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: \_蔡明翰\_ \_曾柏硯\_

Student ID: \_N26120579\_ \_M16121027\_

目錄

**一、Design Explanation**

**二、Simulation Results**

三、**Synthesize**

四、**JasperGold**

**五、Superlint**

**六、心得和問題解決**

**Summary**

|  |  |
| --- | --- |
| Clk Period | 10ns |
| Area | 5903296um2 |
| Data Require Time | 10.79ns |
| Power | 126.7759mW |
| JasperGold | Pass |
| Superlint Coverage | 99.735% |
| Design Explanation | V |
| RTL simulation | pass |
| SYN simulation | pass |
| Waveform | V |
| Synthesize Results | V |
| 問題說明和解決 | V |

|  |  |  |
| --- | --- | --- |
| Name | Percentage | Parts |
| 蔡明翰 | 50% | CPU\_wrapper、SRAM\_wrapper、wrapper說明、vip和合成驗證 |
| 曾柏硯 | 50% | AXI、AXI說明、波形驗證 |

1. **Design Explanation**

當CPU設計完成後，可能會需要擴充記憶體，但又無法直接更改port的大小，故需要透過AXI通道排程各個記憶體讀取和寫入的順序，在兩個記憶體都發出request時，使其能夠輪流使用，以實現round-robin的設計，並同時read和write兩channel可同時進行不同MEM的read和write，以提升傳輸速度。AXI的核心價值在於handshake，作為訊號是否能傳遞的根據，此次作業僅考慮outstanding=1之情況，沒有增加ooo之設計，但之後也會加入考慮其他情況的設計。

一張含有 文字, 圖表, 螢幕擷取畫面, 方案 的圖片

自動產生的描述整體架構:

1. AXI\_Master
2. READ channel:

IDLE state: 當read\_en訊號拉起時，拉起ARVALID訊號，並檢查ARREADY訊號是否拉起，進行handshake確認，是則進入R state並拉起RREADY訊號；否則進入AR state。read\_en拉起後，即拉起stall訊號。

AR state: 繼續拉起ARVALID訊號，並等待ARREADY訊號拉起，是則進入R state；否則則繼續維持在AR state。皆拉起stall訊號。

R state: 拉起RREADTY訊號，並確認RVALID和RLAST訊號是否拉起，是則回到IDLE state，並解除stall訊號；否則繼續維持在R state，繼續拉起stall訊號。

一張含有 文字, 圖表, 圓形, 螢幕擷取畫面 的圖片

自動產生的描述

1. WRITE channel:

IDLE state: 當write\_en訊號拉起時，拉起AWVALID訊號，並檢查AWREADY訊號是否拉起，進行handshake確認，是則進入W state並拉起WREADY訊號；否則進入AW state。write\_en拉起後，即拉起stall訊號。

AW state: 繼續拉起AWVALID訊號，並等待AWREADY訊號拉起，是則進入W state；否則則繼續維持在AW state。皆拉起stall訊號。

W state: 拉起WVALID訊號，確認WREADY訊號是否拉起，是則進入B state；否則繼續維持在W state，繼續拉起stall訊號。

B state: 拉起BREADY訊號，確認BVALID訊號是否拉起，是則回到IDLE state，並解除stall訊號；否則，維持B state，繼續拉起stall訊號。

一張含有 文字, 圖表, 圓形, 螢幕擷取畫面 的圖片

自動產生的描述

1. AXI\_Slave

IDLE state: 初始狀態，因為同個MEM讀取和寫入不會同時發生，但因為IM僅供讀取使用，故設計讀取優先。當ARVALID拉起，進入R state，進行讀取資料的傳輸；若ARVALDI未拉起且AWVALID拉起，則進入W state，進行寫入資料的傳輸；若兩訊號皆未拉起，則維持在IDLE state。

