VLSI System Design (Graduate Level)

Fall 2023

HOMEWORK III

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 | 吳欣航 | 呂修麟 |
| Student ID | N26124395 | N26124272 |

**Summary**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Hardware | | | | | | | | |
|  | | | | | | RTL | | synthesis |
| Top | CPU\_wrapper | | CPU | | | Pass | | pass |
| New instructions | | | Pass | | Pass |
| SRAM\_wrapper (IM & DM) | | | | | Pass | | Pass |
| ROM\_wrapper | | | | | Pass | | Pass |
| DRAM\_wrapper | | | | | Pass | | Pass |
| AXI | | | | | Pass | | Pass |
| Sensor control wrapper | | | | | Pass | | Pass |
| Watch Dog Timer | | | | | pass | | Pass |
| Synthesis result | | | | | | | | |
| Area | | | | | Clock cycle(ns) | | | |
| 6020980 | | | | | 10 | | | |
| Firmware & Software | | | | | | | | |
|  | | RTL pass | | syn pass | | | Execution time(ns) | |
| Booting | | Pass | | Pass | | | - | |
| Prog 0 | | Pass | | Pass | | | 1459430 | |
| Prog 1 | | Pass | | Pass | | | 24045230 | |
| Prog 2 | | Pass | | Pass | | | 31941840 | |
| Prog 3 | | Pass | | Pass | | | 19086630 | |
| Prog 4 | | Pass | | Pass | | | 12264220 | |
| Prog 5 | | Pass | | Pass | | | 11266720 | |
| Spyglass summary(number of inline messages) | | | | | | | | |
| Information | | Warning | | Error | | | Fatal | |
| 206 | | 3 | | 0 | | | 0 | |
| Superlint(number of inline messages) | | | | | | | | |
| Total lines | | Warning | | Error | | | coverage(%) | |
| 5311 | | 87 | | 0 | | | 98.36 | |

**Contribution**

|  |  |
| --- | --- |
| 吳欣航50% | 呂修麟 50% |
| Interrupt, CPU, CDC, debug | DRAM, CPU, CDC, debug |

**Hardware Design Description**

* System Block Diagram

本次作業為2個master和6個slave的系統，其中ROM裡面儲存boot程式，DRAM裡面存放instruction和data，要透過boot將資料從DRAM搬到IM和DM。

一張含有 螢幕擷取畫面, 文字, Rectangle, 數字 的圖片

自動產生的描述

* Interrupt mechanism description and flow chart

本次作業為了處理interrupt的機制，新增了8個指令，分別有對CSR register做讀寫的6個指令外，還有WFI和MRET，。在第一次作業就有紀錄cycle和instruction數量的register，因此在這邊擴充CSR module，在原有的基礎上新增紀錄interrupt狀態的registers如mstatus, mie, mip等等，這些暫存器不光會自動記錄目前狀態，也可以透過指令被設定。此外還設計了一個Interrupt Controller來針對目前CSR紀錄的狀態判斷是否可以進到interrupt的狀態。

**CSR**

CSR暫存器中分為可以被寫入的暫存器和只能透過當前狀態改變的暫存器，我將CSR寫在EXE stage裡。可以透過instruction被寫入的暫存器有mstatus,mie, mepc，而只能透過當前狀態改變的有mtvec, mip, instret, cycle。mstatus紀錄了整個系統的狀態，會根據intr\_ex(開始進入interrupt的訊號), intr\_end\_ex(結束interrupt的訊號)做變動，也可以被指令改變，第三位代表global interrupt-enable，第7位會將上一次的mie保持住。Mie暫存器則代表了interrupt-enableregister，第7位是給timer的，第11位則是給I/O。Mip為interrupt-enable register。在本次作業Mtvec則是0x00010000，這代表了只要有interrupt發生，pc就會跳到00010000的位置。Mepc則記錄了了interrupt前運行到的pc位置。

有6種CSR instructions，可以對可被寫入的mstatus, mie, mepc做改變，而6種不同之處為與rs1\_data做不同的運算。

**Interrupt Controller**

Interrupt Controller的部分，使用FSM控制，有NORMAL, WFI, TRAP, TIMEOUT等部分，一開始會在NORMAL狀態，若檢測到有sensor發出的full訊號，且mie和meie又為高的話則可以進到TRAP，同樣的若檢測到WDT發出的timeout訊號，且mie和mtie為高則可以進到TIMEOUT，WFI為接到wfi指令會進到的狀態。

進到WFI狀態後，會通知各級pipeline registers不要動作，並百pc固定在pc+4的地方，直到可以進到TRAP狀態。

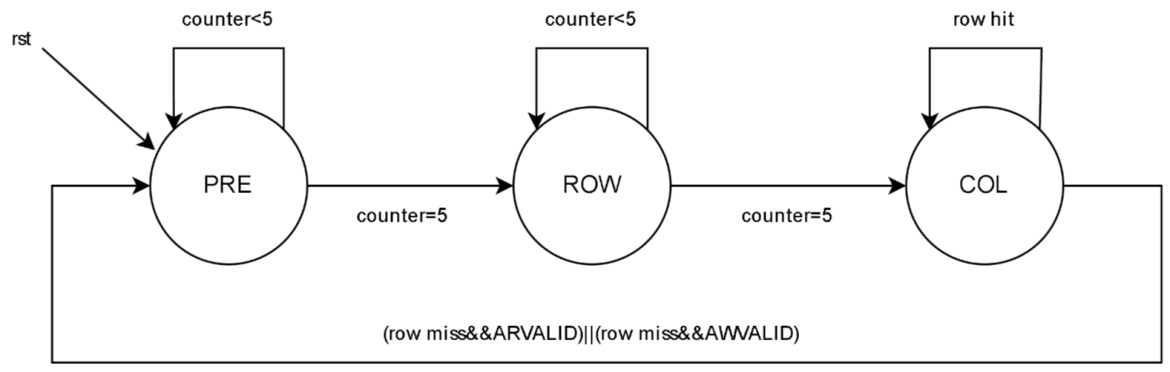
在進到TRAP時，controller會發出intr\_ex訊號，通知pc, reg\_IF\_ID, reg\_ID\_EXE flush掉目前在做的事，並通知Reg\_PC將pc換成mtvec的值。做完interrupt的事後要等執行到mret指令，controller才會發出intr\_end\_ex通知reg\_PC返回mepc紀錄的位置。

