VLSI System Design (Graduate Level)

Fall 2021

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

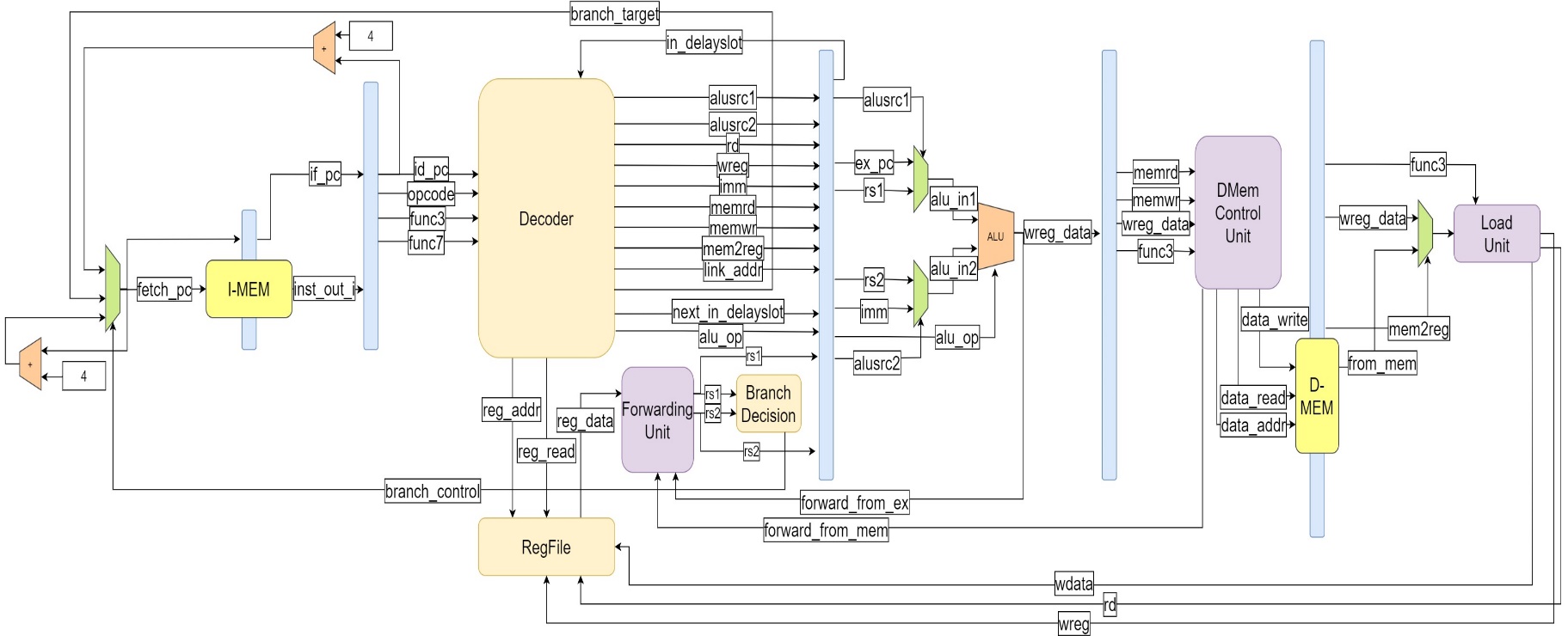
No any waveform files in deliverables

Student name: \_\_\_鄭惟\_\_\_\_\_\_\_\_

Student ID: \_\_\_P76091226\_\_\_\_\_\_\_

1. Summary:

In this assignment, the following goals have been completed.

1. Implement 5-stage pipeline RISC-V CPU with SystemVerilog.
2. Implement 33 RISC-V instructions.
3. Pass RTL simulation and gate-level simulation for ***prog0, prog1, prog2, prog3.***
4. Write the testing program in assembly for ***prog1, prog2, prog3.***
5. Use superlint to correct problems in my design.
6. Block diagram
7. Processor details
   1. **Single-issue** **in-order** **5-stage pipeline** with **full forwarding and hazard detection**.
   2. **Harvard architecture** with separate instruction and data ports.
   3. Implement **33 RV32I instructions**.
8. Highlights in this design

Unlike the traditional five-stage pipeline design from the textbook, this design has made the following changes:

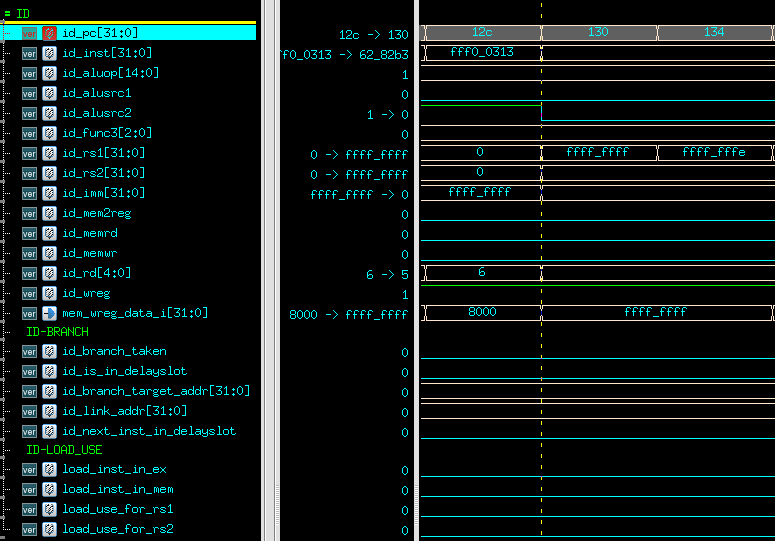
* 1. Add additional logic such that branch decisions are made in ID-stage. The branch delay slot for this processor is reduced to 2 cycles.
  2. Forwarding path forward results from EX, and ME stage back to ID-stage. This helps branch instructions to make decision in ID-stage.
  3. Registers file, instruction memory, and data memory are all triggered at positive clock edge. NO signals will be triggered at negative edge.
  4. Add a signal synchronizer to cope with the asynchronous reset signal.

