Arch Lab5

Dynamically Scheduled Pipelines using Scoreboarding

Chenlu Miao 11/2022

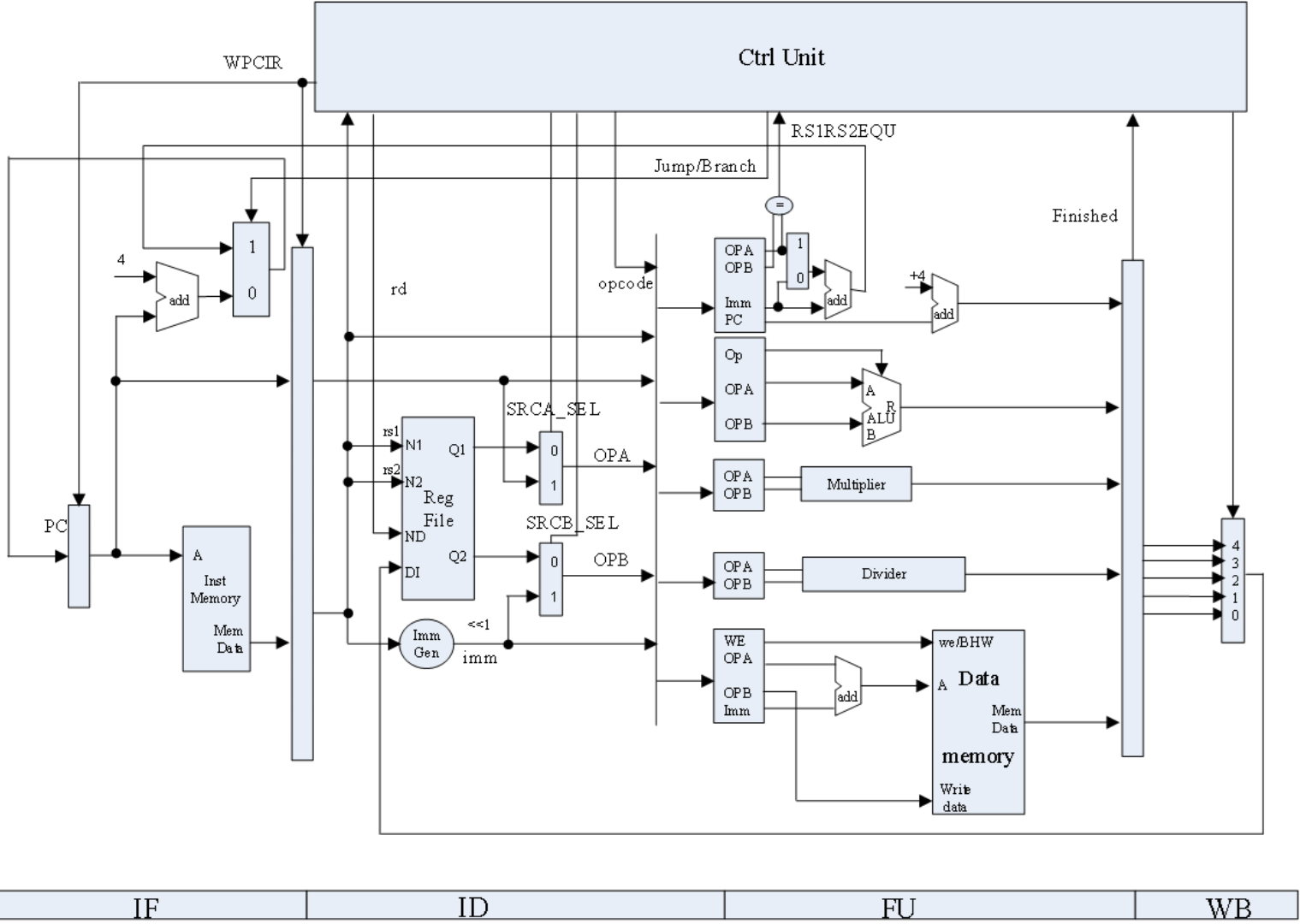
Tasks

- Redesign the pipelines with IF/ID/FU/WB stages and FU stage supporting **multicycle** operations.
- Redesign of CPU Controller.

Overview

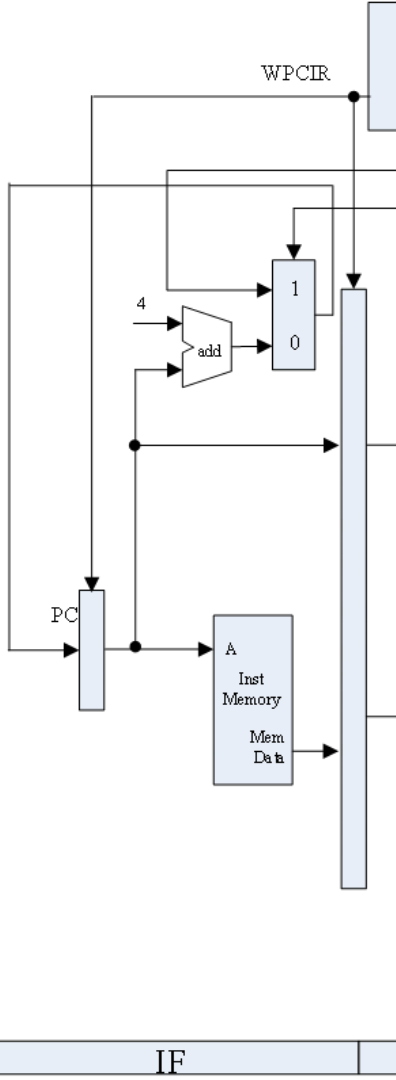
- Architecture Overview
- Control Unit
- Function Unit
- Pipelines resolving Data Hazards
- Pipelines resolving Control Hazards

Architecture Overview



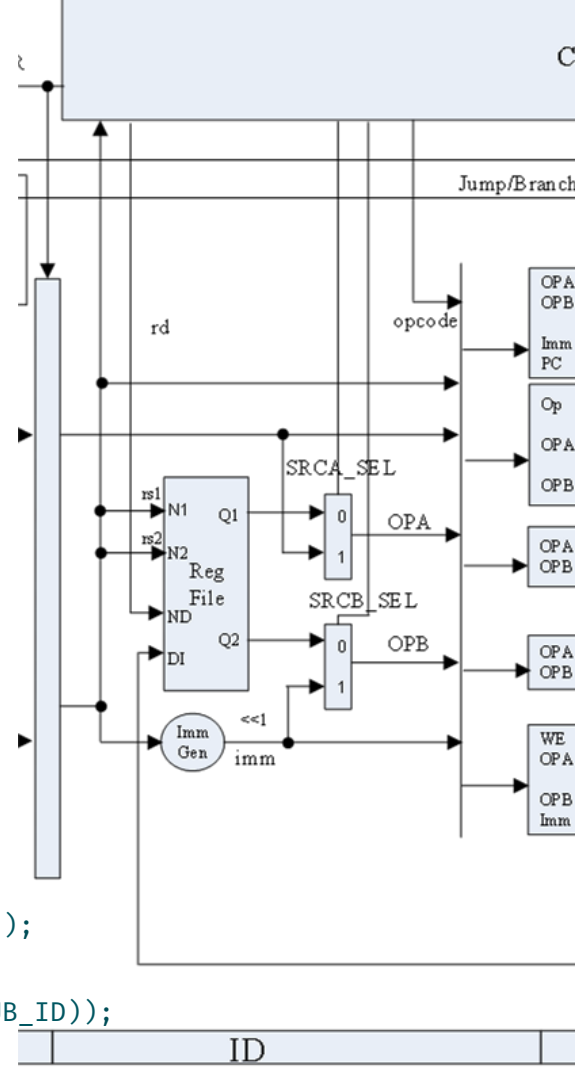
Architecture Overview – IF

```
// IF
REG32
REG_PC(.clk(debug_clk),.rst(rst),.CE(reg_IF_EN),.D(next_PC_IF),.Q(PC_IF));

add_32 add_IF(.a(PC_IF),.b(32'd4),.c(PC_4_IF));

MUX2T1_32
mux_IF(.I0(PC_4_IF),.I1(PC_jump_FU),.s(branch_ctrl),.o(next_PC_IF));

ROM_D inst_rom(.a(PC_IF[8:2]),.spo(inst_IF));
```