在進到TIMEOUT時，會發出intr\_t訊號，通知pc, reg\_IF\_ID, reg\_ID\_EXE flush掉目前在做的事，並通知Reg\_PC將pc換成mtvec的值，值得注意的是，我們的設計，timer interrupt的優先級大於IO裝置。

一張含有 螢幕擷取畫面, 圖表, 文字, 圓形 的圖片

自動產生的描述

* DRAM wrapper FSM chart

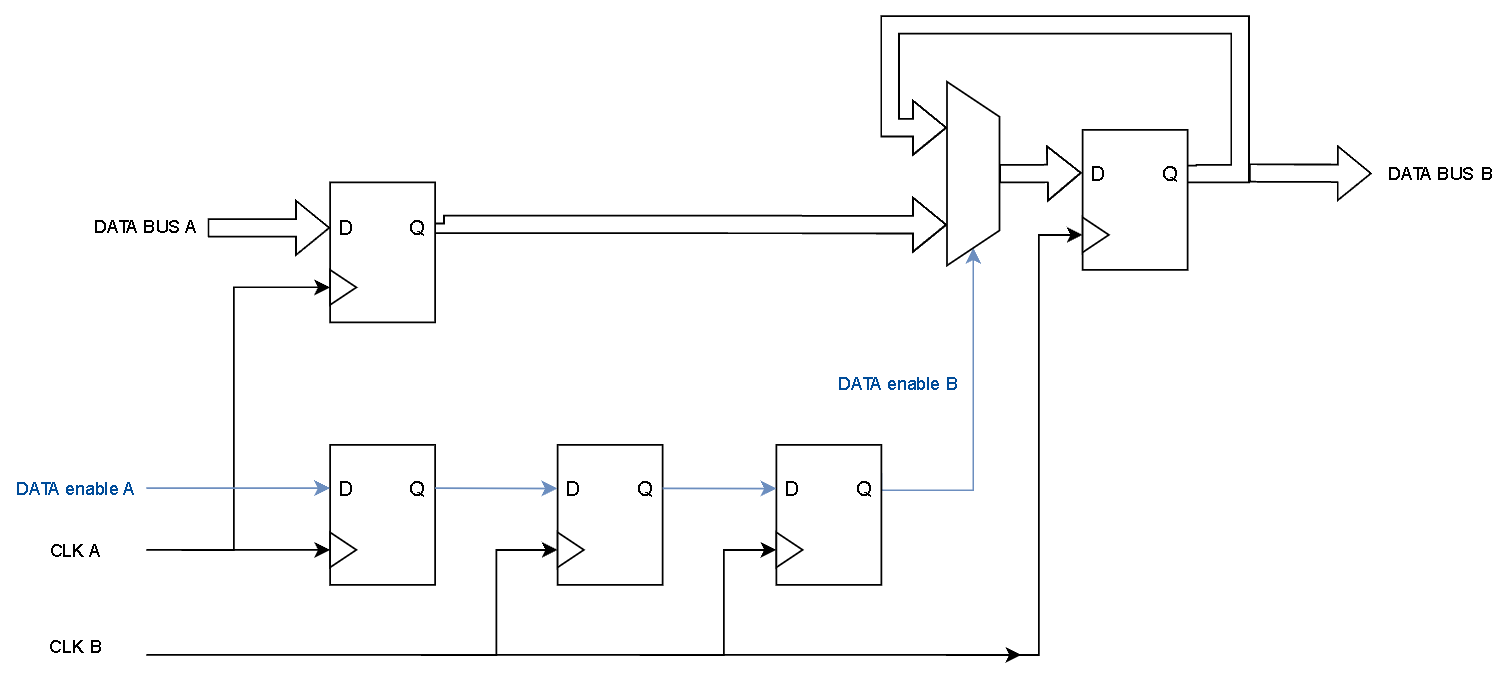


Dram wrapper的部分，使用FSM來進行控制，分為PRE、ROW、COL三個state，並且搭配一個3bits的counter進行資料存取控制，其中包含RAS、CAS和WEB的控制，否則會有violation的風險。

一開始初始化會進入PRE，進行precharge和active row後進入COL，隨時監控來自AXI的ARVALID或是AWVALID來進行next state 的邏輯控制，若發現row address 不匹配，則判定為row miss，state會回到PRE重新進行precharge和active row; 若發現row address 匹配，則判定為row hit，直接執行COL的操作。

在讀取的部分，會隨著valid訊號去控制RVALID，讓AXI另一端的MASTER能獲取正確的RDATA;在寫入的部分，則透過counter控制BRESP，能讓AXI的MASTER和BRIDGE良好的進行FSM控制。

* WDT &CDC circuit description and diagram

****

由於本次Clock domain crossing的訊號部會經常的改變，因此我們在實作上沒有選擇常見的aFIFO，而是採取較為簡易但有效的方法，如上圖為我們為了解決快到慢clk的CDC所採取的電路，所應用的訊號有WDEN、WDLIVE以及WTOCNT，慢到快clk的WTO則是使用兩個DFF 解決。

上圖這個電路的關鍵為DATA必須能夠hold住，直到能順利從posedge clk1 將資料傳遞到 posedge clk2，其中也搭配了enable 的傳遞讓clk2 能順利取到正確的值，必須注意的是，在spyglass驗測中，我們注意到了enable 為高時，data 不能改變的這個constraint，對於CDC傳遞資料是很重要的一點。

這個CDC電路和WDT wrapper的結合，只要判斷WVALID和AWADDR是否符合WDEN、WDLIVE以及WTOCNT的地址條件，即可開始進行CDC電路的傳輸動作。我們是使用counter進行控制而非FSM。

