VLSI System Design (Graduate Level)

Fall 2022

HOMEWORK II

REPORT

Must do self-checking before submission:

Compress all files described in the problem into one tar

All SystemVerilog files can be compiled under SoC Lab environment

All port declarations comply with I/O port specifications

Organize files according to File Hierarchy Requirement

No any waveform files in deliverables

Student name:陸啟倫

Student ID:N26114950

**outline**

**1.Summary 3**

2.Major problem encountered and resolution3

3.Block diagram3

**4. State diagram introduction 6**

(a)CPU wrapper**6**

(b)AXI**7**

(c)SRAM wrapper**9**

5. ABVIP10

6. Waveform12

7.Simulation result 15

8.Superlint result 20

9.Performance 21

10.Lesson learned 23

1. **Summary**

這次作業根據AXI4 protocol實作AXI作為CPU與SRAM之間的bus，其中CPU wrapper中包含兩個Master，分別對應CPU當中IF stage(讀取指令)，及Mem stage(存取資料)的部分。SRAM wrapper中包含兩個slave，分別對應IM及DM

。

將三個module分別通過JasperGold ABVIP的驗證後，再將bus接上兩個wrapper並通過rtl層級的prog0~prog4的模擬，且Superlint的結果約為99.56%。合成後clock period為10ns(slack=0)，且通過gate-level的prog0~prog4的模擬。

作業分工為陸啟倫(100%)。

1. Major problem encountered and resolution
2. 將bus與兩個wrapper相接後，發生數值未正確維持直到被存取的情況。

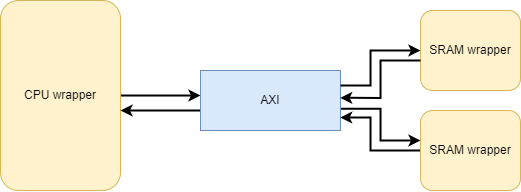
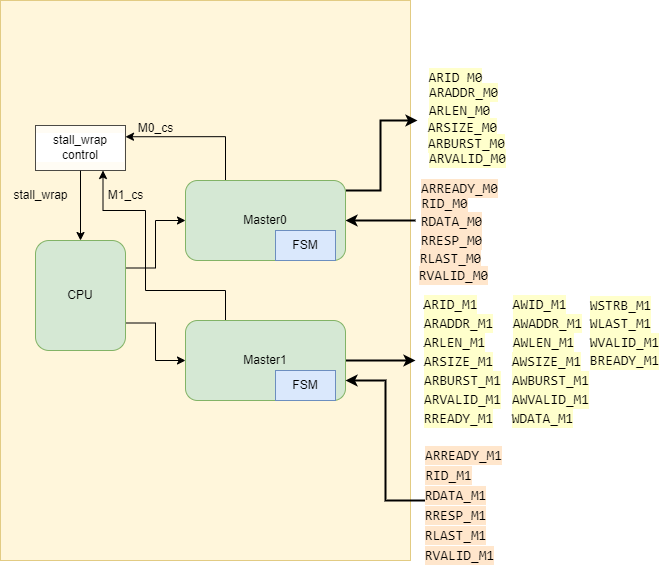
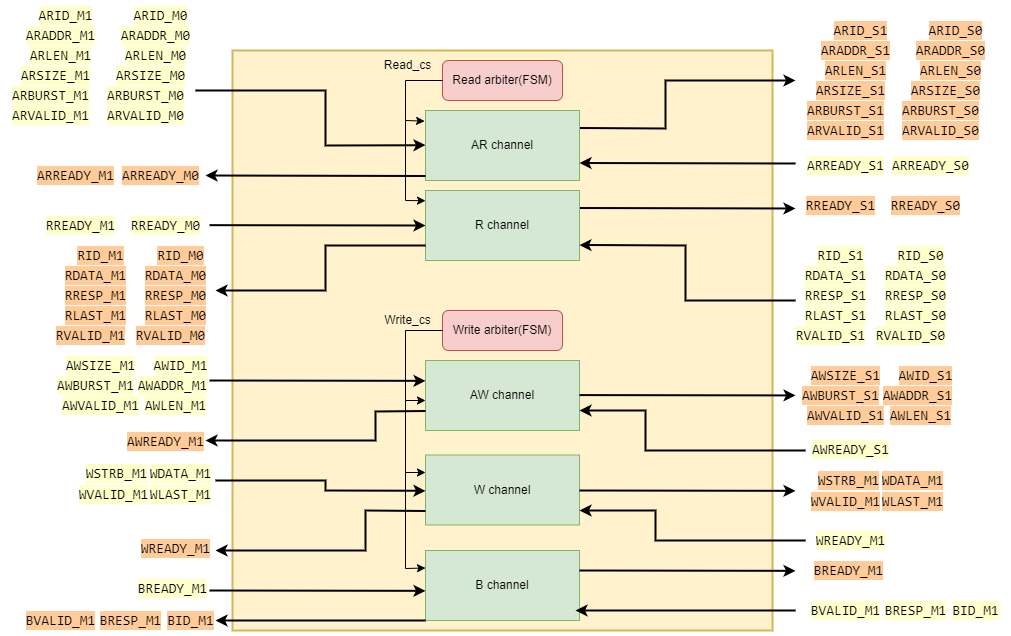
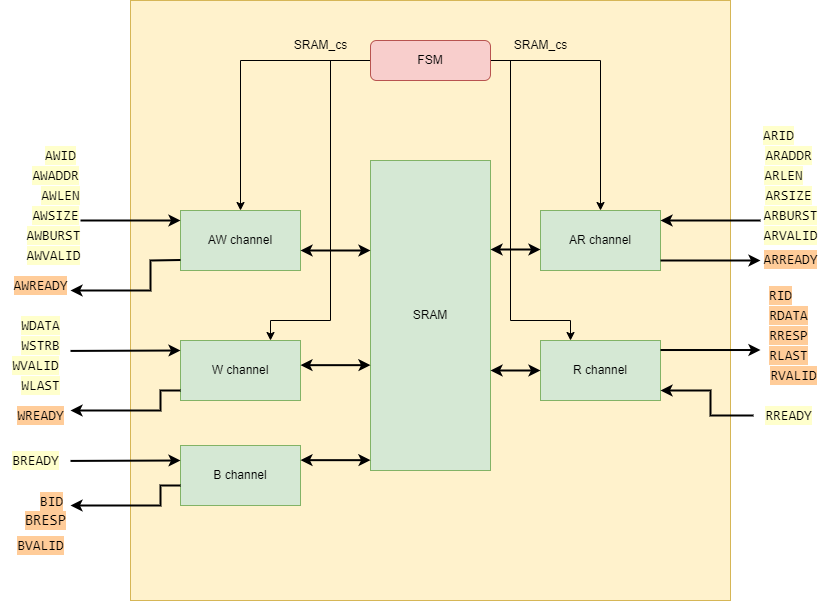
Sol:我將addr及data等相關數值以暫存器的形式存起來後即可解決。

