

EE533_Lab6_Report

1. Extending Synchronous Adder into ALU

1.1 Verilog of 64-bit ALU

```
`timescale 1ns / 1ps

module ALU (
    input  [63:0] A,
    input  [63:0] B,
    input  [3:0]  ALU_OP,
    output reg [63:0] ALU_Out,
    output reg Zero_Flag,
    output reg overflow
);

always @(*) begin
    case (ALU_OP)
        4'b0000: begin // Addition
            {overflow, ALU_Out} = A + B;
        end
        4'b0001: begin // Subtraction
            {overflow, ALU_Out} = A - B;
        end
        4'b0010: ALU_Out = A & B;           // Bitwise AND
        4'b0011: ALU_Out = A | B;           // Bitwise OR
        4'b0100: ALU_Out = A ^~ B;          // Bitwise XNOR
        4'b0101: ALU_Out = (A == B) ? 64'b1 : 64'b0; // Compare (Equality)
        4'b0110: ALU_Out = A << B[5:0];     // Logical Left Shift
        4'b0111: ALU_Out = A >> B[5:0];     // Logical Right Shift
        4'b1000: ALU_Out = substring_match(A, B); // Substring Compare
        4'b1001: ALU_Out = shift_then_compare(A, B); // Shift-then-Compare
        default: ALU_Out = 64'b0;
    endcase

    // Zero Flag
    Zero_Flag = (ALU_Out == 64'b0) ? 1'b1 : 1'b0;

end

// Function to check if B is a substring of A
function [63:0] substring_match;
    input [63:0] A, B;
    integer i;
    begin
        substring_match = 64'b0;
        for (i = 0; i < 64; i = i + 1) begin
            if ((A >> i) & B == B) begin
                substring_match = 64'b1;
            end
        end
    end
end
```

```

endfunction

// Function to shift A and then compare with B
function [63:0] shift_then_compare;
    input [63:0] A, B;
    integer i;
    begin
        shift_then_compare = 64'b0;
        for (i = 0; i < 64; i = i + 1) begin
            if ((A >> i) == B) begin
                shift_then_compare = 64'b1;
            end
        end
    end
endfunction
endmodule

```

1.2 ALU Control Signal Table

aluctrl	Operation	A	B	Expected Output
4'b0000	ADD	1	3	4
4'b0001	SUB	4	2	2
4'b0010	AND	5	7	5
4'b0011	OR	8	3	11
4'b0100	XNOR	13	3	-15
4'b0101	Compare	5	5	1
4'b0110	Left Shift	4	2	16
4'b0111	Right Shift	256	4	16
4'b1000	Substring Compare	15	3	1
4'b1001	Shift-then-Compare	15	3	0

1.3 ALU Testbench

- Verilog

```

`timescale 1ns / 1ps

/////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    14:18:38 02/19/2025
// Design Name:    ALU
// Module Name:    E:/Documents and Settings/student/EE533_Lsb6/ALU_tb.v
// Project Name:   EE533_Lsb6
// Target Device:

```

```

// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: ALU
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////

module ALU_tb;

    // Inputs
    reg [63:0] A;
    reg [63:0] B;
    reg [3:0] aluctrl;
    reg clk;

    // Outputs
    wire [63:0] ALU_Out;
    wire Zero_Flag;
    wire Overflow;

    // Instantiate the Unit Under Test (UUT)
    ALU uut (
        .A(A),
        .B(B),
        .aluctrl(aluctrl),
        .clk(clk),
        .ALU_Out(ALU_Out),
        .Zero_Flag(Zero_Flag),
        .Overflow(Overflow)
    );

    always #50 clk = ~clk;

    initial begin
        // Initialize Inputs
        A = 0;
        B = 0;
        aluctrl = 0;
        clk = 1;

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

        // Add stimulus here
        A = 64'd1;
        B = 64'd3;
        aluctrl = 4'b0000;
        #100;

        A = 64'd4;
    end

```

```

    B = 64'd2;
    aluctrl = 4'b0001;
    #100;

    A = 64'd5;
    B = 64'd7;
    aluctrl = 4'b0010;
    #100;

    A = 64'd8;
    B = 64'd3;
    aluctrl = 4'b0011;
    #100;

    A = 64'd13;
    B = 64'd3;
    aluctrl = 4'b0100;
    #100;

    A = 64'd5;
    B = 64'd5;
    aluctrl = 4'b0101;
    #100;

    A = 64'd4;
    B = 64'd2;
    aluctrl = 4'b0110;
    #100;

    A = 64'd256;
    B = 64'd4;
    aluctrl = 4'b0111;
    #100;

    A = 64'd15;
    B = 64'd3;
    aluctrl = 4'b1000;
    #100;

    A = 64'd15;
    B = 64'd5;
    aluctrl = 4'b1001;
    #100;

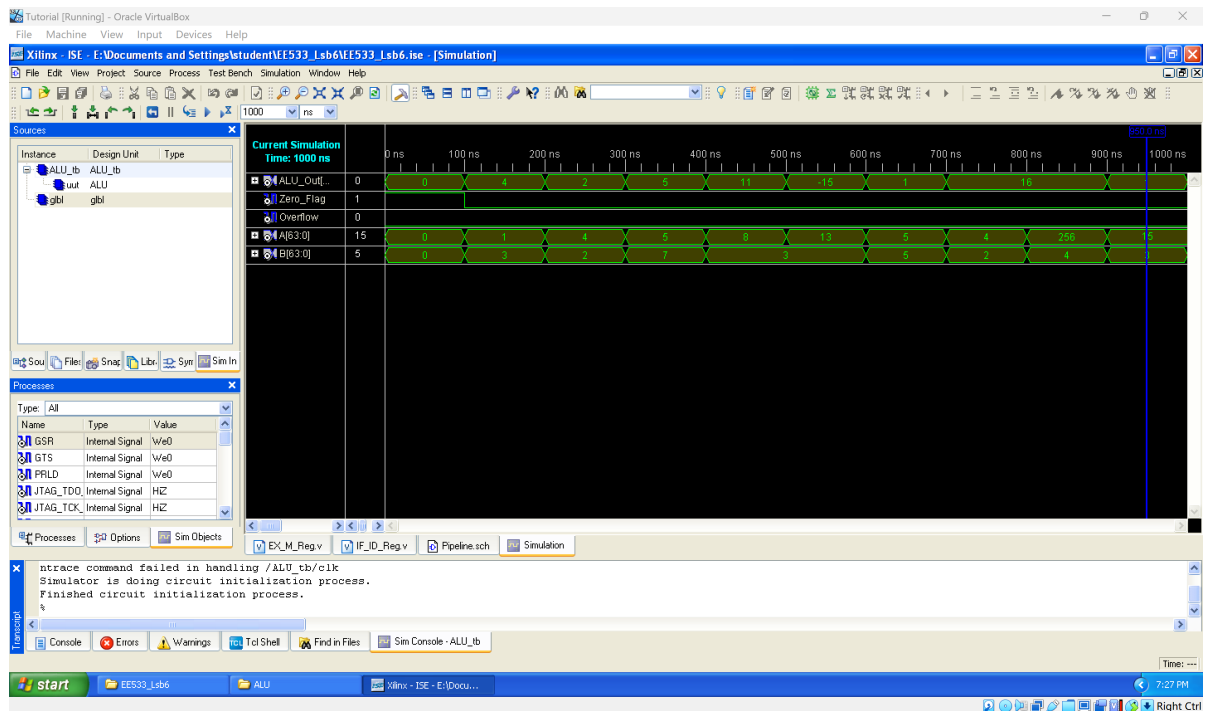
    $stop;

end

endmodule

```

- Screenshot



2. Building Register File and Memories

2.1 Register File

- Verilog

```

module RF
(
    input clk,
    input rst,
    input wena,
    input [63:0] wdata,
    input [2:0] waddr,
    input [2:0] r0addr,
    input [2:0] r1addr,

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

    reg [63:0] RF [7:0];

    integer i;

    always @(posedge clk) begin
        if (rst == 1)
            begin
                for (i = 0; i < 8; i = i + 1) begin
                    RF[i] <= 64'b0;
                end
            end
        else if (wena == 1)
            begin
                RF[waddr] <= wdata;
            end
    end

```

```

end

always @(*) begin
    r0data = ((waddr == r0addr) && wena) ? RF[waddr] : RF[r0addr];
    r1data = ((waddr == r1addr) && wena) ? RF[waddr] : RF[r1addr];
end

endmodule

```

- Testbench

```

`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    16:33:48 02/19/2025
// Design Name:    RF
// Module Name:    E:/Documents and Settings/student/EE533_Lsb6/RF_tb.v
// Project Name:   EE533_Lsb6
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: RF
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

module RF_tb;

    // Inputs
    reg clk;
    reg rst;
    reg wena;
    reg [63:0] wdata;
    reg [2:0] waddr;
    reg [2:0] r0addr;
    reg [2:0] r1addr;

    // Outputs
    wire [63:0] r0data;
    wire [63:0] r1data;

    // Instantiate the Unit Under Test (UUT)
    RF uut (
        .clk(clk),
        .rst(rst),
        .wena(wena),

```

```
.wdata(wdata),  
.waddr(waddr),  
.r0addr(r0addr),  
.r1addr(r1addr),  
.r0data(r0data),  
.r1data(r1data)  
);
```

```
always #50 clk = ~clk;
```

```
initial begin
```

```
    // Initialize Inputs
```

```
    clk = 1;  
    rst = 1;  
    wena = 0;  
    wdata = 0;  
    waddr = 0;  
    r0addr = 0;  
    r1addr = 0;
```

```
    // wait 100 ns for global reset to finish
```

```
    #100;
```

```
    rst = 0;
```

```
    // Add stimulus here
```

```
    wena = 1;  
    waddr = 3'd1;  
    wdata = 64'd17;  
    r0addr = 3'b000;  
    r1addr = 3'b001;  
    #100;
```

```
    wena = 1;  
    waddr = 3'd2;  
    wdata = 64'd85;  
    r0addr = 3'd1;  
    r1addr = 3'd1;  
    #100;
```

```
    wena = 0;  
    waddr = 3'd2;  
    wdata = 64'd17;  
    r0addr = 3'd2;  
    r1addr = 3'd3;  
    #100;
```

```
    wena = 1;  
    waddr = 3'd3;  
    wdata = 64'd17;  
    r0addr = 3'd0;  
    r1addr = 3'd2;  
    #100;
```

```
    wena = 1;  
    waddr = 3'd4;  
    wdata = 64'd7;
```

```

r0addr = 3'd2;
r1addr = 3'd3;
#100;

```

```

wena = 1;
waddr = 3'd5;
wdata = 64'd14;
r0addr = 3'd4;
r1addr = 3'd3;
#100;

```

```

wena = 1;
waddr = 3'd6;
wdata = 64'd9;
r0addr = 3'd1;
r1addr = 3'd4;
#100;

```

```

wena = 1;
waddr = 3'd7;
wdata = 64'd31;
r0addr = 3'd5;
r1addr = 3'd6;
#100;

```

```

$stop;

