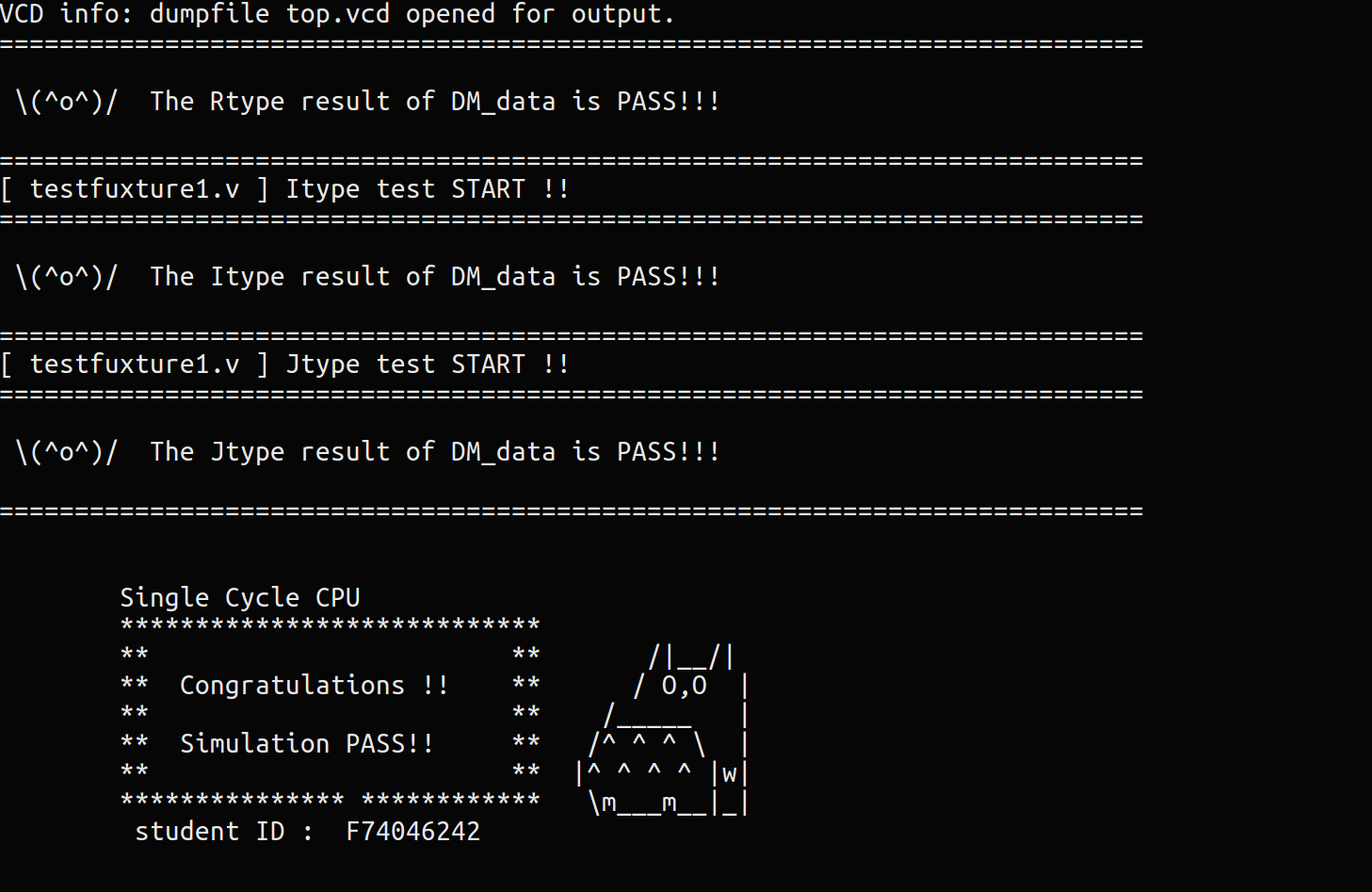
**Computer Organization 2017**

**HOMEWORK II**

系級: 資訊108 學號: F74046242 姓名: 謝耀賢

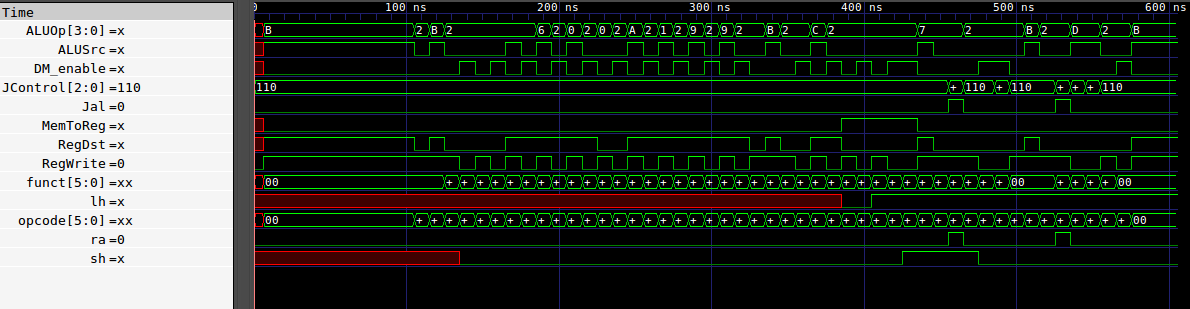
**實驗結果圖(snapshot of result)**



**指令波型圖( Instructions waveform )**

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

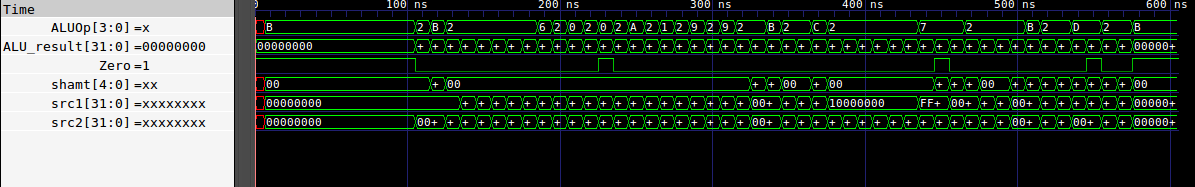
Controller:



Reason:

controller.v 輸入 op 和 func ，依照不同的op跟func判斷此指令是何指令，並依照該指令來設定7個選擇器、regWrite、DM\_enable、Jcontrol(給Jump\_ctrl.v使用)、AluOp(給alu.v使用)

ALU

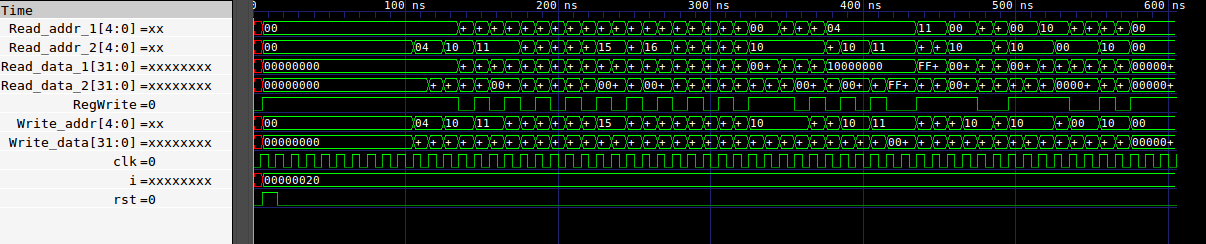


Reason:

依照輸入的AluOp，來判斷該執行何運算(add、sub、and、or、...etc)運算結果傳出

其中若有使用到beq、bne指令，則會傳出Zero值(給jump\_ctrl.v使用)

