計結 Project 1 Report

B07902001 徐維謙
B07902075 林楷恩
B07902123 蔡奇夆

Module Explanation

Control and ALU_Control

- Control
  input signal 新增了一個 NoOp_i，如果有這個 signal 的時候，就讓所有的輸出變成 0。
- ALU_Control
  新增了 lw,sw,beq 三個指令的 ALUCtrl_o 的輸出。

Pipeline Registers

- IF/ID
  PipelineRegIFID module 讀 clk_i,rst_i,instr_i,pc_i,stall_i,flush_i，如果 rst_i 或 flush_i 的話就歸零，如果 stall_i 的話就什麼都不動，都不是前面的情況的話就正常更新輸出的 PC 跟 instruction
- ID/EX
  PipelineRegIDEX module 讀 clock,reset,RegWrite,MemtoReg,MemRead,MemWrite,ALUOp,ALUSrc,RS1data,RS2data,imm,instrution 進來
  並根據 clock 和 reset 輸出除了 clock 和 reset 以外的 input
  如果 clock signal is positive，就正常輸出
  如果 reset signal is positive，就將所有 input 歸零輸出
- EX/MEM
  PipelineRegEXMEM module 讀 clock,reset,RegWrite,MemtoReg,MemRead,MemWrite,ALUResult,RS2data,RDaddr,
  並根據 clock 和 reset 輸出除了 clock 和 reset 以外的 input
  如果 clock signal is positive，就正常輸出
  如果 reset signal is positive，就將所有 input 歸零輸出
- MEM/WB
  PipelineRegEXMEM module 讀 clock,reset,RegWrite,MemtoReg,ALUResult,Memdata,RDaddr,
  並根據 clock 和 reset 輸出除了 clock 和 reset 以外的 input
  如果 clock signal is positive，就正常輸出
  如果 reset signal is positive，就將所有 input 歸零輸出

Forwarding

- Forwarding Unit
  Forwarding_Unit module 從 PipelineRegIDEX 讀取 rs1、rs2，並從 PipelineRegEXMEM 及 PipelineRegMEMWB 讀取 rd、RegWrite 以判斷要以哪一個值作爲 ALU 的輸入，並傳送相應的 selection signal 給分別對應到 2 個 ALU operands 的 multiplexers (MUX_Forward_A, MUL_Forward_B)。對於 2 個 ALU 的 operand，判斷的邏輯都是一樣的，因此以下我僅就第一個 operand 做說明。
  若 Forwarding Unit 偵測到 rs1 有 EX hazard，就需要把 ALU 的第一個 operand 改成 EX/MEM 的 ALUResult，對應的 selection signal 爲 10。若沒有 EX hazard，才需要去檢查 rs1 有沒有 MEM Hazard。若有 MEM Hazard，就需要把 ALU 的第一個 opearnd 改成

MEM/WB 中要寫回 register 的值 (MUX_RegWriteSrc 的 output)，對應的 selection signal 爲 01。若均沒有 Data Hazard，代表不需要 forwarding，則將 operand 設爲上一個 stage 從 register 中讀出來的值，對應的 selection signal 爲 00。
不論是 EX Hazard 或 MEM Hazard，發生的邏輯都是一樣的，只是發生的 stage 不同。他們均要滿足3個條件:

1. 後面的 stage 要讀取的 register 等於前面的 stage 要寫入的 register (rs1 == rd)
2. 前面 stage 的 instruction 要會對 register 寫入 (RegWrite == 1)
3. 前面 stage 要寫入的 register 不能是 x0 (rd != 0)

- ForwardA， ForwardB
  即在 Forwarding Unit 中提到的負責控制 ALU operands 的 multiplexers。2者規格相同，均根據 2-bit 的 selection signal 從 4 個 32-bit 的輸入中選擇一個作爲 output。在這個 project 的 forwarding 中只會使用到其中 3 個: 00, 01, 10。其對應的輸入請見 Forwarding Unit 的說明。

Hazard Detection

- Hazard_Detection
  Hazard_Detection module 依照 data path 讀取 pipeline register 的輸出，如果 MEM 的部份要執行 load 指令而且存放的 register 跟即將要使用的 register 一樣的話，Stall_o 就設成 1；而 NoOp_o 跟 Stall_o 是等價的，所以就直接 assign；PCWrite_o 則是跟 Stall_o 相反，所以就 negate 之後 assign。

testbench

- testbench
  TestBench module 我們沒有做任何的改動，因為所有 register 的初始化都應該要在 rst_i 的 signal 處理好。

