VLSI System Design (Graduate Level)

Fall 2020

HOMEWORK IV

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: 高士鈞 黃建智 陳韋呈 黑崇瑜 邱建智

Student ID: N26094922 N26080020 N26084723 N26090114 N26081440

1. **Summary**

* 完成CSR指令以及CSR module。
* 修改CPU, 加入interrupt模塊。
* 加入ROM，DRAM，sensor module。
* 結合上述及前次作業之code，通過模擬prog0,1,，以及合成後模擬syn0,1,。
* 完成CPU在APR上之佈線。

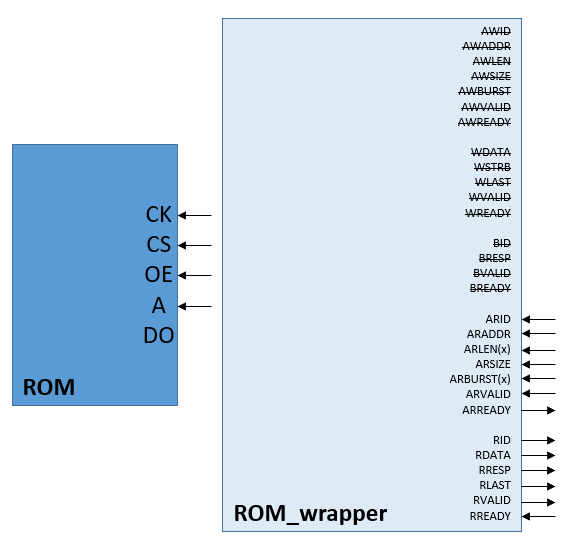
1. **Check list**

|  |  |  |  |
| --- | --- | --- | --- |
| RTL code | | synthesis | |
| prog0 | v | prog0 | v |
| prog1 | v | prog1 | v |
| prog2 | v | prog2 | v |
| superlint | 97.57% | | |
| APR | **v** | | |

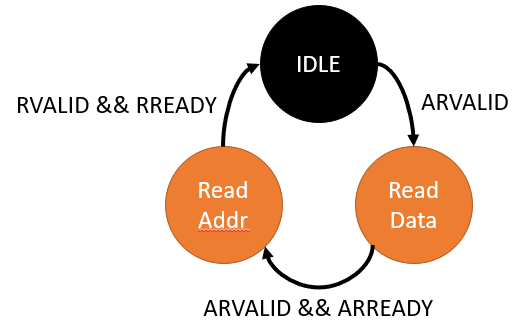
1. **Problem1**

Booting + instructions verification.

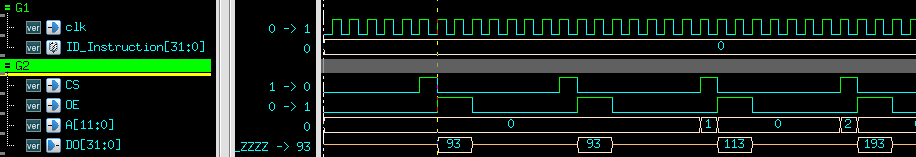
在PROG0中，我們需要實作出ROM wrapper、DRAM wrapper以完成程式，同時CSR要在Instruction中加入對CSR指令的支援，以下各自說明。



**ROM**即是read onlu memory，他用來儲存基本的開機程式，所以他只有read的功能，行為相對簡單，可以把他視為是簡化版的SRAM。

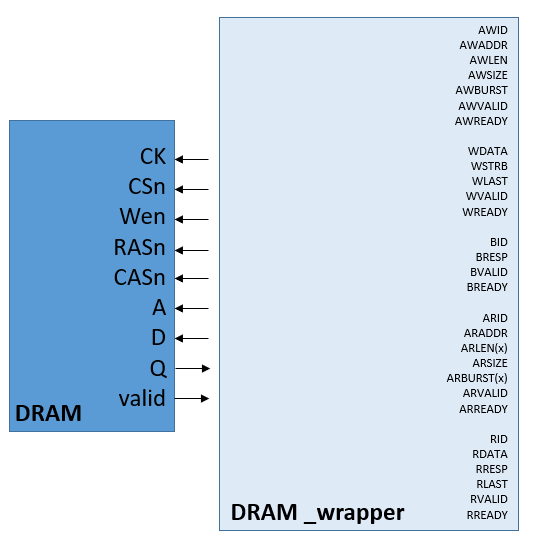
以FSM來看，他只需要三個state，IDLE，發addr的ReadAddr，即收data的ReadData。

以波形來看，送入addr，拿到資料，一氣呵成。

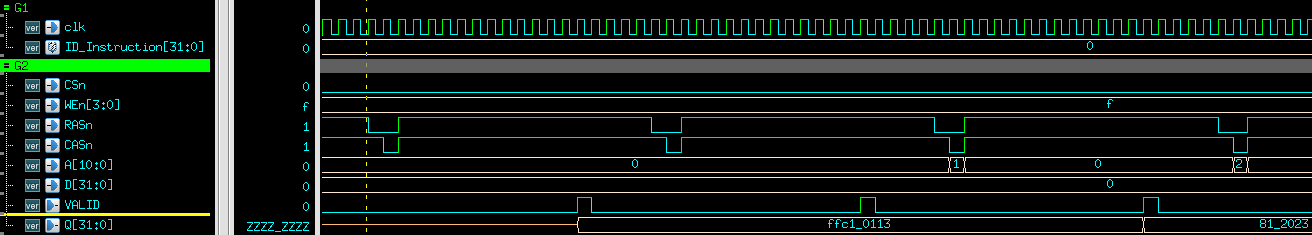
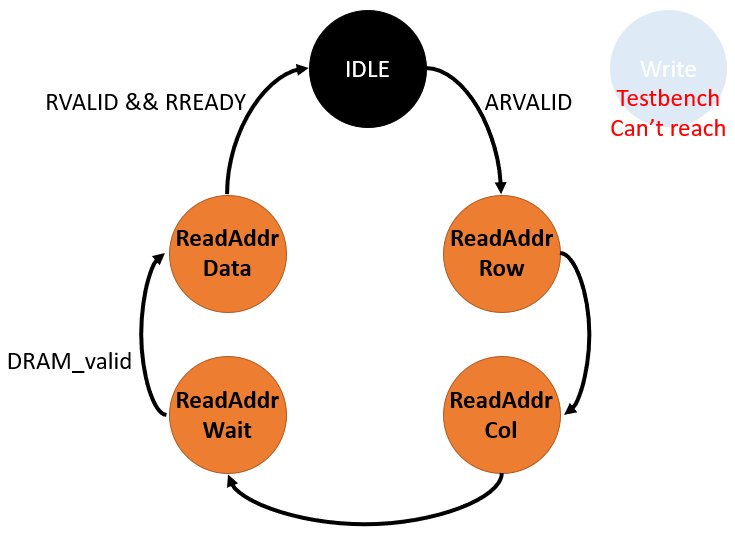


