林庭寫

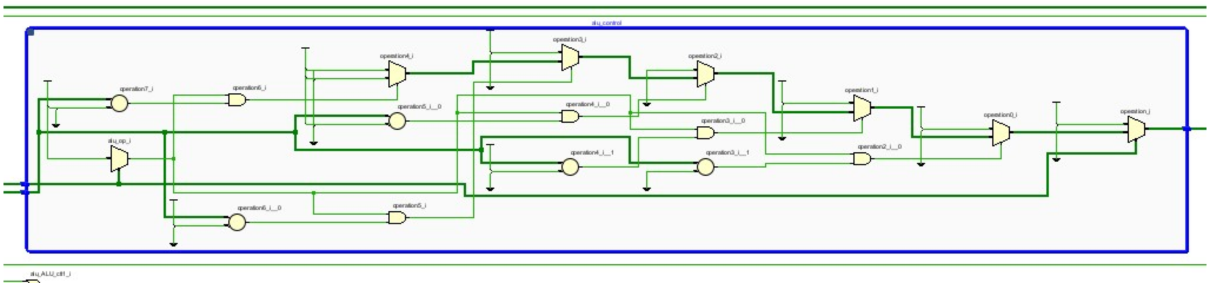
2024年4月24日

Computer Organization

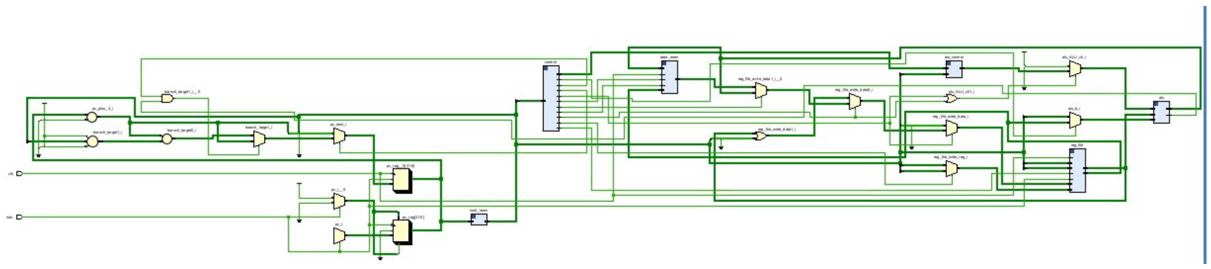
HW2 Single-Cycle Processor Report

Architecture Diagrams

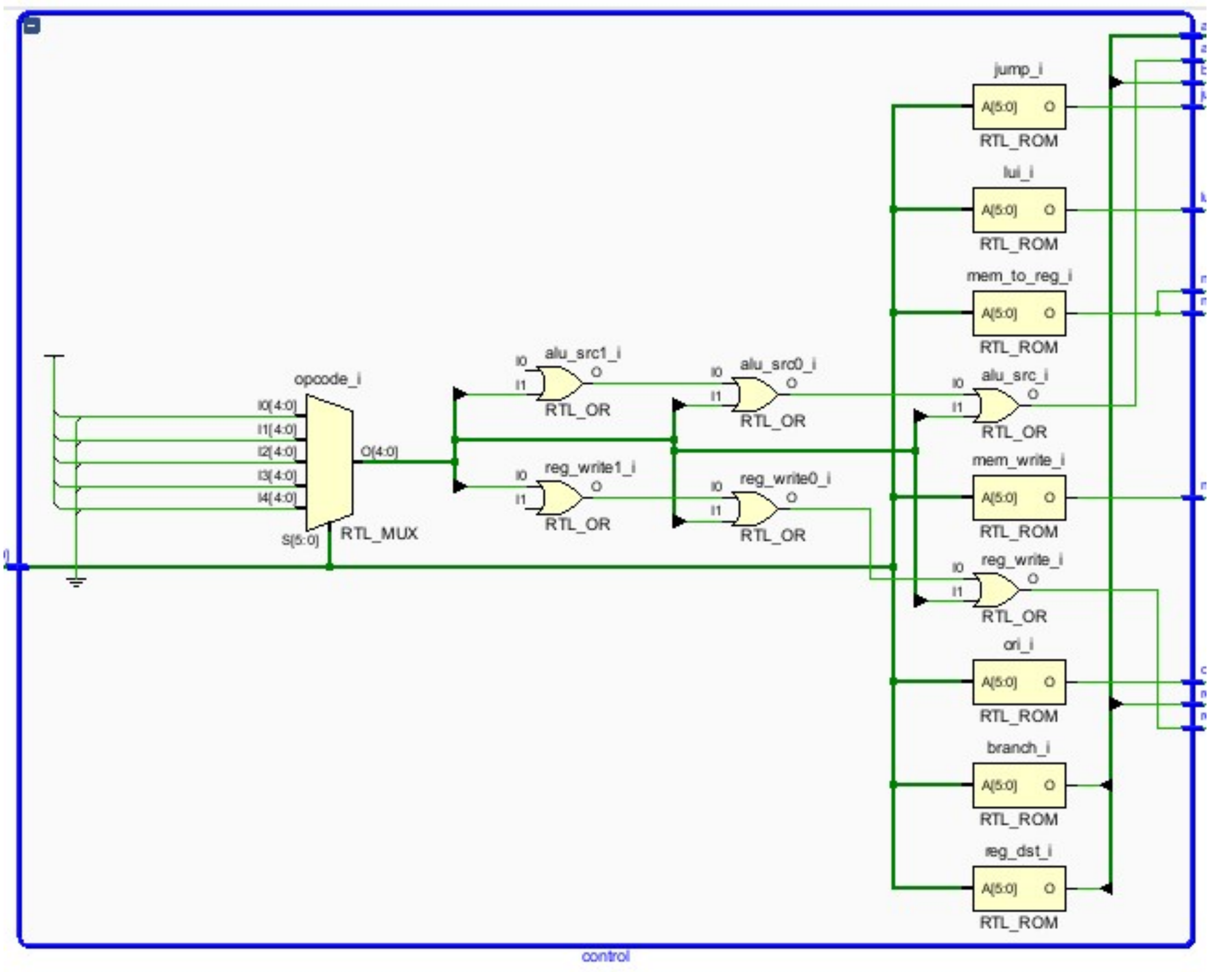
ALU Control



Single-Cycle Processor

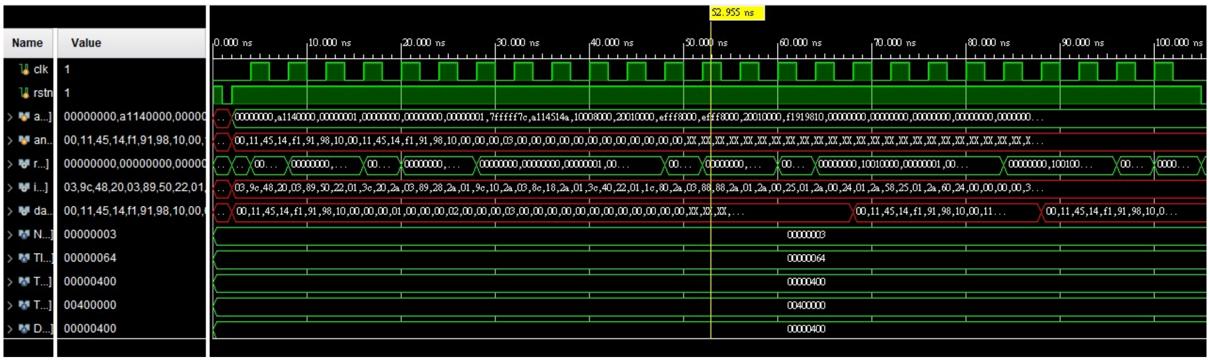


Main Control



Experimental Results

Wave Forms



Extra text case

```
main:
    lui $t0, 0x1000                   # Load upper immediate to set upper 16 bits of $t0 to 0x1000
    lw $t1,  hun                      # Load word from memory at address hun into $t1
    ori $t2, $t1, 0xFF                # Bitwise OR immediate to set $t2 with value of $t1 OR 0xFF
    slt $t3, $t1, $t2                 # Set less than to compare $t1 and $t2, result stored in $t3
    j loop                            # Jump to the loop label

loop:
    lw $t4, hah
    sw $t4, 8($gp)                    # Store word from $t4 into memory at address $gp + 8
    or $t7, $zero, $gp                # should not execute
    lw $t6, 0($gp)                    # should not execute
```

Creating a new 1.s file with these instructions and I simply want to check out whether "lui", "lw", "ori", "slt", "j", "sw" work.

Answer Questions

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

a. Write to Register/Memory:

Write operations typically occur on the positive edge (always @(posedge clk & rstn).) of the clock signal.

b. Read from Register/Memory:

Read operations can occur asynchronously, meaning they can happen at any time without being synchronized to the clock signal.

2. Translate:

- blt

  slt $at, $t0, $t1

  bne $at, $zero, b_target
- bgt

  slt $at, $t0, $t1

  bne $at, $zero, target_label
- ble

  slt $at, $t0, $t1

  beq $at, $zero, b_target
- bge

  slt $at, $t0, $t1

  beq $at, $zero, b_target

3. Infinite Loop

loop: beq $zero, $zero, loop

4. Jump to next block

addi $ra, $ra, 0x1000   // assuming a block is 4096bytes, $t0 stores next block address

j label

label: // label locates in the former line of next memory block(last line in current one)

5. Why Single-Cycle implementation is not used today.

- **Long Clock Cycles**: Single-cycle designs have long clock cycles, inefficiently treating all instructions equally and resulting in slower overall performance., which indicates the waste of clock cycles: Shorter instructions finish early, leading to wasted clock cycles as the processor idles until the next cycle begins.
- **Complexity of Control Logic**: Managing all instructions within a single clock cycle requires complex control logic, which becomes larger and more intricate as the instruction set grows.
- **Pipelining Difficulty**: Single-cycle designs are not easily pipelined due to their unitary nature, making it challenging to overlap instruction execution stages efficiently.
- **Inefficient Area and Power Usage**: Complex control logic increases chip area and power consumption compared to more modern designs, impacting efficiency and cost-effectiveness.