VLSI System Design (Graduate Level)

Fall 2023

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

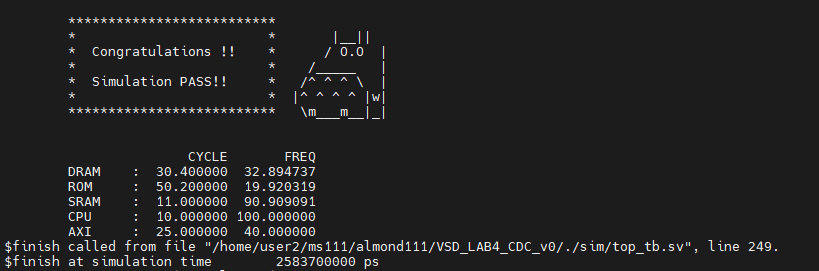
No waveform files in deliverables

Student name: \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

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

|  |  |  |  |
| --- | --- | --- | --- |
| Performance & Area | | | |
|  | rtl | syn | pr |
| Prog0 time |  | 2583700000 |  |
| Prog1 time |  |  |  |
| Prog2 time |  |  |  |
| Prog3 time |  |  |  |
| Area(um^2)  In APR |  | | |
| CPU cycle | 10ns | | |

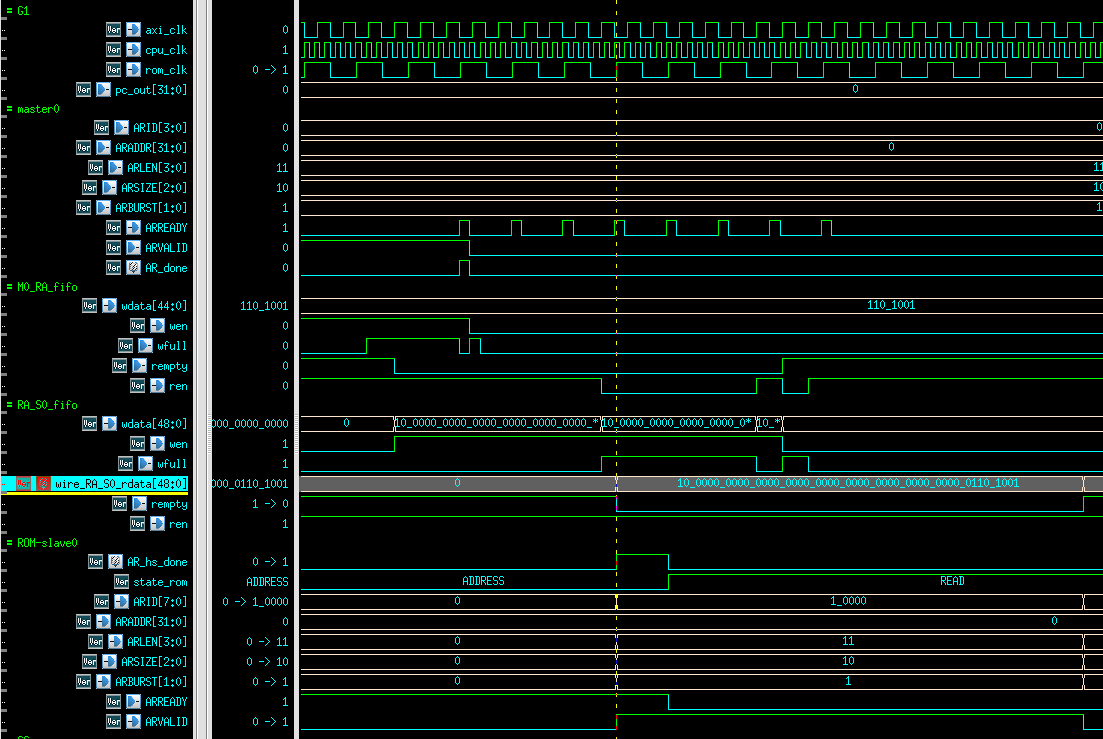
Refer to the figure to fill in the simulation time !!



// Initiate your report from this point.

