EE533_Lab9_Report

1. Hardware Augmentation

1.1 New Introduced Components in Pipeline

1.1.1 PCX

- PC0
  - Verilog

```
`timescale 1ns / 1ps

module PC0
(
    input clk,
    input rst,
    input PC_ctrl,
    input [6:0] PC_next,
    input [1:0] thread,
    input [1:0] thread_ID,

    output reg [6:0] PC
);

    always @(posedge clk) begin
        if (rst)
            PC <= 0;
        else if ((thread == 2'b00) || ((thread_ID == 2'b00) && PC_ctrl))
            PC <= PC_next;
    end

endmodule
```

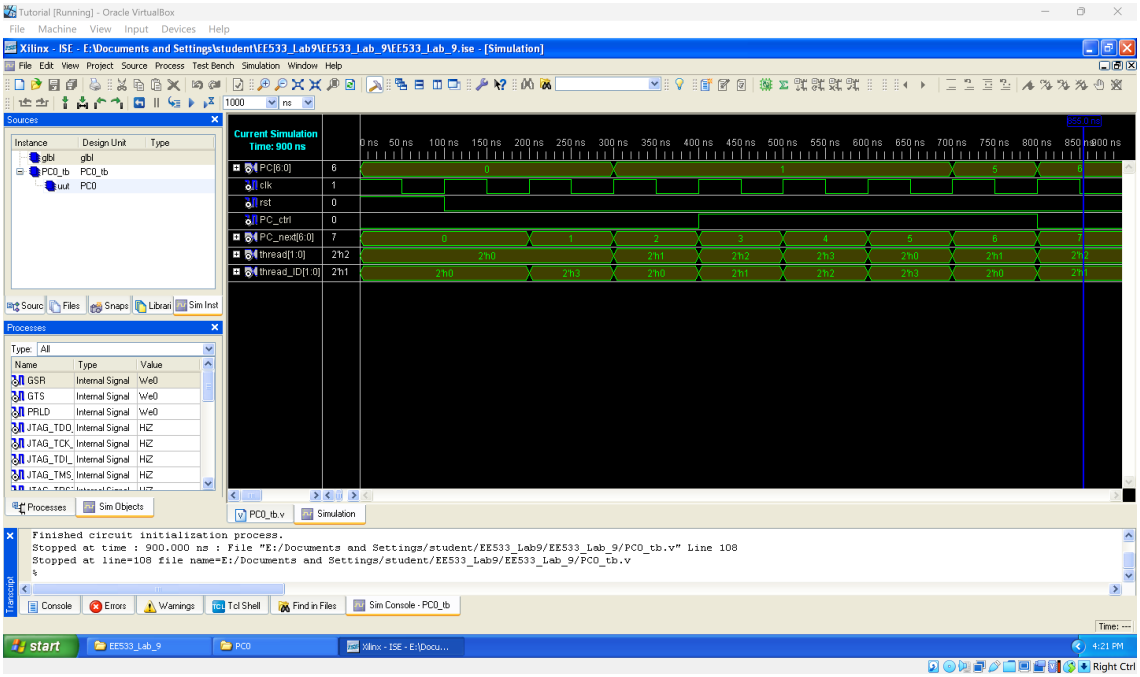
  - Testbench

```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////
///
// Company:
// Engineer:
//
// Create Date:   16:05:28 03/13/2025
// Design Name:   PC0
// Module Name:   E:/Documents and
Settings/student/EE533_Lab9/EE533_Lab_9/PC0_tb.v
// Project Name:  EE533_Lab_9
// Target Device:
// Tool versions:
// Description:
//
```

```
// Verilog Test Fixture created by ISE for module: PC0
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
///

module PC0_tb;

    // Inputs
    reg clk;
    reg rst;
    reg PC_ctrl;
    reg [6:0] PC_next;
    reg [1:0] thread;
    reg [1:0] thread_ID;

    // Outputs
    wire [6:0] PC;

    // Instantiate the Unit Under Test (UUT)
    PC0 uut (
        .clk(clk),
        .rst(rst),
        .PC_ctrl(PC_ctrl),
        .PC_next(PC_next),
        .thread(thread),
        .thread_ID(thread_ID),
        .PC(PC)
    );

    always #50 clk = ~clk;

    initial begin
        // Initialize Inputs
        clk = 1;
        rst = 1;
        PC_ctrl = 0;
        PC_next = 0;
        thread = 0;
        thread_ID = 0;

        // Wait 100 ns for global reset to finish
        @(posedge clk);
        rst = 0;

        // Add stimulus here
        @(posedge clk);
        PC_ctrl = 0;
        PC_next = 7'd1;
        thread = 2'b00;
        thread_ID = 2'b11;
```

```
        @(posedge clk);
        PC_ctrl = 0;
        PC_next = 7'd2;
        thread = 2'b01;
        thread_ID = 2'b00;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd3;
        thread = 2'b10;
        thread_ID = 2'b01;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd4;
        thread = 2'b11;
        thread_ID = 2'b10;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd5;
        thread = 2'b00;
        thread_ID = 2'b11;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd6;
        thread = 2'b01;
        thread_ID = 2'b00;

        @(posedge clk);
        PC_ctrl = 0;
        PC_next = 7'd7;
        thread = 2'b10;
        thread_ID = 2'b01;

        @(posedge clk);
        $stop;



    end

endmodule
```

- Waveform



- PC1
  - Verilog

```verilog
`timescale 1ns / 1ps

module PC1
(
    input clk,
    input rst,
    input PC_ctrl,
    input [6:0] PC_next,
    input [1:0] thread,
    input [1:0] thread_ID,

    output reg [6:0] PC
);

    always @(posedge clk) begin
        if (rst)
            PC <= 0;
        else if ((thread == 2'b01) || ((thread_ID == 2'b01) && PC_ctrl))
            PC <= PC_next;
    end

endmodule
```

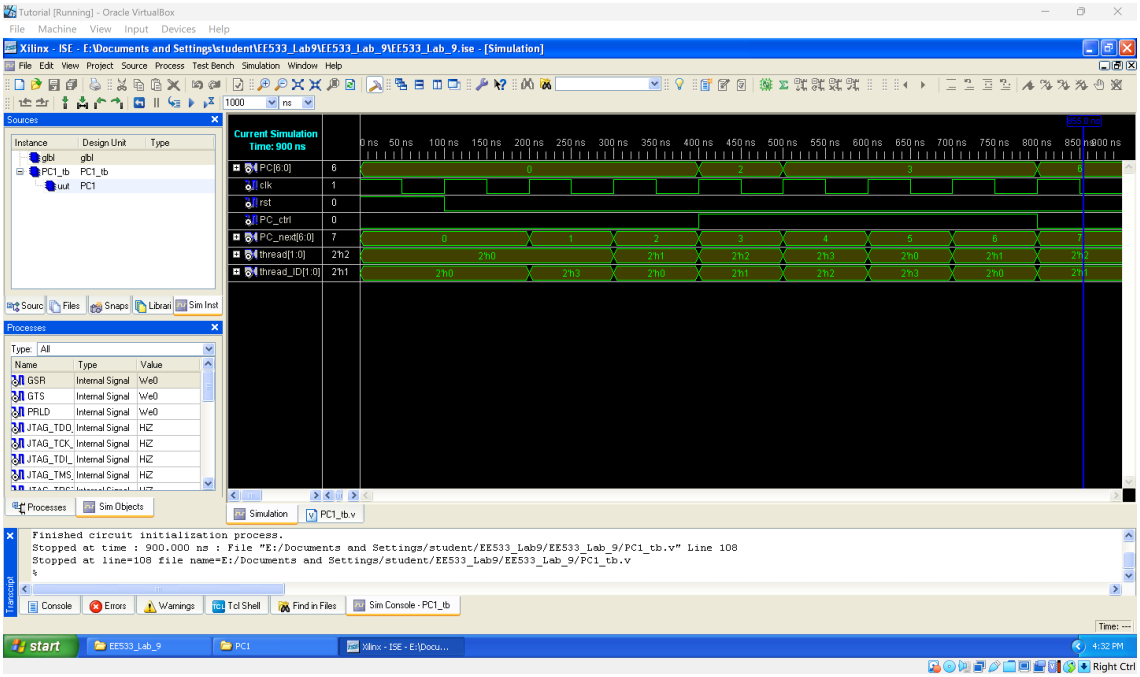
  - Testbench

```verilog
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////
///
// Company:
// Engineer:
//
// Create Date:   16:27:57 03/13/2025
```

```
// Design Name:   PC1
// Module Name:   E:/Documents and
Settings/student/EE533_Lab9/EE533_Lab_9/PC1_tb.v
// Project Name:  EE533_Lab_9
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: PC1
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
///

module PC1_tb;

    // Inputs
    reg clk;
    reg rst;
    reg PC_ctrl;
    reg [6:0] PC_next;
    reg [1:0] thread;
    reg [1:0] thread_ID;

    // Outputs
    wire [6:0] PC;

    // Instantiate the Unit Under Test (UUT)
    PC1 uut (
        .clk(clk),
        .rst(rst),
        .PC_ctrl(PC_ctrl),
        .PC_next(PC_next),
        .thread(thread),
        .thread_ID(thread_ID),
        .PC(PC)
    );

    always #50 clk = ~clk;

    initial begin
        // Initialize Inputs
        clk = 1;
        rst = 1;
        PC_ctrl = 0;
        PC_next = 0;
        thread = 0;
        thread_ID = 0;

        // Wait 100 ns for global reset to finish
        @(posedge clk);
```

```
        rst = 0;

        // Add stimulus here
        @(posedge clk);
        PC_ctrl = 0;
        PC_next = 7'd1;
        thread = 2'b00;
        thread_ID = 2'b11;

        @(posedge clk);
        PC_ctrl = 0;
        PC_next = 7'd2;
        thread = 2'b01;
        thread_ID = 2'b00;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd3;
        thread = 2'b10;
        thread_ID = 2'b01;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd4;
        thread = 2'b11;
        thread_ID = 2'b10;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd5;
        thread = 2'b00;
        thread_ID = 2'b11;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd6;
        thread = 2'b01;
        thread_ID = 2'b00;

        @(posedge clk);
        PC_ctrl = 0;
        PC_next = 7'd7;
        thread = 2'b10;
        thread_ID = 2'b01;

        @(posedge clk);
        $stop;

    end

endmodule
```

- Waveform



- PC2
  - Verilog

```
`timescale 1ns / 1ps

module PC2
(
    input clk,
    input rst,
    input PC_ctrl,
    input [6:0] PC_next,
    input [1:0] thread,
    input [1:0] thread_ID,

    output reg [6:0] PC
);

    always @(posedge clk) begin
        if (rst)
            PC <= 0;
        else if ((thread == 2'b10) || ((thread_ID == 2'b10) && PC_ctrl))
            PC <= PC_next;
    end

endmodule
```

  - Testbench

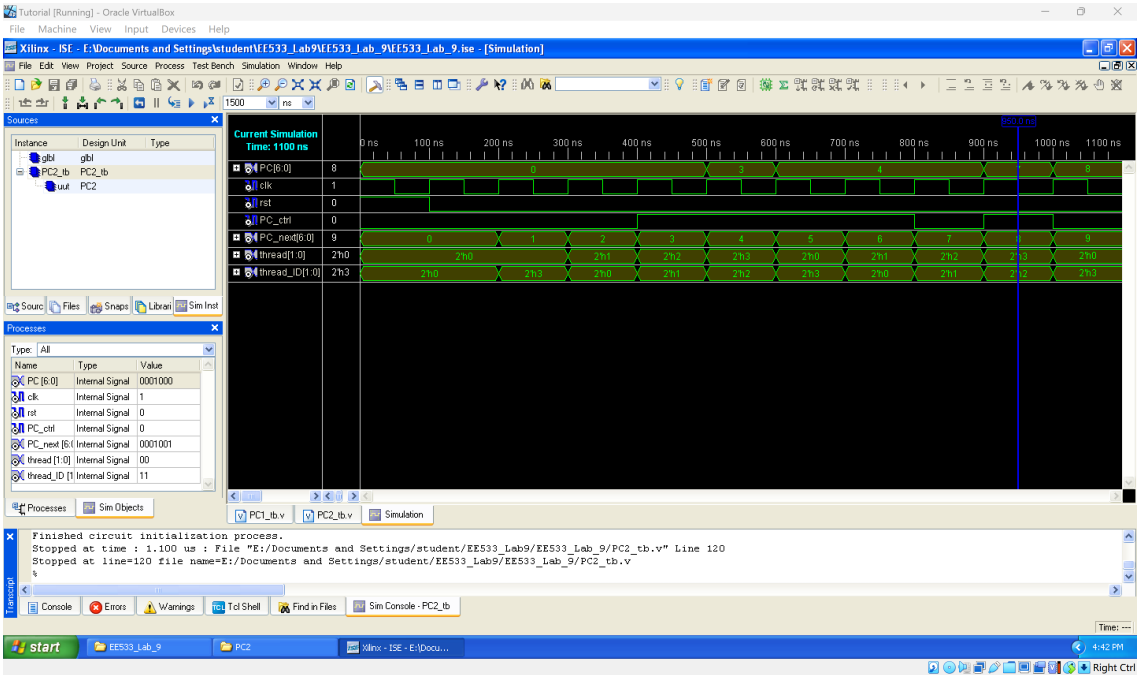
```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////
///
// Company:
// Engineer:
//
// Create Date:   16:28:12 03/13/2025
```

```
// Design Name:   PC2
// Module Name:   E:/Documents and
Settings/student/EE533_Lab9/EE533_Lab_9/PC2_tb.v
// Project Name:  EE533_Lab_9
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: PC2
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
///

module PC2_tb;

    // Inputs
    reg clk;
    reg rst;
    reg PC_ctrl;
    reg [6:0] PC_next;
    reg [1:0] thread;
    reg [1:0] thread_ID;

    // Outputs
    wire [6:0] PC;

    // Instantiate the Unit Under Test (UUT)
    PC2 uut (
        .clk(clk),
        .rst(rst),
        .PC_ctrl(PC_ctrl),
        .PC_next(PC_next),
        .thread(thread),
        .thread_ID(thread_ID),
        .PC(PC)
    );

    always #50 clk = ~clk;

    initial begin
        // Initialize Inputs
        clk = 1;
        rst = 1;
        PC_ctrl = 0;
        PC_next = 0;
        thread = 0;
        thread_ID = 0;

        // Wait 100 ns for global reset to finish
        @(posedge clk);
```

```
        rst = 0;

        // Add stimulus here
        @(posedge clk);
        PC_ctrl = 0;
        PC_next = 7'd1;
        thread = 2'b00;
        thread_ID = 2'b11;

        @(posedge clk);
        PC_ctrl = 0;
        PC_next = 7'd2;
        thread = 2'b01;
        thread_ID = 2'b00;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd3;
        thread = 2'b10;
        thread_ID = 2'b01;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd4;
        thread = 2'b11;
        thread_ID = 2'b10;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd5;
        thread = 2'b00;
        thread_ID = 2'b11;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd6;
        thread = 2'b01;
        thread_ID = 2'b00;

        @(posedge clk);
        PC_ctrl = 0;
        PC_next = 7'd7;
        thread = 2'b10;
        thread_ID = 2'b01;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd8;
        thread = 2'b11;
        thread_ID = 2'b10;

        @(posedge clk);
        PC_ctrl = 0;
        PC_next = 7'd9;
        thread = 2'b00;
        thread_ID = 2'b11;
```

```
        @(posedge clk);
        $stop;

    end

endmodule
```