```

```

end

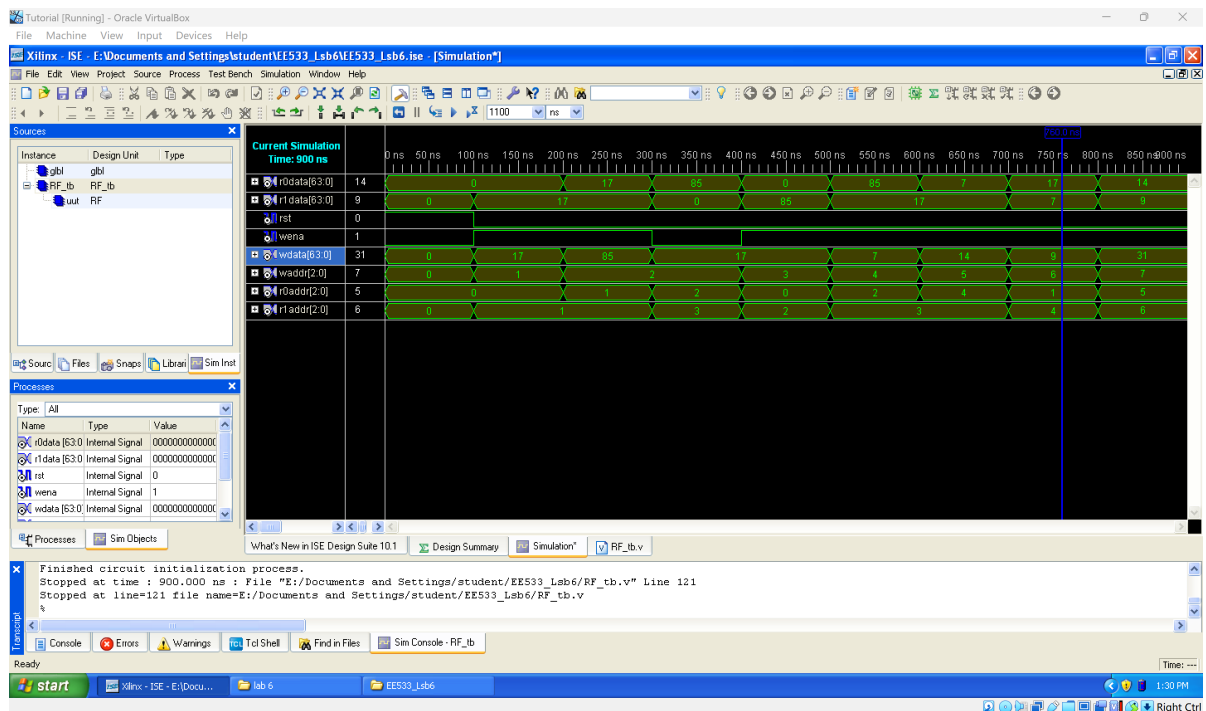
```

```

endmodule

```

- Screenshot



2.2 IP Core based single-port BRAM for Instruction Memory

- Width: 32
- Depth: 512
- Verilog

```

/*****
*   This file is owned and controlled by xilinx and must be used
*   solely for design, simulation, implementation and creation of
*   design files limited to xilinx devices or technologies. Use
*   with non-xilinx devices or technologies is expressly prohibited
*   and immediately terminates your license.
*
*   XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"
*   SOLELY FOR USE IN DEVELOPING PROGRAMS AND SOLUTIONS FOR
*   XILINX DEVICES.  BY PROVIDING THIS DESIGN, CODE, OR INFORMATION
*   AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, APPLICATION
*   OR STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS
*   IMPLEMENTATION IS FREE FROM ANY CLAIMS OF INFRINGEMENT,
*   AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE
*   FOR YOUR IMPLEMENTATION.  XILINX EXPRESSLY DISCLAIMS ANY
*   WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE
*   IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR
*   REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF
*   INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
*   FOR A PARTICULAR PURPOSE.
*
*   Xilinx products are not intended for use in life support
*   appliances, devices, or systems. Use in such applications are
*   expressly prohibited.
*
*   (c) Copyright 1995-2007 Xilinx, Inc.
*   All rights reserved.
*****/
// The synthesis directives "translate_off/translate_on" specified below are
// supported by Xilinx, Mentor Graphics and Synplify synthesis
// tools. Ensure they are correct for your synthesis tool(s).

// You must compile the wrapper file Instr_Mem.v when simulating
// the core, Instr_Mem. When compiling the wrapper file, be sure to
// reference the XilinxCoreLib Verilog simulation library. For detailed
// instructions, please refer to the "CORE Generator Help".

`timescale 1ns/1ps

module Instr_Mem(
    addr,
    clk,
    dout);

input  [8 : 0] addr;
input  clk;
output [31 : 0] dout;

```

```
// synthesis translate_off
```

```
BLKMEMSP_V6_2 #(
    .c_addr_width(9),
    .c_default_data("0"),
    .c_depth(512),
    .c_enable_rlocs(0),
    .c_has_default_data(1),
    .c_has_din(0),
    .c_has_en(0),
    .c_has_limit_data_pitch(0),
    .c_has_nd(0),
    .c_has_rdy(0),
    .c_has_rfd(0),
    .c_has_sinit(0),
    .c_has_we(0),
    .c_limit_data_pitch(18),
    .c_mem_init_file("mif_file_16_1"),
    .c_pipe_stages(0),
    .c_reg_inputs(0),
    .c_sinit_value("0"),
    .c_width(32),
    .c_write_mode(0),
    .c_ybottom_addr("0"),
    .c_ycclk_is_rising(1),
    .c_yen_is_high(1),
    .c_yhierarchy("hierarchy1"),
    .c_ymake_bmm(0),
    .c_yprimitive_type("16kx1"),
    .c_ysinit_is_high(1),
    .c_ytop_addr("1024"),
    .c_yuse_single_primitive(0),
    .c_ywe_is_high(1),
    .c_yydisable_warnings(1))
inst (
    .ADDR(addr),
    .CLK(clk),
    .DOUT(dout),
    .DIN(),
    .EN(),
    .ND(),
    .RFD(),
    .RDY(),
    .SINIT(),
    .WE());
```

```
// synthesis translate_on
```

```
// XST black box declaration
```

```
// box_type "black_box"
```

```
// synthesis attribute box_type of Instr_Mem is "black_box"
```

```
endmodule
```

- Testbench

```
`timescale 1ns / 1ps

/////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    19:39:36 02/21/2025
// Design Name:    Instr_Mem
// Module Name:    E:/Documents and Settings/student/EE533_Lsb6/Instr_Mem_tb.v
// Project Name:   EE533_Lsb6
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: Instr_Mem
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
/////////////////////////////////////////////////////////////////

module Instr_Mem_tb;

    // Inputs
    reg [8:0] addr;
    reg clk;

    // Outputs
    wire [31:0] dout;

    // Instantiate the Unit Under Test (UUT)
    Instr_Mem uut (
        .addr(addr),
        .clk(clk),
        .dout(dout)
    );

    always #50 clk = ~clk;

    initial begin
        // Initialize Inputs
        addr = 9'd255;
        clk = 1;

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

        // Add stimulus here
        @(posedge clk);
        addr = 9'd0;
    end
endmodule
```

```

@ (posedge clk);
addr = 9'd1;

@ (posedge clk);
addr = 9'd2;

@ (posedge clk);
addr = 9'd3;

@ (posedge clk);
addr = 9'd4;

@ (posedge clk);
addr = 9'd5;

@ (posedge clk);

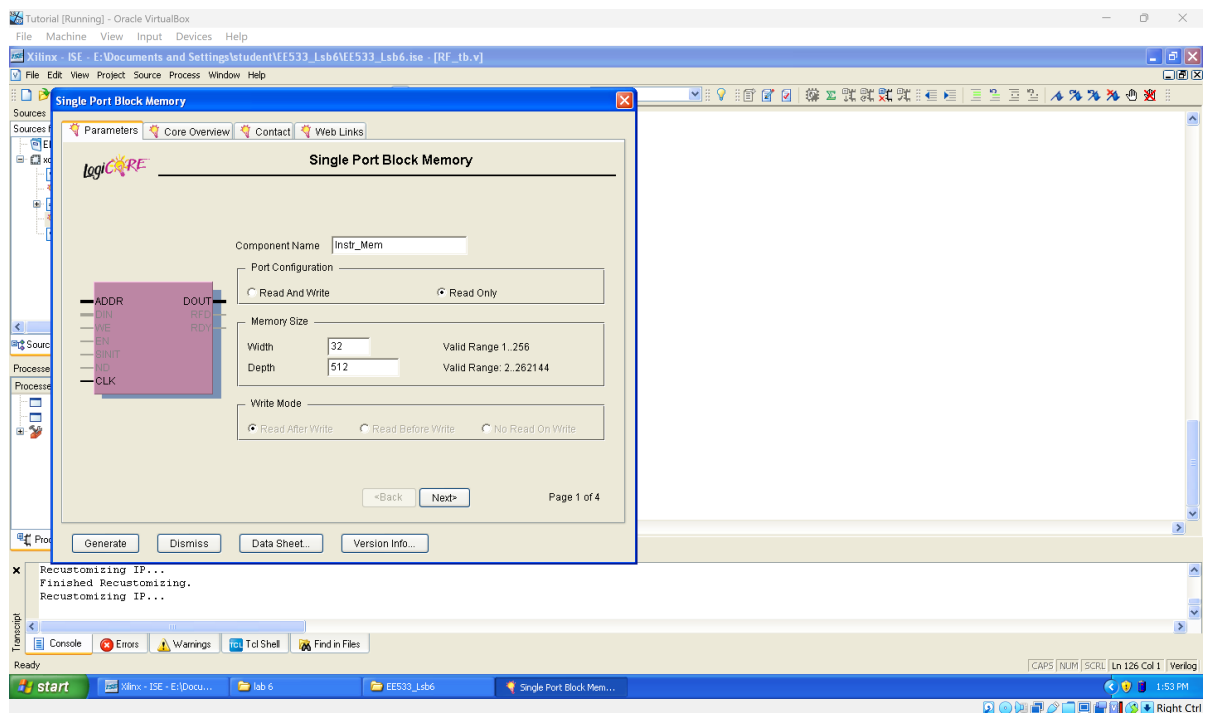
@ (posedge clk);
$stop;

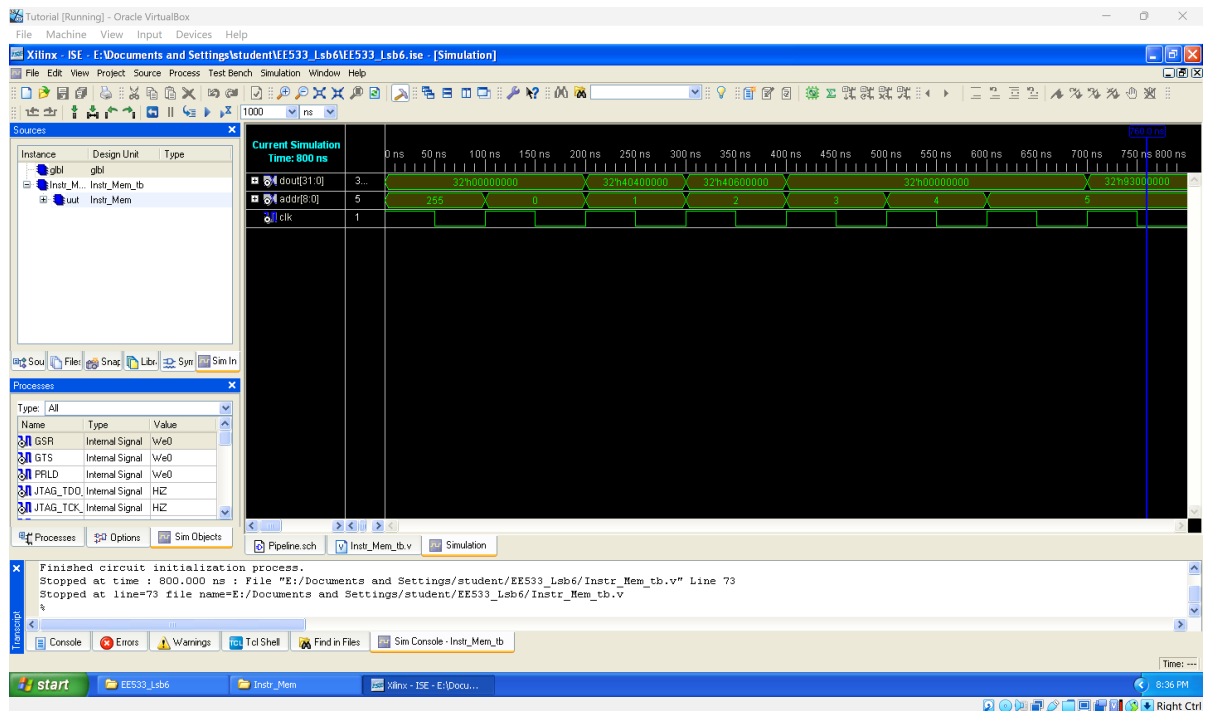
```

end

endmodule

- Screenshot





2.3 IP Core based dual-port BRAM for Data Memory

- Width: 64
- Depth: 256
- Verilog

```

/*****
*   This file is owned and controlled by Xilinx and must be used
*   solely for design, simulation, implementation and creation of
*   design files limited to Xilinx devices or technologies. Use
*   with non-Xilinx devices or technologies is expressly prohibited
*   and immediately terminates your license.
*
*   XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"
*   SOLELY FOR USE IN DEVELOPING PROGRAMS AND SOLUTIONS FOR
*   XILINX DEVICES. BY PROVIDING THIS DESIGN, CODE, OR INFORMATION
*   AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, APPLICATION
*   OR STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS
*   IMPLEMENTATION IS FREE FROM ANY CLAIMS OF INFRINGEMENT,
*   AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE
*   FOR YOUR IMPLEMENTATION. XILINX EXPRESSLY DISCLAIMS ANY
*   WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE
*   IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR
*   REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF
*   INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
*   FOR A PARTICULAR PURPOSE.
*
*   Xilinx products are not intended for use in life support
*   appliances, devices, or systems. Use in such applications are
*   expressly prohibited.
*
*   (c) Copyright 1995-2007 Xilinx, Inc.
*   All rights reserved.
*****/

// The synthesis directives "translate_off/translate_on" specified below are