寫地址

拿資料



**DRAM**的行為就複雜許多了，因為DRAM本身的地址特性，在地址寫入時分為row和column，所以在設計FSM時就需要多加幾個狀態，另外DRAM在讀取資料時並非及時兔出訊號，而是透過valid訊號提示資料狀態，所以要有判斷valid的state以免我們拿到尚未讀取完成的訊號。

順帶一提，此次做的的兩個testbench其實都驗不到DRAM的write功能，所以不贅述write。

第三筆資料讀出

**81\_2023**

寫地址

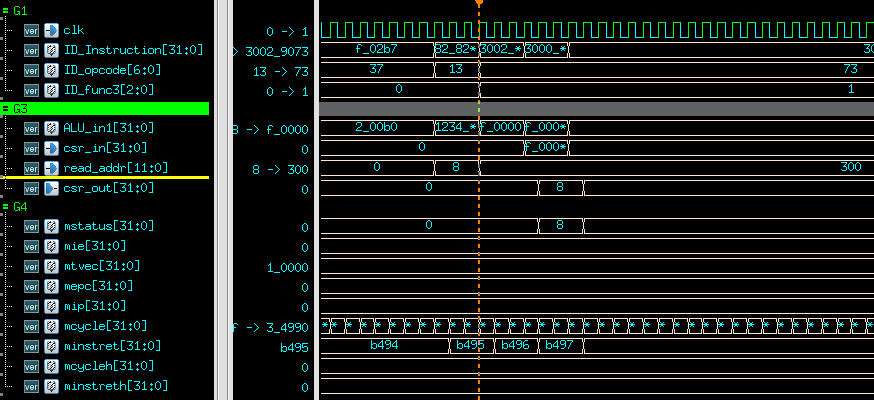
第一筆資料讀出

**ffc1\_0113**

寫地址

-----------------------------------------------------------------------

**CSR module**是用來記錄細整個電路的運作狀態的，其中有八個register記載了不同資訊，例如mcycle記錄整電路運行多少cycle，CSR指令除了WFI和MRET以外都大同小異，所以在此只列出其中一個的波形。

此為CSRRW指令，所以寫入值等於register讀出來的值，而其讀出來的值根據指令定義的讀取位置為’h300，讀出mstatus的值8。

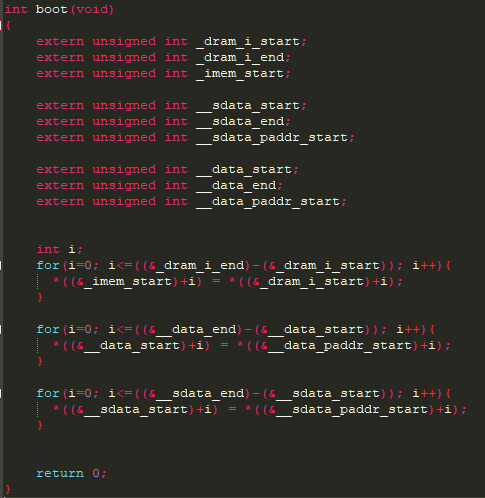
**read mstatus**

**so output is 8**

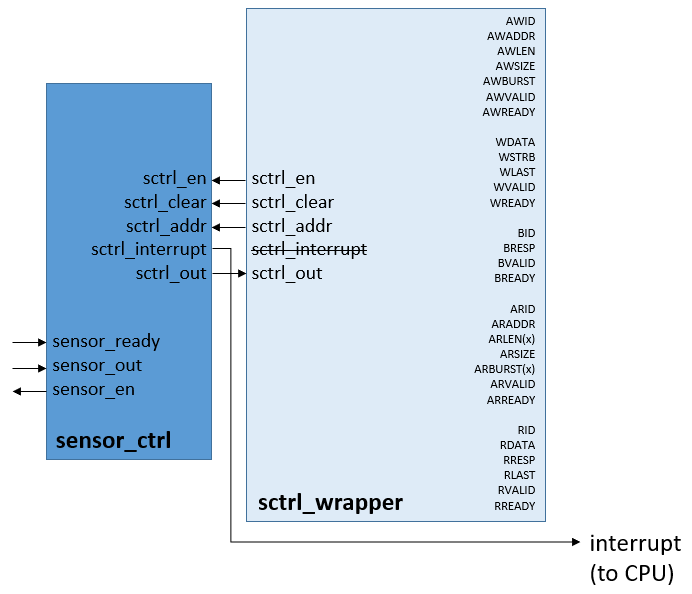
**read CSR reg’s address**

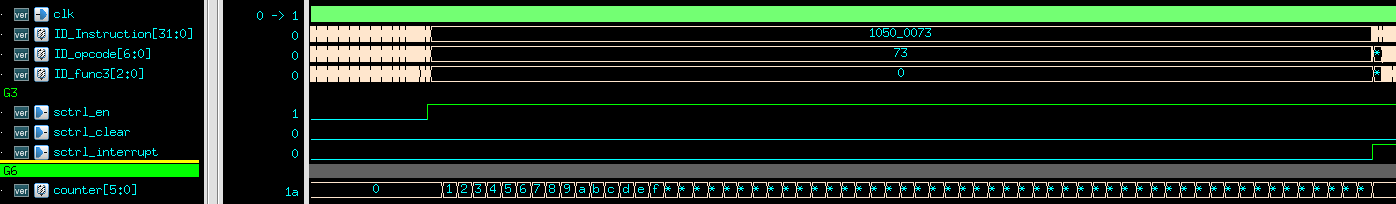
**CSR input = rs1**

**CSRRW**

**boot.c**的部分，其實助教定義的很清楚了，所以沒有花太多時間，主要就是目標的寫入位置和被寫入位置要搞清楚，舉例來說，\_dram\_i\_start要被寫入\_imem\_start的值，而寫入幾個值就看\_dram\_i\_end-\_dram\_i\_start的長度決定，其他兩個區塊也以此類推即可!