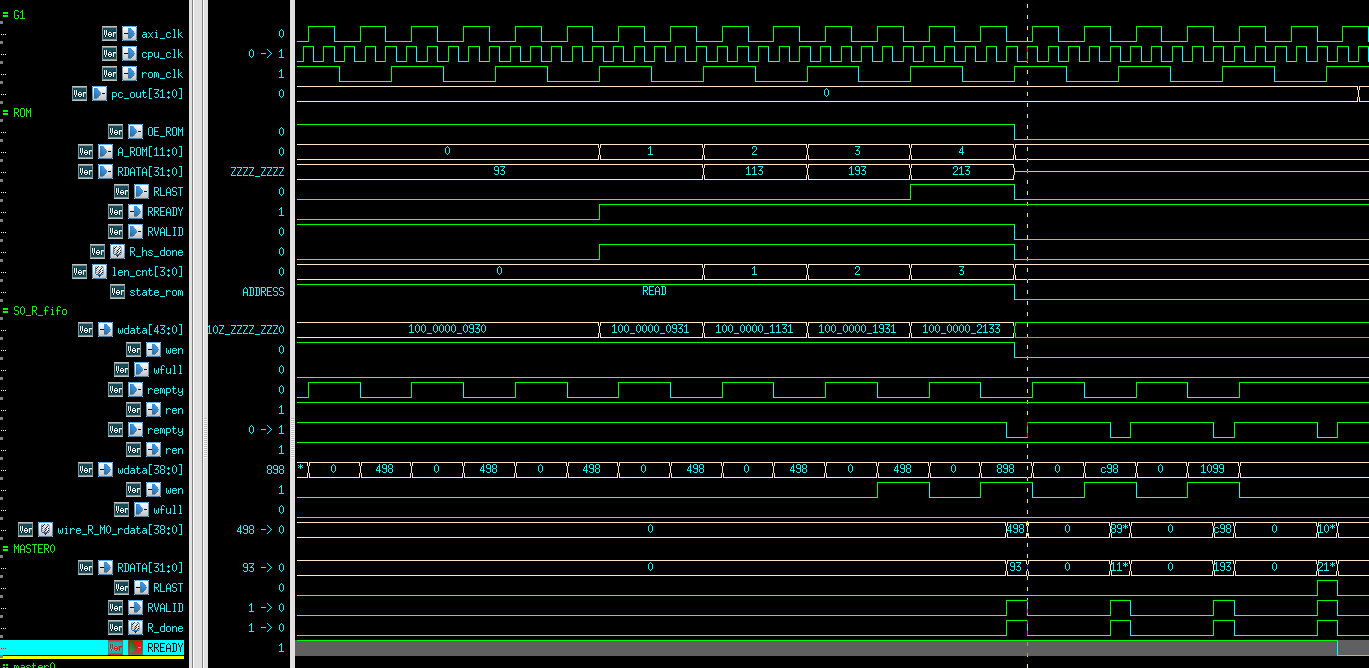
FIFO

M0\_to RA channel :



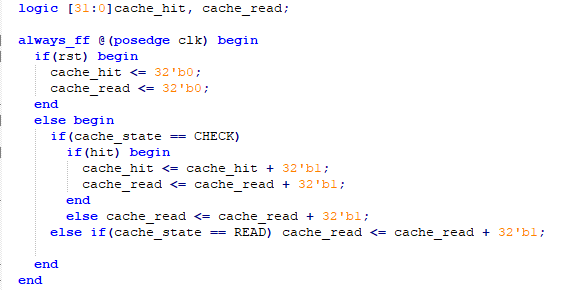
* 在PC\_out 為0時，master0會去ROM搬instruction回來CPU中，因此當整個系統開始時，是master0透過M0\_AR\_fifo再經由AXI給到AR\_S0\_fifo，上圖為其過程的波形截圖。
* 紅框處可以看到，在進入M0\_AR\_fifo之前，會先將data打包成一組資料，即{ARID, ARADDR, ARLEN, ARSIZE, ARBURST}，再等待wen拉起傳入fifo的wdata中。
* M0\_RA\_fifo的wen設定為ARVALID。
* M0\_RA\_fifo的ren設定為 ~RA\_S0\_fifo的wfull (黃框處)，因為當接收端的fifo寫滿時若M0\_RA\_fifo仍在讀出資料，可能會在AXI中遺失。(而接收端為哪一個slave則是透過decoder決定)
* 在上圖中可以看到RA\_S0\_rdata這條線，其是由RA\_S0\_fifo讀出，讀出後再依照{ARID, ARADDR, ARLEN, ARSIZE, ARBURST}的順序解開傳給ROM。
* 藍框處可以看到ROM (slave0)正確接收到由master0傳出的資料。

R to M0 channel :



