Advanced VLSI System Design (Graduate Level)

Fall 2024

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: \_\_\_\_王彥珽\_\_\_\_\_\_

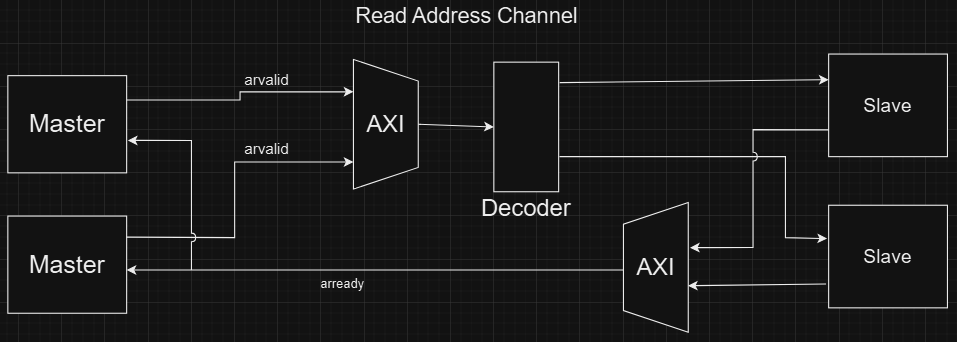
Student ID: \_\_\_\_\_P76131416\_\_\_\_\_\_

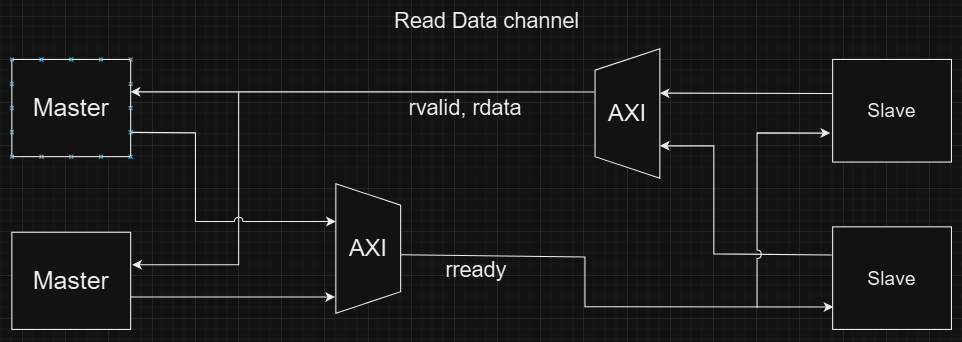
Summary:

這次我實作了AXI的5個channel分別為read address, read data, write address, write data, write response，其中read address與write address中間還會有Arbiter去決定Master的優先權、Decoder去決定地址要送往哪個Slave；SRAM的話有Slave\_wrapper則決定如何跟AXI互動；CPU的話則是將原本的IM與DM修改為Instruction Master與Data Master透過CPU\_wrapper與AXI進行資料傳輸。

Problem1:AXI bus

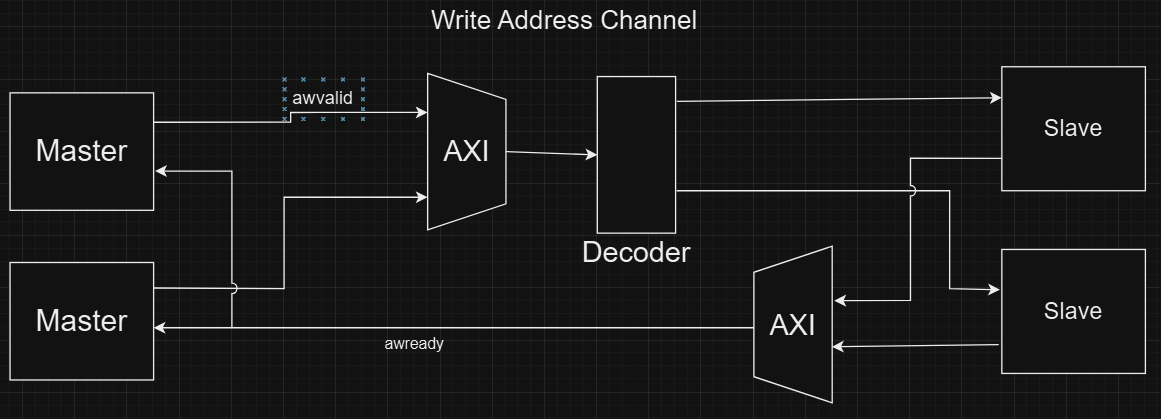
ReadAddress and ReadData Channels:

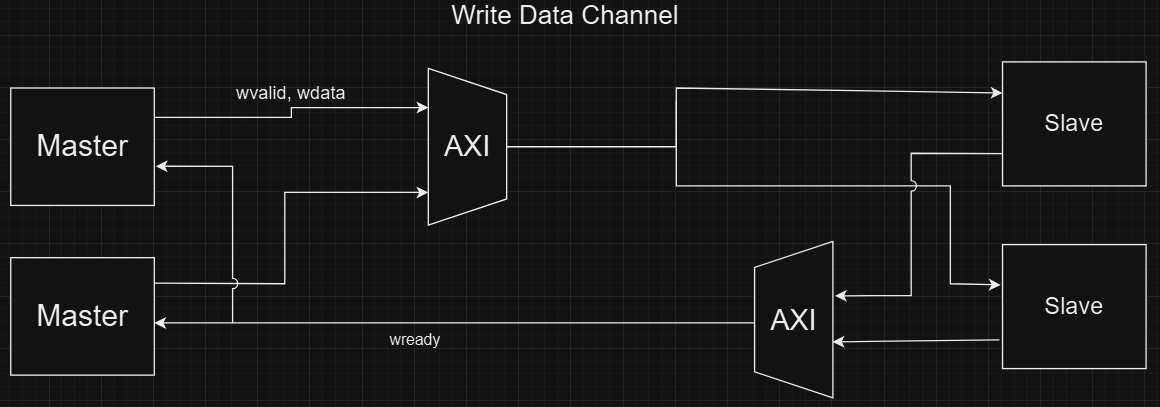


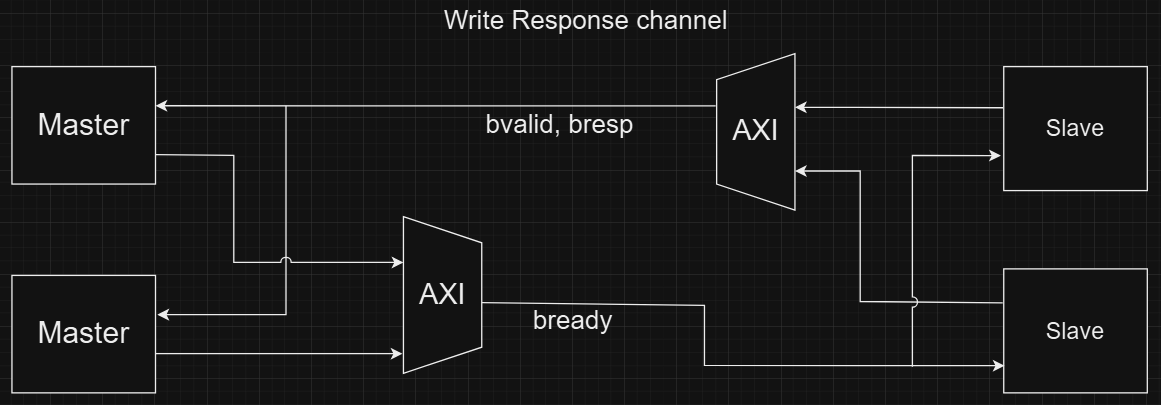


如果Master要做read的指令，他會先在Read Address channel發送ARVALID說明此次是有效的地址，接著Read Address channel會透過Arbiter和Decoder去決定要優先讓哪個Master做read和判斷這個地址是要送去哪個Slave；當Slave接收到ARVALID後，會將ARREADY發送回當初發送ARVALID的Master，此時Read Address握手完成，即地址傳送完成。此時被選中的Slave會根據傳送來的address去準備相對應的data，並且透過ARID去判斷要將data回傳給哪個Master，當資料準備好時，Slave會傳送RVALID給Read Data channel給Master端，告訴他資料已經準備好，此時Master會發送RREADY表示可以接收資料，此時Read Data握手完成，整個Read動作完成。而在Read Address傳送過程中，當Instruction Master與Data Master同時去access相同Slave時，我會先讓Data Master優先存取，等到Data Master存取完成才會讓Instruction Master存取。

WriteAddress, WriteData and WriteResponse Channels:





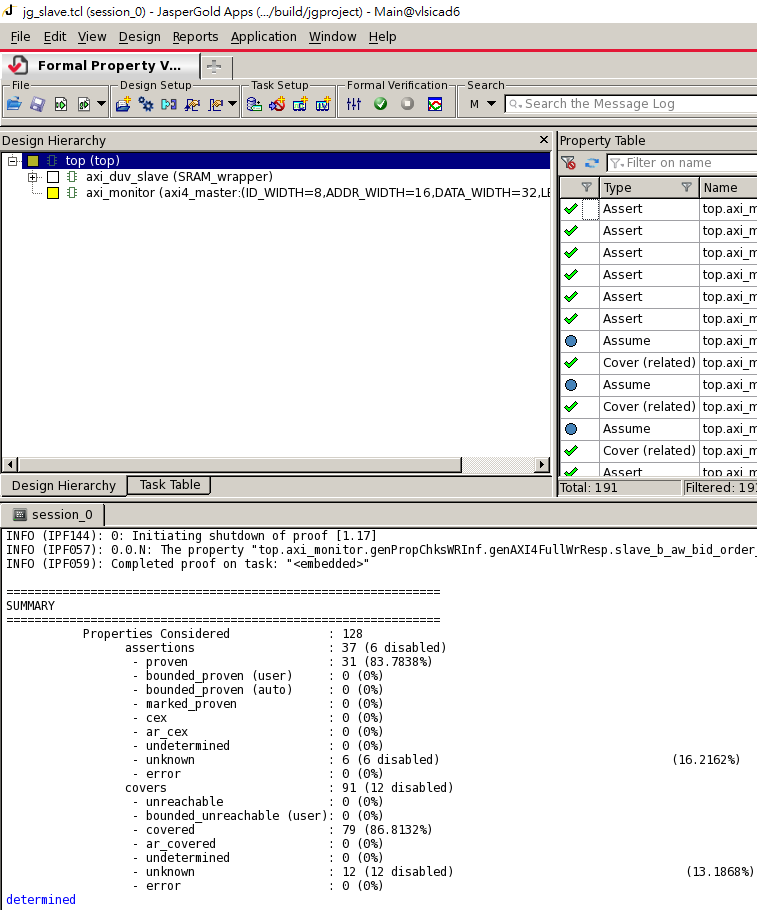


若Master想要對Slave做寫入時，會先在Write Address channel發起AWVALID，表示想要做寫入動作且此時的write address為有效的地址，此時Write Address channel會透過Arbiter與Decoder判斷Master寫入優先權與要將地址傳送給誰，這時地址對應的Slave會傳送AWREADY表示可以接收地址，此時Write Address握手完成，地址傳輸成功。Master判斷Write Address傳送完成後，會將要寫入的資料準備好，並透過Write Data channel傳送WVALID，此時被選中的Slave會收到WVALID，透過剛剛收到的AWID去決定要接收誰的資料並回傳WREADY，此時Write Data握手完成，Write data傳輸成功。