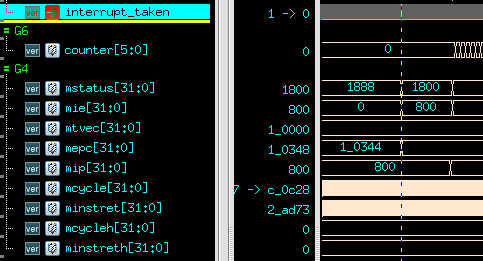
1. **Problem2**

booting + interrupt verification

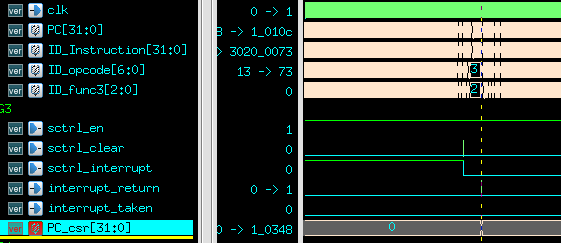
sensor\_ctrl的執行很特別，需要對某個位置寫入，事實上，sensor\_ctrl只有兩個寫入位置，對0x1000\_0100些入非0值會觸發sctrl\_en，對0x1000\_0200些入非0值會觸發sctrl\_clear，透過波形可以看到sensor\_ctrl一直到觸發sctrl\_en才開始累積資料，並且在滿64筆資料後觸發interrupt，可以看到因為剛好執行到WFI指令，所以這這段時間內instruction並沒有改變，直到收到interrupt。

**sctrl enable -> start count**

**WFI**

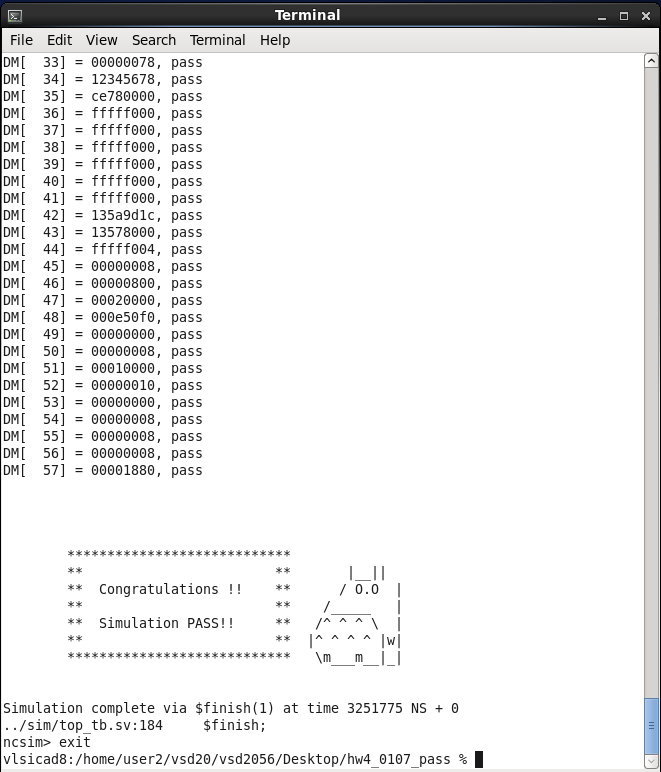
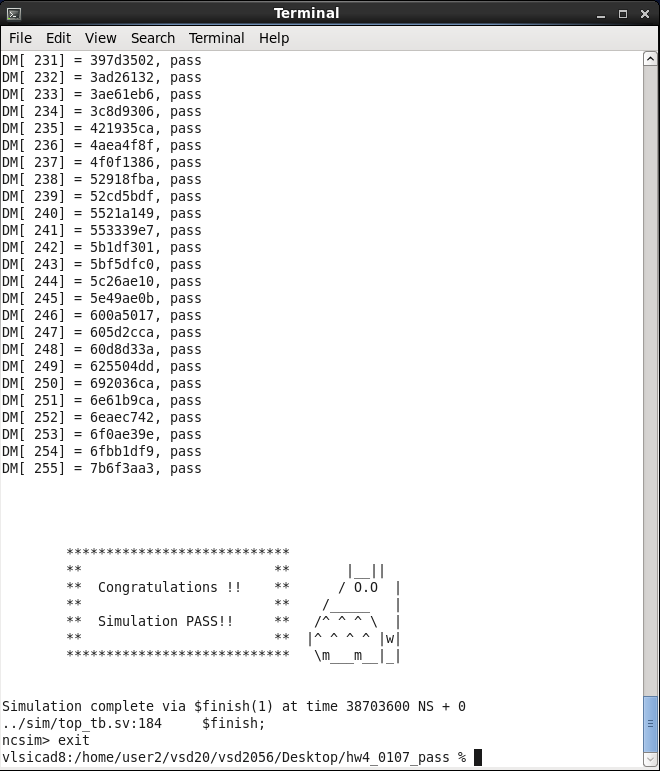
在觸發interrupt後，interrupt訊號會和WFI指令共同構成interrupt\_taken，interrupt\_taken訊號會讓CSR內部的值有大量變化發生，例如mepc將被寫入當前的pc值。

**interrupt\_taken change value in CSR**

除了WFI另一個和interrupt有關的指令是MRET，這個指令會和interrupt訊號共同構成interrupt\_return，他最大的特色就是會讓當前的pc換成mepc內存有的pc。