1. 驗證ABVIP時，發生assert的前提條件未被滿足，導致部分assert產生打勾驚嘆號的結果。

Sol:可以觀察到Jasper Gold會在驗assert時，順便產生對應各個assert的cover。當發生這樣的結果時，這些cover都不會被滿足，此時只要去滿足這些cover後，就能確保assert能被驗到。

1. 因資料相依性在CPU中產生的stall及因讀取資料未完成而在CPU wrapper產生的stall有優先級的問題

Sol:在設計上應確保CPU wrapper給的stall有最高的優先級，而CPU內部產生的stall必須確保在CPU wrapper沒給stall訊號的情況下才能發生。

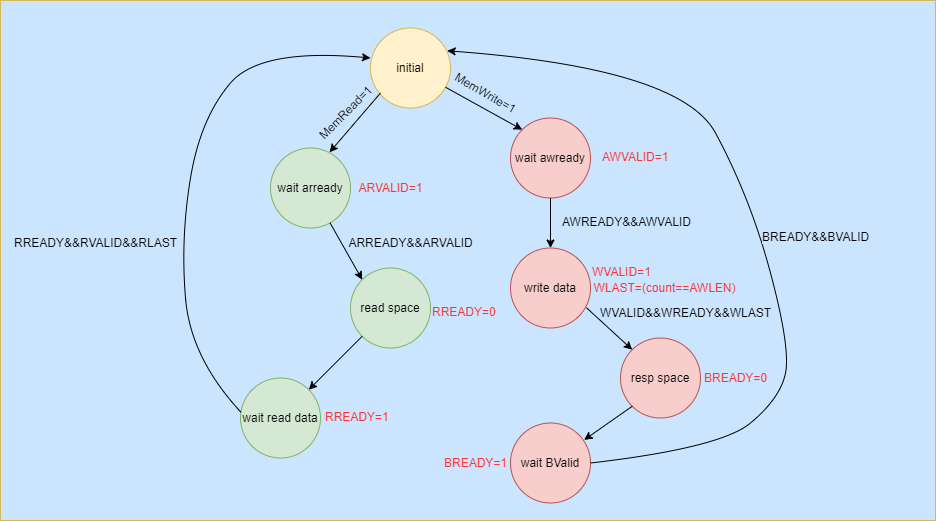
1. **Block diagram**
2. top的block diagram
3. CPU wrapper的block diagram
4. AXI的block diagram
5. SRAM wrapper的block diagram

說明:

在設計CPU wrapper上，我採取兩個狀態機去分別控制Master0及Master1的設計，並透過這兩個狀態機是否都回到初始狀態來決定CPU需不需要stall，若有任何一個Master還未執行完，則管線就要stall。

在設計AXI上，我採取兩個狀態機分別控制讀及寫的部分。在讀資料的部分由read arbiter控制AR、R 通道的開關及資料的傳輸。在寫資料的部分由write arbiter控制AW、W、B通道的開關及資料的傳輸。由於read arbiter一次只能讀取一筆資料，所以我採取的AXI設計架構為SASD。

在設計SRAM wrapper上，我採取狀態機去控制讀取和寫入的狀態，在設計上我會將addr、RDATA的數值存取起來，以確保資料在傳回CPU時能維持住。

1. **State diagram introduction**
2. CPU wrapper的狀態機(Master1)

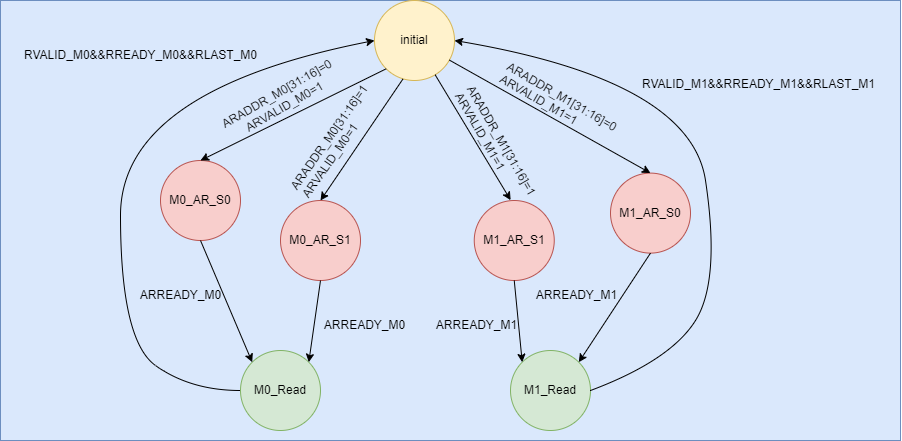
說明:

上圖為Master1中的狀態機，主要可分為讀和寫兩部分。在起始狀態時，先根據管線中的MemRead及MemWrite來決定進入寫或讀，其中我設計寫的優先級是比較高的。

若MemRead=1，則進入wait arready等待slave傳遞ARREADY訊號，在此狀態Master1須將ARADDR和ARVALID訊號穩定住，等待握手的時候再進入read space。在read space則需先將RReady=0，接著直接進入wait read data。(會新增read space的用意在於ABVIP的驗證有需要看到RREADY=0、RVALID=1時的情況)。接著進入wait read data等待RVALID訊號和RDATA，當資料全部讀取完後則會吃到slave傳遞的RLAST訊號，同時完成握手以進入初始狀態。