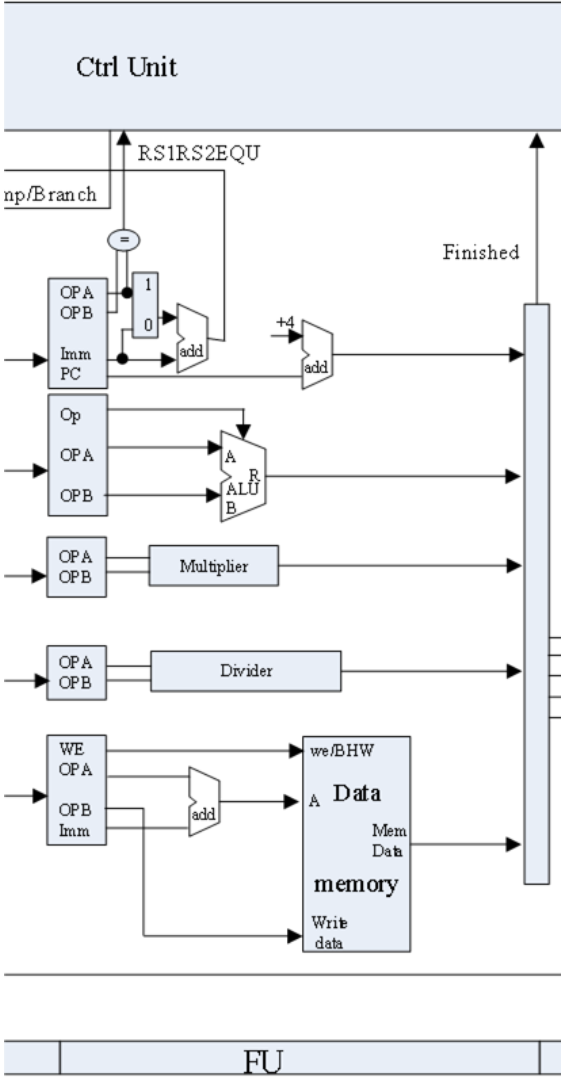
Architecture Overview – ID

```
//Issue
REG_ID reg_ID(.clk(debug_clk),.rst(rst),.EN(reg_ID_EN),
    .flush(reg_ID_flush),.PCOUT(PC_IF),.IR(inst_IF),

    .IR_ID(inst_ID),.PCurrent_ID(PC_ID),.valid(valid_ID));

CtrlUnit ctrl...

ImmGen imm_gen(.ImmSel(ImmSel_ctrl),.inst_field(inst_ID),.Imm_out(Imm_out_ID));

Regs register(.clk(debug_clk),.rst(rst),
    .R_addr_A(inst_ID[19:15]),.rdata_A(rs1_data_ID),
    .R_addr_B(inst_ID[24:20]),.rdata_B(rs2_data_ID),
    .L_S(RegWrite_ctrl),.Wt_addr(rd_ctrl),.Wt_data(wt_data_WB),
    .Debug_addr(debug_addr[4:0]),.Debug_regs(debug_regs));

MUX2T1_32 mux_imm_ALU_ID_A(.I0(rs1_data_ID),.I1(PC_ID),.s(ALUSrcA_ctrl),.o(ALUA_ID));

MUX2T1_32 mux_imm_ALU_ID_B(.I0(rs2_data_ID),.I1(Imm_out_ID),.s(ALUSrcB_ctrl),.o(ALUB_ID));
```



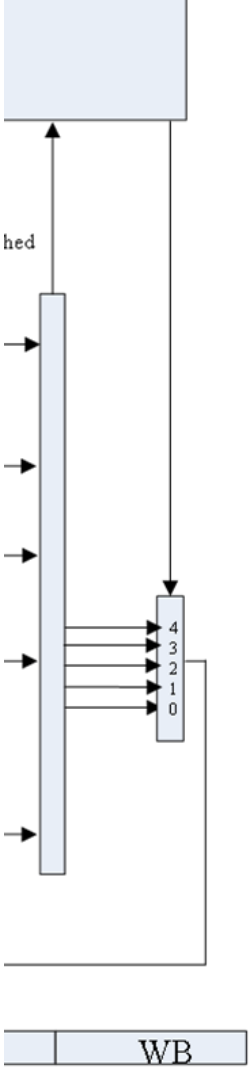
Architecture Overview – FU

```
// FU
FU_ALU alu(...);

FU_mem mem(...);

FU_mul mu(...);

FU_div du(...);

FU_jump ju(...);
```



Architecture Overview – WB

```
// WB
REG32 reg_WB_ALU(.clk(debug_clk),.rst(rst),.CE(FU_ALU_finish),.D(ALUout_FU),.Q(ALUout_WB));

REG32
reg_WB_mem(.clk(debug_clk),.rst(rst),.CE(FU_mem_finish),.D(mem_data_FU),.Q(mem_data_WB));

REG32 reg_WB_mul(.clk(debug_clk),.rst(rst),.CE(FU_mul_finish),.D(mulres_FU),.Q(mulres_WB));

REG32 reg_WB_div(.clk(debug_clk),.rst(rst),.CE(FU_div_finish),.D(divres_FU),.Q(divres_WB));

REG32 reg_WB_jump(.clk(debug_clk),.rst(rst),.CE(FU_jump_finish),.D(PC_wb_FU),.Q(PC_wb_WB));

MUX8T1_32
mux_DtR(.s(DatatoReg_ctrl),.I0(32'd0),.I1(ALUout_WB),.I2(mem_data_WB),.I3(mulres_WB),
    .I4(divres_WB),.I5(PC_wb_WB),.I6(32'd0),.I7(32'd0),.o(wt_data_WB));
```



Control Unit

```
module CtrlUnit(
    input clk,
    input rst,

    input[31:0] inst,
    input valid_ID,

    input ALU_done,
    input MEM_done,
    input MUL_done,
    input DIV_done,
    input JUMP_done,
    input cmp_res_FU,

    // IF
    output reg_IF_en, branch_ctrl,

    // ID
    output reg_ID_en, reg_ID_flush,
    output[2:0] ImmSel,
    output ALU_en, MEM_en, MUL_en, DIV_en, JUMP_en,

    // FU
    output[3:0] JUMP_op,
    output[3:0] ALU_op,
    output ALUSrcA,
    output ALUSrcB,
    output MEM_we,

    // WB
    output reg[2:0] write_sel,
    output reg[4:0] rd_ctrl,
    output reg reg_write
);
```

Function Unit – ALU

```
module FU_ALU(
    input clk, EN,
    input[3:0] ALUControl,
    input[31:0] ALUA, ALUB,
    output[31:0] res,
    output zero, overflow,
    output finish
);

    reg state;
    assign finish = state == 1'b1;
    initial begin
        state = 0;
    end
```

```
reg[3:0] Control;
reg[31:0] A, B;

always@(posedge clk) begin
    if(EN & ~state) begin // state == 0
        A <= ...;
        B <= ...;
        Control <= ...;
        state <= 1;
    end
    else state <= 0;
end
```

Function Unit – MUL

```
module FU_mul(
    input clk, EN,
    input[31:0] A, B,
    output[31:0] res,
    output finish
);

    reg[6:0] state;
    assign finish = state[0] == 1'b1;
    initial begin
        state = 0;
    end

    reg[31:0] A_reg, B_reg;
```

```
    always@(posedge clk) begin
        if(EN & ~|state) begin
            A_reg ...
            B_reg ...
            state ...
        end
        else state <= {1'b0, state[6:1]};
    end

    wire[63:0] mulres;

    multiplier
mul(.CLK(clk),.A(A_reg),.B(B_reg),.P(mulres));

    assign res = mulres[31:0];

endmodule
```

Function Unit – DIV

```
module FU_div(
    input clk, EN,
    input[31:0] A, B,
    output[31:0] res,
    output finish
);

    wire res_valid;
    wire[63:0] divres;

    reg state;
    assign finish = res_valid & state;
    initial begin
        state = 0;
    end

    reg A_valid, B_valid;
    reg[31:0] A_reg, B_reg;

    always@(posedge clk) begin
        if(EN & ~state) begin  // state == 0
            A_reg ...
            B_reg ...
            A_valid ...
            B_valid ...
            state ...
        end
        else if(res_valid) begin
            A_valid ...
            B_valid ...
            state ...
        end
    end


    divider div(.aclk(clk),
        .s_axis_dividend_tvalid(A_valid),
        .s_axis_dividend_tdata(A_reg),
        .s_axis_divisor_tvalid(B_valid),
        .s_axis_divisor_tdata(B_reg),
        .m_axis_dout_tvalid(res_valid),
        .m_axis_dout_tdata(divres)
    );

    assign res = divres[63:32];

endmodule
```

