VLSI System Design (Graduate Level)

Fall 2020

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: \_\_\_楊子毅\_\_\_

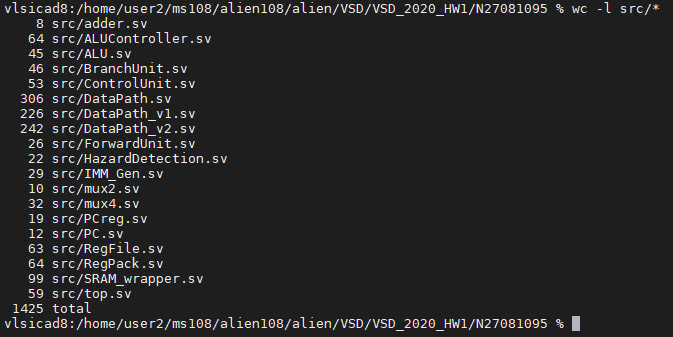
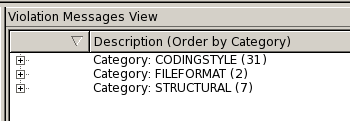
Student ID: \_\_\_N27081095\_\_\_

* Simulation

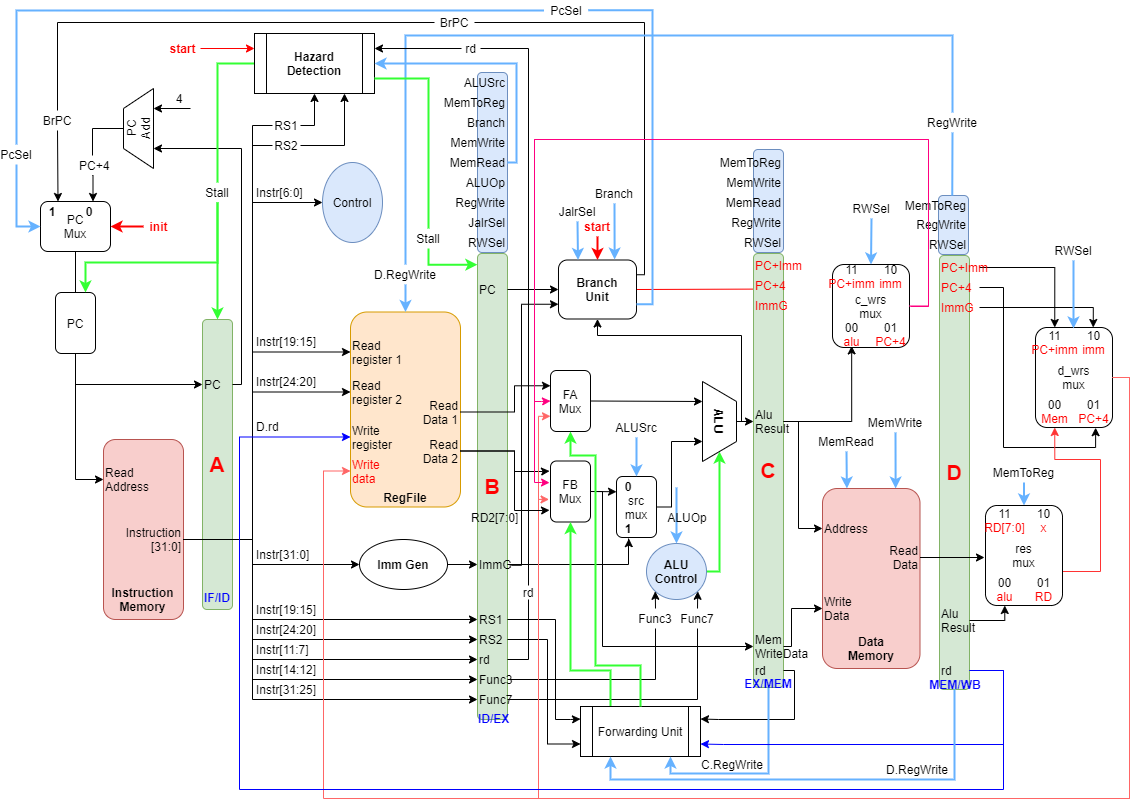
|  |  |
| --- | --- |
| Prog0 | Prog1 |
|  |  |
| Prog2 | Prog3 |
|  |  |

