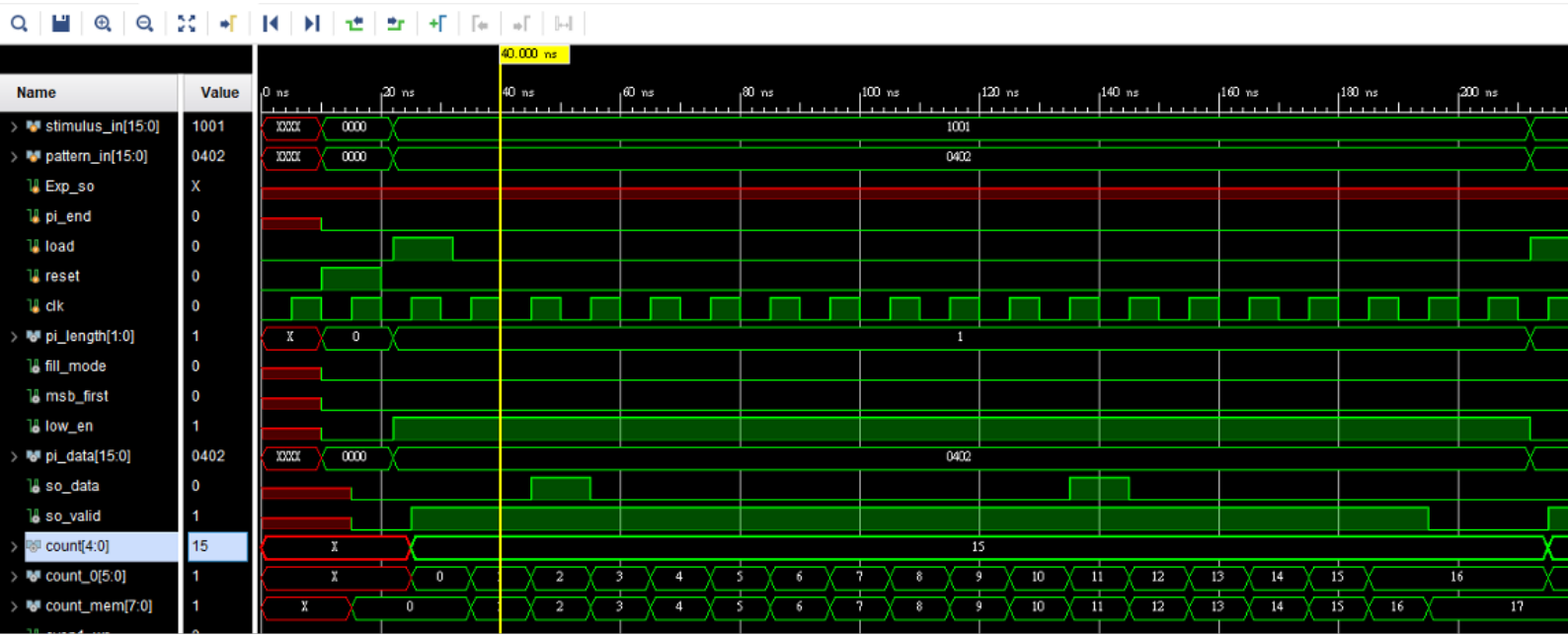
圖九、序列傳輸資料長度

4.4. DAC系統功能

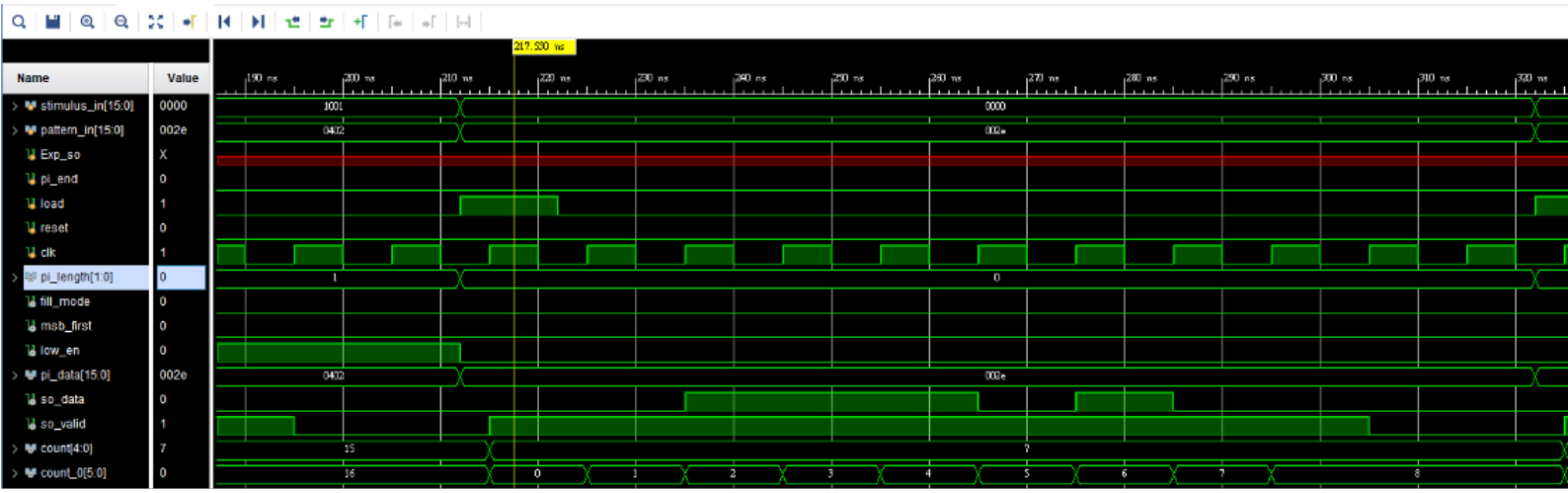
將STI序列重新排列之訊號，每8bit依序輸出至odd_mem或even_mem，第一筆輸出至odd_mem第二筆輸出至even_mem依序將所有資料輸出至memory。



5.模擬結果(vivado 2019)



上圖顯示，reset後load抬起，輸入pattern為16bit “0402”二進位為”0000 0100 0000 0010”，pi_length=1所以資料長度不用做改變，msb_first=0 資料做倒序輸出，so_data輸出為”0100 0000 0010 0000”，so_data輸出完畢so_valid隨即歸0，load再次抬起，下一個pattern輸入。