**Software & Firmware design description**

* Prog 1

在本次作業中的排序法，如同前一次作業使用bubblesort，不一樣的地方是在於short的資料型態，也就是說CPU在運算時會將32bit的資料拆掉來看，與前幾次作業有些許差異，這也使得我們在load資料的時候會出現一些問題，最後修改了load資料的mask得以解決。

* Prog 2

Prog2是要將圖片轉灰階，這一部分的lb也令我們遇到一些錯誤，最終也是靠mask的方式解決。如果目前處理的像素的三個分量都相等，則將目標區域的對應分量都設定為這個相等的值，以達到灰度化的效果。如果分量不相等，則使用一個簡單的加權平均公式0.11\*b+0.59\*g+0.3\*r計算新的灰度值，並將目標區域的三個分量都設定為這個新的灰度值。

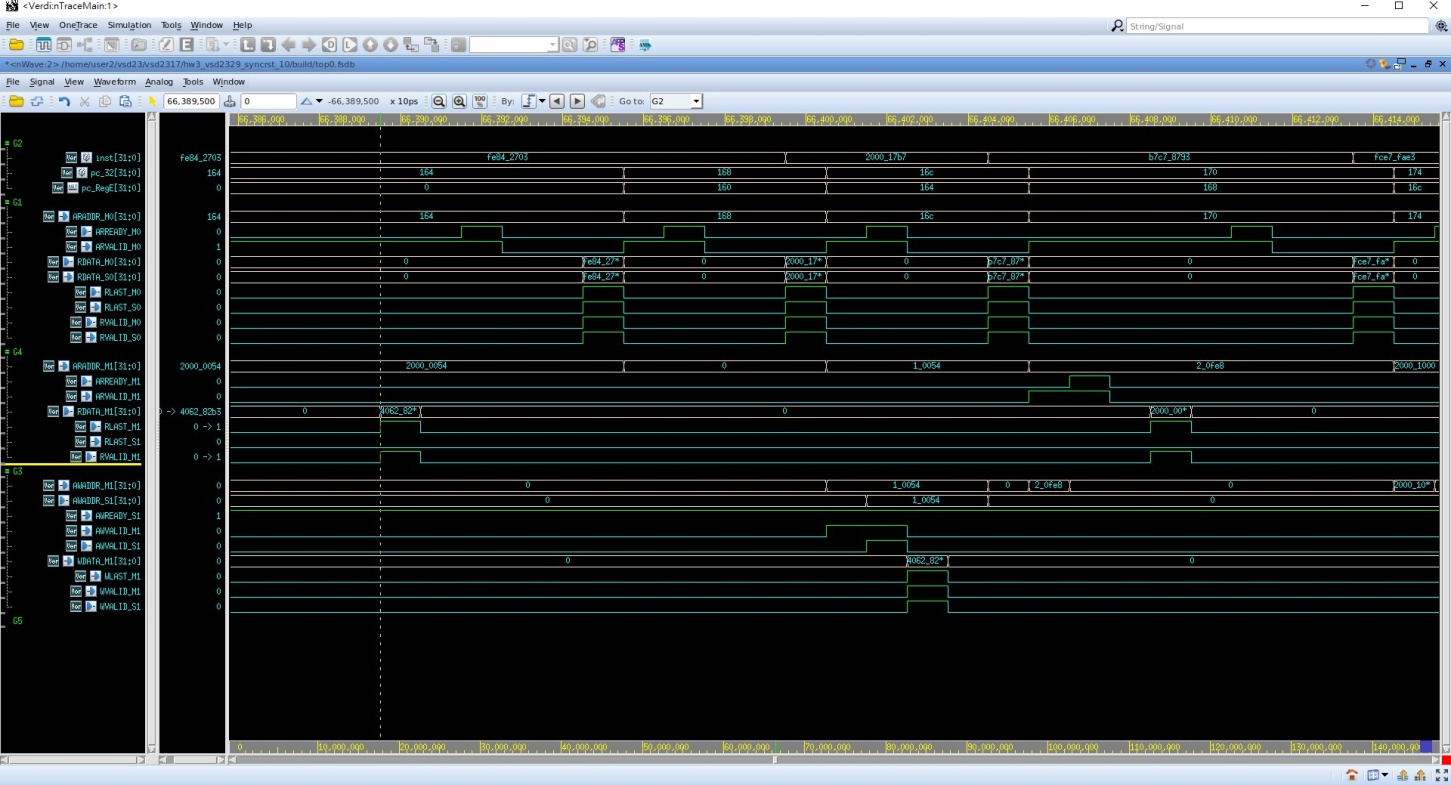
* Booting

Boot 這個程式會儲存在ROM，一開始CPU會讀取ROM的中的Boot程式將Instruction、sdata、data等資料從DRAM搬到IM 和 DM。這個程式我們使用 3 個 While Loop 搭配\_dram\_i\_end、\_\_sdata\_end和\_\_data\_end來完成Instruction、sdata、data等資料的搬運。