R state: 傳輸讀取資料，直到RREADY和RLAST皆拉起，結束傳輸，回到IDLE state。

W state: 傳輸寫入資料，當WVALID和WLAST訊號皆拉起時，進入B state；否則維持在W state。

B state: 回應，當BREADY資料拉起時，表示回應並傳輸完全結束，回到IDLE state，否則繼續維持在B state。

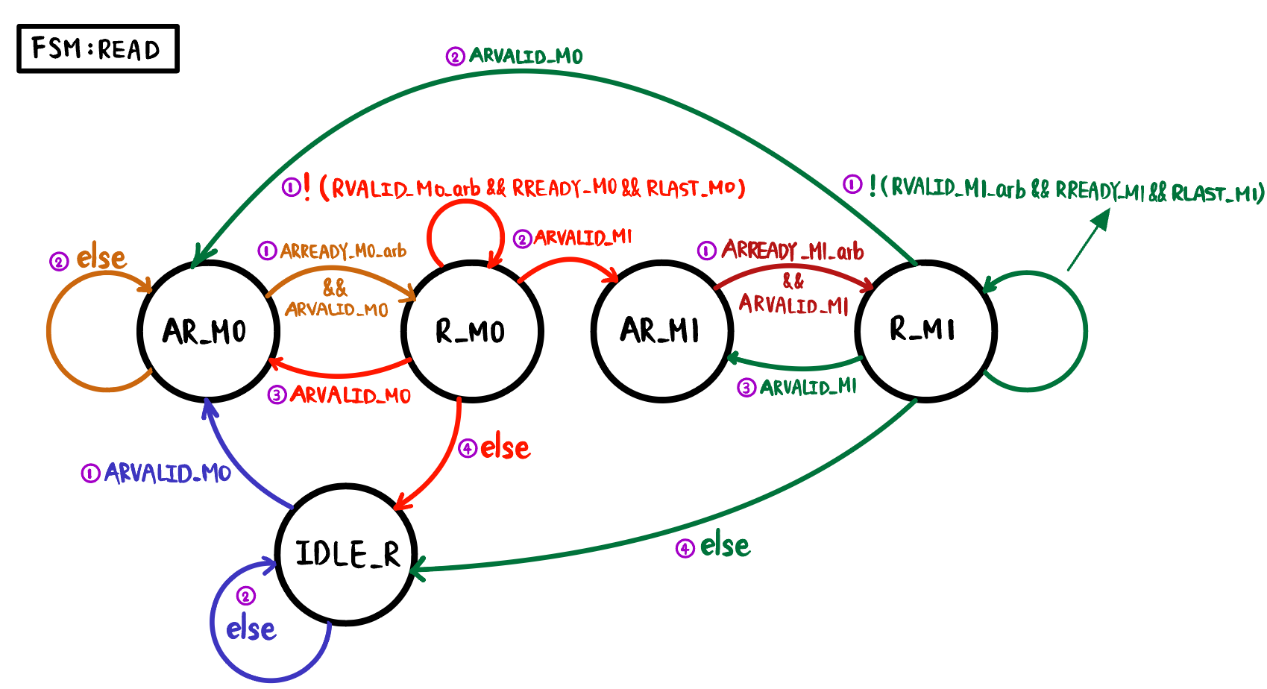
一張含有 文字, 圖表, 圓形, 螢幕擷取畫面 的圖片

自動產生的描述

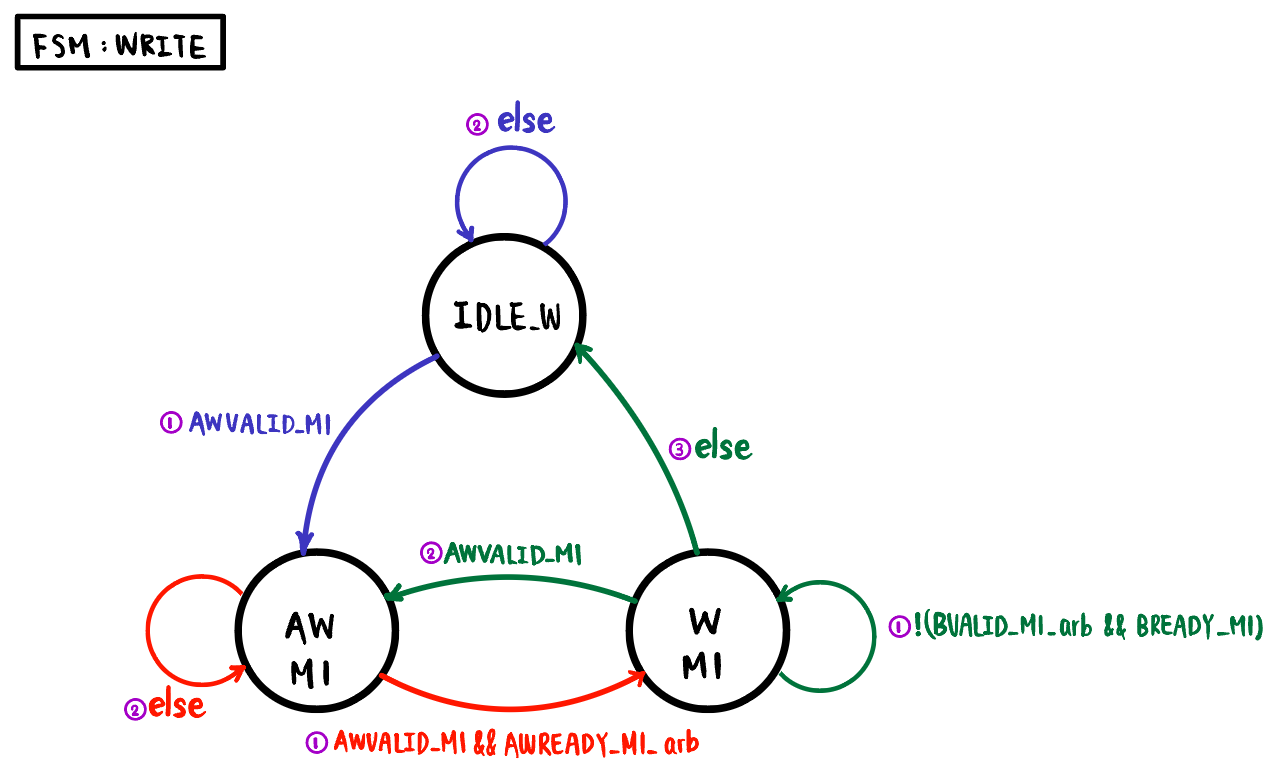
3.AXI\_Bridge

我們的AXI將Read與Write以不同的FSM進行控制，因此可以同時處理讀與寫的傳輸。

1. Read: 在我們的設計中，設定CPU M0的read prioirty高於CPU M1，因為CPU執行任何指令之前都必須都必須讀取IM，而DM是否需要讀取則是取決於目前在memory stage的指令是否為L -ype指令，因此當狀態在IDLE\_R時接收到ARVALID\_M0時，下一個cycle進入AR\_M0狀態（即使ARVALID\_M1也為有效），並在AR\_M0狀態等待Address read