VLSI System Design (Graduate Level)

Fall 2021

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

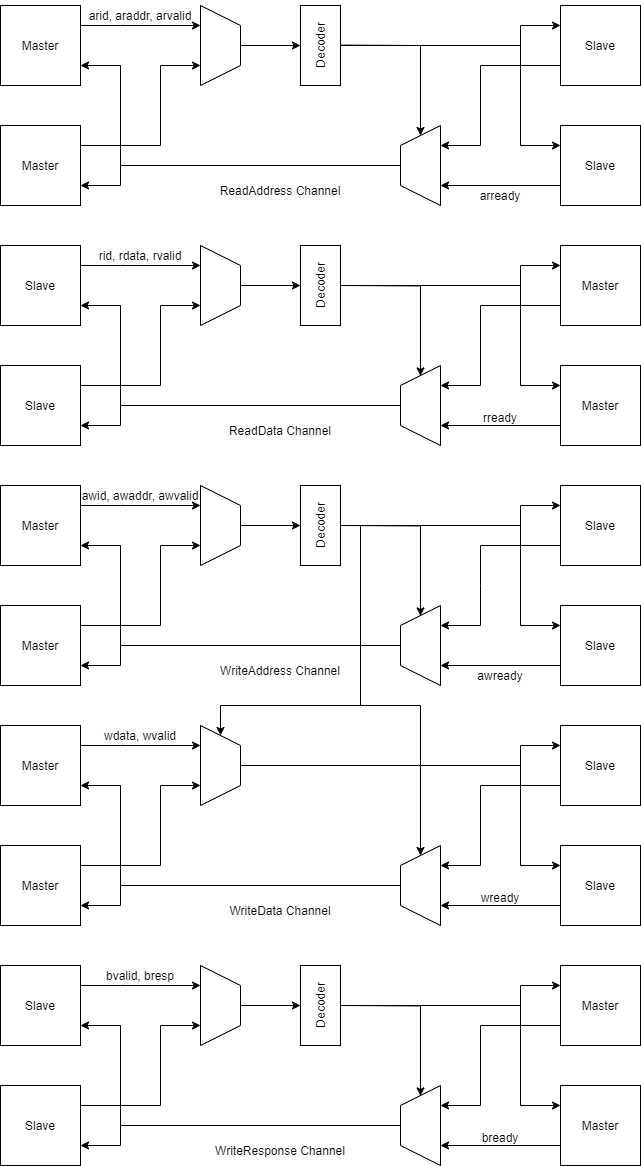
Student name: 楊承翰(70%)

Student ID: NE6091124

Student name: 謝宜紘(30%)

Student ID: P76091161

Problem1: AXI Bus:

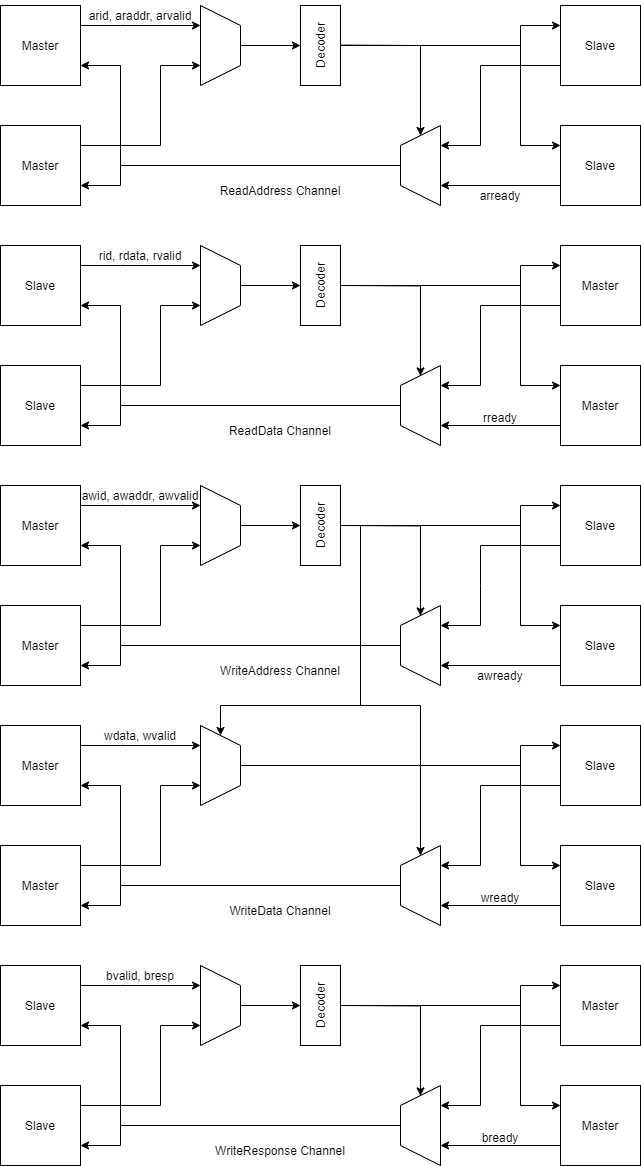
ReadAddress & ReadData:

在我的架構若master想要讀取data，首先master會在read address channel發出arvalid告訴read address channel此筆address是有效的address，而address channel會去對address去做decoder的行為判斷此筆address 是要去access哪一個Slave，而當Slave收到arvalid的訊號時會發出arready的訊號，透過Address channel傳輸到發出arvalid的Master端。此時read Address channel 完成handshaking，代表此次的read address傳輸完成。

而當instruction Master 與 data Master 同時去access read address channel時，在我的架構上為data Master的優先權較高，也就是說會先讓data Master 優先去存取read address channel，等到data Master完成handshaking時才會讓instruction Master去access read address channel。