**Screen shot of wave forms and simulation results**

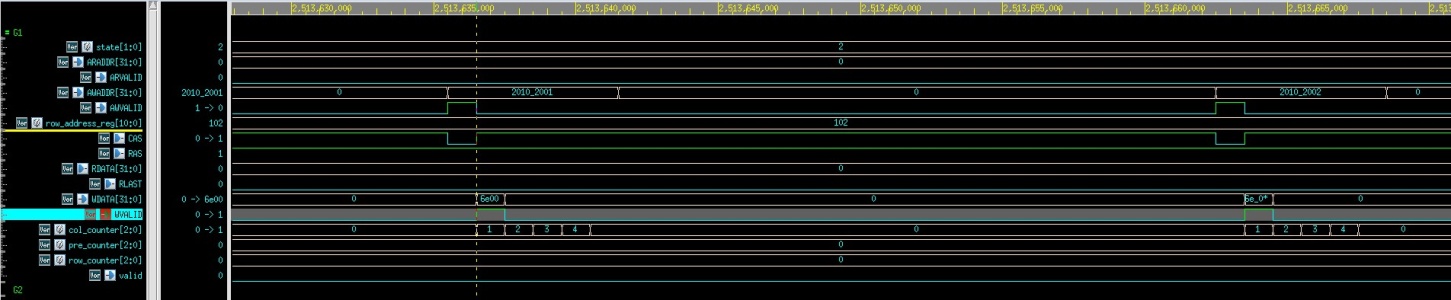
**Waveform**

**Booting**



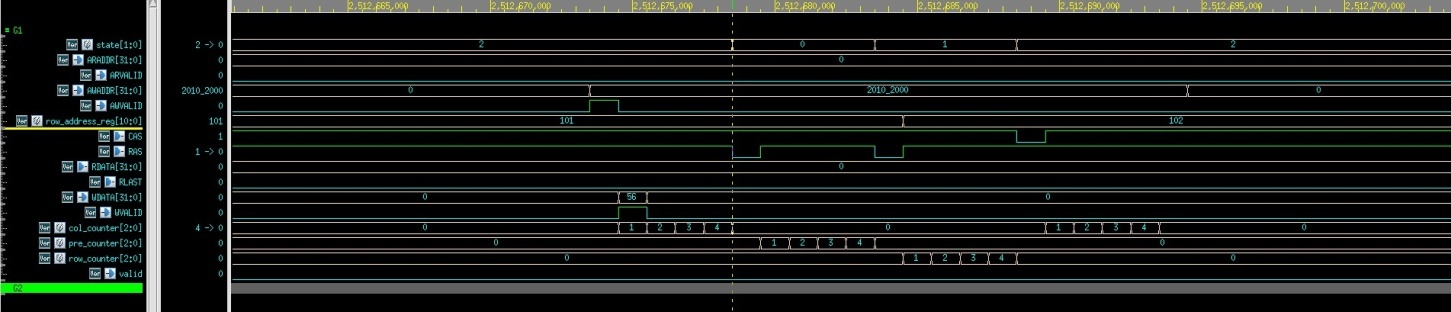
以從DRAM到IM為例。由波型可以看到第一個紅色框框的RDATA\_M1，是從DRAM的20000054讀取，隨著instruction的進行，在第二個紅色框框被寫入IM的10064。

**DRAM row hit**



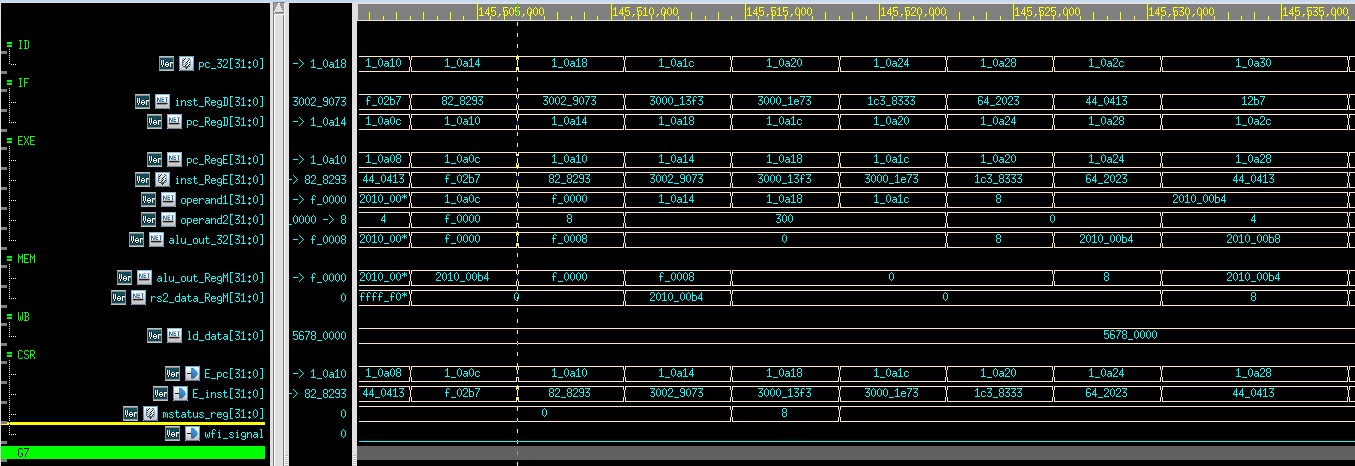
以寫入為例，若是發現row address 和上一次使用的相同，則只需要5個cycle即可完成操作，由上圖可以發現CAS和counter隨著AWVALID成功運作，會將6e00這個資料寫入到DRAM。

**DRAM row not hit**



以寫入為例，若是發現row address 和上一次使用的不同，則需要15個cycle才可完成操作，必須進行precharge、row active和 column active，由上圖可以發現RAS、CAS和counter隨著AWVALID成功運作，會將56這個資料寫入到DRAM。

**CSRRW**



以CSRRW來作範例，可以看到10a40的指令為CSRRW，要將0xf0008寫入mstatus，但由於mstatus只有3、7、12:11可以被寫入，因此可以看到紅框內的mstatus值正確地被改變，0x0008。

**Wfi**

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

由波型可以看到EXE stage看到wfi指令後通知Pc和各級registers做stall的動作，並將pc保持在pc+4的位置。

**Mret**

一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

Mret可以讓CPU回到被中斷的program，透過mepc紀錄的pc值，當發現MRET指令後，會通知reg\_PC將PC換成mepc的值，回到原本的program，如圖所示。

**Interrupt\_sensor**

一張含有 電子產品, 螢幕擷取畫面, 電路, 電腦 的圖片

自動產生的描述

此為prog3的波型，可以看到中間會有一些stall住的訊號，這邊就是在wfi state，還在等interrupt的發生，可以看到會發生8次interrupt，interrupt會從SENSOR搬運資料到DRAM，搬完8次之後就會進行排序，最後輸出。

一張含有 螢幕擷取畫面, 文字, 電子產品 的圖片

自動產生的描述

這張圖則為intr\_ex訊號的產生，也就是說是interrupt controller進到TRAP state的時候，會發出intr\_ex通知PC要把PC換成mtvec的值，在這次作業是0x0001\_0000。

