VLSI System Design (Graduate Level)

Fall 2024

HOMEWORK I

REPORT

Must do self-checking before submission:

☑ Compress all files described in the problem into one tar

☑ All SystemVerilog files can be compiled under SoC Lab environment

☑ All port declarations comply with I/O port specifications

☑ Organize files according to File Hierarchy Requirement

☑ No any waveform files in deliverables

Student name: \_\_\_\_\_\_王彥珽\_\_\_\_\_

Student ID: \_\_\_\_\_P76131416\_\_\_\_\_

**Outline**

1. RISC-V pipeline CPU Overview
2. Summary
3. Lesson Learned
4. Instruction
5. R-type
6. I-type
7. S-type
8. B-type
9. U-type
10. J-type
11. F-type
12. CSR
13. Synthesis Result

1.Area Report

2.Timing Report

1. Simulation

1.syn0

2.syn1

3.syn2

4.syn3

5.syn4

6.syn5

7.syn6

1. Superlint

**I. RISC-V pipeline CPU Overview**

一張含有 文字, 圖表, 方案, 平行 的圖片

自動產生的描述IF and ID stages

**一張含有 文字, 圖表, 方案, 平行 的圖片

自動產生的描述**EXE、MEM and WB stages

一張含有 文字, 圖表, 方案, 平行 的圖片

自動產生的描述Total design

**II. Summary**

這次的作業實作了基於RISC-V指令集的pipeline CPU。不同於以往的Single Cycle CPU，pipeline CPU提升效能與縮短timing。這次的設計中包含有5個stages，分別為IF(Instruction Fetch)、ID(Instruction Decode)、EXE(Execute)、MEM(Memory Access)及WB(Write Back)。

設計過程會遇到2種Hazard，分別為Data Hazard以及Control Hazard。我使用Forward來解決一般的R-type Data Hazard；對於Load-Use Data Hazard我則使用Forward及Stall解決。Control Hazard的部分則是會在EXE的時候決定要跳的地址，同時傳回IF；並將IF及ID內錯誤的指令清除。

在CSR的部分，我在ID/EXE之間的pipeline register中計算cycle及instruction個數，再將當前的cycles及instructions個數傳到ALU中作取值。

對於浮點數指令的處理，我另外建立了一個浮點數Register File用來暫存浮點數以及建立運算浮點數的ALU；還有3個浮點數控制訊號，分別為f\_RegWrite(控制浮點數寫回Register File)、ALUSel\_f(選擇一般ALU或是浮點數ALU運算結果)及Memoryin\_f(選擇將一般資料或是浮點數存回DM)。

**III. Lesson Learned**

以下列出本次在實作RISV-V pipeline CPU實遇到的問題。

1. Data Hazard

i. 一般Data Hazard

新增一個Forwarding Unit即可解決。

一張含有 文字, 圖表, 方案, 平行 的圖片

自動產生的描述

(1)如上圖，可以透過檢查淺綠色的線(mem\_write\_addr、wb\_write\_addr)與exe\_rd\_r1\_addr、exe\_rd\_r2\_addr是否有相依，再將MEM與WB的資料送回EXE將錯誤及時更正。

ii. Load-use Data Hazard

新增一個Hazard Detection Unit，stall一個cycle，配合Forwarding Unit解決。

一張含有 文字, 圖表, 方案, 平行 的圖片

自動產生的描述

(1)如上圖，透過檢查ID stage中的rd\_r1, rd\_r2以及EXE stage中的rd\_addr, MemRead是否存在Load type；以及ID中的rd與EXE中的addr來判斷Registers是否有相關。

一張含有 文字, 圖表, 方案, 平行 的圖片

自動產生的描述

(2)若存在Load-use，則會將IF、ID stall一個cycle，並發出ID/EXE的Control\_flush訊號，使得下一個進入EXE的指令不會執行，並且Program Counter不會更新為下個指令。

