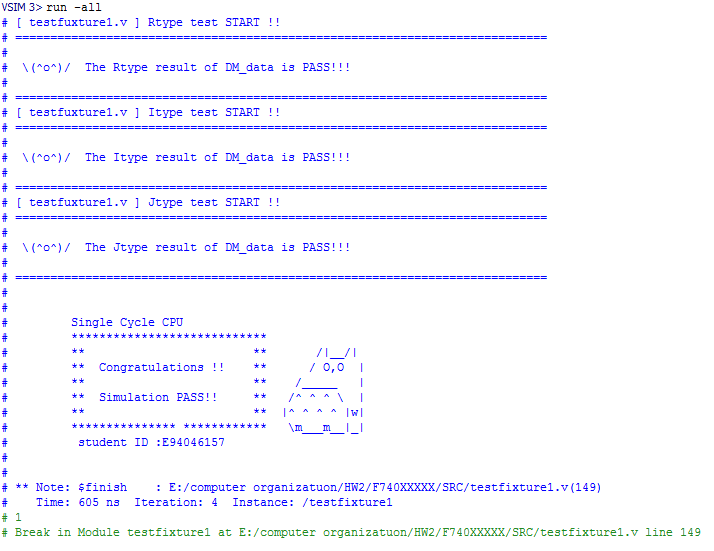
**Computer Organization 2017**

**HOMEWORK II**

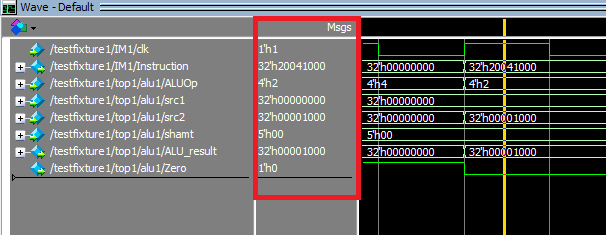
系級: 資訊108 學號: e94046157 姓名: 蔡宇軒

**實驗結果圖(snapshot of result)**

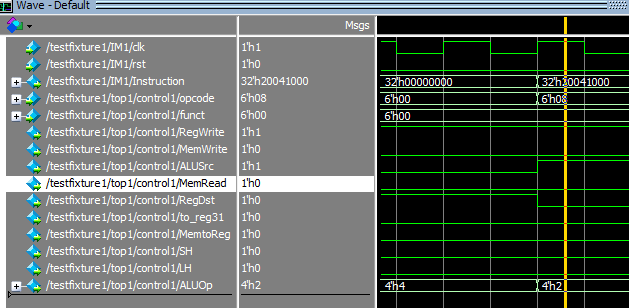
**指令波型圖( Instructions waveform )**

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

我取的指令是20041000，addi $a0, $0, 4096這個指令。可以看到ALUOP是4’h2，也就是二進位的0010，執行的是加法，而src1是$0的部分，因為一開始我有把暫存器都初始成0。所以$zero會是0無誤。而加上src2，也就是經過sign-extented的4096，而結果ALU\_result會是4096，因為答案不是0，zero的訊號會是0。



Controller:



Reason:

我的controller，是根據opcode與function來控制不同指令能夠做的事，而controller的output，輸出接到許多不同元件例如多工器，IM、DM等。RegWrite/MemWrite是控制是否要寫入暫存器/記憶體。