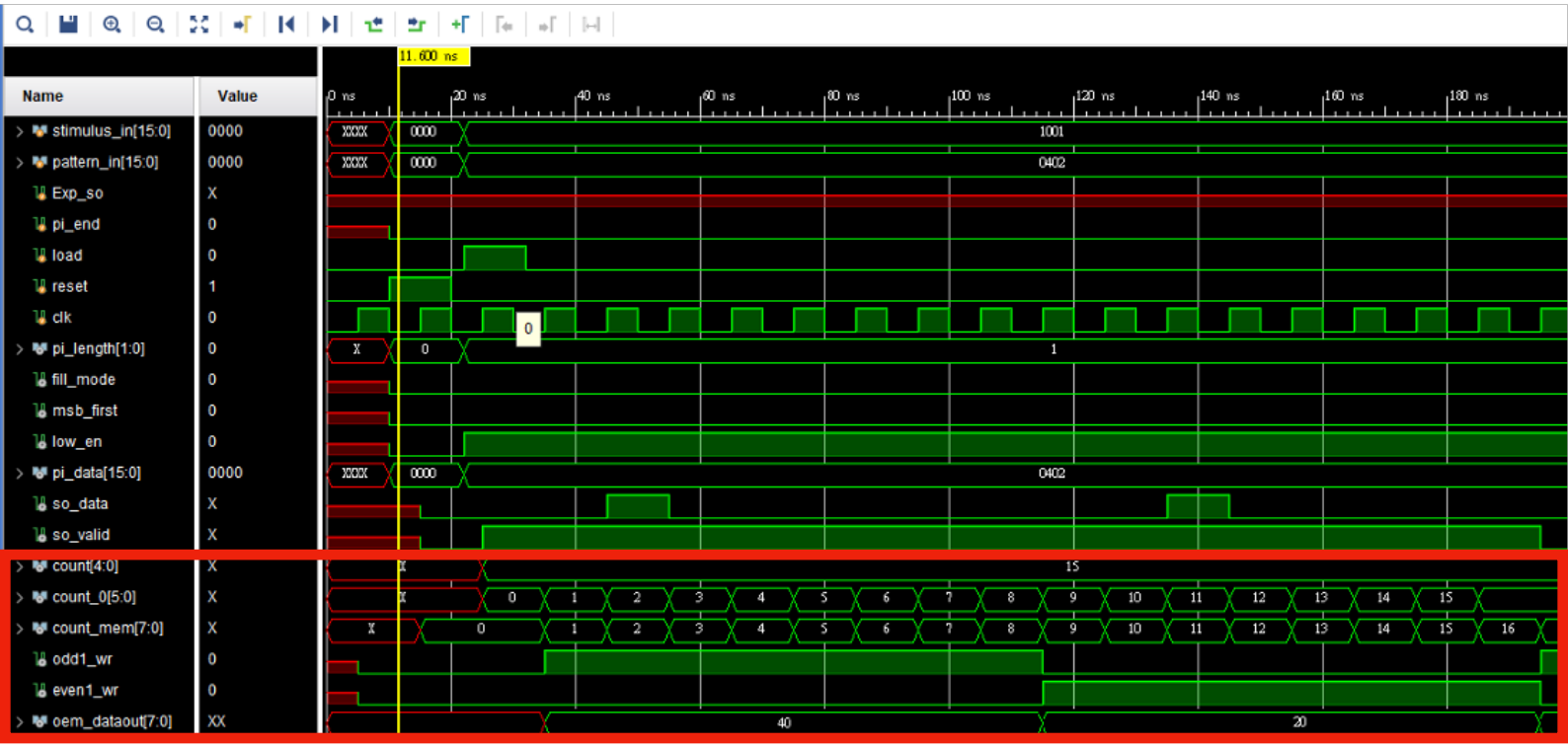


上圖顯示，load抬起，讀入pattern “002e”二進位為 “0000 0000 0010 1110”，pi_length=0資料長度為8bit，msb_mode=0 資料做倒序輸出，low_en=0 資料縮減為後半端之8bit，so_data輸出為”

01110100”，so_data輸出完畢so_valid隨即歸0，等待load再次抬起。

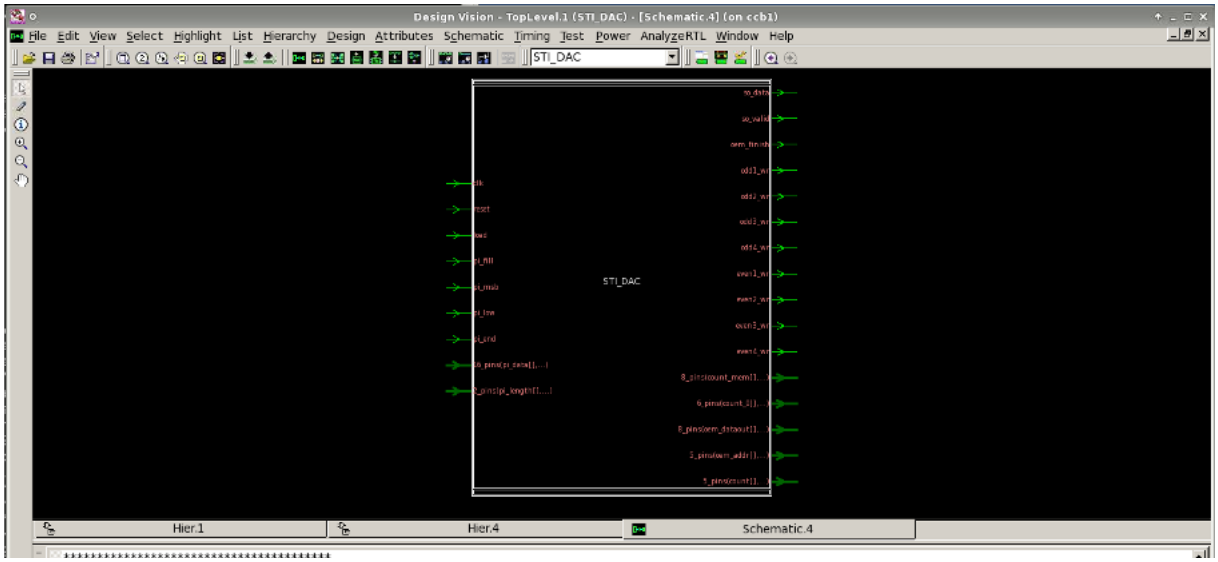
上圖顯示，load抬起，讀入pattern “08cd”二進位為 ”0000 1000 1100 1101”，pi_length=2 資料擴展為24bit，fill_mode=0 所以高位之23~16bit補0，msb_first=1 資料為依序輸出，so_data輸出為” 0000 0000 0000 1000 1100 1101”，so_data輸出完畢so_valid隨即歸0，等待load再次抬起。

load抬起，讀入pattern “0020”二進位為 ”0000 0000 0010 0000”，pi_length=3 資料擴展為32bit，fill_mode=1 所以低位之15~0bit補0，msb_first=0 資料為倒序輸出，so_data輸出為”0000 0000 0000 0000 0000 0100 0000 0000”，so_data輸出完畢so_valid隨即歸0，等待load再次抬起。