(3)透過(2)作完stall後，我們可以透過Forwarding Unit將WB中Load取得的資料Forward回EXE作後續運算。

2. Control Hazard

因為Branch的條件達成，所以要將ID、IF中的指令Flush，避免錯誤發生。

一張含有 文字, 圖表, 方案, 平行 的圖片

自動產生的描述

(1)如上圖，透過傳入Hazard Detection Unit的exe\_branchCtrl(紅色線)訊號可以判斷需要branch時，發出IFID\_Flush(紅色線)與Control\_Flush(紅色線)將IF/ID reg與ID/EXE reg的指令清除，就可以執行錯誤指令。

3. Float

由於Float相關的指令加入，所以需要額外增加Float Register File儲存浮點數；f\_ALU作浮點數運算

一張含有 文字, 圖表, 方案, 行 的圖片

自動產生的描述

(1)上圖為一浮點數Register file，透過f\_RegWrite決定是否要寫入暫存器，後續的Forwarding Unit透過f\_RegWrite，決定是否要將浮點數的資料傳回前面stage使用。

一張含有 文字, 圖表, 行, 繪圖 的圖片

自動產生的描述

(2)上圖為浮點數運算的ALU，紫色的線為從MEM、WB Forward回來EXE使用的資料，透過Forwarding Unit決定是否存在相依；隨後會透過2to1 Mux決定要使用ALU或是Float ALU運算完的資料。

4. CSR instructions

透過CSR指令計算目前CYCLE與INSTRUCTION個數

一張含有 文字, 圖表, 行, 數字 的圖片

自動產生的描述一張含有 文字, 圖表, 行, 數字 的圖片

自動產生的描述

(1)在ID/EXE reg中計算當前的cycle與instruction個數；若是有branch則instructions減一；若是load-use data hazard則instructions不加不減；其餘情況則instructions加一。

(2)將當前的資料傳入ALU中，透過alu\_ctrl決定是否為CSR operation；若是則alu\_out為CSR結果。

**IV. Instructions**

下表是我的ALU control signal

**一張含有 文字, 螢幕擷取畫面, 功能表, 字型 的圖片

自動產生的描述**

1.R-type

1.1 ADD

一張含有 文字, 螢幕擷取畫面, 設計 的圖片

自動產生的描述

上圖為單一ADD運算

1. ID中的opcode = 7’b011\_0011、funct3 = 3’b000、funct7 = 7’b000\_0000，此為ADD instruction。
2. EXE中的ALU control = 5’b0\_0000是ADD control signal。

3. out的結果為in1 + in2。

1.2 SUB

一張含有 螢幕擷取畫面, 多媒體軟體, 行 的圖片

自動產生的描述

上圖為連續SUB運算

1. ID中的opcode = 7’b011\_0011、funct3 = 3’b000、funct7 = 7’b010\_0000，此為SUB instruction。
2. EXE中的ALU control = 5’b0\_0001為SUB control signal。
3. out的結果為in1 - in2。

一張含有 螢幕擷取畫面, 行, 多媒體軟體 的圖片

自動產生的描述1.3 SLL

上圖為一SLL運算

1. ID中的opcode = 7’b011\_0011、funct3 = 3’b001、funct7 = 7’b000\_0000，為SLL instructoin。
2. EXE中的ALU control = 5’b0\_0010為SLL control signal。
3. out的結果為in1 << in2[4:0]。

1.4 SLT

一張含有 螢幕擷取畫面, 多媒體軟體, 軟體 的圖片

自動產生的描述

上圖為一SLT運算

1. ID中的opcode = 7’b011\_0011、funct3 = 3’b010、funct7 = 7’b000\_0000，為SLT instruction。
2. EXE中的ALU control = 5’b0\_0011為SLT control signal。
3. out的結果為sign(in1)<sign(in2)輸出32’h1。

