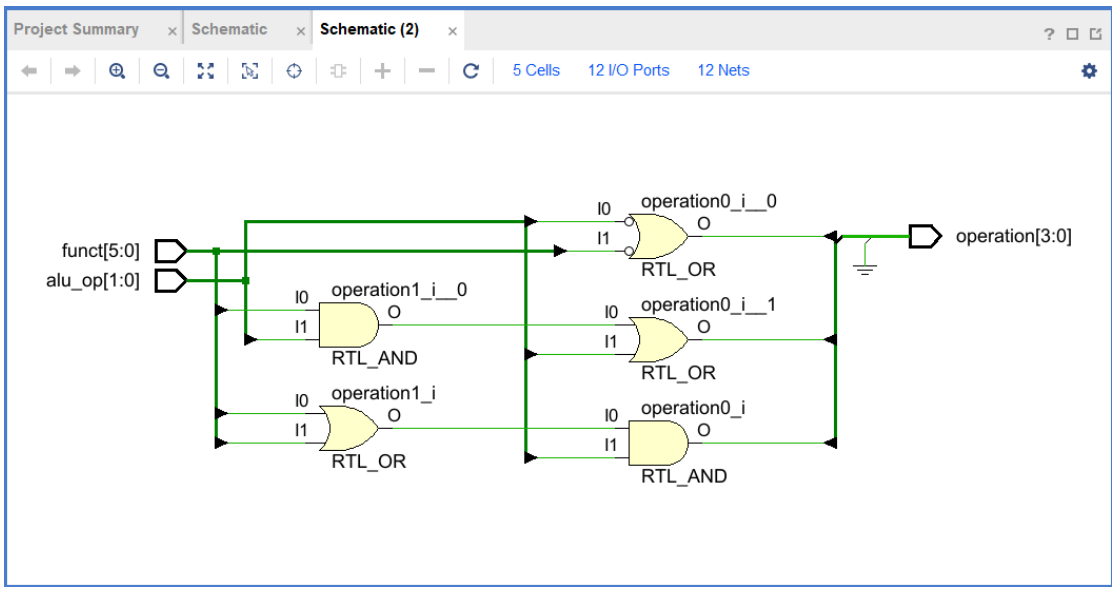
Lab2:report

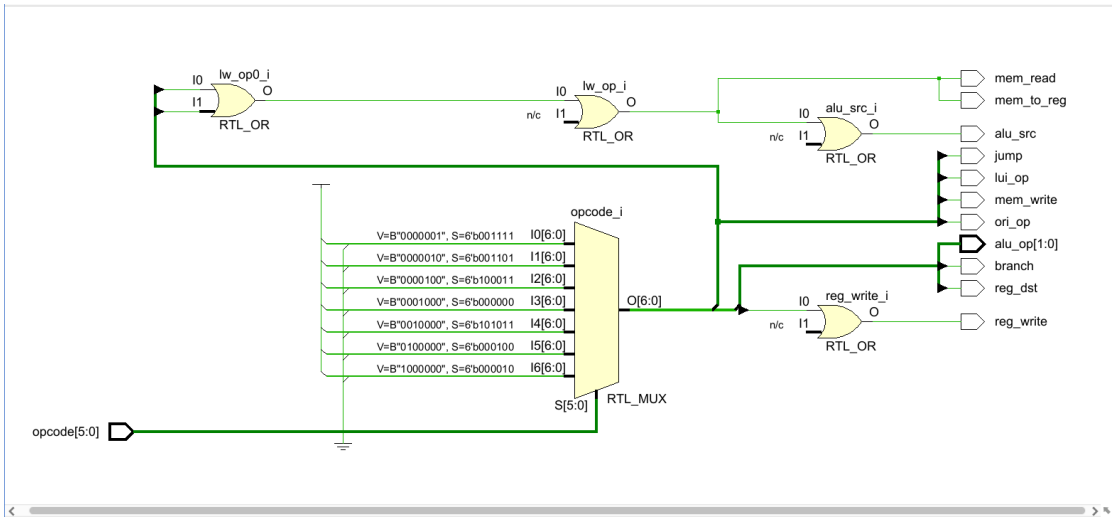
111550088 張育維

1. Architecture Diagrams



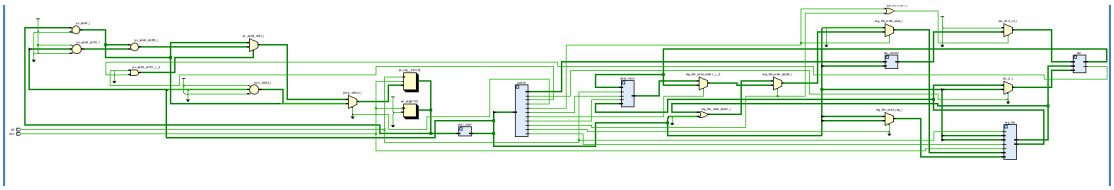
ALU control design

I follow the figure in textbook to implement. For operation[3], since it is always zero, I assign it =0. For operation[2], it equals to one when alu_op[0] = 1 or alu_op[1] = 1 and funct[1] = 1. For operation[1], it equals to one when funct[2] = 0 or alu_op[1] = 0. For operation[0], it equals to one when either funct[0] = 1 or funct[3] = 1 and alu_op[1] = 1.