* Superlint

Total: 1425, Violation: 40, Percentage: 97.2%

* Block diagram



* Instruction

1. R-Type

|  |  |  |
| --- | --- | --- |
|  | ADD | AND |
| PC | 130 | 380 |
| Instruction | 0000 0000 0110 0010 1000 0010 1011 0011 (006282b3) | 0000 0000 0110 0010 1111 0010 1011 0011 (0062f2b3) |
| Control Unit | Opcode: 0110011 (R\_Type)  Func3: 000 (ADD)  RegWrite: 1  MemRead: 0  MemWrite: 0  ALUSrc: 0  MemToReg: 00 | Opcode: 0110011 (R\_Type)  Func3: 111 (AND)  RegWrite: 1  MemRead: 0  MemWrite: 0  ALUSrc: 0  MemToReg: 00 |
|  | ALU\_Src1: ffffffff  ALU\_Src2: ffffffff  ALU\_Operation: ADD  ALU\_Result: fffffffe  rd: 5 | ALU\_Src1: 12345678  ALU\_Src2: ffffffff  ALU\_Operation: AND  ALU\_Result: 12345678  rd: 5 |
|  | Store “ALU\_Result” and “rd” to next pipeline register  ALU\_Result: fffffffe  rd: 5 | Store “ALU\_Result” and “rd” to next pipeline register  ALU\_Result: 12345678  rd: 5 |
|  | Store “ALU\_Result” and “rd” to next pipeline register  ALU\_Result: fffffffe  rd: 5 | Store “ALU\_Result” and “rd” to next pipeline register  ALU\_Result: 12345678  rd: 5 |
|  | Write back to Reg\_file  RegWrite: 1  ALU\_Result: fffffffe  rd: 5 | Write back to Reg\_file  RegWrite: 1  ALU\_Result: 12345678  rd: 5 |
|  |  |  |

2. I-Type

|  |  |  |
| --- | --- | --- |
|  | LW | LB |
| PC | 3b8 | 430 |
| Instruction | 0000 0000 0000 0011 0010 0010 1000 0011 (00032283) | 1111 1111 0000 0100 0000 0011 0000 0011 (ff040303) |
| Control Unit | Opcode: 0000011 (I-Type)  Func3: 010 (LW)  RegWrite: 1  MemRead: 1  MemWrite: 0  ALUSrc: 1  MemToReg: 01 | Opcode: 0000011 (I-Type)  Func3: 000 (LB)  RegWrite: 1  MemRead: 1  MemWrite: 0  ALUSrc: 1  MemToReg: 11 |
| ALU\_Result: MemAdrs | ALU\_Src1: b34  ALU\_Src2: 0  ALU\_Operation: ADD  ALU\_Result: b34 | ALU\_Src1: 8038  ALU\_Src2: fffffff0  ALU\_Operation: ADD  ALU\_Result: 8028 |
|  | Store “ALU\_Result” and “rd” to next pipeline register  ALU\_Result: b34  rd: 5 | Store “ALU\_Result” and “rd” to next pipeline register  ALU\_Result: 8028  rd: 6 |
| DM\_ReadData | propagate adrs and read\_enable from pipeline registerC) to DM to read Data from DM.  DM\_Adrs = C.AluResult[15:2]  → 2cd = b34[15:2]    DM read data in next clk  DM\_ReadData = 66666666 | propagate adrs and read\_enable from pipeline register(C) to DM to read Data from DM.  DM\_Adrs = C.AluResult[15:2]  → 200a = 8028[15:2]    DM read data in next clk  DM\_ReadData = cccccccc |
| LW: load word  Store regfile data: DM\_ReadData[31:0]  LB: load word  Store regfile data: DM\_ReadData[7:0] | Write back to Reg\_file  RegWrite: 1  DM\_ReadData: 66666666  rd: 5 | Write back to Reg\_file  RegWrite: 1  DM\_ReadData[7:0]: cc  C.AluResult[1:0]: 00 → write to LSB  Reg\_file DATA\_W  = signed-ext(cc) = ffffffcc  rd: 6 |

3. S-Type

|  |  |  |
| --- | --- | --- |
|  | SW | SB |
| PC | 440 | 7ec |
| Instruction | 1111 1110 0110 0100 0010 1010 0010 0011(fe642a23) | 1111 1111 1110 0100 0000 1100 0010 0011(ffe40c23) |
| Control Unit | Opcode: 0100011 (S-Type)  Func3: 010 (SW)  RegWrite: 0  MemRead: 0  MemWrite: 1  ALUSrc: 1 | Opcode: 0100011 (S-Type)  Func3: 000 (SB)  RegWrite: 0  MemRead: 0  MemWrite: 1  ALUSrc: 1 |
| MemAdrs: ALU\_Result  Mem\_WriteData:  C.MemWriteData | ALU\_Src1: 8038  ALU\_Src2: fffffff4  ALU\_Operation: ADD  ALU\_Result: 802c | ALU\_Src1: 808c  ALU\_Src2: fffffff8  ALU\_Operation: ADD  ALU\_Result: 8084 |
|  | MemWriteData: rs2[31:0]  → MemWriteData = ffffffcc | MemWriteData: rs2[7:0]  → MemWriteData = 12345678[7:0]=78 |
|  | Store “ALU\_Result” and “MemWriteData” to next pipeline register  ALU\_Result: 802c  MemWriteData: ffffffcc | Store “ALU\_Result” and “MemWriteData” to next pipeline register  ALU\_Result: 8084  MemWriteData: 78 |
| DM\_WriteData | Propagate adrs and write\_enable from pipeline register(C) to DM to write Data.  DM\_Adrs = C.AluResult[15:2]  → DM\_Adrs = 200b = 802c[15:2]  DM\_WriteData = ffffffcc | Propagate adrs and write\_enable from pipeline register(C) to DM to write Data.  DM\_Adrs = C.AluResult[15:2]  → DM\_Adrs = 2021 = 8084[15:2]  DM\_WriteData = 78 |