1.5 SLTU

**一張含有 螢幕擷取畫面, 多媒體軟體, 軟體 的圖片

自動產生的描述**

上圖為一SLTU運算

1. ID中的opcode = 7’b011\_0011、funct3 = 3’b011、funct7 = 7’b000\_0000，為SLTU instruction。
2. EXE中的ALU control = 5’b0\_0100為SLTU control signal。
3. out的結果為unsign(in1)>unsign(in2)輸出32’h0。

一張含有 螢幕擷取畫面, 行, 多媒體軟體 的圖片

自動產生的描述1.6 XOR

上圖為一XOR運算

1. ID中的opcode = 7’b011\_0011、funct3 = 3’b100、funct7 = 7’b000\_0000，為XOR instruction。
2. EXE中的ALU control = 5’b0\_0101，為XOR control signal。
3. out的結果為in1^in2。

一張含有 螢幕擷取畫面, 多媒體軟體 的圖片

自動產生的描述1.7 SRL

上圖為一SRL運算

1. ID中的opcode = 7’b011\_0011、funct3 = 3’b101、funct7 = 7’b000\_0000，為SRL instruction。
2. EXE中的ALU control = 5’b0\_0110，為SRL control signal。
3. out的結果為unsign(in1) >> in2[4:0]。

一張含有 螢幕擷取畫面, 行, 鮮豔, 多媒體軟體 的圖片

自動產生的描述1.8 SRA

1.9 OR

上圖為一SRA運算

1. ID中的opcode = 7’b011\_0011、funct3 = 3’b101、funct7 = 7’b010\_0000，為SRA instruction。
2. EXE中的ALU control = 5’b0\_0111，為SRA control signal。
3. out的結果為sign(in1) >> in2[4:0]。

一張含有 螢幕擷取畫面, 行 的圖片

自動產生的描述1.9 OR

上圖為一OR運算

1. ID中的opcode = 7’b011\_0011，funct3 = 3’b110、funct7 = 7’b000\_0000，為OR instruction。
2. EXE中的ALU control = 5’b0\_1000，為OR control signal。
3. Out的結果為in1 | in2。

一張含有 螢幕擷取畫面, 多媒體軟體, 軟體, 繪圖軟體 的圖片

自動產生的描述1.10 AND

上圖為一AND運算

1. ID中的opcode = 7’b011\_0011、funct3 = 3’b111、funct7 = 7’b000\_0000，為AND instruction。
2. EXE中的ALU control = 5’b0\_1001，為AND control signal。
3. Out的結果為in1 & in2。

1.11 MUL

一張含有 螢幕擷取畫面, 多媒體軟體, 軟體, 繪圖軟體 的圖片

自動產生的描述

上圖為一MUL運算

1. ID中的opcode = 7’b011\_0011、funct3 = 3’b000、funct7 = 7’b000\_0001，為MUL instruction。
2. EXE中的ALU control = 5’b1\_0001，為MUL control signal。
3. Out的結果為in1\*in2取lower 32bits。

1.12 MULH

一張含有 螢幕擷取畫面, 多媒體軟體, 軟體 的圖片

自動產生的描述

上圖為一MULH運算

1. ID中的opcode = 7’b011\_0011、funct3 = 3’b001、funct7 = 7’b000\_0001，為MULH instruction。
2. EXE中的AUL control = 5’b1\_0010，為MULH control signal。
3. out的結果為sign(in1)\*sign(in2)取upper 32bits。

1.13 MULHSU

一張含有 螢幕擷取畫面, 多媒體軟體, 軟體 的圖片

自動產生的描述

上圖為一MULHSU運算

1. ID中的opcode = 7’b011\_0011、funct3 = 3’b010、funct7 = 7’b000\_0001，為MULHSU instruction。
2. EXE中的ALU control = 5’b1\_0011，為MULHSU control signal。
3. out的結果為sign(in1)\*unsign(in2)取upper 32bits。