Function Unit – JUMP

```
module FU_jump(
    input clk, EN, JALR,
    input[2:0] cmp_ctrl,
    input[31:0] rs1_data, rs2_data, imm,
PC,
    output[31:0] PC_jump, PC_wb,
    output cmp_res, finish
);

    reg state;
    assign finish = state == 1'b1;
    initial begin
        state = 0;
    end

    reg JALR_reg;
    reg[2:0] cmp_ctrl_reg;
    reg[31:0] rs1_data_reg, rs2_data_reg,
imm_reg, PC_reg;
```

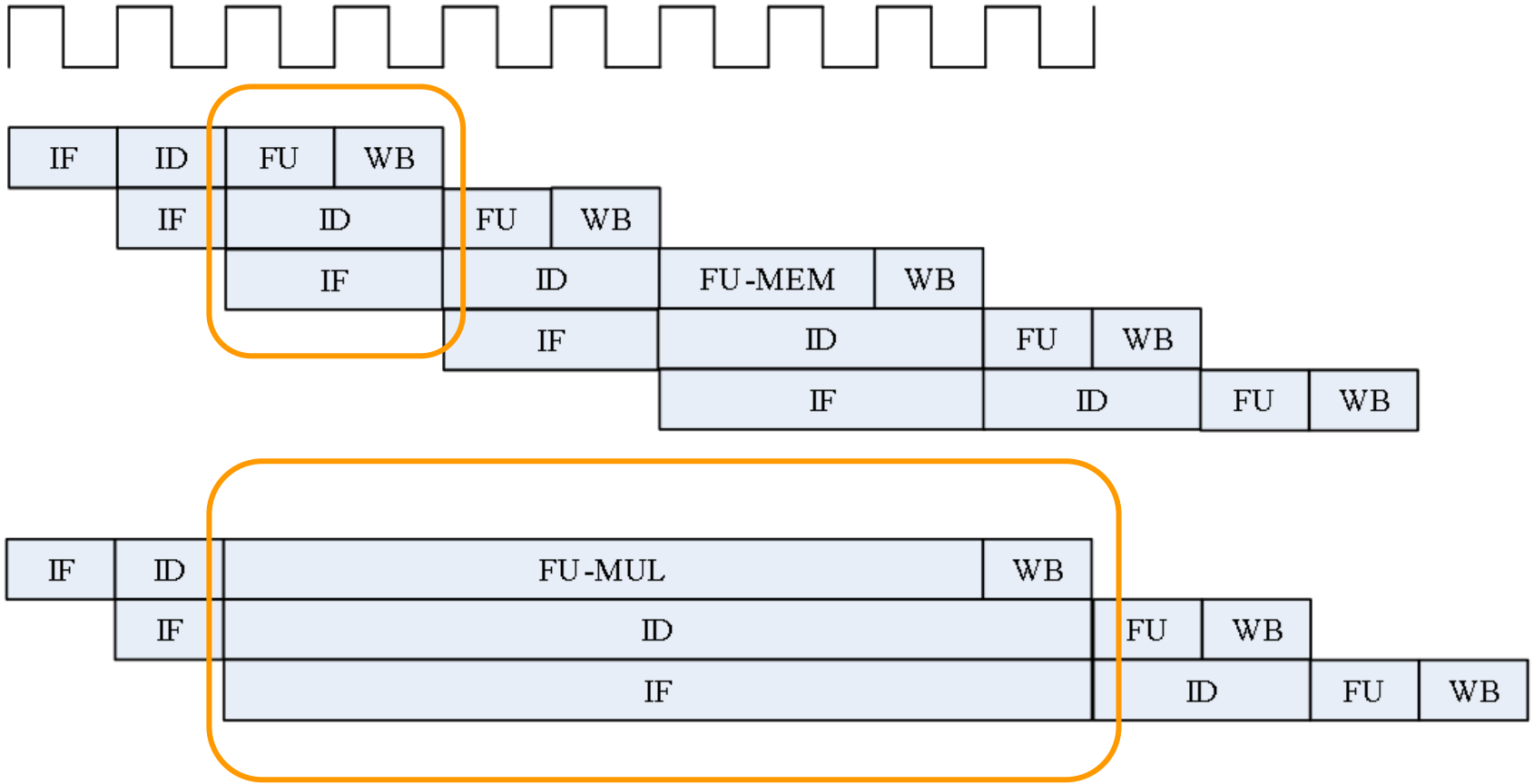
```
    always@(posedge clk) begin
        if(EN & ~state) begin // state == 0
            JALR_reg ...
            cmp_ctrl_reg ...
            rs1_data_reg ...
            rs2_data_reg ...
            imm_reg ...
            PC_reg ...
            state ...
        end
        else state ...
    end

    cmp_32 cmp ...

    add_32 a...

    add_32 b...

endmodule
```

Function Unit – MEM

```
module FU_mem(
    input clk, EN, mem_w,
    input[2:0] bhw,
    input[31:0] rs1_data, rs2_data, imm,
    output[31:0] mem_data,
    output finish
);

    reg[1:0] state;
    assign finish = state[0] == 1'b1;
    initial begin
        state = 0;
    end

    reg mem_w_reg;
    reg[2:0] bhw_reg;
    reg[31:0] rs1_data_reg, rs2_data_reg,
imm_reg;
```

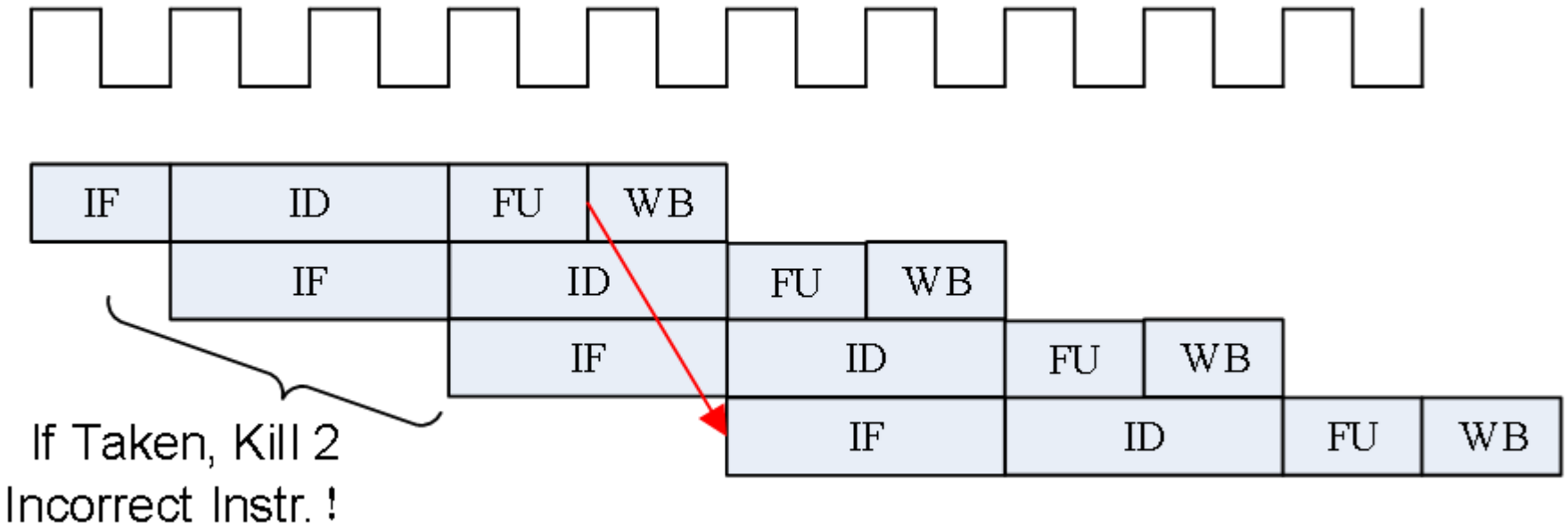
```
    always@(posedge clk) begin
        if(EN & ~|state) begin
            mem_w_reg ...
            bhw_reg ...
            rs1_data_reg ...
            rs2_data_reg ...
            imm_reg ...
            state ...
        end
        else state ...
    end

    wire[31:0] addr;

    add_32 add...

    RAM_B
ram(.clka(clk),.addra(addr),.dina(rs2_data_reg),
.wea(mem_w_reg),
        .douta(mem_data),.mem_u_b_h_w(bhw_reg));

endmodule
```