- Waveform



- PC3
    - Verilog

```
`timescale 1ns / 1ps

module PC3
(
    input clk,
    input rst,
    input PC_ctrl,
    input [6:0] PC_next,
    input [1:0] thread,
    input [1:0] thread_ID,

    output reg [6:0] PC
);

    always @(posedge clk) begin
        if (rst)
            PC <= 0;
        else if ((thread == 2'b11) || ((thread_ID == 2'b11) && PC_ctrl))
            PC <= PC_next;
    end

endmodule
```

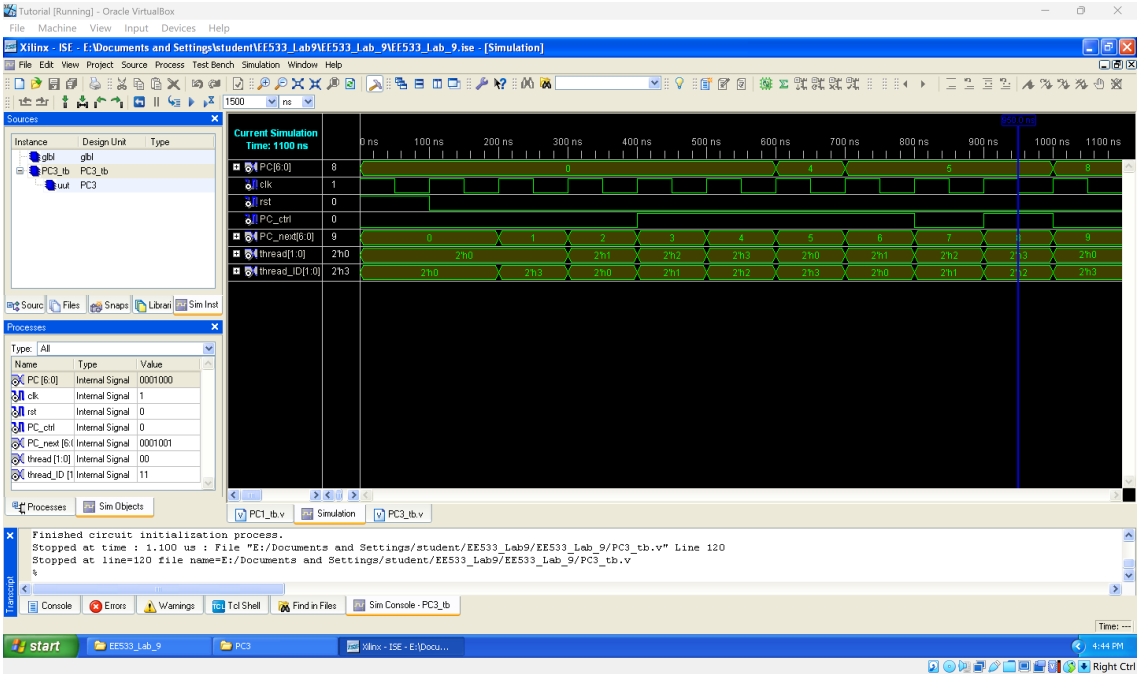
- Testbench

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
///
// Company:
// Engineer:
//
// Create Date:   16:28:23 03/13/2025
// Design Name:   PC3
// Module Name:   E:/Documents and
Settings/student/EE533_Lab9/EE533_Lab_9/PC3_tb.v
// Project Name:  EE533_Lab_9
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: PC3
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
///

module PC3_tb;

    // Inputs
    reg clk;
    reg rst;
    reg PC_ctrl;
    reg [6:0] PC_next;
    reg [1:0] thread;
    reg [1:0] thread_ID;

    // Outputs
    wire [6:0] PC;

    // Instantiate the Unit Under Test (UUT)
    PC3 uut (
        .clk(clk),
        .rst(rst),
        .PC_ctrl(PC_ctrl),
        .PC_next(PC_next),
        .thread(thread),
        .thread_ID(thread_ID),
        .PC(PC)
    );

    always #50 clk = ~clk;

    initial begin
        // Initialize Inputs
```

```
        clk = 1;
        rst = 1;
        PC_ctrl = 0;
        PC_next = 0;
        thread = 0;
        thread_ID = 0;

        // Wait 100 ns for global reset to finish
        @(posedge clk);
        rst = 0;

        // Add stimulus here
        @(posedge clk);
        PC_ctrl = 0;
        PC_next = 7'd1;
        thread = 2'b00;
        thread_ID = 2'b11;

        @(posedge clk);
        PC_ctrl = 0;
        PC_next = 7'd2;
        thread = 2'b01;
        thread_ID = 2'b00;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd3;
        thread = 2'b10;
        thread_ID = 2'b01;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd4;
        thread = 2'b11;
        thread_ID = 2'b10;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd5;
        thread = 2'b00;
        thread_ID = 2'b11;

        @(posedge clk);
        PC_ctrl = 1;
        PC_next = 7'd6;
        thread = 2'b01;
        thread_ID = 2'b00;

        @(posedge clk);
        PC_ctrl = 0;
        PC_next = 7'd7;
        thread = 2'b10;
        thread_ID = 2'b01;

        @(posedge clk);
        PC_ctrl = 1;
```

```
        PC_next = 7'd8;
        thread = 2'b11;
        thread_ID = 2'b10;

        @(posedge clk);
        PC_ctrl = 0;
        PC_next = 7'd9;
        thread = 2'b00;
        thread_ID = 2'b11;

        @(posedge clk);
        $stop;

    end

endmodule
```

- Waveform



1.1.2 PC_Thread_MUX

- Verilog

```
`timescale 1ns / 1ps

module PC_Thread_MUX
(
    input [1:0] thread,

    input [6:0] PC0,
    input [6:0] PC1,
    input [6:0] PC2,
    input [6:0] PC3,

    output [8:0] PC
);
```

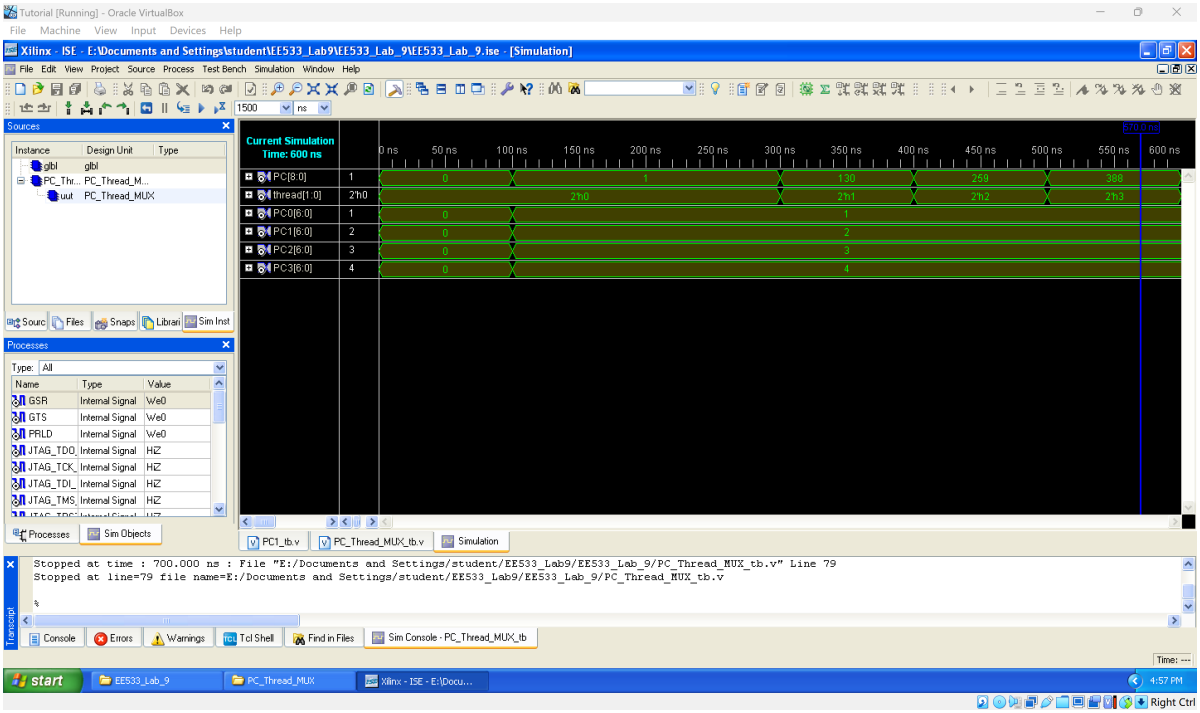
```
    assign PC = (thread == 2'b00) ? {thread, PC0} :
                (thread == 2'b01) ? {thread, PC1} :
                (thread == 2'b10) ? {thread, PC2} :
                {thread, PC3};

endmodule
```

- Testbench

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   11:24:50 03/13/2025
// Design Name:   PC_Thread_MUX
// Module Name:   E:/Documents and Settings/student/EE533_Lab9/PC_Thread_MUX_tb.v
// Project Name:  EE533_Lab9
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: PC_Thread_MUX
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module PC_Thread_MUX_tb;

    // Inputs
    reg [1:0] thread;
    reg [8:0] PC0;
    reg [8:0] PC1;
    reg [8:0] PC2;
    reg [8:0] PC3;

    // Outputs
    wire [8:0] PC;

    // Instantiate the Unit Under Test (UUT)
    PC_Thread_MUX uut (
        .thread(thread),
        .PC0(PC0),
        .PC1(PC1),
        .PC2(PC2),
        .PC3(PC3),
        .PC(PC)
    );
```

```
    initial begin
        // Initialize Inputs
        thread = 0;
        PC0 = 0;
        PC1 = 0;
        PC2 = 0;
        PC3 = 0;

        // Wait 100 ns for global reset to finish
        #100;
        PC0 = 7'd1;
        PC1 = 7'd2;
        PC2 = 7'd3;
        PC3 = 7'd4;

        // Add stimulus here
        #100;
        thread = 2'b00;

        #100;
        thread = 2'b01;

        #100;
        thread = 2'b10;

        #100;
        thread = 2'b11;

        #100;

        #100;
        $stop;

    end

endmodule
```

- Waveform



1.1.3 Thread_control_unit

- Verilog

```
`timescale 1ns / 1ps

module Thread_control_unit
(
    input [1:0] thread_ID,
    input PC_ctrl,

    output PC0_ctrl,
    output PC1_ctrl,
    output PC2_ctrl,
    output PC3_ctrl
);

    assign PC0_ctrl = ((thread_ID == 2'b00) && PC_ctrl) ? 1 : 0;
    assign PC1_ctrl = ((thread_ID == 2'b01) && PC_ctrl) ? 1 : 0;
    assign PC2_ctrl = ((thread_ID == 2'b10) && PC_ctrl) ? 1 : 0;
    assign PC3_ctrl = ((thread_ID == 2'b11) && PC_ctrl) ? 1 : 0;

endmodule
```

