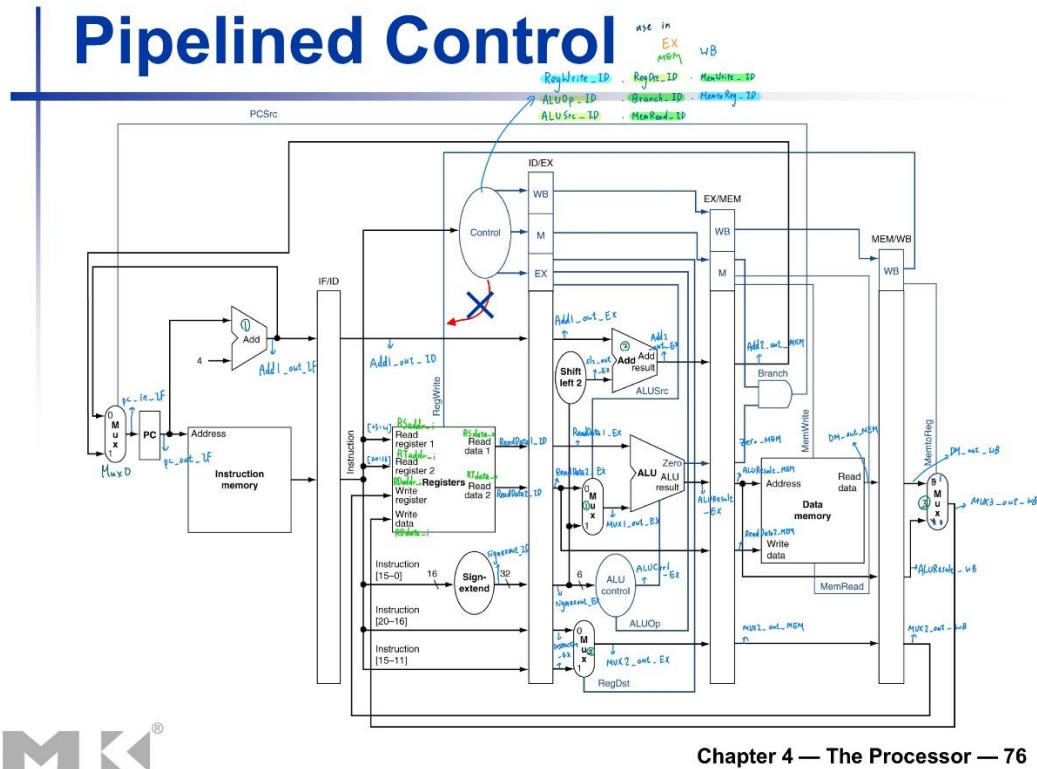


Computer Organization Lab4

ID:110550065

Name:尤茂為

Architecture diagrams:



Hardware module analysis:

(Explain how you design work and what are the difference between this lab and Lab3)

In Lab4 ,we don't need to handle the jal and jr instruction, so we can use MUX 2to1 instead of MUX 3to1.Also,The input of MUX3(MemtoReg) is different from this lab. In order to deal with XOR and MULT instructions, I add some code for these instructions.

ALUCtrl:

```
always@(ALUOp_i, funct_i) begin
    if (ALUOp_i == 3'b000) begin // R-type
        case (funct_i)
            6'b100000: ALUCtrl_o <= 4'b0010; // add
            6'b100010: ALUCtrl_o <= 4'b0110; // sub
            6'b100100: ALUCtrl_o <= 4'b0000; // and
            6'b100101: ALUCtrl_o <= 4'b0001; // or
            6'b100111: ALUCtrl_o <= 4'b1100; // xor
            6'b101010: ALUCtrl_o <= 4'b0111; // slt
            // new
            6'b011000: ALUCtrl_o <= 4'b1101; // mult
            6'b100110: ALUCtrl_o <= 4'b1110; // xor
        endcase
    end
end
```

