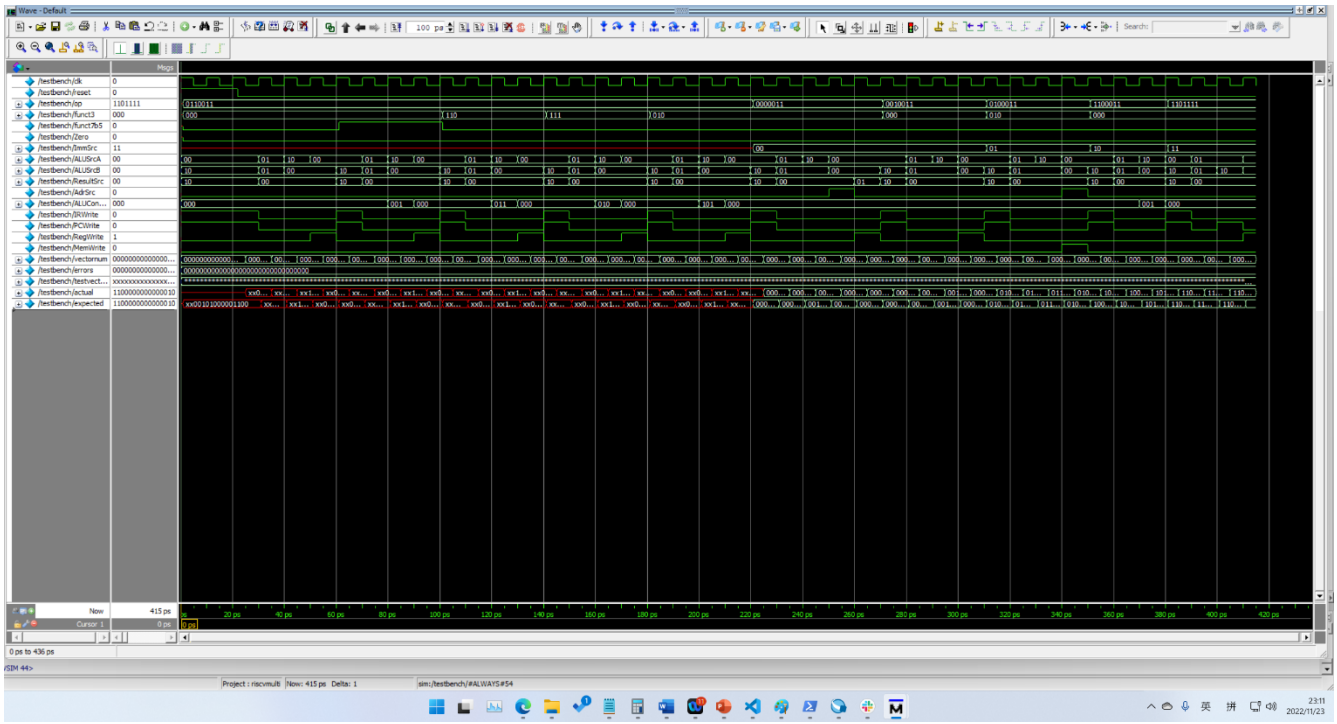
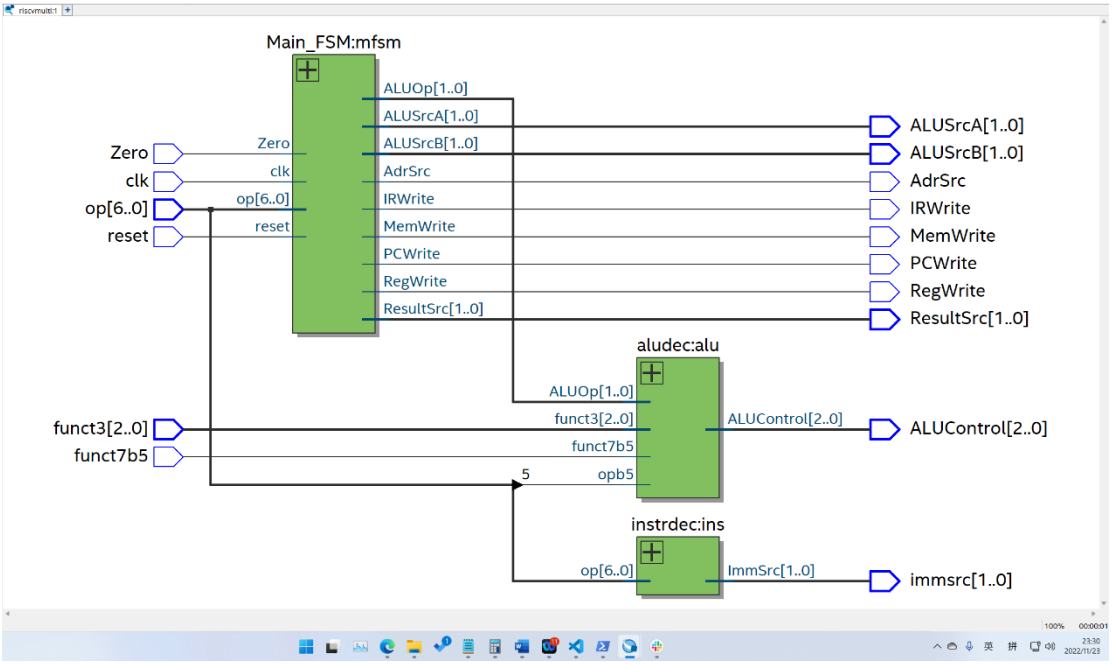
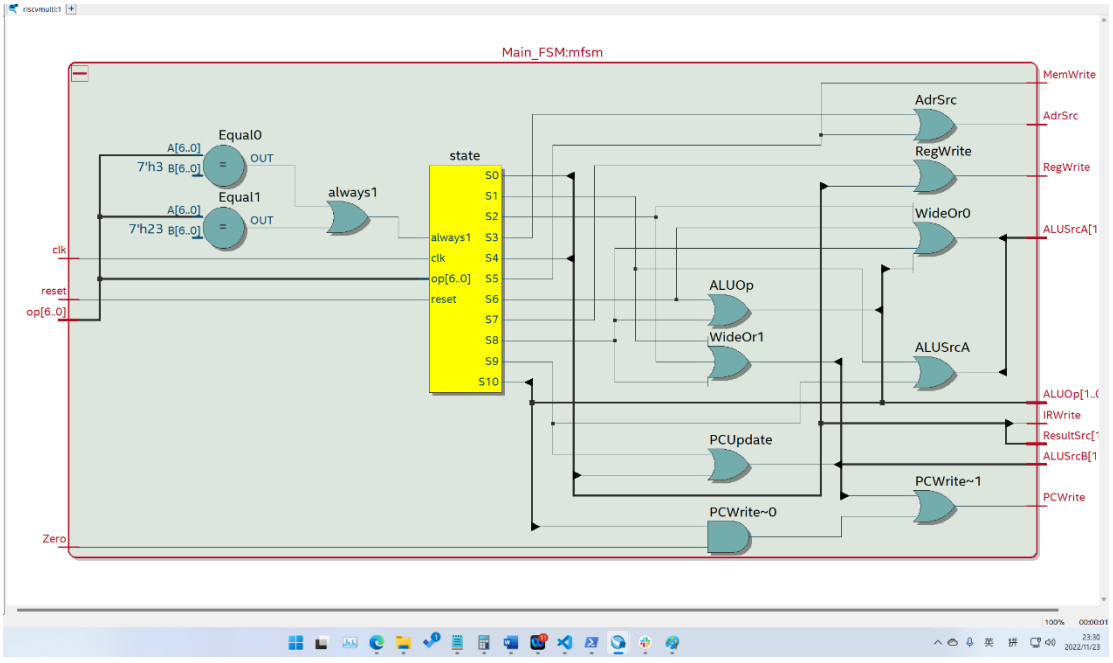
I spent 4h.

