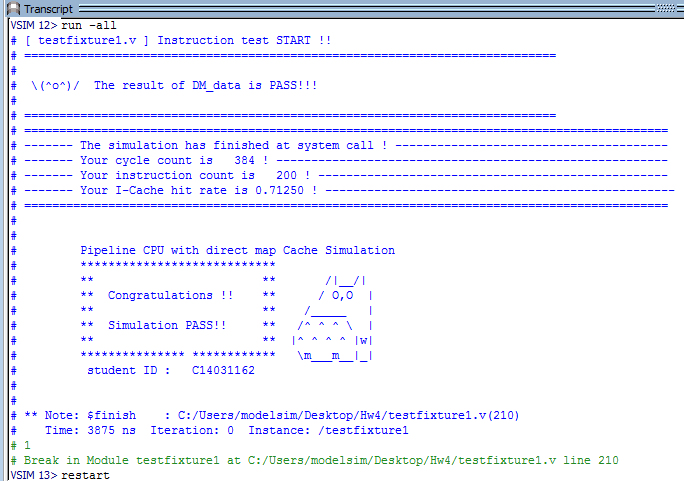
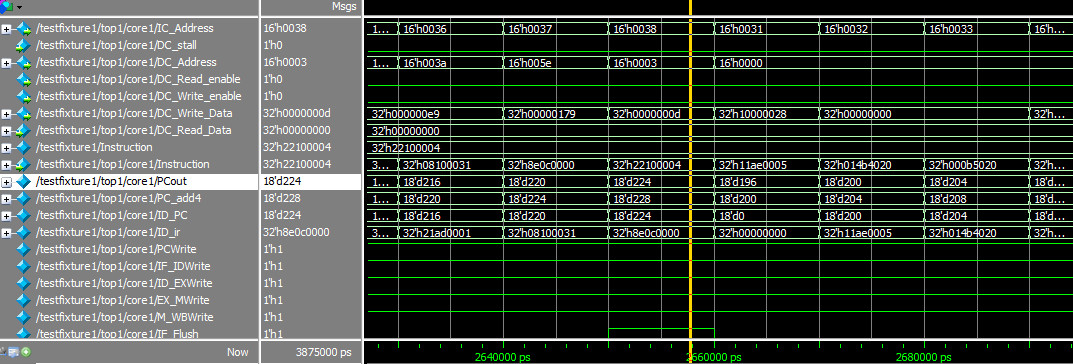
**Computer Organization 2016**

**HOMEWORK IV**

系級: 資訊107 學號: C14031162 姓名: 洪正皇

**實驗結果圖(snapshot of result)**

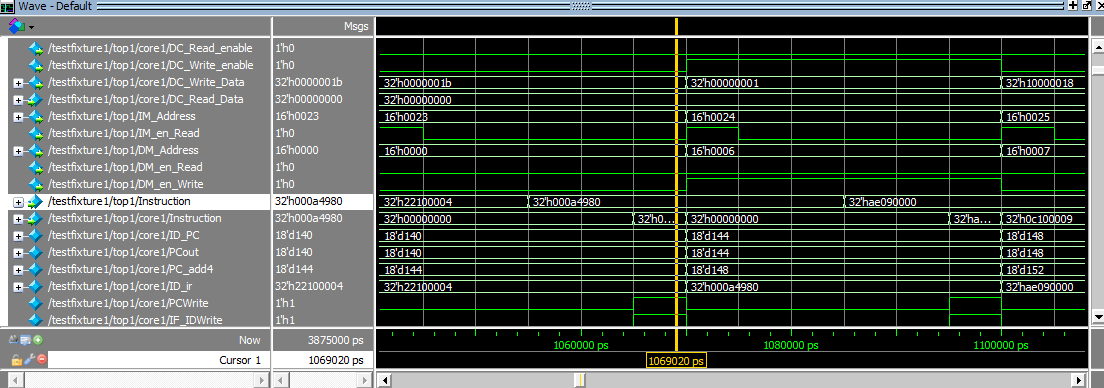


**The I-Cache hit rate of your simulation & your reason:**

根據I-Cache hit rate和instruction count算起來，hit到的指令約莫有142.5個(為什麼不是整數QAQ?)，波形圖中可以看到PC在196~224之間不斷重複，應該有8個指令，但jump指令在EX階段才決定，因此後面2個指令會被flush掉，結合波形圖與IM\_data看來，應該是第51~56行的指令loop執行了25次，除了第一次以外都hit，總hit指令數=>6\*24=144，hit\_rate=>144/200=0.72，指令數與hit\_rate結果皆與程式執行結果接近，故為此結果。