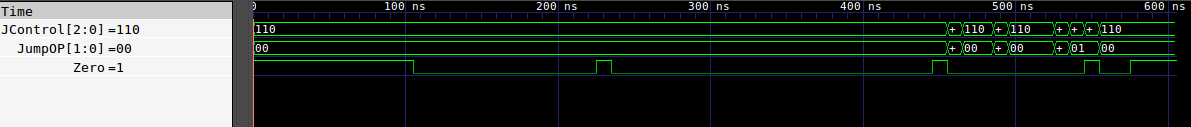
Regfile



Reason:

若regWrite為true，則依照輸入的address和data，將data存進reg[address]內

Jump\_Ctrl



Reason:

依照輸入的Jcontrol(j、jr、jal、jalr、beq、bne、default)來轉換成其相應的JumpOp(PC+4、PC+4+imm、Rs、jumpaddr)，若指令是beq或bne，則會依照Zero值進行判斷

CPU datapath (If your CPU datapath is different from HW2 Fig.2.)

Reason:

same as HW2 Fig.2.

**心得(Report)**

一開始因為太久沒有寫verilog，所以看到datapath的時候無從下手，之後先寫完PC.v、regFile.v之後，就逐漸知道之後的檔案該如何寫。不過在debug的時候，因為controller內選擇器有忘記做設定的動作，或是選錯線的緣故，導致debug的時後bug一大堆，且在debug J type的時候，因為沒有答案對照表可以用，只能看波形圖來找出錯誤，這時因為有同學的波形圖做參考，所以debug的時間沒有花非常久。總體來說，這次作業讓我回憶起verilog該怎麼寫....，並從中學到了如何刻一個single cycle CPU