Computer Architecture Project 1

Group：balloon

組員：資工三 李建霖 資工三 邱榆洋 資工三 許耀文

Module Explanation

我們將 Data path 中的各個元件當成一個 Object，每個 File 都只包含一個 Object，以下以各個 Object 分別做說明：

1. Adder

將兩個 input 做相加，結果以 output 輸出，用於 PC + 4 及 Reducing the Delay of Branches

2. ALU_Control

與上次作業相同，另外加上 load (I-type)、sw (S-type)、beq (SB-type)，都是先利用來自 Control 的 ALUOp_i 判斷屬於哪種 type 的 instruction，然後再利用 {Ins[31:25, 14:12]} 判斷需要使用何種 Operation，接著傳給 ALU

3. ALU

根據 ALUCtrl_i 決定用何種 Operation。

4. And_Gate

用以判斷 Rs1 與 Rs2 是否相等且當前是否為 Branch 的指令，若前述條件成立，即 flush 發生，將是否要 flush 的判斷傳到 IF_ID

5. CPU

將元件相互連接，並接收 clk, rst, start 等 input

6. Equal

判斷 Rs1 與 Rs2 是否相等，將結果傳至 And_Gate

7. Forwarding

藉由 RegWrite 及 Register Destination 來判斷現在這個 Cycle 需不需要 Forwarding，以及是 Mem Hazard 或是 EX Hazard，條件如 spec 所示

8. Hazard _Detection

判斷 stall 的條件是否成立，即

```
If (ID/EX.MemRead and((ID/EX.RegisterRd = IF/ID.RegisterRs1) or(ID/EX.RegisterRd =
IF/ID.REgisterRs2)))
```

若成立，傳送訊號至 PC, IF/ID, Control

9. Control

藉由 instruction [6:0] 分辨要傳送何種 Option 至各元件，如 ALU Control 、Data Memory 等；若收到 NoOp 的信號，表示下一個需要 stall，則各 Option 接設為 0，代表插入一個 bubble。

10. MUX32

選擇兩個 32 bits 的 input 的其中一個，用於選擇 ALU source、PC source、Register Source

11. MUX4

選擇四個 2 bits 的 input 的其中一個，用於 Forwarding unit

12. Shift

將 input data 往左 Shift 一位，用於 Reducing the Delay of Branches 前的 PC 處理

13. Sign_Extend

根據不同 Type 的 instruction 的 immediate 部份的值都 extend 成 32 bits，即將最高位 bits (sign bit) 複製使的 immediate 長度為 32 bits

14. Testbench

初始化 pipeline 中的 register (都為零)，且紀錄 stall 及 flush 次數

15. IF_ID

實現 pipeline 所需的 Register，還需要判斷需要 stall 或是 flush，若是 stall，則不更新當前的 instruction，也不更新 PC；若是 flush，則將 IF_ID_o 設成 0

16. ID_EX

實現 pipeline 所需的 Register，以 clk 作為更新的時間，當 clk 從 0 變 1 時，將當前的值傳到下一個 pipeline register

17. MEM_WB

實現 pipeline 所需的 Register，以 clk 作為更新的時間，當 clk 從 0 變 1 時，將當前的值傳到下一個 pipeline register

18. EX_MEM

實現 pipeline 所需的 Register，以 clk 作為更新的時間，當 clk 從 0 變 1 時，將當前的值傳到下一個 pipeline register

Member Teamwork

組員	工作分配
邱榆洋	建立 lw, sw、製作forwarding、測試及debug
許耀文	建立 beq、wire 連接、建立 pipeline register、flush
李建霖	建立 hazard detection, stall、報告撰寫

Difficulties Encountered and Solutions

1. Missing No Operation Parameter in Control

一開始 Control 元件中的 always 條件如下：

```
always @(Op_i)
```

這代表只有 Op_i 更動時，才會更新 control 輸出的各種 option，可能導致

2. Missing i, input_00_i, input_01_i, input_10_i, input_11_i in MUX4

原 MUX4 中的 always 條件如下：

```
always@(select_i)
```

這很顯然是個錯誤，必須將所有 input 放到 always 條件中，如此一來才能確保 input 改變時，能輸出正確的值，修改後結果如下：

```
always@(select_i, input_00_i, input_01_i, input_10_i, input_11_i)
```

3. And Gate to Or Gate

我們一開始不小心將判斷 Branch 的 and_gate 寫成 or_gate

4. Signed Extension bugs

原先只有處理 I-Type 的 immediate，後來加上 S-Type 及 SB-Type 的時候，將最高位 (即 signed bits) 往高位複製時，複製的次數錯誤，例如：應複製 19次，但複製的 20 次

5. Other

1. 因為此次使用的元件相當的多，互相連接的 wire 也因此增加，容易接錯，接錯時要除錯也比較困難。
2. 因為是小組作業，一些變數名稱會不統一，線的接法也可能不一致

Development Environment

Linux Kernel：5.7.19-2-MANJARO

Operating System：MANJARO (Based on Archlinux)

iVerilog Version：11.0