若MemWrite=1，則進入wait awready等待slave傳遞AWREADY訊號，在此狀態Master1需要將AWADDR和AWVALID訊號穩定住，等待握手完成後再進入write data，此時Master需給予WVALID=1訊號，以告訴slave能開始寫資料了，每當握手完成後則會將資料傳遞到slave，當WLAST訊號拉起，則代表Master正在做最後一筆的資料寫入，再待握手後進入resp space狀態。resp space狀態存在的目的也是為了ABVIP的驗證，因為證實需要看到BREADY=0、BVALID=1的情況。接著進入wait BValid傳送BREADY=1訊號，以通知salve已經準備好接收RESP，接著再待握手後回到出始狀態。

Master0的狀態機與Master1很相似，因為Master0只會讀取指令而不會寫入資料，所以狀態機只有Master1的左半部分，在此不再贅述。



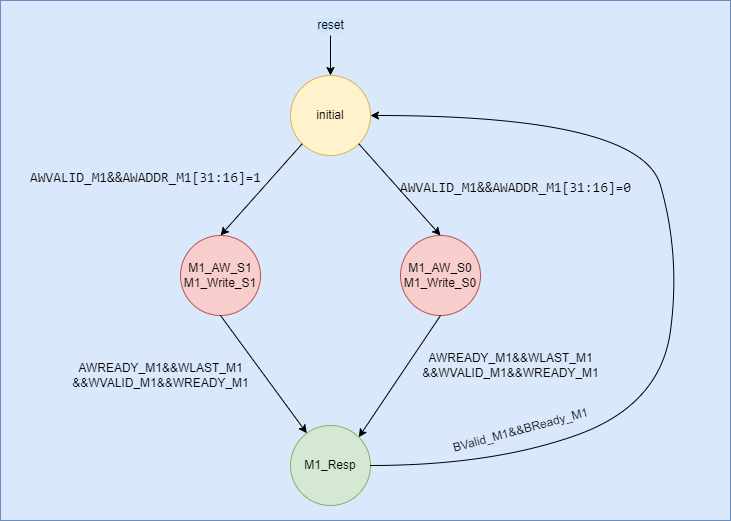
1. AXI的狀態機(Read Arbiter)

說明:

上圖為AXI中負責控制讀取的狀態機，起始狀態時需根據送入AXI的ARADDR\_M1[31:16]，ARADDR\_M0[31:16]來判斷這個地址是要送到哪一個slave當中(因此次功課IM的addr介於0x0000\_0000~0x0000\_ffff，而DM的addr介於0x0001\_0000~0x0001\_ffff)，再待對應的ARVALID後即可進入對應的AR狀態(共有M0傳給S0、M0傳給S1、M1傳給S0、M1傳給S1四種情況)。進入AR狀態時，AXI則會將AR通道打開，並將ARID、ARLEN、ARADDR、ARSIZE、ARVALD、ARREADY等資訊傳送至對應的slave或master。接著開始等待slave傳送ARREADY，代表slave已經準備接收資訊，兩方握手後，資料就能成功傳輸並進入下個狀態(M0\_read或M1\_read)。

進入Read狀態後，AXI則會打開Read通道，並將RID、RDATA、RRESP、RLAST、RVALID、RREADY等訊號傳送至對應的slave或master。

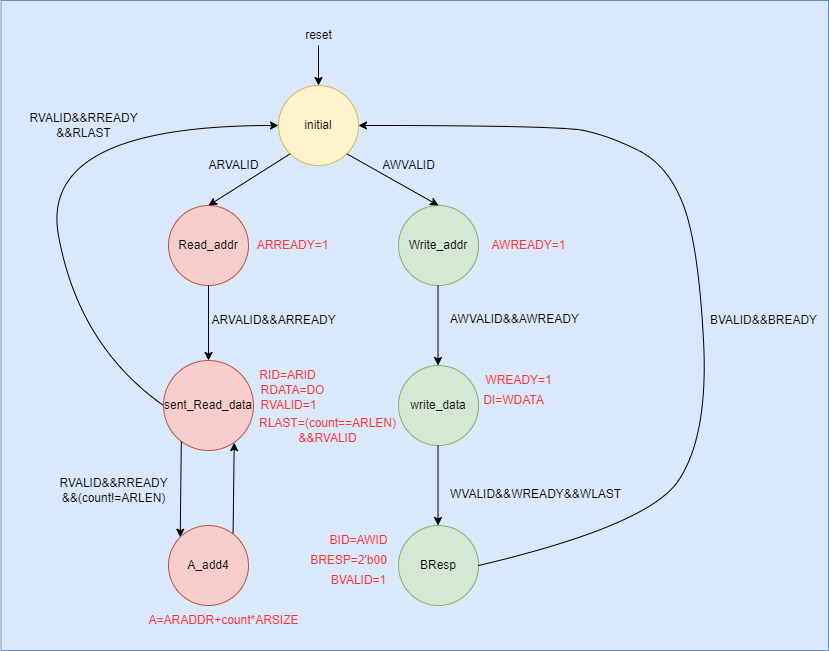
由於站在CPU讀取資料的角度，可能會有來自slave0或是slave1的情況，所以需要根據AXI在接收到來自不同Master的ARID時，做不同的高位元的編碼。待slave接收到ARID後則會將高位元的編碼存起來，待傳送RDATA時能順便加在RID的高位元部分，以便AXI分別資料要送至哪個Master。接著等待RVALID、RREADY、RLAST訊號，以確保握手成功及傳送的是最後一筆資料。才能回到初始狀態。



1. AXI的狀態機(Write Arbiter)

說明:

上圖為AXI中控制寫入的狀態機，初始狀態時須根據AWADDR\_M1[31:16]的數值來決定要將資料寫入slave0或slave1(不須考慮AWDDR\_M0的數值，因為只有Master1能夠寫入數值)，再待AWVALID訊號後即可至對應的狀態。接著進入AW和Write狀態，會將這兩個狀態寫在一起是因為要考慮ABVIP驗證的問題。在此狀態時AXI會打開AW及W通道，並將AWID、AWLEN、AWADDR、AWSIZE、AWVALD、AWREADY等AW通道相關資訊及WID、WDATA、WLAST、WVALID、WREADY等W通道相關資訊傳送至對應的master或slave。接著待AW通道及W通道都握到手後，則可進入M1\_Resp。在M1\_Resp中，AXI會打開B通道，並傳送BID、BRESP、BVALID、BREADY等B通道相關資訊至對應的slave或master，待B通道握手後則可再回到初始狀態。