1.14 MULHU

一張含有 螢幕擷取畫面, 多媒體軟體, 軟體, 繪圖軟體 的圖片

自動產生的描述

上圖為一MULHU運算

1. ID中的opcode = 7’b011\_0011、funct3 = 3’b011、funct7 = 000\_0001，為MULHU instruction。
2. EXE中的ALU control = 5’b1\_0100，為MULHU control signal。
3. out的結果為unsign(in1)\*unsign(in2)取upper 32bits。

2.I-type

* 1. LW

一張含有 螢幕擷取畫面, 電子產品, 文字 的圖片

自動產生的描述

上圖為一LW指令

1. ID中的opcode = 7’b000\_0011、funxt3 = 3’b010為LW instruction。
2. EXE中的ALU control = 5’b0\_0000，為 ADD。
3. MEM從DM中取出來的Data = 32’h6666\_6666。
4. WB的RegWrite為high，此時要寫回的data為32’h6666\_6666。
5. Register file中要寫入的值為32’h6666\_6666。

2.2 ADDI

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

上圖為一ADDI指令

1. ID中的opcode = 7’b001\_0011，funct3 = 3’b000，為ADDI instruction。
2. EXE中ALU control = 5’b0\_0000，為ADD；in2 = immediate。
3. out的結果為in1 + in2 = in1 + imm。

2.3 SLTI

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

上圖為SLTI指令

1. ID中的opcode = 7’b001\_0011，funct3 = 3’b10，為SLTI instruction。
2. EXE中的ALU control = 5’b0\_0011，為SLT；in2 = immediate。
3. Out的結果為sign(in1) > sign(in2)輸出32’h0。

2.4 SLTIU

一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述

上圖為一SLTIU指令

1. ID中的opcode = 7’b001\_0011，funct3 = 3’b011，為SLTI instruction。
2. EXE中的ALU control = 5’b0\_0100，為SLT；in2 = immediate。
3. out的結果為unsign(in1) < unsign(in2) 輸出32’h1。

2.5 XORI

一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述

上圖為一XORI指令

1. ID中的opcode = 7’b001\_0011，funct3 = 3’b100，為XORI instruction。
2. EXE中的ALU control = 5’b0\_0101，為XOR；in2 = immediate。
3. Out的結果為in1 ^ in2 = in1 ^ imm。

2.6 ORI

一張含有 螢幕擷取畫面, 文字, 多媒體軟體, 軟體 的圖片

自動產生的描述

上圖為一ORI指令

1. ID中的opcode = 7’b001\_0011，funct3 = 3’b110，為ORI instruction。
2. EXE中的ALU control = 5’b0\_1000，為OR；in2 = immediate。
3. Out的結果為in1 | in2 = in1 | imm。

2.7 ANDI

一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述

上圖為一ANDI指令

1. ID中的opcode = 7’b001\_0011，funct3 = 3’b111，為ANDI instruction。
2. EXE中的ALU control = 5’b0\_1001，為AND；in2 = immediate。
3. Out的結果為in1 & in2 = in1 & imm。

2.8 LB

一張含有 螢幕擷取畫面, 文字, 軟體, 多媒體軟體 的圖片

自動產生的描述

上圖為一LB指令

1. ID中的opcode = 7’b000\_0011、funct3 = 3’b000，為LB instruction。
2. EXE中的ALU control = 5’b0\_0000，為ADD
3. MEM中從DM取出的data = 32’hcccc\_cccc。
4. WB中將data取1byte，即32’hffff\_ffcc；且RegWrite為high。
5. Register file要寫回的值為32’hffff\_ffcc。

2.9 SLLI

一張含有 螢幕擷取畫面, 文字, 多媒體軟體, 軟體 的圖片

自動產生的描述

上圖為一SLLI指令

1. ID中的opcode = 7’b001\_0011，funct3 = 3’b001、funxt7 = 7’b000\_0000，為SLLI instruction。
2. EXE中的ALU control = 5’b0\_0010，為SLL；in2 = immediate。
3. out的結果為unsign(in1) << in2[4:0] = unsign(in1) << imm[4:0]。