ALU:

```
57 @
58 ○
59 @
60 @
61 ○
62 @
63 ○
64 @

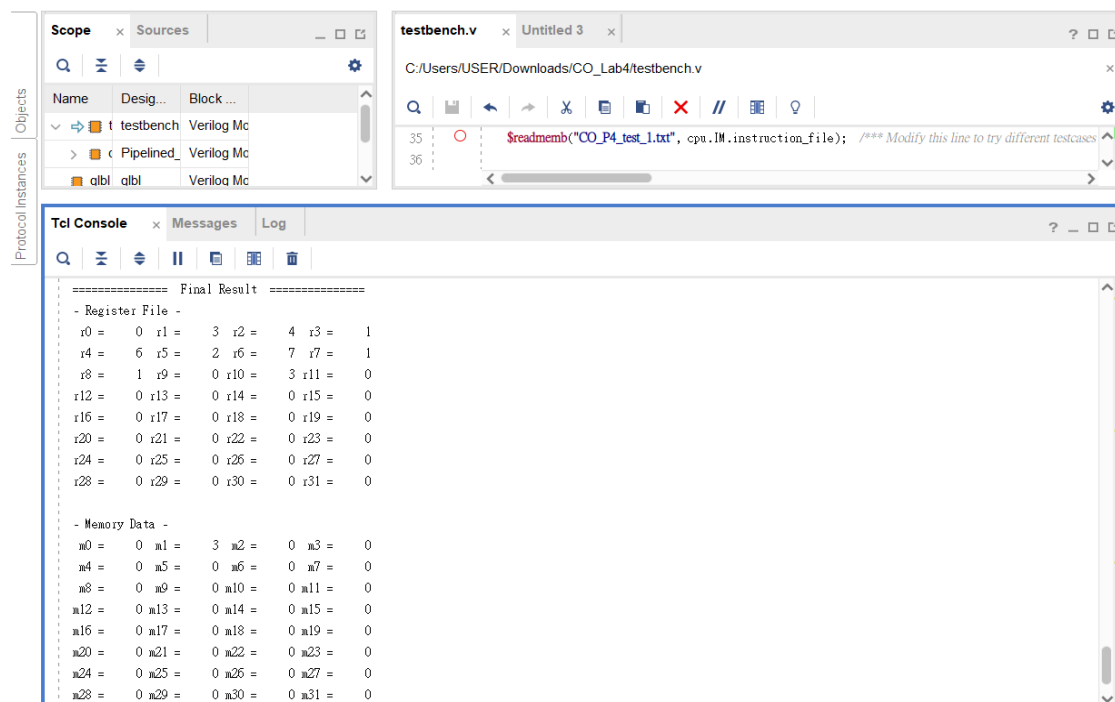
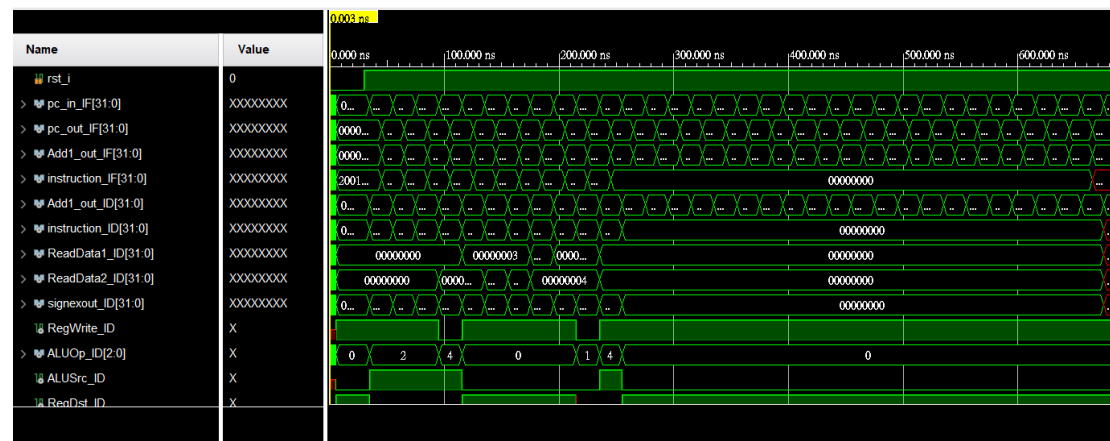
4'b1101: begin // mult
    result_o <= src1_i * src2_i;
end
4'b1110: begin // xor
    result_o <= src1_i ^ src2_i;
end
default: result_o <= 0;
endcase
```

The main difference between lab4 and lab3 is we need numbers of wires to store the data in different stage since different stages has different instructions in the same time.