```

```
// supported by Xilinx, Mentor Graphics and Synplicity synthesis
// tools. Ensure they are correct for your synthesis tool(s).

// You must compile the wrapper file Data_Mem.v when simulating
// the core, Data_Mem. When compiling the wrapper file, be sure to
// reference the XilinxCoreLib Verilog simulation library. For detailed
// instructions, please refer to the "CORE Generator Help".
```

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

```
module Data_Mem(
```

```
    addra,
    addrb,
    clka,
    clkb,
    dina,
    doutb,
    wea);
```

```
input [7 : 0] addra;
input [7 : 0] addrb;
input clka;
input clkb;
input [63 : 0] dina;
output [63 : 0] doutb;
input wea;
```

```
// synthesis translate_off
```

```
    BLKMEMDP_V6_3 #(
        .c_addra_width(8),
        .c_addrb_width(8),
        .c_default_data("0"),
        .c_depth_a(256),
        .c_depth_b(256),
        .c_enable_rlocs(0),
        .c_has_default_data(1),
        .c_has_dina(1),
        .c_has_dinb(0),
        .c_has_douta(0),
        .c_has_doutb(1),
        .c_has_ena(0),
        .c_has_enb(0),
        .c_has_limit_data_pitch(0),
        .c_has_nda(0),
        .c_has_ndb(0),
        .c_has_rdyb(0),
        .c_has_rfda(0),
        .c_has_rfdb(0),
        .c_has_sinita(0),
        .c_has_sinitb(0),
        .c_has_wea(1),
        .c_has_web(0),
        .c_limit_data_pitch(18),
```

```

.c_mem_init_file("mif_file_16_1"),
.c_pipe_stages_a(0),
.c_pipe_stages_b(0),
.c_reg_inputsa(0),
.c_reg_inputsb(0),
.c_sim_collision_check("NONE"),
.c_sinita_value("0"),
.c_sinitb_value("0"),
.c_width_a(64),
.c_width_b(64),
.c_write_modea(1),
.c_write_modeb(0),
.c_ybottom_addr("0"),
.c_yclka_is_rising(1),
.c_yclkb_is_rising(1),
.c_yena_is_high(1),
.c_yenb_is_high(1),
.c_yhierarchy("hierarchy1"),
.c_ymake_bmm(0),
.c_yprimitive_type("16kx1"),
.c_ysinita_is_high(1),
.c_ysinitb_is_high(1),
.c_ytop_addr("1024"),
.c_yuse_single_primitive(0),
.c_ywea_is_high(1),
.c_yweb_is_high(1),
.c_yydisable_warnings(1))
inst (
    .ADDRA(addr_a),
    .ADDRB(addr_b),
    .CLKA(clk_a),
    .CLKB(clk_b),
    .DINA(din_a),
    .DOUTB(dout_b),
    .WEA(wea),
    .DINB(),
    .DOUTA(),
    .ENA(),
    .ENB(),
    .NDA(),
    .NDB(),
    .RFDA(),
    .RFDB(),
    .RDYA(),
    .RDYB(),
    .SINITA(),
    .SINITB(),
    .WEB());

// synthesis translate_on

// xst black box declaration
// box_type "black_box"
// synthesis attribute box_type of Data_Mem is "black_box"

```

endmodule

- Testbench

```
`timescale 1ns / 1ps

/////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    20:46:12 02/21/2025
// Design Name:    Data_Mem
// Module Name:    E:/Documents and Settings/student/EE533_Lsb6/Data_Mem_tb.v
// Project Name:   EE533_Lsb6
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: Data_Mem
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
/////////////////////////////////////////////////////////////////

module Data_Mem_tb;

    // Inputs
    reg [7:0] addra;
    reg [7:0] addrb;
    reg clka;
    reg clkb;
    reg [63:0] dina;
    reg wea;

    // Outputs
    wire [63:0] doutb;

    // Instantiate the Unit Under Test (UUT)
    Data_Mem uut (
        .addra(addra),
        .addrb(addrb),
        .clka(clka),
        .clkb(clkb),
        .dina(dina),
        .doutb(doutb),
        .wea(wea)
    );

    always #50 clka = ~clka;
    always #50 clkb = ~clkb;
```



```

initial begin
    // Initialize Inputs
    addra = 8'd255;
    addrb = 8'd255;
    clka = 1;
    clkb = 1;
    dina = 0;
    wea = 0;

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

    // Add stimulus here
    @(posedge clka);
    addra = 8'd0;
    addrb = 8'd0;

    @(posedge clka);
    addra = 8'd1;
    addrb = 8'd1;

    @(posedge clka);
    addra = 8'd2;
    addrb = 8'd2;

    @(posedge clka);
    addra = 8'd3;
    addrb = 8'd3;

    @(posedge clka);
    addra = 8'd4;
    addrb = 8'd4;

    @(posedge clka);

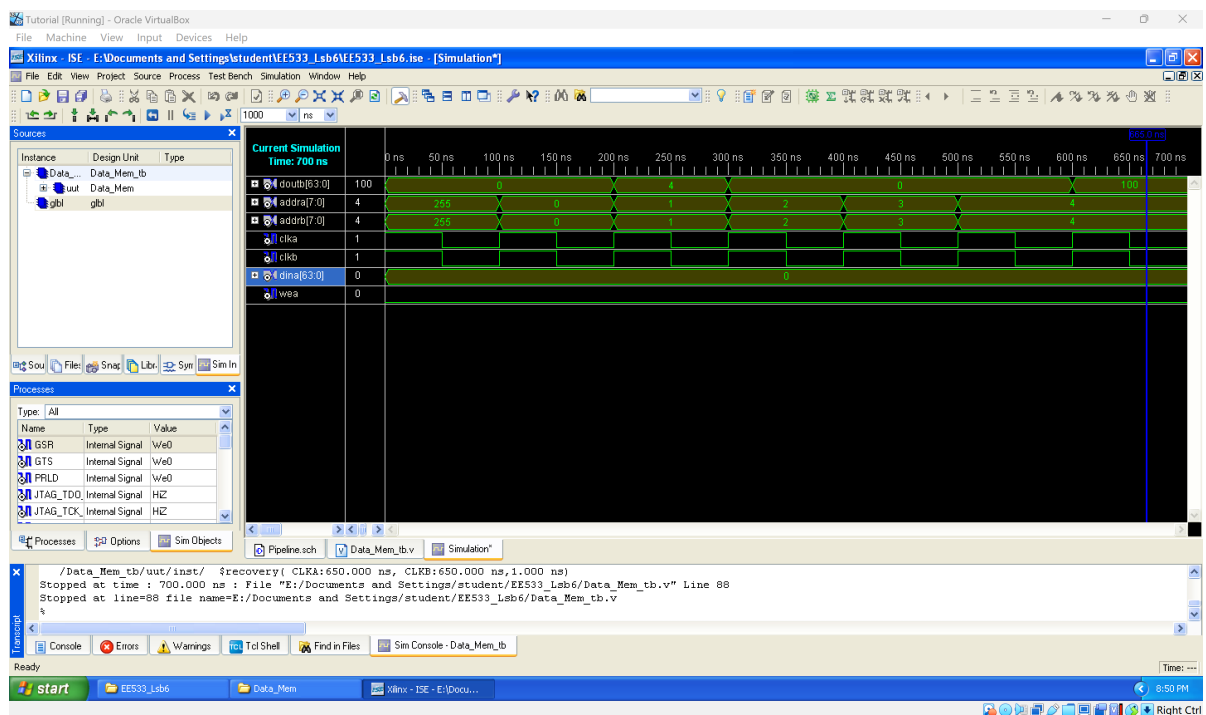
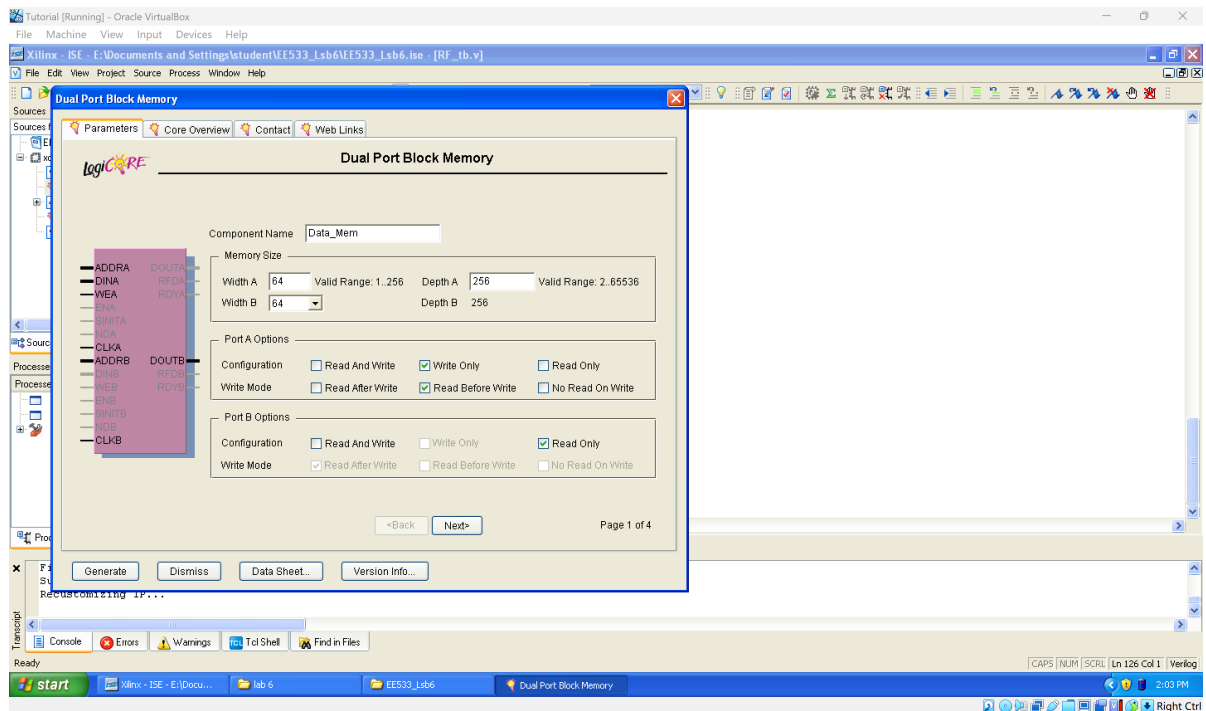
    @(posedge clka);
    $stop;

end

endmodule

```

- Screenshot



3. Building Pipeline Datapath

3.1 PC reg

- Verilog

```

`timescale 1ns / 1ps

module PC
(
    input clk,
    input rst,
    input [63:0] PC_next,

    output reg [63:0] PC
);

```

```

always @(posedge clk) begin
    if (rst)
        PC <= 63'b0;
    else
        PC <= PC_next;
end

endmodule

```

- Testbench

```

`timescale 1ns / 1ps

/////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    13:50:22 02/21/2025
// Design Name:    PC
// Module Name:    E:/Documents and Settings/student/EE533_Lsb6/PC_tb.v
// Project Name:   EE533_Lsb6
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: PC
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
/////////////////////////////////////////////////////////////////

module PC_tb;

    // Inputs
    reg clk;
    reg rst;
    reg [63:0] PC_next;

    // Outputs
    wire [63:0] PC;

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

    always #50 clk = ~clk;