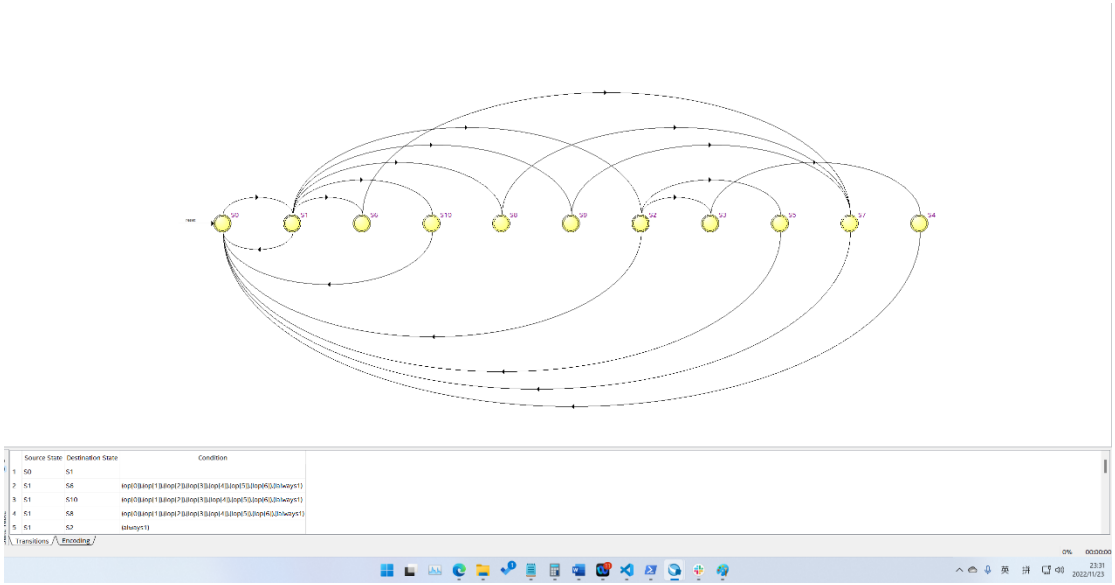
The FSM write enable signals (RegWrite, MemWrite, IRWrite, PCUpdate, and Branch) are 0 if not listed in a state. Other signals are don't care if not listed in a state. Then fetch the input, decode it and goes to memadr if it is lw. Then memread and memwb. Then PC+4 and go back to fetch. In memadr, if it is sw, goes to memwrite and then goes back to fetch as before. For R-Type, data goes into ALU with ALUSrcB 00 from register and then aluwb. For I-Type, data goes into ALU with ALUSrcB 01 from extend and then aluwb. For the decoding state, Read Registers and Calculate Target Address (PC+imm). If the op is beq, compare registers and Send Target PC (ALUOut) to PCNext. If op is jal, calculate PC + 4 and Send Target Address (ALUOut) to PCNext.