一張含有 螢幕擷取畫面, 行 的圖片

自動產生的描述2.10 SRLI

上圖為一SRLI指令

1. ID中的opcode = 7’b001\_0011，funct3 = 3’b101、funxt7 = 7’b000\_0000，為SRLI instruction。
2. EXE中的ALU control = 5’b0\_0110，為SRL；in2 = immediate。
3. out的結果為unsign(in1) >> in2[4:0] = unsign(in1) >> imm[4:0]。

一張含有 螢幕擷取畫面, 行 的圖片

自動產生的描述2.11 SRAI

上圖為一SRAI指令

1. ID中的opcode = 7’b001\_0011，funct3 = 3’b101、funxt7 = 7’b010\_0000，為SRLI instruction。
2. EXE中的ALU control = 5’b0\_0111，為SRA。
3. out的結果為sign(in1) >> in2[4:0] = sign(in1) >> imm[4:0]。

2.12 JALR

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

上圖為一JALR(rd = PC + 4)指令

1. ID中的opcode = 7’b110\_0111，funct3 = 3’b000，為JALR instruction。
2. EXE中從ID傳來的PC = 32’h7dc；訊號PCtoRegSrc為low，選擇pc+4 = 32’h7e0。
3. MEM寫回rd的data = 32’h7e0。
4. WB中的RegWrite為high；寫回rd的data = 32’h7e0。
5. Register file寫回rd的data = 32’h7e0。

一張含有 文字, 螢幕擷取畫面, 行 的圖片

自動產生的描述

上圖為一JALR(PC = imm + rs1)指令

1. ID中的opcode = 7’b110\_0111、funct3 = 3’b000，為JALR instruction。
2. EXE中的ALU control = 5’b0\_0000，為ADD。
3. ALU的結果為in1 + in2 = in1 + imm。
4. Branch Control Unit輸出branchCtrl = 2’b2。
5. Pc selection根據branchCtrl選擇 {(imm+rs1)[31:1],1’b0} = 32’h7f4。
6. Program Counter輸入保持32’h7f4。

2.13 LH

一張含有 螢幕擷取畫面, 文字, 軟體, 多媒體軟體 的圖片

自動產生的描述

上圖為一LH指令

1. ID中的opcode = 7’b000\_0011、funct3 = 3’b001，為LH instruction。
2. EXE中的ALU control = 5’b0\_0000，為ADD。
3. MEM中DM取出的data = 32’hcccc\_cccc。
4. WB中將data取half byte即32’ffff\_cccc；RegWrite為high。
5. Register file寫回的data = 32’hffff\_cccc。

2.14 LBU一張含有 螢幕擷取畫面, 文字, 軟體, 多媒體軟體 的圖片

自動產生的描述

上圖為一LBU指令

1. ID中的opcode = 7’b000\_0011、funct3 = 3’b100，為LBU instruction。
2. EXE中的ALU control = 5’b0\_0000，為ADD。
3. MEM中從DM取出來的data = 32’hcccc\_cccc。
4. WB中取unsigned byte即32’hcc；RegWrite為high。
5. Register file寫回的data = 32’hcc。

2.15 LHU

一張含有 螢幕擷取畫面, 文字, 軟體 的圖片

自動產生的描述

上圖為一LHU指令

1. ID中的opcode = 7’b000\_0011、funct3 = 3’b101，為LHU instruction。
2. EXE中的ALU control = 5’b0\_0000，為ADD。
3. MEM中從DM取出來的data = 32’hcccc\_cccc。
4. WB中取unsigned half word，即32’hcccc；RegWrite為high。
5. Register file寫回的data = 32’hcccc。