Simulation results:

(Show the screenshot of the simulation results and waveform and explain them)

Test1:



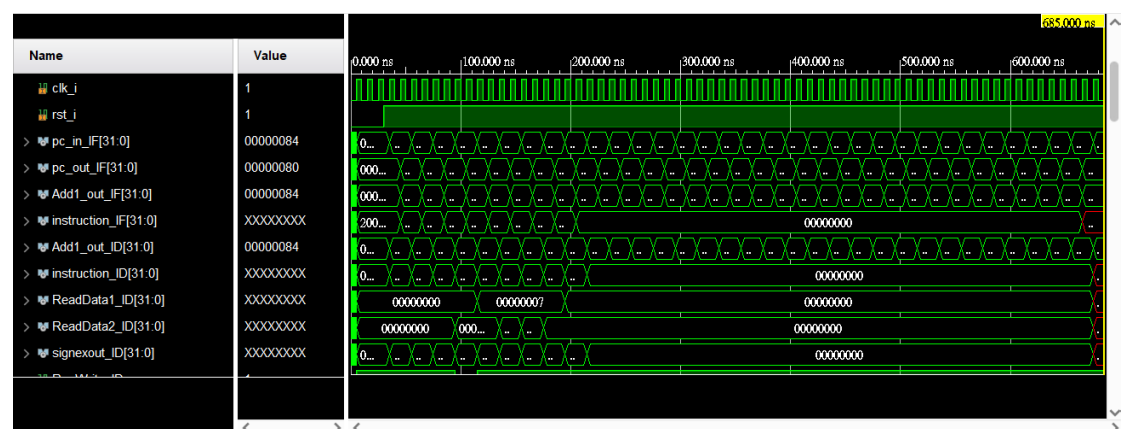
CO_P4_Result...			
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明			
r0 =	0	r1 =	3
r2 =	4	r3 =	1
r4 =	6	r5 =	2
r6 =	7	r7 =	1
r8 =	1	r9 =	0
r10 =	3	r11 =	0
r12 =	0	r13 =	0
r14 =	0	r15 =	0
r16 =	0	r17 =	0
r18 =	0	r19 =	0
r20 =	0	r21 =	0
r22 =	0	r23 =	0
r24 =	0	r25 =	0
r26 =	0	r27 =	0
r28 =	0	r29 =	0
r30 =	0	r31 =	0
m0 =	0	m1 =	3
m2 =	0	m3 =	0
m4 =	0	m5 =	0
m6 =	0	m7 =	0
m8 =	0	m9 =	0
m10 =	0	m11 =	0
m12 =	0	m13 =	0
m14 =	0	m15 =	0
m16 =	0	m17 =	0
m18 =	0	m19 =	0
m20 =	0	m21 =	0
m22 =	0	m23 =	0
m24 =	0	m25 =	0
m26 =	0	m27 =	0
m28 =	0	m29 =	0
m30 =	0	m31 =	0

10 Windows (CRLF)

UTF-8

same as the CO_P4_test_1_result

Test2:



Scope Sources

- CO_P4_test_1.txt
- CO_P4_test_2.txt

Hierarchy Libraries Compile Order

testbench.v x Untitled 2 x

C:/Users/USER/Downloads/CO_Lab4/testbench.v

```
35 $readmemb("CO_P4_test_2.txt", cpu.IW.instruction_file); /*** Modify this line to try different testcases
```

Tel Console x Messages Log

Final Result

```
===== Final Result =====
- Register File -
r0 = 0 r1 = 0 r2 = 4 r3 = 5
r4 = 49 r5 = 0 r6 = 3 r7 = 5
r8 = 1 r9 = 0 r10 = 7 r11 = 7
r12 = 0 r13 = 0 r14 = 0 r15 = 0
r16 = 0 r17 = 0 r18 = 0 r19 = 0
r20 = 0 r21 = 0 r22 = 0 r23 = 0
r24 = 0 r25 = 0 r26 = 0 r27 = 0
r28 = 0 r29 = 0 r30 = 0 r31 = 0

- Memory Data -
m0 = 0 m1 = 7 m2 = 0 m3 = 0
m4 = 0 m5 = 0 m6 = 0 m7 = 0
m8 = 0 m9 = 0 m10 = 0 m11 = 0
m12 = 0 m13 = 0 m14 = 0 m15 = 0
m16 = 0 m17 = 0 m18 = 0 m19 = 0
m20 = 0 m21 = 0 m22 = 0 m23 = 0
m24 = 0 m25 = 0 m26 = 0 m27 = 0
m28 = 0 m29 = 0 m30 = 0 m31 = 0
```

Type a Tcl command here

CO_P4_Result...