Pipelines resolving Data Hazards



Pipelines resolving Control Hazards

Predict-not-taken
Condition and Addr. Calculation in FU



ROM

NO.	Instruction	Addr.	Label	ASM	Comment
0	00000013	0	__start:	addi x0, x0, 0	
1	00402103	4		lw x2, 4(x0)	
2	00802203	8		lw x4, 8(x0)	
3	004100b3	C		add x1, x2, x4	
4	fff08093	10		addi x1, x1, -1	
5	00c02283	14		lw x5, 12(x0)	
6	01002303	18		lw x6, 16(x0)	
7	01402383	1C		lw x7, 20(x0)	
8	402200b3	20		sub x1,x4,x2	
9	ffd50093	24		addi x1,x10,-3	
10	00520c63	28		beq x4,x5,label0	
11	00420a63	2C		beq x4,x4,label0	
12	00000013	30		addi x0,x0,0	
13	00000013	34		addi x0,x0,0	
14	00000013	38		addi x0,x0,0	

ROM

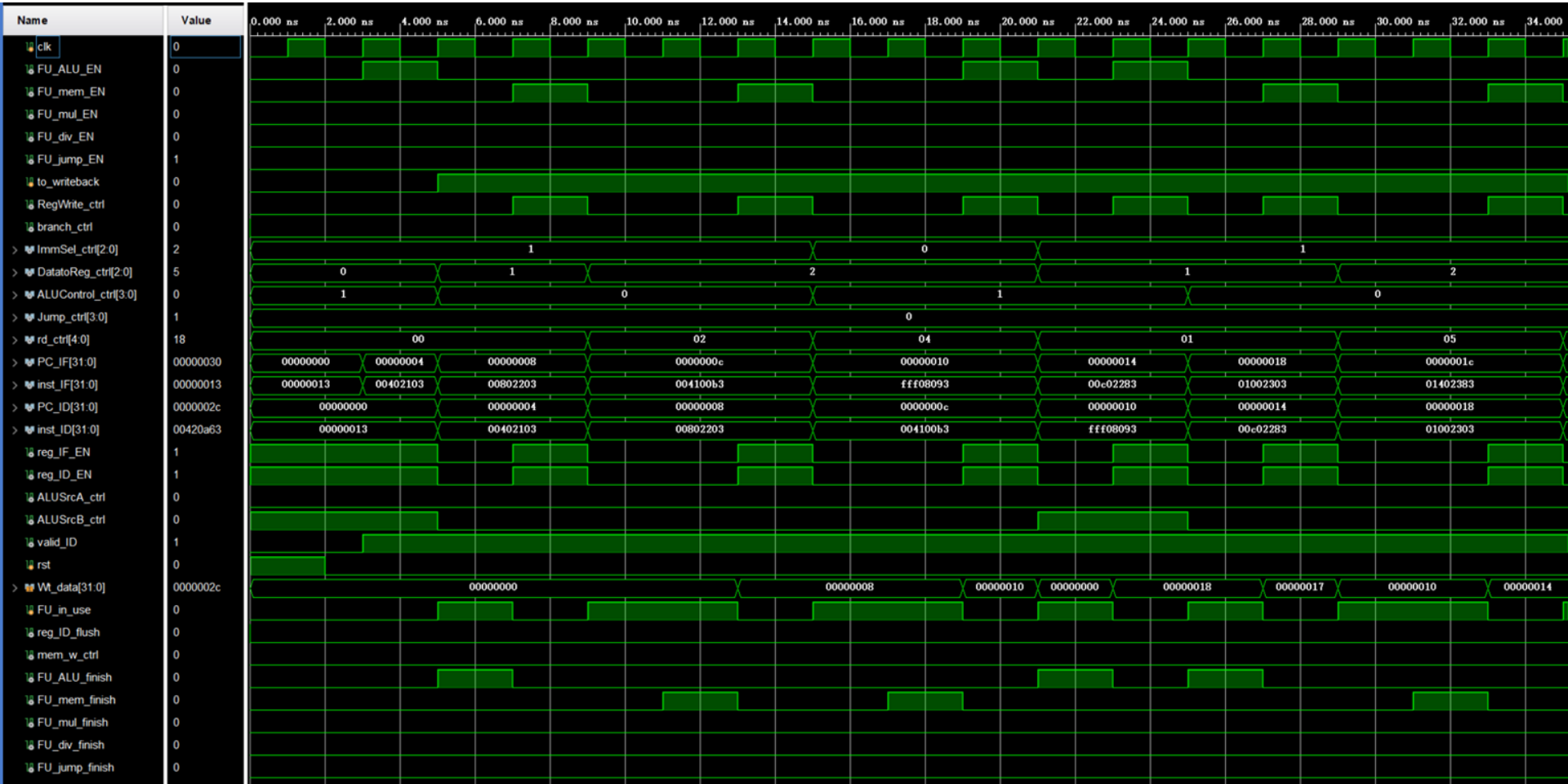
NO.	Instruction	Addr.	Label	ASM	Comment
15	00000013	3C		addi x0,x0,0	
16	000040b7	40	label0:	lui x1,4	
17	00c000ef	44		jal x1,12	
18	00000013	48		addi x0,x0,0	
19	00000013	4C		addi x0,x0,0	
20	00000013	50		addi x0,x0,0	
21	00000013	54		addi x0,x0,0	
22	ffff0097	58		auipc x1, 0xffff0	
23	0223c433	5C		div x8, x7, x2	
24	025204b3	60		mul x9, x4, x5	
25	022404b3	64		mul x9, x8, x2	
26	00400113	68		addi x2, x0, 4	
27	000000e7	6C		jalr x1,0(x0)	