1. SRAM wrapper的狀態機

說明:

上圖為SRAM wrapper中的控制狀態機，此狀態機將讀和寫分為兩部分。在初始狀態時，先透過AWVALID和ARVALID訊號來決定要進入Read\_addr或是Write\_addr。若進入Read\_addr，則會將ARREADY訊號傳送出去，以通知AXI和CPU，SRAM已準備好接收AR。再待AR訊號握手後，則可進入sent\_Read\_data狀態，並將RID、RDATA、RVALID等相關訊號傳送出去。此時會透過count來計數傳送出去的read data個數，當滿足ARLEN的要求後，則會輸出RLAST訊號，以通知CPU及AXI這是最後一筆資料。再待R訊號握手後，則可回到初始狀態。

若進入Write\_addr，則會將AWREADY訊號出送出去，以通知CPU和AXI，SRAM準備好接收AW了，再待AW握手後，則可進入write\_data，在此狀態則須將WREADY訊號送出，以通知CPU及AXI，SRAM準備好接收要寫入的資料了，再待W相關訊號握手後，則可進入BResp狀態，在此狀態SRAM則會將BID以剛接收到的AWID傳送，以確保響應能回到正確的Master，並傳送BResp訊號及BValid訊號。再待B訊號握手後則可回到初始狀態。