當read address的transition完成時，被選中的Slave會根據address去準備Master所需要的data，當Slave將data準備好時會先去拆解從read address channel所傳送過來的arid去判斷要將data送往哪一個Master。當data準備好時，Slave會透過read data channel 發出rvalid去通知Master端資料已經準備好了，此時需要這筆data的Master端也會透過read data channel 發出rready去通知Slave可以將資料送過來了。此時read data channel完成handshaking，代表此次整個read transition已經完成。

WriteAddress, WriteData & WriteResponse:



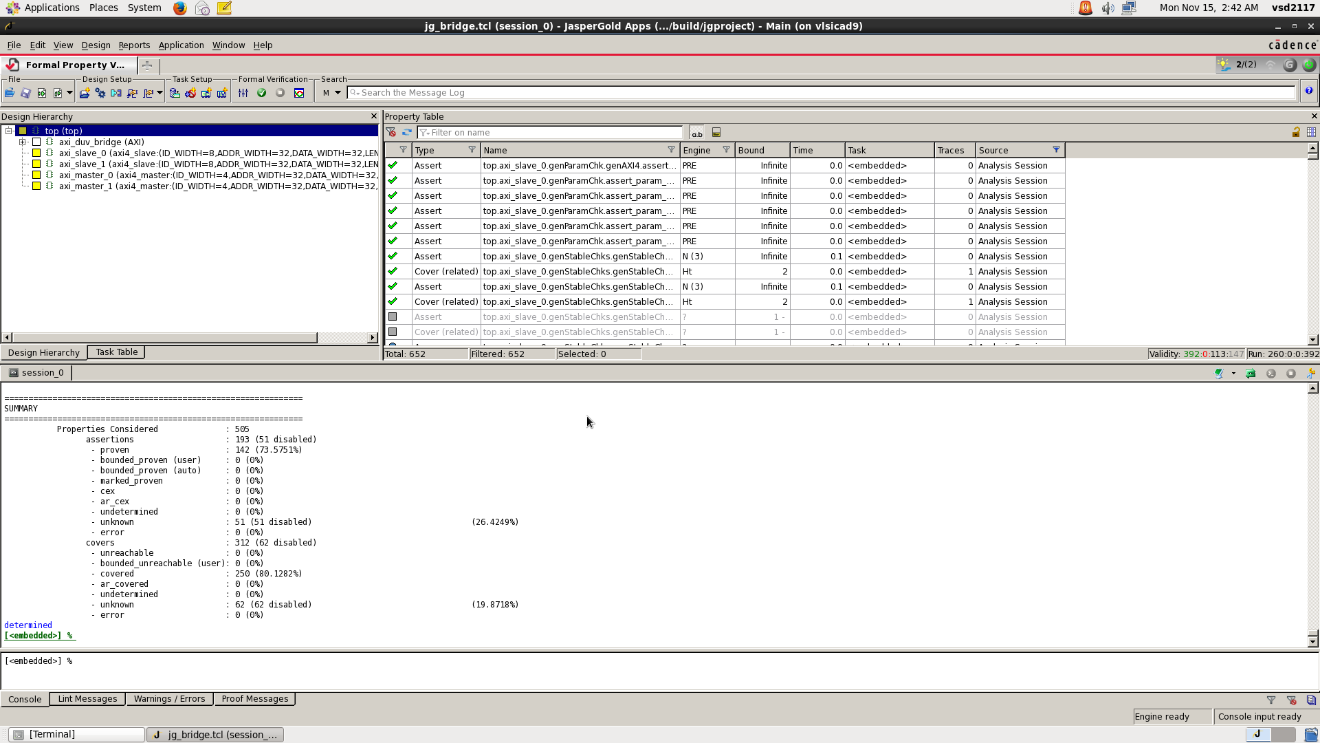
若Master想要去對Slave進行寫入的動作，會對write address channel發出awvalid，代表Master想要做寫入的動作，而且此筆的write address是一筆有效的write address，而write address channel會將這筆address去做解碼判斷Master想要去對哪一個Slave做寫入的動作， Slave接受到awvalid時，會透過Write address channel發出awready通知Master此筆write address 已經傳輸完成。此時write Address channel 完成handshaking，代表此次的write address傳輸完成。