1. Instruction behavior
   1. Behaviors in IF-stage

To avoid the mixing of posedge clk and negedge clk in the design, all sequential logics, instruction memory, and data memory will be triggered by posedge clk signal. Thus, it takes an additional cycle to read/write instruction/data memory. To comply with this design constraint, a new stage will be added in front of IF-stage, called prefetch stage, to send the address of next instruction ahead of IF-stage. This mechanism will cause an additional branch delayslot in case of branch misprediction (the branch penalty of this processor is 2 cycles).

Nonetheless, prefetch-stage and IF-stage are controlled by the same set of control signals; as a result, they will be viewed as stage pipeline stage.

* 1. Behaviors in ID-stage
     1. R-type instruction (add)



Given **id\_pc** = 130, the corresponding instruction is **add t0, t0, t1**. This instruction is decoded into the following signals:

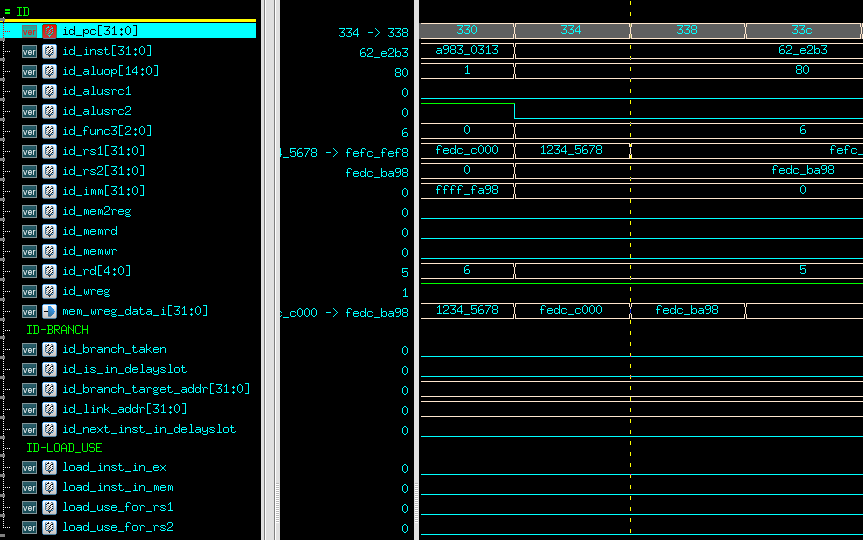
|  |  |
| --- | --- |
| **Signals** | **Value** |
| id\_aluop | 0x80 |
| id\_alusrc1 | 0x0 |
| id\_alusrc2 | 0x0 |
| id\_func3 | 0x6 |
| id\_rs1 | 0x1234\_5678 |
| id\_rs2 | 0xfedc\_ba98 |
| id\_imm | 0x0 |
| id\_mem2reg | 0x0 |
| id\_memrd | 0x0 |
| id\_memwr | 0x0 |
| id\_rd | 0x5 |
| id\_wreg | 0x1 |
| id\_branch\_taken | 0x0 |
| id\_is\_in\_delayslot | 0x0 |
| id\_next\_inst\_in\_delayslot | 0x0 |
| id\_link\_addr | 0x0 |
| id\_branch\_target\_addr | 0x0 |

**id\_aluop** is the signal that controls ALU operation in EX-stage. **id\_alusrc1**, and **id\_alusrc2** controls the input to ALU unit. **id\_rs1**, and **id\_rs2** is the latest register value forwarded from EX and ME-stage. **id\_imm** is the sign-extended immediate field. **id\_mem2reg** controls the MUX in WB-stage. The value from memory will be loaded into register if it equals to 1. **id\_memrd** is 1 if an instruction reads from memory. **id\_memwr** is 1 if an instruction writes to memory. **id\_rd** is the address of destination register. **id\_wreg** controls if data will be written back to registers.

Asides from these signals, there are branch-related, and load-use detection signals generated in ID-stage. **id\_branch\_taken** signal is 1 if a branch is taken. **id\_link\_addr** is required for jal and jalr instructions. Mentioned earlier, the branch delay slot for this design is 2 cycles. **id\_is\_in\_delayslot** and **id\_next\_inst\_in\_delayslot** signals indicates whether the current instruction in ID-stage is in branch delay slot.

Load-use hazards will be triggered if there’s a load instruction in EX or ME-stage. **load\_inst\_in\_ex** and **load\_inst\_in\_mem** indicates the position of a load instruction in the pipeline. **load\_use\_for\_rs1,** and **load\_use\_for\_rs2** indicate load use hazards in the pipeline.

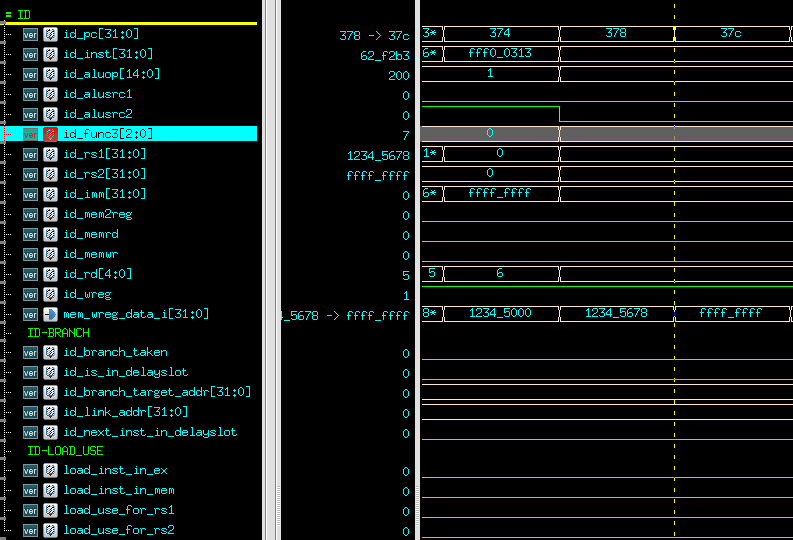
* + 1. R-type instruction (or)