1. 一張含有 文字 的圖片

   自動產生的描述**ABVIP**

一張含有 文字 的圖片

自動產生的描述(a)CPU wrapper驗證

(b)AXI 驗證

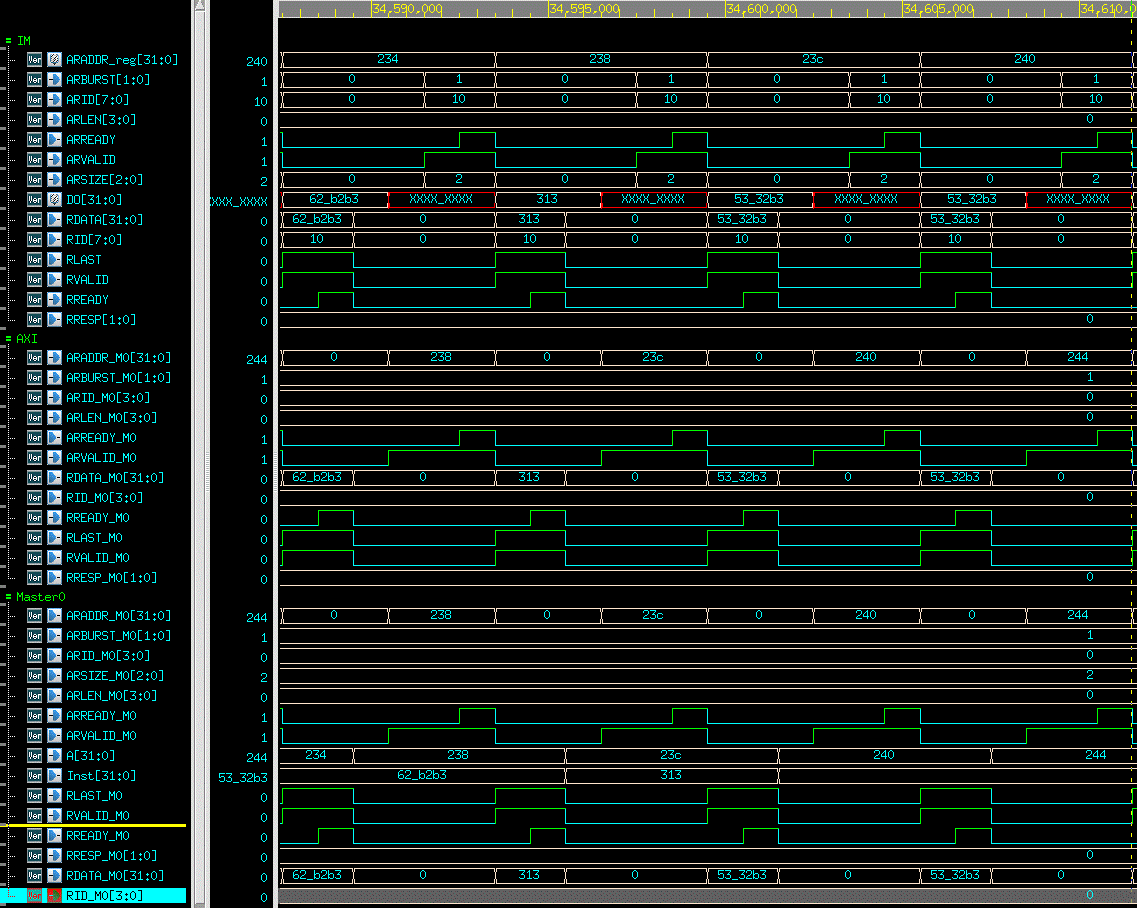
一張含有 文字 的圖片

自動產生的描述

(c)SRAM wrapper 驗證

1. **Waveform**

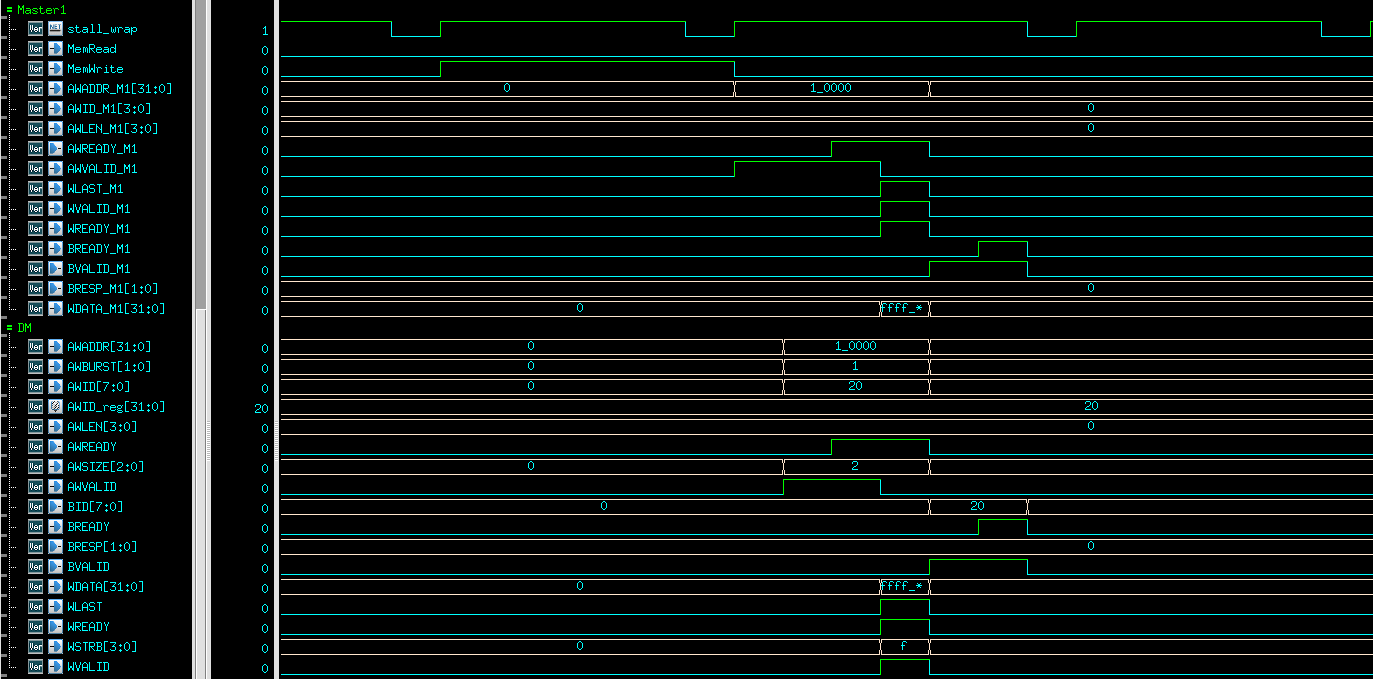




1. M0讀取

說明:

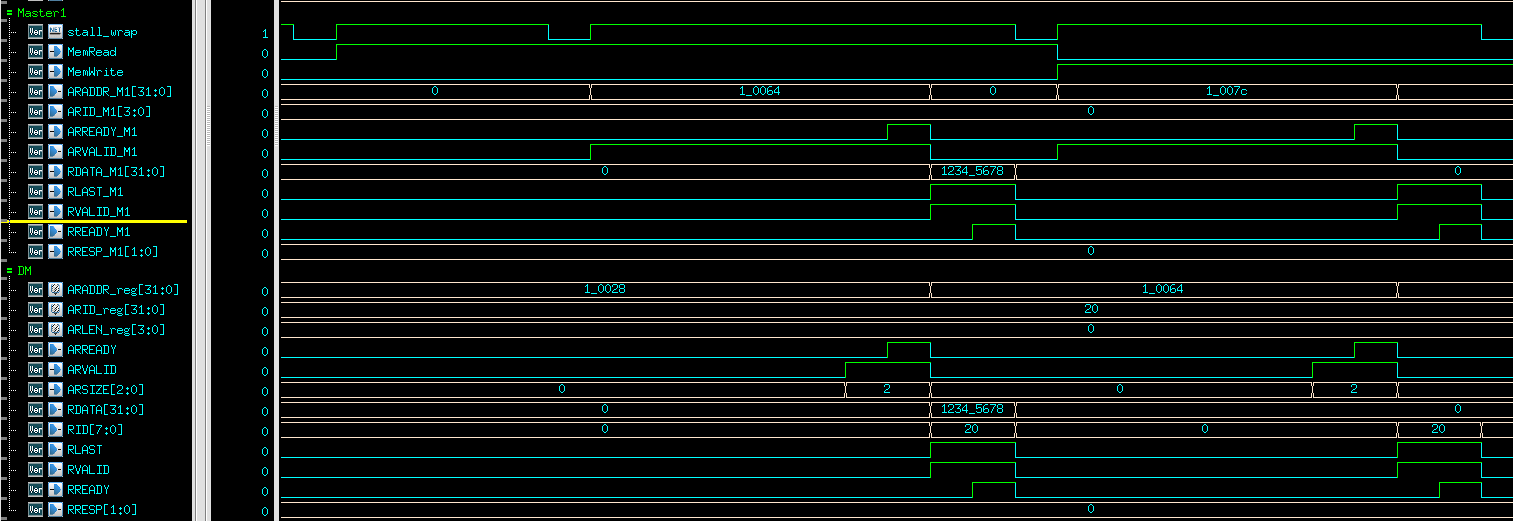
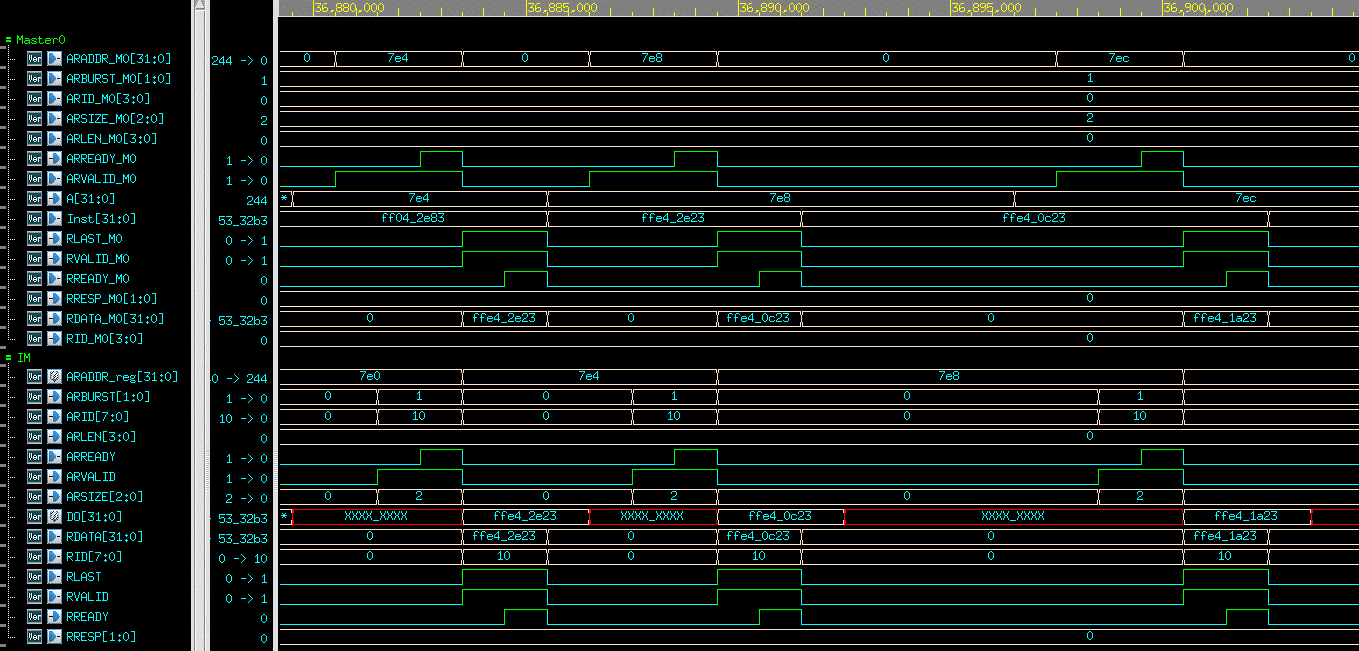
由上圖可觀察，M0要從IM讀取指令時，會先進行ARREADY和ARVALID的握手，以確保地址及其他相關資訊有正確被IM收到，待IM讀取出指令後，則會送出RLAST訊號，並進行RVALID和RREADY的握手，並將指令送出(如橘色框內所示)。Master0在接收到指令後則會將指令以暫存器存起來，再將其傳入至管線內。