```

```

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

    // wait 100 ns for global reset to finish
    #100;
    rst = 0;

    // Add stimulus here
    @(posedge clk);
    PC_next = 64'd1;

    @(posedge clk);
    PC_next = 64'd2;

    @(posedge clk);
    PC_next = 64'd3;

    @(posedge clk);
    PC_next = 64'd4;

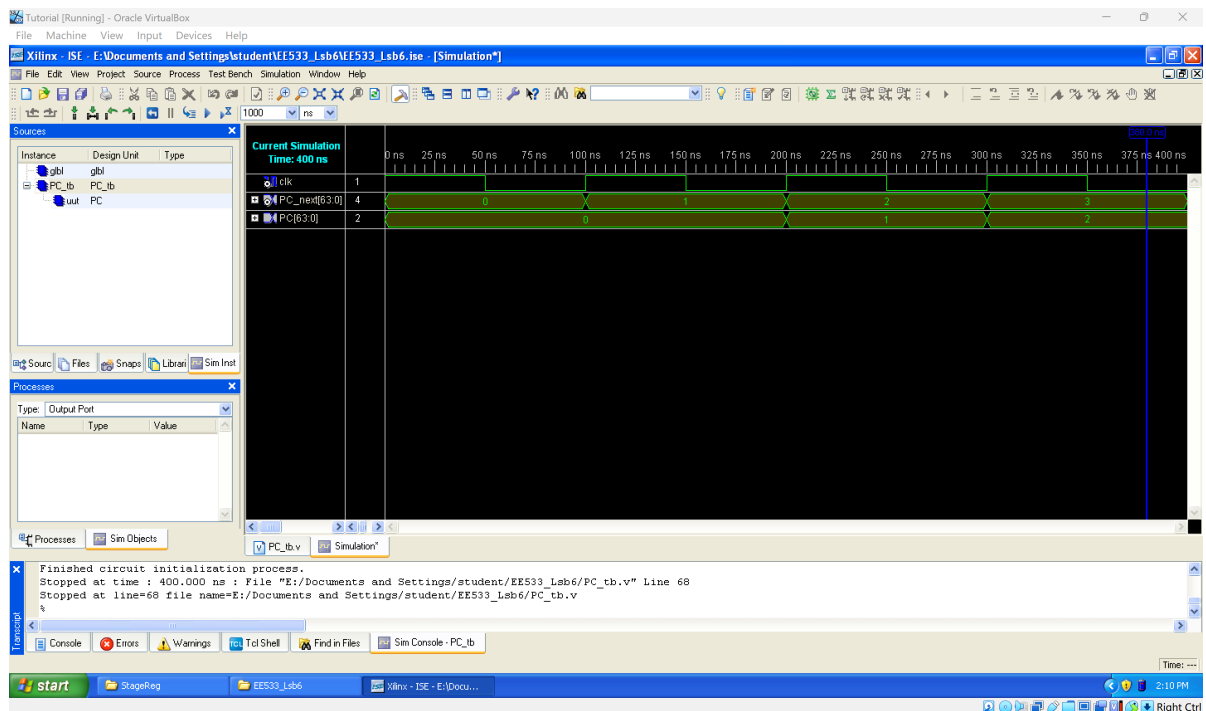
    $stop;

end

endmodule

```

- Screenshot



3.2 IF_ID_Reg

- Verilog

```
`timescale 1ns / 1ps

module IF_ID_Reg
(
    input WMemEn_IF,
    input WRegEn_IF,
    input [2:0] Reg1_IF,
    input [2:0] Reg2_IF,
    input [2:0] WReg1_IF,
    input [20:0] Unused_IF,

    output WMemEn_ID,
    output WRegEn_ID,
    output [2:0] Reg1_ID,
    output [2:0] Reg2_ID,
    output [2:0] WReg1_ID,
    output [20:0] Unused_ID
);

    assign WMemEn_ID = WMemEn_IF;
    assign WRegEn_ID = WRegEn_IF;
    assign Reg1_ID = Reg1_IF;
    assign Reg2_ID = Reg2_IF;
    assign WReg1_ID = WReg1_IF;
    assign Unused_ID = Unused_IF;

endmodule
```

- Testbench

```
`timescale 1ns / 1ps

/////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 14:30:43 02/21/2025
// Design Name: IF_ID_Reg
// Module Name: E:/Documents and Settings/student/EE533_Lsb6/IF_ID_Reg_tb.v
// Project Name: EE533_Lsb6
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: IF_ID_Reg
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
```

```

// Additional Comments:
//
/////////////////////////////////////////////////////////////////

module IF_ID_Reg_tb;

    // Inputs
    reg WMemEn_IF;
    reg WRegEn_IF;
    reg [2:0] Reg1_IF;
    reg [2:0] Reg2_IF;
    reg [2:0] WReg1_IF;
    reg [20:0] Unused_IF;

    // Outputs
    wire WMemEn_ID;
    wire WRegEn_ID;
    wire [2:0] Reg1_ID;
    wire [2:0] Reg2_ID;
    wire [2:0] WReg1_ID;
    wire [20:0] Unused_ID;

    // Instantiate the Unit Under Test (UUT)
    IF_ID_Reg uut (
        .WMemEn_IF(WMemEn_IF),
        .WRegEn_IF(WRegEn_IF),
        .Reg1_IF(Reg1_IF),
        .Reg2_IF(Reg2_IF),
        .WReg1_IF(WReg1_IF),
        .Unused_IF(Unused_IF),
        .WMemEn_ID(WMemEn_ID),
        .WRegEn_ID(WRegEn_ID),
        .Reg1_ID(Reg1_ID),
        .Reg2_ID(Reg2_ID),
        .WReg1_ID(WReg1_ID),
        .Unused_ID(Unused_ID)
    );

    initial begin
        // Initialize Inputs

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

        // Add stimulus here
        #100;
        WMemEn_IF = 1'b0;
        WRegEn_IF = 1'b1;
        Reg1_IF = 3'd0;
        Reg2_IF = 3'd0;
        WReg1_IF = 3'd2;
        Unused_IF = 21'd0;

        #100;
        WMemEn_IF = 1'b0;
        WRegEn_IF = 1'b1;
    end

```

```
Reg1_IF = 3'd0;  
Reg2_IF = 3'd0;  
WReg1_IF = 3'd3;  
Unused_IF = 21'd0;
```

```
#100;  
WMemEn_IF = 1'b0;  
WRegEn_IF = 1'b0;  
Reg1_IF = 3'd0;  
Reg2_IF = 3'd0;  
WReg1_IF = 3'd0;  
Unused_IF = 21'd0;
```

```
#100;  
WMemEn_IF = 1'b0;  
WRegEn_IF = 1'b0;  
Reg1_IF = 3'd0;  
Reg2_IF = 3'd0;  
WReg1_IF = 3'd0;  
Unused_IF = 21'd0;
```

```
#100;  
WMemEn_IF = 1'b0;  
WRegEn_IF = 1'b0;  
Reg1_IF = 3'd0;  
Reg2_IF = 3'd0;  
WReg1_IF = 3'd0;  
Unused_IF = 21'd0;
```

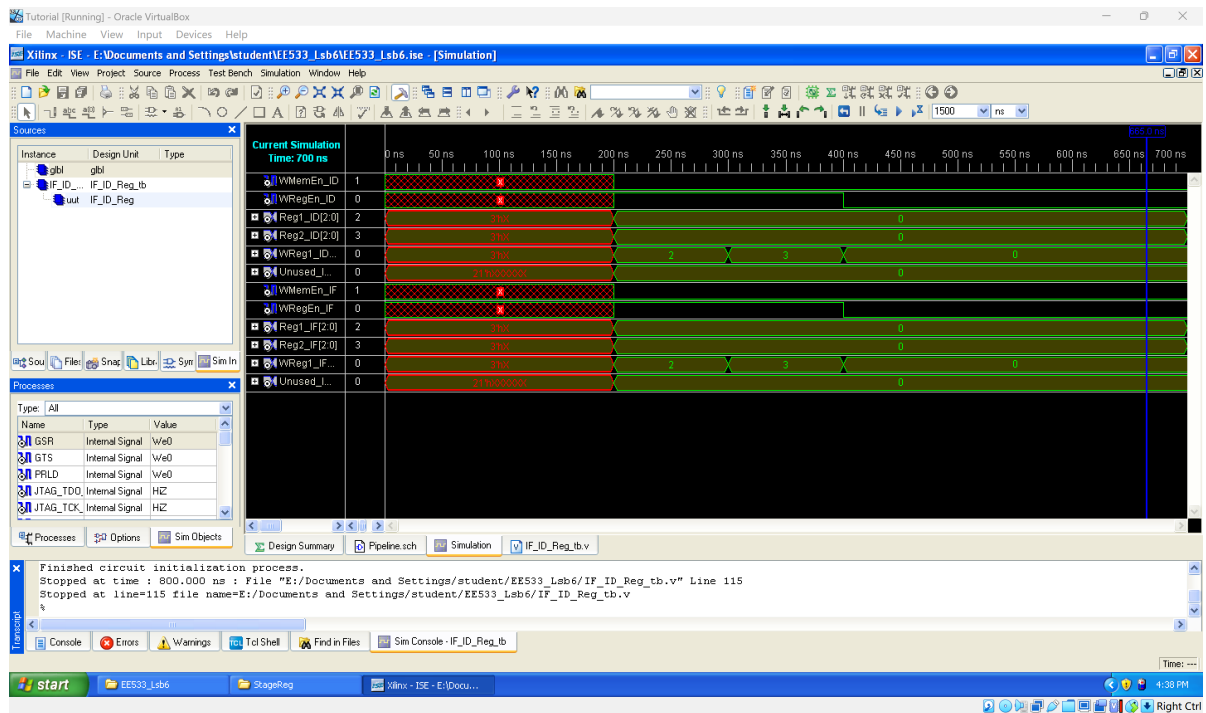
```
#100;  
WMemEn_IF = 1'b1;  
WRegEn_IF = 1'b0;  
Reg1_IF = 3'd2;  
Reg2_IF = 3'd3;  
WReg1_IF = 3'd0;  
Unused_IF = 21'd0;
```

```
#100;  
$stop;
```

```
end
```

```
endmodule
```

- Screenshot



3.3 ID_EX_Reg

- Verilog

```

`timescale 1ns / 1ps

module ID_EX_Reg
(
    input clk,
    input rst,
    input WRegEn_ID,
    input WMemEn_ID,
    input [63:0] R1_out_ID,
    input [63:0] R2_out_ID,
    input [2:0] WReg1_ID,

    output reg WRegEn_EX,
    output reg WMemEn_EX,
    output [63:0] R1_out_EX,
    output [63:0] R2_out_EX,
    output reg [2:0] WReg1_EX
);

    assign R1_out_EX = R1_out_ID;
    assign R2_out_EX = R2_out_ID;

    always @(posedge clk) begin
        if (rst) begin
            WRegEn_EX <= 0;
            WMemEn_EX <= 0;
            WReg1_EX <= 0;
        end
        else begin
            WRegEn_EX <= WRegEn_ID;
            WMemEn_EX <= WMemEn_ID;

```



```

        WReg1_EX <= WReg1_ID;

    end

end

endmodule

```

- Testbench

```

`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    15:04:47 02/21/2025
// Design Name:    ID_EX_Reg
// Module Name:    E:/Documents and Settings/student/EE533_Lsb6/ID_EX_Reg_tb.v
// Project Name:   EE533_Lsb6
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: ID_EX_Reg
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

module ID_EX_Reg_tb;

    // Inputs
    reg clk;
    reg rst;
    reg WRegEn_ID;
    reg WMemEn_ID;
    reg [63:0] R1_out_ID;
    reg [63:0] R2_out_ID;
    reg [2:0] WReg1_ID;

    // Outputs
    wire WRegEn_EX;
    wire WMemEn_EX;
    wire [63:0] R1_out_EX;
    wire [63:0] R2_out_EX;
    wire [2:0] WReg1_EX;

    // Instantiate the Unit Under Test (UUT)
    ID_EX_Reg uut (
        .clk(clk),
        .rst(rst),
        .WRegEn_ID(WRegEn_ID),