4. B-Type

|  |  |  |
| --- | --- | --- |
|  | BEQ | BNE |
| PC | 840 | 880 |
| Instruction | 0000 0000 0101 0011 0000 1100 0110 0011 (00530c63) | 0000 0010 0110 0010 1001 1100 0110 0011 (02629c63) |
| Control Unit | Opcode: 1100011 (B-Type)  Func3: 000 (beq)  Branch: 1 | Opcode: 1100011 (B-Type)  Func3: 001 (bne)  Branch: 1 |
| Read data from reg\_file | RS1 = fffff000  RS2 = fffff000 | RS1 = fffff000  RS2 = 1000 |
| Forward | Forward\_A=2  Forward\_B=0 | Forward\_A=0  Forward\_B=2 |
| ALU | Forward data from EX/MEM reg to alu    ALU\_Operation: 4 (equal)  ALU\_Result: 0 | Forward data from EX/MEM reg to alu    Mux for ALU\_Src2    ALU\_Operation: 5 (not equal)  ALU\_Result: 0 |
| Branch | Branch: 1  BrPC: 0  PcSel: 0 (select pc=pc+4) | Branch: 1  BrPC: 0  PcSel: 0 (select pc=pc+4) |
| Next PC | Next PC = PC + 4 | Next PC = PC + 4 |
|  |  |  |

5. U-Type

|  |  |  |
| --- | --- | --- |
|  | AUIPC | LUI |
| PC | Ad0 | Af0 |
| Instruction | 1111 1111 1111 1111 1111 0011 0001 0111 (fffff317) | 0001 0011 0101 0111 1001 0011 1011 0111 (135793b7) |
| Control Unit | Opcode: 0010111 (auipc)  RegWrite: 1  MemRead: 0  MemWrite: 0  MemToReg: 00  RWSel: 11 | Opcode: 0110111 (lui)  RegWrite: 1  MemRead: 0  MemWrite: 0  MemToReg: 00  RWSel: 10 |
| Imm\_Generate | Calculate imm=fffff000 | Calculate imm=13579000 |
|  | Propagate rd to next pipeline register(B) | Propagate rd to next pipeline register(B) |
| Branch | Calculate “PC+4”、”ImmG”、”PC+imm” | Calculate “PC+4”、”ImmG”、”PC+imm” |
|  | Propagate “rd”、“PC+4”、”ImmG”、”PC+imm” to next pipeline register(C) | Propagate “rd”、“PC+4”、”ImmG”、”PC+imm” to next pipeline register(C) |
|  | Propagate “rd”、“PC+4”、”ImmG”、”PC+imm” to next pipeline register(D) | Propagate “rd”、“PC+4”、”ImmG”、”PC+imm” to next pipeline register(D) |
|  | Mux for write back data of reg\_file | Mux for write back data of reg\_file |
| Reg\_file | Write back to Reg\_file  Adrs: 6  Data: fffffad0 | Write back to Reg\_file  Adrs: 7  Data: 13579000 |

6. J-Type

|  |  |  |
| --- | --- | --- |
|  | JAL |  |
| PC | B08 |  |
| Instruction | 0000 0000 1000 0000 0000 0011 0110 1111 (0080036f) |  |
| Control Unit | Opcode: 1101111 (jal)  Branch: 1  RegWrite: 1  MemRead: 0  MemWrite: 0  MemToReg: 00  RWSel: 01 |  |
| Imm\_Generate | Imm: 8 |  |
|  | Propagate rd to next pipeline register(B) |  |
| Branch | Calculate “PC+4”、”ImmG”、”PC+imm”  PcSel: 1  BrPC: b10 |  |
| Next PC  PC=PC+imm | PcSel:1  BrPC: b10    Next clk  PC: b10 |  |
|  | Propagate “rd”、“PC+4”、”ImmG”、”PC+imm” to next pipeline register(C) |  |
|  | Propagate “rd”、“PC+4”、”ImmG”、”PC+imm” to next pipeline register(D) |  |
|  | Mux for write back data of reg\_file |  |
| Reg\_file  rd=PC+4 | Write back to Reg\_file  Adrs: 6  Data: b0c |  |

* Lessons learned

1. RISC-V CPU design and system-verilog usage.

2. Load/store bytes on instruction memory and data memory.

3. How to design a simple code with c-language to verification own RISC-V cpu.

4. Debug CPU by assembly code in file main.S and main.log.

* Summary

1. RISC