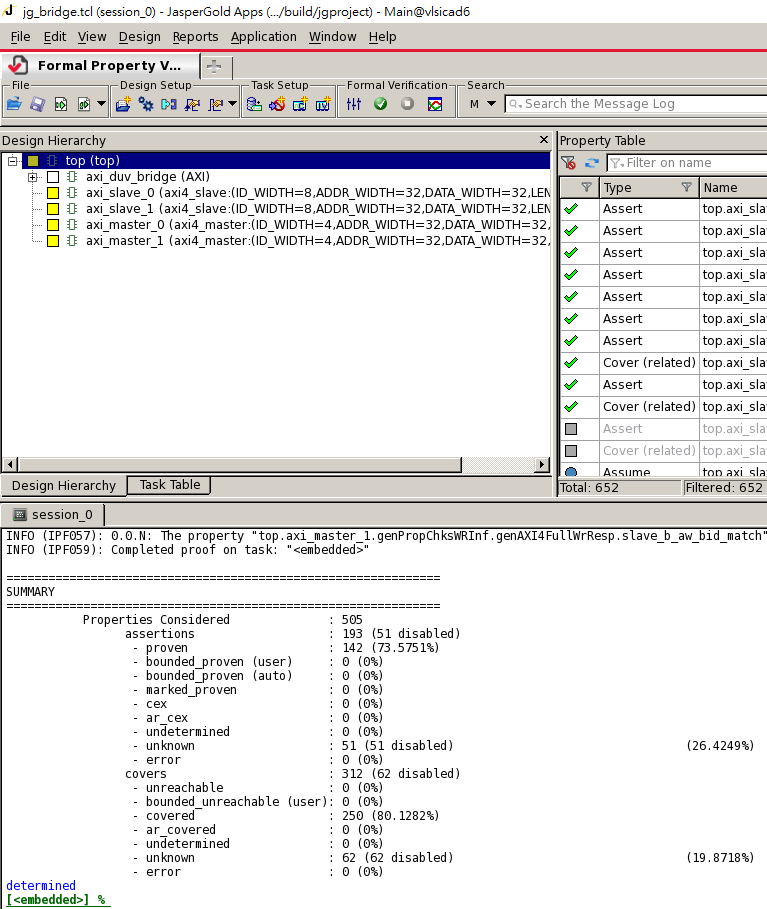
當Slave完成Write Data後，會透過Write Response channel發出BVALID去告訴Master資料已經寫入完畢，此時Master接收到BVALID後會回傳BVREADY表示已經接收到Write Response訊號，此時Write Response握手成功，整個寫入的動作完成。