channel的handshake完成，否則繼續保持在AR\_M0狀態，handshake完成後在下一個cycle進入R\_M0狀態進行資料讀取，接著將保持在R\_M0狀態一直到拉起RLAST\_M0，表示是目前讀資料傳輸的最後一筆資料，同一時刻立即偵測是否需要讀取DM(以ARVALID\_M1)判斷，若ARVALID\_M1為1則表示此時在mem\_stage的指令為L type指令，將在下一個cycle時進入AR\_M1狀態讀取地址，否則回到IDLE\_R狀態等待下一次的request，若進入AR\_M1狀態，則與在AR\_M0狀態相同，必須等待Address read channel的handshake完成後進入R\_M1狀態，接著將保持在R\_M1狀態一直到拉起RLAST\_M1時，偵測是否需要讀取IM(以ARVALID\_M0)判斷，若ARVALID\_M0為1則直接進入AR\_M0狀態而不需要再回到IDLE\_R等待，重複以上步驟直到結束。

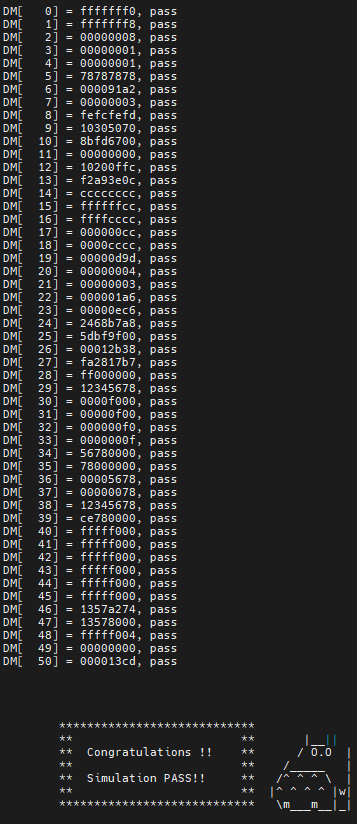


1. Write: 本次作業只有M1可以進行寫入S0或S1，當狀態在IDLE\_W時接收到AWVALID\_M1時，下一個cycle進入AW\_M1狀態，等待Address write channel的handshake完成，否則繼續保持在AW\_M1狀態，handshake完成後在下一個cycle進入W\_M1狀態進行資料寫入，接著將保持在W\_M1狀態一直到完成B channel handshake完成，表示已完成資料寫入，最後判斷是否需要繼續寫入，若AWVALID\_M1拉起則回到AW\_M1，若無則回到IDLE\_W等待下一次寫入請求。