3.S-type

3.1 SW

一張含有 螢幕擷取畫面, 文字, 多媒體軟體, 軟體 的圖片

自動產生的描述

上圖為一SW指令

1. ID中的opcode = 7’b010\_0011、funct3 = 3’b010，為SW instruction。
2. EXE中的ALU control = 5’b0\_0000，為ADD。
3. out的結果為in1 + in2 = in1 + imm。
4. MEM中要寫的byte數為32bits，MemWrite = 32’h0000\_0000；寫入data = 32’hffff\_fff8。

3.2 SB

一張含有 螢幕擷取畫面, 文字, 軟體, 多媒體軟體 的圖片

自動產生的描述

上圖為一SB指令

1. ID中的opcode = 7’b010\_0011、funct3 = 3’b000，為SB instruction。
2. EXE中的ALU control = 5’b0\_0000，為ADD。
3. out的結果為in1 + in2 = in1 + imm。
4. MEM中要寫的byte數為8bits，MemWrite = 32’hffff\_ff00；寫入data = 32’h1234\_5678。

3.3 SH

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

上圖為一SH指令

1. ID中的opcode = 7’b010\_0011、funct3 = 3’b001，為SH instruction。
2. EXE中的ALU control = 5’b0\_0000，為ADD。
3. out的結果為in1 + in2 = in1 + imm。
4. MEM中要寫的byte數為16bits，MemWrite = 32’hffff\_0000；寫入data = 32’h1234\_5678。

4. B-type

4.1 BEQ

一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

上圖為一BEQ指令

1. ID中的opcode = 7’b110\_0011、funct3 = 3’b000，為BEQ instruction。
2. EXE中的ALU control = 5’b0\_1010，為BEQ。
3. ALU的結果為in1 == in2，zero\_flag輸出high。
4. Branch Control Unit輸出branchCtrl = 2’b1。
5. Pc selection根據branchCtrl選擇pc + imm = 32’h914。
6. Program Counter輸入保持32’h914。

4.2 BNE

一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述

上圖為一BNE指令

1. ID中的opcode = 7’b110\_0011、funct3 = 3’b001，為BNE instruction。
2. EXE中的ALU control = 5’b0\_1011，為BNE。
3. ALU的結果為in1 != in2，zero\_flag輸出high。
4. Branch Control Unit輸出branchCtrl = 2’b1。
5. Pc selection根據branchCtrl選擇pc + imm = 32’h974。
6. Program Counter輸入保持32’h974。

4.3 BLT

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

上圖為一BLT指令

1. ID中的opcode = 7’b110\_0011、funct3 = 3’b100，為BLT instruction。
2. EXE中的ALU control = 5’b0\_1100，為BLT。
3. ALU的結果為sign(in1) < sign(in2)，zero\_flag輸出high。
4. Branch Control Unit輸出branchCtrl = 2’b1。
5. Pc selection根據branchCtrl選擇pc + imm = 32’h9d8。
6. Program Counter輸入保持32’h9d8。

4.4 BGE

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

上圖為一BGE指令

1. ID中的opcode = 7’b110\_0011、funct3 = 3’b101，為BGE instruction。
2. EXE中的ALU control = 5’b0\_1101，為BGE。
3. ALU的結果為sign(in1) >= sign(in2)，zero\_flag輸出high。
4. Branch Control Unit輸出branchCtrl = 2’b1。
5. Pc selection根據branchCtrl選擇pc + imm = 32’ha68。
6. Program Counter輸入保持32’ha68。

4.5 BLTU

一張含有 螢幕擷取畫面, 文字, 軟體, 多媒體軟體 的圖片

自動產生的描述

上圖為一BLTU指令

1. ID中的opcode = 7’b110\_0011、funct3 = 3’b110，為BLTU instruction。
2. EXE中的ALU control = 5’b0\_1110，為BLTU。
3. ALU的結果為unsign(in1) < unsign(in2)，zero\_flag輸出high。
4. Branch Control Unit輸出branchCtrl = 2’b1。
5. Pc selection根據branchCtrl選擇pc + imm = 32’had0。
6. Program Counter輸入保持32’had0。