```

```

        .WMemEn_ID(WMemEn_ID),
        .R1_out_ID(R1_out_ID),
        .R2_out_ID(R2_out_ID),
        .WReg1_ID(WReg1_ID),
        .WRegEn_EX(WRegEn_EX),
        .WMemEn_EX(WMemEn_EX),
        .R1_out_EX(R1_out_EX),
        .R2_out_EX(R2_out_EX),
        .WReg1_EX(WReg1_EX)
    );

    always #50 clk = ~clk;

    initial begin
        // Initialize Inputs
        clk = 1;
        rst = 1;

        // wait 100 ns for global reset to finish
        #100;
        rst = 0;

        // Add stimulus here
        @(posedge clk);
        WRegEn_ID = 0;
        WMemEn_ID = 1;
        R1_out_ID = 64'd4;
        R2_out_ID = 64'd4;
        WReg1_ID = 3'd2;

        @(posedge clk);
        WRegEn_ID = 0;
        WMemEn_ID = 1;
        R1_out_ID = 64'd4;
        R2_out_ID = 64'd4;
        WReg1_ID = 3'd3;

        @(posedge clk);
        WRegEn_ID = 0;
        WMemEn_ID = 0;
        R1_out_ID = 64'd4;
        R2_out_ID = 64'd4;
        WReg1_ID = 3'd0;

        @(posedge clk);
        WRegEn_ID = 0;
        WMemEn_ID = 0;
        R1_out_ID = 64'd4;
        R2_out_ID = 64'd4;
        WReg1_ID = 3'd0;

        @(posedge clk);
        WRegEn_ID = 0;
        WMemEn_ID = 0;
        R1_out_ID = 64'd4;
        R2_out_ID = 64'd4;
    end

```

```
WReg1_ID = 3'd0;
```

```
@(posedge clk);
```

```
WRegEn_ID = 1;
```

```
WMemEn_ID = 0;
```

```
R1_out_ID = 64'd4;
```

```
R2_out_ID = 64'd4;
```

```
WReg1_ID = 3'd0;
```

```
@(posedge clk);
```

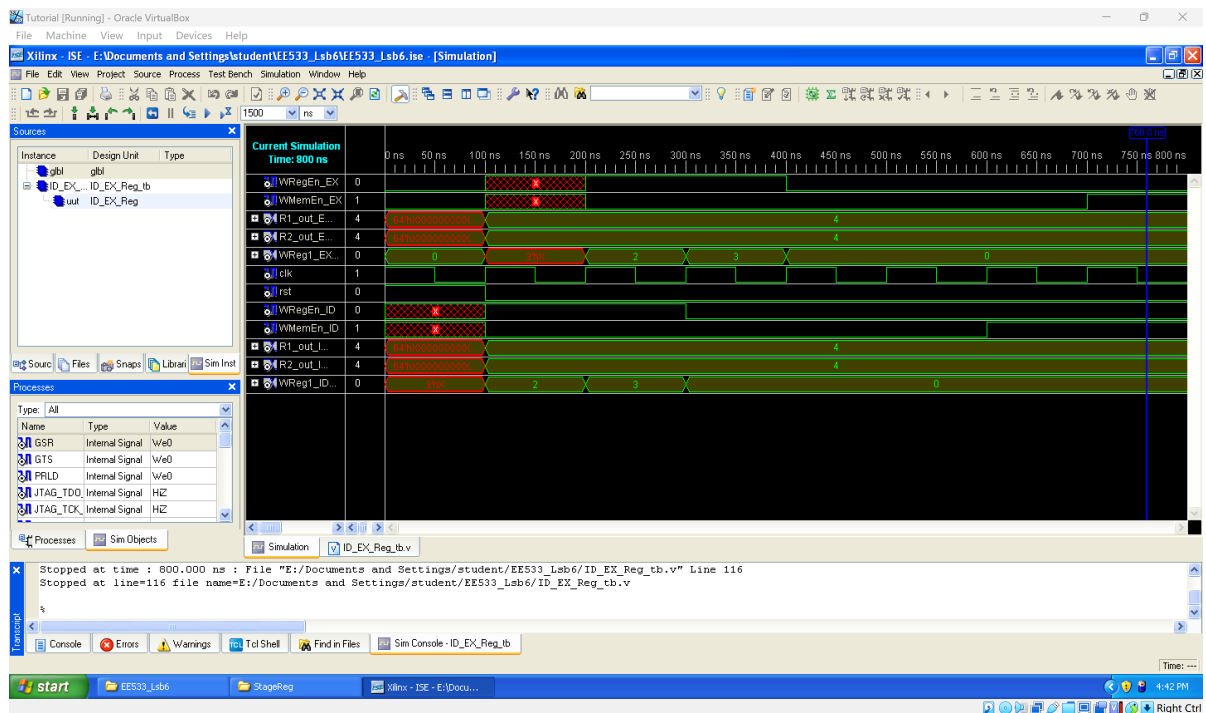
```
@(posedge clk);
```

```
$stop;
```

```
end
```

```
endmodule
```

- Screenshot



3.4 EX_M_Reg

- Verilog

```
`timescale 1ns / 1ps

module EX_M_Reg
(
    input clk,
    input rst,
    input WRegEn_EX,
    input WMemEn_EX,
    input [63:0] R1_out_EX,
    input [63:0] R2_out_EX,
    input [2:0] WReg1_EX,
```



```

reg clk;
reg rst;
reg WRegEn_EX;
reg WMemEn_EX;
reg [63:0] R1_out_EX;
reg [63:0] R2_out_EX;
reg [2:0] WReg1_EX;

// Outputs
wire WRegEn_M;
wire WMemEn_M;
wire [7:0] R1_out_M;
wire [63:0] R2_out_M;
wire [2:0] WReg1_M;

// Instantiate the Unit Under Test (UUT)
EX_M_Reg uut (
    .clk(clk),
    .rst(rst),
    .WRegEn_EX(WRegEn_EX),
    .WMemEn_EX(WMemEn_EX),
    .R1_out_EX(R1_out_EX),
    .R2_out_EX(R2_out_EX),
    .WReg1_EX(WReg1_EX),
    .WRegEn_M(WRegEn_M),
    .WMemEn_M(WMemEn_M),
    .R1_out_M(R1_out_M),
    .R2_out_M(R2_out_M),
    .WReg1_M(WReg1_M)
);

always #50 clk = ~clk;

initial begin
    // Initialize Inputs
    clk = 1;
    rst = 1;

    // wait 100 ns for global reset to finish
    #100;
    rst = 0;

    // Add stimulus here
    @(posedge clk);
    WRegEn_EX = 1;
    WMemEn_EX = 0;
    R1_out_EX = 64'd4;
    R2_out_EX = 64'd4;
    WReg1_EX = 3'd2;

    @(posedge clk);
    WRegEn_EX = 1;
    WMemEn_EX = 0;
    R1_out_EX = 64'd4;
    R2_out_EX = 64'd4;
    WReg1_EX = 3'd3;

```

```
@(posedge clk);  
WRegEn_EX = 0;  
WMemEn_EX = 0;  
R1_out_EX = 64'd4;  
R2_out_EX = 64'd4;  
WReg1_EX = 3'd0;
```

```
@(posedge clk);  
WRegEn_EX = 0;  
WMemEn_EX = 0;  
R1_out_EX = 64'd4;  
R2_out_EX = 64'd4;  
WReg1_EX = 3'd0;
```

```
@(posedge clk);  
WRegEn_EX = 0;  
WMemEn_EX = 0;  
R1_out_EX = 64'd4;  
R2_out_EX = 64'd4;  
WReg1_EX = 3'd0;
```

```
@(posedge clk);  
WRegEn_EX = 0;  
WMemEn_EX = 1;  
R1_out_EX = 64'd4;  
R2_out_EX = 64'd4;  
WReg1_EX = 3'd0;
```

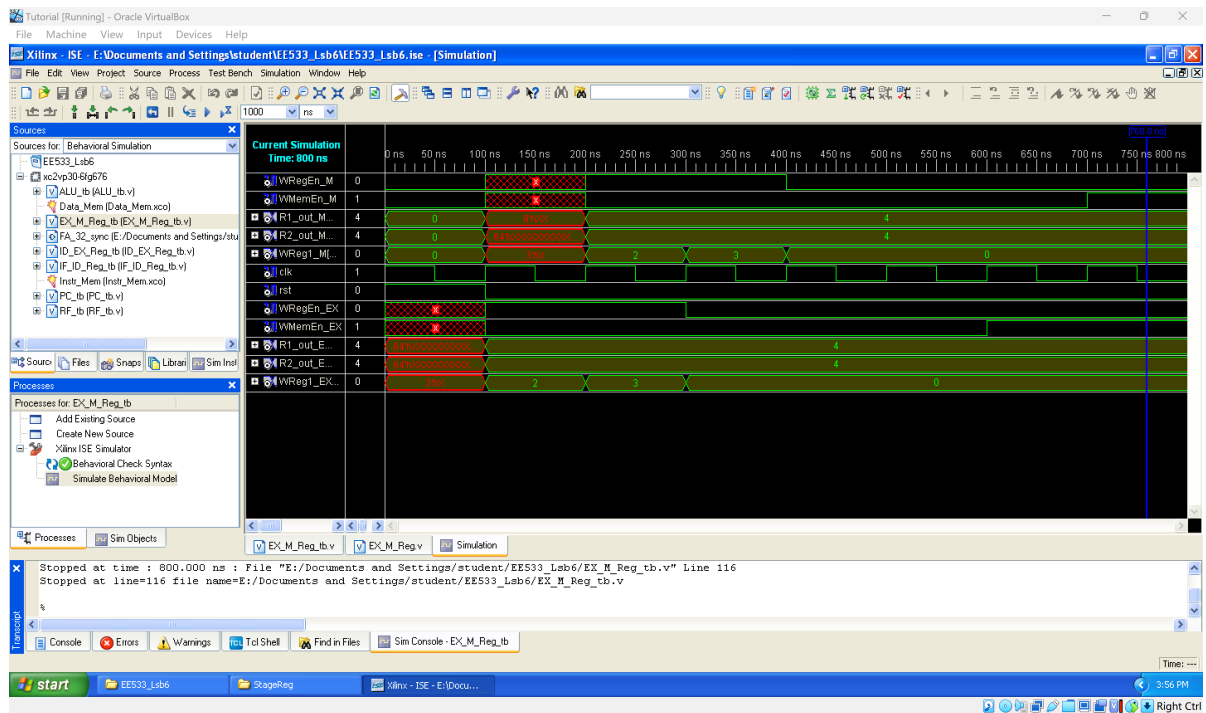
```
@(posedge clk);
```

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

```
end
```

```
endmodule
```

- Verilog



3.5 M_WB_Reg

- Verilog

```
`timescale 1ns / 1ps

module M_WB_Reg
(
    input clk,
    input rst,
    input WRegEn_M,
    input [63:0] Dout_M,
    input [2:0] WReg1_M,

    output reg WRegEn_WB,
    output reg [63:0] Dout_WB,
    output reg [2:0] WReg1_WB
);

    always @(posedge clk) begin
        if (rst) begin
            WRegEn_WB <= 0;
            Dout_WB <= 64'd0;
            WReg1_WB <= 3'd0;
        end
        else begin
            WRegEn_WB <= WRegEn_M;
            Dout_WB <= Dout_M;
            WReg1_WB <= WReg1_M;
        end
    end

endmodule
```