ALUSrc則是告訴ALU src2是來自於rt或是imm。

MemRead是告訴DM是否讀出資料。

RegDst則是告訴暫存器結果是存在Rd或是Rt。

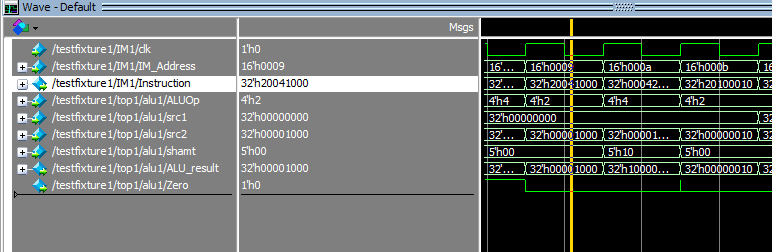
to\_Reg31是用來處理jal跟jalr，要把資料存進R[31]。

MemtoReg則是控制是由memory或是ALU的結果傳進reg中。

SH、LH是解sh/lh指令時的獨有訊號。

ALUOp則是控制ALU應該做甚麼運算。

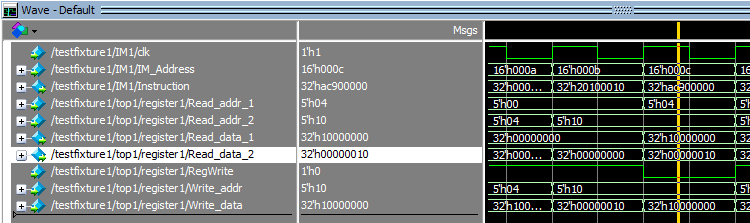
ALU



Reason:

ALU是根據controller輸出的ALUOP去運作，在這裡，我設定ALUOP=0000~0111分別是做and、or、add、XOR、sll、srl、sub、slt這幾項運算。

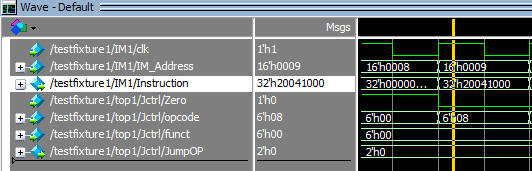
Regfile



Reason:

我的register，在一開始寫完時發現都是紅線XXXXX，後來發現把他在rst處先歸零所有reg就沒問題了。而其中運作，我先用Read\_data\_1 = Register[Read\_addr\_1];，把output資料取出，而後根據RegWrite，決定可不可以寫入新資料，若不行就用不做任何事跳出。

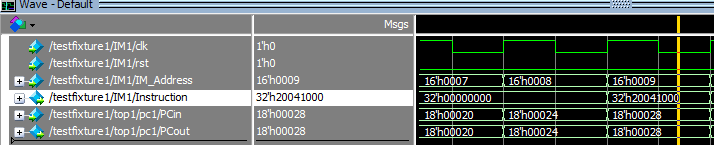
Jump\_Ctrl



Reason:

我的Jump\_ctrl與助教畫的datapath，輸入較不相同。課本是和助教類似，用zero這個訊號，加上branch等訊號作處理得到結果。而我是直接把opcode和funct輸入，根據其指令直接判斷要跳到哪個PC地址。其中，JumpOp會接到一個多工器，判斷下一個位置是跳到1.jr、jalr指定的位置2.bne、bnq指定的位置3.j、jal指定的位置4.其他，即為原本的PC+4的位置。之所以會想要改變設計是因為，Zero課本中是用來判斷beq，若是相減=0，zero訊號輸出為1就跳branch。但是若是多加bne，zero訊號為0時才要branch，就可能還要分兩種狀況寫了。因此我就用感覺較簡易的方法撰寫。

PC



Reason:

算PC的部分，我是直接把initial設為4，且clk來時就固定+4，在datapath要傳入IM的數據再用wire pc\_in\_IM = PCout-4來做。而輸入的部分，就是由Jump\_Ctrl控制的多工器來選擇輸入。

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

Reason:

我的datapath與助教提供的圖幾乎相同。唯一不同的地方在JumpCtrl的input，是用opcode和function 而使ALU的zero訊號無作用。

**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)

這個作業，花了我十分長的時間去作。光是先看懂4.1~4.4的內容，再把code打出來，我就花了一周半的時間，其中也上網看了一些資料才知道。而剩下的部分我覺得才是最難的，就是debug的部分。一開始debug根本不知道從何看起，只能從頭到尾一直看，都摸不到頭緒。看了許多資料才了解，波形圖紅線可能就是有問題的地方，我才終於從其中改了一些錯誤，例如reg要歸零。而到做完的那周，只花了一個晚上的時間，就從test一開始就錯誤變成全部通過了。因此我認為工欲善其事，必先利其器，這次的作業最困難的部分我認為還是在verilog的語法上，如何有效率地找出邏輯或語法錯誤，而我學到的方法是邊看wave圖，邊看助教的IM\_data檔案，才能找到到底是哪裡有問題。