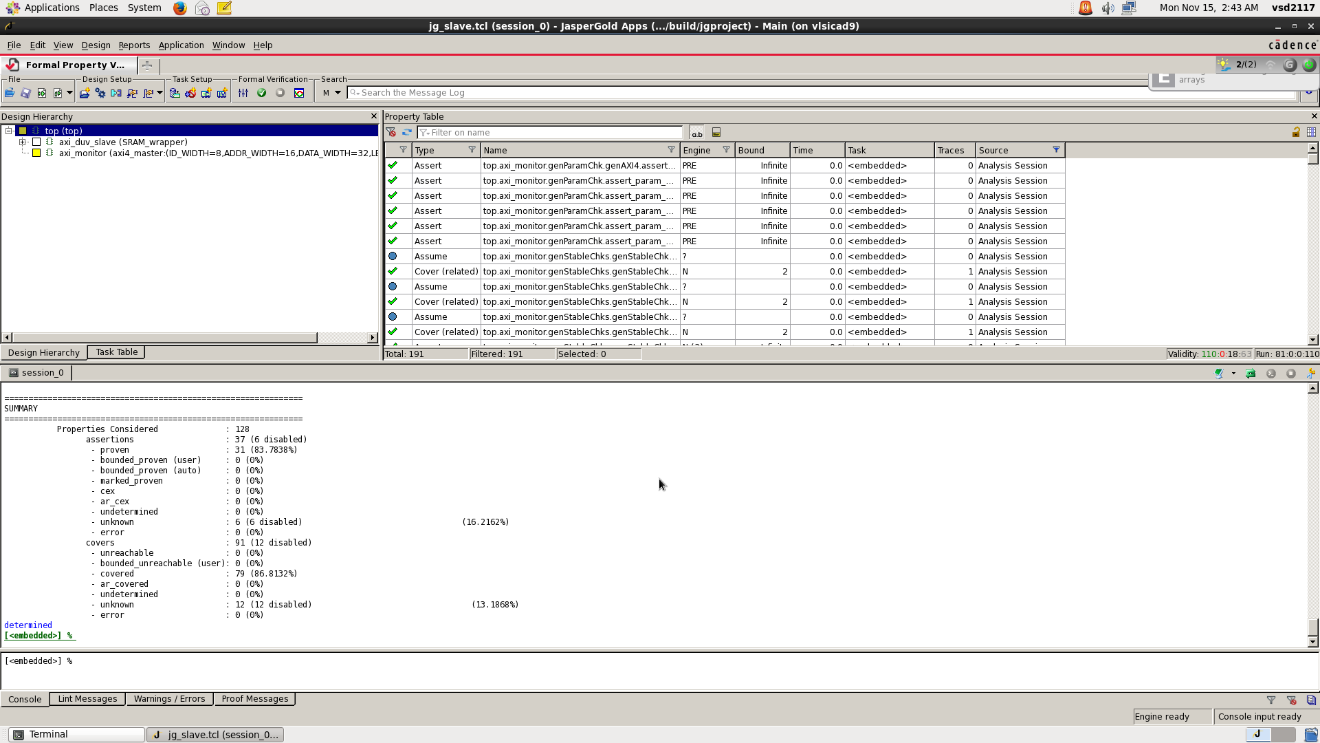
Master判斷write address有完成Handshaking會將write data準備好並透過Write data channel發出wvalid，而此時會透過剛剛write address channel Arbiter去決定我的write data的來源是要來自於哪一個Master。當Slave 接受到wvalid時，會透過剛剛從write address channel接受到的awrid去分析要將awready送往哪一個Master。此時write data channel 完成handshaking，代表此次的write data傳輸完成。

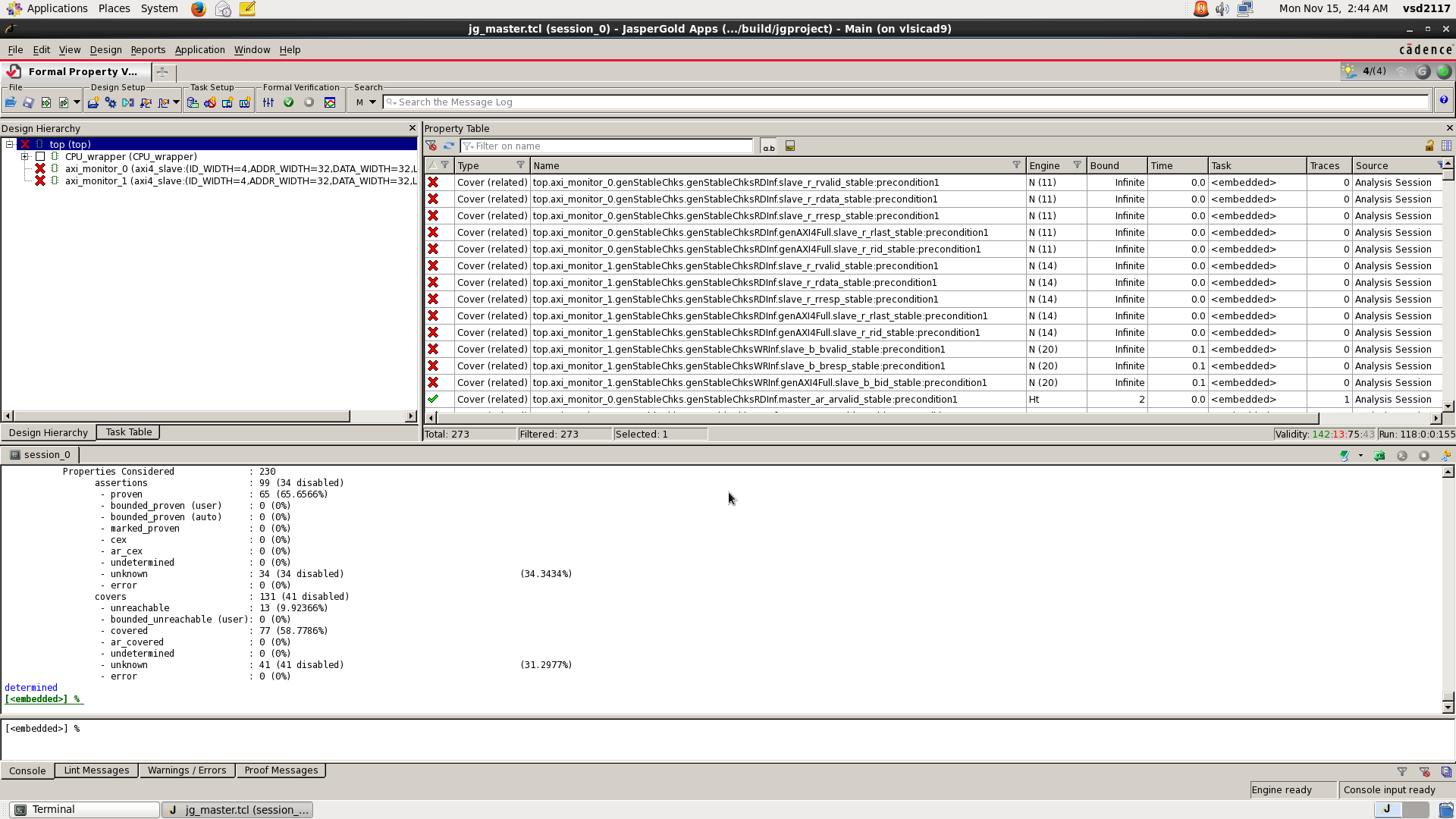
當Slave判斷write data完成handshaking的傳輸時，才會開始準備write response。write的行為完成時，Slave會透過write response channel 發出bvalid去通知Master已經完成write 的transition，此時當Master接受到bvalid時，Master會發出bready去通知Slave接受到write response了。此時write response channel 完成handshaking，代表此次整個write transtion傳輸完成。