- Testbench

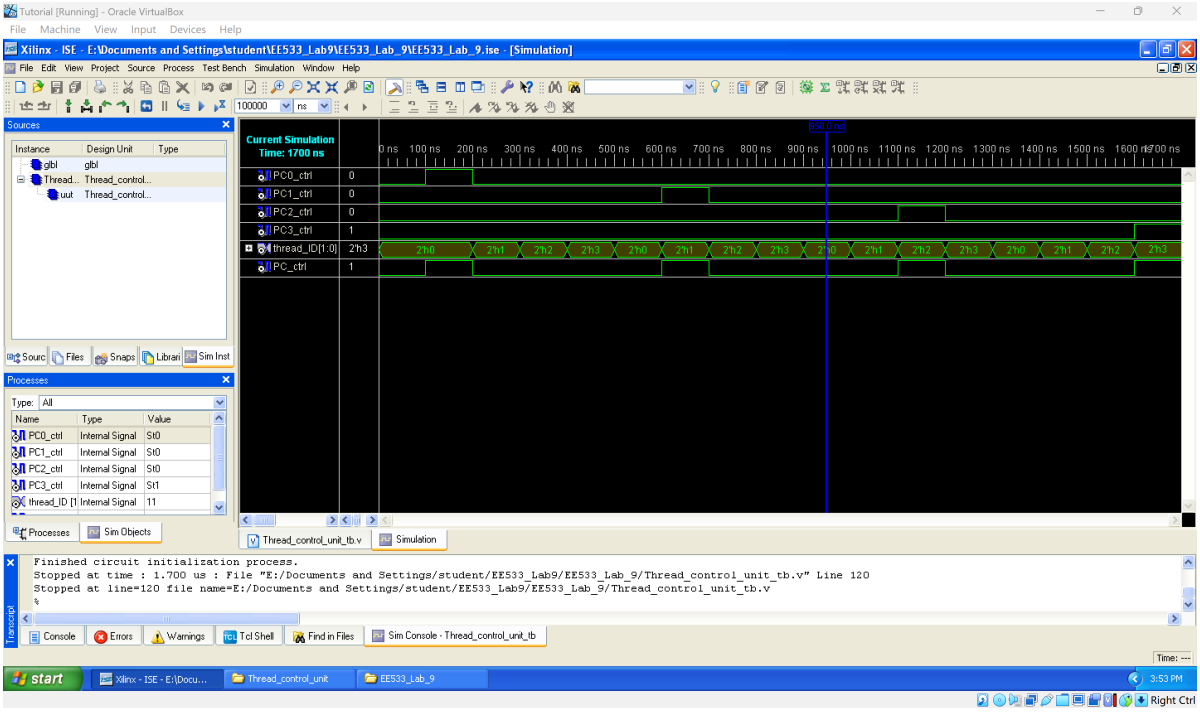
```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   15:51:08 03/14/2025
// Design Name:   Thread_control_unit
```

```
// Module Name:   E:/Documents and
Settings/student/EE533_Lab9/EE533_Lab_9/Thread_control_unit_tb.v
// Project Name:  EE533_Lab_9
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: Thread_control_unit
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module Thread_control_unit_tb;

    // Inputs
    reg [1:0] thread_ID;
    reg PC_ctrl;

    // Outputs
    wire PC0_ctrl;
    wire PC1_ctrl;
    wire PC2_ctrl;
    wire PC3_ctrl;

    // Instantiate the Unit Under Test (UUT)
    Thread_control_unit uut (
        .thread_ID(thread_ID),
        .PC_ctrl(PC_ctrl),
        .PC0_ctrl(PC0_ctrl),
        .PC1_ctrl(PC1_ctrl),
        .PC2_ctrl(PC2_ctrl),
        .PC3_ctrl(PC3_ctrl)
    );

    initial begin
        // Initialize Inputs
        thread_ID = 0;
        PC_ctrl = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        thread_ID = 2'b00;
        PC_ctrl = 1;
        #100;

        thread_ID = 2'b01;
        PC_ctrl = 0;
        #100;
```

```
        thread_ID = 2'b10;
        PC_ctrl = 0;
        #100;

        thread_ID = 2'b11;
        PC_ctrl = 0;
        #100;

        thread_ID = 2'b00;
        PC_ctrl = 0;
        #100;

        thread_ID = 2'b01;
        PC_ctrl = 1;
        #100;

        thread_ID = 2'b10;
        PC_ctrl = 0;
        #100;

        thread_ID = 2'b11;
        PC_ctrl = 0;
        #100;

        thread_ID = 2'b00;
        PC_ctrl = 0;
        #100;

        thread_ID = 2'b01;
        PC_ctrl = 0;
        #100;

        thread_ID = 2'b10;
        PC_ctrl = 1;
        #100;

        thread_ID = 2'b11;
        PC_ctrl = 0;
        #100;

        thread_ID = 2'b00;
        PC_ctrl = 0;
        #100;

        thread_ID = 2'b01;
        PC_ctrl = 0;
        #100;

        thread_ID = 2'b10;
        PC_ctrl = 0;
        #100;

        thread_ID = 2'b11;
        PC_ctrl = 1;
        #100;
```

```
        $stop;

    end

endmodule
```

- Waveform



1.1.4 PC_part

- Verilog

```
//////////////////////////////////////////////////////////////////////////////////
// Copyright (c) 1995-2008 Xilinx, Inc.  All rights reserved.
//////////////////////////////////////////////////////////////////////////////////
//   ____  ____
//  /   /\/   /
// /___/  \  /    Vendor: Xilinx
// \   \   \/     Version : 10.1
//  \   \         Application : sch2verilog
//  /   /         Filename : PC_part_debug.vf
// /___/   /\     Timestamp : 03/14/2025 12:52:50
// \   \  /  \
//  \___\/\___\
//
//Command: C:\Xilinx\10.1\ISE\bin\nt\unwrapped\sch2verilog.exe -intstyle ise -
family virtex2p -w "E:/Documents and
Settings/student/EE533_Lab9/PC_part_debug.sch" PC_part_debug.vf
//Design Name: PC_part_debug
//Device: virtex2p
//Purpose:
//    This verilog netlist is translated from an ECS schematic.It can be
//    synthesized and simulated, but it should not be modified.
//
`timescale 1ns / 1ps
```

```
module PC_part_debug(BTA,
                     clk,
                     ONE,
                     PC_ctrl,
                     rst,
                     thread_ID,
                     thread_IF,
                     PC,
                     PC0,
                     PC1,
                     PC2,
                     PC3);

    input [6:0] BTA;
    input clk;
    input [6:0] ONE;
    input PC_ctrl;
    input rst;
    input [1:0] thread_ID;
    input [1:0] thread_IF;
   output [8:0] PC;
   output [6:0] PC0;
   output [6:0] PC1;
   output [6:0] PC2;
   output [6:0] PC3;

   wire PC0_ctrl;
   wire PC1_ctrl;
   wire PC2_ctrl;
   wire PC3_ctrl;
   wire [6:0] XLXN_5;
   wire [6:0] XLXN_6;
   wire [6:0] XLXN_7;
   wire [6:0] XLXN_8;
   wire [6:0] XLXN_17;
   wire [6:0] XLXN_18;
   wire [6:0] XLXN_19;
   wire [6:0] XLXN_20;
   wire [6:0] PC0_DUMMY;
   wire [6:0] PC1_DUMMY;
   wire [6:0] PC2_DUMMY;
   wire [6:0] PC3_DUMMY;

   assign PC0[6:0] = PC0_DUMMY[6:0];
   assign PC1[6:0] = PC1_DUMMY[6:0];
   assign PC2[6:0] = PC2_DUMMY[6:0];
   assign PC3[6:0] = PC3_DUMMY[6:0];
   PC0 XLXI_1 (.clk(clk),
               .PC_ctrl(PC0_ctrl),
               .PC_next(XLXN_17[6:0]),
               .rst(rst),
               .thread(thread_IF[1:0]),
               .thread_ID(thread_ID[1:0]),
               .PC(PC0_DUMMY[6:0]));
   PC1 XLXI_2 (.clk(clk),
               .PC_ctrl(PC1_ctrl),
```

```
            .PC_next(XLXN_18[6:0]),
            .rst(rst),
            .thread(thread_IF[1:0]),
            .thread_ID(thread_ID[1:0]),
            .PC(PC1_DUMMY[6:0]));
PC2 XLXI_3 (.clk(clk),
            .PC_ctrl(PC2_ctrl),
            .PC_next(XLXN_19[6:0]),
            .rst(rst),
            .thread(thread_IF[1:0]),
            .thread_ID(thread_ID[1:0]),
            .PC(PC2_DUMMY[6:0]));
PC3 XLXI_4 (.clk(clk),
            .PC_ctrl(PC3_ctrl),
            .PC_next(XLXN_20[6:0]),
            .rst(rst),
            .thread(thread_IF[1:0]),
            .thread_ID(thread_ID[1:0]),
            .PC(PC3_DUMMY[6:0]));
PC_plus_1 XLXI_5 (.ONE(ONE[6:0]),
                  .PC(PC0_DUMMY[6:0]),
                  .PC_next(XLXN_8[6:0]));
PC_plus_1 XLXI_6 (.ONE(ONE[6:0]),
                  .PC(PC1_DUMMY[6:0]),
                  .PC_next(XLXN_7[6:0]));
PC_plus_1 XLXI_7 (.ONE(ONE[6:0]),
                  .PC(PC2_DUMMY[6:0]),
                  .PC_next(XLXN_6[6:0]));
PC_plus_1 XLXI_8 (.ONE(ONE[6:0]),
                  .PC(PC3_DUMMY[6:0]),
                  .PC_next(XLXN_5[6:0]));
PC_MUX XLXI_9 (.BTA(BTA[6:0]),
               .PC_ctrl(PC0_ctrl),
               .PC_next_in(XLXN_8[6:0]),
               .PC_next_out(XLXN_17[6:0]));
PC_MUX XLXI_10 (.BTA(BTA[6:0]),
                .PC_ctrl(PC1_ctrl),
                .PC_next_in(XLXN_7[6:0]),
                .PC_next_out(XLXN_18[6:0]));
PC_MUX XLXI_11 (.BTA(BTA[6:0]),
                .PC_ctrl(PC2_ctrl),
                .PC_next_in(XLXN_6[6:0]),
                .PC_next_out(XLXN_19[6:0]));
PC_MUX XLXI_12 (.BTA(BTA[6:0]),
                .PC_ctrl(PC3_ctrl),
                .PC_next_in(XLXN_5[6:0]),
                .PC_next_out(XLXN_20[6:0]));
PC_Thread_MUX XLXI_13 (.PC0(PC0_DUMMY[6:0]),
                       .PC1(PC1_DUMMY[6:0]),
                       .PC2(PC2_DUMMY[6:0]),
                       .PC3(PC3_DUMMY[6:0]),
                       .thread(thread_IF[1:0]),
                       .PC(PC[8:0]));
Thread_control_unit XLXI_14 (.PC_ctrl(PC_ctrl),
                             .thread_ID(thread_ID[1:0]),
                             .PC0_ctrl(PC0_ctrl),
```

```
                                .PC1_ctrl(PC1_ctrl),
                                .PC2_ctrl(PC2_ctrl),
                                .PC3_ctrl(PC3_ctrl));
endmodule
```

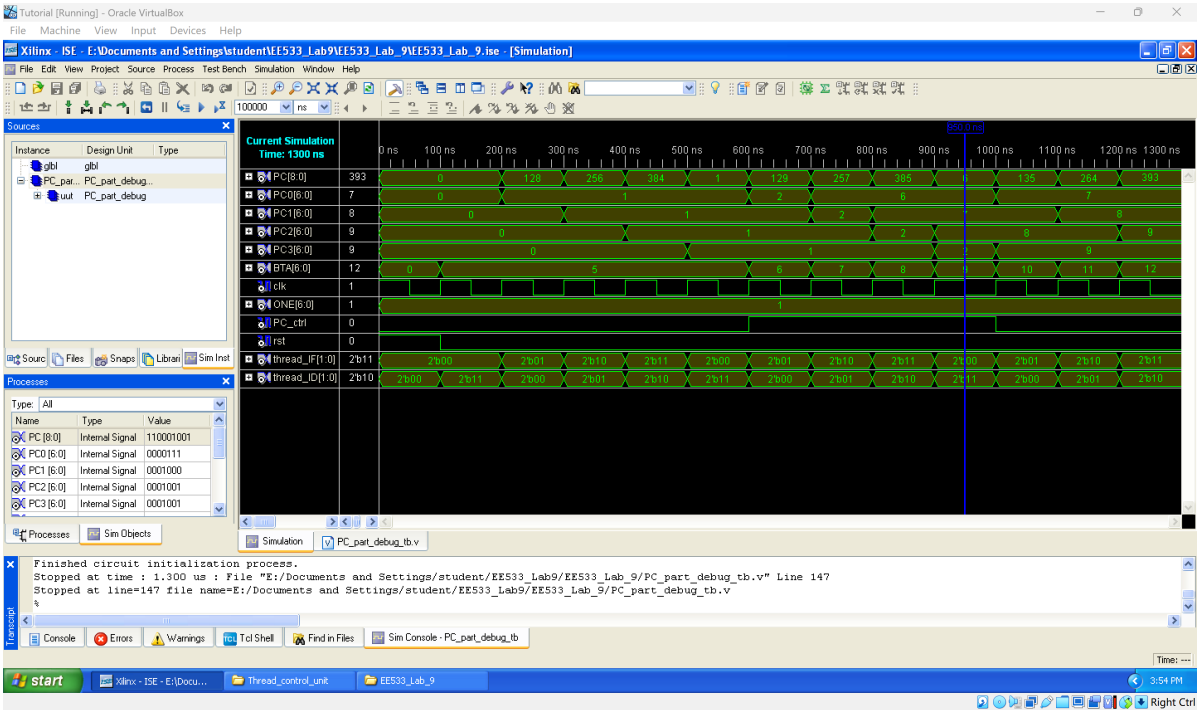
- Testbench

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   12:54:25 03/14/2025
// Design Name:   PC_part_debug
// Module Name:   E:/Documents and
Settings/student/EE533_Lab9/EE533_Lab_9/PC_part_debug_tb.v
// Project Name:  EE533_Lab_9
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: PC_part_debug
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module PC_part_debug_tb;

    // Inputs
    reg [6:0] BTA;
    reg clk;
    reg [6:0] ONE;
    reg PC_ctrl;
    reg rst;
    reg [1:0] thread_ID;
    reg [1:0] thread_IF;

    // Outputs
    wire [8:0] PC;
    wire [6:0] PC0;
    wire [6:0] PC1;
    wire [6:0] PC2;
    wire [6:0] PC3;

    // Instantiate the Unit Under Test (UUT)
    PC_part_debug uut (
        .BTA(BTA),
        .clk(clk),
        .ONE(ONE),
```

```
        .PC_ctrl(PC_ctrl),
        .rst(rst),
        .thread_ID(thread_ID),
        .thread_IF(thread_IF),
        .PC(PC),
        .PC0(PC0),
        .PC1(PC1),
        .PC2(PC2),
        .PC3(PC3)
    );

    always #50 clk = ~clk;

    initial begin
        // Initialize Inputs
        BTA = 0;
        clk = 1;
        ONE = 7'd1;
        PC_ctrl = 0;
        rst = 1;
        thread_ID = 0;
        thread_IF = 0;

        // Wait 100 ns for global reset to finish
        @(posedge clk);

        // Add stimulus here
        rst = 0;
        BTA = 7'd5;
        PC_ctrl = 0;
        thread_IF = 2'b00;
        thread_ID = 2'b11;
        @(posedge clk);