- Testbench

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

```
/////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 16:03:48 02/21/2025
// Design Name: M_WB_Reg
// Module Name: E:/Documents and Settings/student/EE533_Lsb6/M_WB_Reg_tb.v
// Project Name: EE533_Lsb6
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: M_WB_Reg
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
/////////////////////////////////////////////////////////////////
```

```
module M_WB_Reg_tb;
```

```
    // Inputs
```

```
    reg clk;
```

```
    reg rst;
```

```
    reg WRegEn_M;
```

```
    reg [63:0] Dout_M;
```

```
    reg [2:0] WReg1_M;
```

```
    // Outputs
```

```
    wire WRegEn_WB;
```

```
    wire [63:0] Dout_WB;
```

```
    wire [2:0] WReg1_WB;
```

```
    // Instantiate the Unit Under Test (UUT)
```

```
    M_WB_Reg uut (
```

```
        .clk(clk),
```

```
        .rst(rst),
```

```
        .WRegEn_M(WRegEn_M),
```

```
        .Dout_M(Dout_M),
```

```
        .WReg1_M(WReg1_M),
```

```
        .WRegEn_WB(WRegEn_WB),
```

```
        .Dout_WB(Dout_WB),
```

```
        .WReg1_WB(WReg1_WB)
```

```
    );
```

```
    always #50 clk = ~clk;
```

```
    initial begin
```

```
        // Initialize Inputs
```

```
        clk = 1;
```



```

rst = 1;

// wait 100 ns for global reset to finish
#100;
rst = 0;

// Add stimulus here
@(posedge clk);
WRegEn_M = 1;
Dout_M = 64'd4;
WReg1_M = 3'd2;

@(posedge clk);
WRegEn_M = 1;
Dout_M = 64'd4;
WReg1_M = 3'd3;

@(posedge clk);
WRegEn_M = 0;
Dout_M = 64'd4;
WReg1_M = 3'd0;

@(posedge clk);
WRegEn_M = 0;
Dout_M = 64'd4;
WReg1_M = 3'd0;

@(posedge clk);
WRegEn_M = 0;
Dout_M = 64'd4;
WReg1_M = 3'd0;

@(posedge clk);
WRegEn_M = 0;
Dout_M = 64'd4;
WReg1_M = 3'd0;

@(posedge clk);

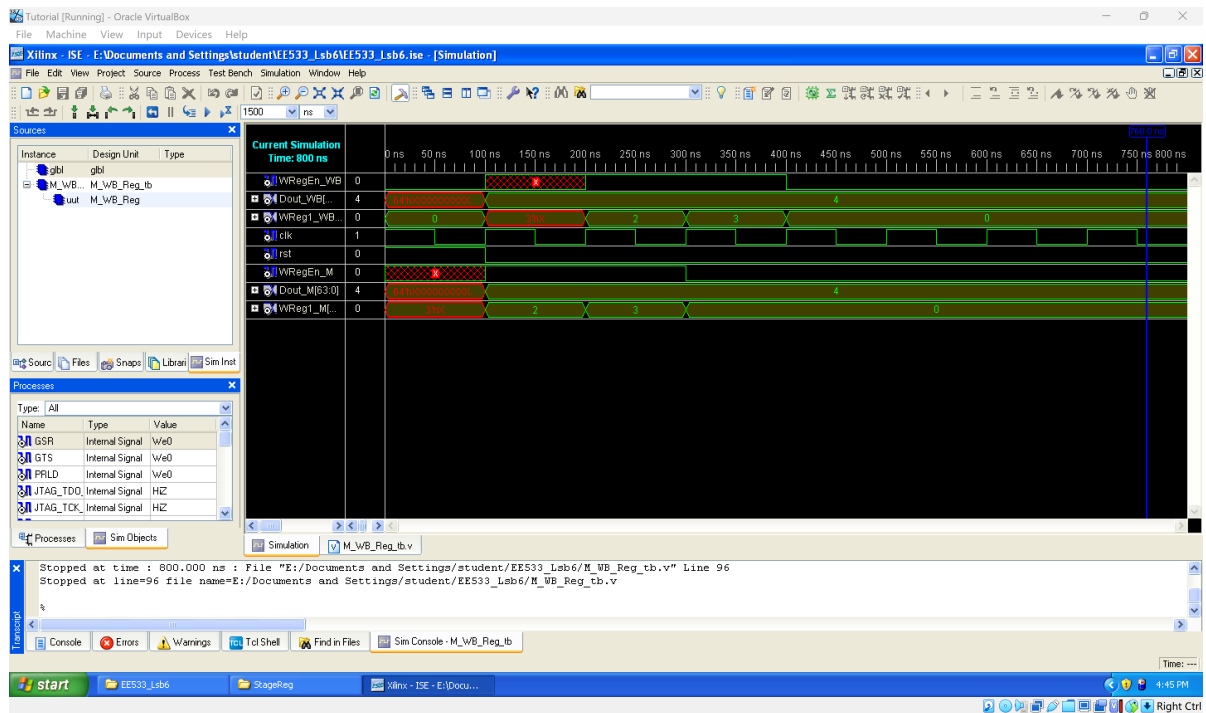
@stop;

end

endmodule

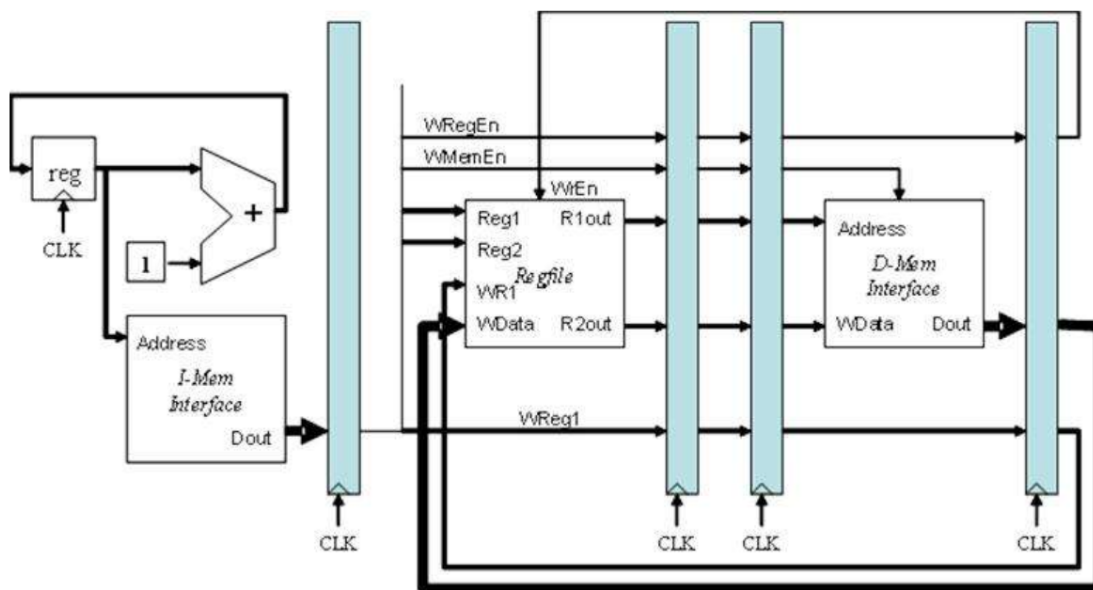
```

- Screenshot

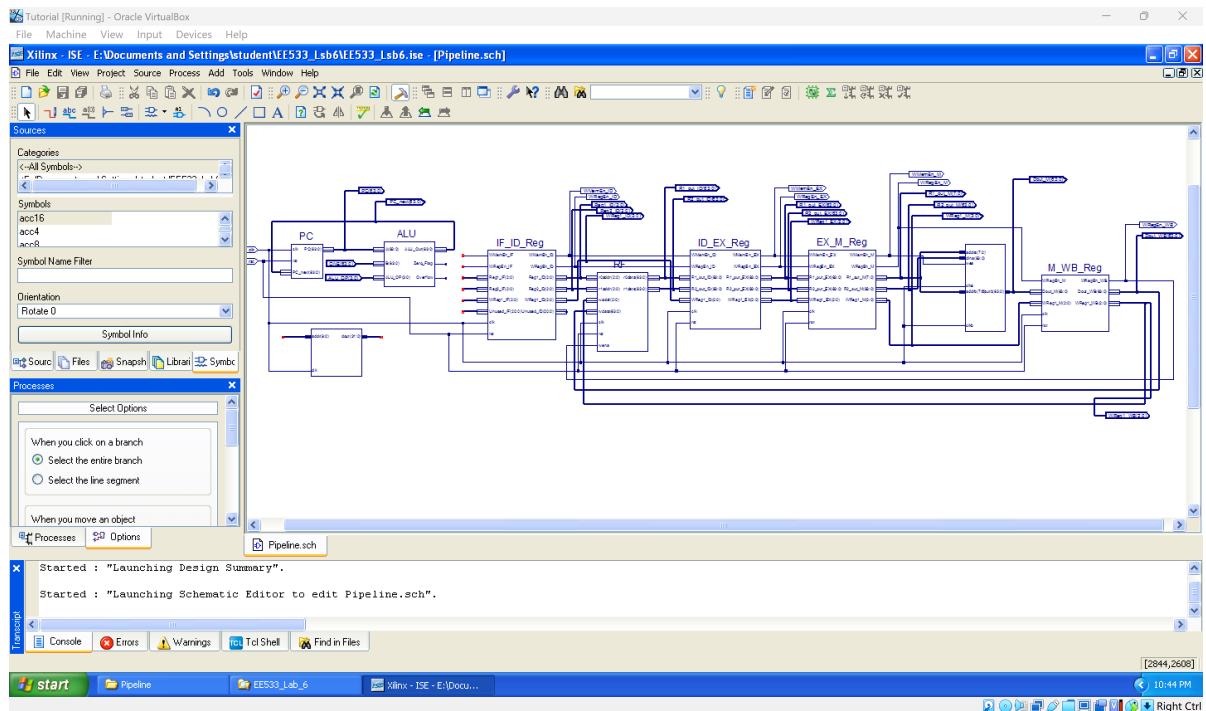


3.6 The Final Pipeline Datapath

- We Built the Datapath following Figure 3 in the lab instructions:



- Schematic



- Verilog

```

////////////////////////////////////
// Copyright (c) 1995-2008 xilinx, Inc. All rights reserved.
////////////////////////////////////
//      ____
//  /  /  \  /
// /___/  \ /      vendor: xilinx
// \  \  \ /      version : 10.1
// \  \      Application : sch2verilog
// /  /      Filename : Pipeline.vf
// /___/  \      Timestamp : 02/21/2025 22:24:18
// \  \  /  \
// \___\ \___\
//
//Command: C:\Xilinx\10.1\ISE\bin\nt\unwrapped\sch2verilog.exe -intstyle ise -
family virtex2p -w "E:/Documents and Settings/student/EE533_Lsb6/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(ALU_OP,
               clk,
               ONE,
               rst,
               Dout_M,
               Dout_WB,
               PC,
               PC_next,
               Reg1_ID,
               Reg2_ID,

```

```

        R1_out_EX,
        R1_out_ID,
        R1_out_M,
        R2_out_EX,
        R2_out_ID,
        R2_out_M,
        WMemEn_EX,
        WMemEn_ID,
        WMemEn_M,
        WRegEn_EX,
        WRegEn_ID,
        WRegEn_M,
        WRegEn_WB,
        WReg1_EX,
        WReg1_ID,
        WReg1_M,
        WReg1_WB);

input  [3:0]  ALU_OP;
input  clk;
input  [63:0] ONE;
input  rst;
output [63:0] Dout_M;
output [63:0] Dout_WB;
output [63:0] PC;
output [63:0] PC_next;
output [2:0]  Reg1_ID;
output [2:0]  Reg2_ID;
output [63:0] R1_out_EX;
output [63:0] R1_out_ID;
output [7:0]  R1_out_M;
output [63:0] R2_out_EX;
output [63:0] R2_out_ID;
output [63:0] R2_out_M;
output WMemEn_EX;
output WMemEn_ID;
output WMemEn_M;
output WRegEn_EX;
output WRegEn_ID;
output WRegEn_M;
output WRegEn_WB;
output [2:0]  WReg1_EX;
output [2:0]  WReg1_ID;
output [2:0]  WReg1_M;
output [2:0]  WReg1_WB;

wire [31:0] Instr;
wire [2:0]  WReg1_WB_DUMMY;
wire [2:0]  WReg1_EX_DUMMY;
wire WRegEn_M_DUMMY;
wire [63:0] R2_out_M_DUMMY;
wire [2:0]  Reg2_ID_DUMMY;
wire WMemEn_ID_DUMMY;
wire [63:0] Dout_M_DUMMY;
wire WRegEn_ID_DUMMY;
wire [63:0] PC_DUMMY;