Jaspergold Verification:

Bridge: all pass



Slave: all pass

Master: 13 cover violation

|  |
| --- |
| rvalid stable cover violation |
| rdata stable cover violation |
| rresp stable cover violation |
| rlast stable cover violation |
| rid stable cover violation |
| bvalid stable cover violation |
| bresp stable cover violation |
| bid stable cover violation |

在Master有些cover並沒有辦法通過驗證，這是因為在write response與read response，我的Master只有當Slave發出valid訊號時才會同時將ready拉起來，所以導致某些cover會沒有辦法驗證出來。

在write response，我們可以看到stable的條件皆為當bvalid為1時且bready為0時我在Master接受的bid、bvalid、bresp皆不能做改變。但在我的電路設計並不會出現bvalid為1且bready為0的情況。因為我的Master設計是當write data完成handshaking的當下就將bready拉起來，所以導致並不會出現bvalid為1且bready為0的情況。

而在read response也是因為跟write response相同的情況，也就是說我的rready在完成read address的handshaking時，就會馬上拉起來。因此並不會出現rvalid為1且rready為0的情況，所以才會沒辦法通過rid、rlast、rdata、rresp、rvalid的cover。

Problem2: CPU:

RTL & Gate-level simulation:

RTL: Gate-level:

Program0:

一張含有 文字 的圖片

自動產生的描述 一張含有 文字 的圖片

自動產生的描述

Program1:

一張含有 文字 的圖片

自動產生的描述 一張含有 文字 的圖片

自動產生的描述

Program2:

一張含有 文字 的圖片

自動產生的描述 一張含有 文字 的圖片

自動產生的描述

Program3:

一張含有 文字 的圖片

自動產生的描述 一張含有 文字 的圖片

自動產生的描述

Area report:

一張含有 文字 的圖片

自動產生的描述

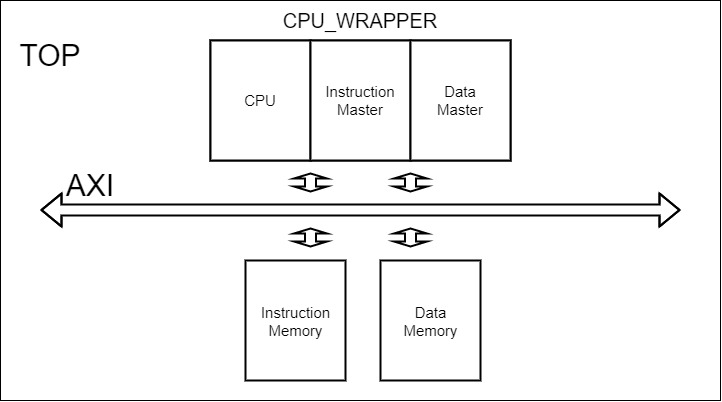
Timing report: cycle = 7.4

一張含有 文字 的圖片

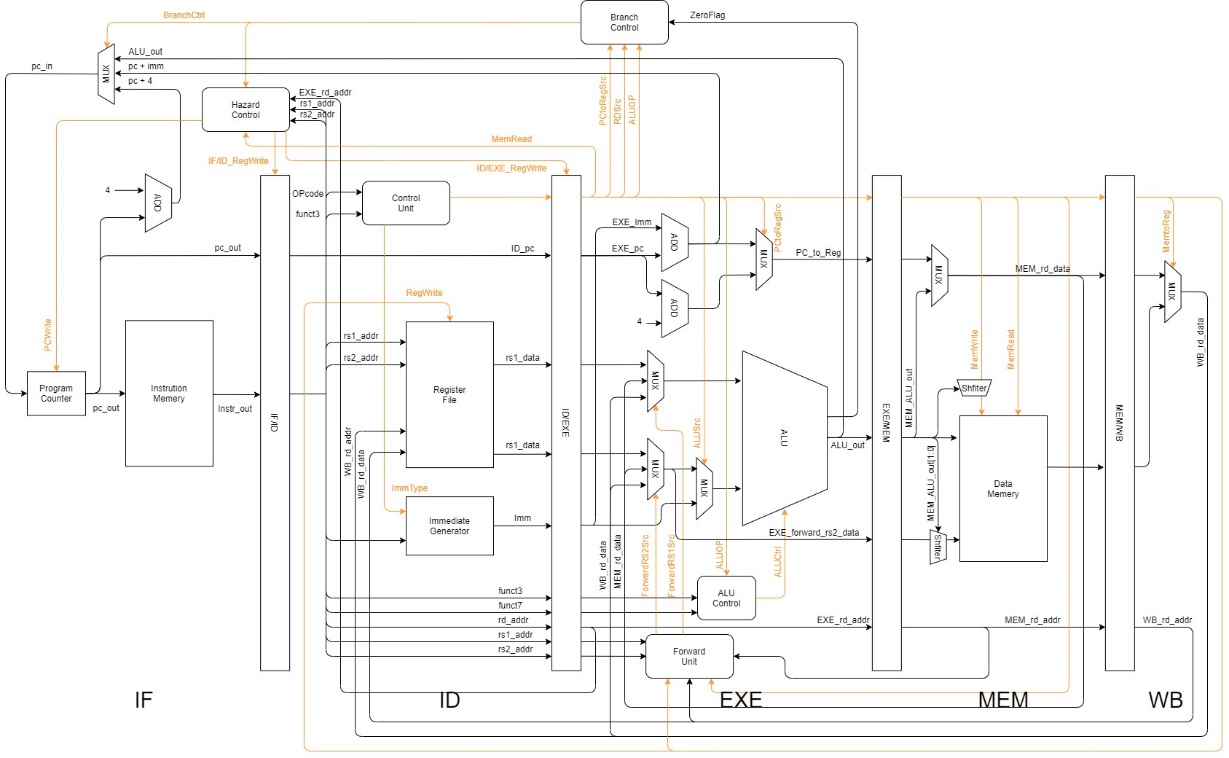
自動產生的描述

Block Diagram:

TOP:



CPU:



Wave form:

這次的作業主要是將AXI加入CPU裡面，所以大部分指令的波型圖都會是相同的，主要會解釋當今天Instruction Master跟Data Master各別去向memory存取資料時AXI的行為。

Only Instruction Master read data from Slave:

一張含有 文字, 電子用品, 電腦 的圖片

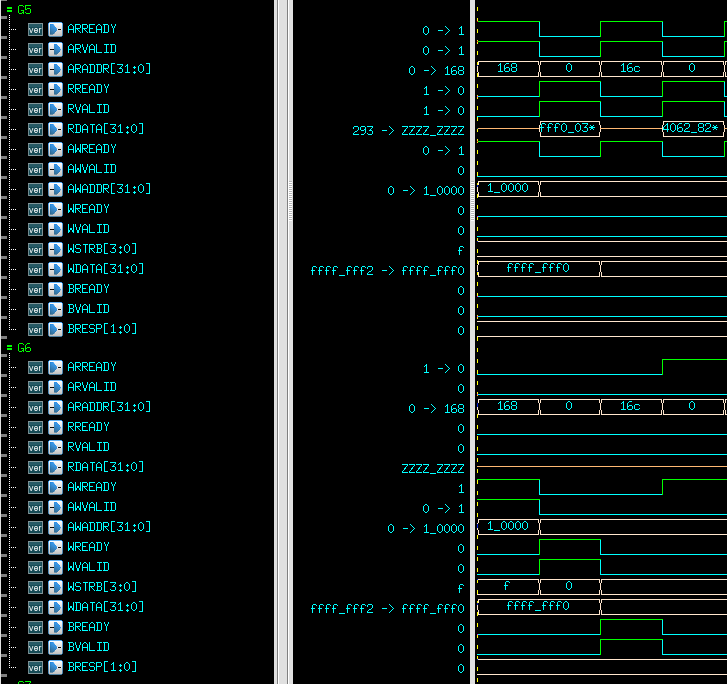
自動產生的描述

reset訊號拉下來，Program Counter開始從0x0計算，會先送Instr\_read以及PC\_out給Instruction Master，Instruction Master會立即把arvalid以及araddr透過AXI傳給Slave1，並把stall拉起來暫停其他所有的module的動作，而Slave1因為當時並沒有在處理其他事情所以arready訊號為1，所以在當下就會完成handshake。

下一個clk來時Slave1將對應的rdata以及rvalid藉由rid傳回去Instruction Master，而Instruction Master也已經將rready拉起來準備好接收rdata，完成handshake，並且將stall拉下來，讓其他module回復運作。

Instruction Master read from & Data Master store in different Slaves





當今天Instruction Master read & Data Master store from different Slaves因為AXI的5個channel並不會互相干擾，所以能夠同時進行。

Program Counter送Instr\_read以及PC\_out給Instruction Master，Instruction Master會立即把arvalid以及araddr透過AXI傳給Slave1，並把stall拉起來暫停其他所有的module的動作，而Slave1因為當時並沒有在處理其他事情所以arready訊號為1，所以在當下就會完成handshake。

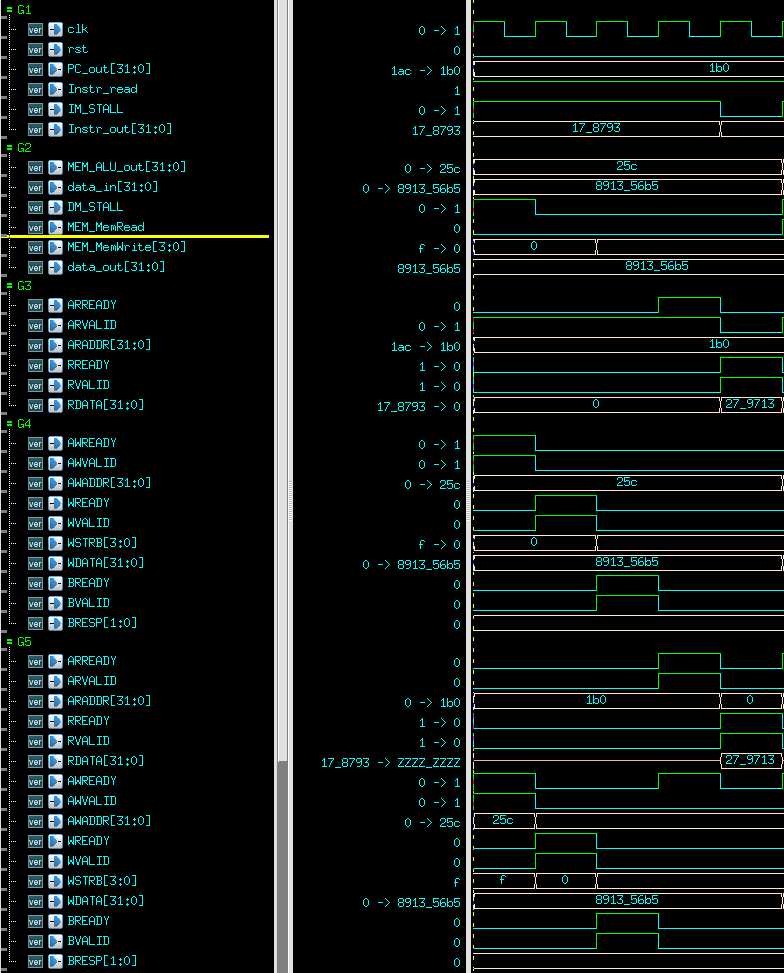
同時CPU正好要將0xfffffff0存進位址0x10000，所以會將data\_in、MemWrite以及Mem\_ALU\_out傳給Data Master，Data Master會立即把awvalid以及awaddr透過AXI傳給對應的Slave，並把stall拉起來暫停其他所有的module的動作，因為剛好對應的是Slave2，此時Slave2並沒有在處理其他事情，所以awready訊號為1，當下就會完成handshake。