JasperGold

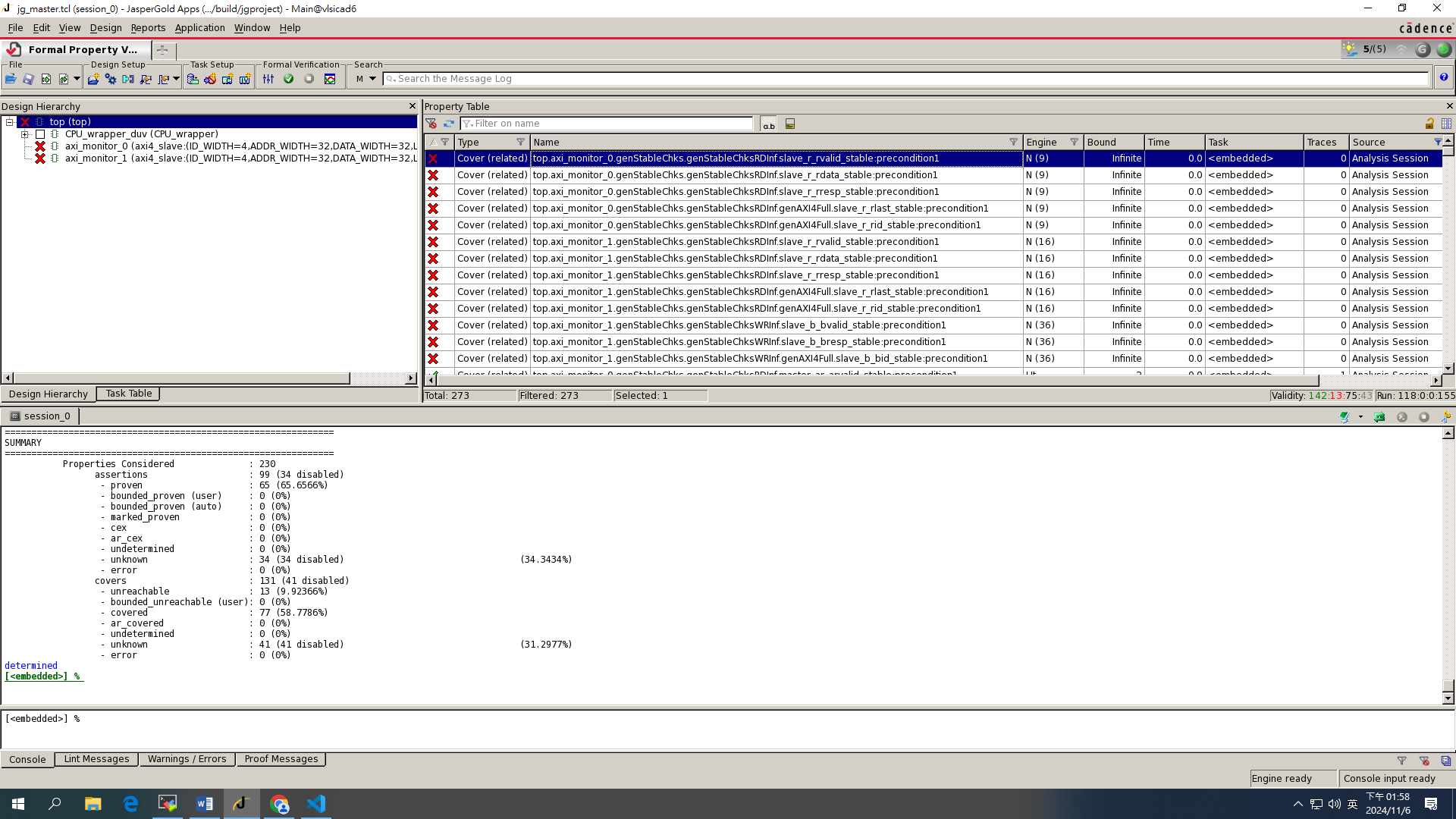
Slave : all pass



Bridge : all pass



Master : 13 covered violation



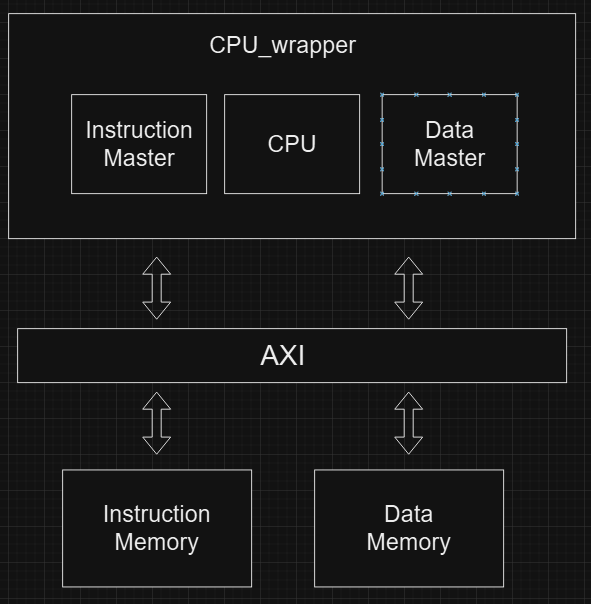
Master有些cover沒有辦法通過驗證，是因為在Write Response與Read Response的過程中，Master只有當Slave發出valid訊號才會將ready舉起，所以導致有些cover沒有辦法通過。

在write response時，可以看到當bvalid & ~bready時，bid、bvalid與bresp皆不可發生改變。但在這邊並不會產生bvalid先舉起的情況，因為當Write data握手完成當下就會將bready舉起。

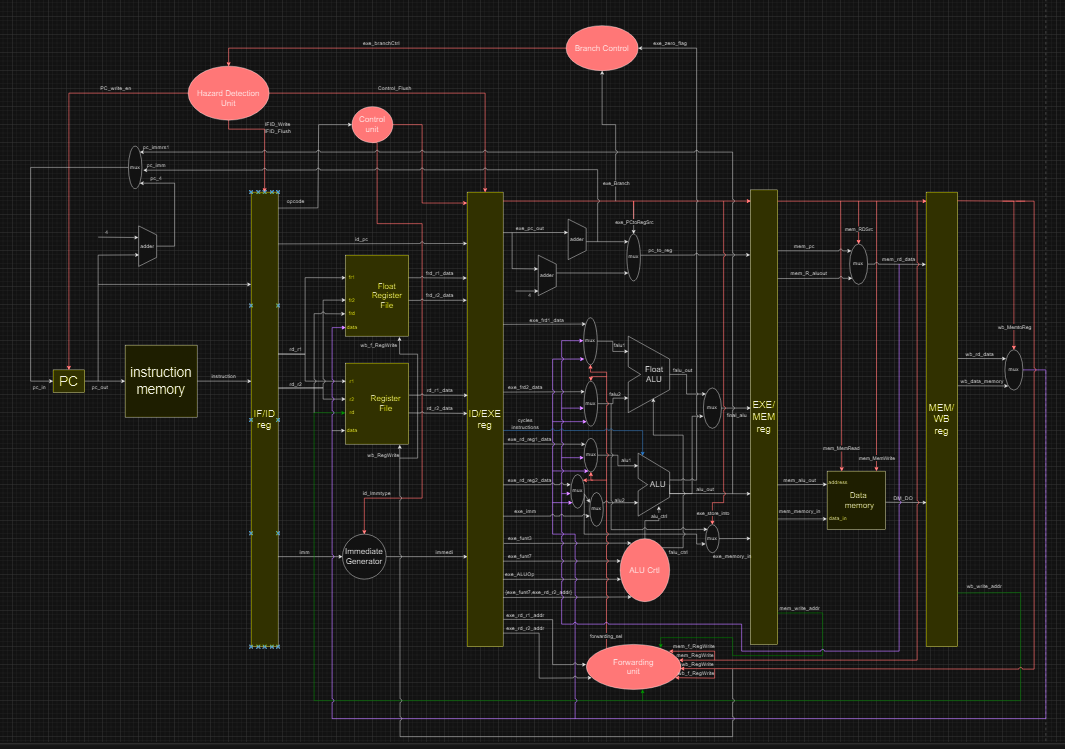
而在read response也會發生跟write response一樣的情況，當read address完成握手時，rready就會舉起，因此會產生跟r channel有關的cover violation。