```

```

wire [63:0] R2_out_EX_DUMMY;
wire [63:0] R1_out_ID_DUMMY;
wire [2:0] WReg1_M_DUMMY;
wire WRegEn_WB_DUMMY;
wire WMemEn_EX_DUMMY;
wire [2:0] WReg1_ID_DUMMY;
wire WRegEn_EX_DUMMY;
wire [63:0] Dout_WB_DUMMY;
wire WMemEn_M_DUMMY;
wire [7:0] R1_out_M_DUMMY;
wire [63:0] R1_out_EX_DUMMY;
wire [63:0] PC_next_DUMMY;
wire [63:0] R2_out_ID_DUMMY;
wire [2:0] Reg1_ID_DUMMY;

assign Dout_M[63:0] = Dout_M_DUMMY[63:0];
assign Dout_WB[63:0] = Dout_WB_DUMMY[63:0];
assign PC[63:0] = PC_DUMMY[63:0];
assign PC_next[63:0] = PC_next_DUMMY[63:0];
assign Reg1_ID[2:0] = Reg1_ID_DUMMY[2:0];
assign Reg2_ID[2:0] = Reg2_ID_DUMMY[2:0];
assign R1_out_EX[63:0] = R1_out_EX_DUMMY[63:0];
assign R1_out_ID[63:0] = R1_out_ID_DUMMY[63:0];
assign R1_out_M[7:0] = R1_out_M_DUMMY[7:0];
assign R2_out_EX[63:0] = R2_out_EX_DUMMY[63:0];
assign R2_out_ID[63:0] = R2_out_ID_DUMMY[63:0];
assign R2_out_M[63:0] = R2_out_M_DUMMY[63:0];
assign WMemEn_EX = WMemEn_EX_DUMMY;
assign WMemEn_ID = WMemEn_ID_DUMMY;
assign WMemEn_M = WMemEn_M_DUMMY;
assign WRegEn_EX = WRegEn_EX_DUMMY;
assign WRegEn_ID = WRegEn_ID_DUMMY;
assign WRegEn_M = WRegEn_M_DUMMY;
assign WRegEn_WB = WRegEn_WB_DUMMY;
assign WReg1_EX[2:0] = WReg1_EX_DUMMY[2:0];
assign WReg1_ID[2:0] = WReg1_ID_DUMMY[2:0];
assign WReg1_M[2:0] = WReg1_M_DUMMY[2:0];
assign WReg1_WB[2:0] = WReg1_WB_DUMMY[2:0];
ALU XLXI_1 (.A(PC_DUMMY[63:0]),
            .ALU_OP(ALU_OP[3:0]),
            .B(ONE[63:0]),
            .ALU_Out(PC_next_DUMMY[63:0]),
            .Overflow(),
            .Zero_Flag());
Instr_Mem XLXI_2 (.addr(PC_DUMMY[8:0]),
                  .clk(clk),
                  .dout(Instr[31:0]));
IF_ID_Reg XLXI_3 (.clk(clk),
                  .Reg1_IF(Instr[29:27]),
                  .Reg2_IF(Instr[26:24]),
                  .rst(rst),
                  .Unused_IF(Instr[20:0]),
                  .WMemEn_IF(Instr[31]),
                  .WRegEn_IF(Instr[30]),
                  .WReg1_IF(Instr[23:21]),
                  .Reg1_ID(Reg1_ID_DUMMY[2:0]),

```

```

        .Reg2_ID(Reg2_ID_DUMMY[2:0]),
        .Unused_ID(),
        .WMemEn_ID(WMemEn_ID_DUMMY),
        .WRegEn_ID(WRegEn_ID_DUMMY),
        .WReg1_ID(WReg1_ID_DUMMY[2:0]));
PC XLXI_4 (.clk(clk),
        .PC_next(PC_next_DUMMY[63:0]),
        .rst(rst),
        .PC(PC_DUMMY[63:0]));
RF XLXI_5 (.clk(clk),
        .rst(rst),
        .r0addr(Reg1_ID_DUMMY[2:0]),
        .r1addr(Reg2_ID_DUMMY[2:0]),
        .waddr(WReg1_WB_DUMMY[2:0]),
        .wdata(Dout_WB_DUMMY[63:0]),
        .wena(WRegEn_WB_DUMMY),
        .r0data(R1_out_ID_DUMMY[63:0]),
        .r1data(R2_out_ID_DUMMY[63:0]));
ID_EX_Reg XLXI_6 (.clk(clk),
        .rst(rst),
        .R1_out_ID(R1_out_ID_DUMMY[63:0]),
        .R2_out_ID(R2_out_ID_DUMMY[63:0]),
        .WMemEn_ID(WMemEn_ID_DUMMY),
        .WRegEn_ID(WRegEn_ID_DUMMY),
        .WReg1_ID(WReg1_ID_DUMMY[2:0]),
        .R1_out_EX(R1_out_EX_DUMMY[63:0]),
        .R2_out_EX(R2_out_EX_DUMMY[63:0]),
        .WMemEn_EX(WMemEn_EX_DUMMY),
        .WRegEn_EX(WRegEn_EX_DUMMY),
        .WReg1_EX(WReg1_EX_DUMMY[2:0]));
EX_M_Reg XLXI_7 (.clk(clk),
        .rst(rst),
        .R1_out_EX(R1_out_EX_DUMMY[63:0]),
        .R2_out_EX(R2_out_EX_DUMMY[63:0]),
        .WMemEn_EX(WMemEn_EX_DUMMY),
        .WRegEn_EX(WRegEn_EX_DUMMY),
        .WReg1_EX(WReg1_EX_DUMMY[2:0]),
        .R1_out_M(R1_out_M_DUMMY[7:0]),
        .R2_out_M(R2_out_M_DUMMY[63:0]),
        .WMemEn_M(WMemEn_M_DUMMY),
        .WRegEn_M(WRegEn_M_DUMMY),
        .WReg1_M(WReg1_M_DUMMY[2:0]));
Data_Mem XLXI_8 (.addra(R1_out_M_DUMMY[7:0]),
        .addrb(R1_out_M_DUMMY[7:0]),
        .clka(clk),
        .clkb(clk),
        .dina(R2_out_M_DUMMY[63:0]),
        .wea(WMemEn_M_DUMMY),
        .doutb(Dout_M_DUMMY[63:0]));
M_WB_Reg XLXI_9 (.clk(clk),
        .Dout_M(Dout_M_DUMMY[63:0]),
        .rst(rst),
        .WRegEn_M(WRegEn_M_DUMMY),
        .WReg1_M(WReg1_M_DUMMY[2:0]),
        .Dout_WB(Dout_WB_DUMMY[63:0]),
        .WRegEn_WB(WRegEn_WB_DUMMY),

```

```

        .WReg1_WB(WReg1_WB_DUMMY[2:0])));
endmodule

```

- Testbench

```

`timescale 1ns / 1ps

/////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    22:27:30 02/21/2025
// Design Name:    Pipeline
// Module Name:    E:/Documents and
Settings/student/EE533_Lsb6/EE533_Lab_6/Pipeline_tb.v
// Project Name:   EE533_Lab_6
// 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 [3:0] ALU_OP;
    reg clk;
    reg [63:0] ONE;
    reg rst;

    // Outputs
    wire [63:0] Dout_M;
    wire [63:0] Dout_WB;
    wire [63:0] PC;
    wire [63:0] PC_next;
    wire [2:0] Reg1_ID;
    wire [2:0] Reg2_ID;
    wire [63:0] R1_out_EX;
    wire [63:0] R1_out_ID;
    wire [7:0] R1_out_M;
    wire [63:0] R2_out_EX;
    wire [63:0] R2_out_ID;
    wire [63:0] R2_out_M;
    wire WMemEn_EX;
    wire WMemEn_ID;
    wire WMemEn_M;
    wire WRegEn_EX;

```

```

wire WRegEn_ID;
wire WRegEn_M;
wire WRegEn_WB;
wire [2:0] WReg1_EX;
wire [2:0] WReg1_ID;
wire [2:0] WReg1_M;
wire [2:0] WReg1_WB;

// Instantiate the Unit Under Test (UUT)
Pipeline uut (
    .ALU_OP(ALU_OP),
    .clk(clk),
    .ONE(ONE),
    .rst(rst),
    .Dout_M(Dout_M),
    .Dout_WB(Dout_WB),
    .PC(PC),
    .PC_next(PC_next),
    .Reg1_ID(Reg1_ID),
    .Reg2_ID(Reg2_ID),
    .R1_out_EX(R1_out_EX),
    .R1_out_ID(R1_out_ID),
    .R1_out_M(R1_out_M),
    .R2_out_EX(R2_out_EX),
    .R2_out_ID(R2_out_ID),
    .R2_out_M(R2_out_M),
    .WMemEn_EX(WMemEn_EX),
    .WMemEn_ID(WMemEn_ID),
    .WMemEn_M(WMemEn_M),
    .WRegEn_EX(WRegEn_EX),
    .WRegEn_ID(WRegEn_ID),
    .WRegEn_M(WRegEn_M),
    .WRegEn_WB(WRegEn_WB),
    .WReg1_EX(WReg1_EX),
    .WReg1_ID(WReg1_ID),
    .WReg1_M(WReg1_M),
    .WReg1_WB(WReg1_WB)
);

always #50 clk = ~clk;

initial begin
    // Initialize Inputs
    ALU_OP = 0;
    clk = 1;
    ONE = 1;
    rst = 1;

    // wait 100 ns for global reset to finish
    #100;
    rst = 0;

    // Add stimulus here
    @(posedge clk);

    @(posedge clk);

```



```

    @(posedge clk);

    @(posedge clk);

    @(posedge clk);

    @(posedge clk);

    @(posedge clk);

    @(posedge clk);

    @(posedge clk);

    @(posedge clk);

    @(posedge clk);

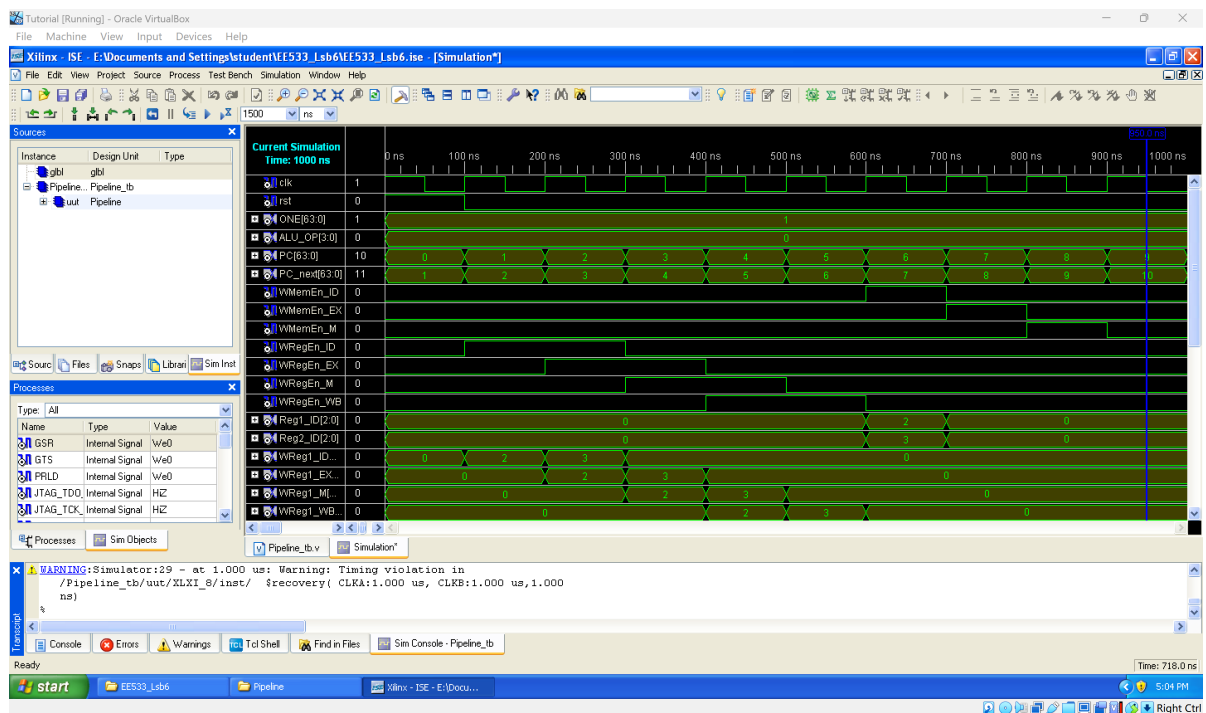
    $stop;

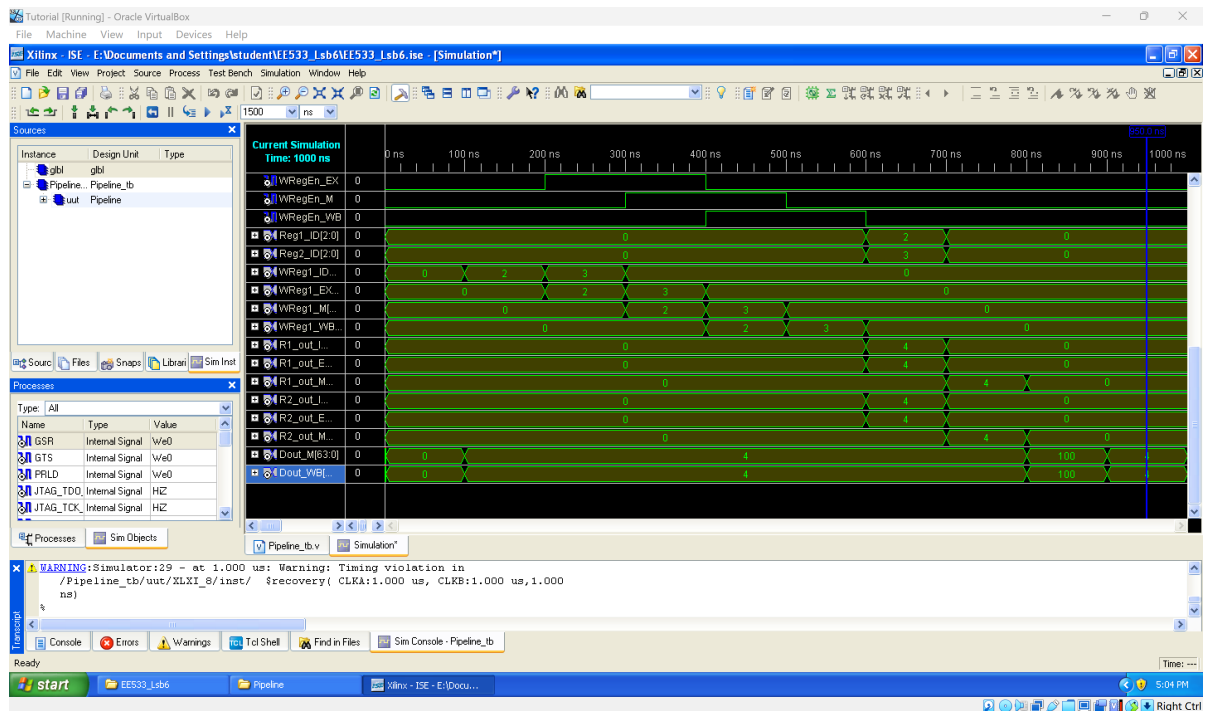
```

```
end
```

```
endmodule
```