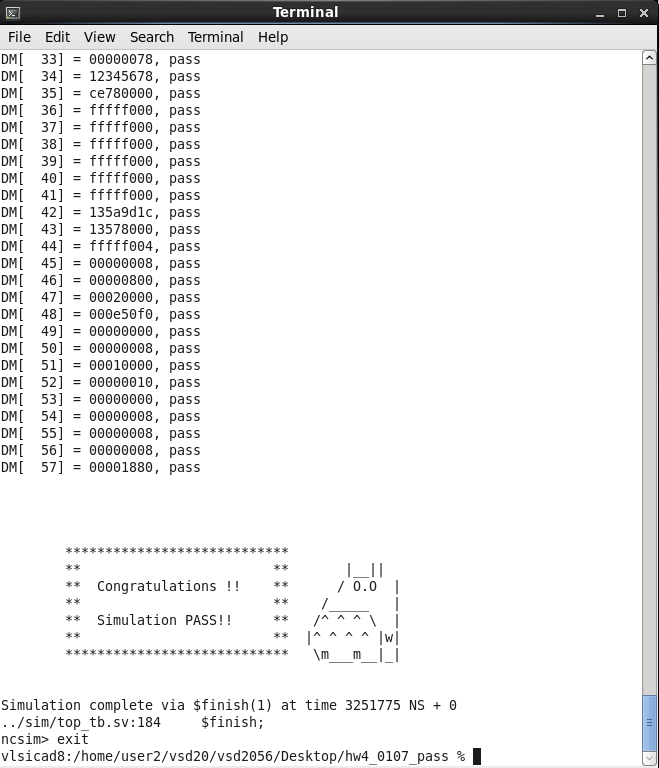
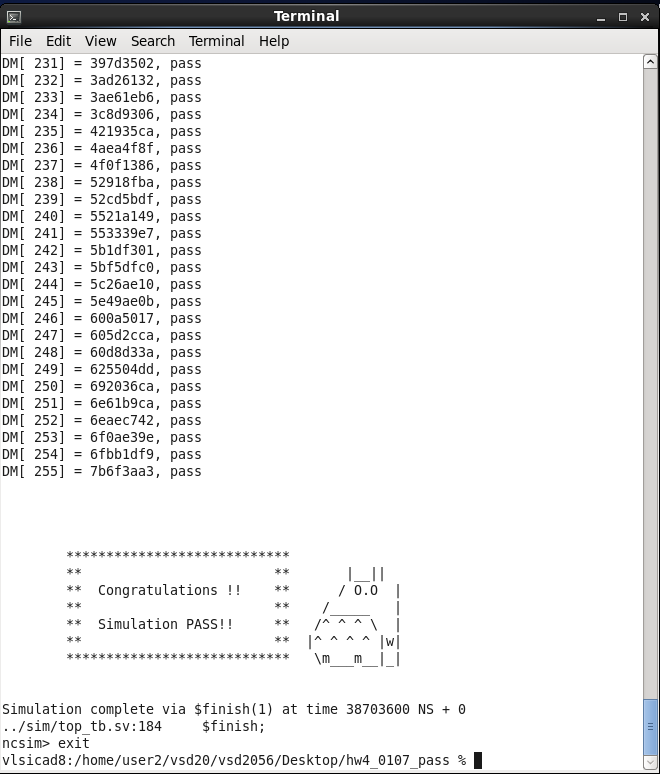
**Change PC**

**MRET**

1. **Testbench**

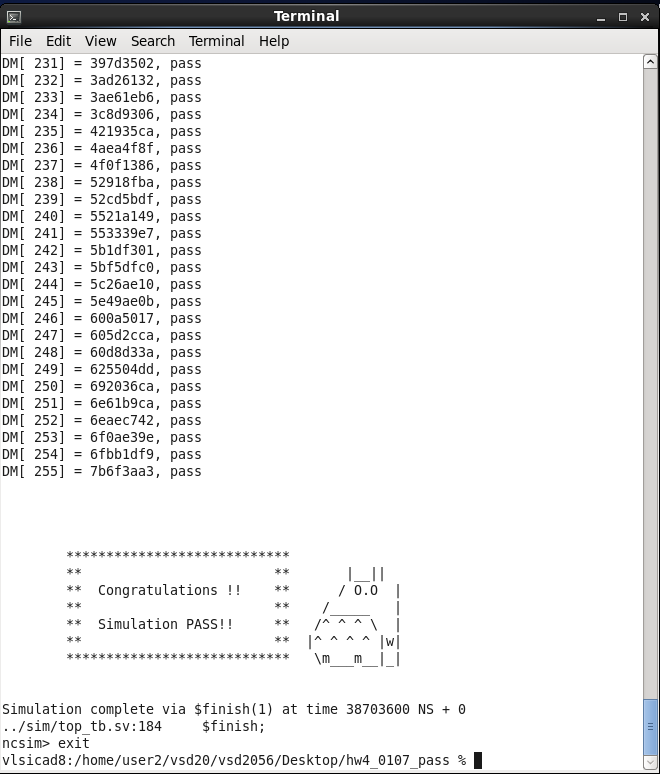
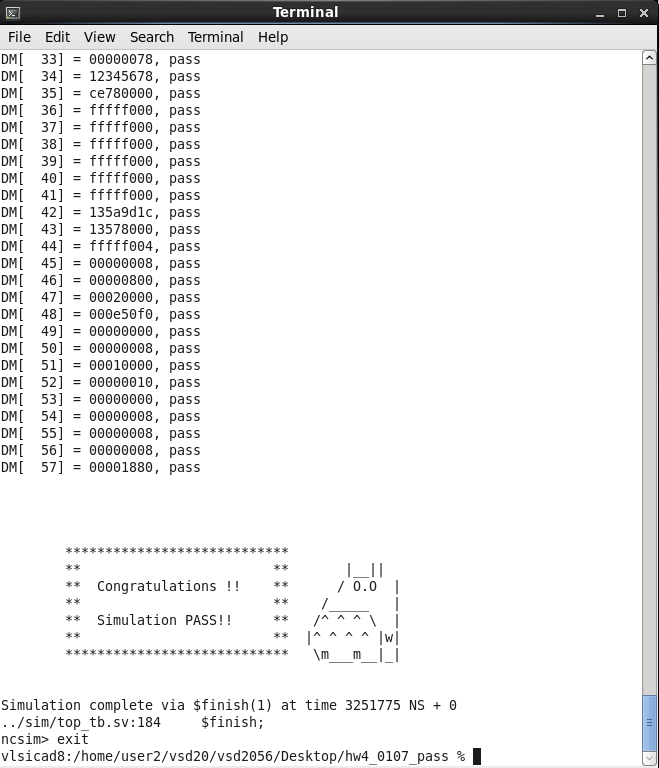
**RTL1**