4.6 BGEU

一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述

上圖為一BGEU指令

1. ID中的opcode = 7’b110\_0011、funct3 = 3’b111，為BGEU instruction。
2. EXE中的ALU control = 5’b0\_1111，為BGEU。
3. ALU的結果為unsign(in1) >= unsign(in2)，zero\_flag輸出high。
4. Branch Control Unit輸出branchCtrl = 2’b1。
5. Pc selection根據branchCtrl選擇pc + imm = 32’hb60。
6. Program Counter輸入保持32’hb60。

5. U-type

5.1 AUIPC

一張含有 螢幕擷取畫面, 文字, 設計 的圖片

自動產生的描述

上圖為一AUIPC(rd = PC + imm)指令

1. ID中的opcode = 7’b001\_0111，為AUIPC instruction。
2. EXE中從ID傳來的PC = 32’h460；訊號PCtoRegSrc為high，選擇pc+imm = 32’h1460。
3. MEM寫回rd的data = 32’h1460。
4. WB中的RegWrite為high；寫回rd的data = 32’h1460。
5. Register file寫回rd的data = 32’h1460。

5.2 LUI

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

上圖為一LUI指令

1. ID中的opcode = 7’b011\_0111，為LUI instruction。
2. EXE中的ALU control = 5’b1\_0000，為LUI。
3. out的結果為imm。

6. J-type

6.1 JAL

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

上圖為一JAL(rd = PC + 4)指令

1. ID中的opcode = 7’b110\_1111，為JAL instruction。
2. EXE中從ID傳來的PC = 32’hbb4；訊號PCtoRegSrc為low，選擇pc+4 = 32’hbb8。
3. MEM寫回rd的data = 32’hbb8。
4. WB中的RegWrite為high；寫回rd的data = 32’hbb8。
5. Register file寫回rd的data = 32’hbb8。

一張含有 文字, 螢幕擷取畫面, 行 的圖片

自動產生的描述

上圖為一JAL(PC = PC + imm)指令

1. ID中的opcode = 7’b110\_1111，為JAL instruction。
2. Branch Control Unit輸出branchCtrl = 2’b1。
3. Pc selection根據branchCtrl選擇 pc+imm = 32’hbbc。
4. Program Counter輸入保持32’hbbc。

7. F-type

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述7.1 FLW

上圖為一FLW指令

1. ID中的opcode = 7’b000\_0111、funct3 = 3’b010為FLW instruction。
2. EXE中的ALU control = 5’b0\_0000，為 ADD。
3. MEM從DM中取出來的Data = 32’h40bb\_3276。
4. WB的f\_RegWrite為high，此時要寫回的data為32’h40bb\_3276。
5. Register file中要寫入的值為32’h40bb\_3276。

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述7.2 FSW

上圖為一FSW指令

1. ID中的opcode = 7’b010\_0111、funct3 = 3’b010，為FSW instruction。
2. EXE中的ALU control = 5’b0\_0000，為ADD。
3. EXE中float2 = 32’h40bb\_3276，要存入DM中。
4. out的結果為in1 + in2 = in1 + imm。
5. MEM中要寫的byte數為32bits，MemWrite = 32’h0000\_0000；寫入data = float2 = 32’h40bb\_3276。

7.3 FADD.S

一張含有 螢幕擷取畫面, 多媒體軟體, 繪圖軟體 的圖片

自動產生的描述

上圖為一FADD.S運算

1. ID中的opcode = 7’b101\_0011、funct3 = 3’b111、funct7 = 7’b000\_0000，此為FADD.S instruction。
2. EXE中的f\_ALU operand = 2’b00為FADD.S control signal。
3. float\_ans的結果為float1 + float2。。

7.4 FSUB.S

一張含有 螢幕擷取畫面, 文字, 多媒體軟體, 軟體 的圖片

自動產生的描述

上圖為連續FSUB.S運算

1. ID中的opcode = 7’b101\_0011、funct3 = 3’b111、funct7 = 7’b000\_0100，此為FSUB.S instruction。
2. EXE中的f\_ALU operand = 2’b01為FSUB.S control signal。
3. float\_ans的結果為float1 - float2。