下一個clk，Slave1將對應的rdata以及rvalid藉由rid傳回去Instruction Master，而Instruction Master也已經將rready拉起來準備好接收rdata，完成handshake，並且將stall拉下來，讓其他module回復運作。

同時Data Master將wdata以及wvalid傳給Slave2，而Slave2也已經將wready拉起來準備好接收wdata，完成handshake，並將data寫入對應的位址。

再下一個clk，Slave2藉由bid回傳bvalid以及bresp給Data Master，告訴Data Master完成寫入，而Data Master的bready也已經拉起來並完成handshake。

Instruction Master read from & Data Master store in same Slave



當今天Instruction Master read from & Data Master store in same Slave，我這邊設計的Slave會讓store指令優先執行。

Program Counter送Instr\_read以及PC\_out給Instruction Master，Instruction Master會立即把arvalid以及araddr傳給AXI，並把stall拉起來暫停其他所有的module的動作，而當時Data Master剛好也要對Slave1做store動作，所以arready會拉下來，Instruction Master會在此時將arvalid一直拉著等待handshake。

此時CPU正好要去寫位址0x25c的值，所以會將MemWrite、data\_in以及Mem\_ALU\_out傳給Data Master，Data Master會立即把awvalid以及awaddr透過AXI傳給對應的Slave，並把stall拉起來暫停其他所有的module的動作，此時Slave1並沒有在處理其他事情，所以awready訊號為1，雖然在這當下Instruction Master也同樣將arvalid以及araddr傳給AXI，但我這邊的設計Slave會讓store指令有優先權，所以當下就能回傳awready並完成handshake。

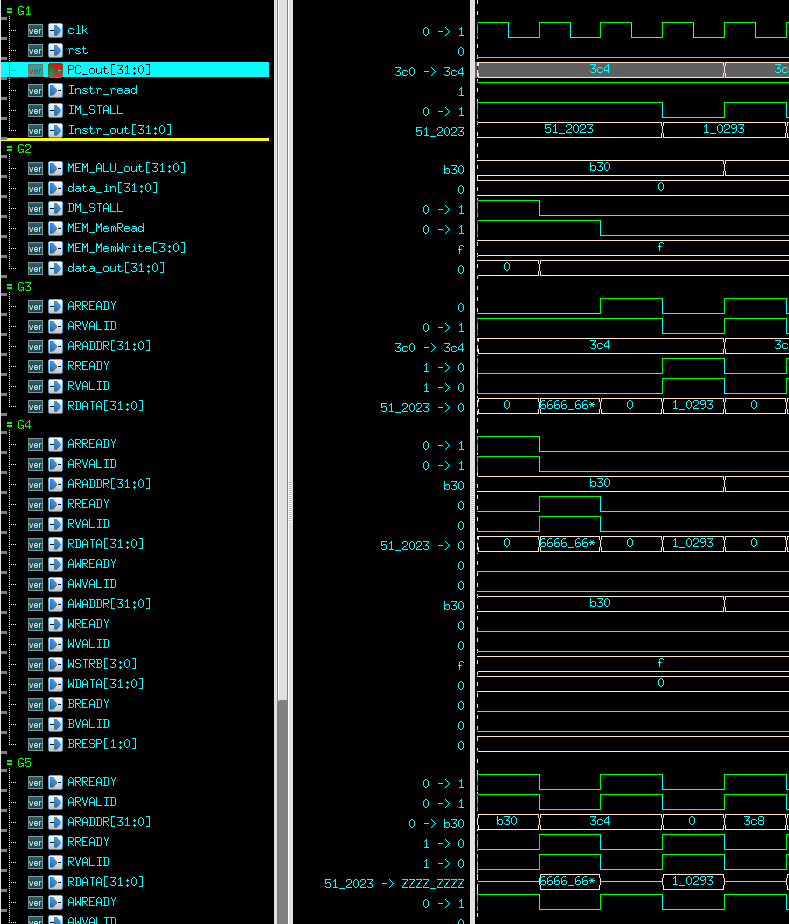
下一個clk，Data Master將wdata以及wvalid傳給Slave1，而Slave1也已經將wready拉起來準備好接收wdata，完成handshake，並將data寫入對應的位址。

再下一個clk，Slave1藉由bid回傳bvalid以及bresp給Data Master，告訴Data Master完成寫入，而Data Master的bready也已經拉起來並完成handshake。

再下一個clk，Slave1完成store指令，arready再次拉起來，此時Instruction Master收到後完成handshake。

再下一個clk，Slave1將對應的rdata以及rvalid藉由rid傳回去Instruction Master，而Instruction Master也已經將rready拉起來準備好接收rdata，完成handshake，並且將stall拉下來，讓其他module回復運作。

Instruction Master read & Data Master read from same Slave



當今天Instruction Master read & Data Master read from same Slave，我這邊設計的AXI會讓Data Master 優先read。