        BTA = 7'd5;
        PC_ctrl = 0;
        thread_IF = 2'b01;
        thread_ID = 2'b00;
        @(posedge clk);

        BTA = 7'd5;
        PC_ctrl = 0;
        thread_IF = 2'b10;
        thread_ID = 2'b01;
        @(posedge clk);

        BTA = 7'd5;
        PC_ctrl = 0;
        thread_IF = 2'b11;
        thread_ID = 2'b10;
        @(posedge clk);

        BTA = 7'd5;
        PC_ctrl = 0;
        thread_IF = 2'b00;
        thread_ID = 2'b11;
```

```
        @(posedge clk);

        BTA = 7'd6;
        PC_ctrl = 1;
        thread_IF = 2'b01;
        thread_ID = 2'b00;
        @(posedge clk);

        BTA = 7'd7;
        PC_ctrl = 1;
        thread_IF = 2'b10;
        thread_ID = 2'b01;
        @(posedge clk);

        BTA = 7'd8;
        PC_ctrl = 1;
        thread_IF = 2'b11;
        thread_ID = 2'b10;
        @(posedge clk);

        BTA = 7'd9;
        PC_ctrl = 1;
        thread_IF = 2'b00;
        thread_ID = 2'b11;
        @(posedge clk);

        BTA = 7'd10;
        PC_ctrl = 0;
        thread_IF = 2'b01;
        thread_ID = 2'b00;
        @(posedge clk);

        BTA = 7'd11;
        PC_ctrl = 0;
        thread_IF = 2'b10;
        thread_ID = 2'b01;
        @(posedge clk);

        BTA = 7'd12;
        PC_ctrl = 0;
        thread_IF = 2'b11;
        thread_ID = 2'b10;
        @(posedge clk);
        $stop;

    end

endmodule
```

- Waveform



1.1.5 RF_DEMUX

- Verilog

```
`timescale 1ns / 1ps

module RF_DEMUX
(
    input [1:0] thread_WB,

    input [2:0] waddr,
    input [63:0] wdata,
    input WRE_WB,

    output [2:0] waddr_DEMUX_out,
    output [63:0] wdata_DEMUX_out,
    output RF0_WRE_WB,
    output RF1_WRE_WB,
    output RF2_WRE_WB,
    output RF3_WRE_WB
);

    assign waddr_DEMUX_out = waddr;
    assign wdata_DEMUX_out = wdata;
    assign RF0_WRE_WB = ((thread_WB == 2'b00) && WRE_WB) ? 1 : 0;
    assign RF1_WRE_WB = ((thread_WB == 2'b01) && WRE_WB) ? 1 : 0;
    assign RF2_WRE_WB = ((thread_WB == 2'b10) && WRE_WB) ? 1 : 0;
    assign RF3_WRE_WB = ((thread_WB == 2'b11) && WRE_WB) ? 1 : 0;

endmodule
```

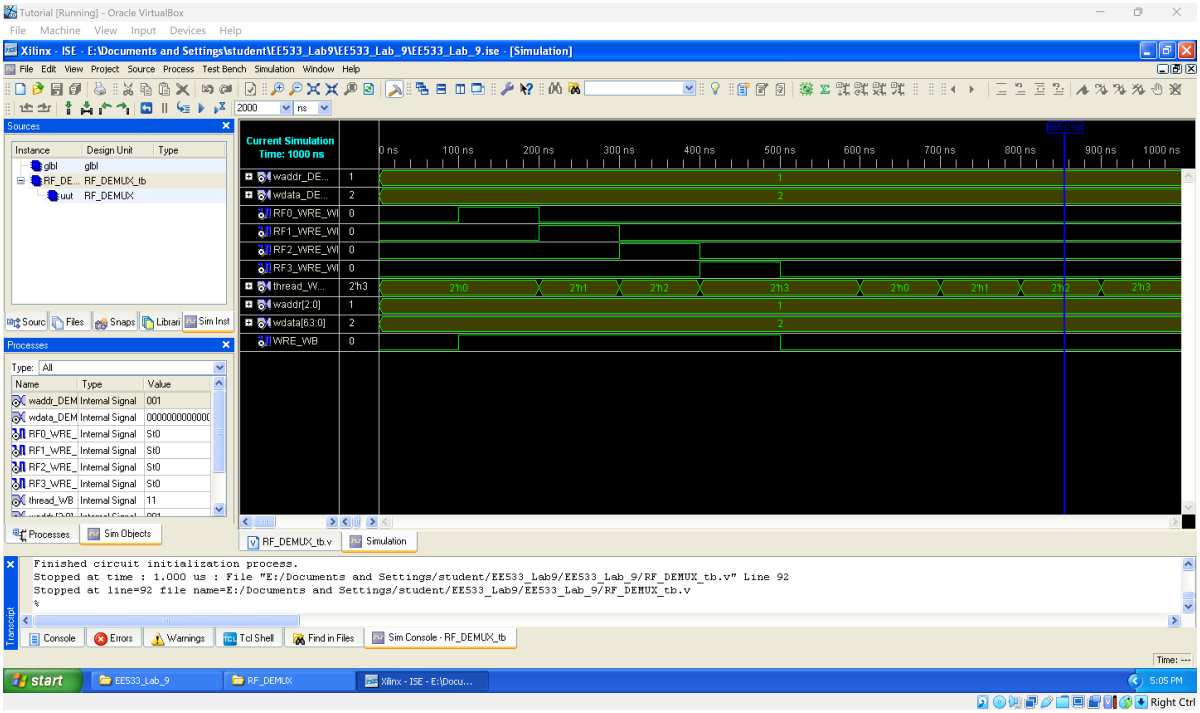
- Testbench

```
`timescale 1ns / 1ps
```

```
////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   12:15:14 03/13/2025
// Design Name:   RF_DEMUX
// Module Name:   E:/Documents and Settings/student/EE533_Lab9/RF_DEMUX_tb.v
// Project Name:  EE533_Lab9
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: RF_DEMUX
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module RF_DEMUX_tb;

    // Inputs
    reg [1:0] thread_WB;
    reg [2:0] waddr;
    reg [63:0] wdata;
    reg WRE_WB;

    // Outputs
    wire [2:0] waddr_DEMUX_out;
    wire [63:0] wdata_DEMUX_out;
    wire RF0_WRE_WB;
    wire RF1_WRE_WB;
    wire RF2_WRE_WB;
    wire RF3_WRE_WB;

    // Instantiate the Unit Under Test (UUT)
    RF_DEMUX uut (
        .thread_WB(thread_WB),
        .waddr(waddr),
        .wdata(wdata),
        .WRE_WB(WRE_WB),
        .waddr_DEMUX_out(waddr_DEMUX_out),
        .wdata_DEMUX_out(wdata_DEMUX_out),
        .RF0_WRE_WB(RF0_WRE_WB),
        .RF1_WRE_WB(RF1_WRE_WB),
        .RF2_WRE_WB(RF2_WRE_WB),
        .RF3_WRE_WB(RF3_WRE_WB)
    );

    initial begin
        // Initialize Inputs
        thread_WB = 2'b00;
        waddr = 3'd1;
```

```
        wdata = 3'd2;
        WRE_WB = 0;

        // Wait 100 ns for global reset to finish
        #100;
        WRE_WB = 1;

        // Add stimulus here
        #100;
        thread_WB = 2'b01;

        #100;
        thread_WB = 2'b10;

        #100;
        thread_WB = 2'b11;

        #100;
        WRE_WB = 0;

        #100;
        thread_WB = 2'b00;

        #100;
        thread_WB = 2'b01;

        #100;
        thread_WB = 2'b10;

        #100;
        thread_WB = 2'b11;

        #100;
        $stop;

    end

endmodule
```

- Waveform



1.1.6 RF_MUX

- Verilog

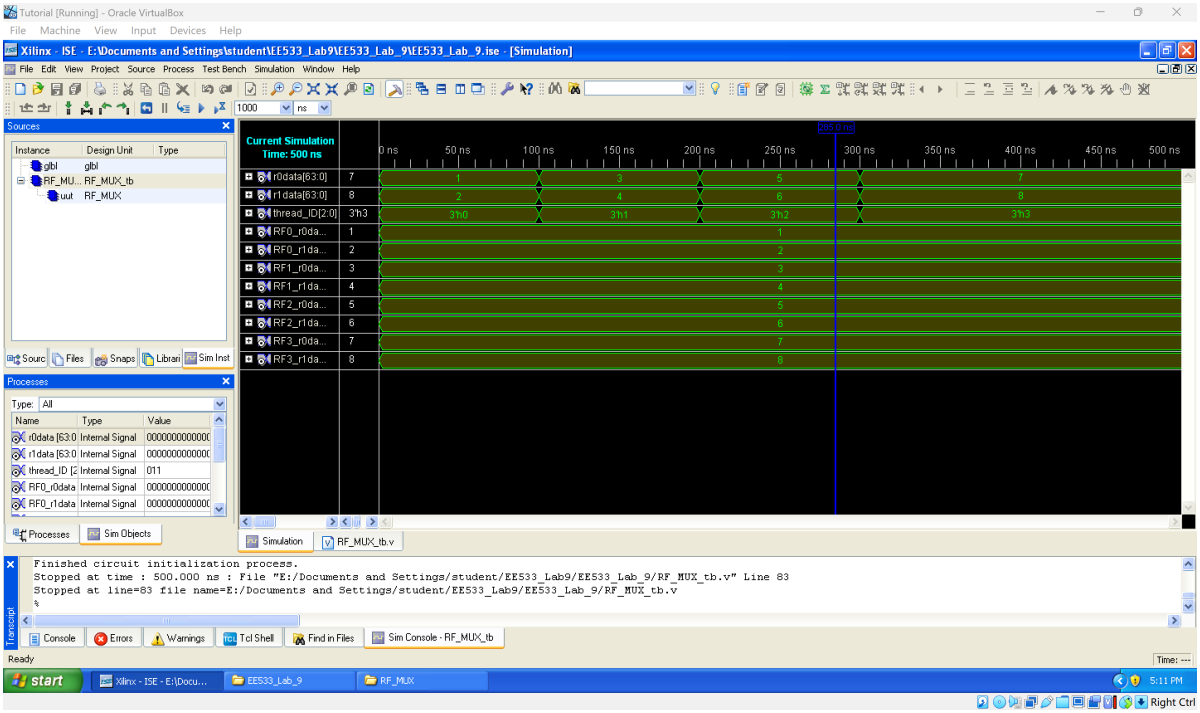
```
`timescale 1ns / 1ps

module RF_MUX
(
    input [1:0] thread_ID,

    input [63:0] RF0_r0data,
    input [63:0] RF0_r1data,
    input [63:0] RF1_r0data,
    input [63:0] RF1_r1data,
    input [63:0] RF2_r0data,
    input [63:0] RF2_r1data,
    input [63:0] RF3_r0data,
    input [63:0] RF3_r1data,

    output [63:0] r0data,
    output [63:0] r1data
);

    assign r0data = (thread_ID == 2'b00) ? RF0_r0data :
                    (thread_ID == 2'b01) ? RF1_r0data :
                    (thread_ID == 2'b10) ? RF2_r0data :
                    RF3_r0data;

    assign r1data = (thread_ID == 2'b00) ? RF0_r1data :
                    (thread_ID == 2'b01) ? RF1_r1data :
                    (thread_ID == 2'b10) ? RF2_r1data :
                    RF3_r1data;

endmodule
```

- Testbench

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   12:03:34 03/13/2025
// Design Name:   RF_MUX
// Module Name:   E:/Documents and Settings/student/EE533_Lab9/RF_MUX_tb.v
// Project Name:  EE533_Lab9
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: RF_MUX
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module RF_MUX_tb;

    // Inputs
    reg [2:0] thread_ID;
    reg [63:0] RF0_rOdata;
    reg [63:0] RF0_r1data;
    reg [63:0] RF1_rOdata;
    reg [63:0] RF1_r1data;
    reg [63:0] RF2_rOdata;
    reg [63:0] RF2_r1data;
    reg [63:0] RF3_rOdata;
    reg [63:0] RF3_r1data;

    // Outputs
    wire [63:0] rOdata;
    wire [63:0] r1data;

    // Instantiate the Unit Under Test (UUT)
    RF_MUX uut (
        .thread_ID(thread_ID),
        .RF0_rOdata(RF0_rOdata),
        .RF0_r1data(RF0_r1data),
        .RF1_rOdata(RF1_rOdata),
        .RF1_r1data(RF1_r1data),
        .RF2_rOdata(RF2_rOdata),
        .RF2_r1data(RF2_r1data),
        .RF3_rOdata(RF3_rOdata),
        .RF3_r1data(RF3_r1data),
        .rOdata(rOdata),
```

```
        .r1data(r1data)
    );

    initial begin
        // Initialize Inputs
        thread_ID = 2'b00;
        RF0_r0data = 64'd1;
        RF0_r1data = 64'd2;
        RF1_r0data = 64'd3;
        RF1_r1data = 64'd4;
        RF2_r0data = 64'd5;
        RF2_r1data = 64'd6;
        RF3_r0data = 64'd7;
        RF3_r1data = 64'd8;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        thread_ID = 2'b01;
        #100;

        thread_ID = 2'b10;
        #100;

        thread_ID = 2'b11;
        #100;

        #100;
        $stop;


    end

endmodule
```

- Waveform



1.2 Other changes Summary

1.2.1 StageReg

- Added 2 bits for every stage register with:
  - thread_STAGE[1:0]

1.2.2 D_MEM

- Changed the D_MEM to Read Before Write.
  - Since D_MEM is a sync component for both read port and write port, we want to make the read port acting as an async port, so we changed the D_MEM into Read Before Write type.
  - Otherwise it would read some random values and write into out Register File, the pkt_out would not be what we expected.