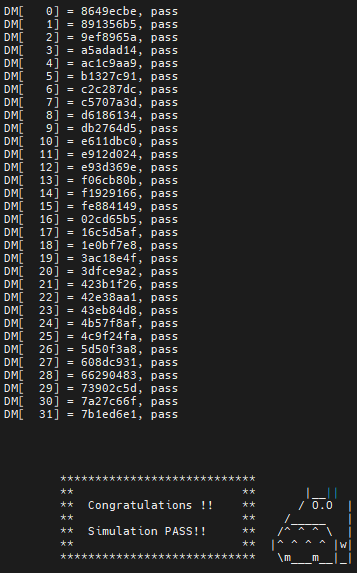


1. **Simulationtion Results**
2. **Pre-sim** (clk period: 10ns)

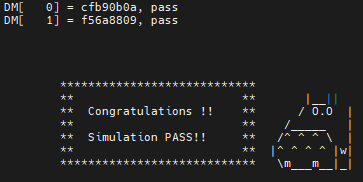
Prog0



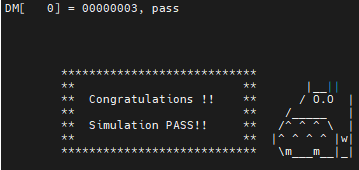
Prog1



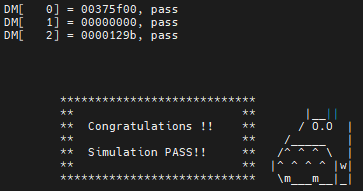
Prog2



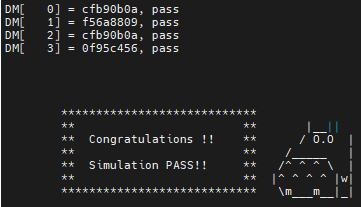
Prog3



Prog4

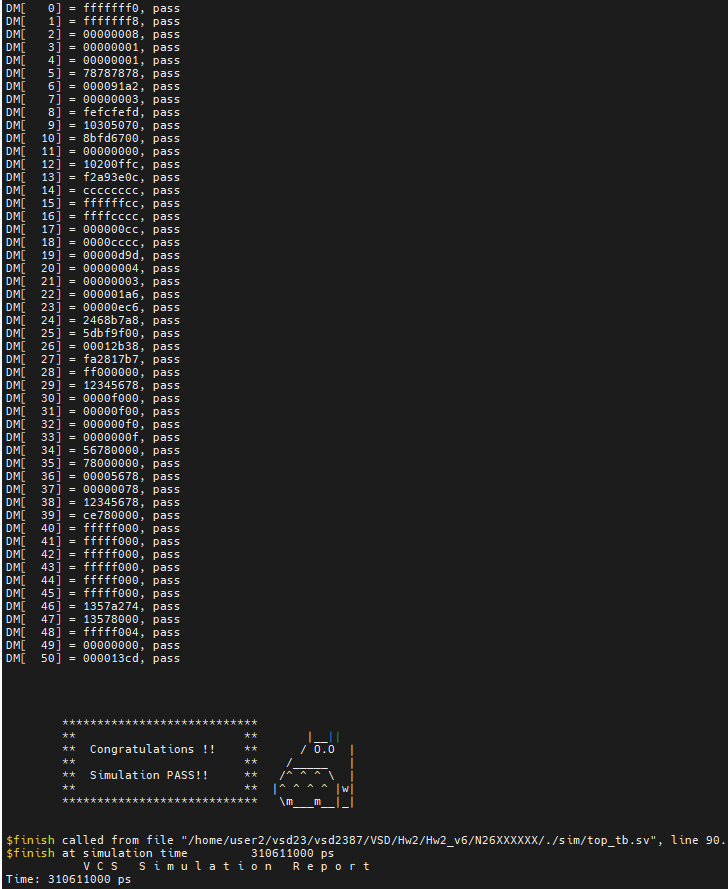


Prog5

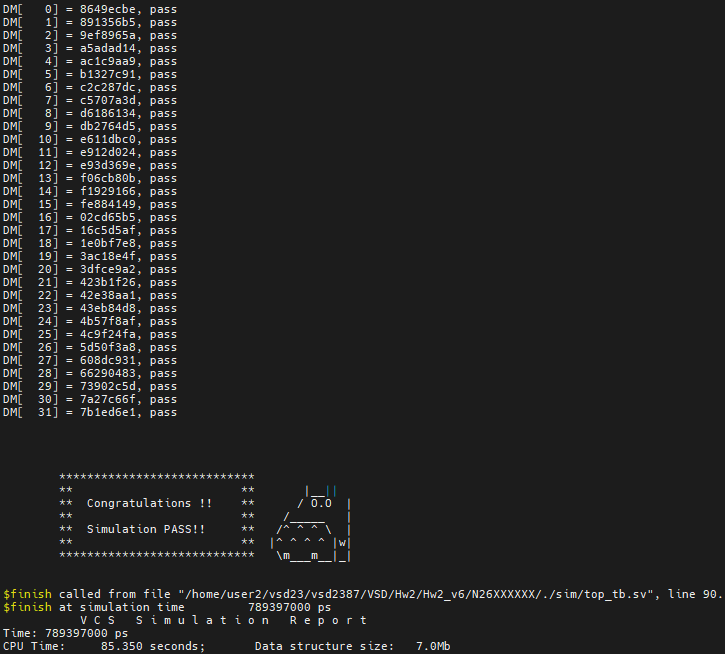


1. **Post-sim** (clk period: 10ns)

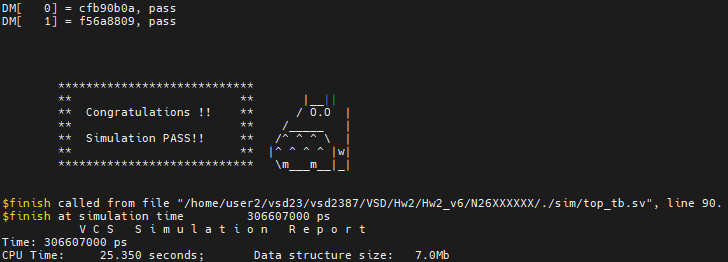
Prog0: 310611000ps



Prog1: 789397000ns



Prog2: 306607000ps



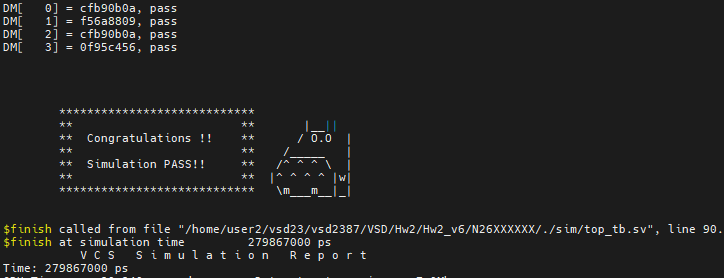
Prog3:373177000ps



Prog4: 300713000ps