Main control design

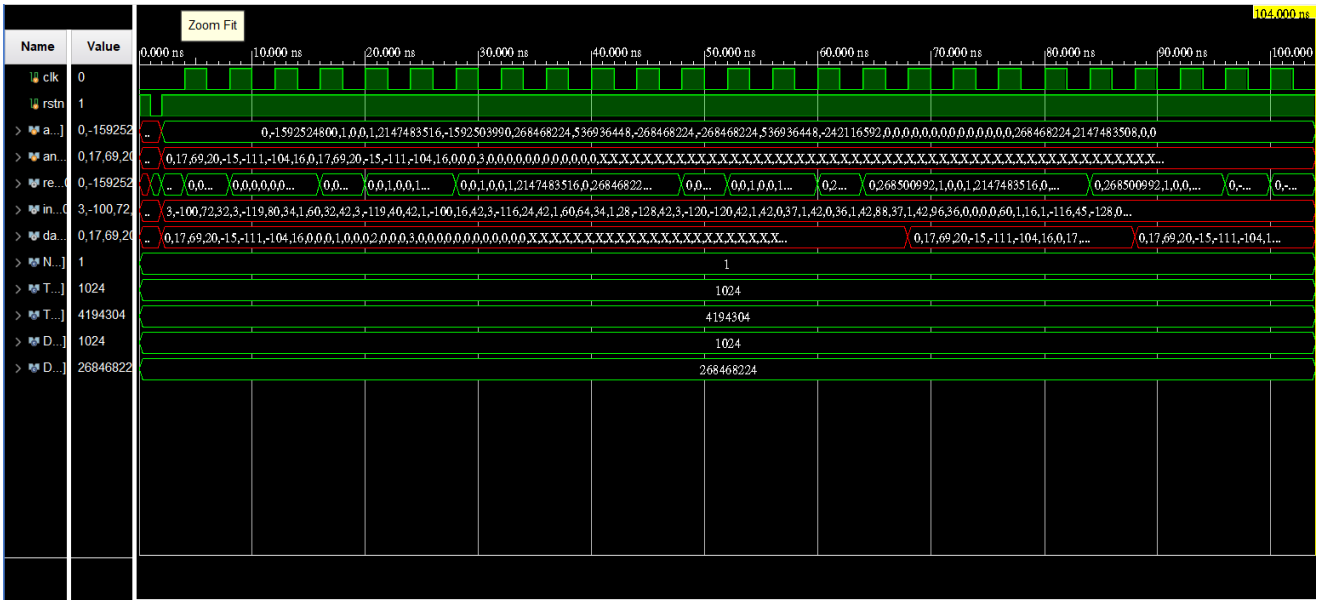
Lui_op equals to 1 when opcode = 001111. Ori_op equals to 1 when opcode = 001101. jump equals to 1 when opcode = 000010. Others output signal is generated by following the figure in appendix D of textbook.



Single-cycle processor design

I follow the figure in textbook and connect the input and output of different modules. Which are not on the figure are lui and ori. I add wires call ‘upper’ and ‘lower’ which store the messange of immediate type. One stores in upper 16 bits and the other stores in lower 16 bits. If lui_op equals to 1, the write data to register should be ‘upper’. If ori_op equals to 1, the write data to register should be the result of read data 1 doing ‘or’ operation with lower. Others just follow the figure and use ‘assign’ to wire, and use one ‘always’ with PC to synchronize the clock cycle. When rstn changes to zero, make PC back to the TEXT_START(the original address).

2. Experimental Result



2. What other cases you've tested? Why you choose them?

I choose “j 0x400060” because I’m not sure whether my jump instruction is correct, and make sure that I know what the Verilog does in jump instruction.

3. Answer the following Questions

1. When does write to register/memory happen during the clock cycle? How about read?

Write to memory happen when register read address 2 and alu result have the correct value, and memory write equals to 1. Write to register happen after multiplexer selects the value between read data of data memory and alu result, and

register write equals to 1. Read memory happen when alu result and register's read data 2 have the correct value and mem_read equals to 1. Read register happens when instruction[25:21], [20:16] get into read register 1 and 2.
(Because it is assign, every signal changes at the same time. Therefore I assume that happen means that the correct process is done.)

2. Translate the "branch" pseudo instructions (blt , bgt , ble , bge) in the Green Card into real instructions. Only at register can be modified, and other common registers should not be modified.
(using rs, rt to represent the component in the Green Card)

```
blt:
    slt $t1 rs rt,
    bne $t1 $zero Label
bgt:
    slt $t1 rt rs,
    bne $t1 $zero Label
ble :
    slt $t1 rt rs,
    beq $t1 $zero Label
bge:
    slt $t1 rs rt,
    beq $t1 $zero Label
```

3. Give a single beq assembly instruction that causes infinite loop. (consider that there's no delay slot)

start: beq $zero $zero start

Because $zero is always zero, it will be a infinite loop.

4. The j instruction can only jump to instructions within the "block" defined by "(PC+4)[31:28]". Design a method to allow j to jump to the next block (block number +1) using another j.

We can set the another jump instruction at the destination of first jump instruction. By jump twice, we can jump to the farther address.

eg:

```
j 0x03ffffff,
…
j 0x…(the address wanted to jump)
```

Because after first jump, the address will have …100 in the end. After the

PC + 4, (PC + 4)[31:28] = original one[31:28] + 1, meaning that it jumps to the next block.

5. Why a Single-Cycle Implementation Is Not Used Today?

First, because it run one command exactly in one cycle, the cycle length must be long enough. However, it will cause excessively long cycle, and reduce the overall performance.

Next, single-cycle design needs the complex control units because it needs to control all operations in one cycle. This will increase the design complexity and cost.