RAM

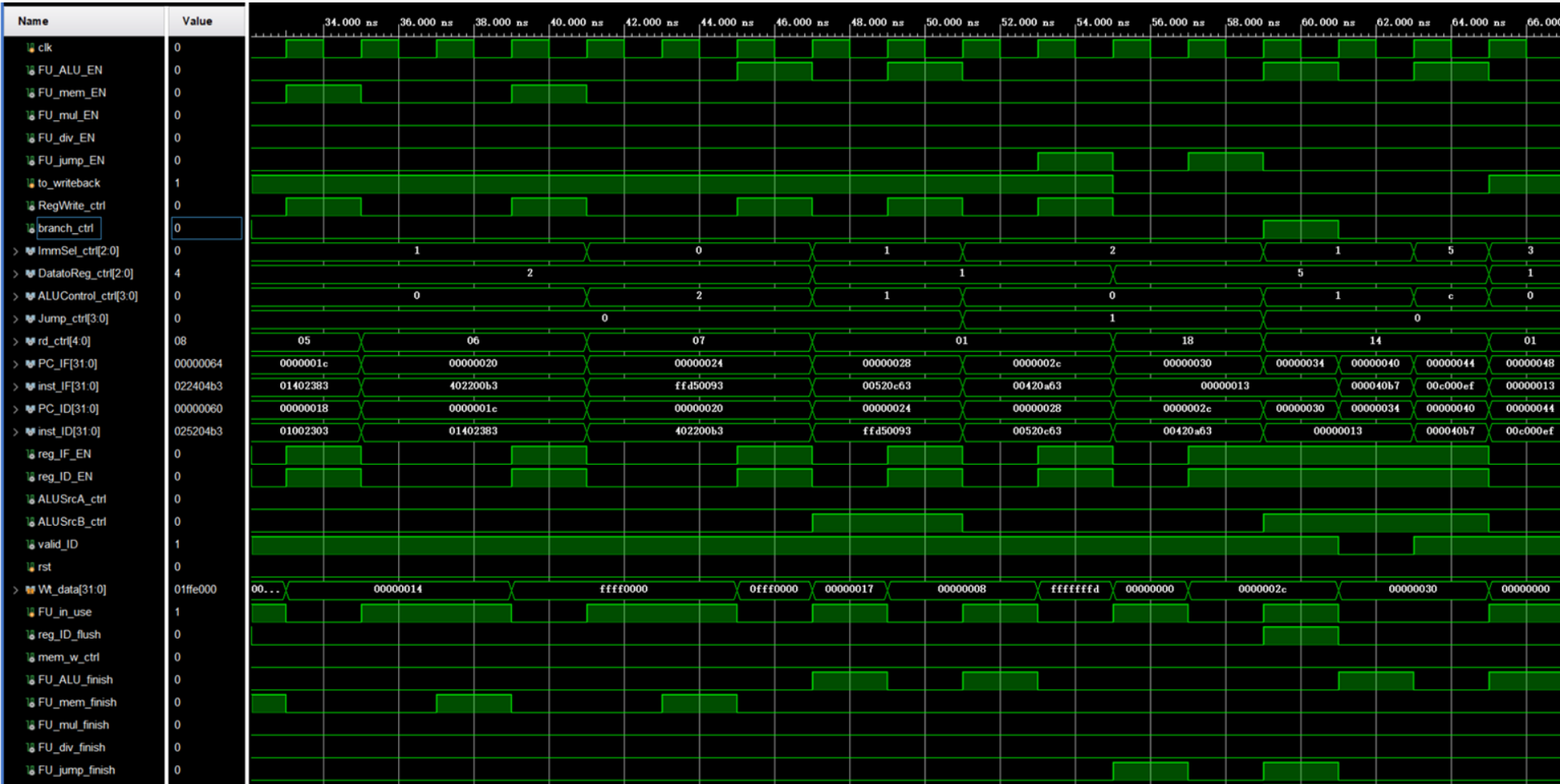
NO.	Data	Addr.
0	000080BF	0
1	00000008	4
2	00000010	8
3	00000014	C
4	FFFF0000	10
5	0FFF0000	14
6	FF000F0F	18
7	F0F0F0F0	1C
8	00000000	20
9	00000000	24
10	00000000	28
11	00000000	2C
12	00000000	30
13	00000000	34
14	00000000	38
15	00000000	3C

NO.	Instruction	Addr.
16	00000000	40
17	00000000	44
18	00000000	48
19	00000000	4C
20	A3000000	50
21	27000000	54
22	79000000	58
23	15100000	5C
24	00000000	60
25	00000000	64
26	00000000	68
27	00000000	6C
28	00000000	70
29	00000000	74
30	00000000	78
31	00000000	7C

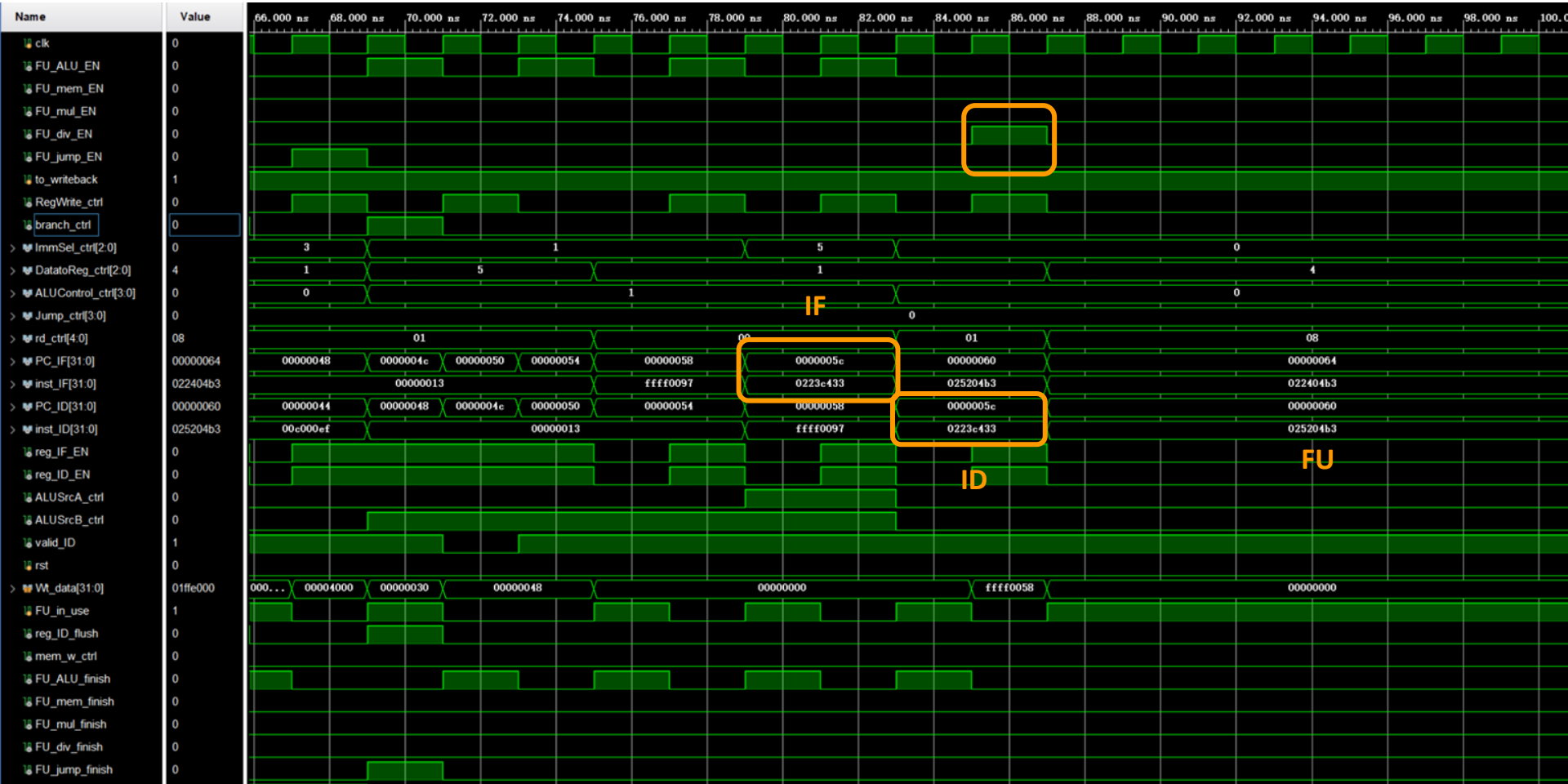
Simulation


Name
Value
clk 0
FU_ALU_EN 0
FU_mem_EN 0
FU_mul_EN 0
FU_div_EN 0
FU_jump_EN 1
to_writeback 0
RegWrite_ctrl 0
branch_ctrl 0
ImmSel_ctrl[2:0] 2
DatatoReg_ctrl[2:0] 5
ALUControl_ctrl[3:0] 0
Jump_ctrl[3:0] 1
rd_ctrl[4:0] 18
PC_IF[31:0] 00000030
inst_IF[31:0] 00000013
PC_ID[31:0] 0000002c
inst_ID[31:0] 00420a63
reg_IF_EN 1
reg_ID_EN 1
ALUSrcA_ctrl 0
ALUSrcB_ctrl 0
valid_ID 1
rst 0
Wt_data[31:0] 0000002c
FU_in_use 0
reg_ID_flush 0
mem_w_ctrl 0
FU_ALU_finish 0
FU_mem_finish 0
FU_mul_finish 0
FU_div_finish 0
FU_jump_finish 0
0.000 ns
2.000 ns
4.000 ns
6.000 ns
8.000 ns
10.000 ns
12.000 ns
14.000 ns
16.000 ns
18.000 ns
20.000 ns
22.000 ns
24.000 ns
26.000 ns
28.000 ns
30.000 ns
32.000 ns
34.000
00000000 00000004 00000008 0000000c 00000010 00000014 00000018 0000001c
00000013 00402103 00802203 004100b3 fff08093 00c02283 01002303 01402383
00000000 00000004 00000008 0000000c 00000010 00000014 00000018
00000013 00402103 00802203 004100b3 fff08093 00c02283 01002303
00000000 00000008 00000010 00000000 00000018 00000017 00000010 00000014