Program Counter送Instr\_read以及PC\_out給Instruction Master，Instruction Master會立即把arvalid以及araddr傳給AXI，並把stall拉起來暫停其他所有的module的動作，而Slave1當時並沒有在處理其他事情所以arready訊號為1，但是因為Data Master剛好也要對Slave1做read動作，所以arready並沒有傳回給Instruction Master，Instruction Master會在此時將arvalid一直拉著等待handshake。

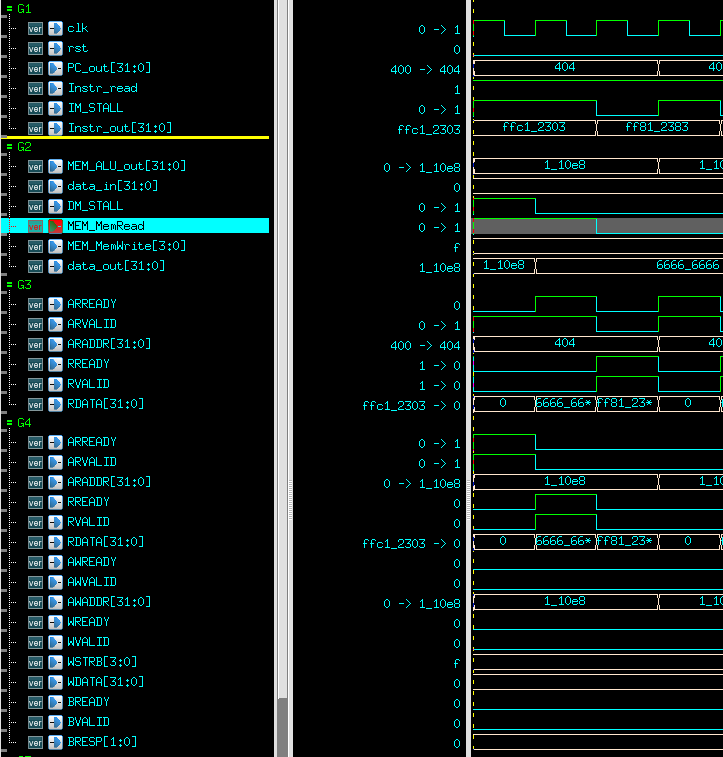
此時CPU正好要去讀位址0xb30的值，所以會將MemRead以及Mem\_ALU\_out傳給Data Master，Data Master會立即把arvalid以及araddr透過AXI傳給Slave1，並把stall拉起來暫停其他所有的module的動作，此時Slave1並沒有在處理其他事情，所以arready訊號為1，雖然在這當下Instruction Master也同樣將arvalid以及araddr傳給AXI，但我這邊的AXI設計會讓Data Master有優先權，所以當下就能收到arready並完成handshake。

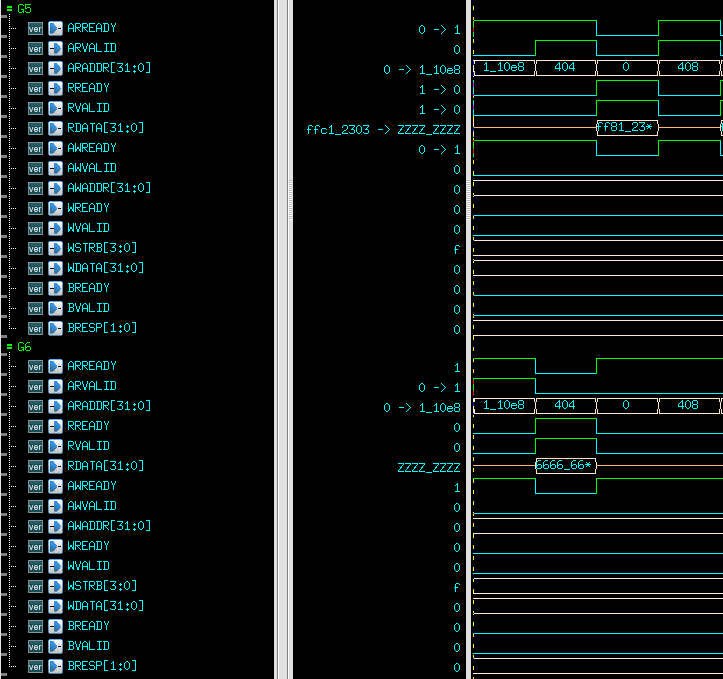
下一個clk來時Slave1將對應的rdata以及rvalid藉由rid傳回去Data Master，而Data Master也已經將rready拉起來準備好接收rdata，完成handshake，並且將stall拉下來，讓其他module回復運作。

再下一個clk，Slave1完成Data Master的read，所以arready拉回1，而此時AXI會將arready傳給Instruction Master，完成handshake。

再下一個clk，Slave1將對應的rdata以及rvalid藉由rid傳回去Instruction Master，而Instruction Master也已經將rready拉起來準備好接收rdata，完成handshake，並且將stall拉下來，讓其他module回復運作。

Instruction Master read & Data Master read from different Slaves





當今天Instruction Master read & Data Master read from different Slaves，我這邊設計的AXI會讓Data Master 優先read。

Program Counter送Instr\_read以及PC\_out給Instruction Master，Instruction Master會立即把arvalid以及araddr傳給AXI，並把stall拉起來暫停其他所有的module的動作，而Slave1當時並沒有在處理其他事情所以arready訊號為1，但是因為Data Master剛好也要做read動作，所以arready並沒有被AXI傳回給Instruction Master，Instruction Master會在此時將arvalid一直拉著等待handshake。