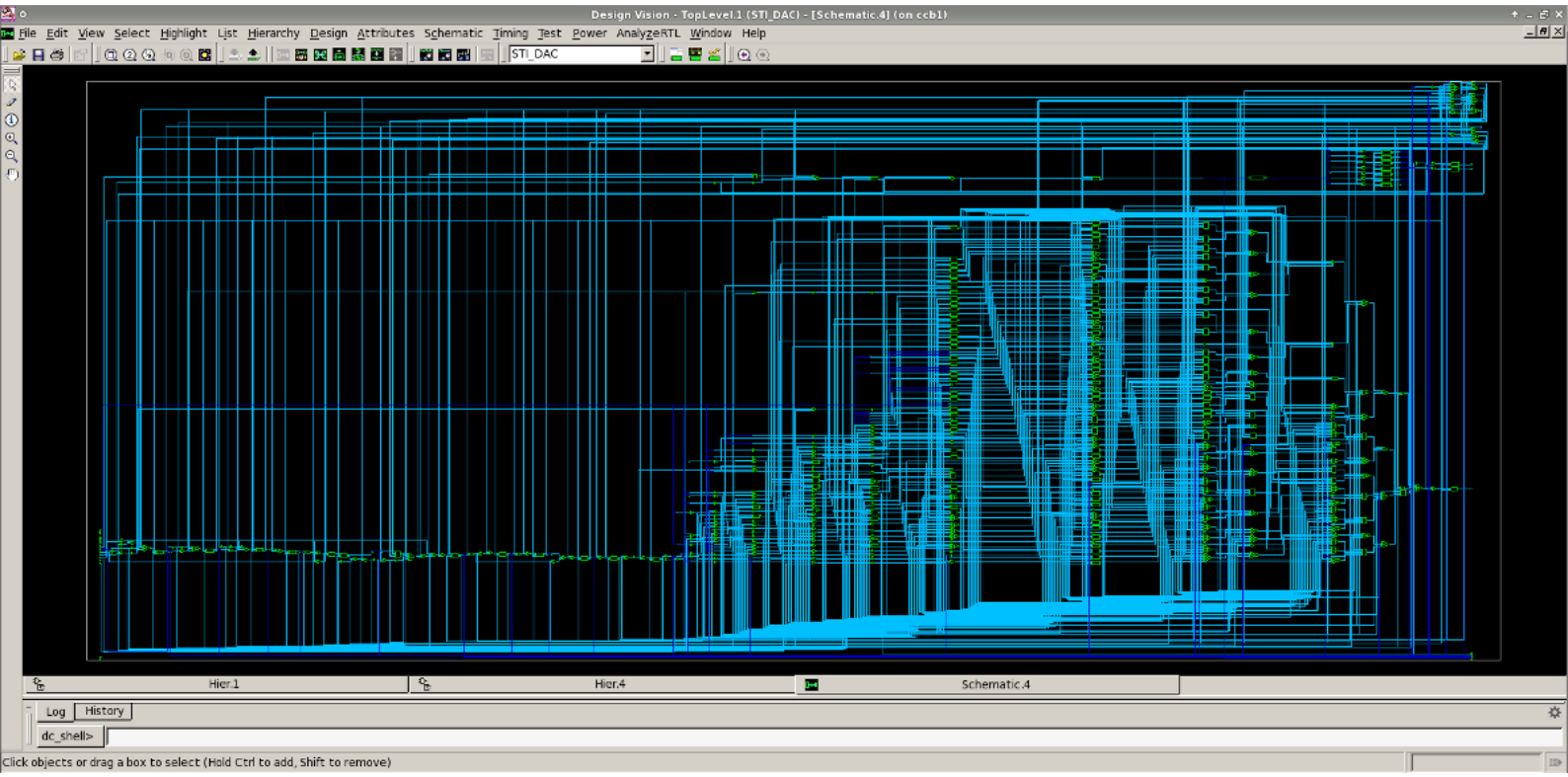


紅框處為DAC將結果輸出至memory之模擬圖，odd_wr與even_wr依序代表輸出至odd_mem或even_mem，每次輸出8bit，以第一個pattern舉例，經STI電路重新排列的data為” 0100 0000 0010 0000”所以輸出至oem_dataout之資料依序為 “0100 0000” “0010 0000” 即為 “40” “20”