Given **id\_pc** = 334, the corresponding instruction is **or t0, t0, t1**. This instruction is decoded into the following signals:

|  |  |
| --- | --- |
| **Signals** | **Value** |
| id\_aluop | 0x80 |
| id\_alusrc1 | 0x0 |
| id\_alusrc2 | 0x0 |
| id\_func3 | 0x6 |
| id\_rs1 | 0x1234\_5678 |
| id\_rs2 | 0xfedc\_ba98 |
| id\_imm | 0x0 |
| id\_mem2reg | 0x0 |
| id\_memrd | 0x0 |
| id\_memwr | 0x0 |
| id\_rd | 0x5 |
| id\_wreg | 0x1 |
| id\_branch\_taken | 0x0 |
| id\_is\_in\_delayslot | 0x0 |
| id\_next\_inst\_in\_delayslot | 0x0 |
| id\_link\_addr | 0x0 |
| id\_branch\_target\_addr | 0x0 |

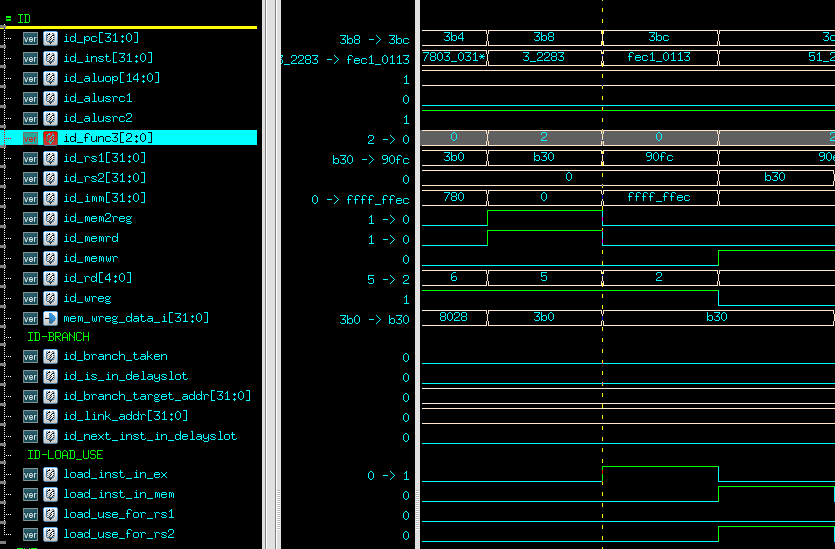
* + 1. R-type instruction (and)



Given **id \_pc** = 378, the corresponding instruction is **and t0, t0, t1**. This instruction is decoded into the following signals:

|  |  |
| --- | --- |
| **Signal** | **Value** |
| id\_aluop | 0x80 |
| id\_alusrc1 | 0x0 |
| id\_alusrc2 | 0x0 |
| id\_func3 | 0x7 |
| id\_rs1 | 0x1234\_5678 |
| id\_rs2 | 0xffff\_ffff |
| id\_imm | 0x0 |
| id\_mem2reg | 0x0 |
| id\_memrd | 0x0 |
| id\_memwr | 0x0 |
| id\_rd | 0x5 |
| id\_wreg | 0x1 |
| id\_branch\_taken | 0x0 |
| id\_is\_in\_delayslot | 0x0 |
| id\_next\_inst\_in\_delayslot | 0x0 |
| id\_link\_addr | 0x0 |
| id\_branch\_target\_addr | 0x0 |

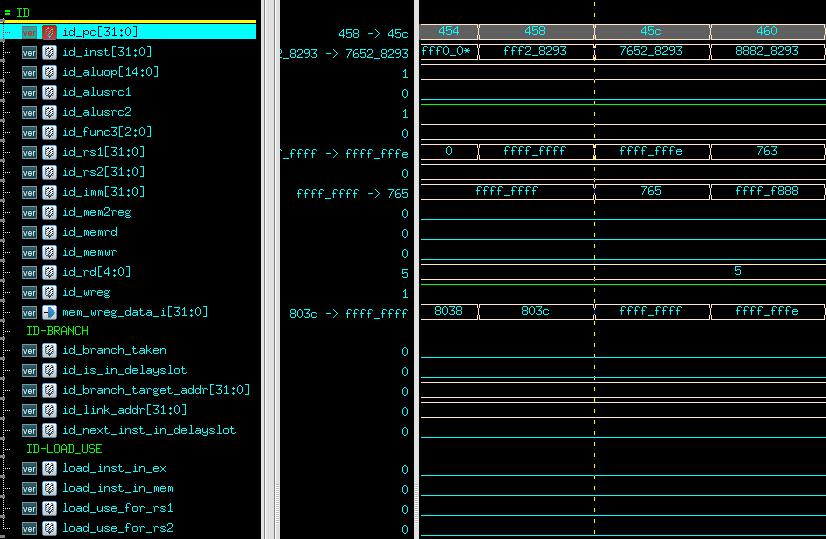
* + 1. I-type instruction (lw)



Given **id \_pc** = 3b8, the corresponding instruction is **lw t0, 0(t1)**. This instruction is decoded into the following signals:

|  |  |
| --- | --- |
| **Signal** | **Value** |
| id\_aluop | 0x1 |
| id\_alusrc1 | 0x0 |
| id\_alusrc2 | 0x1 |
| id\_func3 | 0x2 |
| id\_rs1 | 0x0b30 |
| id\_rs2 | 0x0 |
| id\_imm | 0x0 |
| id\_mem2reg | 0x1 |
| id\_memrd | 0x1 |
| id\_memwr | 0x0 |
| id\_rd | 0x5 |
| id\_wreg | 0x1 |
| id\_branch\_taken | 0x0 |
| id\_is\_in\_delayslot | 0x0 |
| id\_next\_inst\_in\_delayslot | 0x0 |
| id\_link\_addr | 0x0 |
| id\_branch\_target\_addr | 0x0 |

