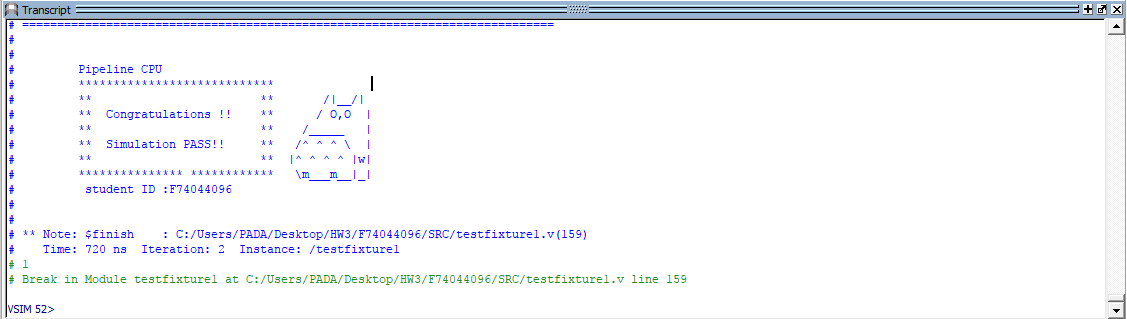
**Computer Organization 2017**

**HOMEWORK III**

系級: 資訊系108 學號: F74044096 姓名: 吳元智

**實驗結果圖(snapshot of the results)**

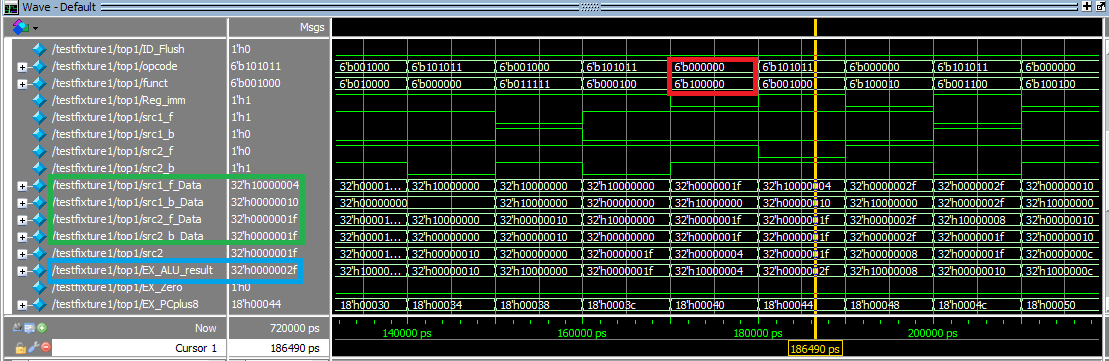
****

**指令波型圖(Snapshot of the instruction execution waveforms.)**

(Please explain why your snapshot is correct, including the wires, signals. The description should be as detailed as possible, e.g. why this situation occurs, and in waveform where does it occur?)

1. Instruction with Forwarding

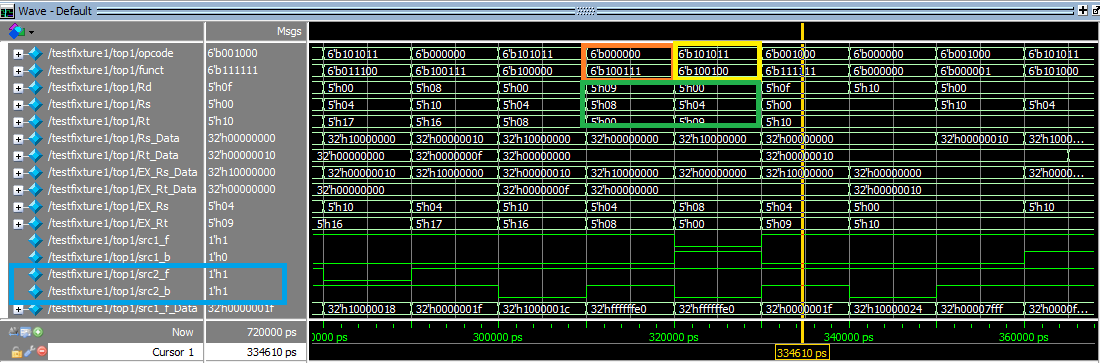
R-type:



Description:

因為pipeline的關係，在看計算後的資料得要到下一個階段來查看，像是目前所對應到的資料是根據紅色框框所框住的指令，而此指令會將Rs的資料(src1\_b\_Data) 及 Rt的資料(src2\_b\_Data)做add，最後得到ALU\_result

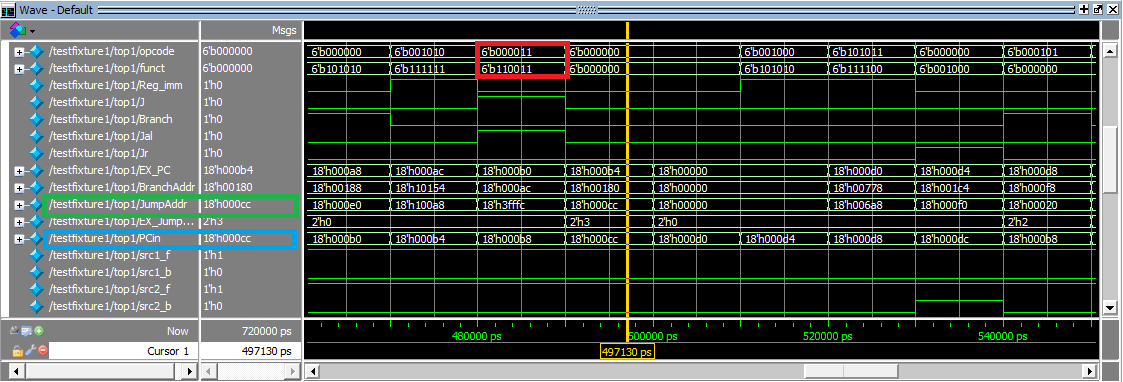
I-type:



Description:

橘色框框所執行的指令是nor，而他的下一個指令是sw，從綠色框框可看到在nor時，其Rd為sw時為Rt，而為了處理forwarding，會讀取其傳至MEM區段的資料，而從src2\_f = 1及src2\_b = 1即可看出

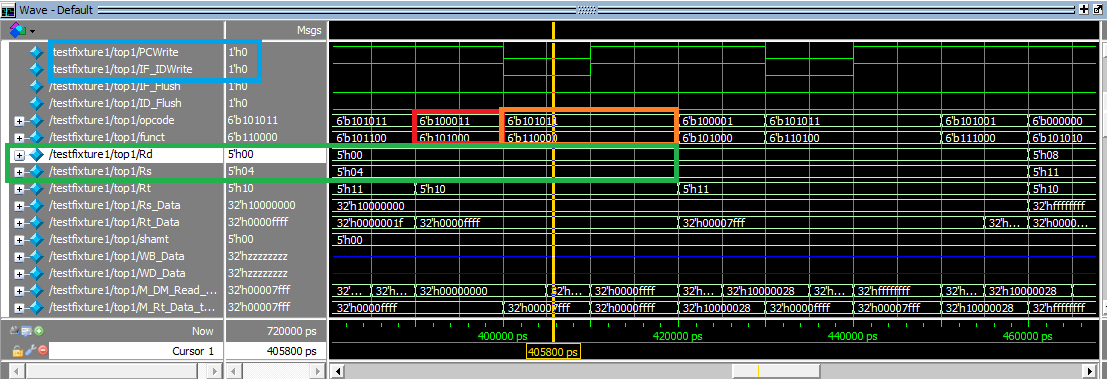
J-type:



Description:

由紅色框框可判斷出其指令為jal，而接下來可以看到藍色框框所讀取的PCin為綠色框框的JumpAddr的位址，所以可以判斷jal為正確的

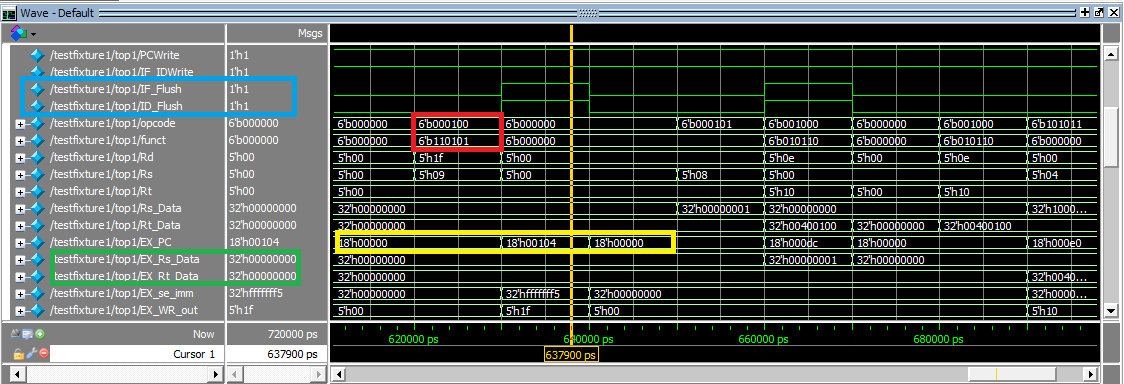
1. Load Stall:



Description:

由圖可看出紅色框框為lw，因為他要存取的Rd是下一個指令的Rs，所以在這裡發生stall，而從PCWrite = 0及IF\_IDWrite = 0可看出局色框框的opcode及funct都沒變

1. Branch Delay (& Flush):



Description:

由圖中紅色框框可看出其為beq指令，所以在EX區間中，會將EX\_Rs\_Data及EX\_Rt\_Data做比較，因此下一個EX\_PC的值不會+4(00000 != 00108)，而IF\_Flush = 1及ID\_Flush = 1是為了清除錯誤的指令

If you CPU data is from Figure 5 in the home, show your CPU datapath and explain why you want to design your CPU this way.

Reason:

**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)

從這次作業當中，我對於pipeline有更多的了解，像是在處理每條指令的時候，不同的區間可以做不同的事情，可能MEM區間正在儲存上一條指令得到的值的時候，在ID或EX已經需要這個值了，這時藉由forwarding來處理，不僅可以避免回傳值的錯誤，還可以減少執行指令的時間，大大降低所耗時間，而在寫程式的時候遇到的最大困難點便是forwarding和hazard detect這兩個部分，hazard的功用一方面是在處理資料和執行beq，bne時，為避免出錯，所以會用flush來將它處理掉，另外forwarding則是另一個要處理的重點，為了在儲存上一條指令的值時，可能下一條指令就需要用到，所以降forwarind用好花了我好一段時間