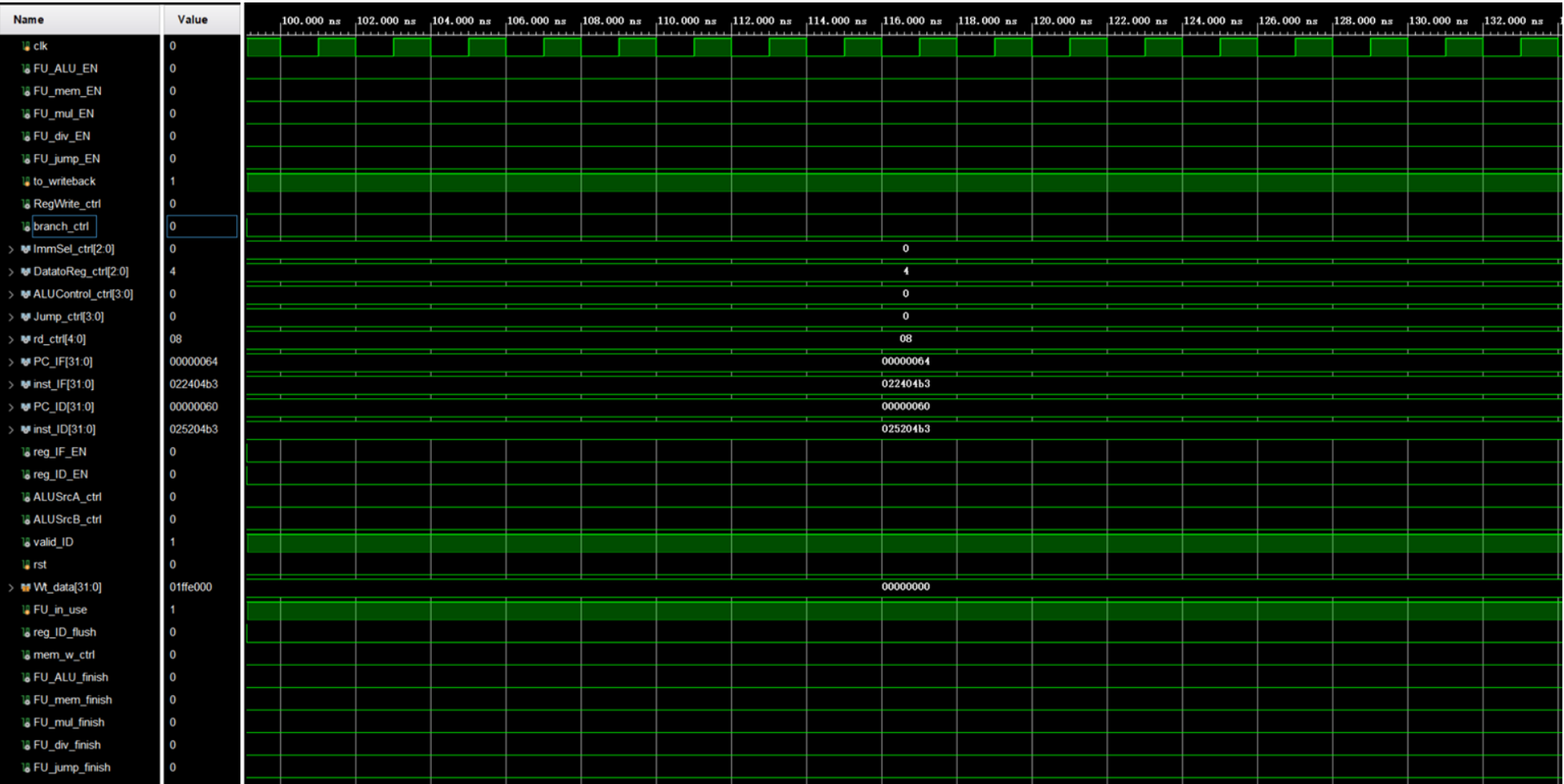
Simulation


Name
Value
34.000 ns
36.000 ns
38.000 ns
40.000 ns
42.000 ns
44.000 ns
46.000 ns
48.000 ns
50.000 ns
52.000 ns
54.000 ns
56.000 ns
58.000 ns
60.000 ns
62.000 ns
64.000 ns
66.000
clk 0
FU_ALU_EN 0
FU_mem_EN 0
FU_mul_EN 0
FU_div_EN 0
FU_jump_EN 0
to_writeback 1
RegWrite_ctrl 0
branch_ctrl 0
ImmSel_ctrl[2:0] 0
DatatoReg_ctrl[2:0] 4
ALUControl_ctrl[3:0] 0
Jump_ctrl[3:0] 0
rd_ctrl[4:0] 08
PC_IF[31:0] 00000064
inst_IF[31:0] 022404b3
PC_ID[31:0] 00000060
inst_ID[31:0] 025204b3
reg_IF_EN 0
reg_ID_EN 0
ALUSrcA_ctrl 0
ALUSrcB_ctrl 0
valid_ID 1
rst 0
Wt_data[31:0] 01ffe000
FU_in_use 1
reg_ID_flush 0
mem_w_ctrl 0
FU_ALU_finish 0
FU_mem_finish 0
FU_mul_finish 0
FU_div_finish 0
FU_jump_finish 0
1 0 1 2 1 5 3
2 1 5 1
0 2 1 0 1 c 0
0 1 0
05 06 07 01 18 14 01
0000001c 00000020 00000024 00000028 0000002c 00000030 00000034 00000040 00000044 00000048
01402383 402200b3 ffd50093 00520c63 00420a63 00000013 000040b7 00c000ef 00000013
00000018 0000001c 00000020 00000024 00000028 0000002c 00000030 00000034 00000040 00000044
01002303 01402383 402200b3 ffd50093 00520c63 00420a63 00000013 000040b7 00c000ef
00... 00000014 ffff0000 0fff0000 00000017 00000008 fffffffd 00000000 0000002c 00000030 00000000