**RTL0**

****

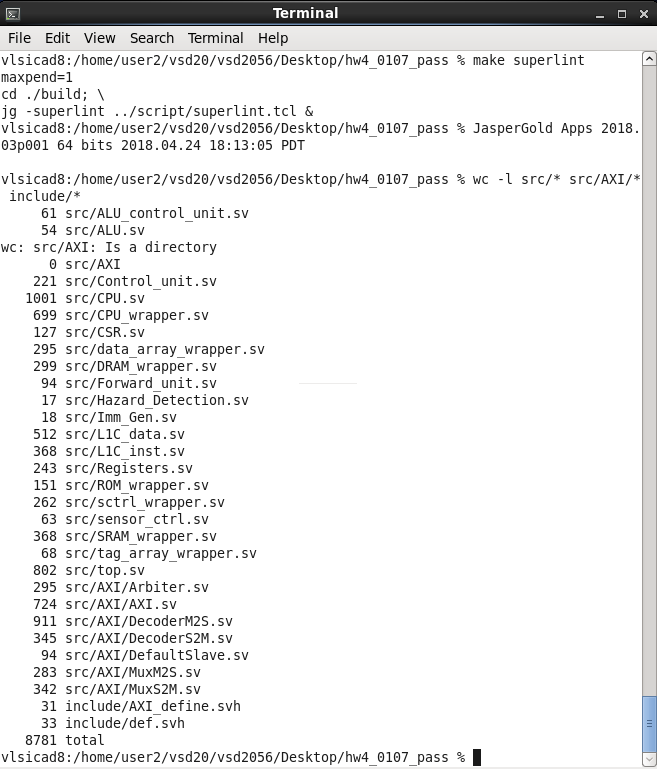
**SYN0**

**SYN1**

****

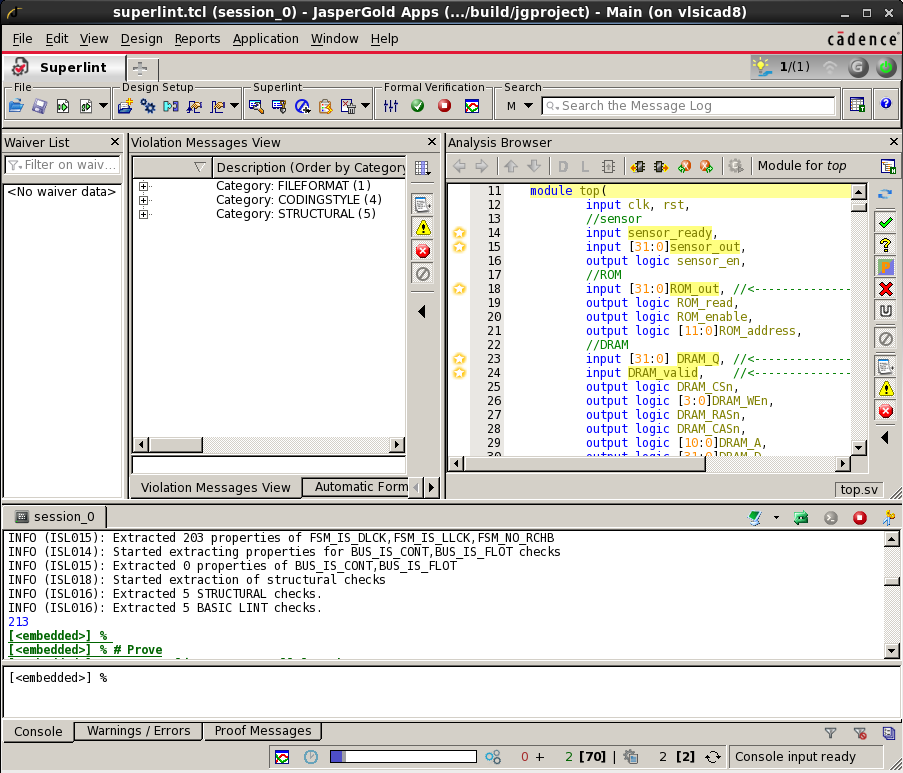
**PR0**

**PR1**

1. **Superlint**

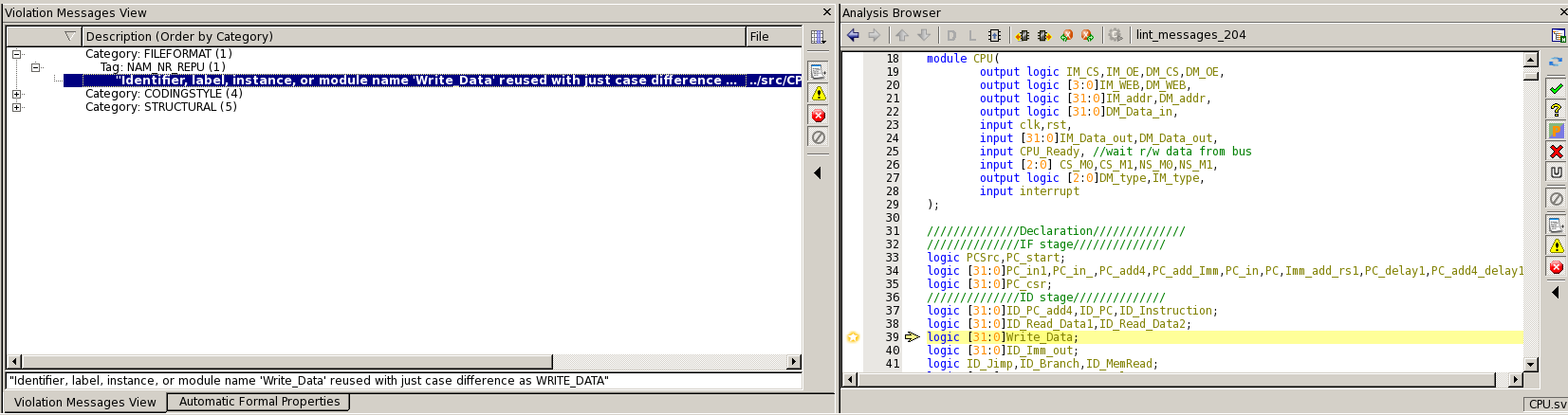
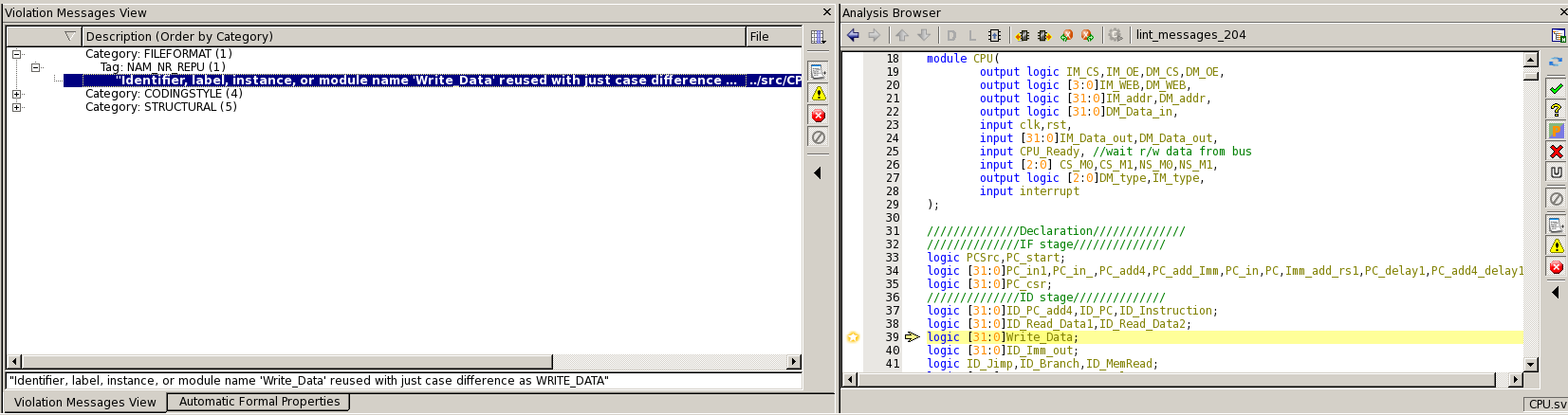
總行數 : 8781

