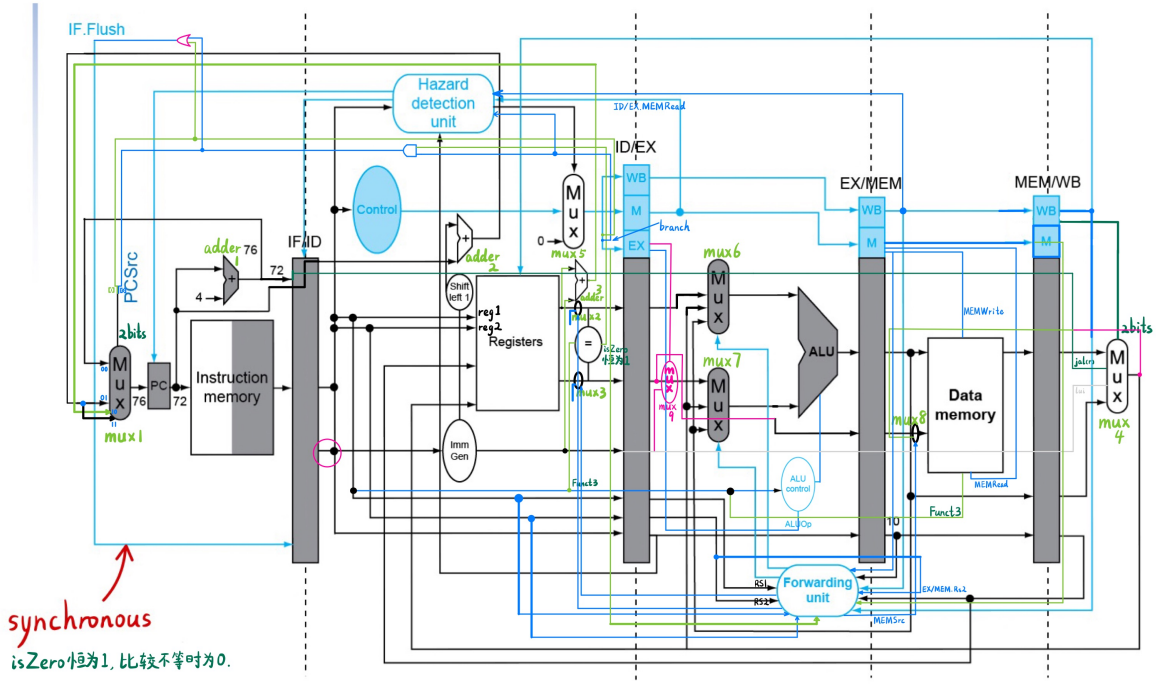
VE370 Lab 5 Report

Group 4 Wenqi Cao, Jingye Lin, Lan Wang

Modeling and Implementation

We modified the graph of pipelined processor to support hazard detection and fix. The new design is shown below:



Data and control hazards are solved by adding Hazard detection unit and Forwarding unit. The logic of judgement is shown below (some default conditions are emitted):

Hazard detection unit:

```
if (IDEX_MemRead && !ID_MemWrite) begin
    if (IDEX_RegRd == IFID_r1 || IDEX_RegRd == IFID_r2) begin
        // load-use
        PC_write=1'b0; IFID_write=1'b0; mux_con=1'b0; // insert nop
    end
    else  // default
end
else if (ID_branch && IDEX_RegWrite) begin
    if ((IDEX_RegRd!=0) && (IFID_r1 == IDEX_RegRd || IFID_r2 == IDEX_RegRd))
begin
        // R-branch
        PC_write=1'b0; IFID_write=1'b0; mux_con=1'b0;
    end
    else  // default
end
else if (ID_branch && EXMEM_MemRead) begin
```

```
    if ((EXMEM_RegRd!=0) && (IFID_r1 == EXMEM_RegRd || IFID_r2 == EXMEM_RegRd))
begin
        // load-branch (two nops)
        // another nop is inserted by R-branch type
        PC_write=1'b0; IFID_write=1'b0; mux_con=1'b0;
    end
    else  // default
end
else  // default
```