Others

- BEQ_Detection
  BEQ_Detection moudle 是把 data path 裡面在 register 後面的等於和 Branch_o signal 的 AND 給包起來，讓程式更好維護。輸入是 Control.Branch_o,Registers.RS1data_o,Registers.RS2data_o。
- Sign_Extend
  ImmGen module 的 input 被我們改成整個 instruction，因為不同 type 的 instruction 的 imm field 不同，所以需要整個 instruction 來知道他的 imm field 長怎樣。
- MUX
  MUX32_4 module 是擴充了 multiplexer，讓其可以接收四個 input。

CPU

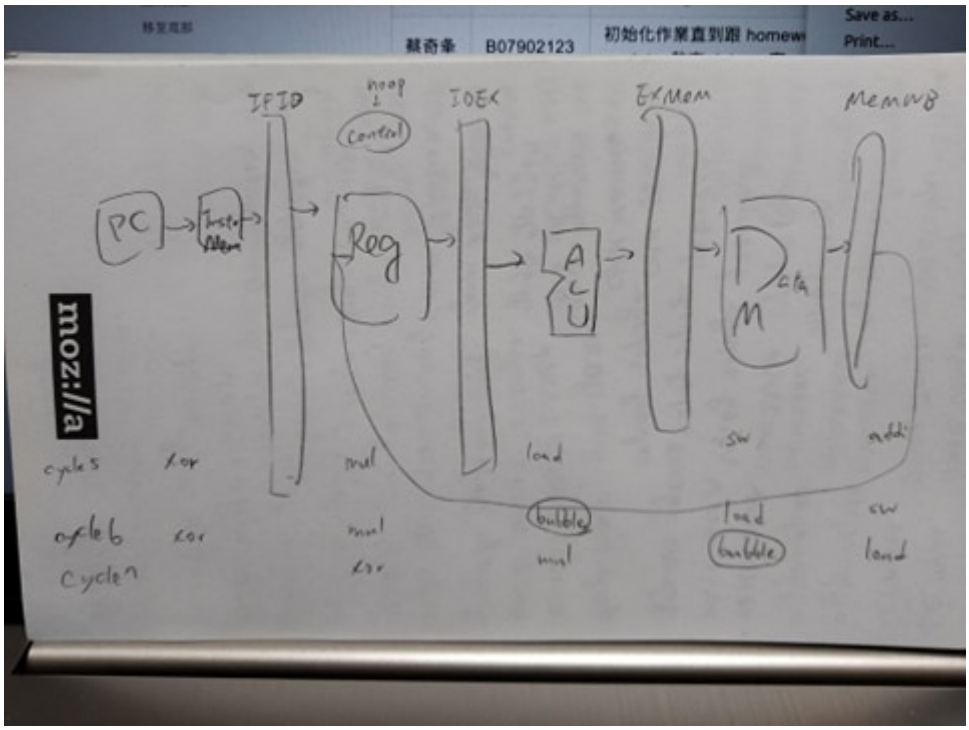
- CPU
  新增了一個 wire Flush 為了讓 testbench 的輸出使用。把所有新 module 依照 data path 的圖好好接好而已。

Members & Teamwork

姓名	學號	工作
徐維謙	B07902001	Pipeline Register, 寫 report
林楷恩	B07902075	Forwarding, 微調其他 module, 熬夜 debug, 寫 report
蔡奇夆	B07902123	初始化作業直到跟 homework 4 一樣, Hazard detection, 微調其他 module, 熬夜 debug, 寫 report

Difficulties Encountered and Solutions in This Project

1. 因爲有 pipeline, flush, stall，所以很難分辨現在這個 cycle 的各個 stage 裏是什麼 instruction。Solution: 把 instruction 在各個 stage 間的移動及對應的 signal value 畫在紙上。



2. 最後合併 forwarding 跟 hazard detection 的時候一直有很多小問題，其中一個是 beq 指令沒有用，開 gtkwave 來 trace 各 module 的值很久，才發現 PC.pc_i 沒接好（沒接到新的 PC_Source module 上）。總之大部分的小問題都是 data path 漏接。

Development Environment

- OS: Ubuntu 20.04
- Compiler: `iverilog 10.3 (stable)`