1. 一張含有 文字 的圖片

   自動產生的描述M0讀取M1寫入

說明:

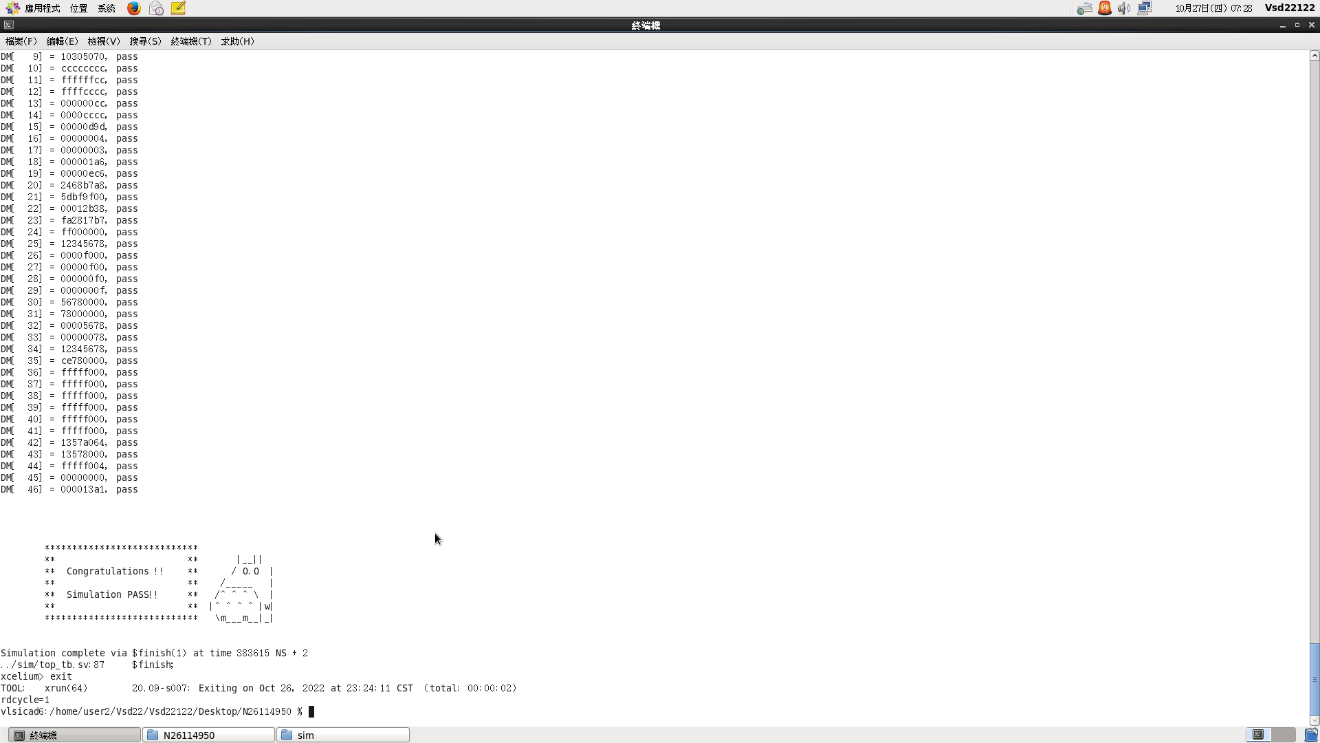
由上圖可觀察當M0要求讀且M1要求寫入時，會先做讓M1做寫入，先待AWREADY及AWVALID握手後則可將AW傳至DM，接著開始寫入資料，等待WVALID和WREADY握手且WLAST拉起後，則將資料寫入DM，接著進行BVALID和BREADY的握手，則可將響應傳給Master1。M1完成寫入後，則可開始讓M0讀取指令，可以看到ARREADY和ARVALID握手後，則可將AR傳給IM，待IM將指令讀出後，且RREADY和RVALID握手及RLAST訊號拉起時，則可成功將指令傳回Master0。