Forwarding unit:

```
//ForwardingA (00: from register; 01: from MEM; 10: from EX)
        //EX
        if (EXMEM_RegWrite && (EXMEM_RegRd!=0) && EXMEM_RegRd == IDEX_r1 &&
!EXMEM_MemRead && !IDEX_MemRead && !IDEX_MemWrite)
            Forwarding_A = 2'b10;
        //MEM
        else if (MEMWB_RegWrite && (MEMWB_RegRd!=0) && MEMWB_RegRd == IDEX_r1)
            Forwarding_A = 2'b01;
        else
            Forwarding_A = 2'b00;

//ForwardingB (00: from register; 01: from MEM; 10: from EX)
         //EX
        if (EXMEM_RegWrite && (EXMEM_RegRd!=0) && EXMEM_RegRd == IDEX_r2 &&
!EXMEM_MemRead && !IDEX_MemRead && !IDEX_MemWrite)
            Forwarding_B = 2'b10;
         //MEM
        else if (MEMWB_RegWrite && (MEMWB_RegRd!=0) && MEMWB_RegRd == IDEX_r2)
            Forwarding_B = 2'b01;
        else
            Forwarding_B = 2'b00;

//Memsrc (for the mux before writeData of memory)
        if (MEMWB_RegRd == EXMEM_r2 && EXMEM_MemWrite)
            MemSrc = 1'b1;
        else
            MemSrc = 1'b0;

//ForwardingEq (the selection signal of the mux of inputs for comparator)
        if (EXMEM_RegWrite && (EXMEM_RegRd!=0) && EXMEM_RegRd == IFID_r1 &&
IFID_branch)
            Forwarding_Eq1 = 1'b1;
        else
            Forwarding_Eq1 = 1'b0;

        if (EXMEM_RegWrite && (EXMEM_RegRd!=0) && EXMEM_RegRd == IFID_r2 &&
IFID_branch)
            Forwarding_Eq2 = 1'b1;
        else
            Forwarding_Eq2 = 1'b0;
```

Simulation Result

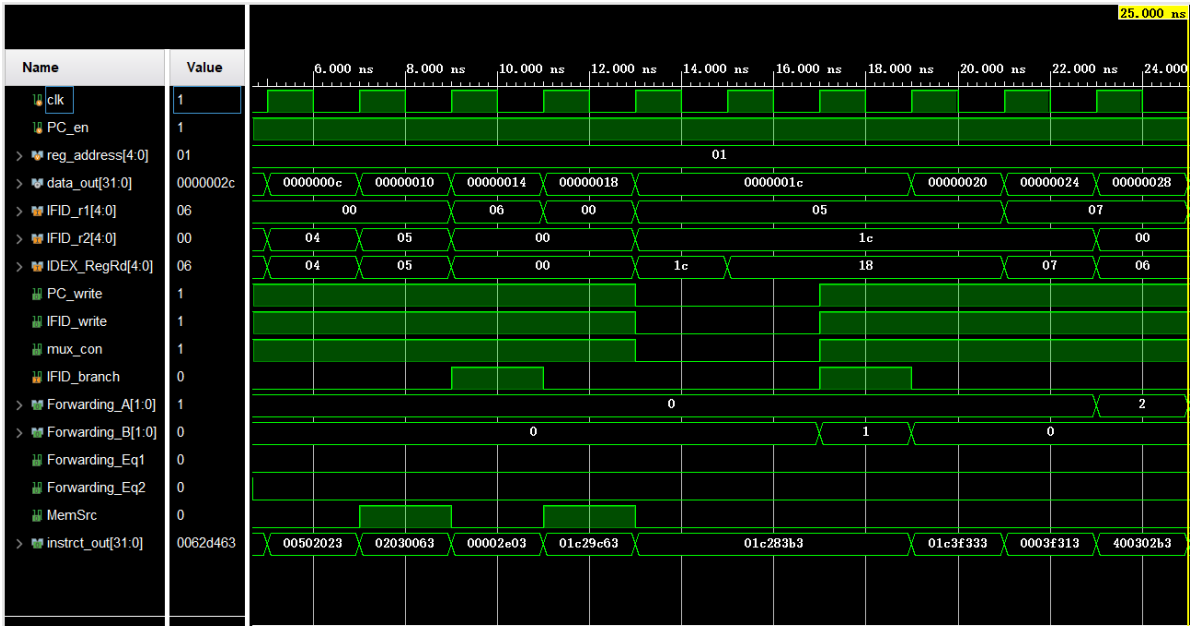
Hazard 1

```
00002e03  lw x28 0 x0
01c29c63  bne x5 x28 24 <wrong_branch>
01c283b3  add x7 x5 x28
```

instruction：01c283b3 `add x7 x5 x28`

hazard type: a data hazard by load

resolve: use the hazard detection unit to add bubbles in it: `PC_write=0`, `IFID_write=0`, `mux_con=0` for two clock cycles, and use forwarding unit to give one MEM forwarding for forwarding B.