8. CSR

8.1 RDINSTRETH

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

上圖為一RDINSTRETH指令

1. ID中的opcode = 7’b111\_0011、funct3 = 3’b010、EXE中的CSRimm = 12’b1100\_1000\_0010為RDINSTRETH instruction。
2. EXE中的ALU control = 5’b1\_0101，為RDINSTRETH。
3. out的結果為instr\_cnt[63:32]。

8.2 RDINSTRET

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

上圖為一RDINSTRET指令

1. ID中的opcode = 7’b111\_0011、funct3 = 3’b010、EXE中的CSRimm = 12’b1100\_0000\_0010為RDINSTRET instruction。
2. EXE中的ALU control = 5’b1\_0110，為RDINSTRETH。
3. out的結果為instr\_cnt[31:0]。

8.3 RDCYCLEH

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

上圖為一RDCYCLEH指令

1. ID中的opcode = 7’b111\_0011、funct3 = 3’b010、EXE中的CSRimm = 12’b1100\_1000\_0000為RDCYCLEH instruction。
2. EXE中的ALU control = 5’b1\_0111，為RDCYCLEH。
3. out的結果為instr\_cnt[63:32]。

8.4 RDCYCLE

一張含有 螢幕擷取畫面, 文字, 軟體 的圖片

自動產生的描述

上圖為一RDINSTRET指令

1. ID中的opcode = 7’b111\_0011、funct3 = 3’b010、EXE中的CSRimm = 12’b1100\_0000\_0000為RDCYCLE instruction。
2. EXE中的ALU control = 5’b1\_1000，為RDCYCLE。
3. out的結果為instr\_cnt[31:0]。

V. Synthesis result

一張含有 文字, 螢幕擷取畫面, 字型, 軟體 的圖片

自動產生的描述

1. Timing report

一張含有 文字, 螢幕擷取畫面, Rectangle, 針線 的圖片

自動產生的描述

2. Area report

一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

VI. Simulation

|  |  |  |  |
| --- | --- | --- | --- |
| rtl0 | pass | syn0 | pass |
| rtl1 | pass | syn1 | pass |
| rtl2 | pass | syn2 | pass |
| rtl3 | pass | syn3 | pass |
| rtl4 | pass | syn4 | pass |
| rtl5 | pass | syn5 | pass |
| rt16 | pass | syn6 | pass |

1. syn0(49個instructions)

一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述

2. syn1(bubble sort with ascending order)

一張含有 文字, 螢幕擷取畫面, 設計 的圖片

自動產生的描述

3. syn2

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

4. syn3(Greatest common divisor)

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

5. syn4

一張含有 文字, 螢幕擷取畫面, 軟體, 繪圖軟體 的圖片

自動產生的描述

6. syn5

一張含有 文字, 螢幕擷取畫面, 繪圖軟體, 字型 的圖片

自動產生的描述

7. syn6

一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述

VII. Superlint

wc -l src/\* include/\*



make superlint

一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

4 most frequently warnings in my code:

1. 值沒給完全

Ex: wire\_a = 1; --> wire\_a = 1’b1;

2. left shift時sign bit會消失

Ex: SH時data要存[31:16]位置

memory\_data = data << 8;

--> memory\_data = {data[15:0], 16’d0};

3. 2個不同bit數資料作運算

Ex:a為**8**bit，b為**16**bit，兩者相加存入**16**bit的c

c = a + b

--> c = {8’d0,a} + b;

4. case中已寫滿但多寫一個default

Ex:op為1bit signal

case(op)

1’b0 : -

1’b1 : -

default : -

endcase

-->case(op) 或 case(op)

1’b0 : - 1’b0 : -

1’b1 : - default : -

endcase endcase