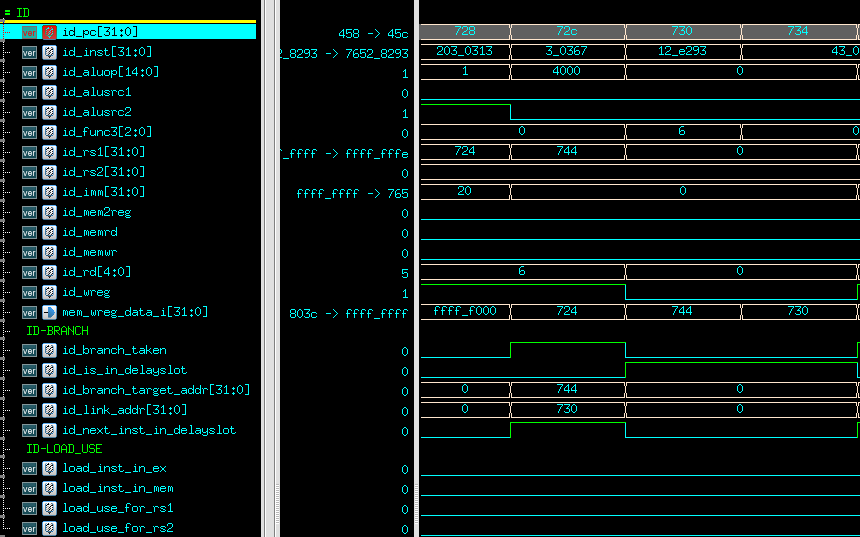
* + 1. I-type instruction (addi)



Given **id \_pc** = 458, the corresponding instruction is **addi t0, -1**. This instruction is decoded into the following signals:

|  |  |
| --- | --- |
| **Signal** | **Value** |
| id\_aluop | 0x1 |
| id\_alusrc1 | 0x0 |
| id\_alusrc2 | 0x1 |
| id\_func3 | 0x0 |
| id\_rs1 | 0xffff\_ffff |
| id\_rs2 | 0x0 |
| id\_imm | 0xffff\_ffff |
| id\_mem2reg | 0x0 |
| id\_memrd | 0x0 |
| id\_memwr | 0x0 |
| id\_rd | 0x5 |
| id\_wreg | 0x1 |
| id\_branch\_taken | 0x0 |
| id\_is\_in\_delayslot | 0x0 |
| id\_next\_inst\_in\_delayslot | 0x0 |
| id\_link\_addr | 0x0 |
| id\_branch\_target\_addr | 0x0 |

* + 1. I-type instruction (jalr)

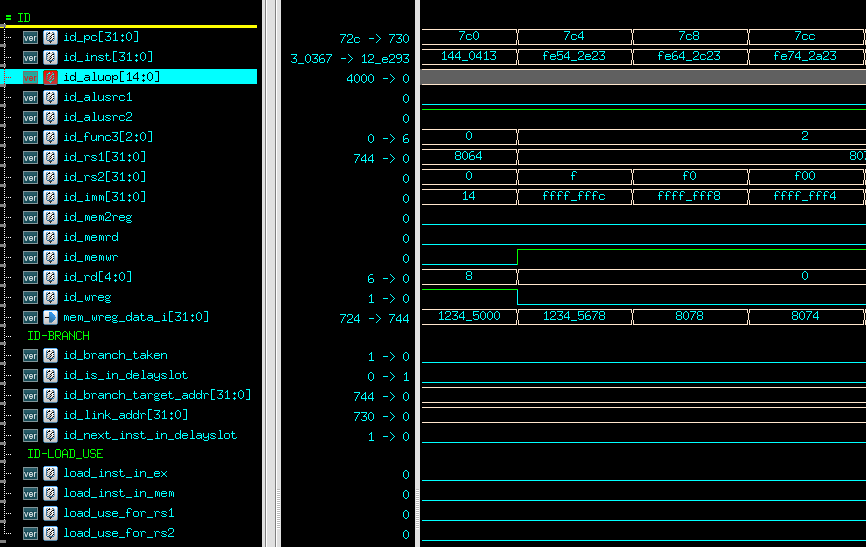


Given **id \_pc** = 72c, the corresponding instruction is **jalr t1, t1**. This instruction is decoded into the following signals:

|  |  |
| --- | --- |
| **Signal** | **Value** |
| id\_aluop | 0x4000 |
| id\_alusrc1 | 0x0 |
| id\_alusrc2 | 0x0 |
| id\_func3 | 0x0 |
| id\_rs1 | 0x744 |
| id\_rs2 | 0x0 |
| id\_imm | 0x0 |
| id\_mem2reg | 0x0 |
| id\_memrd | 0x0 |
| id\_memwr | 0x0 |
| id\_rd | 0x6 |
| id\_wreg | 0x1 |
| id\_branch\_taken | 0x1 |
| id\_is\_in\_delayslot | 0x0 |
| id\_next\_inst\_in\_delayslot | 0x1 |
| id\_link\_addr | 0x730 |
| id\_branch\_target\_addr | 0x744 |

Once this instruction is decoded, the processor immediately knows this branch must be taken. Thus, the next two instructions will fall in branch delayslot. The next PC will be set according to the computed branch target address.