Hazard 2

```
01c283b3 add x7 x5 x28
01c3f333 and x6 x7 x28
0003f313 andi x6 x7 0
400302b3 sub x5 x6 x0
0062d463 bge x5 x6 8 <right_branch>
```

instruction：01c3f333 `and x6 x7 x28`

hazard type: data hazard on `x7`

resolve: give an EX forwarding on forwarding A

instruction：0003f313 `andi x6 x7 0`

hazard type: data hazard on `x7`

resolve: give an MEM forwarding on forwarding A

instruction：400302b3 `sub x5 x6 x0`

0062d463 `bge x5 x6 8 <right_branch>`

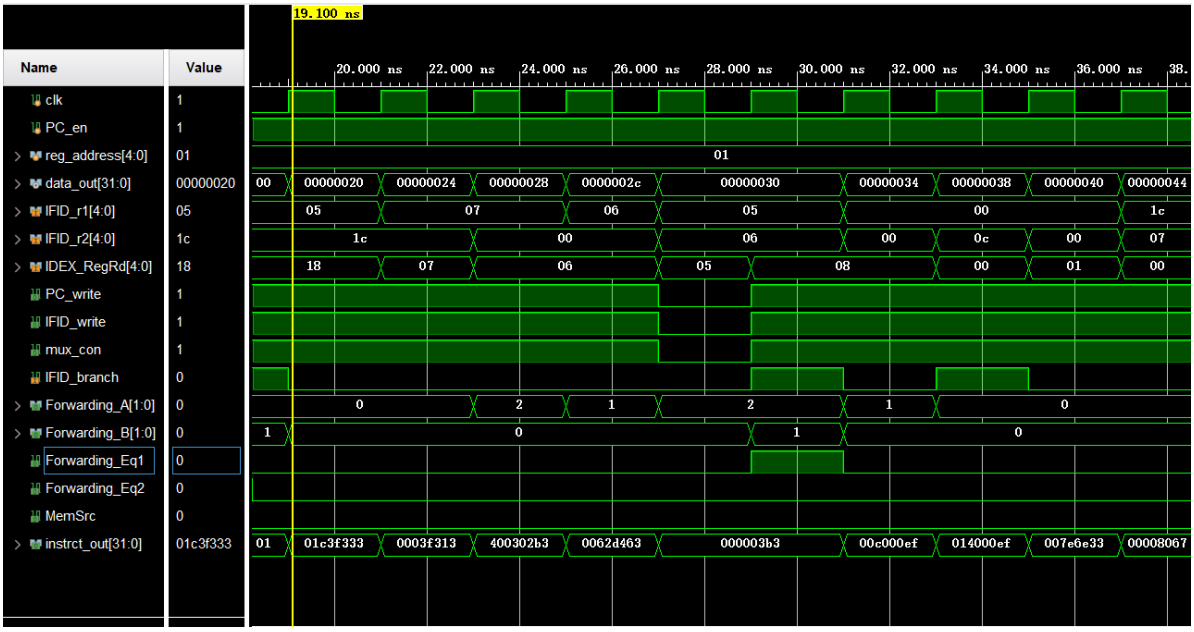
hazard type: data hazard on `x6` and control hazard

resolve: give an EX forwarding on forwarding A for two clock cycles, and an MEM forwarding on forwarding B for one clock cycle, with `forwarding_eq1=1` for one clock cycle, and use the hazard detection unit to add bubbles in it: `PC_write=0`, `IFID_write=0`, `mux_con=0` for one clock cycle.

instruction：0062d463 `bge x5 x6 8 <right_branch>`

hazard type: data hazard on `x5`

resolve: give an MEM forwarding on forwarding A



RTL Design