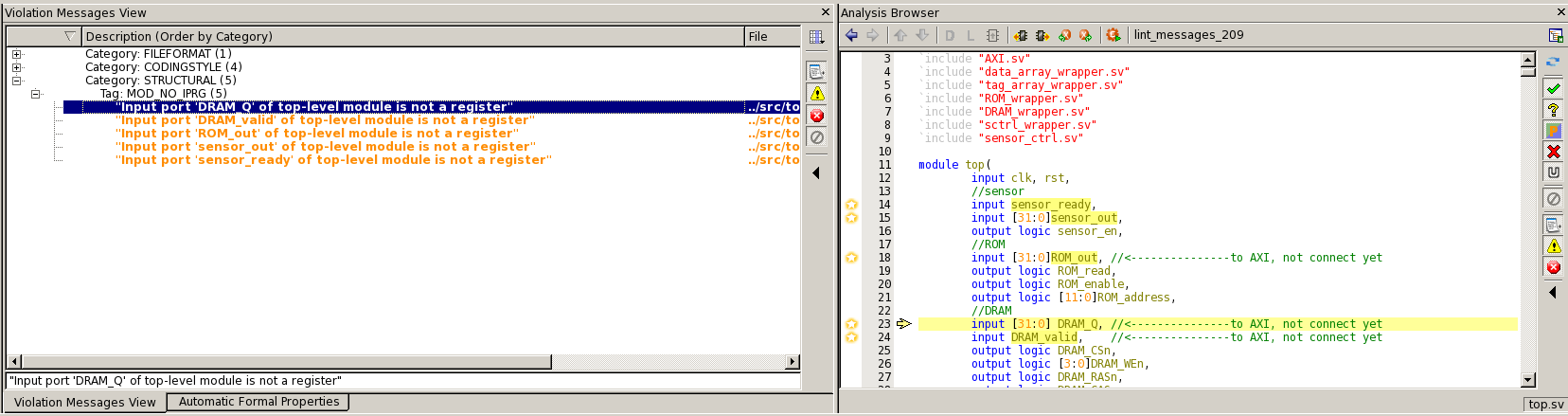
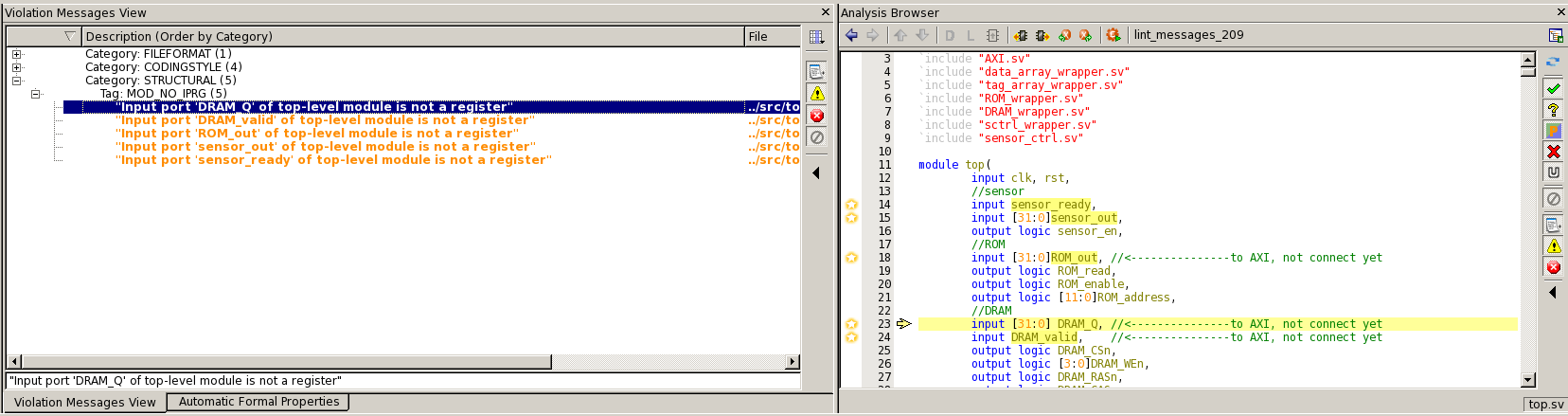
**1-(213/8781)=97.57%**



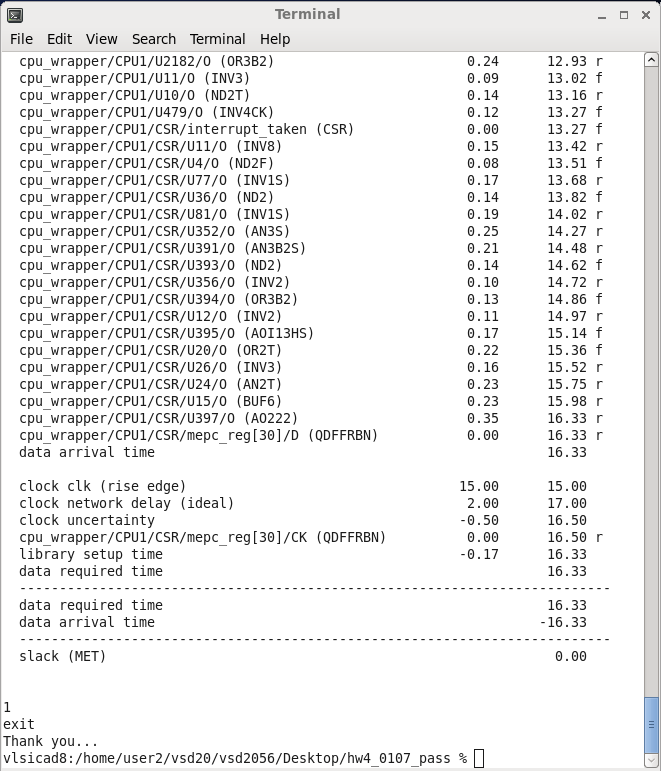
**Superlint analysis**

Superlint只剩下部分問題未解，在此說明

1.有兩個logic命名相似，只有大小寫差異，但未針對此點修改(怕改壞)