Prog5: 279867000ps



1. **Waveform**
2. 【Rtype】 add : 測試 add 指令。

一張含有 螢幕擷取畫面, 文字, 軟體, 多媒體軟體 的圖片

自動產生的描述

【說明】當 opcode = 0110011，alu\_op = 00000，執行add。

rs1 = -3(ffff\_fffd)，rs2 = -1(ffff\_ffff)，運算結果alu\_out = -4(ffff\_fffc)，運算結果正確。

1. 【Utype】 lui : 測試 lui 指令。

一張含有 螢幕擷取畫面, 文字, 軟體, 多媒體軟體 的圖片

自動產生的描述

【說明】當alu\_op = 00011，執行lui。

rs1 = ffff\_fffc，rs2 = ffff\_f000，運算結果alu\_out = ffff\_000，運算結果正確。

1. 【Itype】 lw : 測試 lw 指令。

一張含有 螢幕擷取畫面, 軟體, 多媒體軟體, 繪圖軟體 的圖片

自動產生的描述

【說明】rs1 = 0001\_10f8，等待ARVALID\_M1以及ARREADY\_S1 handshake後，透過AXI bus傳輸ARADDR(0001\_10f8)，等待RREADY\_M1以及RVALID\_S1 handshake後，透過AXI bus將RDATA(0000\_10f4)傳輸回CPU，運算結果正確。