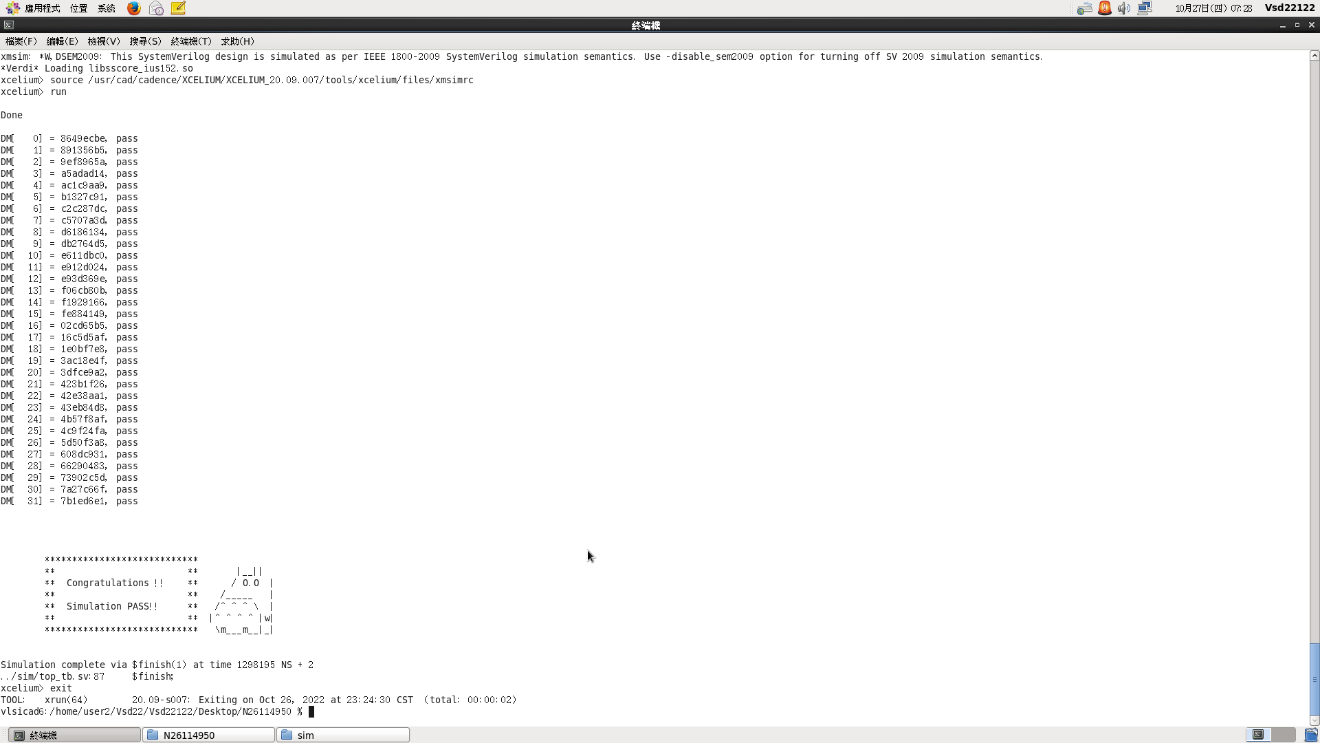


1. M0讀取M1讀取

說明:由上圖可觀察當M0和M1同時要求讀取時，會先讓M1進行讀取，待M1的ARREADY和ARVALID握手後，代表AR成功傳到DM中，接著待DM將資料讀取出來，且RVALID和RREADY握手且RLAST訊號拉起，則可成功將RDATA傳回Master1中。M1讀取完畢後換M0開始讀取，待ARREADY和ARVALID握手後，代表AR成功傳到IM中，接著待IM將指令讀取出來，且RVALID和RREADY握手且RLAST訊號拉起，則可成功將指令傳回Master0中。