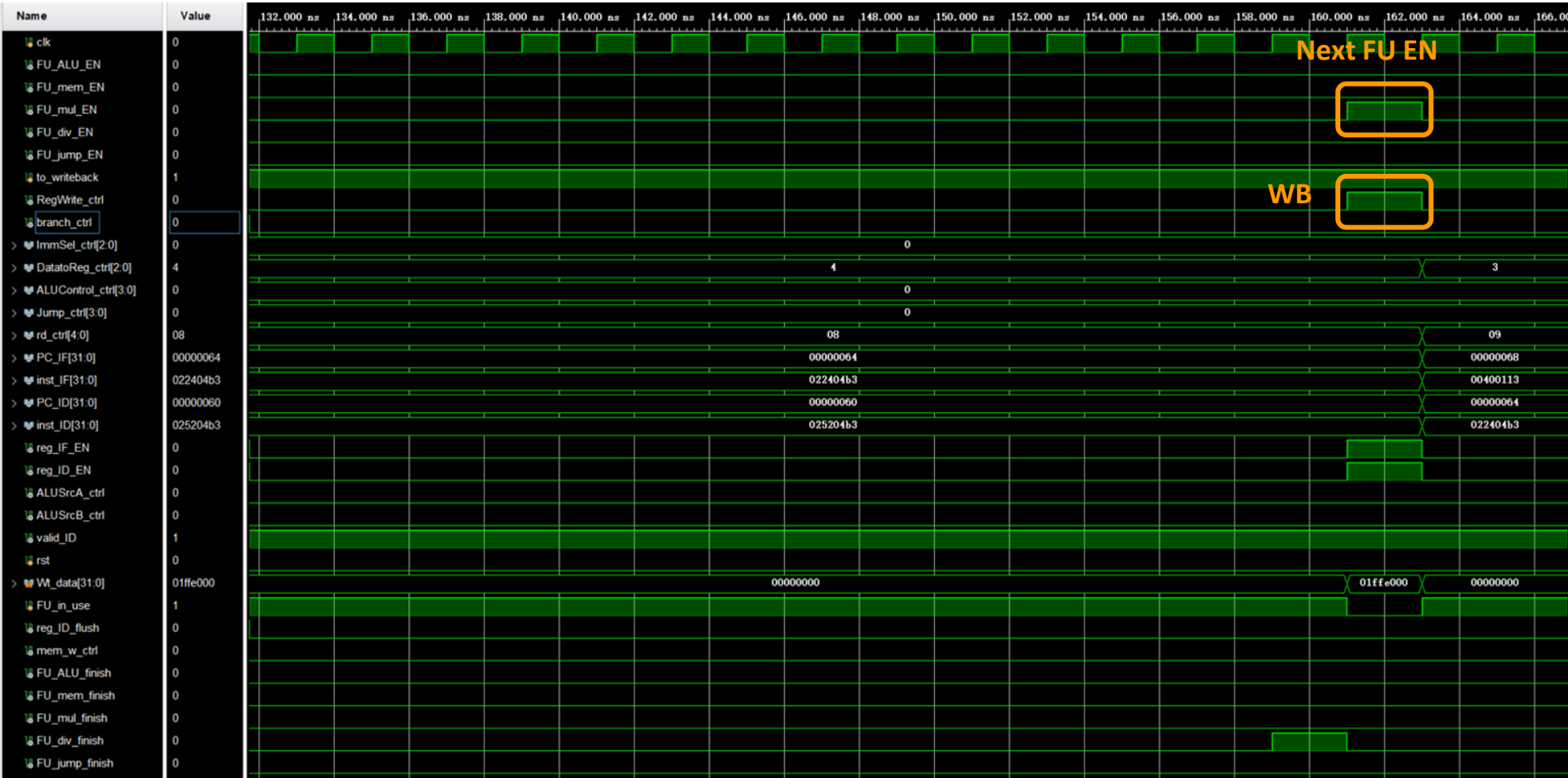
Simulation


Name
Value
clk
FU_ALU_EN
FU_mem_EN
FU_mul_EN
FU_div_EN
FU_jump_EN
to_writeback
RegWrite_ctrl
branch_ctrl
ImmSel_ctrl[2:0]
DatatoReg_ctrl[2:0]
ALUControl_ctrl[3:0]
Jump_ctrl[3:0]
rd_ctrl[4:0]
PC_IF[31:0]
inst_IF[31:0]
PC_ID[31:0]
inst_ID[31:0]
reg_IF_EN
reg_ID_EN
ALUSrcA_ctrl
ALUSrcB_ctrl
valid_ID
rst
Wt_data[31:0]
FU_in_use
reg_ID_flush
mem_w_ctrl
FU_ALU_finish
FU_mem_finish
FU_mul_finish
FU_div_finish
FU_jump_finish
IF
ID
FU

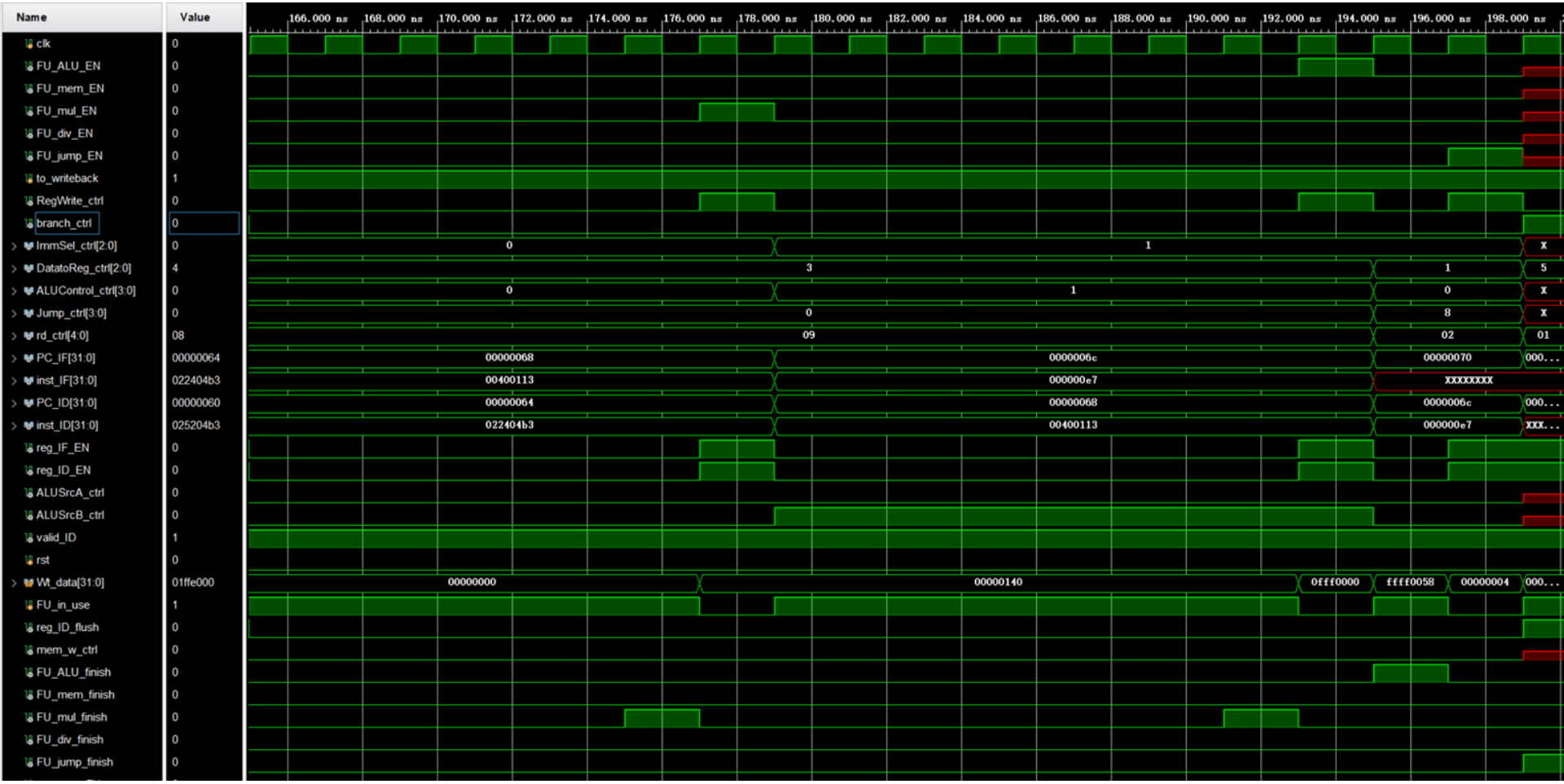
Simulation


Name
Value
clk 0
FU_ALU_EN 0
FU_mem_EN 0
FU_mul_EN 0
FU_div_EN 0
FU_jump_EN 0
to_writeback 1
RegWrite_ctrl 0
branch_ctrl 0
ImmSel_ctrl[2:0] 0
DatatoReg_ctrl[2:0] 4
ALUControl_ctrl[3:0] 0
Jump_ctrl[3:0] 0
rd_ctrl[4:0] 08
PC_IF[31:0] 00000064
inst_IF[31:0] 022404b3
PC_ID[31:0] 00000060
inst_ID[31:0] 025204b3
reg_IF_EN 0
reg_ID_EN 0
ALUSrcA_ctrl 0
ALUSrcB_ctrl 0
valid_ID 1
rst 0
Wt_data[31:0] 01ffe000
FU_in_use 1
reg_ID_flush 0
mem_w_ctrl 0
FU_ALU_finish 0
FU_mem_finish 0
FU_mul_finish 0
FU_div_finish 0
FU_jump_finish 0
100.000 ns
102.000 ns
104.000 ns
106.000 ns
108.000 ns
110.000 ns
112.000 ns
114.000 ns
116.000 ns
118.000 ns
120.000 ns
122.000 ns
124.000 ns
126.000 ns
128.000 ns
130.000 ns
132.000 ns
0
4
0
0
08
00000064
022404b3
00000060
025204b3
00000000