Problem2 : Combine AXI with CPU

Block diagram:

top:

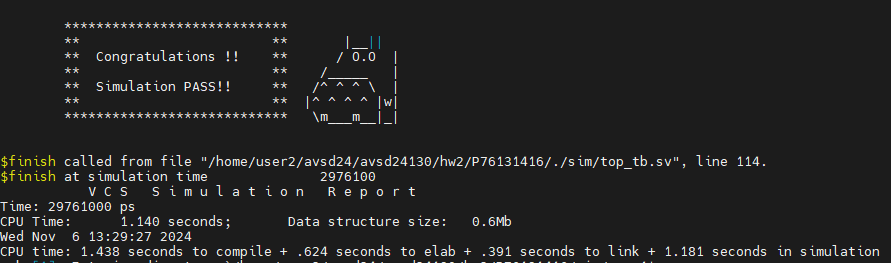
CPU:



RTL and Gate-level simulation : all pass

RTL

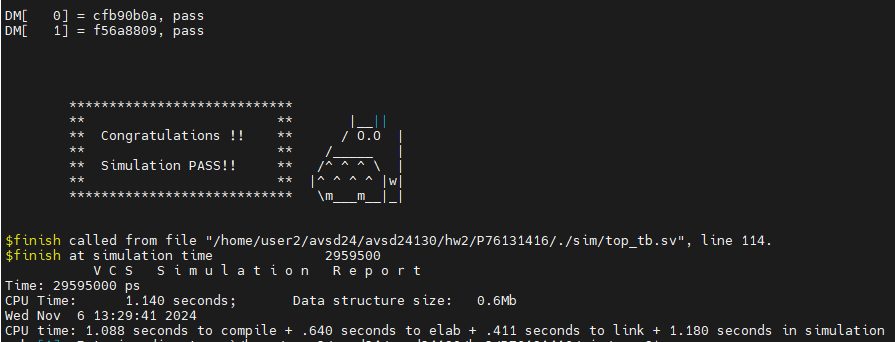
prog0:



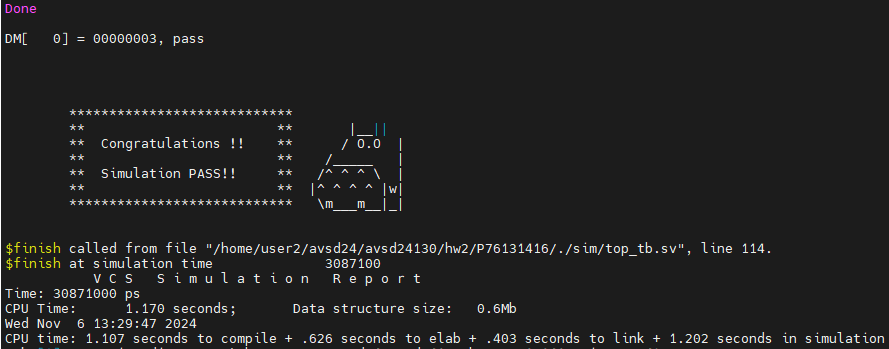
prog1:



prog2:



prog3:



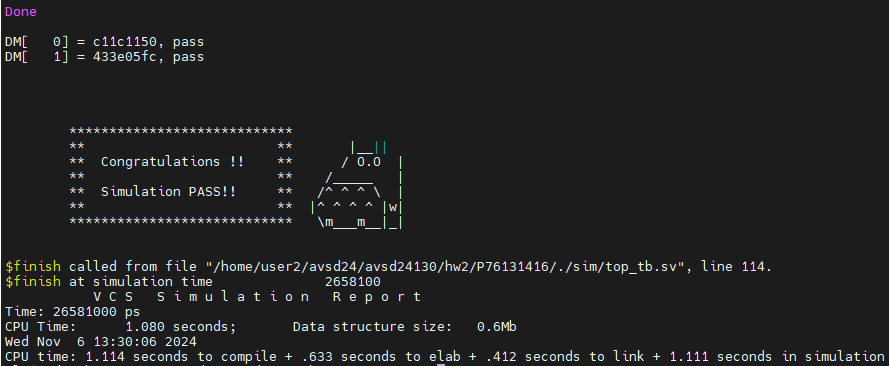
prog4:



prog5:

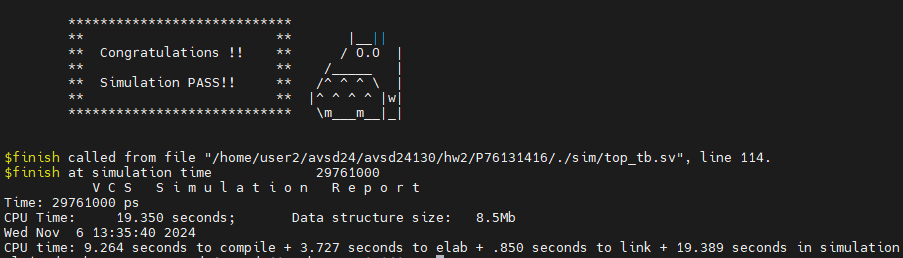


prog6:

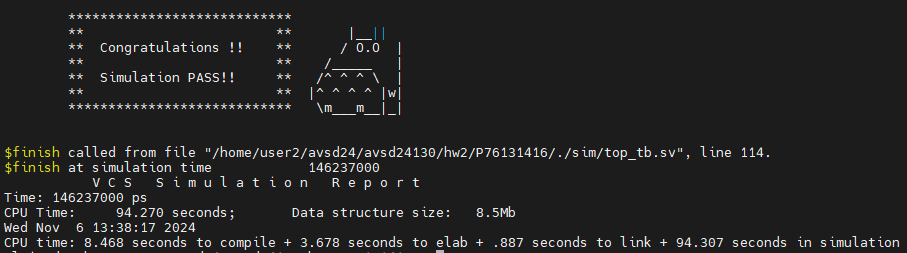


Gate-level simulation

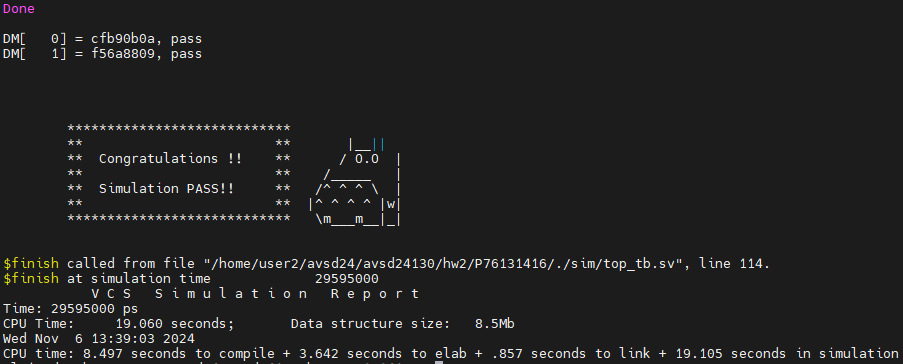
Prog0:



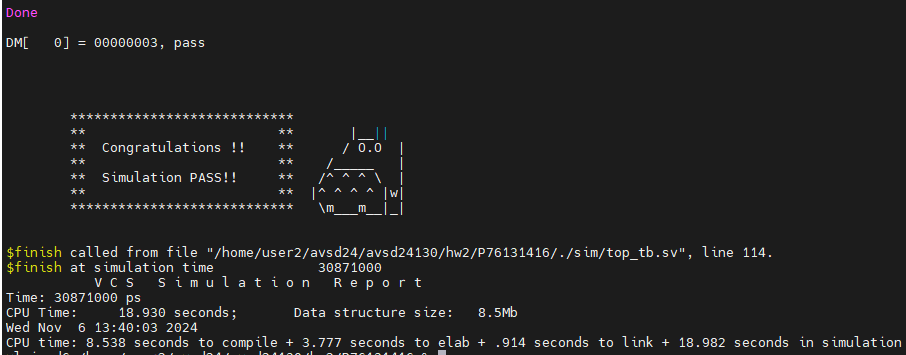
Prog1:



Prog2:



Prog3:



Prog4:

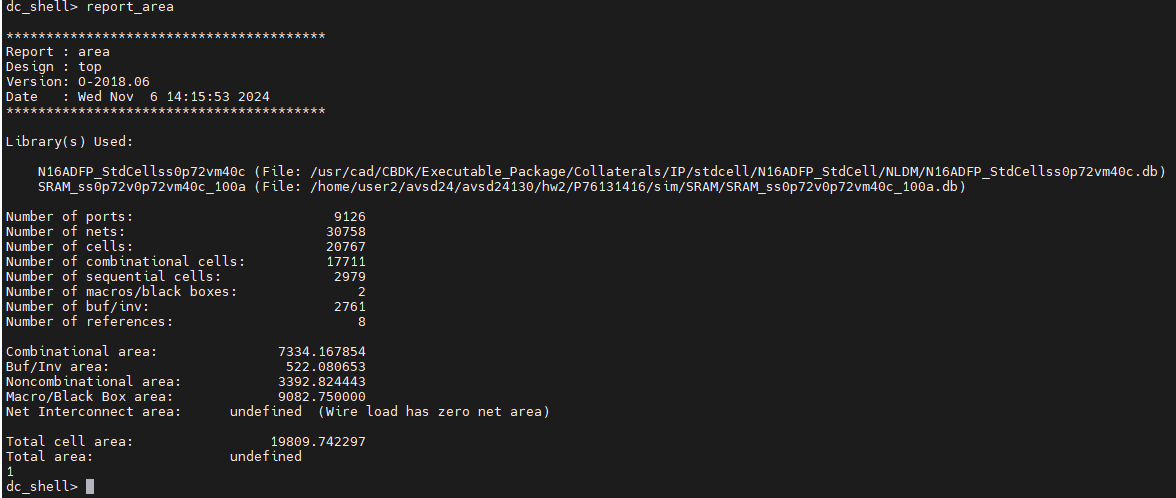


Prog5:

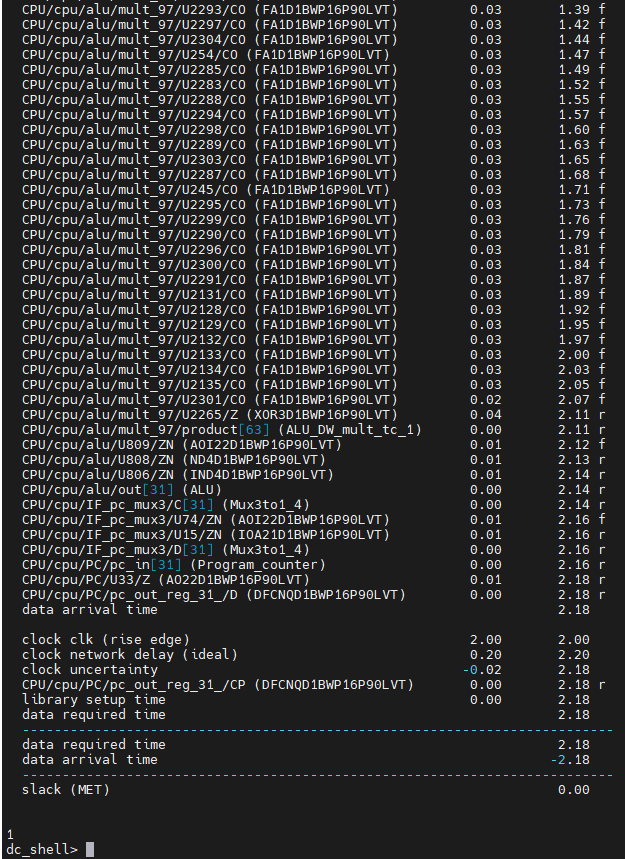


Prog6:



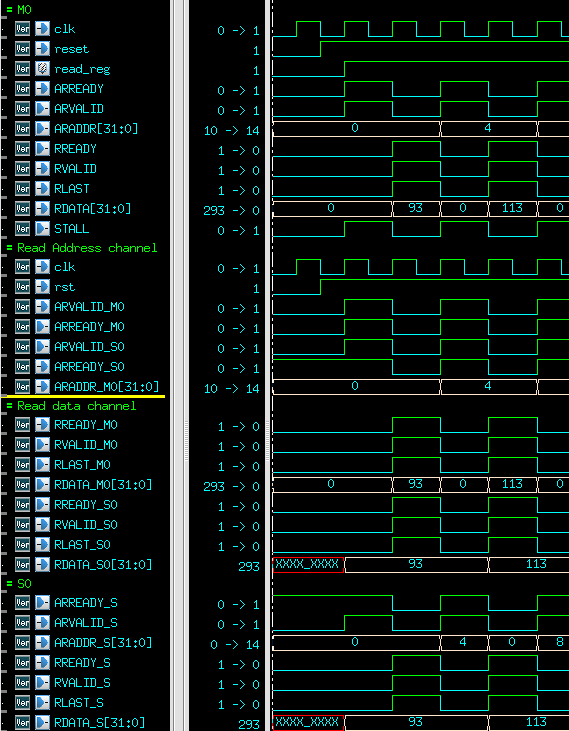
area report:

timing report:



Wave Form:

Instruction Master read data from S0:

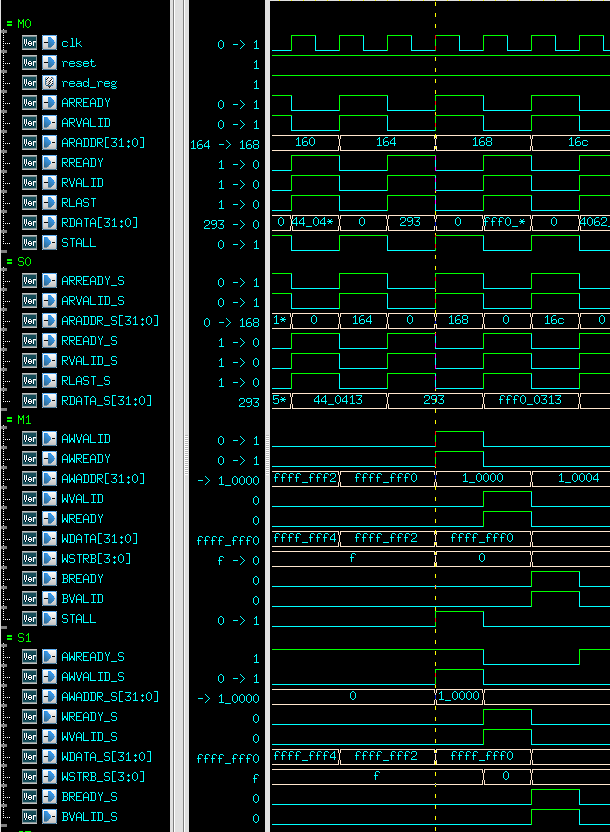


從上圖可以看到reset後，M0要做read的指令，所以發出STALL將CPU停住，並且傳送arvalid與araddr給AXI的read address channel。透過araddr找到要將地址傳送給S0拿資料，所以將arvalid與araddr傳給S0，S0此時因為沒有其他事所以arready為1，因此handshake成功。

下個clock來時，S0藉由rid將rvalid與rdata透過

read data channel傳送給對應的M0，而此時的M0也將rready拉起準備接收rdata，完成handshake後會將stall降下來，讓CPU繼續執行。