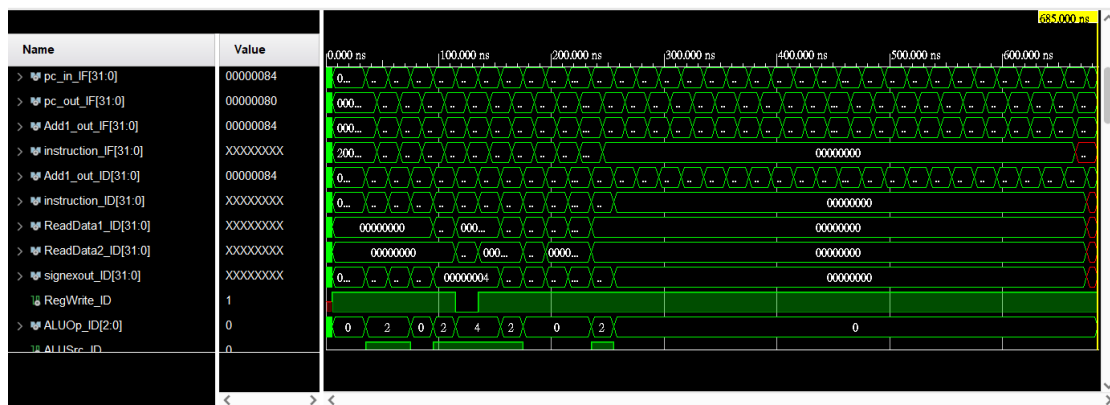
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

r0 =	0	r1 =	0
r2 =	4	r3 =	5
r4 =	49	r5 =	0
r6 =	3	r7 =	5
r8 =	1	r9 =	0
r10 =	7	r11 =	7
r12 =	0	r13 =	0
r14 =	0	r15 =	0
r16 =	0	r17 =	0
r18 =	0	r19 =	0
r20 =	0	r21 =	0
r22 =	0	r23 =	0
r24 =	0	r25 =	0
r26 =	0	r27 =	0
r28 =	0	r29 =	0
r30 =	0	r31 =	0
m0 =	0	m1 =	7
m2 =	0	m3 =	0
m4 =	0	m5 =	0
m6 =	0	m7 =	0
m8 =	0	m9 =	0
m10 =	0	m11 =	0
m12 =	0	m13 =	0
m14 =	0	m15 =	0
m16 =	0	m17 =	0
m18 =	0	m19 =	0
m20 =	0	m21 =	0
m22 =	0	m23 =	0
m24 =	0	m25 =	0
m26 =	0	m27 =	0
m28 =	0	m29 =	0
m30 =	0	m31 =	0

10 Windows (CRLF) UTF-8

same as the CO_P4_test_2_result

Bonus Test3:



Scope Sources

CO_P4_test_3_hazard-modify.txt

Hierarchy Libraries Compile Order

testbench.v x Untitled 1 x

C:/Users/USER/Downloads/CO_Lab4/testbench.v

35 \$readmemb("CO_P4_test_3_hazard-modify.txt", cpu.instruction_file);