一張含有 文字, 螢幕擷取畫面, 電子產品, 電腦 的圖片

自動產生的描述

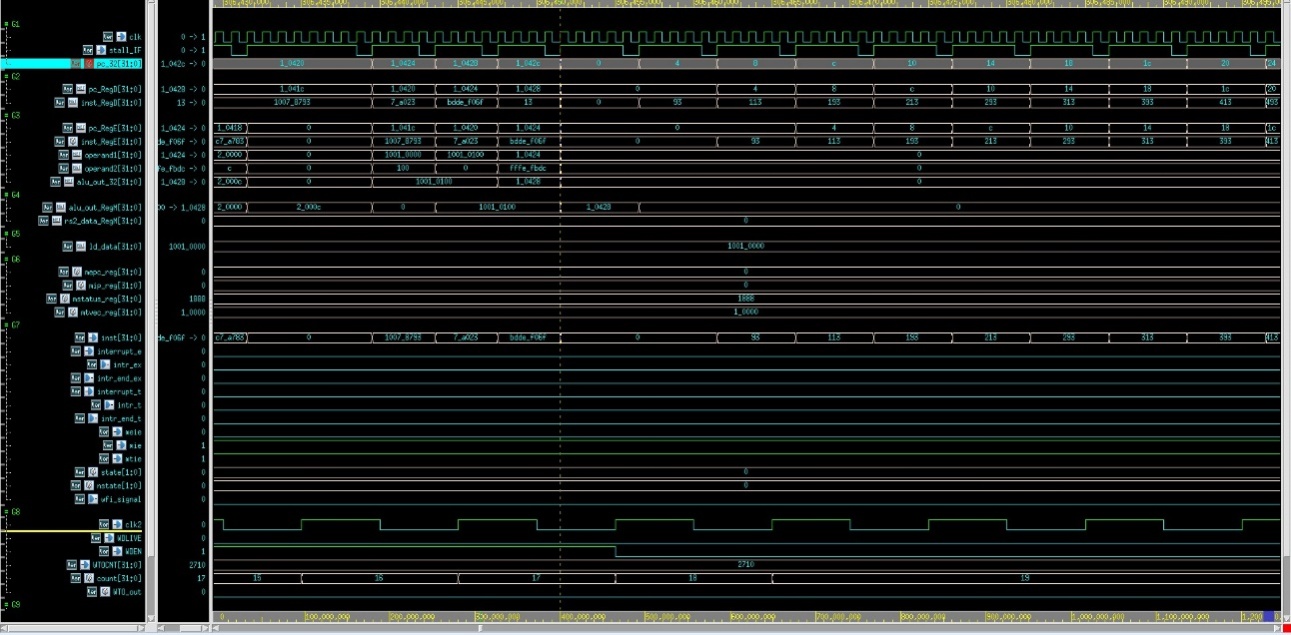
等到資料搬運完後，interrupt controller會收到mret的指令，並發出intr\_end\_ex的訊號，告知reg\_PC將現有的pc替換成mepc裡面的值。

**Interrupt\_timer**

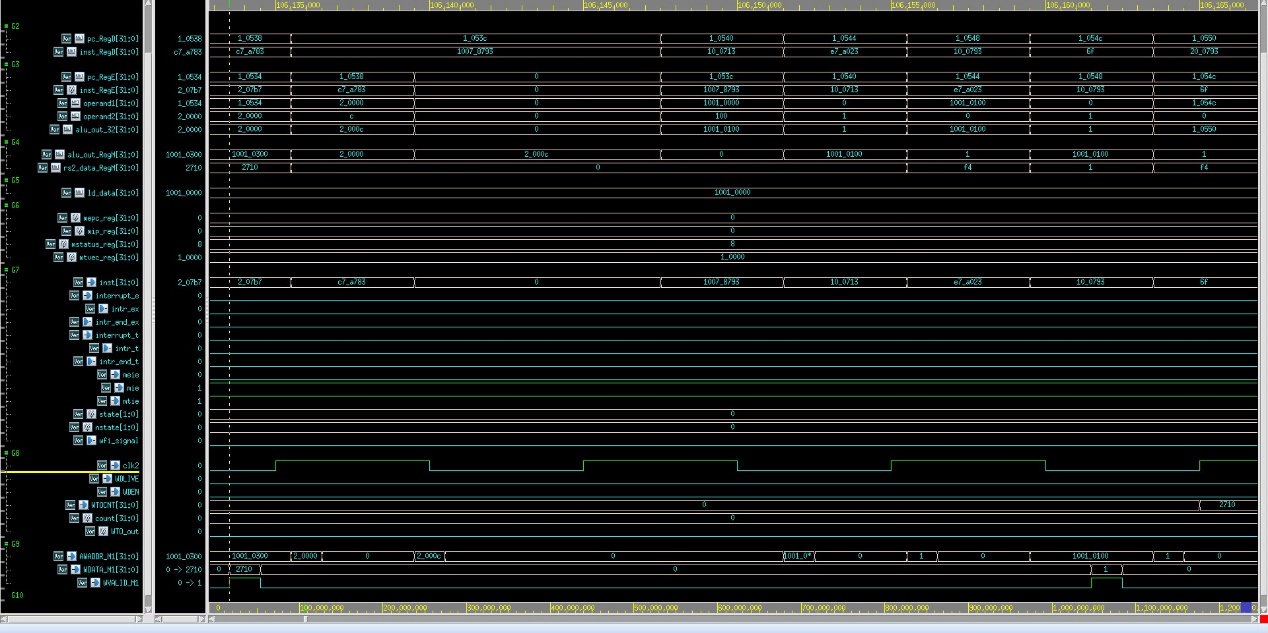
一張含有 文字, 螢幕擷取畫面, 行 的圖片

自動產生的描述

這一張圖可以看到當intr\_t拉起來後，pc一樣會到mtvec的位置。



Timer interrupt與external interrupt比較不一樣的地方是，timer interrupt不會像external interrupt再透過mret被切換回來，而是進行重置的工作，原以為重置是要靠rst訊號做處理，結果是透過指令進行重置，如圖所示。