1. 【Itype】 sw : 測試 sw 指令。

一張含有 螢幕擷取畫面, 文字, 軟體, 多媒體軟體 的圖片

自動產生的描述

【說明】pc = 0000\_087c，alu\_out計算出要儲存的地址(0001\_007c)，等待AWVALID\_M1以及AWREADY\_S1 handshake後，透過AXI bus傳輸AWADDR(0001\_007c)，等待WVALID\_M1以及WREADY\_S1 handshake後，透過AXI bus將WDATA(0000\_0f00)傳輸回CPU，同時拉起WLAST\_M1，最後BREADY\_M1以及BVALID\_S1 handshake完成，寫入DM，運算結果正確。

1. 【Jtype】 jal : 測試 jal 指令。

一張含有 軟體, 多媒體軟體, 文字, 繪圖軟體 的圖片

自動產生的描述

【說明】pc = 0000\_0bb4，執行jal指令時發現需要跳轉，將E\_jb拉起，表示下一個cycle需要CPU沖刷IF與ID中不正確的指令，完成後再將pc修正為正確的地址(0000\_0bbc)，運算結果正確。

1. 【Btype】 beq : 測試beq指令。

一張含有 軟體, 多媒體軟體, 文字, 行 的圖片

自動產生的描述

【說明】當pc = 0000\_090c，執行beq指令時發現需要跳轉，將branch拉起，表示下一個cycle需要CPU沖刷IF與ID中不正確的指令，完成後再將pc修正為正確的地址(0000\_0914)，運算結果正確。

1. AXI Read

一張含有 軟體, 多媒體軟體, 文字, 電腦圖示 的圖片

自動產生的描述

【說明】當 AR channel handshake完成，slave接收到地址後，下個cycle拉起RVALID表示可以讀取資料，等待R channel handshake 完成後，master已經讀取完成，將RLAST拉起表示完成資料讀取。

1. AXI Write

一張含有 軟體, 多媒體軟體, 文字, 電腦圖示 的圖片

自動產生的描述

【說明】當 AW channel handshake完成，slave接收到地址後，下個cycle拉起WVALID表示可以寫入資料，等待W channel handshake 完成後，master已經寫入完成，將WLAST拉起表示完成資料寫入，最後等待BVALID與BREADY完成B channel handshake後，完成一次完整的資料寫入。

1. **Synthesize**

|  |  |  |  |
| --- | --- | --- | --- |
| **Clock Period(ns)** | **Area(um2)** | **Power** | **Data Require time** |
| **10** | **5903296** | **126.7759** | **10.79ns** |

1. **Timing: 10.79ns**

一張含有 文字, 螢幕擷取畫面, 針線 的圖片

自動產生的描述

1. **Area: 5903296um2**

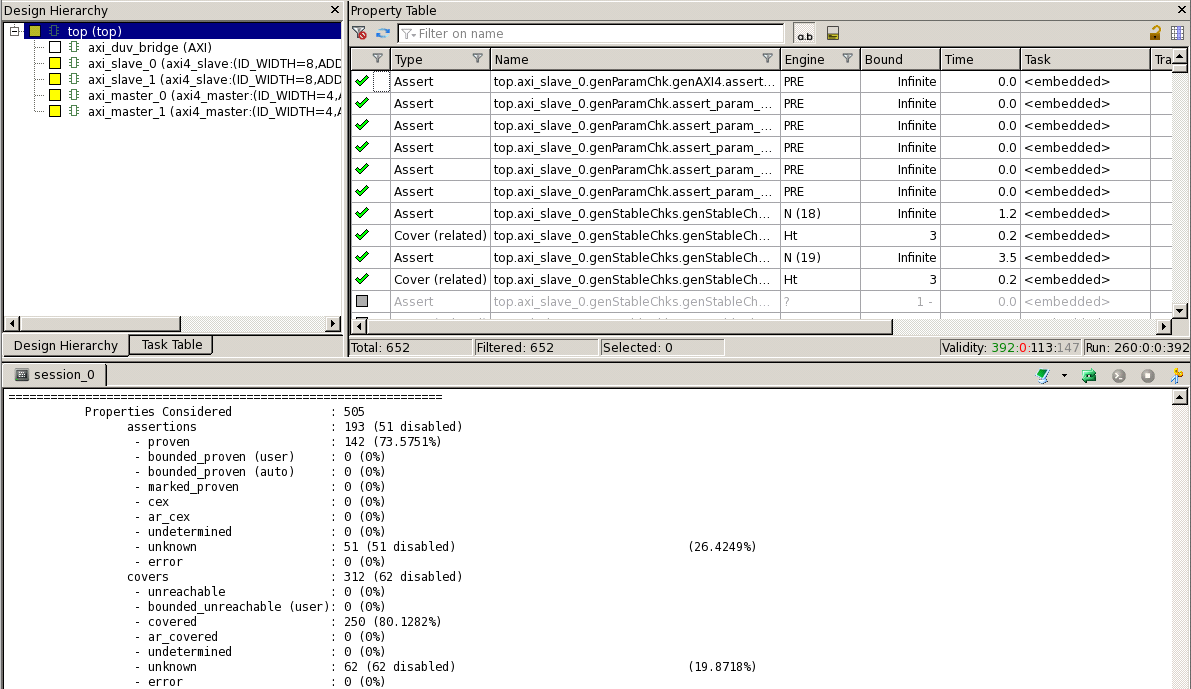
**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述**