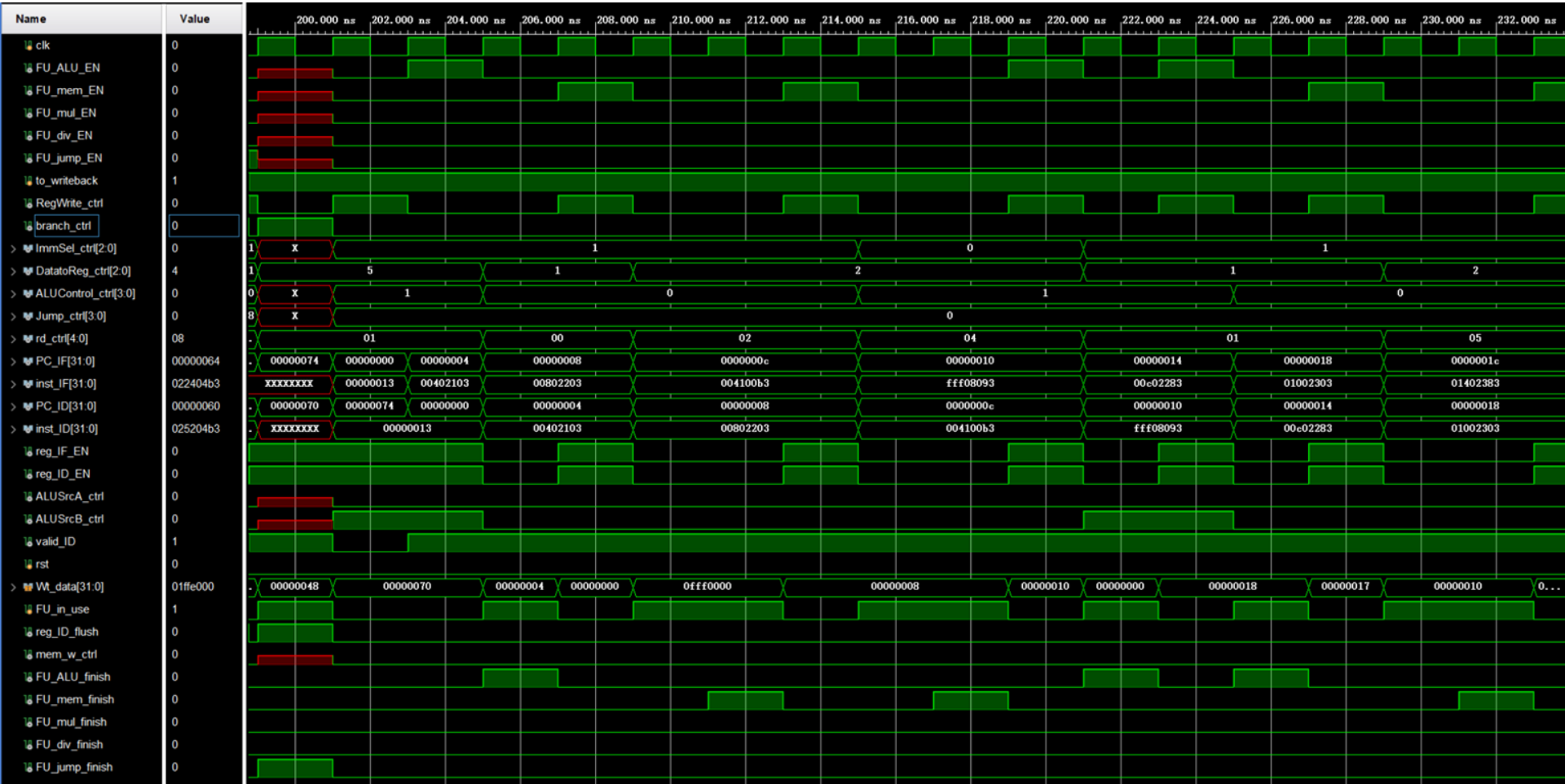
Simulation


Name
Value
clk
FU_ALU_EN
FU_mem_EN
FU_mul_EN
FU_div_EN
FU_jump_EN
to_writeback
RegWrite_ctrl
branch_ctrl
ImmSel_ctrl[2:0]
DatatoReg_ctrl[2:0]
ALUControl_ctrl[3:0]
Jump_ctrl[3:0]
rd_ctrl[4:0]
PC_IF[31:0]
inst_IF[31:0]
PC_ID[31:0]
inst_ID[31:0]
reg_IF_EN
reg_ID_EN
ALUSrcA_ctrl
ALUSrcB_ctrl
valid_ID
rst
Wt_data[31:0]
FU_in_use
reg_ID_flush
mem_w_ctrl
FU_ALU_finish
FU_mem_finish
FU_mul_finish
FU_div_finish
FU_jump_finish
0
0
0
0
0
0
1
0
0
0
4
0
0
08
00000064
022404b3
00000060
025204b3
0
0
1
0
01ffe000
1
0
0
0
0
0
0
0
132.000 ns
134.000 ns
136.000 ns
138.000 ns
140.000 ns
142.000 ns
144.000 ns
146.000 ns
148.000 ns
150.000 ns
152.000 ns
154.000 ns
156.000 ns
158.000 ns
160.000 ns
162.000 ns
164.000 ns
Next FU EN
WB
0
4
3
0
0
08
09
00000064
00000068
022404b3
00400113
00000060
00000064
025204b3
022404b3
00000000
01ffe000
00000000

Simulation


Name
Value
166.000 ns
168.000 ns
170.000 ns
172.000 ns
174.000 ns
176.000 ns
178.000 ns
180.000 ns
182.000 ns
184.000 ns
186.000 ns
188.000 ns
190.000 ns
192.000 ns
194.000 ns
196.000 ns
198.000 ns
clk 0
FU_ALU_EN 0
FU_mem_EN 0
FU_mul_EN 0
FU_div_EN 0
FU_jump_EN 0
to_writeback 1
RegWrite_ctrl 0
branch_ctrl 0
ImmSel_ctrl[2:0] 0
DatatoReg_ctrl[2:0] 4
ALUControl_ctrl[3:0] 0
Jump_ctrl[3:0] 0
rd_ctrl[4:0] 08
PC_IF[31:0] 00000064
inst_IF[31:0] 022404b3
PC_ID[31:0] 00000060
inst_ID[31:0] 025204b3
reg_IF_EN 0
reg_ID_EN 0
ALUSrcA_ctrl 0
ALUSrcB_ctrl 0
valid_ID 1
rst 0
Wt_data[31:0] 01ffe000
FU_in_use 1
reg_ID_flush 0
mem_w_ctrl 0
FU_ALU_finish 0
FU_mem_finish 0
FU_mul_finish 0
FU_div_finish 0
FU_jump_finish 0
0
1
X
3
1
5
0
1
0
X
0
8
X
09
02
01
00000068
0000006c
00000070
000...
00400113
000000e7
XXXXXXXX
00000064
00000068
0000006c
000...
022404b3
00400113
000000e7
XXX...
00000000
00000140
0fff0000
ffff0058
00000004
000...

Simulation


Name
Value
200.000 ns
202.000 ns
204.000 ns
206.000 ns
208.000 ns
210.000 ns
212.000 ns
214.000 ns
216.000 ns
218.000 ns
220.000 ns
222.000 ns
224.000 ns
226.000 ns
228.000 ns
230.000 ns
232.000 ns
clk 0
FU_ALU_EN 0
FU_mem_EN 0
FU_mul_EN 0
FU_div_EN 0
FU_jump_EN 0
to_writeback 1
RegWrite_ctrl 0
branch_ctrl 0
ImmSel_ctrl[2:0] 0
DatatoReg_ctrl[2:0] 4
ALUControl_ctrl[3:0] 0
Jump_ctrl[3:0] 0
rd_ctrl[4:0] 08
PC_IF[31:0] 00000064
inst_IF[31:0] 022404b3
PC_ID[31:0] 00000060
inst_ID[31:0] 025204b3
reg_IF_EN 0
reg_ID_EN 0
ALUSrcA_ctrl 0
ALUSrcB_ctrl 0
valid_ID 1
rst 0
Wt_data[31:0] 01ffe000
FU_in_use 1
reg_ID_flush 0
mem_w_ctrl 0
FU_ALU_finish 0
FU_mem_finish 0
FU_mul_finish 0
FU_div_finish 0
FU_jump_finish 0
00000074 00000000 00000004 00000008 0000000c 00000010 00000014 00000018 0000001c
xxxxxxxx 00000013 00402103 00802203 004100b3 fff08093 00c02283 01002303 01402383
00000070 00000074 00000000 00000004 00000008 0000000c 00000010 00000014 00000018
xxxxxxxx 00000013 00402103 00802203 004100b3 fff08093 00c02283 01002303
00000048 00000070 00000004 00000000 0fff0000 00000008 00000010 00000000 00000018 00000017 00000010