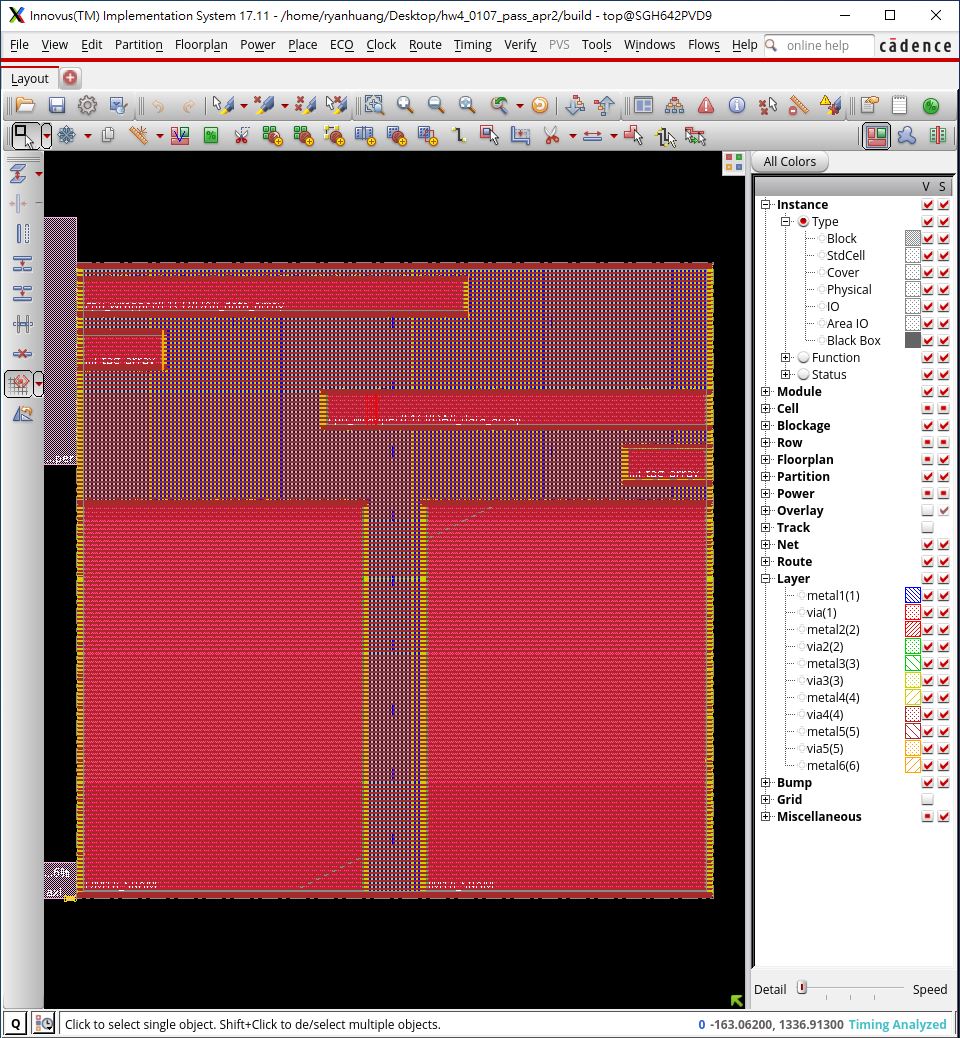
2.Input port not a register，但考慮到這是接到top module之外的port，故沒有特別修正他。