這張圖顯示了WTOCNT成功的從CPU被寫入WDT Wrapper裡面。

**Simulation Results**

Prog0

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

Prog1

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

Prog2

一張含有 文字, 螢幕擷取畫面, 陳列, 軟體 的圖片

自動產生的描述

Prog3

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

Prog4

一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

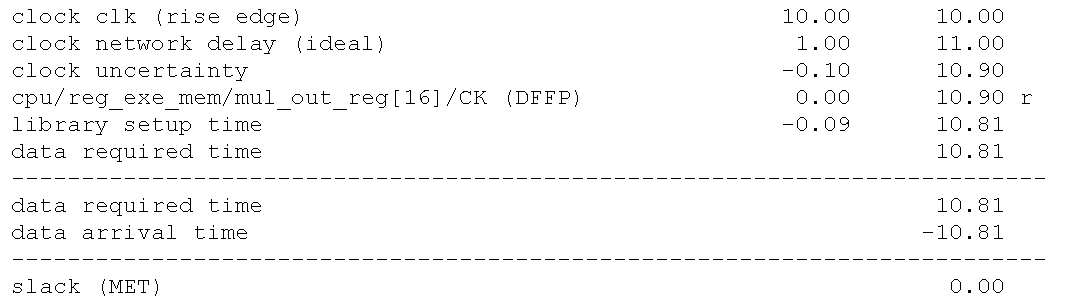
自動產生的描述

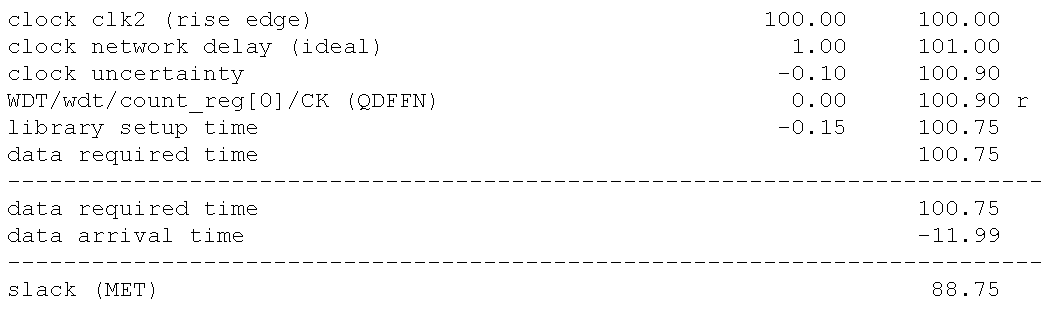
Prog5

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

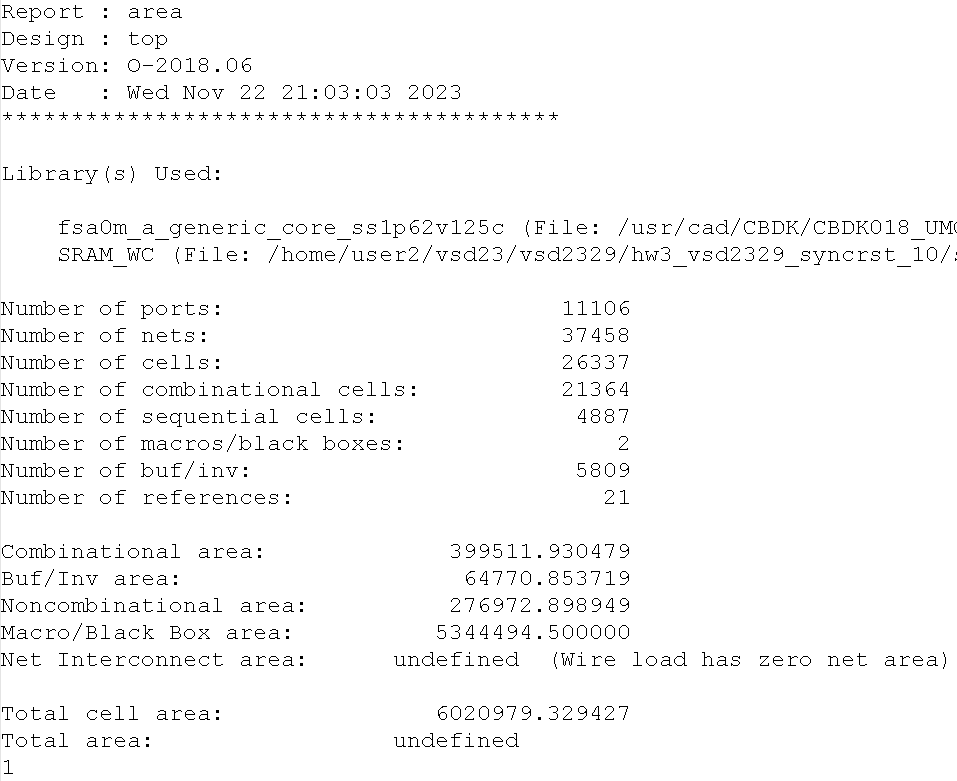