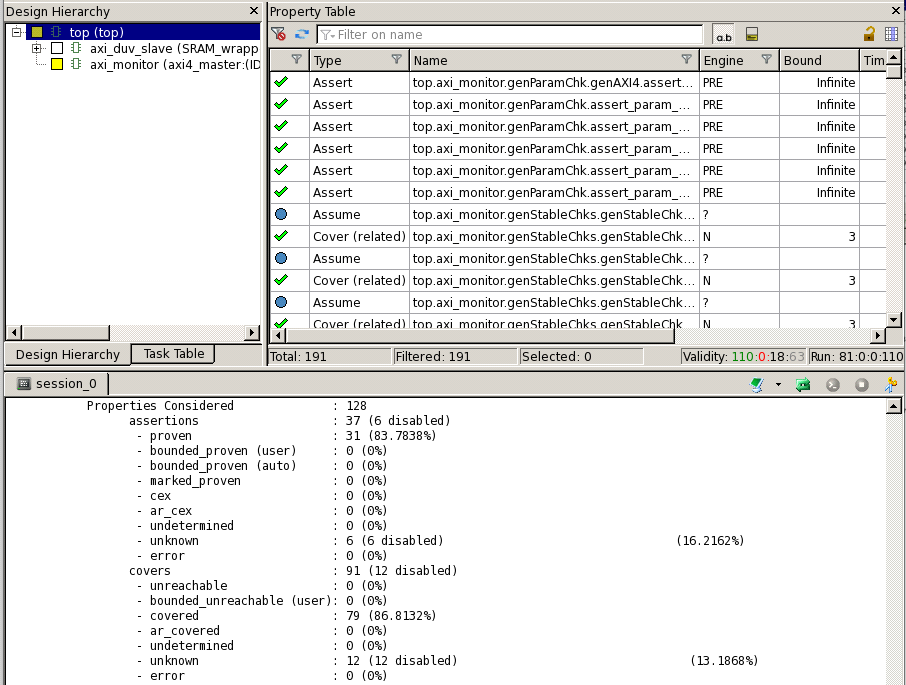
1. **Power: 126.7759mW**一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