1. **Synthesis**

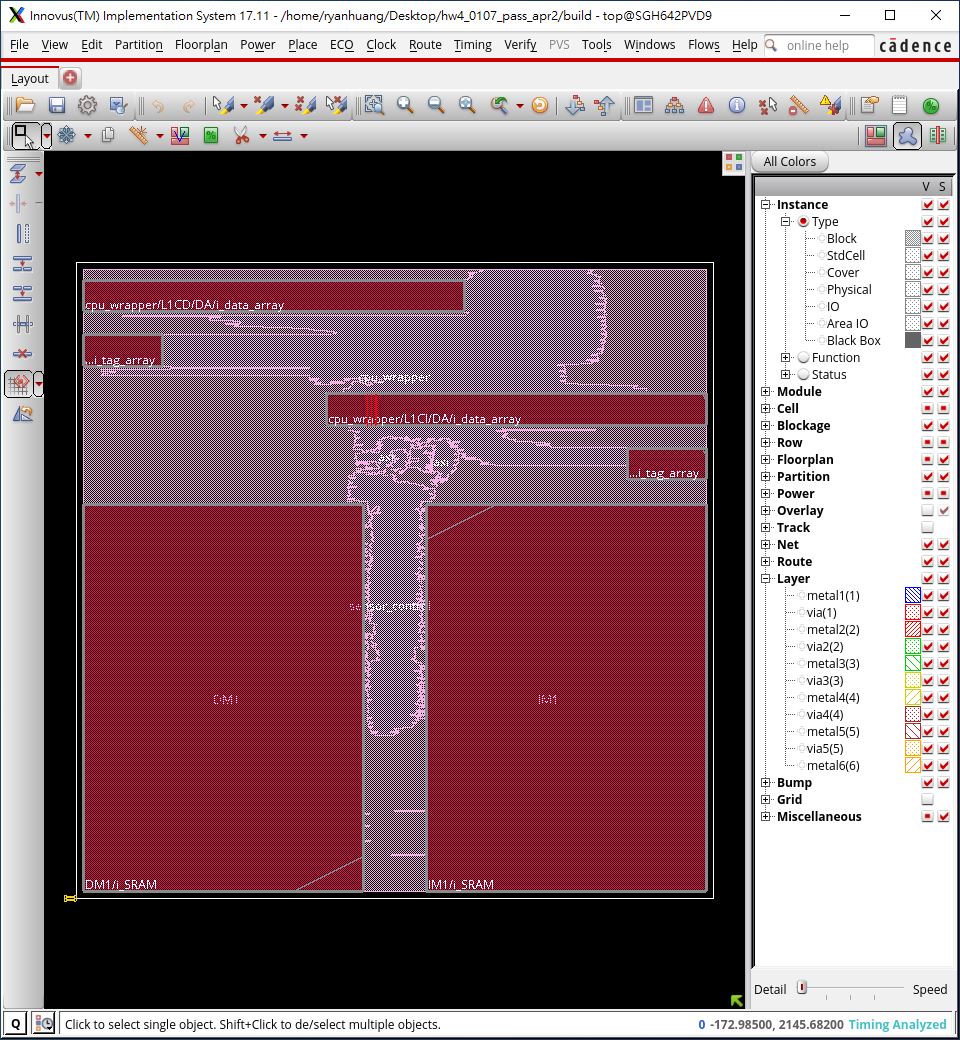
****CLOCK PERIOD = 15

1. **APR**

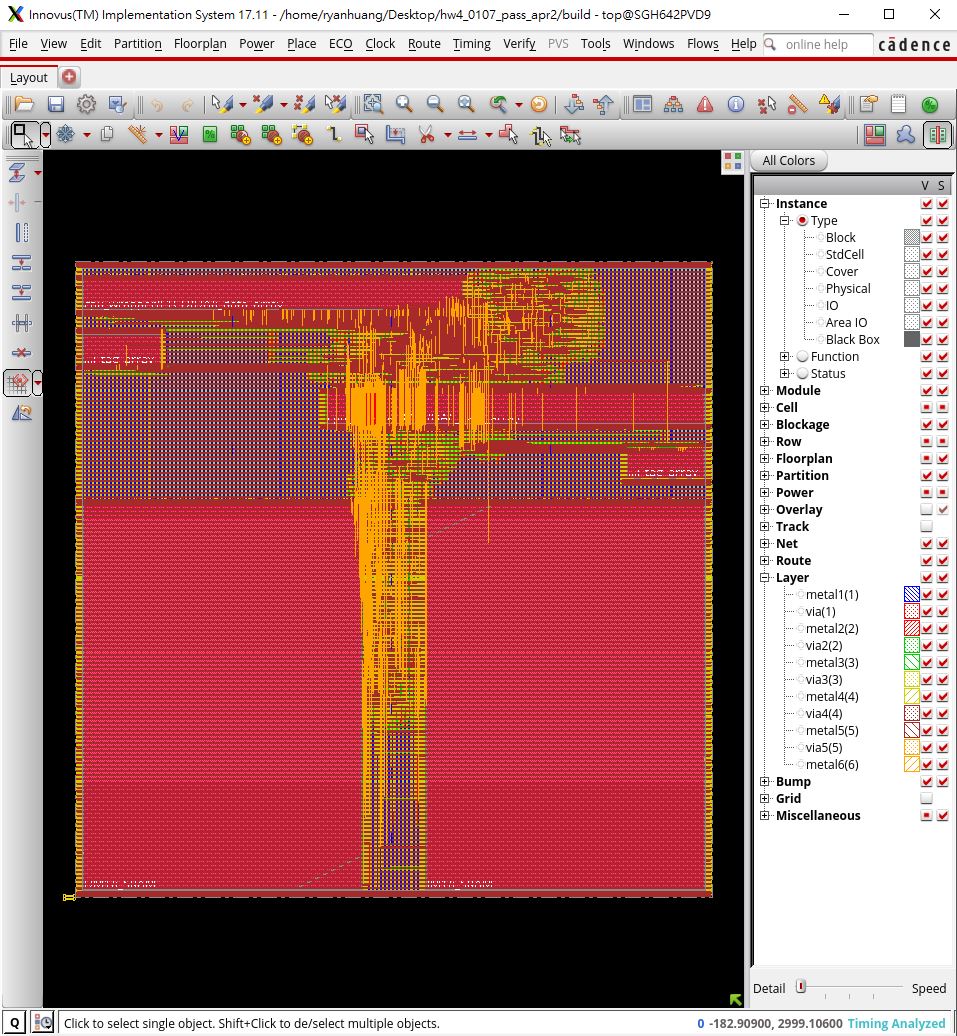
**Floorplan view**



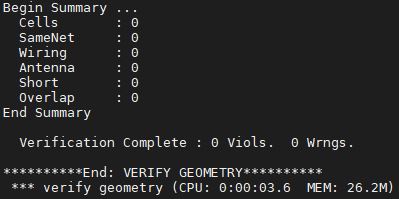
**Amoeba View**



**Physical View**



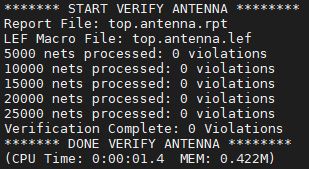
**Geometry Verification**

****No violation

**Connectivity Verification**

****No violation

**Antenna Verification**

****No violation

**Describe the major problems you encountered and your resolutions:**

這次作業我覺得有兩個最困擾我的部分，第一個是SRAM的行為，因為SRAM不管在讀值或者寫值時address都分為row和column，這件事在設計wrapper時需要而外的state來處理，比支前的SRAM會複雜一點；另外，關於CSR module和sensor interrupt的運作也是比較麻煩的，CSR指令在讀資料時只需看給的地址拿相對的CSR register value，但在寫入時卻需要針對寫入資料拆分再寫入CSR register，需要特別注意寫入位置問題，此外，如果考慮interrupt之後更須小心，因為有些CSR register的行為會有所不同，特別注意的是interrupt\_return時，要用mepc內存的pc值取代系統的pc，這也要特別小心，因為有一個自訂義的bias value，還有像是mcycle及mcycleh、minstret及minstreth是進位關係，要設計一段電路來處理mcycle滿了之後進位到mcycleh，以上是我覺得要小心的地方。

**Lesson learned:**

本次我覺得麻煩的點是加入了許多module(CSR ROM DRAM sensor\_control)所以在debug時要先判斷出是誰的問題，而且我覺得關於interrupt的說明其實沒有很夠，所以一些interrupt行為都是和同學多次討論後才得到的結論，雖說如此，但我覺得我對整個系統更加熟悉了，尤其是之前一知半解的AXI部分，現在也更能融會貫通了!