2. Instruction Part

2.1 Instruction Format Definition

- OP_CODE Lookup table

Instr	OP Code [31:26]
NOOP	000000
ADDI	000001
MOVI	000010
LW	000011
SW	000100
BEQ	000101

Instr	OP Code [31:26]
BGT	000110
BLT	000111
J	001000
SUBI	001001

2.2 Thread 0

- Purpose
  - First element in payload + 1
- Instruction Table

Addr	Label	Instr	OP Code [31:26]	Rs [25:21]	Rt [20:16]	Offset [15:0]
0		LW R1, #0(R0)	000011	5'd0	5'd1	16'd0
1		ADDI R1, R1, #1	000001	5'd1	5'd1	16'd1
2		SW R1, #0(R0)	000100	5'd0	5'd1	16'd0

- Instruction Initilaization File / Vector File

```
000011 00000 00001 0000000000000000 = 0000 1100 0000 0001 0000 0000 0000 0000 = 0C010000
000001 00001 00001 0000000000000001 = 0000 0100 0010 0001 0000 0000 0000 0001 = 04210001
000100 00000 00001 0000000000000000 = 0001 0000 0000 0001 0000 0000 0000 0000 = 10010000
```

2.3 Thread 1

- Purpose
  - Second element in payload - 1
- Instruction Table

Addr	Label	Instr	OP Code [31:26]	Rs [25:21]	Rt [20:16]	Offset [15:0]
0		LW R1, #1(R0)	000011	5'd0	5'd1	16'd1
1		SUBI R1, R1, #1	001001	5'd1	5'd1	16'd1

Addr	Label	Instr	OP Code [31:26]	Rs [25:21]	Rt [20:16]	Offset [15:0]
2		SW R1, #1(R0)	000100	5'd0	5'd1	16'd1

- Instruction Initilaization File / Vector File

```
000011 00000 00001 0000000000000001 = 0000 1100 0000 0001 0000 0000 0000 0001 =
0C010001
001001 00001 00001 0000000000000001 = 0010 0100 0010 0001 0000 0000 0000 0001 =
24210001
000100 00000 00001 0000000000000001 = 0001 0000 0000 0001 0000 0000 0000 0001 =
10010001
```

2.4 Thread 2

- Purpose
  - Load word third element and forth element, if third element >= forth element, then swap them; otherwise don't do anything.
- Instruction Table

Addr	Label	Instr	OP Code [31:26]	Rs [25:21]	Rt [20:16]	Offset [15:0]
0		LW R1, #2(R0)	000011	5'd0	5'd1	16'd2
1		LW R2, #3(R0)	000011	5'd0	5'd2	16'd3
2		BGT R1, R2, no_swap	000110	5'd1	5'd2	16'd5
3		SW R1, #3(R0)	000100	5'd0	5'd1	16'd3
4		SW R2, #2(R0)	000100	5'd0	5'd2	16'd2
5	no_swap	NOOP	000000	5'd0	5'd0	16'd0

- Instruction Initilaization File / Vector File

```
000011 00000 00001 0000000000000010 = 0000 1100 0000 0001 0000 0000 0000 0010 =
0C010002
000011 00000 00010 0000000000000011 = 0000 1100 0000 0010 0000 0000 0000 0011 =
0C020003
000110 00001 00010 0000000000000101 = 0001 1000 0010 0010 0000 0000 0000 0101 =
18220005
000100 00000 00001 0000000000000011 = 0001 0000 0000 0001 0000 0000 0000 0011 =
10010003
000100 00000 00010 0000000000000010 = 0001 0000 0000 0010 0000 0000 0000 0010 =
10020002
000000 00000 00000 0000000000000000 = 0000 0000 0000 0000 0000 0000 0000 0000 =
00000000
```

2.5 Thread 3

- Purpose
  - Load word fifth element and sixth element, if fifth element >= sixth element, then swap them; otherwise don't do anything.
- Instruction Table

Addr	Label	Instr	OP Code [31:26]	Rs [25:21]	Rt [20:16]	Offset [15:0]
0		LW R1, #4(R0)	000011	5'd0	5'd1	16'd4
1		LW R2, #5(R0)	000011	5'd0	5'd2	16'd5
2		BGT R1, R2, no_swap	000110	5'd1	5'd2	16'd5
3		SW R1, #5(R0)	000100	5'd0	5'd1	16'd5
4		SW R2, #4(R0)	000100	5'd0	5'd2	16'd4
5	no_swap	NOOP	000000	5'd0	5'd0	16'd0

- Instruction Initialization File / Vector File

```
000011 00000 00001 0000000000000100 = 0000 1100 0000 0001 0000 0000 0000 0100 =
0C010004
000011 00000 00010 0000000000000101 = 0000 1100 0000 0010 0000 0000 0000 0101 =
0C020005
000110 00001 00010 0000000000000101 = 0001 1000 0010 0010 0000 0000 0000 0101 =
18220005
000100 00000 00001 0000000000000101 = 0001 0000 0000 0001 0000 0000 0000 0101 =
10010005
000100 00000 00010 0000000000000100 = 0001 0000 0000 0010 0000 0000 0000 0100 =
10020004
000000 00000 00000 0000000000000000 = 0000 0000 0000 0000 0000 0000 0000 0000 =
00000000
```

2.6 Total Instruction

Thread	PCx	Label	Instr	OP Code [31:26]	Rs [25:21]	Rt [20:16]	Offset [15:0]
00	0000000		LW R1, #0(R0)	000011	5'd0	5'd1	16'd0
00	0000001		ADDI R1, R1, #1	000001	5'd1	5'd1	16'd1
00	0000010		SW R1, #0(R0)	000100	5'd0	5'd1	16'd0

Thread	PCx	Label	Instr	OP Code [31:26]	Rs [25:21]	Rt [20:16]	Offset [15:0]
01	0000000		LW R1, #1(R0)	000011	5'd0	5'd1	16'd1
01	0000001		SUBI R1, R1, #1	001001	5'd1	5'd1	16'd1
01	0000010		SW R1, #1(R0)	000100	5'd0	5'd1	16'd1
10	0000000		LW R1, #2(R0)	000011	5'd0	5'd1	16'd2
10	0000001		LW R2, #3(R0)	000011	5'd0	5'd2	16'd3
10	0000010		BLT R1, R2, no_swap	000111	5'd1	5'd2	16'd5
10	0000011		SW R1, #3(R0)	000100	5'd0	5'd1	16'd3
10	0000100		SW R2, #2(R0)	000100	5'd0	5'd2	16'd2
10	0000101	no_swap	NOOP	000000	5'd0	5'd0	16'd0
11	0000000		LW R1, #4(R0)	000011	5'd0	5'd1	16'd4
11	0000001		LW R2, #5(R0)	000011	5'd0	5'd2	16'd5
11	0000010		BLT R1, R2, no_swap	000111	5'd1	5'd2	16'd5
11	0000011		SW R1, #5(R0)	000100	5'd0	5'd1	16'd5
11	0000100		SW R2, #4(R0)	000100	5'd0	5'd2	16'd4
11	0000101	no_swap	NOOP	000000	5'd0	5'd0	16'd0

- Instruction Write in process in testbench should be:

```
// Thread 0 Instruction Input
Instr_IN = 32'h0C010000;
Instr_IN_addr = 9'b0000000;
Instr_IN_en = 1;

Instr_IN = 32'h04210001;
```

```
Instr_IN_addr = 9'b0000001;
Instr_IN_en = 1;

Instr_IN = 32'h10010000;
Instr_IN_addr = 9'b0000010;
Instr_IN_en = 1;

// Thread 1 Instruction Input
Instr_IN = 32'h0C010001;
Instr_IN_addr = 9'b0100000;
Instr_IN_en = 1;

Instr_IN = 32'h24210001;
Instr_IN_addr = 9'b0100001;
Instr_IN_en = 1;

Instr_IN = 32'h10010001;
Instr_IN_addr = 9'b0100010;
Instr_IN_en = 1;

// Thread 2 Instruction Input
Instr_IN = 32'h0C010002;
Instr_IN_addr = 9'b1000000;
Instr_IN_en = 1;

Instr_IN = 32'h0C020003;
Instr_IN_addr = 9'b1000001;
Instr_IN_en = 1;

Instr_IN = 32'h18220005;
Instr_IN_addr = 9'b1000010;
Instr_IN_en = 1;

Instr_IN = 32'h10010003;
Instr_IN_addr = 9'b1000011;
Instr_IN_en = 1;

Instr_IN = 32'h10020002;
Instr_IN_addr = 9'b1000100;
Instr_IN_en = 1;

Instr_IN = 32'h00000000;
Instr_IN_addr = 9'b1000101;
Instr_IN_en = 1;

// Thread 3 Instruction Input
Instr_IN = 32'h0C010004;
Instr_IN_addr = 9'b1100000;
Instr_IN_en = 1;

Instr_IN = 32'h0C020005;
Instr_IN_addr = 9'b1100001;
Instr_IN_en = 1;

Instr_IN = 32'h18220005;
Instr_IN_addr = 9'b1100010;
```

```
Instr_IN_en = 1;

Instr_IN = 32'h10010005;
Instr_IN_addr = 9'b1100011;
Instr_IN_en = 1;

Instr_IN = 32'h10020004;
Instr_IN_addr = 9'b1100100;
Instr_IN_en = 1;

Instr_IN = 32'h00000000;
Instr_IN_addr = 9'b1100101;
Instr_IN_en = 1;
```

- Total Instruction Fetch Process through different thread should be expected as following:

Thread	PCx	Label	Instr	OP Code [31:26]	Rs [25:21]	Rt [20:16]	Offset [15:0]
00	0000000		LW R1, #0(R0)	000011	5'd0	5'd1	16'd0
01	0000000		LW R1, #1(R0)	000011	5'd0	5'd1	16'd1
10	0000000		LW R1, #2(R0)	000011	5'd0	5'd1	16'd2
11	0000000		LW R1, #4(R0)	000011	5'd0	5'd1	16'd4
00	0000001		ADDI R1, R1, #1	000001	5'd1	5'd1	16'd1
01	0000001		SUBI R1, R1, #1	001001	5'd1	5'd1	16'd1
10	0000001		LW R2, #3(R0)	000011	5'd0	5'd2	16'd3
11	0000001		LW R2, #5(R0)	000011	5'd0	5'd2	16'd5
00	0000010		SW R1, #0(R0)	000100	5'd0	5'd1	16'd0
01	0000010		SW R1, #1(R0)	000100	5'd0	5'd1	16'd1
10	0000010		BLT R1, R2, no_swap	000111	5'd1	5'd2	16'd5

Thread	PCx	Label	Instr	OP Code [31:26]	Rs [25:21]	Rt [20:16]	Offset [15:0]
11	0000010		BLT R1, R2, no_swap	000111	5'd1	5'd2	16'd5
00	0000011		NOOP	000000	5'd0	5'd0	16'd0
01	0000011		NOOP	000000	5'd0	5'd0	16'd0
10	0000011		SW R1, #3(R0)	000100	5'd0	5'd1	16'd3
11	0000101	no_swap	NOOP	000000	5'd0	5'd0	16'd0
00	0000100		NOOP	000000	5'd0	5'd0	16'd0
01	0000100		NOOP	000000	5'd0	5'd0	16'd0
10	0000100		SW R2, #2(R0)	000100	5'd0	5'd2	16'd2
11	0000110		NOOP	000000	5'd0	5'd0	16'd0
00	0000101		NOOP	000000	5'd0	5'd0	16'd0
01	0000101		NOOP	000000	5'd0	5'd0	16'd0
10	0000101	no_swap	NOOP	000000	5'd0	5'd0	16'd0
11	0000111		NOOP	000000	5'd0	5'd0	16'd0

3. Packet Part

3.1 Purpose

- Use four independent threads to change the packet payload field.

3.2 Packet Header Definition

VER	HLEN	Service	Total Length	Identification	Flags	Fragmentation Offset
4 bit	4 bit	8 bits	16 bits	16 bits	3 bits	13 bits
TTL		Protocol	Header Checksum	Source IP Address		
8 bits		8 bits	16 bits	32 bits		
Destination IP Address				Option		
32 bits				32 bits		

3.3 Initial Packet

VER	HLEN	Service	Total Length	Identification	Flags	Fragmentation Offset
c	4 bit	8 bits	16 bits	16 bits	3 bits	13 bits
IPv4	24 Byte	0	72 Bytes	1C 46	(0, D, M)	0
0100	0110	0000 0000	0000 0000 0100 1000	0001 1100 0100 0110	010	0 0000 0000 0000

TTL	Protocol	Header Checksum	Source IP Address
8 bits	8 bits	16 bits	32 bits
4	TCP	2A6D	10.0.13.3 (node 1)
0000 0100	0000 0110	0010 1010 0110 1101	0000 1010 0000 0000 0000 1101 0000 0011

Destination IP Address	Option
32 bits	32 bits
10.0.14.3 (node 2)	0
0000 1010 0000 0000 0000 1110 0000 0011	0000 0000 0000 0000 0000 0000 0000 0000

Payload
64 bits
3
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011
4
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0100
3
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011
2
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0010
1
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001
2
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0010

- Packet Initialization File / Vector File

```
0100 0110 0000 0000 0000 0000 0100 1000 0001 1100 0100 0110 0100 0000 0000 0000 = 460000481C464000
0000 0100 0000 0110 0010 1010 0110 1101 0000 1010 0000 0000 0000 1101 0000 0011 = 04062A6D0A000D03
0000 1010 0000 0000 0000 1110 0000 0011 0000 0000 0000 0000 0000 0000 0000 0000 = 0A000E0300000000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011 = 0000000000000003
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0100 = 0000000000000004
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011 = 0000000000000003
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0010 = 0000000000000002
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 = 0000000000000001
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0010 = 0000000000000002
```

3.4 Packet Expected after Processing

VER	HLEN	Service	Total Length	Identification	Flags	Fragmentation Offset
4 bit	4 bit	8 bits	16 bits	16 bits	3 bits	13 bits
IPv4	24 Byte	0	72 Bytes	1C 46	(0, D, M)	0
0100	0110	0000 0000	0000 0000 0100 1000	0001 1100 0100 0110	010	0 0000 0000 0000

TTL	Protocol	Header Checksum	Source IP Address
8 bits	8 bits	16 bits	32 bits
4	TCP	2A6D	10.0.13.3 (node 1)
0000 0100	0000 0110	0010 1010 0110 1101	0000 1010 0000 0000 0000 1101 0000 0011

