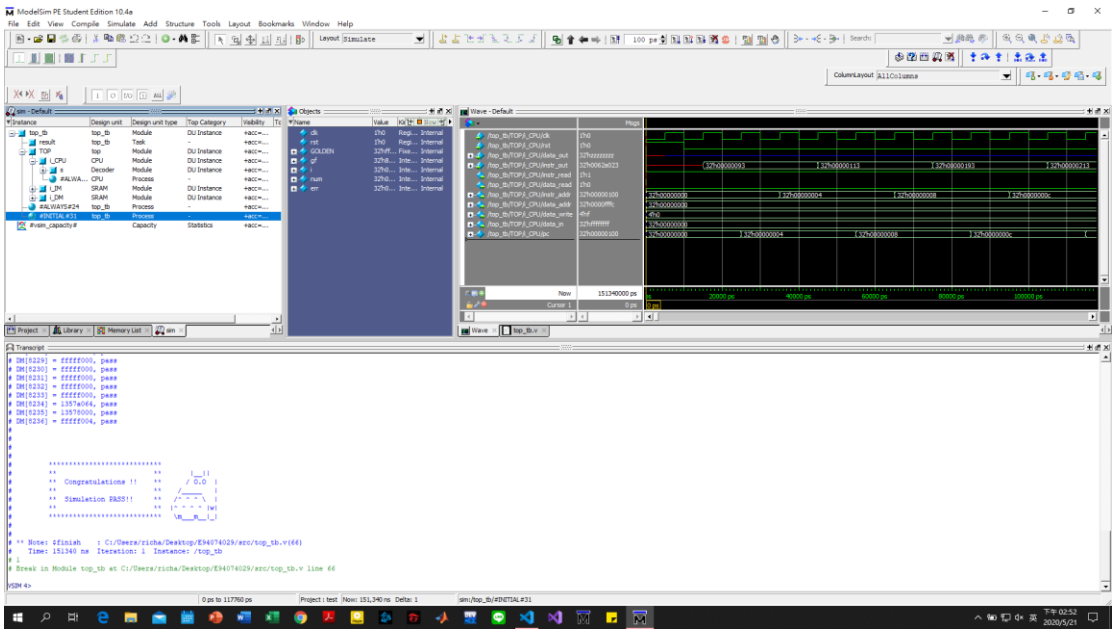
Computer Organization 2019

HOMEWORK 4

系級: 資訊 111　　學號: E94074029　　姓名: 江羿賢

實驗結果圖:

(波形圖及模擬完成截圖)



程式運作流程:

(簡單說明波形變化的意義)

波形變化:

clk:進來的 clk 的變化

rst:當這個訊號為 1 則做 reset 的動作

data_out:當有 data_read 的訊號，下一個 clk 會給 data_out，load 會需要用其值來 load

instr_out:當有 instr_read 的訊號，下一個 clk 會給 instr_out，用來讀取當前的指令來做 decode

instr_read:要得到 instr_out 時，必須先使得其變成 1 之後才會拿到 instr_out

data_read:要得到 data_out 時，必須先使得其變成 1 之後才會拿到 data_out

instr_addr:為 instr 的地址

data_addr:為 data 的地址
data_wirte:為要寫入 data 時所需的訊號，訊號 1 為該 byte 要寫入，此訊號共有 4bits
data_in:為要寫入 data 的東西，需要搭配著 data_write 做使用才能順利寫入
pc:當前 pc 的位址，以便查看 pc 到哪裡了

程式運作:

講一點比較大塊的，先利用別的.v 檔處理一些 decode 的工作如 opcode funct3 跟 funct7，並且依照輸入的指令，給出各個指令的 imm，有不需要用到 imm 的指令的話就可以把其設在 default 給值 0，再來一開始利用 initial 去做把各個數值做初始化的動作，就可以進入 always 的部分了，裡面首先做的事情是看有沒有 rst 的訊號，如果有就把值初始化，再來是我把流程劃分成 5 個 state，來處理，首先 state == 0 時就先把 instr_addr 設為 pc，以及確認 instr_read 有給值，跟把 register_file 中的 0 號確保為 0 就可以進行下一個 state，在 state == 1 時因為 instr_read 給值會在下一個 clk 給，而為了確保不會讓資料撞車，不在同一個 clk 中做運算，怕這樣處理完的資料會產生錯誤，所以在這邊就再等一個 clk，在 state == 2 時開始做指令的運作，先把 pc+4 後，把 rs1,rs2,rd 給值，並先清空 data_read 跟 data_write，就可以開始做運算了，並且依照 opcode funct3 funct7 來確認目前要做的是哪個指令，並且將其完成，其餘的指令就不多做贅述了，先來說一下 store 的部分因為 data_addr 的位置不一定是 4 的倍數，所以需要幫其確認好位置，並且分成 4 種情況來寫入 data，餘數每多 1 就往左挪動一個 byte 才能夠順利地執行指令，而這些指令都做完以後就可以回到 state == 0 做下個指令了，但是除了 load，因為需要先給 data_read 的訊號就要等下個 clk 給值才能做，所以在這邊會到 state == 3 先去確定訊號不會撞在一起導致運算的錯誤產生，並且在這邊順便將 read 的訊號設成 0，再來就可以去 state == 4 的地方去做運算了，大致上的程式流程就是這樣子。

心得

（請寫下完成本次作業的心得、學到哪些東西、困難點的部分。）

這次的作業可以說是一大挑戰，因為要處理的指令滿多的，在 debug 的部分會比較困難一點，而還有一個比較困難的地方是給出 read 之後，下一個 clk 才會

給我值，但是如果在下一個 clk 處理的話，有的時候資料會撞在一起，所以變成我選擇用多等 1 個 clk 的方式，來避免這個情況的，而這個的問題其實我找了很久都沒有發現，因為指令真的太多了，有時候會以為是 pc 亂跳到其他地方所導致的錯誤，要仔細的印出各個數值出來或者是慢慢看波形才比較有辦法去找出問題點所在，以及 store 的部分，在確認位置時也要小心的確認是哪裡的問題，才能夠準確的找到是哪裡出了問題，在這次的作業當中學到了非常多東西，也寫了非常久，其中學到最多的是在眾多指令當中學到了怎麼樣子去有耐心的 debug，去找到到底哪裡出現的問題，整體來說學到了真的很多。