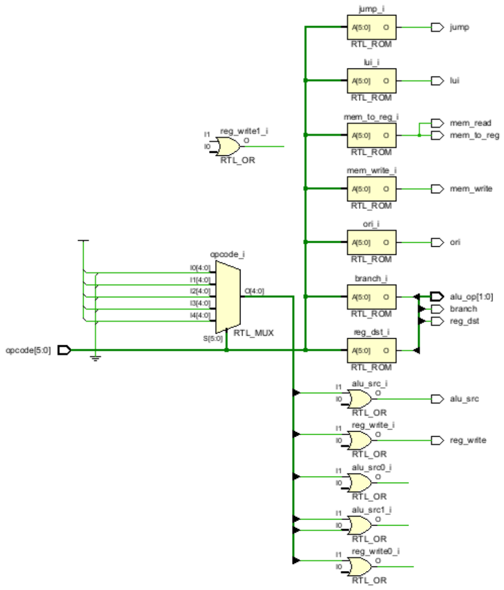
Lab 2: Single-Cycle Processor

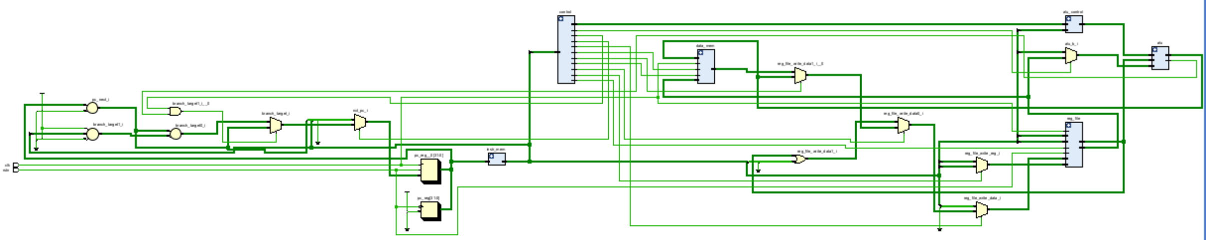
學號: 111550129　姓名: 林彥亨

1. Architecture Diagrams.

main_control

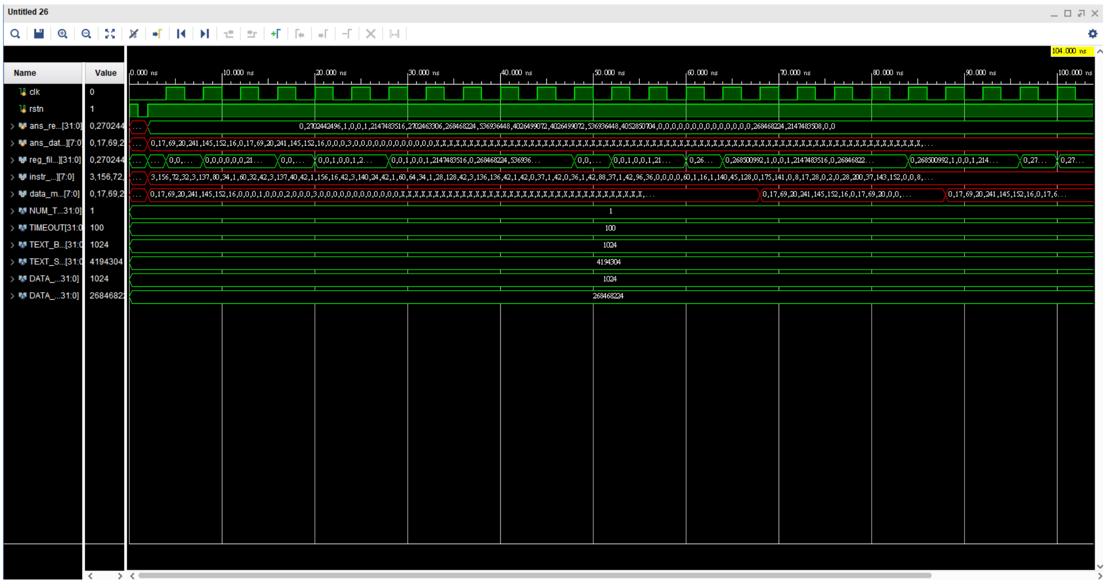


single_cycle processor



2. Experimental Result

1. Show the waveform screen shot of the test we provided.



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

Here is the 1.s I generate.

```
testbench > 1.s
            .data   0x10008000      # start of Dynamic Data (pointed by $gp)
    hun:    .word   0x00114514      # 0($gp)
    hah:    .word   0xf1919810
            .word   0x1
            .word   0x2
            .word   0x4             # Changed value from 0x3 to 0x4 at 16($gp)

            .text   0x00400000      # start of Text (pointed by PC),
                                    # Be careful there might be some other instructions in JsSPIM.
                                    # Recommend at least 9 instructions to cover out those other instructions.
    main:   add     $t1, $gp, $gp   # $t1 = 2 * $gp
            sub     $t2, $gp, $t1   # $t2 = - $gp
            slt     $a0, $t1, $gp   # $a0 = 0
            slt     $a1, $gp, $t1   # $a1 = 1
            slt     $v0, $t4, $gp   # $v0 = 1 (since $t4 is negative)
            slt     $v1, $gp, $t4   # $v1 = 0
            add     $t0, $t1, $gp   # Changed from sub to add, $t0 = 3 * $gp
            slt     $s0, $t0, $gp   # $s0 = 0
            slt     $s1, $gp, $t0   # $s1 = 1, Changed from 0
            or      $zero, $t1, $t2 # test write to zero
            and     $zero, $t1, $t2
            or      $t3, $t1, $t2   # test OR
            and     $t4, $t1, $t2   # test AND
            nop                     # test NOP
            lw      $t5, hun        # test LW
            sw      $t5, 16($gp)    # Changed offset from 8 to 16
    bst:    beq     $t0, $gp, btg   # [bst] should branch
            or      $t9, $zero, $gp # should not execute
            lw      $t8, 0($gp)     # should not execute
    btg:    j       end             # [btg] should jump
            or      $t7, $zero, $gp # should not execute
            lw      $t6, 0($gp)     # should not execute
    end:    lw      $t5, hah        # [end]
            sw      $t5, 20($gp)    # Changed offset from 12 to 20
            beq     $zero, $gp, bst # should not branch
            li      $a3, 0xa114514b # Changed immediate in li (lui, ori)
```

3. Answer the following Questions

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

A:

- read register: It occurs during the **Instruction Decode** stage.
- read memory: It occurs during the **Memory Access** stage
- write register: It happens during the **Write Back** stage.
- write memory: It happes during the **Memory Access** stage.

2. Trnaslate the “brach” 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.

   A:

   - blt:

     slt $at, $rs, $rt
     bne $at, $zero, label

   - bgt:

     slt $at, $rt, $rs

     bne $at, $zero, label

   - ble:

     slt $at, $rt, $rs
     beq $at, $zero, label

   - bge:

     slt $at, $rs, $rt
     beq $at, $zero, label

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

   A:

   beq $zero, $zero, loop
   loop:

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

   A:

1. **Load Upper Immediate (`lui`)**: Use the `lui` instruction to load the upper 16 bits of the target address into a register. This sets the upper half of the register to the upper 16 bits of the address and the lower half to zeros.

2. **Jump Register (`jr`)**: Use the `jr` (jump register) instruction to jump to the address contained in a register. This allows for an indirect jump to any 32-bit address loaded into that register.

For example:

Assume $t0 will be used to store the jump target address

Let's say the current PC is at 0x00400000 and we want to jump to 0x00800000 (start of the next 256 MB block)

```
lui $t0, 0x0080   # Load upper immediate; sets $t0 to 0x00800000
jr $t0            # Jump to the address in $t0
```

This would effectively jump to 0x00800000, beginning of the next block

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

A: Because the single-cycle implementation is inefficient.

4. Problems Encoutered & Solution

First, since there are some mistakes of my lab 1 in the part of slt. I spent a lot of time to find out the correct way to write the msb_alu. Second, when i complete the single_cycle.v there are lots of problems, however, i don't know how to use the JsSpim. It took me a lot of time to understand it. Third, during the test, the control.v and the alu_control may have some mistakes lead to the output is delay and the reg_file cannot study in. It took me a lot of time to modify them. Finally, due to not familiar to Vivado, in the beginnig, the diagrams still comstruct the same and i didn't know why, then i set the control.v as top the diasgram shows, it makes me happy.

5. Feedback

Although TAs set the deadline later, i still late for the submit the homeworks. It makes me upset, thanks for the TAs' instructions.