Destination IP Address	Option
32 bits	32 bits
10.0.14.3 (node 2)	0
0000 1010 0000 0000 0000 1110 0000 0011	0000 0000 0000 0000 0000 0000 0000 0000

Payload
64 bits
4
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0100
3
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011
2
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0010
3
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011
1
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001
2
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0010

- Packet Initialization File / Vector File

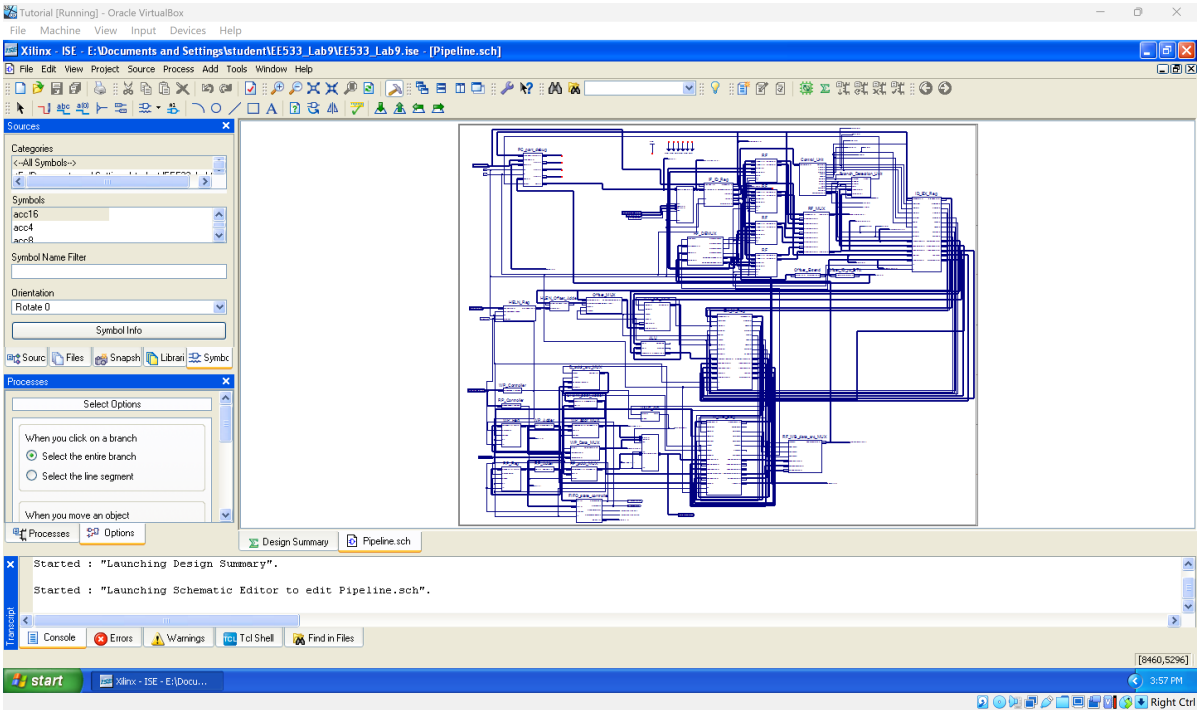
```
0100 0110 0000 0000 0000 0000 0100 1000 0001 1100 0100 0110 0100 0000 0000 0000
0000 0100 0000 0110 0010 1010 0110 1101 0000 1010 0000 0000 0000 1101 0000 0011
0000 1010 0000 0000 0000 1110 0000 0011 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0100
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0010
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0010
```

- Packet Input process

```
mode_code = 2'b00;
pkt_in = 64'h460000481C464000;

mode_code = 2'b00;
pkt_in = 64'h04062A6D0A000D03;

mode_code = 2'b00;
pkt_in = 64'h0A000E0300000000;

mode_code = 2'b00;
pkt_in = 64'h0000000000000003;

mode_code = 2'b00;
pkt_in = 64'h0000000000000004;

mode_code = 2'b00;
pkt_in = 64'h0000000000000003;

mode_code = 2'b00;
pkt_in = 64'h0000000000000002;

mode_code = 2'b00;
pkt_in = 64'h0000000000000001;

mode_code = 2'b00;
pkt_in = 64'h0000000000000002;
```