It has passed the test vector. The test vector is red in front is because there are 2 xx, the data is as expect, the terminal has no error output.


Main_FSM:mfsm
ALUOp[1..0]
ALUSrcA[1..0]
ALUSrcB[1..0]
AdrSrc
IRWrite
MemWrite
PCWrite
RegWrite
ResultSrc[1..0]
Zero
clk
op[6..0]
reset
ALUSrcA[1..0]
ALUSrcB[1..0]
AdrSrc
IRWrite
MemWrite
PCWrite
RegWrite
ResultSrc[1..0]
aludec:alu
ALUOp[1..0]
funct3[2..0]
funct7b5
opb5
ALUControl[2..0]
ALUControl[2..0]
funct3[2..0]
funct7b5
5
instrdec:ins
op[6..0]
ImmSrc[1..0]
immsrc[1..0]


Main_FSM:mfsm
Equal0
Equal1
A[6..0]
B[6..0]
7'h3
7'h23
OUT
always1
state
S0
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
clk
reset
op[6..0]
AdrSrc
RegWrite
WideOr0
ALUOp
WideOr1
ALUSrcA
PCUpdate
PCWrite~1
PCWrite~0
Zero
MemWrite
AdrSrc
RegWrite
IRWrite
PCWrite



	Source State	Destination State	Condition
1	S0	S1	
2	S1	S6	inpl[0]&inpl[1]&inpl[2]&inpl[3]&inpl[4]&inpl[5]&inpl[6]&(always1)
3	S1	S10	inpl[0]&inpl[1]&inpl[2]&inpl[3]&inpl[4]&inpl[5]&inpl[6]&(always1)
4	S1	S8	inpl[0]&inpl[1]&inpl[2]&inpl[3]&inpl[4]&inpl[5]&inpl[6]&(always1)
5	S1	S2	(always1)

Transitions | Encoding

The alu and instrdec is provided in the lab, so RTL schematics is not shown.