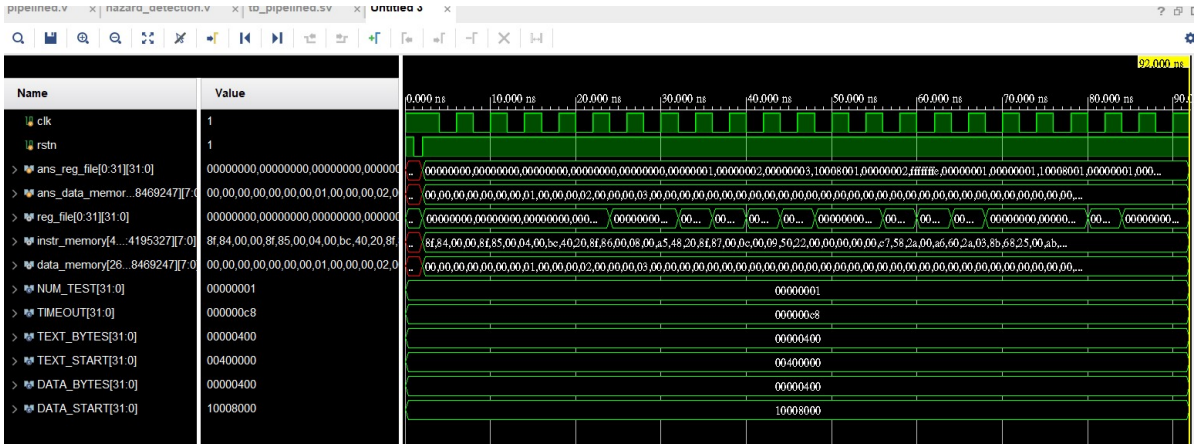
HW4 Report

分類	
Date	@May 17, 2024 8:55 PM
Review	☐

Experiment Result

1. Waveform



1. Other test case
   - test if addi can work properly by the following test case:

```
addi $t3 $t4 100
add $t5 $t4 $t3
```

   - test for `lw` and `beq` (required additional stall condition):

```
lw  $t4, 8($gp)
nop
beq $t3, $t4, end
```

Questions

1. The equation and the false in text book
   - EX hazard:

     ```
     if (EX_MEM_reg_write && (EX_MEM_rd != 0) &&
                 (EX_MEM_rd == ID_EX_rs)) forward_A <= 2'b10;
     else if (MEM_WB_reg_write && (MEM_WB_rd != 0) &&
                 (MEM_WB_rd == ID_EX_rs)) forward_A <= 2'b01;
     else forward_A <= 2'b00;
     ```

   - MEM hazard:

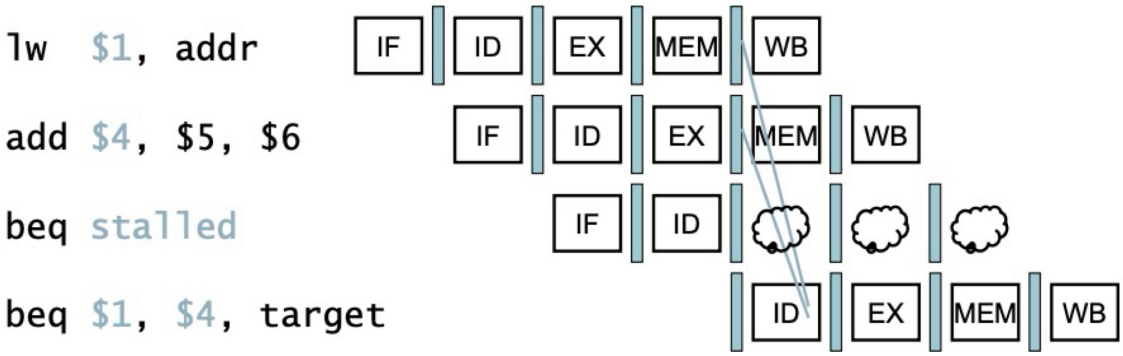
     ```
     if (EX_MEM_reg_write && (EX_MEM_rd != 0) &&
                 (EX_MEM_rd == ID_EX_rt)) forward_B <= 2'b10;
     else if (MEM_WB_reg_write && (MEM_WB_rd != 0) &&
                 (MEM_WB_rd == ID_EX_rt)) forward_B <= 2'b01;
     else forward_B <= 2'b00;
     ```

   - False

     If EX_MEM_rd is derived directly from ID_EX_rd, than is is wrong. However, I set the following condition, making the EX_MEM_rd can fit the formula provided in text_book.

     ```
     if (ID_EX_EX[3]) EX_MEM_rd <= ID_EX_rd;
     else EX_MEM_rd <= ID_EX_rt;
     ```

2. Is forwarding form MEM_WB needed?
   - We try to use reduce the branch delay, so we need to forward the read data from MEM_WB to ID stage.
   - Considering the following situation:



- The correct value of $1 will be write back to ID stage in WB stage, but it should take more time to derive data from memory; that is, the ID stage may not able to reach the correct write back data in time. Adding forwarding will solve the problem.
- This cause additional stall or forwarding.

3. insert 2 stalls

```
always @(*)begin
    if (branch_read_reg_after_load)begin
        nxt_stall <= 2;
    end
    else if (branch_read_reg_after_ALU || stall_condition)beg
        nxt_stall <= 1;
    end
end

always @(posedge clk && rstn)begin
    if(nxt_stall > 0)begin
        nxt_stall <= nxt_stall - 1;
    end
    else begin
        nxt_stall <= 0;
    end
end

always @(negedge rstn) begin
```

```
    nxt_stall <= 0;
end
```

- use a register “nxt_stall” to record the remaining stall(not yet down), and

  `assign stall = (nxt_stall > 0) ;`

- when a stall is taken, do

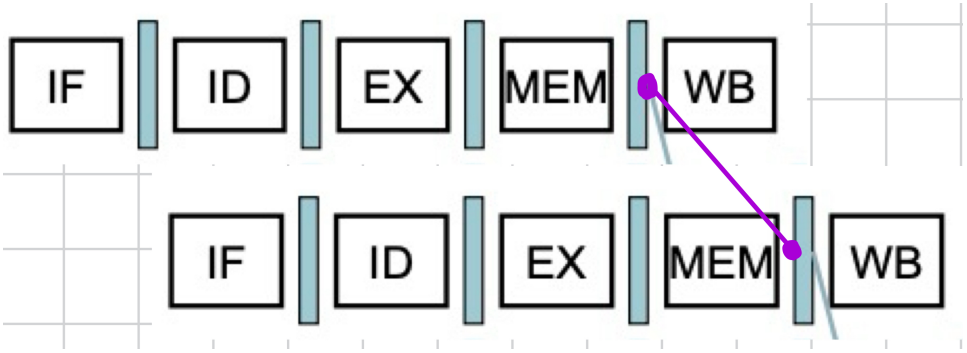
  `nxt_stall <= nxt_stall - 1`

- stall_condition including additional stall-needed condition including addi.

4. sw right after lw with no stall:
   - We can add a forwarding:

     #1 `sw $1 $addr`

     #2 `lw $2 $addr`



     - if (MEM_WB_rd == EX_MEM_rd) && (MEM_WB_offset == EX_MEM_offset):
       - forward #1 MEM_read_data from MEM_WB to replace the #2 MEM_read_data