**Timing Report**





SLACK不為負數，沒有violation。

**Area Report**



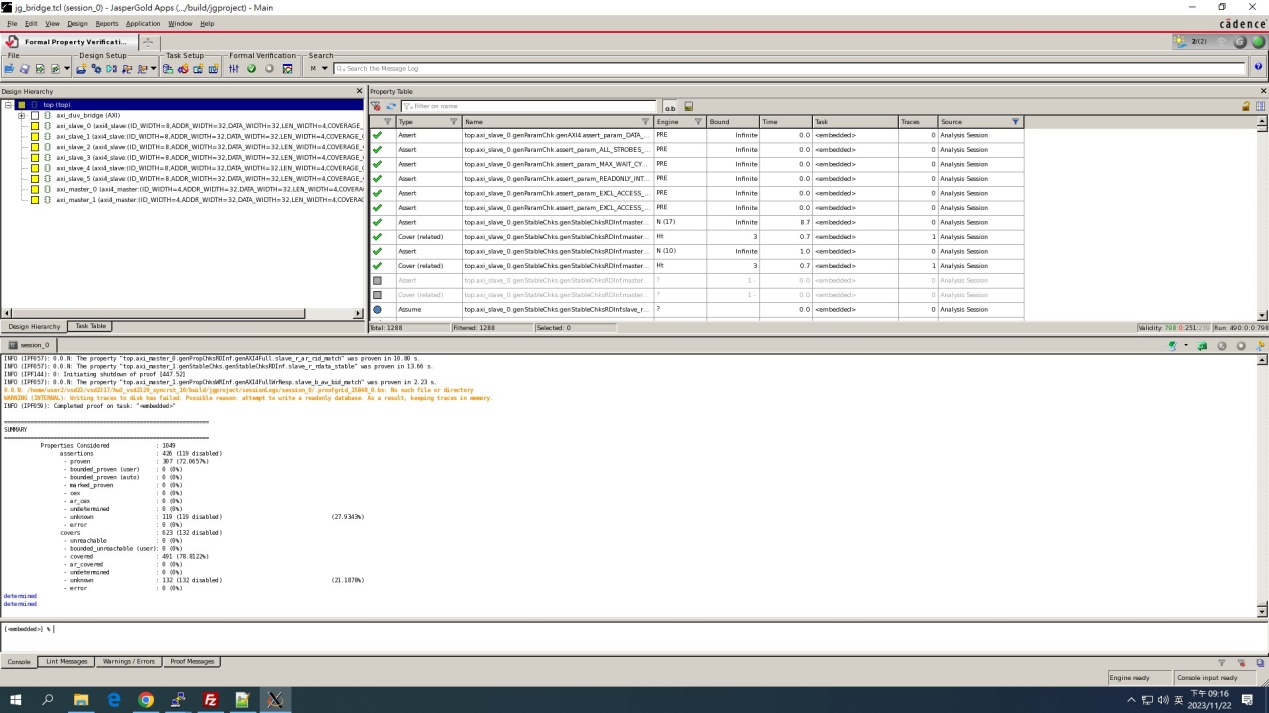
面積使用了 6020979.32。

**Superlint**



Superlint只有87行，我們的程式有5000行以上，因此(5311-87)/5311 = 98.36% 符合標準。可以看到在structural有82則warning，這是因為我們為了符合spyglass的規範，將系統的reset從asynchronous reset改成synchronous reset，所以他跳出了82則warning。其餘warning多為一些微不足道的小事。

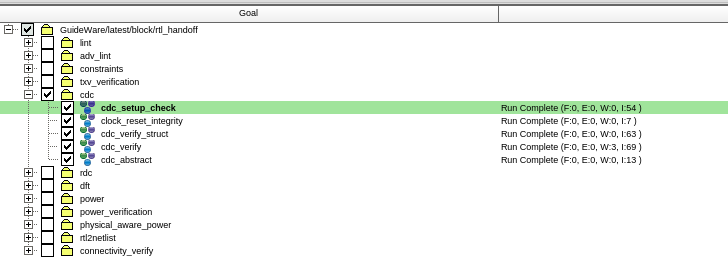
**VIP**

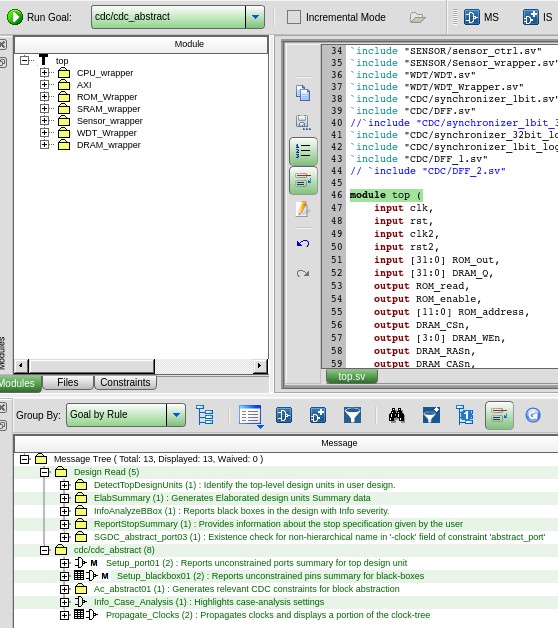
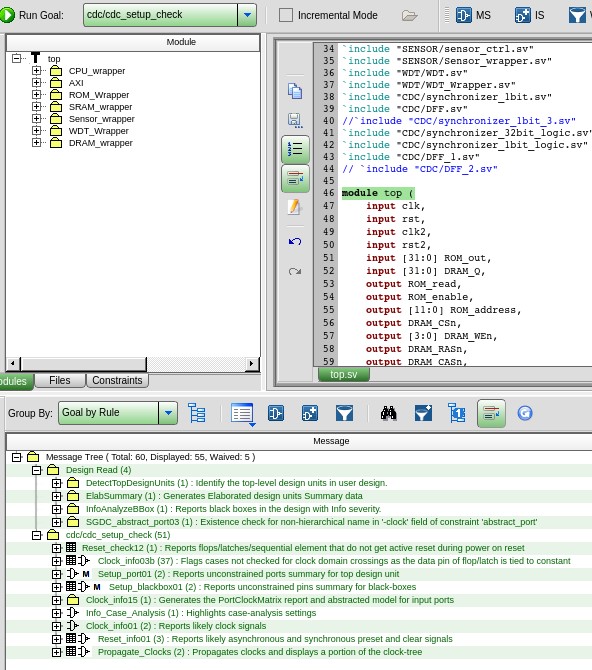
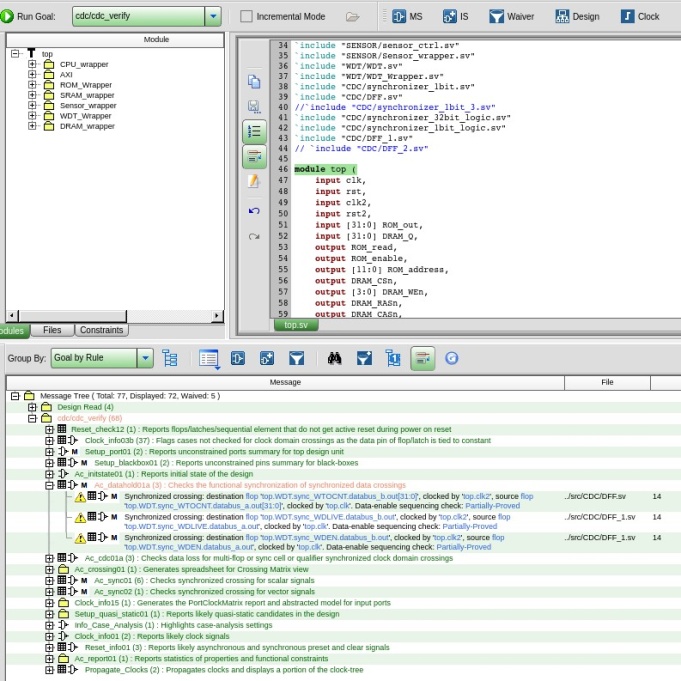
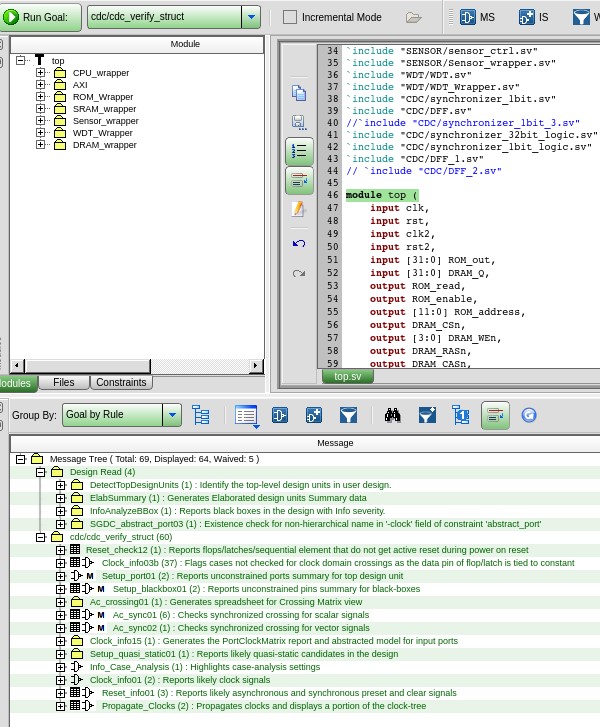