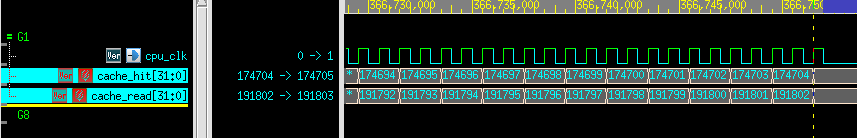
* 紅框處可以看到，在進入S0\_R\_fifo之前，會先將data打包成一組資料，即{ RID, RDATA, RRESP, RLAST, R\_hs\_done}，再等待wen拉起傳入fifo的wdata中。
* S0\_R\_fifo的wen設定為RVALID。
* S0\_R\_fifo的ren設定為 ~R\_M0\_fifo的wfull。
* 藍框處可以看到Master0正確接收到由ROM傳出的資料。
* 此處的細節將在lesson learned提到。

IM cache hit rate



上圖為L1C\_inst的片段程式碼用以計算IM cache hit rate。

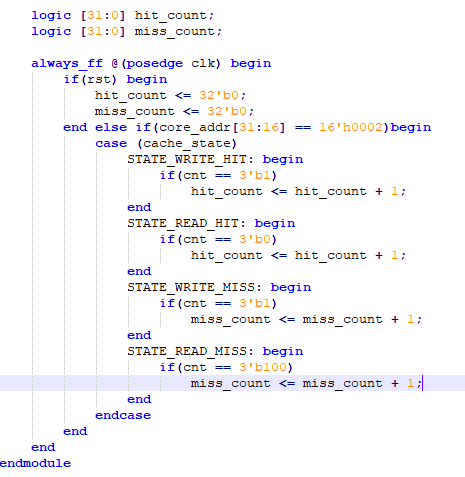
當cache\_state為CHECK時會比對Tag以及查看是否valid，因此在此狀態下判斷cache\_hit是否加一，而一般讀取情況則將cache\_read加一，作為讀取IM的總次數。



查看prog0的波型可以看到在此program執行結束時的hit\_count=174704、read\_count=191803。

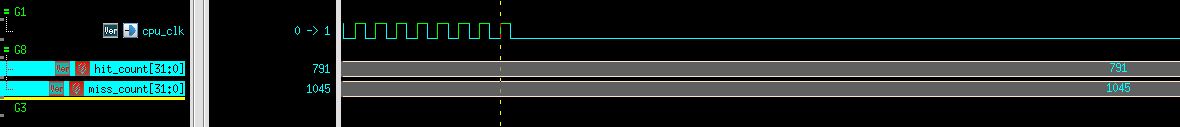
IM cache hit rate = 174704/191802 \* 100% = 91.08%

DM cache hit rate



上圖為L1C\_data的片段程式碼用以計算DM cache hit rate。

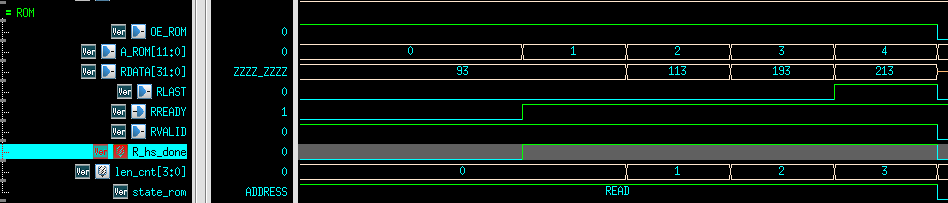
利用core\_addr[31:16]來判斷是否是要存取DM。

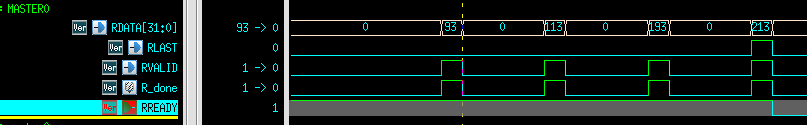
當cache\_state為STATE\_WRITE\_HIT以及STATE\_READ\_HIT時代表判斷為hit，而STATE\_WRITE\_MISS以及STATE\_READ\_MISS則是判斷為hit。  


查看prog0的波型可以看到在此program執行結束時的hit\_count=791、miss\_count=1045，total access count=1836。

DM cache hit rate = 791/1836 \* 100 % = 43.3%

Lesson learned





因為時域不同的關係，在slave的地方address會等待handshake後才開始按照ARLEN的長度以及ARBURST的設定做往後加的動作，因此在本次作業雖然RVALID已經拉高，但是master端並不會直接接收到，因為當master端接收到RVALID後，也會相應的計算count來接收對應資料，但是因為時域不同，ROM端的資料送出較慢，而CPU端會有資料遺失的狀況。

因此我們在本次作業新增一條R\_hs\_done，由slave送出到RD channel中，新增(R\_hs\_done)?的條件在master接收RVALID之時。

因此可以看到上圖波形中master端的RVALID會和資料同時high以及同時low。