4. Pipeline

4.1 Schematic



4.2 Verilog

```
//////////////////////////////////////////////////////////////////////////////////
// Copyright (c) 1995-2008 Xilinx, Inc.  All rights reserved.
//////////////////////////////////////////////////////////////////////////////////
//   ____  ____
//  /   /\/   /
// /___/  \  /    Vendor: Xilinx
// \   \   \/     Version : 10.1
//  \   \         Application : sch2verilog
//  /   /         Filename : Pipeline.vf
// /___/   /\     Timestamp : 03/14/2025 15:40:43
// \   \  /  \
//  \___\/\___\
//
//Command: C:\Xilinx\10.1\ISE\bin\nt\unwrapped\sch2verilog.exe -intstyle ise -
family virtex2p -w "E:/Documents and Settings/student/EE533_Lab9/Pipeline.sch"
Pipeline.vf
//Design Name: Pipeline
//Device: virtex2p
//Purpose:
//    This verilog netlist is translated from an ECS schematic.It can be
//    synthesized and simulated, but it should not be modified.
//
`timescale 1ns / 1ps

module Pipeline(clk,
                HLEN,
                Instr_IN,
                Instr_IN_addr,
                Instr_IN_en,
                mode_code,
                pkt_in,
                rst,
                rst_FIFO,
```

```
                thread_IF,
                FIFO_EMPTY,
                FIFO_FULL,
                pkt_out);

 input clk;
 input [63:0] HLEN;
 input [31:0] Instr_IN;
 input [8:0] Instr_IN_addr;
 input Instr_IN_en;
 input [1:0] mode_code;
 input [63:0] pkt_in;
 input rst;
 input rst_FIFO;
 input [1:0] thread_IF;
output FIFO_EMPTY;
output FIFO_FULL;
output [63:0] pkt_out;

wire ADDI_EX;
wire ADDI_ID;
wire ADDI_M;
wire ADDI_WB;
wire [3:0] ALU_OP_EX;
wire [3:0] ALU_OP_ID;
wire [63:0] ALU_result_M;
wire [63:0] ALU_result_WB;
wire BEQ_ID;
wire BGT_ID;
wire BLT_ID;
wire [6:0] BTA;
wire [7:0] depth;
wire [7:0] D_addr_src_MUX_out;
wire [63:0] D_out_WB;
wire [7:0] D_raddr;
wire [7:0] D_waddr;
wire [63:0] D_wdata;
wire FIFO_almost_EMPTY;
wire FIFO_almost_FULL;
wire [31:0] Instruction;
wire J_ID;
wire LW_EX;
wire LW_ID;
wire LW_M;
wire LW_WB;
wire MOVI_EX;
wire MOVI_ID;
wire MOVI_M;
wire MOVI_WB;
wire NOOP_EX;
wire NOOP_ID;
wire NOOP_M;
wire [63:0] Offset_EX;
wire [63:0] Offset_ID;
wire [63:0] Offset_M;
wire [63:0] Offset_WB;
```

```
    wire [6:0] ONE;
    wire [5:0] OP_CODE_ID;
    wire [8:0] PC;
    wire PC_ctrl;
    wire [6:0] PC0;
    wire [6:0] PC1;
    wire [6:0] PC2;
    wire [6:0] PC3;
    wire [63:0] RF_WB_Din;
    wire [63:0] RF0_rs_data;
    wire [63:0] RF0_rt_data;
    wire [63:0] RF1_rs_data;
    wire [63:0] RF1_rt_data;
    wire [63:0] RF2_rs_data;
    wire [63:0] RF2_rt_data;
    wire [63:0] RF3_rs_data;
    wire [63:0] RF3_rt_data;
    wire [7:0] RP;
    wire RP_en;
    wire [7:0] RP_next;
    wire [63:0] rs_data_EX;
    wire [63:0] rs_data_ID;
    wire [63:0] rs_data_M;
    wire [4:0] rs_ID;
    wire [63:0] rt_data_EX;
    wire [63:0] rt_data_ID;
    wire [63:0] rt_data_M;
    wire [4:0] rt_EX;
    wire [4:0] rt_ID;
    wire [4:0] rt_M;
    wire [2:0] rt_WB;
    wire [7:0] SRAM_addr;
    wire SUBI_EX;
    wire SUBI_ID;
    wire SUBI_M;
    wire SUBI_WB;
    wire SW_EX;
    wire SW_ID;
    wire SW_M;
    wire [1:0] thread_EX;
    wire [1:0] thread_ID;
    wire [1:0] thread_M;
    wire [1:0] thread_WB;
    wire [2:0] waddr_DEMUX_out;
    wire [63:0] wdata_DEMUX_out;
    wire WME_EX;
    wire WME_ID;
    wire WME_M;
    wire WME_OR_out;
    wire [7:0] WP;
    wire WP_en;
    wire [7:0] WP_next;
    wire WRE_EX;
    wire WRE_ID;
    wire WRE_M;
    wire WRE_WB;
```

```
    wire [15:0] XLXN_76;
    wire XLXN_130;
    wire XLXN_131;
    wire XLXN_134;
    wire XLXN_135;
    wire [63:0] XLXN_171;
    wire [63:0] XLXN_173;
    wire [63:0] XLXN_175;
    wire [63:0] XLXN_177;
    wire [63:0] XLXN_179;
    wire FIFO_FULL_DUMMY;
    wire FIFO_EMPTY_DUMMY;
    wire [63:0] pkt_out_DUMMY;

    assign FIFO_EMPTY = FIFO_EMPTY_DUMMY;
    assign FIFO_FULL = FIFO_FULL_DUMMY;
    assign pkt_out[63:0] = pkt_out_DUMMY[63:0];
    I_MEM XLXI_14 (.addra(PC[8:0]),
                   .addrb(Instr_IN_addr[8:0]),
                   .clka(clk),
                   .clkb(clk),
                   .dinb(Instr_IN[31:0]),
                   .web(Instr_IN_en),
                   .douta(Instruction[31:0]));
    IF_ID_Reg XLXI_15 (.clk(clk),
                       .Instruction(Instruction[31:0]),
                       .rst(rst),
                       .thread_IF(thread_IF[1:0]),
                       .Offset_ID(XLXN_76[15:0]),
                       .OP_CODE_ID(OP_CODE_ID[5:0]),
                       .rs_ID(rs_ID[4:0]),
                       .rt_ID(rt_ID[4:0]),
                       .thread_ID(thread_ID[1:0]));
    RF XLXI_16 (.clk(clk),
                .rst(rst),
                .r0addr(rs_ID[2:0]),
                .r1addr(rt_ID[2:0]),
                .waddr(waddr_DEMUX_out[2:0]),
                .wdata(wdata_DEMUX_out[63:0]),
                .wena(XLXN_130),
                .r0data(RF0_rs_data[63:0]),
                .r1data(RF0_rt_data[63:0]));
    RF XLXI_17 (.clk(clk),
                .rst(rst),
                .r0addr(rs_ID[2:0]),
                .r1addr(rt_ID[2:0]),
                .waddr(waddr_DEMUX_out[2:0]),
                .wdata(wdata_DEMUX_out[63:0]),
                .wena(XLXN_131),
                .r0data(RF1_rs_data[63:0]),
                .r1data(RF1_rt_data[63:0]));
    RF XLXI_18 (.clk(clk),
                .rst(rst),
                .r0addr(rs_ID[2:0]),
                .r1addr(rt_ID[2:0]),
                .waddr(waddr_DEMUX_out[2:0]),
```

```
            .wdata(wdata_DEMUX_out[63:0]),
            .wena(XLXN_134),
            .r0data(RF2_rs_data[63:0]),
            .r1data(RF2_rt_data[63:0]));
RF XLXI_19 (.clk(clk),
            .rst(rst),
            .r0addr(rs_ID[2:0]),
            .r1addr(rt_ID[2:0]),
            .waddr(waddr_DEMUX_out[2:0]),
            .wdata(wdata_DEMUX_out[63:0]),
            .wena(XLXN_135),
            .r0data(RF3_rs_data[63:0]),
            .r1data(RF3_rt_data[63:0]));
RF_DEMUX XLXI_29 (.thread_WB(thread_WB[1:0]),
                  .waddr(rt_WB[2:0]),
                  .wdata(RF_WB_Din[63:0]),
                  .WRE_WB(WRE_WB),
                  .RF0_WRE_WB(XLXN_130),
                  .RF1_WRE_WB(XLXN_131),
                  .RF2_WRE_WB(XLXN_134),
                  .RF3_WRE_WB(XLXN_135),
                  .waddr_DEMUX_out(waddr_DEMUX_out[2:0]),
                  .wdata_DEMUX_out(wdata_DEMUX_out[63:0]));
RF_MUX XLXI_30 (.RF0_r0data(RF0_rs_data[63:0]),
                .RF0_r1data(RF0_rt_data[63:0]),
                .RF1_r0data(RF1_rs_data[63:0]),
                .RF1_r1data(RF1_rt_data[63:0]),
                .RF2_r0data(RF2_rs_data[63:0]),
                .RF2_r1data(RF2_rt_data[63:0]),
                .RF3_r0data(RF3_rs_data[63:0]),
                .RF3_r1data(RF3_rt_data[63:0]),
                .thread_ID(thread_ID[1:0]),
                .r0data(rs_data_ID[63:0]),
                .r1data(rt_data_ID[63:0]));
Control_Unit XLXI_31 (.OP_CODE(OP_CODE_ID[5:0]),
                      .ADDI_ID(ADDI_ID),
                      .ALU_OP_ID(ALU_OP_ID[3:0]),
                      .BEQ_ID(BEQ_ID),
                      .BGT_ID(BGT_ID),
                      .BLT_ID(BLT_ID),
                      .J_ID(J_ID),
                      .LW_ID(LW_ID),
                      .MOVI_ID(MOVI_ID),
                      .NOOP_ID(NOOP_ID),
                      .SUBI_ID(SUBI_ID),
                      .SW_ID(SW_ID),
                      .WME_ID(WME_ID),
                      .WRE_ID(WRE_ID));
Branch_Detection_Unit XLXI_32 (.BEQ_ID(BEQ_ID),
                               .BGT_ID(BGT_ID),
                               .BLT_ID(BLT_ID),
                               .J_ID(J_ID),
                               .rs_data(rs_data_ID[63:0]),
                               .rt_data(rt_data_ID[63:0]),
                               .PC_ctrl(PC_ctrl));
ID_EX_Reg XLXI_37 (.ADDI_ID(ADDI_ID),
```

```
                    .ALU_OP_ID(ALU_OP_ID[3:0]),
                    .clk(clk),
                    .LW_ID(LW_ID),
                    .MOVI_ID(MOVI_ID),
                    .NOOP_ID(NOOP_ID),
                    .Offset_ID(Offset_ID[63:0]),
                    .rst(rst),
                    .rs_data_ID(rs_data_ID[63:0]),
                    .rt_data_ID(rt_data_ID[63:0]),
                    .rt_ID(rt_ID[4:0]),
                    .SUBI_ID(SUBI_ID),
                    .SW_ID(SW_ID),
                    .thread_ID(thread_ID[1:0]),
                    .WME_ID(WME_ID),
                    .WRE_ID(WRE_ID),
                    .ADDI_EX(ADDI_EX),
                    .ALU_OP_EX(ALU_OP_EX[3:0]),
                    .LW_EX(LW_EX),
                    .MOVI_EX(MOVI_EX),
                    .NOOP_EX(NOOP_EX),
                    .Offset_EX(Offset_EX[63:0]),
                    .rs_data_EX(rs_data_EX[63:0]),
                    .rt_data_EX(rt_data_EX[63:0]),
                    .rt_EX(rt_EX[4:0]),
                    .SUBI_EX(SUBI_EX),
                    .SW_EX(SW_EX),
                    .thread_EX(thread_EX[1:0]),
                    .WME_EX(WME_EX),
                    .WRE_EX(WRE_EX));
ALU XLXI_38 (.A(rs_data_EX[63:0]),
             .ALU_OP(ALU_OP_EX[3:0]),
             .B(XLXN_177[63:0]),
             .ALU_Out(XLXN_179[63:0]),
             .Overflow(),
             .Zero_Flag());
HELN_Reg XLXI_39 (.clk(clk),
                  .HLEN_in(HLEN[63:0]),
                  .HLEN_Reg_write_en(WP_en),
                  .rst_FIFO(rst_FIFO),
                  .HLEN_out(XLXN_171[63:0]));
HLEN_Offset_Adder XLXI_40 (.HLEN(XLXN_171[63:0]),
                           .Offset(Offset_EX[63:0]),
                           .result(XLXN_173[63:0]));
ALU_src_MUX XLXI_41 (.ADDI_EX(ADDI_EX),
                     .LW_EX(LW_EX),
                     .Offset_EX(XLXN_175[63:0]),
                     .rt_data(rt_data_EX[63:0]),
                     .SUBI_EX(SUBI_EX),
                     .SW_EX(SW_EX),
                     .ALU_B(XLXN_177[63:0]));
Offset_MUX XLXI_42 (.HLEN_Offset_Adder_result(XLXN_173[63:0]),
                    .LW_EX(LW_EX),
                    .Offset_EX(Offset_EX[63:0]),
                    .SW_EX(SW_EX),
                    .ALU_src_MUX_in(XLXN_175[63:0]));
Offset_Extend XLXI_43 (.Offset(XLXN_76[15:0]),
```

```
                        .Offset_ID(Offset_ID[63:0]));
    EX_M_Reg XLXI_44 (.ADDI_EX(ADDI_EX),
                      .ALU_result_EX(XLXN_179[63:0]),
                      .clk(clk),
                      .LW_EX(LW_EX),
                      .MOVI_EX(MOVI_EX),
                      .NOOP_EX(NOOP_EX),
                      .Offset_EX(Offset_EX[63:0]),
                      .rst(rst),
                      .rs_data_EX(rs_data_EX[63:0]),
                      .rt_data_EX(rt_data_EX[63:0]),
                      .rt_EX(rt_EX[4:0]),
                      .SUBI_EX(SUBI_EX),
                      .SW_EX(SW_EX),
                      .thread_EX(thread_EX[1:0]),
                      .WME_EX(WME_EX),
                      .WRE_EX(WRE_EX),
                      .ADDI_M(ADDI_M),
                      .ALU_result_M(ALU_result_M[63:0]),
                      .LW_M(LW_M),
                      .MOVI_M(MOVI_M),
                      .NOOP_M(NOOP_M),
                      .Offset_M(Offset_M[63:0]),
                      .rs_data_M(rs_data_M[63:0]),
                      .rt_data_M(rt_data_M[63:0]),
                      .rt_M(rt_M[4:0]),
                      .SUBI_M(SUBI_M),
                      .SW_M(SW_M),
                      .thread_M(thread_M[1:0]),
                      .WME_M(WME_M),
                      .WRE_M(WRE_M));
    WP_Reg XLXI_51 (.clk(clk),
                    .FIFO_FULL(FIFO_FULL_DUMMY),
                    .rst(rst_FIFO),
                    .WP_en(WP_en),
                    .WP_next(WP_next[7:0]),
                    .WP(WP[7:0]));
    RP_Reg XLXI_52 (.clk(clk),
                    .FIFO_EMPTY(FIFO_EMPTY_DUMMY),
                    .RP_en(RP_en),
                    .RP_next(RP_next[7:0]),
                    .rst(rst_FIFO),
                    .RP(RP[7:0]));
    RP_Controller XLXI_53 (.mode_code(mode_code[1:0]),
                           .RP_en(RP_en));
    WP_Controller XLXI_54 (.mode_code(mode_code[1:0]),
                           .WP_en(WP_en));
    WP_Adder XLXI_58 (.WP(WP[7:0]),
                      .WP_next(WP_next[7:0]));
    RP_Adder XLXI_59 (.RP(RP[7:0]),
                      .RP_next(RP_next[7:0]));
    D_addr_src_MUX XLXI_60 (.ALU_result_M(ALU_result_M[63:0]),
                            .LW_M(LW_M),
                            .rt_M(rt_M[4:0]),
                            .SW_M(SW_M),
                            .D_addr(D_addr_src_MUX_out[7:0]));
```

```
SRAM_addr_Adder XLXI_61 (.D_addr(D_addr_src_MUX_out[7:0]),
                         .RP(RP[7:0]),
                         .D_addr_in(SRAM_addr[7:0]));
D_MEM XLXI_62 (.addra(D_waddr[7:0]),
               .addrb(D_raddr[7:0]),
               .clka(clk),
               .clkb(clk),
               .dina(D_wdata[63:0]),
               .wea(WME_OR_out),
               .doutb(pkt_out_DUMMY[63:0]));
WME_OR XLXI_63 (.WME_M(WME_M),
                .WP_en(WP_en),
                .WME(WME_OR_out));
RF_WB_data_src_MUX XLXI_65 (.ADDI_WB(ADDI_WB),
                            .ALU_out_WB(ALU_result_WB[63:0]),
                            .D_out_WB(D_out_WB[63:0]),
                            .LW_WB(LW_WB),
                            .MOVI_WB(MOVI_WB),
                            .Offset_WB(Offset_WB[63:0]),
                            .SUBI_WB(SUBI_WB),
                            .RF_WB_Din(RF_WB_Din[63:0]));
RP_addr_MUX XLXI_66 (.RP(RP[7:0]),
                     .RP_ctrl(RP_en),
                     .SRAM_addr(SRAM_addr[7:0]),
                     .D_raddr(D_raddr[7:0]));
WP_addr_MUX XLXI_67 (.SRAM_addr(SRAM_addr[7:0]),
                     .WP(WP[7:0]),
                     .WP_ctrl(WP_en),
                     .D_waddr(D_waddr[7:0]));
WP_Data_MUX XLXI_68 (.FIFO_Din(pkt_in[63:0]),
                     .SRAM_Din(rt_data_M[63:0]),
                     .WP_ctrl(WP_en),
                     .D_MEM_Din(D_wdata[63:0]));
FIFO_state_controller XLXI_69 (.clk(clk),
                               .RP(RP[7:0]),
                               .rst_FIFO(rst_FIFO),
                               .WP(WP[7:0]),
                               .depth(depth[7:0]),
                               .FIFO_almost_empty(FIFO_almost_EMPTY),
                               .FIFO_almost_full(FIFO_almost_FULL),
                               .FIFO_EMPTY(FIFO_EMPTY_DUMMY),
                               .FIFO_FULL(FIFO_FULL_DUMMY));
VCC XLXI_70 (.P(ONE[0]));
GND XLXI_71 (.G(ONE[1]));
GND XLXI_72 (.G(ONE[2]));
GND XLXI_73 (.G(ONE[3]));
GND XLXI_74 (.G(ONE[4]));
GND XLXI_75 (.G(ONE[5]));
GND XLXI_76 (.G(ONE[6]));
M_WB_Reg XLXI_80 (.ADDI_M(ADDI_M),
                  .ALU_result_M(ALU_result_M[63:0]),
                  .clk(clk),
                  .D_out_M(pkt_out_DUMMY[63:0]),
                  .LW_M(LW_M),
                  .MOVI_M(MOVI_M),
                  .NOOP_M(NOOP_M),
```

```
                    .Offset_M(Offset_M[63:0]),
                    .rst(rst),
                    .rs_data_M(rs_data_M[63:0]),
                    .rt_data_M(rt_data_M[63:0]),
                    .rt_M(rt_M[4:0]),
                    .SUBI_M(SUBI_M),
                    .SW_M(SW_M),
                    .thread_M(thread_M[1:0]),
                    .WRE_M(WRE_M),
                    .ADDI_WB(ADDI_WB),
                    .ALU_result_WB(ALU_result_WB[63:0]),
                    .D_out_WB(D_out_WB[63:0]),
                    .LW_WB(LW_WB),
                    .MOVI_WB(MOVI_WB),
                    .NOOP_WB(),
                    .Offset_WB(Offset_WB[63:0]),
                    .rs_data_WB(),
                    .rt_data_WB(),
                    .rt_WB(rt_WB[2:0]),
                    .SUBI_WB(SUBI_WB),
                    .SW_WB(),
                    .thread_WB(thread_WB[1:0]),
                    .WRE_WB(WRE_WB));
   Offset_ID_to_BTA XLXI_81 (.Offset_ID(Offset_ID[63:0]),
                             .BTA(BTA[6:0]));
   PC_part_debug XLXI_82 (.BTA(BTA[6:0]),
                          .clk(clk),
                          .ONE(ONE[6:0]),
                          .PC_ctrl(PC_ctrl),
                          .rst(rst),
                          .thread_ID(thread_ID[1:0]),
                          .thread_IF(thread_IF[1:0]),
                          .PC(PC[8:0]),
                          .PC0(PC0[6:0]),
                          .PC1(PC1[6:0]),
                          .PC2(PC2[6:0]),
                          .PC3(PC3[6:0]));
endmodule
```