此時CPU正好要去讀位址0xb30的值，所以會將MemRead以及Mem\_ALU\_out傳給Data Master，Data Master會立即把arvalid以及araddr透過AXI傳給對應的Slave，並把stall拉起來暫停其他所有的module的動作，此時Slave2並沒有在處理其他事情，所以arready訊號為1，雖然在這當下Instruction Master也同樣將arvalid以及araddr傳給AXI，但我這邊的AXI設計會讓Data Master有優先權，所以當下就能收到arready並完成handshake。

下一個clk來時Slave2將對應的rdata以及rvalid藉由rid傳回去Data Master，而Data Master也已經將rready拉起來準備好接收rdata，完成handshake，並且將stall拉下來，讓其他module回復運作。同時AXI會將Slave1的arready傳給Instruction Master，完成handshake。

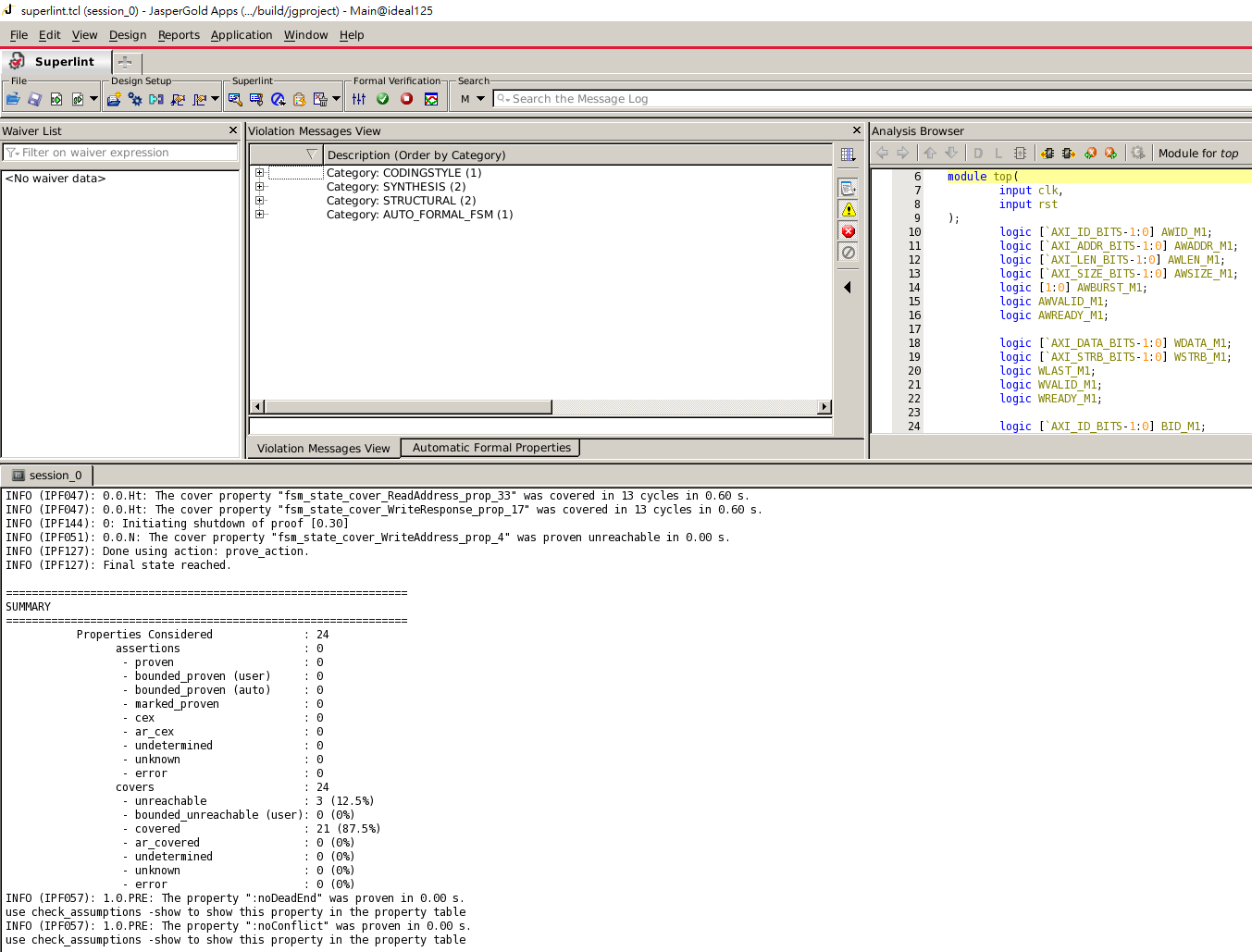
再下一個clk，Slave1將對應的rdata以及rvalid藉由rid傳回去Instruction Master，而Instruction Master也已經將rready拉起來準備好接收rdata，完成handshake，並且將stall拉下來，讓其他module回復運作。

Superlint result:

Total warning/errors: 6

Total lines of code: 3474

Coverage:99.8%



一張含有 文字 的圖片

自動產生的描述 一張含有 文字 的圖片

自動產生的描述

C program implement:

沿用作業一的C program

Lessons learned:

1. 理解AXI bus機制 ，知道 AXI Bus 之五個 channel 如何運作，如何透過 handshake 機制來確保位置及資料的傳輸。
2. 理解到 AXI bus 中每個 component 的作用與定位，interface 、Arbiter、 Decoder 每個 component 之間資料是如何傳輸及溝通。
3. 透過抽象的 spec ，實做出符合 spec 要求的電路。在實作電路之前，最好先徹底理解 spec 中的定義，每個 channel 有哪些腳位，每隻腳位各自的功能為何，腳位跟腳位之間如何對應。
4. 使用 Jaspergold vip 驗證。了解 jaspergold vip 如何使用，運作原理機制為何。了解 assert 跟 cover 的區別，並透過閱讀 source 之檢查規範進一步修改自己的 rtl code。
5. 將 CPU 接上 Bus ，透過 bus 與 SRAM 溝通。