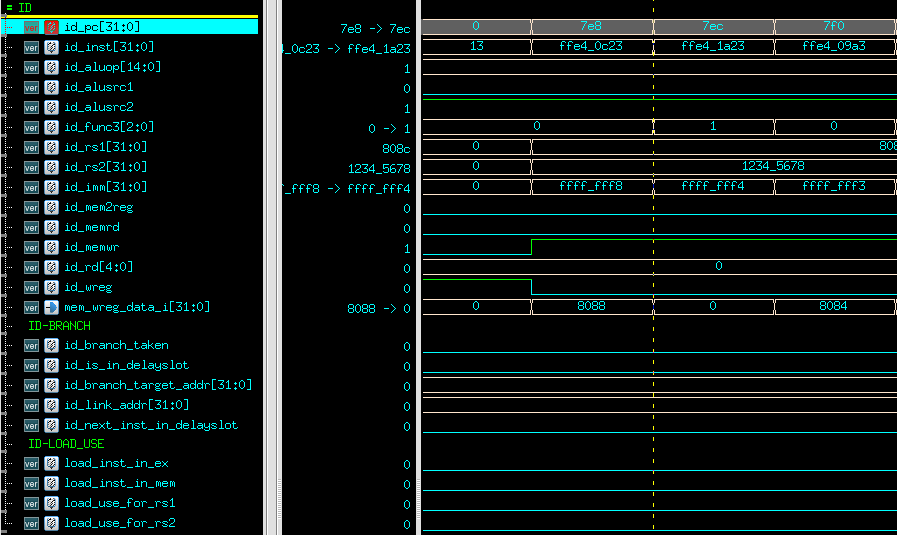
* + 1. S-type instruction (sw)



Given **id \_pc** = 7c4, the corresponding instruction is **sw t0, -4(s0)**. This instruction is decoded into the following signals:

|  |  |
| --- | --- |
| **Signal** | **Value** |
| id\_aluop | 0x1 |
| id\_alusrc1 | 0x0 |
| id\_alusrc2 | 0x1 |
| id\_func3 | 0x2 |
| id\_rs1 | 0x8078 |
| id\_rs2 | 0x000f |
| id\_imm | 0xffff\_fffc |
| id\_mem2reg | 0x0 |
| id\_memrd | 0x0 |
| id\_memwr | 0x1 |
| id\_rd | 0x0 |
| id\_wreg | 0x0 |
| id\_branch\_taken | 0x0 |
| id\_is\_in\_delayslot | 0x0 |
| id\_next\_inst\_in\_delayslot | 0x0 |
| id\_link\_addr | 0x0 |
| id\_branch\_target\_addr | 0x0 |

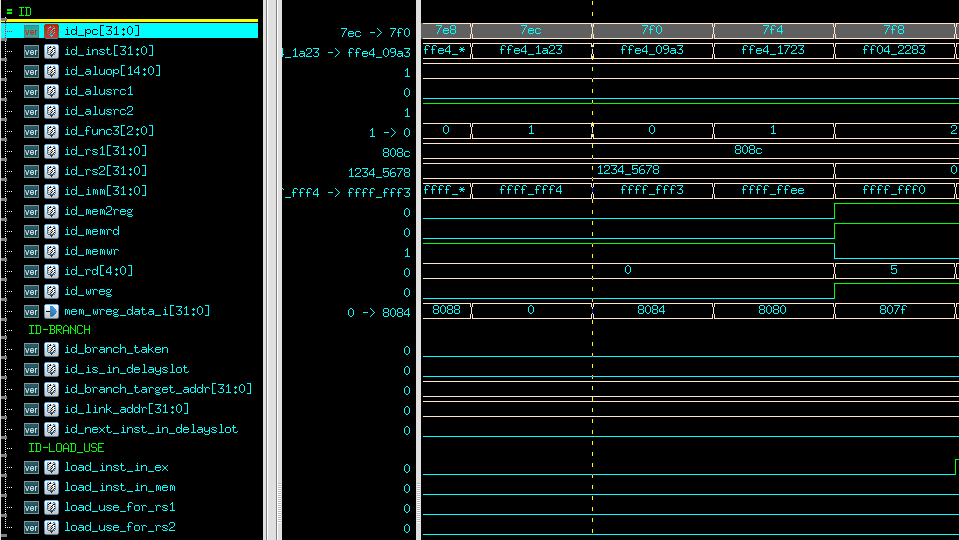
* + 1. S-type instruction (sb)



Given **id \_pc** = 7e8, the corresponding instruction is **sb t5, -8(s0)**. This instruction is decoded into the following signals:

|  |  |
| --- | --- |
| **Signal** | **Value** |
| id\_aluop | 0x1 |
| id\_alusrc1 | 0x0 |
| id\_alusrc2 | 0x1 |
| id\_func3 | 0x0 |
| id\_rs1 | 0x808c |
| id\_rs2 | 0x1234\_5678 |
| id\_imm | 0xffff\_fff8 |
| id\_mem2reg | 0x0 |
| id\_memrd | 0x0 |
| id\_memwr | 0x1 |
| id\_rd | 0x0 |
| id\_wreg | 0x0 |
| id\_branch\_taken | 0x0 |
| id\_is\_in\_delayslot | 0x0 |
| id\_next\_inst\_in\_delayslot | 0x0 |
| id\_link\_addr | 0x0 |
| id\_branch\_target\_addr | 0x0 |

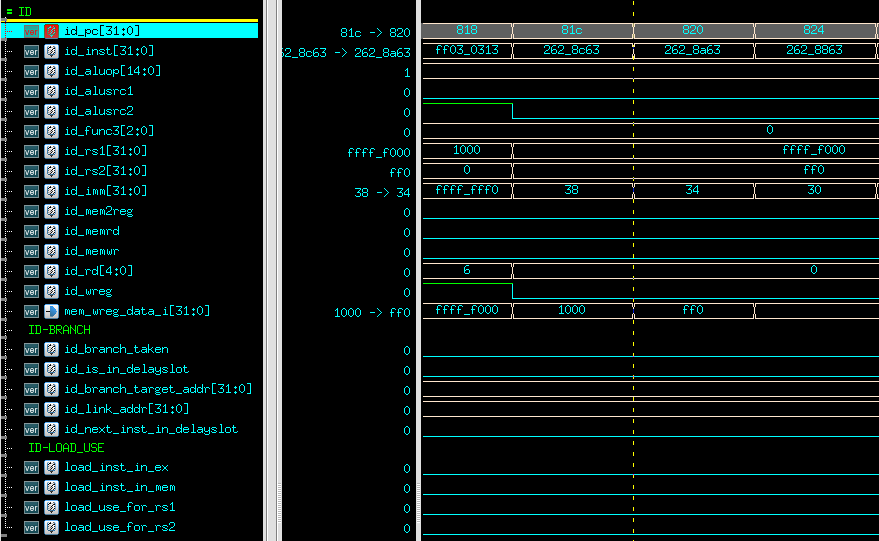
* + 1. S-type instruction (sh)