**指令波型圖( Instruction waveform of some situations)**

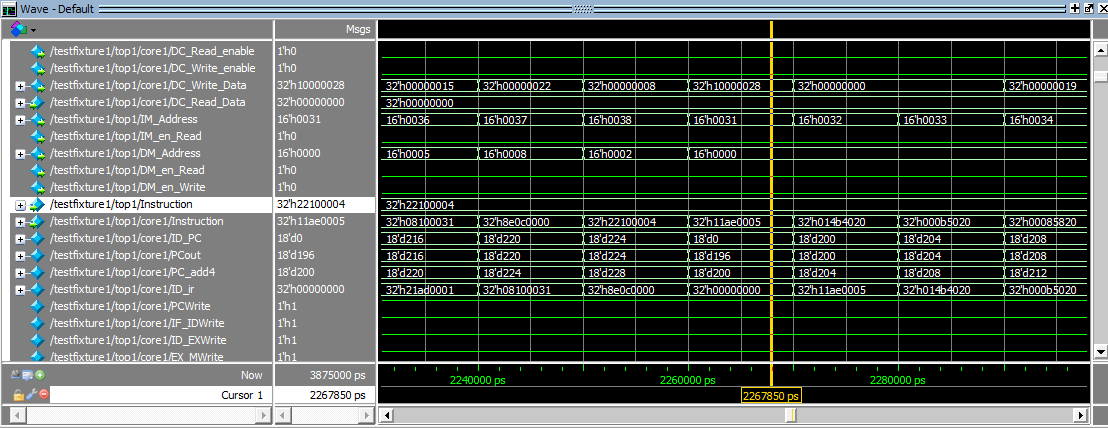
(Please explain why your snapshot is correct, including the wires, signals.)

1. I-Cache Miss

Description:

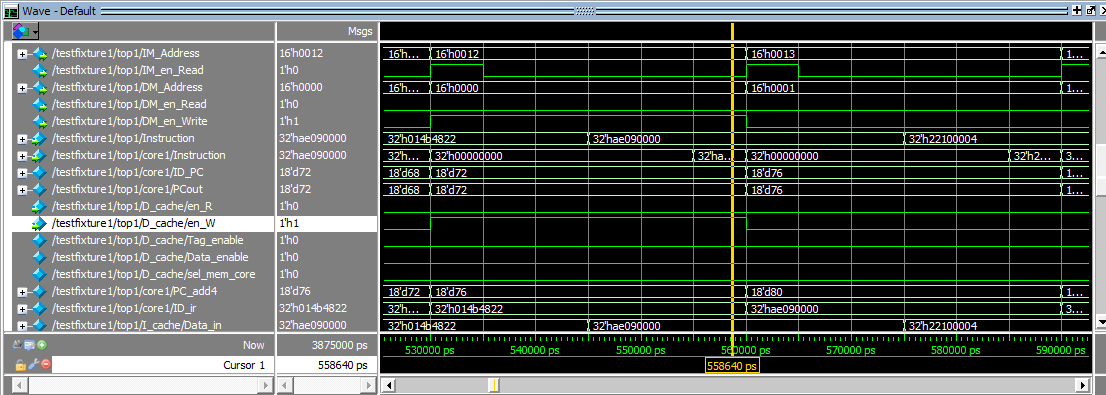
在圖中最左邊可以看到IM\_en\_Read==1，代表I-cache read miss了，故需從Mem中讀取資料，可以看到過一下子後top1/Instruction的值更新為32’h000a4980，此為Mem中讀到的Instruction，隨後可以看到黃線的地方top1/core1/Instruction的值也更新為32’h000a4980，由此可知I-cache read miss的時候，有成功的從Mem讀進data存到I-cache中，並且正確的將data從I-cache輸出到core中，此Miss正確的從Mem中拿到data來使用，同時更新的cache中的值。