AXI的VIP全數通過。

**Spyglass**

重要的Violation中沒有fatal和error，全部總計只有三個warning，個別是三個partially proved，我們嘗試調整fa\_atime去讓他檢測，但調到時間很長皆無法驗證我們的電路是錯的，因此我們判定這個violation會造成問題的機率極小。



****

**Problems to answer**

1. What is the deference between mcycle and timer? When is mcycle used?

在 RISC-V 架構中，mcycle和 timer 是兩個用於性能監控的不同指標。mcycle代表處理器執行的時鐘周期數，即機器週期數，常被用作測量program或特定代碼段執行所需的總週期數的簡單指標，提供了處理器執行的總週期數，可用於性能分析和評估。另一方面，timer 是一般性的計時裝置如這一次作業的WDT，通常為外部設置的時鐘計算，主要的用途是為了與內部CPU做區隔，如同這一次WDT，若使用mcycle作為判斷是否超時的依據的話，CSR如果壞掉了，那他就不會被檢測到，少了第三方公正的感覺。而mcycle主要的功能大部分都是在評估效能或是計算功耗。

1. What is “Potential Qualifier” in Spyglass?

Signal that fails to synchronize the source signal of data crossing due to

the presence of invalid logic at the point of convergence with source, or the

signal is not synchronized itself，以上是spyglass在reference文件所提到的關於Potential qualifiers的描述。總而言之，Potential qualifiers是我們在發現錯誤時，spyglass根據訊號的clk domain、source和domain去標記或識別潛在的有價值或重要訊號，幫助我們進行debug。

**Lesson learned**

**吳欣航**: 在這一次作業，完成了一個系統的設計，一開始面對這個作業可以說是毫無頭緒，Dram的搬運原理、boot的組合語言、interrupt的處理、CDC的問題，都是第一次實作出來，可以說是讓我把計算機組織和作業系統學的東西又複習了一遍，也讓我對電腦系統更熟悉了一點。我覺得這一次作業比較困難的地方在於由於整個系統比較龐大，因此我們會不知道要存哪裡下手，又或是說遇到bug的時候該從哪裡開始解決，這些問題困擾了好久，我想原因是因為我們對這些東西的機制還不太熟造成的。在一開始將每個components兜起來花了最久的時間，成功解決prog0 1 2後還有interrupt的問題和CDC的問題沒有被解決，interrupt的問題在熟讀網路的教材後就有了靈感。而CDC我們這次採用了比較簡單的方法解決，因為我們認為資料沒有要一直讀寫用aFIFO來寫繪有點大材小用。總之這一次作業我真的學習到很多。

**呂修麟**: 在這次作業中，不同以往所修過的數位IC課程，增加了DRAM和boot程式去從DRAM搬運指令和資料到IM和DM，而不是在testbench就完成這部分的工作，我們光是弄懂這一部份就花了好幾天的時間。後續dram 搭配AXI的控制，我們也花費了一段時間反覆閱讀文件才正確且沒有violation地去控制DRAM的存取。除此之外，我們也在prog1的half word存取中發現了以前CPU存在的小錯誤，這讓我們花了一段時間才找到。幸好最後CDC 和WDT的部分都比較順利，也感謝吳欣航組員將CSR這部分完美處理掉。總而言之，這次作業讓我們學習到了在整合不同module的系統時，須注意到的大大小小的事情，要看的波型也非常多，希望HW4順利。