Given **id \_pc** = 7ec, the corresponding instruction is **sh t5, -12(s0)**. This instruction is decoded into the following signals:

|  |  |
| --- | --- |
| **Signal** | **Value** |
| id\_aluop | 0x1 |
| id\_alusrc1 | 0x0 |
| id\_alusrc2 | 0x1 |
| id\_func3 | 0x1 |
| id\_rs1 | 0x808c |
| id\_rs2 | 0x1234\_5678 |
| id\_imm | 0xffff\_fff4 |
| id\_mem2reg | 0x0 |
| id\_memrd | 0x0 |
| id\_memwr | 0x1 |
| id\_rd | 0x0 |
| id\_wreg | 0x0 |
| id\_branch\_taken | 0x0 |
| id\_is\_in\_delayslot | 0x0 |
| id\_next\_inst\_in\_delayslot | 0x0 |
| id\_link\_addr | 0x0 |
| id\_branch\_target\_addr | 0x0 |

* + 1. B-type instruction (beq)

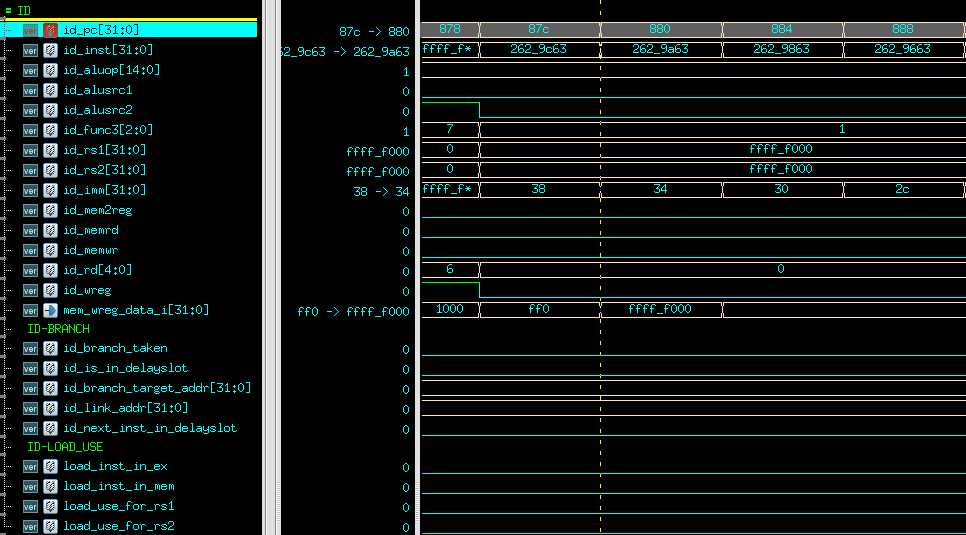


Given **id \_pc** = 81c, the corresponding instruction is **beq t0, t1, 854**. This instruction is decoded into the following signals:

|  |  |
| --- | --- |
| **Signal** | **Value** |
| id\_aluop | 0x1 |
| id\_alusrc1 | 0x0 |
| id\_alusrc2 | 0x0 |
| id\_func3 | 0x0 |
| id\_rs1 | 0xffff\_f000 |
| id\_rs2 | 0x0000\_0ff0 |
| id\_imm | 0x38 |
| id\_mem2reg | 0x0 |
| id\_memrd | 0x0 |
| id\_memwr | 0x0 |
| id\_rd | 0x0 |
| id\_wreg | 0x0 |
| id\_branch\_taken | 0x0 |
| id\_is\_in\_delayslot | 0x0 |
| id\_next\_inst\_in\_delayslot | 0x0 |
| id\_link\_addr | 0x0 |
| id\_branch\_target\_addr | 0x0 |

This processor immediately decided the branch condition is not true. Thus, the branch taken signal isn’t raised.

* + 1. B-type instruction (bne)

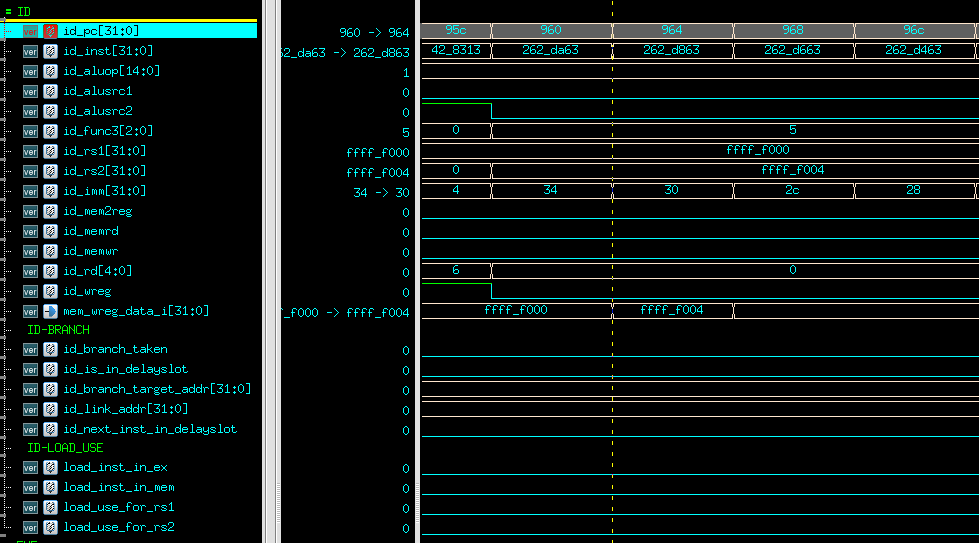


Given **id \_pc** = 87c, the corresponding instruction is **bne t0, t1, 8b4**. This instruction is decoded into the following signals:

|  |  |
| --- | --- |
| **Signal** | **Value** |
| id\_aluop | 0x1 |
| id\_alusrc1 | 0x0 |
| id\_alusrc2 | 0x0 |
| id\_func3 | 0x1 |
| id\_rs1 | 0xffff\_f000 |
| id\_rs2 | 0x ffff\_f000 |
| id\_imm | 0x38 |
| id\_mem2reg | 0x0 |
| id\_memrd | 0x0 |
| id\_memwr | 0x0 |
| id\_rd | 0x0 |
| id\_wreg | 0x0 |
| id\_branch\_taken | 0x0 |
| id\_is\_in\_delayslot | 0x0 |
| id\_next\_inst\_in\_delayslot | 0x0 |
| id\_link\_addr | 0x0 |
| id\_branch\_target\_addr | 0x0 |

This processor immediately decided the branch condition is not true. Thus, the branch taken signal isn’t raised.

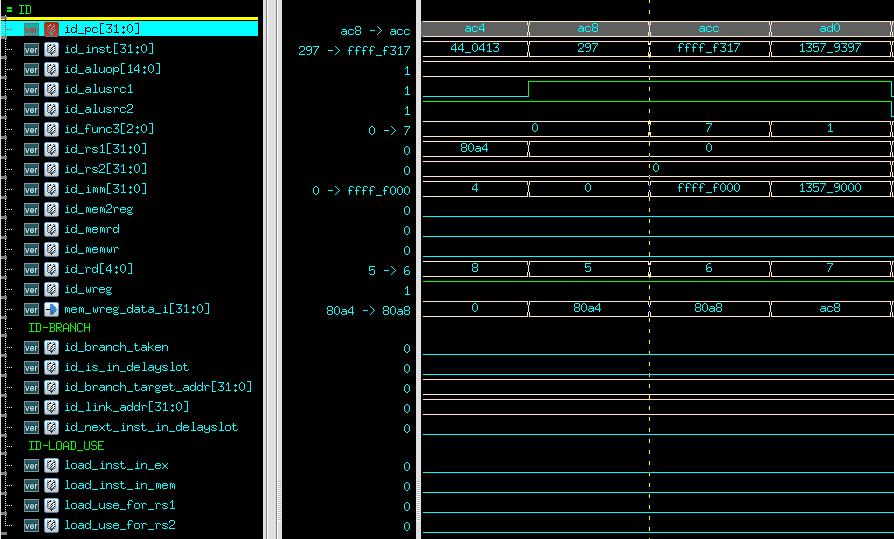
* + 1. B-type instruction (bge)



Given **id \_pc** = 87c, the corresponding instruction is **bge t0, t1, 994**. This instruction is decoded into the following signals:

|  |  |
| --- | --- |
| **Signal** | **Value** |
| id\_aluop | 0x1 |
| id\_alusrc1 | 0x0 |
| id\_alusrc2 | 0x0 |
| id\_func3 | 0x5 |
| id\_rs1 | 0xffff\_f000 |
| id\_rs2 | 0x ffff\_f004 |
| id\_imm | 0x34 |
| id\_mem2reg | 0x0 |
| id\_memrd | 0x0 |
| id\_memwr | 0x0 |
| id\_rd | 0x0 |
| id\_wreg | 0x0 |
| id\_branch\_taken | 0x0 |
| id\_is\_in\_delayslot | 0x0 |
| id\_next\_inst\_in\_delayslot | 0x0 |
| id\_link\_addr | 0x0 |
| id\_branch\_target\_addr | 0x0 |

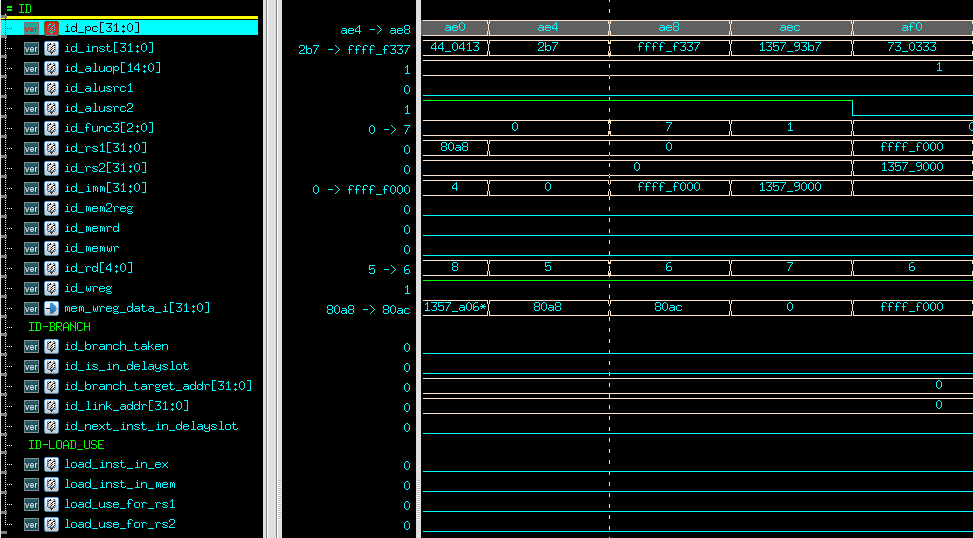
* + 1. U-type instruction (auipc)



