Hw4 report

By 許庭瑋

基本上就按照助教提供的 hw4 簡介去做。Pdf 第 2 頁的 data path 圖和第六頁的 module 提供了很大的幫助。我的 os 是 windows 10，tool 是 iverilog。各個 module 實作介紹如下:

CPU module 是整合所有其他 module 的。哪個 module 的 input 是哪個 module 的 output，哪個 module 的 output 是哪個 module 的 input，都在這裡進行連接。Pc=pc+4，讀出 instruction，分解 instruction，哪個部分 input 到哪裡都在這裡進行。

PC module 是助教提供的，主要用途是把 clock signals, reset bit, start bit 讀進來，然後根據這三個數值的變化進行 pc_o 的更新。當 clock signal 是 positive 時，才會更新 pc_o。當 reset bit is on 時，就重置 pc_o。當 start bit is on 時，PC 就會被更新。

Register module 是助教提供的，是用於存放 32 個 32bits registers 的地方。讀取 clk_i，當 clk_i 是 positive 時會才進行 write 的動作。讀取 RS1addr_i，RS2addr_i，然後根據讀取的 address 把相對應暫存器的值讀出來，然後用 RS1data_o，RS2data_o 輸出。讀取 RDaddr_i，RDdata_i，RegWrite_i，根據讀取的 address 和要寫入的 data，然後根據 RegWrite_i 來判斷要不

要進行 write。

Instruction_Memory module 是助教提供的，是用來存放 Instruction 的。首先用 reg 設置好。然後就根據 addr_i 讀進來的值去判斷要輸出哪個 instruction，然後放到 instr_o 去輸出。

Adder module 是用來做加法的。讀取 data1_in 和 data2_i，n 然後把這 2 個值加起來，放到 data_o 輸出。在這個 simple CPU 裡，它的作用是做 pc=pc+4 這個運算。

Control module 是根據 opcode 來判斷是 control signal。把 instruction 的前 7 個 bits 讀進來(opcode)，然後根據 R type 還是 I type 來確認 ALUOp_o 和 ALUSrc_o 的值來輸出。因為基本上這個 simple CPU 的所有 function 都會 write 回 register memory，所以 RegWrite_o 固定是 1。

MUX32 這個 module 是用來做判斷的，根據 Control module 傳過來的 ALUSrc_o 是不是 0 來判斷說我要選 register 讀出來的值還是 sign extend 的值。

Sign_Extend module 是用來做 sign extend 的。一個 12bits 的值傳進來，要變成 32bits 傳給 MUX32 module，而且數值和正負不會變。因為正負的判斷是看最左邊那一位，0 是正，1 是負。然後做 sign extend 的話，如果是正的，前面多出來的就填 0，如果是

負的，前面多出來的就填 1。因此可以選擇把最左邊那一位複製 20 次，就可以做到以上的要求了。

ALU_Control module 是根據 instruction 分出來的 func7+func3 和 Control module 傳過來的 ALUOp_o 來判斷 ALU 要做什麼運算行為。如果 ALUOp_o 是 2，代表是 R type，所以根據 func7+func3 來傳相對應的 ALUCtrl_o 給 ALU。如果 ALUOp_o 是 0，代表是 I type，所以把 func7+func3 中的 func3 分離出來，然後根據 func3 來傳相對應的 ALUCtrl_o 給 ALU。

ALU module 是做運算的。讀取 register module 傳來的 data1_i 和 MUX32 選出來的 data2_i 這 2 個值和 ALU_Control module 傳過來的 ALUCtrl_o 來判斷要對這 2 個值做什麼運算。然後把結果放到 data_o 裡，傳回 register 去 write。