4.3 Testbench

```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   15:41:55 03/14/2025
// Design Name:   Pipeline
// Module Name:   E:/Documents and
Settings/student/EE533_Lab9/EE533_Lab_9/Pipeline_tb.v
// Project Name:  EE533_Lab_9
// Target Device:
// Tool versions:
// Description:
```

```
//
// Verilog Test Fixture created by ISE for module: Pipeline
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module Pipeline_tb;

    // Inputs
    reg clk;
    reg [63:0] HLEN;
    reg [31:0] Instr_IN;
    reg [8:0] Instr_IN_addr;
    reg Instr_IN_en;
    reg [1:0] mode_code;
    reg [63:0] pkt_in;
    reg rst;
    reg rst_FIFO;
    reg [1:0] thread_IF;

    // Outputs
    wire FIFO_EMPTY;
    wire FIFO_FULL;
    wire [63:0] pkt_out;

    // Instantiate the Unit Under Test (UUT)
    Pipeline uut (
        .clk(clk),
        .HLEN(HLEN),
        .Instr_IN(Instr_IN),
        .Instr_IN_addr(Instr_IN_addr),
        .Instr_IN_en(Instr_IN_en),
        .mode_code(mode_code),
        .pkt_in(pkt_in),
        .rst(rst),
        .rst_FIFO(rst_FIFO),
        .thread_IF(thread_IF),
        .FIFO_EMPTY(FIFO_EMPTY),
        .FIFO_FULL(FIFO_FULL),
        .pkt_out(pkt_out)
    );

    always #50 clk = ~clk;
    // always #100 thread_IF[0] = ~thread_IF[0];
    // always #200 thread_IF[1] = ~thread_IF[1];

    initial begin
        // Initialize Inputs
        clk = 1;
        HLEN = 64'd3;
        Instr_IN = 0;
```

```
        Instr_IN_addr = 0;
        Instr_IN_en = 0;
        mode_code = 2'b11;
        pkt_in = 0;
        rst = 1;
        rst_FIFO = 1;
        thread_IF = 0;

        // Wait 100 ns for global reset to finish
        @(posedge clk);
        rst_FIFO = 0;

        // Add stimulus here
        // FIFO_in mode begin
        @(posedge clk);
        Instr_IN = 32'h0C010000;
        Instr_IN_addr = 9'b000000000;
        Instr_IN_en = 1;
        mode_code = 2'b00;
        pkt_in = 64'h460000481C464000;

        @(posedge clk);
        Instr_IN = 32'h04210001;
        Instr_IN_addr = 9'b000000001;
        Instr_IN_en = 1;
        mode_code = 2'b00;
        pkt_in = 64'h04062A6D0A000D03;

        @(posedge clk);
        Instr_IN = 32'h10010000;
        Instr_IN_addr = 9'b000000010;
        Instr_IN_en = 1;
        mode_code = 2'b00;
        pkt_in = 64'h0A000E0300000000;

        @(posedge clk);
        Instr_IN = 32'h0C010001;
        Instr_IN_addr = 9'b010000000;
        Instr_IN_en = 1;
        mode_code = 2'b00;
        pkt_in = 64'h0000000000000003;

        @(posedge clk);
        Instr_IN = 32'h24210001;
        Instr_IN_addr = 9'b010000001;
        Instr_IN_en = 1;
        mode_code = 2'b00;
        pkt_in = 64'h0000000000000004;

        @(posedge clk);
        Instr_IN = 32'h10010001;
        Instr_IN_addr = 9'b010000010;
        Instr_IN_en = 1;
        mode_code = 2'b00;
        pkt_in = 64'h0000000000000003;
```

```
        @(posedge clk);
        Instr_IN = 32'h0C010002;
        Instr_IN_addr = 9'b100000000;
        Instr_IN_en = 1;
        mode_code = 2'b00;
        pkt_in = 64'h0000000000000002;

        @(posedge clk);
        Instr_IN = 32'h0C020003;
        Instr_IN_addr = 9'b100000001;
        Instr_IN_en = 1;
        mode_code = 2'b00;
        pkt_in = 64'h0000000000000001;

        @(posedge clk);
        Instr_IN = 32'h18220005;
        Instr_IN_addr = 9'b100000010;
        Instr_IN_en = 1;
        mode_code = 2'b00;
        pkt_in = 64'h0000000000000002;

        @(posedge clk);
        Instr_IN = 32'h10010003;
        Instr_IN_addr = 9'b100000011;
        Instr_IN_en = 1;
        mode_code = 2'b11;
        pkt_in = 64'h0000000000000000;

        @(posedge clk);
        Instr_IN = 32'h10020002;
        Instr_IN_addr = 9'b100000100;
        Instr_IN_en = 1;
        mode_code = 2'b11;

        @(posedge clk);
        Instr_IN = 32'h00000000;
        Instr_IN_addr = 9'b100000101;
        Instr_IN_en = 1;
        mode_code = 2'b11;

        @(posedge clk);
        Instr_IN = 32'h0C010004;
        Instr_IN_addr = 9'b110000000;
        Instr_IN_en = 1;
        mode_code = 2'b11;

        @(posedge clk);
        Instr_IN = 32'h0C020005;
        Instr_IN_addr = 9'b110000001;
        Instr_IN_en = 1;
        mode_code = 2'b11;

        @(posedge clk);
        Instr_IN = 32'h18220005;
        Instr_IN_addr = 9'b110000010;
        Instr_IN_en = 1;
```

```
        mode_code = 2'b11;

        @(posedge clk);
        Instr_IN = 32'h10010005;
        Instr_IN_addr = 9'b110000011;
        Instr_IN_en = 1;
        mode_code = 2'b11;

        @(posedge clk);
        Instr_IN = 32'h10020004;
        Instr_IN_addr = 9'b110000100;
        Instr_IN_en = 1;
        mode_code = 2'b11;

        @(posedge clk);
        Instr_IN = 32'h00000000;
        Instr_IN_addr = 9'b110000101;
        Instr_IN_en = 1;
        mode_code = 2'b11;

        // Thread Processing Begin
        @(posedge clk);
        Instr_IN_en = 0;
        thread_IF = 2'b00;
        mode_code = 2'b10;
        rst = 0;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;
```

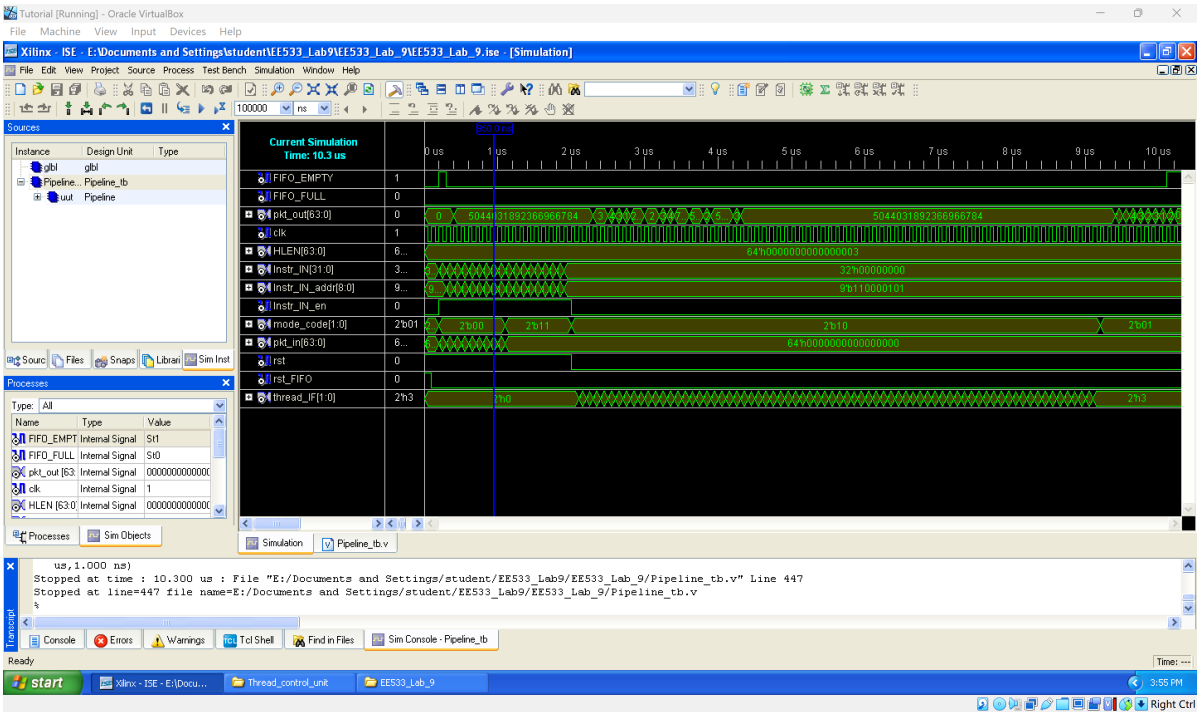
```
        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
```

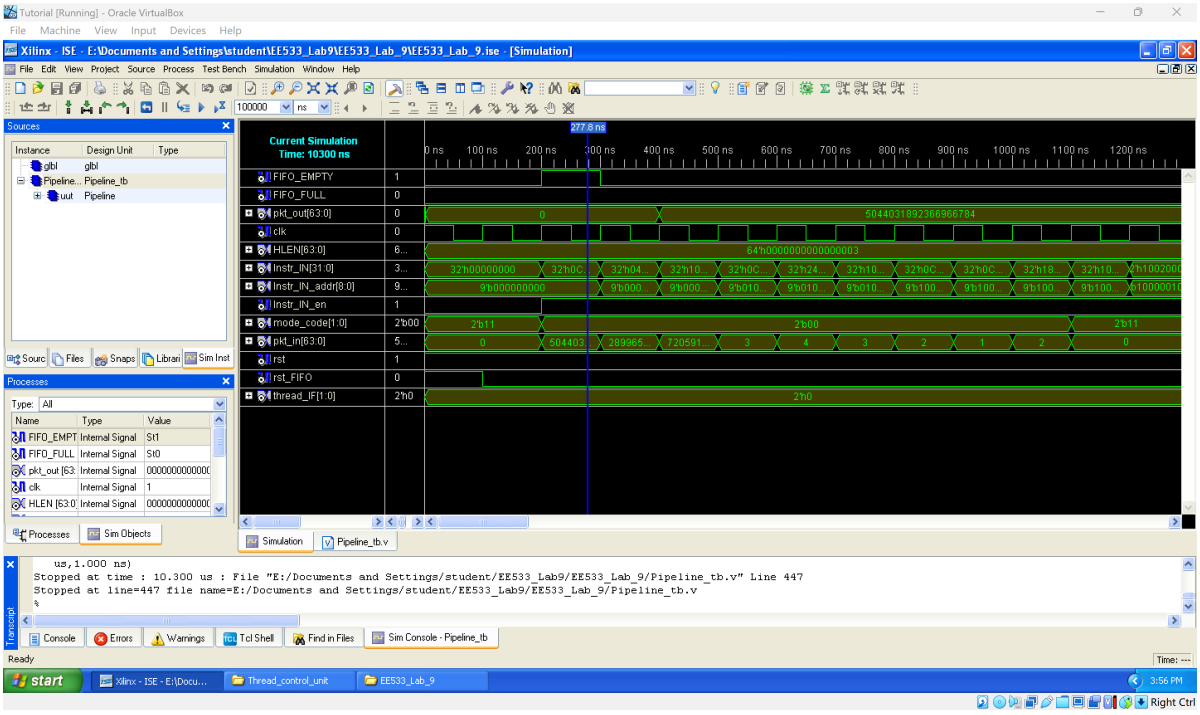
```
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;
```

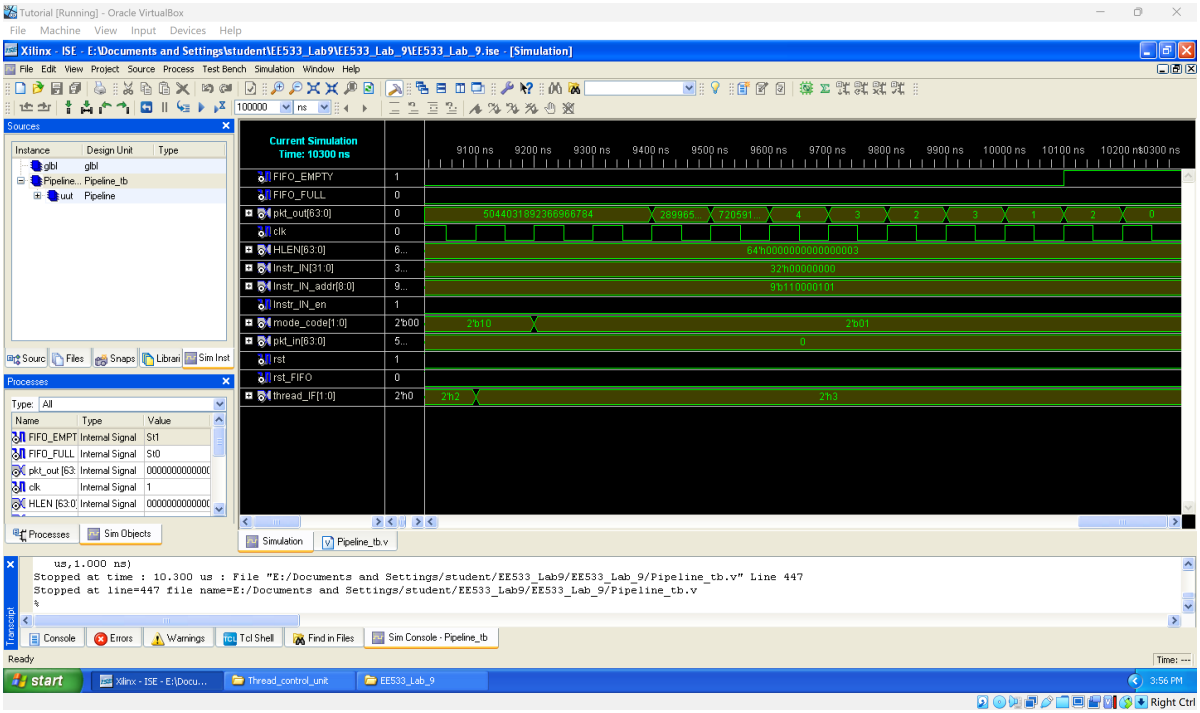
```
        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;
```

```
        @(posedge clk);
        thread_IF = 2'b11;

        @(posedge clk);
        thread_IF = 2'b00;

        @(posedge clk);
        thread_IF = 2'b01;

        @(posedge clk);
        thread_IF = 2'b10;

        @(posedge clk);
        thread_IF = 2'b11;

        // FIFO_out mode begin
        @(posedge clk);
        mode_code = 2'b01;

        @(posedge clk);

        @(posedge clk);

        @(posedge clk);

        @(posedge clk);

        @(posedge clk);

        @(posedge clk);

        @(posedge clk);

        @(posedge clk);

        @(posedge clk);

        @(posedge clk);

        @(posedge clk);

        @(posedge clk);
        $stop;

    end

endmodule
```

4.4 Waveform


Tutorial [Running] - Oracle VirtualBox
Xilinx - ISE - E:\Documents and Settings\student\EE533_Lab9\EE533_Lab_9\EE533_Lab_9.ise - [Simulation]
Current Simulation Time: 10.3 us
FIFO_EMPTY
FIFO_FULL
pkt_out[63:0]
clk
HLEN[63:0]
Instr_IN[31:0]
Instr_IN_addr[8:0]
Instr_IN_en
mode_code[1:0]
pkt_in[63:0]
rst
rst_FIFO
thread_IF[1:0]
us,1.000 ns)
Stopped at time : 10.300 us : File "E:/Documents and Settings/student/EE533_Lab9/EE533_Lab_9/Pipeline_tb.v" Line 447
Stopped at line=447 file name=E:/Documents and Settings/student/EE533_Lab9/EE533_Lab_9/Pipeline_tb.v
Ready
start
3:55 PM
Right Ctrl


Tutorial [Running] - Oracle VirtualBox
Xilinx - ISE - E:\Documents and Settings\student\EE533_Lab9\EE533_Lab_9\EE533_Lab_9.ise - [Simulation]
Current Simulation Time: 10300 ns
FIFO_EMPTY
FIFO_FULL
pkt_out[63:0]
clk
HLEN[63:0]
Instr_IN[31:0]
Instr_IN_addr[8:0]
Instr_IN_en
mode_code[1:0]
pkt_in[63:0]
rst
rst_FIFO
thread_IF[1:0]
us,1.000 ns)
Stopped at time : 10.300 us : File "E:/Documents and Settings/student/EE533_Lab9/EE533_Lab_9/Pipeline_tb.v" Line 447
Stopped at line=447 file name=E:/Documents and Settings/student/EE533_Lab9/EE533_Lab_9/Pipeline_tb.v
Ready
start
3:56 PM
Right Ctrl



5. GitHub Link

- https://github.com/yuezhenglingluan/USC_EE533_lab9.git