1. **Simulation result**



1. prog0模擬成功

(b)prog1模擬成功

一張含有 文字 的圖片

自動產生的描述

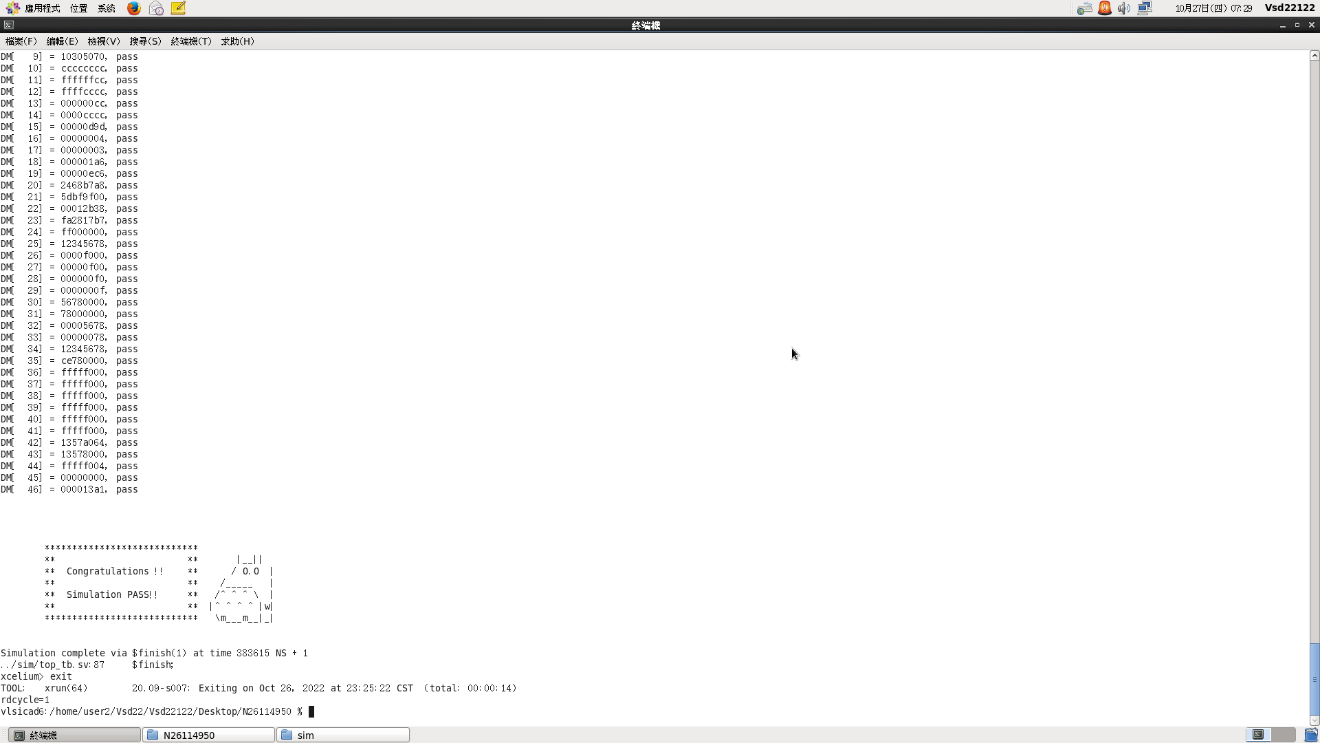
(c)prog2模擬成功一張含有 文字 的圖片

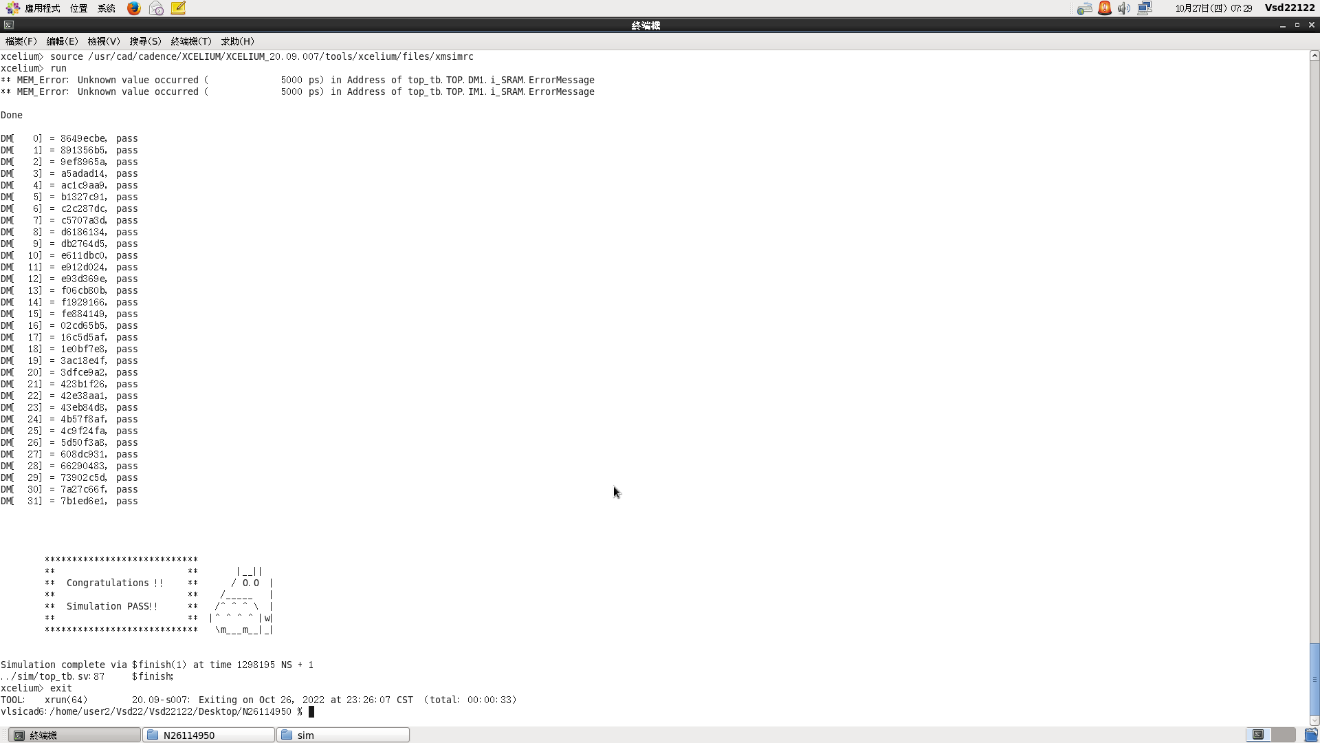
自動產生的描述

(d)prog3模擬成功

一張含有 文字 的圖片

自動產生的描述

1. prog4模擬成功
2. syn0模擬成功



1. syn1模擬成功一張含有 文字 的圖片

   自動產生的描述(g)syn2模擬成功

`一張含有 文字 的圖片

自動產生的描述(h)syn3模擬成功一張含有 文字 的圖片

自動產生的描述

1. syn4模擬成功
2. **Superlint result**

****

**一張含有 文字 的圖片

自動產生的描述**(a)總體程式行數為6563

(b)總體warning為29個

在做superlint的檢查時，有遇到case狀況未寫滿的問題，導致有幾個error發生，由於這次狀態機比較多，所以有幾個狀態機漏寫到default的情況，事後有將defaultr的狀況加上後，即可修掉這些error。

經計算後，整體superlint中的執行結果約為**99.56%**(warning不佔總體程式行數的百分率)

1. 一張含有 桌 的圖片

   自動產生的描述**Performance**

(a)area report

1. **一張含有 桌 的圖片

   自動產生的描述**一張含有 文字, 收據, 螢幕擷取畫面 的圖片

   自動產生的描述power report
2. timing report
3. **Lesson learned**
4. Jaspergold的基礎使用方法

這次實作AXI、CPU wrapper、及SRAM wrapper都有使用Jaspergold來做驗證，在使用Jaspergold debug時，可以發現很多在設計上沒有思考到的點，相較simulation base的檢驗方法，正規驗證能夠檢測出更多難測到的corner case。

1. ABVIP的使用方法與方便性

這次驗證使用的是Jasergold提供的vip做驗證，由於AXI protocol是已經訂好的標準，所以我們有現成的SVA檔案可以使用，各種應該包括或是不該發生的情況都已經寫成assert property。若是沒有VIP的幫助，就需要根據SPEC的理解來自己寫SVA檔案，這會是個浩大的工程。

1. AXI protocol 的了解

這次算是我第一次有先去閱讀SPEC的設計經驗，根據我的了解，業界實際在設計電路時，都必須先花不少時間了解SPEC。這次是我第一次接觸BUS，不像是功課一的CPU是在計算機組織就學過的東西，所以大概花了一個禮拜的時間了解BUS內部的訊號及運作模式。在了解後也可以發現我們這次的作業其實有很多訊號及功能都沒有用到，只算是實作一小部分，但這次設計讓我大概對BUS有初步的認識。

1. Mem delay對performance造成的影響

在模擬時可以發現與模擬時間及所需要的cycle數都比功課一多很多，因為功課一的情況相當單純，Mem只需要1個clk就能將數值讀取或寫入。但這次有了bus後，管線在未獲得資料或是未完成寫入資料時都必須暫停，導致模擬時間大幅增加的情況。有了這次經驗後，可以發現Mem delay確實是在考量設計表現時須認真考慮的問題。