Instruction Master read & Data Master write different Slave



從上圖可以看到，此時Instruction Master(M0)是要做read的指令，而Data Master(M1)則是要做write指令。首先M0先發出STALL暫停CPU，傳送arvalid與araddr進入AXI，透過araddr決定要將arvalid與araddr送給S0；M1也發出STALL暫停CPU，傳送awvalid與awaddr給AXI透過awaddr判斷要將awvalid與awaddr給S1；此時S0與S1(剛好沒有其他事所以awready為1)分別發出arready與awready表示handshake成功。

下個clock來時，S0藉由rid將rvalid與rdata透過

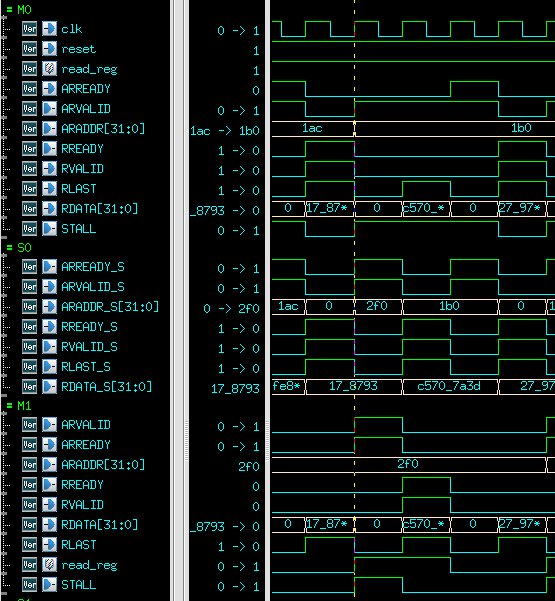
read data channel傳送給對應的M0，而此時的M0也將rready拉起準備接收rdata，完成handshake後會將stall降下來，讓CPU繼續執行；M1會發出wvalid與wdata給AXI傳送給S1，同時S1會發出wready完成handshake。再下一拍時，S1會發出bvalid與bresp給AXI，透過bid將資料傳給Data Master，而Data Master的bready也已經拉起，完成handshake。

若是

Instruction Master read & Data Master read from different Slave

也會跟上面一樣，只不過S1會用到的是read相關的channel。

Instruction Master read & Data Master read from same S



當今天兩個Master都要對相同的Slave做read的動作時，我會先讓Data Master優先做，所以可以看到是M1優先做完read相關handshake後，M0才會開始做，此時M0的stall會全程拉著讓CPU保持暫停狀態。

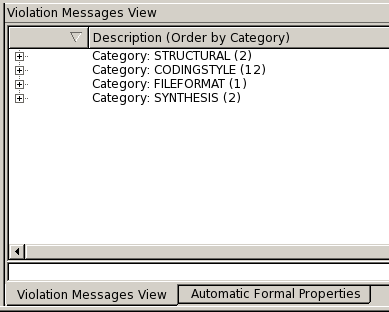
Instruction Master read & Data Master write from same S



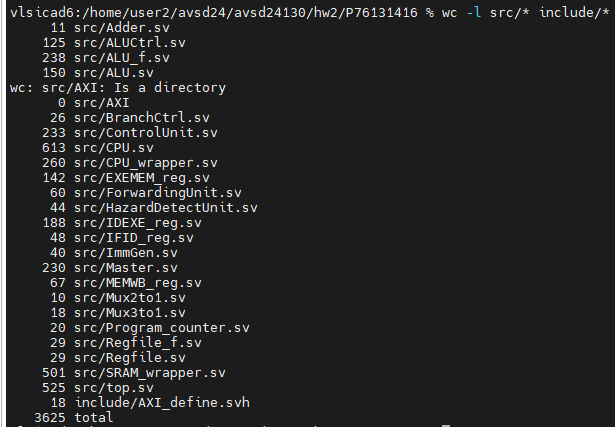
從上圖可以看到，此時M0要對S0進行read動作；而M1要對S0進行write動作。我的設計是會優先讓M1對S0進行write，可以看到M1在做完write的handshake後，M0在過程中會全程將stall拉起暫停CPU，並且等到M0做完read後才會放下。

Superlint result:

Total warning/error:

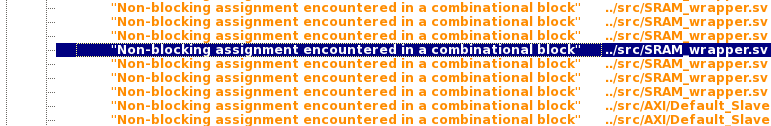


Total line of codes:

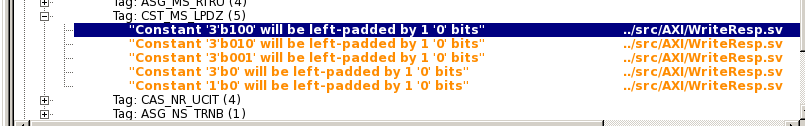


Coverage:99.995%

3 warning/error in my code:



把在combinational內的non-blocking改為blocking。



修改宣告的bit數。



重新修改程式碼，把多餘的地方去除。

Major problem encountered:

在rtl驗證prog1時，發現會將資料寫入S0，原以為只會對S1進行寫入動作，沒想到S0也會有寫入的問題。

Lesson Learned:

1. 理解AXI的5個channel關係與原理，並且利用握手機制進行地址與資料的傳輸及寫入。
2. 將抽象的概念透過理解每個component之間資料是如何傳輸與溝通後，再把他們實作出來並且驗證。
3. 透過使用JasperGold vip進行驗證，了解如何使用jaspergold與輸出的錯誤訊息內容，進一步修改自己的程式碼。
4. 將作業一的CPU接上作業二的AXI bus與SRAM進行溝通。