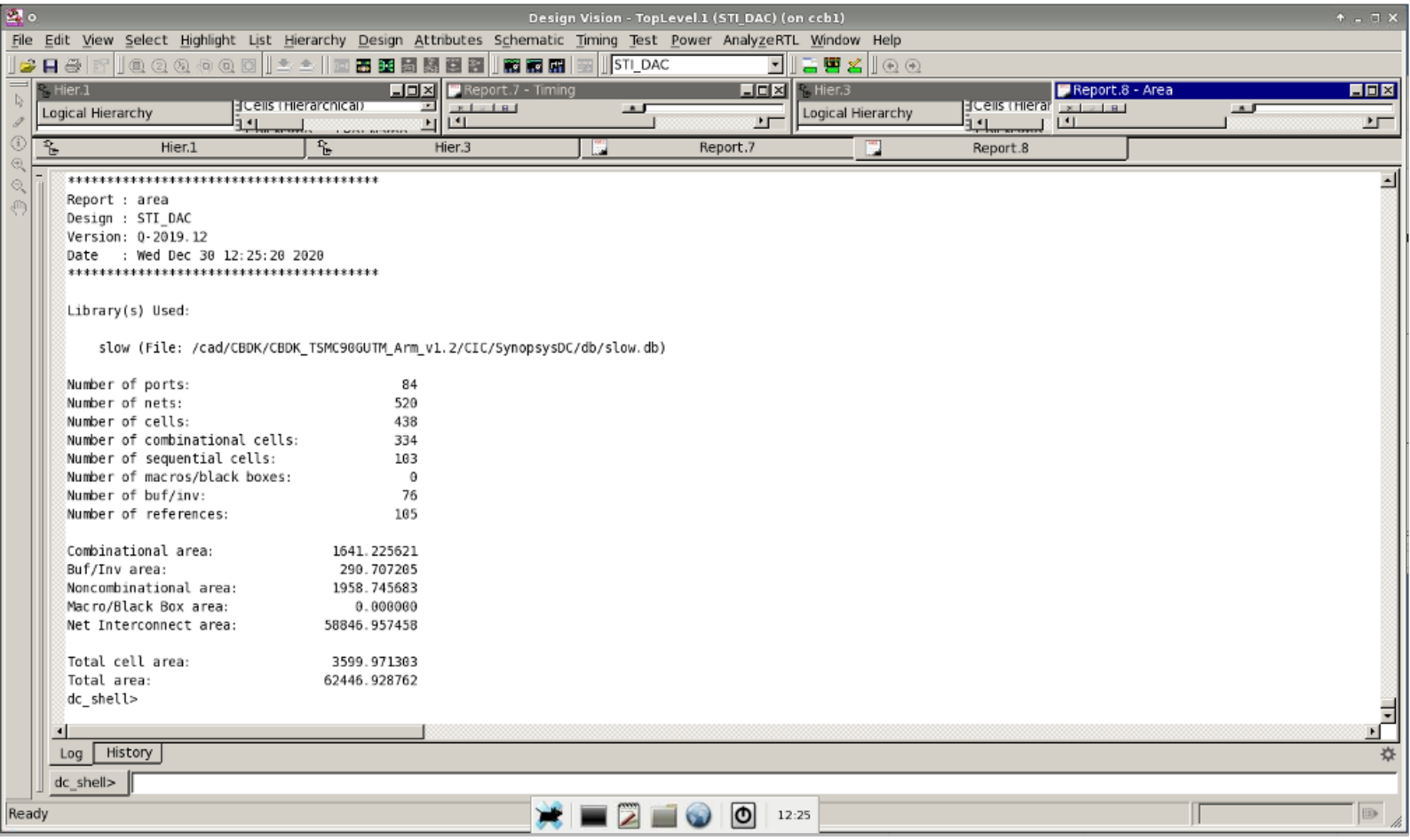
6.合成結果 （design compiler）



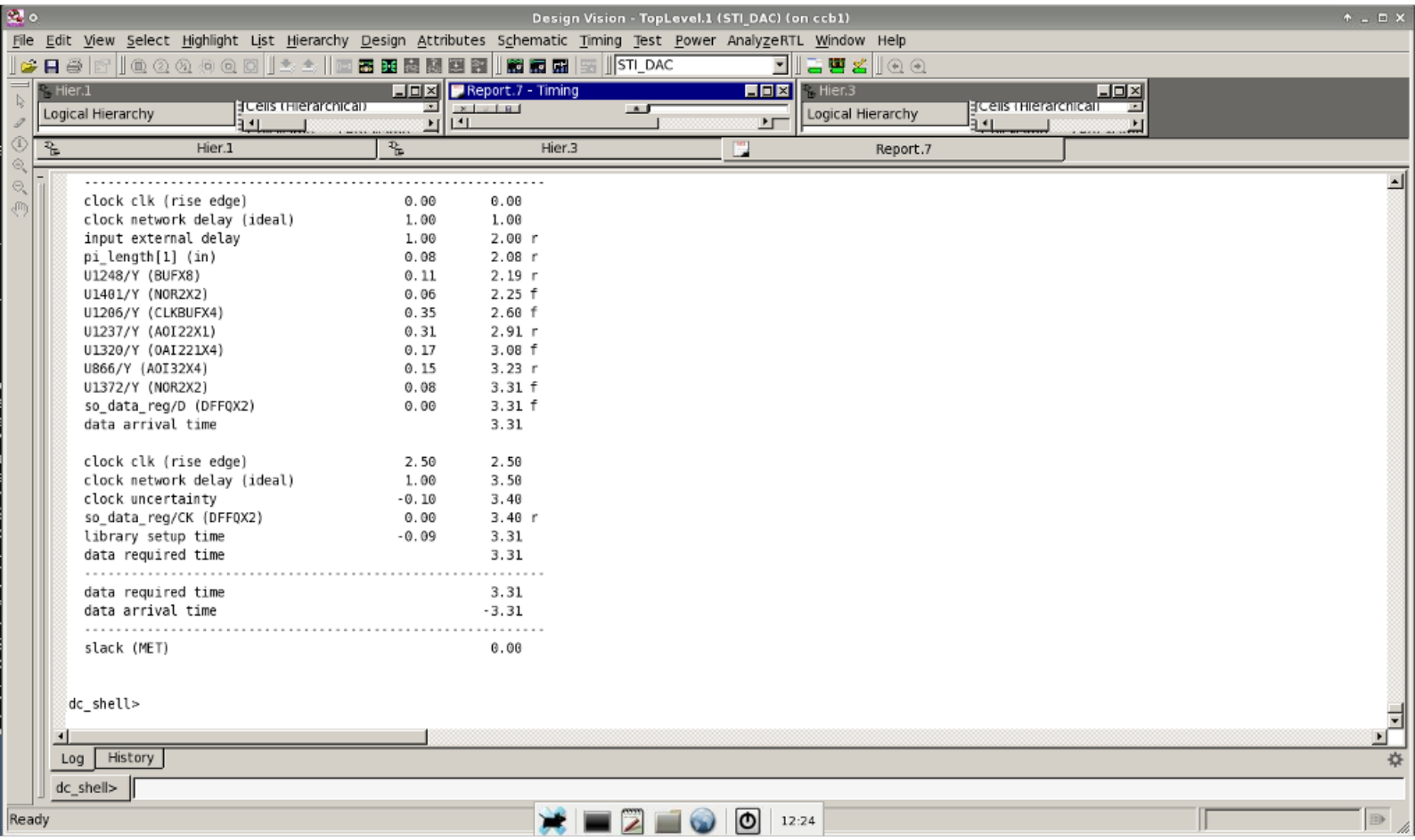
上圖為此次電路最上層I/O接腳圖



上圖為此次電路下層邏輯閘配置圖



上圖為此次電路之面積報告



上圖為此次電路之時序報告，將時脈週期設定為2.5ns，頻率為400MHz，此時電路之slack為0.00

7.討論與結論

在此次專題的verilog code中，我並沒有使用FSM來規劃當前狀態，所以在除錯時耗費了許多不必要的時間，在往後的HDL撰寫前，應該要先分析整個系統架構，規劃每個狀態，讓整個設計的流程能更有系統性，也會更有效率。

- 模擬軟體：vivado2019.1
- 合成軟體：design compiler
- 製　　程：90nm
- 時　　脈：400MHz