Given **id \_pc** = ac8, the corresponding instruction is **auipc t0, 0x0**. This instruction is decoded into the following signals:

|  |  |
| --- | --- |
| **Signal** | **Value** |
| id\_aluop | 0x1 |
| id\_alusrc1 | 0x1 |
| id\_alusrc2 | 0x1 |
| id\_func3 | 0x0 |
| id\_rs1 | 0x0 |
| id\_rs2 | 0x0 |
| id\_imm | 0x0 |
| id\_mem2reg | 0x0 |
| id\_memrd | 0x0 |
| id\_memwr | 0x0 |
| id\_rd | 0x5 |
| id\_wreg | 0x1 |
| id\_branch\_taken | 0x0 |
| id\_is\_in\_delayslot | 0x0 |
| id\_next\_inst\_in\_delayslot | 0x0 |
| id\_link\_addr | 0x0 |
| id\_branch\_target\_addr | 0x0 |

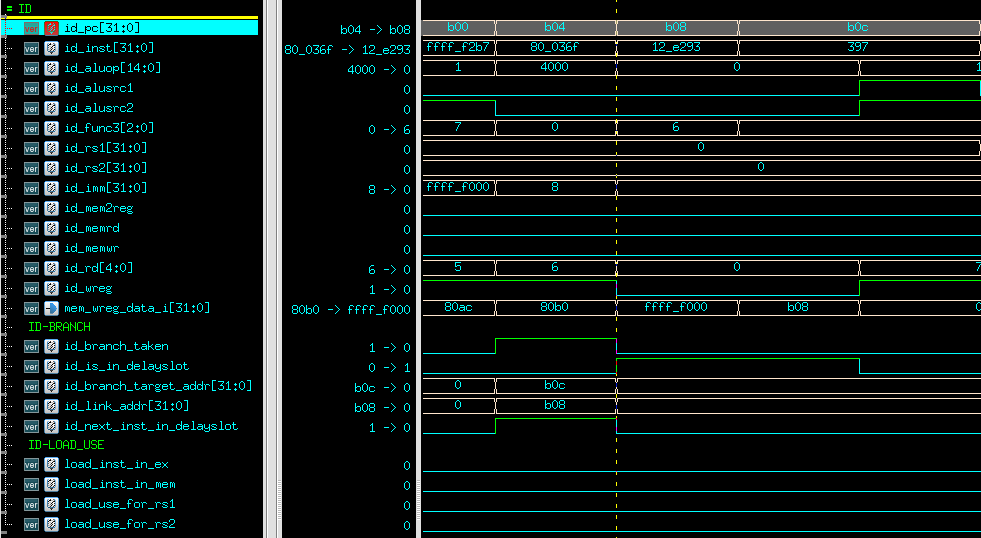
* + 1. U-type instruction (lui)



Given **id \_pc** = ae4, the corresponding instruction is **lui t0, 0x0**. This instruction is decoded into the following signals:

|  |  |
| --- | --- |
| **Signal** | **Value** |
| id\_aluop | 0x1 |
| id\_alusrc1 | 0x0 |
| id\_alusrc2 | 0x1 |
| id\_func3 | 0x0 |
| id\_rs1 | 0x0 |
| id\_rs2 | 0x0 |
| id\_imm | 0x0 |
| id\_mem2reg | 0x0 |
| id\_memrd | 0x0 |
| id\_memwr | 0x0 |
| id\_rd | 0x5 |
| id\_wreg | 0x1 |
| id\_branch\_taken | 0x0 |
| id\_is\_in\_delayslot | 0x0 |
| id\_next\_inst\_in\_delayslot | 0x0 |
| id\_link\_addr | 0x0 |
| id\_branch\_target\_addr | 0x0 |

* + 1. J-type instruction (jal)



|  |  |
| --- | --- |
| **Signal** | **Value** |
| id\_aluop | 0x4000 |
| id\_alusrc1 | 0x0 |
| id\_alusrc2 | 0x0 |
| id\_func3 | 0x0 |
| id\_rs1 | 0x0 |
| id\_rs2 | 0x0 |
| id\_imm | 0x8 |
| id\_mem2reg | 0x0 |
| id\_memrd | 0x0 |
| id\_memwr | 0x0 |
| id\_rd | 0x6 |
| id\_wreg | 0x1 |
| id\_branch\_taken | 0x1 |
| id\_is\_in\_delayslot | 0x0 |
| id\_next\_inst\_in\_delayslot | 0x0 |
| id\_link\_addr | 0x0xb08 |
| id\_branch\_target\_addr | 0x0 |

A **jal** instruction will be treated as an unconditional jump instruction. Thus, a branch will always be taken for jump.

* 1. Behaviors in EX-stage:

ALU unit in EX-stage performs arithmetic operations according to the control signals decoded in ID-stage. For R-type, arithmetic, U-type instructions, the ALU unit is used to calculate the value from registers. For load store instructions and J-type instructions, the ALU unit is used to calculate the address for either load/store target address or branch target address.

* 1. Behaviors in ME-stage:

If the instruction in ME-stage is a load/store instruction, ME-stage sets the control signals for data memory. If not, the latest value from EX-stage will be forwarded back to ID-stage through forwarding path.

* 1. Behaviors in WB-stage:

In WB-stage, the **mem2reg** signal controls the dataflow that eventually goes to the register file (switching between data from memory or data from ALU). For a load instruction, the data will be accessible in WB-stage because it takes one additional cycle to read/write memory.

1. Simulation results (prove of correctness)
   1. Prog0一張含有 文字 的圖片

      自動產生的描述
   2. Prog1一張含有 文字 的圖片

      自動產生的描述
   3. Prog2一張含有 文字 的圖片

      自動產生的描述
   4. Prog3一張含有 文字 的圖片

      自動產生的描述
2. Formal Verification
   1. Number of lines of RTL code: 1778
   2. Final result of superlint一張含有 桌 的圖片

      自動產生的描述

The STRUCTURAL category of violation is reduced to minimum to comply with Superlint.

1. Lesson learned

In this assignment, I learned how to interact with the RAM module properly and implement 33 RISCV instructions. Using the formal verification tool, the problems in my design have been identified and corrected.