1. I-Cache Hit:



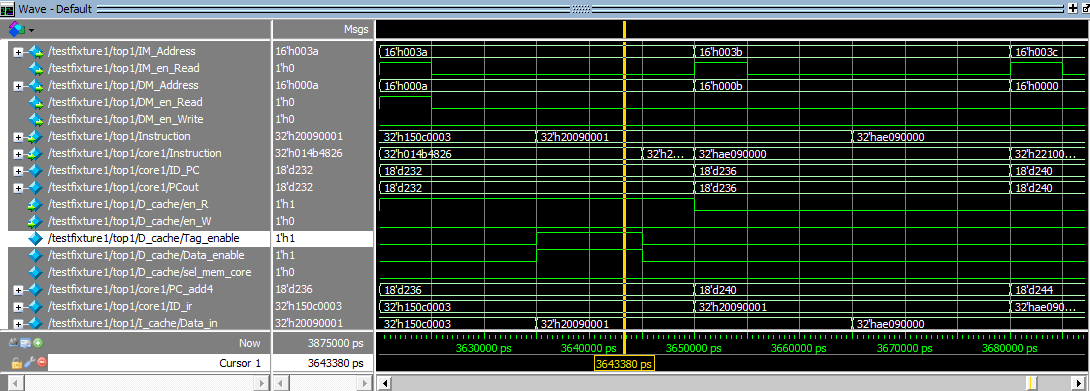
Description:

從IM\_en\_Read==0中可以看到，沒有取用Men中的Instruction，代表I-cache read hit了，直接取用I-cache中存的Instruction，可以看到top1/Instruction一直都沒有變，因為IM\_en\_Read==0，故一直不會從IM去更新Instruction，而top1/core1/Instruction的值一直在更新，code中可以看到此Instrucion是I-cache的輸出結果，又沒透過Mem更新I-cache值，故可知此Instruction是從原先就儲存在I-cache中的資料拿到的，故此hit正確的不透過Men而直接從cache中拿到資料。

1. D-Cache Write Miss

Description:

由D\_cache/en\_W==1和DM\_en\_Write==1可以看到，嘗試從D\_cache中找到可以寫入資料的位置，但下面的Data\_enable訊號為0，故在cache沒有hit，所以D-cache write miss，才需要直接到Mem中寫入資料，此狀況為Write through的機制沒錯，Miss後忽略cache，直接寫入Mem。(IM\_data中可以看到沒有Write hit QQ)

1. D-Cache Read Miss

Description:

由最左邊DM\_en\_Read==1可以知道，此為D-cache read miss，才需要從Mem中取資料，而tag,data,vaild的enable訊號也都設為1，可以將資料存進cacheline ，再讓cache中的data被取用，正確的處理了read miss。(IM\_data中可以看到沒有Read hit QQ)

**心得(Report)**

(請寫下完成本次作業的心得、學到哪些東西、困難點的部分。大約   
 100~200字 )

(Please write your learned lesson and conclusion, and difficult point. About   
 100~200 words)

這次作業助教真的是佛心大發，cache.v的部分完全看圖就接好了xD，但是cache\_control.v的部分就要自己動動腦了，一開始以為Read\_miss就是沒有hit，結果從一開始就沒有hit，產生read\_miss，CPU被stall到死，後來才發現沒hit也可能是write\_miss，de完bug後神清氣爽。原先還沒有看到龍貓上面有印好的Instruction count和Hit rate，一段一段的看了波型和IM\_data後找到迴圈，想自己找出Hit rate，不過後來那些動作就都變成了驗證Hit rate的方法了，差點以為做了白工(如果不用驗證Hit rate的話)。