   自動產生的描述
2. **Jaspergold**

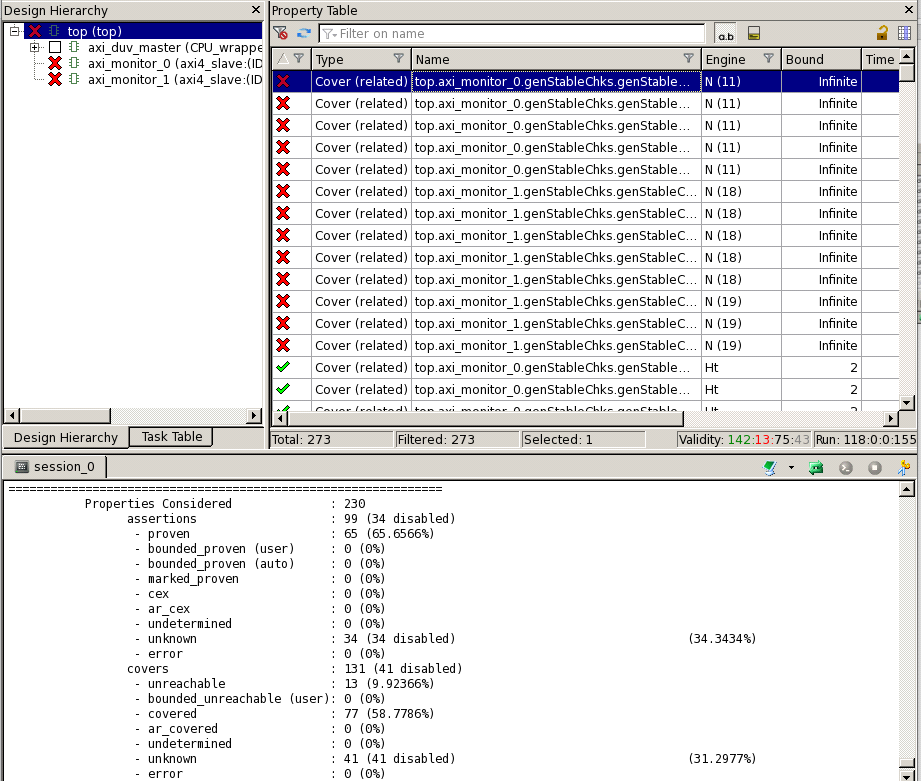
Vip\_b: all pass



Vip\_s: all pass



Vip\_m: pass

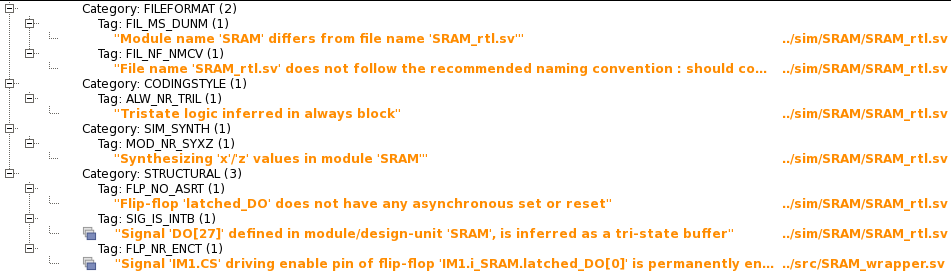


一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

因為AXI master電路設計時的特性，進入RVALID後等待RREADY，所以剩下13個unreachable的problem，對電路功能並無影響。像是rvalid && !rready中，在R state時，會直接讓rready一直拉起，故會違反rules，但能簡化電路並正常運作；bready也是同理，在B state時，會一直拉起。

1. Superlint



剩下的皆是跟SRAM檔案相關之warning。



Total lines: 2643

Code Coverage = 99.735%

解決: 增加非同步reset設計，和確認訊號長度是否相等。

1. 心得與問題解決

曾柏硯: 這次作業對我來說最大的困難點是第一次接觸AXI Bus Protocol，因此花了很多 時間在了解 AXI 以及ABVIP，而且可能發生一種糟糕的情況是master、slave以及bridge都通過ABVIP，但是接在一起跑測試程式卻可能不會動(死掉的龍貓)的情況，原因就是出在兩邊的時序溝通有問題，因此在開始寫電路之前要先和對方溝通好波型圖，而非直接照著自己覺得應該是怎麼樣就寫。這次的作業是基於第一次的CPU再加上bus protocol，雖然老師說bus說白了就是一堆線(匯流排)，但是其中要注意的細節卻意外的很多因此不是想像的那麼簡單，而且每個人的做法不一樣，造成不同人的寫法可能FSM會變得非常複雜或淺顯易懂，透過與別人交流想法之後才更能改善自己的思考邏輯與設計方向。

蔡明翰: 此次作業首先遇到的困難就是，對AXI的理解，因為過去有修過計組，所以在寫CPU時很有概念。但是AXI很難有硬體的想法，更像是一種概念，並且訊號更多，所以在前面的設計和閱讀相關資料，就花了非常多的時間，最後也才發現簡化成幾個state去理解，相對簡單很多，可能也表示自己在狀態機設計能力偏弱，之後也許要學習更多有關有限狀態機的知識。接著是將寫好的code跑JasperGold去驗證，這是非常方便的工具，雖然debug時還時花了很多時間，但至少不用再去仔細比對waveform錯在哪個訊號，透過tools直接告訴你錯在哪邊，並如何修正。最近的課程也正好教到testbench和驗證的寫法，除了設計電路和codind的能力外，驗證我認為也是成為一名好的數位IC工程師應培養的能力，並期待在下次學習更多。