Tcl Console

```

===== Final Result =====
- Register File -
r0 = 0 r1 = 16 r2 = 20 r3 = 8
r4 = 16 r5 = 8 r6 = 24 r7 = 26
r8 = 8 r9 = 100 r10 = 0 r11 = 0
r12 = 0 r13 = 0 r14 = 0 r15 = 0
r16 = 0 r17 = 0 r18 = 0 r19 = 0
r20 = 0 r21 = 0 r22 = 0 r23 = 0
r24 = 0 r25 = 0 r26 = 0 r27 = 0
r28 = 0 r29 = 0 r30 = 0 r31 = 0

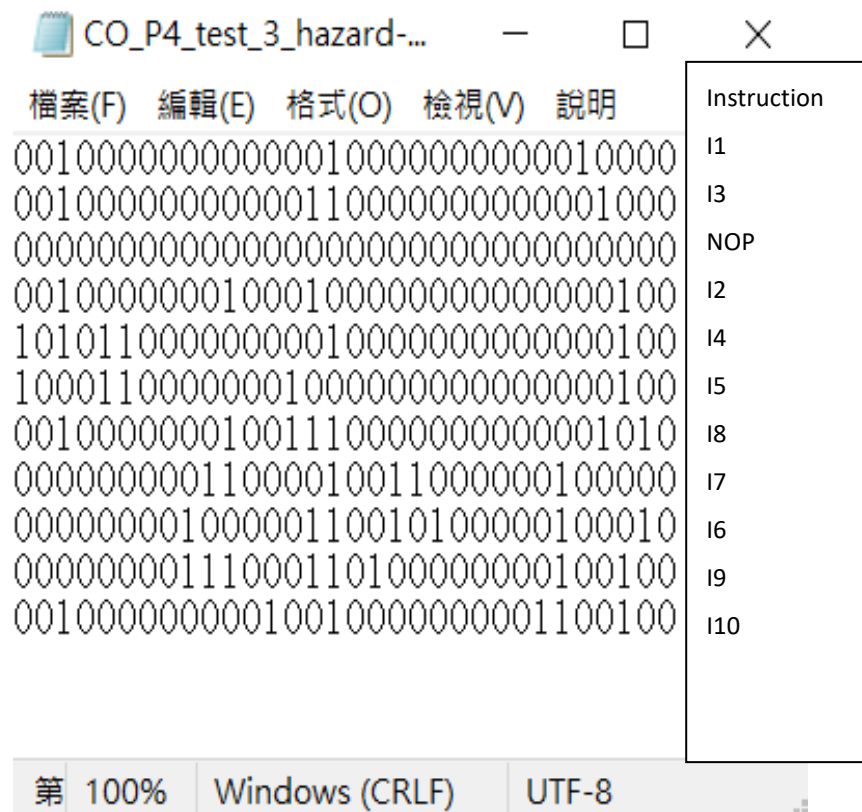
- Memory Data -
m0 = 0 m1 = 16 m2 = 0 m3 = 0
m4 = 0 m5 = 0 m6 = 0 m7 = 0
m8 = 0 m9 = 0 m10 = 0 m11 = 0
m12 = 0 m13 = 0 m14 = 0 m15 = 0
m16 = 0 m17 = 0 m18 = 0 m19 = 0
m20 = 0 m21 = 0 m22 = 0 m23 = 0
m24 = 0 m25 = 0 m26 = 0 m27 = 0
m28 = 0 m29 = 0 m30 = 0 m31 = 0

```

檔案(F)	編輯(E)	格式(O)	檢視(V)	說明
r0 =	0	r1 =	16	
r2 =	20	r3 =	8	
r4 =	16	r5 =	8	
r6 =	24	r7 =	26	
r8 =	8	r9 =	100	
r10 =	0	r11 =	0	
r12 =	0	r13 =	0	
r14 =	0	r15 =	0	
r16 =	0	r17 =	0	
r18 =	0	r19 =	0	
r20 =	0	r21 =	0	
r22 =	0	r23 =	0	
r24 =	0	r25 =	0	
r26 =	0	r27 =	0	
r28 =	0	r29 =	0	
r30 =	0	r31 =	0	
m0 =	0	m1 =	16	
m2 =	0	m3 =	0	
m4 =	0	m5 =	0	
m6 =	0	m7 =	0	
m8 =	0	m9 =	0	
m10 =	0	m11 =	0	
m12 =	0	m13 =	0	
m14 =	0	m15 =	0	
m16 =	0	m17 =	0	
m18 =	0	m19 =	0	
m20 =	0	m21 =	0	
m22 =	0	m23 =	0	
m24 =	0	m25 =	0	
m26 =	0	m27 =	0	
m28 =	0	m29 =	0	
m30 =	0	m31 =	0	

same as the CO_P4_test_3_result

Co_P4_test_3_hazard.txt after modify:



Solve the hazard between I1/I2:

I reorder I2 and I3 and insert a NOP between these two instruction.

Solve the hazard between I5/I6,I8/I9:

I reorder I5, I6, I7, I8.

Problems you met and solutions:

Too many variables makes me confused.

=>add stage name after almost each variables.

In Lab3, I used MUX 3 to 1 to complete my work, but in this lab, it only needs to use MUX 2 to 1.

=>used the MUX 2 to 1 of lab2.

Summary:

Compared to the previous labs, this time only need to focus on a few .v file. It's easier to find errors when debugging.

After understanding how the pipeline CPU works, it was easy to finish this lab if the submodule made before was correct.