- Screenshot





4. Integrating the Pipeline into NetFPGA

4.1 passthrough

- Verilog

```
`timescale 1ns/1ps

`define UDP_REG_ADDR_WIDTH 16
`define CPC1_NF2_DATA_WIDTH 16
`define IDS_BLOCK_TAG 1
`define IDS_REG_ADDR_WIDTH 16

module passthrough
#(
    parameter DATA_WIDTH = 64,
    parameter CTRL_WIDTH = DATA_WIDTH/8,
    parameter UDP_REG_SRC_WIDTH = 2
)
(
    out_data,
    out_ctrl,
    out_wr,
    out_rdy,
    in_data,
    in_ctrl,
    in_wr,
    in_rdy,
    reg_req_in,
    reg_ack_in,
    reg_rd_wr_L_in,
    reg_addr_in,
    reg_data_in,
    reg_src_in,
    reg_req_out,
```

```

        reg_ack_out,
        reg_rd_wr_L_out,
        reg_addr_out,
        reg_data_out,
        reg_src_out,
        clk,
        reset
    );
    (
        input  [DATA_WIDTH-1:0]      in_data,
        input  [CTRL_WIDTH-1:0]     in_ctrl,
        input                               in_wr,
        output                               in_rdy,

        output [DATA_WIDTH-1:0]      out_data,
        output [CTRL_WIDTH-1:0]     out_ctrl,
        output                               out_wr,
        input                               out_rdy,

        // --- Register interface
        input                               reg_req_in,
        input                               reg_ack_in,
        input                               reg_rd_wr_L_in,
        input  [`UDP_REG_ADDR_WIDTH-1:0]  reg_addr_in,
        input  [`CPCI_NF2_DATA_WIDTH-1:0]  reg_data_in,
        input  [UDP_REG_SRC_WIDTH-1:0]     reg_src_in,

        output                               reg_req_out,
        output                               reg_ack_out,
        output                               reg_rd_wr_L_out,
        output  [`UDP_REG_ADDR_WIDTH-1:0]  reg_addr_out,
        output  [`CPCI_NF2_DATA_WIDTH-1:0]  reg_data_out,
        output  [UDP_REG_SRC_WIDTH-1:0]     reg_src_out,

        // misc
        input                               reset,
        input                               clk
    );

```

```

assign out_data = in_data;
assign out_ctrl = in_ctrl;
assign out_wr = in_wr;
assign out_rdy = in_rdy;

assign reg_req_out = reg_req_in;
assign reg_ack_out = reg_ack_in;
assign reg_rd_wr_L_out = reg_rd_wr_L_in;
assign reg_addr_out = reg_addr_in;
assign reg_data_out = reg_data_in;
assign reg_src_out = reg_src_in;

```

Pipeline pipeline

```

(
    .ALU_OP      (),
    .clk         (),
    .ONE         (),

```

```

        .rst                0,
        .Dout_M              0,
        .Dout_WB             0,
        .PC                  0,
        .PC_next             0,
        .Reg1_ID             0,
        .Reg2_ID             0,
        .R1_out_EX           0,
        .R1_out_ID           0,
        .R1_out_M            0,
        .R2_out_EX           0,
        .R2_out_ID           0,
        .R2_out_M            0,
        .WMemEn_EX           0,
        .WMemEn_ID           0,
        .WMemEn_M            0,
        .WRegEn_EX           0,
        .WRegEn_ID           0,
        .WRegEn_M            0,
        .WRegEn_WB           0,
        .WReg1_EX            0,
        .WReg1_ID            0,
        .WReg1_M             0,
        .WReg1_WB            0
    );

```

```
endmodule
```

4.2 user_data_path

- Verilog

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// vim:set shiftwidth=3 softtabstop=3 expandtab:
// $Id: user_data_path.v 4385 2008-08-05 02:10:01Z grg $
//
// Module: user_data_path.v
// Project: NF2.1
// Description: contains all the user instantiated modules
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
`timescale 1ns/1ps

/*****
 * Even numbered ports are IO sinks/sources
 * Odd numbered ports are CPU ports corresponding to
 * IO sinks/sources to provide direct access to them
 *****/
module user_data_path
    #(parameter DATA_WIDTH = 64,
      parameter CTRL_WIDTH=DATA_WIDTH/8,
      parameter UDP_REG_SRC_WIDTH = 2,
      parameter NUM_OUTPUT_QUEUES = 8,
      parameter NUM_INPUT_QUEUES = 8,
      parameter SRAM_DATA_WIDTH = DATA_WIDTH+CTRL_WIDTH,

```

```

parameter SRAM_ADDR_WIDTH = 19)

(
  input  [DATA_WIDTH-1:0]      in_data_0,
  input  [CTRL_WIDTH-1:0]     in_ctrl_0,
  input                                in_wr_0,
  output                                in_rdy_0,

  input  [DATA_WIDTH-1:0]      in_data_1,
  input  [CTRL_WIDTH-1:0]     in_ctrl_1,
  input                                in_wr_1,
  output                                in_rdy_1,

  input  [DATA_WIDTH-1:0]      in_data_2,
  input  [CTRL_WIDTH-1:0]     in_ctrl_2,
  input                                in_wr_2,
  output                                in_rdy_2,

  input  [DATA_WIDTH-1:0]      in_data_3,
  input  [CTRL_WIDTH-1:0]     in_ctrl_3,
  input                                in_wr_3,
  output                                in_rdy_3,

  input  [DATA_WIDTH-1:0]      in_data_4,
  input  [CTRL_WIDTH-1:0]     in_ctrl_4,
  input                                in_wr_4,
  output                                in_rdy_4,

  input  [DATA_WIDTH-1:0]      in_data_5,
  input  [CTRL_WIDTH-1:0]     in_ctrl_5,
  input                                in_wr_5,
  output                                in_rdy_5,

  input  [DATA_WIDTH-1:0]      in_data_6,
  input  [CTRL_WIDTH-1:0]     in_ctrl_6,
  input                                in_wr_6,
  output                                in_rdy_6,

  input  [DATA_WIDTH-1:0]      in_data_7,
  input  [CTRL_WIDTH-1:0]     in_ctrl_7,
  input                                in_wr_7,
  output                                in_rdy_7,

  /**** not used
  // --- Interface to SATA
  input  [DATA_WIDTH-1:0]      in_data_5,
  input  [CTRL_WIDTH-1:0]     in_ctrl_5,
  input                                in_wr_5,
  output                                in_rdy_5,

  // --- Interface to the loopback queue
  input  [DATA_WIDTH-1:0]      in_data_6,
  input  [CTRL_WIDTH-1:0]     in_ctrl_6,
  input                                in_wr_6,
  output                                in_rdy_6,

```

```

// --- Interface to a user queue
input  [DATA_WIDTH-1:0]      in_data_7,
input  [CTRL_WIDTH-1:0]     in_ctrl_7,
input                                     in_wr_7,
output                                     in_rdy_7,
*****/

output [DATA_WIDTH-1:0]      out_data_0,
output [CTRL_WIDTH-1:0]     out_ctrl_0,
output                                     out_wr_0,
input                                     out_rdy_0,

output [DATA_WIDTH-1:0]      out_data_1,
output [CTRL_WIDTH-1:0]     out_ctrl_1,
output                                     out_wr_1,
input                                     out_rdy_1,

output [DATA_WIDTH-1:0]      out_data_2,
output [CTRL_WIDTH-1:0]     out_ctrl_2,
output                                     out_wr_2,
input                                     out_rdy_2,

output [DATA_WIDTH-1:0]      out_data_3,
output [CTRL_WIDTH-1:0]     out_ctrl_3,
output                                     out_wr_3,
input                                     out_rdy_3,

output [DATA_WIDTH-1:0]      out_data_4,
output [CTRL_WIDTH-1:0]     out_ctrl_4,
output                                     out_wr_4,
input                                     out_rdy_4,

output [DATA_WIDTH-1:0]      out_data_5,
output [CTRL_WIDTH-1:0]     out_ctrl_5,
output                                     out_wr_5,
input                                     out_rdy_5,

output [DATA_WIDTH-1:0]      out_data_6,
output [CTRL_WIDTH-1:0]     out_ctrl_6,
output                                     out_wr_6,
input                                     out_rdy_6,

output [DATA_WIDTH-1:0]      out_data_7,
output [CTRL_WIDTH-1:0]     out_ctrl_7,
output                                     out_wr_7,
input                                     out_rdy_7,

/**** not used
// --- Interface to SATA
output [DATA_WIDTH-1:0]      out_data_5,
output [CTRL_WIDTH-1:0]     out_ctrl_5,
output                                     out_wr_5,
input                                     out_rdy_5,

// --- Interface to the loopback queue
output [DATA_WIDTH-1:0]      out_data_6,

```

```

output [CTRL_WIDTH-1:0]      out_ctrl_6,
output                      out_wr_6,
input                      out_rdy_6,

// --- Interface to a user queue
output [DATA_WIDTH-1:0]      out_data_7,
output [CTRL_WIDTH-1:0]      out_ctrl_7,
output                      out_wr_7,
input                      out_rdy_7,
*****/

// interface to SRAM
output [SRAM_ADDR_WIDTH-1:0] wr_0_addr,
output                      wr_0_req,
input                      wr_0_ack,
output [SRAM_DATA_WIDTH-1:0] wr_0_data,

input                      rd_0_ack,
input [SRAM_DATA_WIDTH-1:0] rd_0_data,
input                      rd_0_vld,
output [SRAM_ADDR_WIDTH-1:0] rd_0_addr,
output                      rd_0_req,

// interface to DRAM
/* TBD */

// register interface
input                      reg_req,
output                      reg_ack,
input                      reg_rd_wr_L,
input [`UDP_REG_ADDR_WIDTH-1:0] reg_addr,
output [`CPCI_NF2_DATA_WIDTH-1:0] reg_rd_data,
input [`CPCI_NF2_DATA_WIDTH-1:0] reg_wr_data,

// misc
input                      reset,
input                      clk);

function integer log2;
    input integer number;
    begin
        log2=0;
        while(2**log2<number) begin
            log2=log2+1;
        end
    end
endfunction // log2

//----- Internal parameters -----

localparam NUM_IQ_BITS = log2(NUM_INPUT_QUEUES);

localparam IN_ARB_STAGE_NUM = 2;
localparam OP_LUT_STAGE_NUM = 4;
localparam OQ_STAGE_NUM     = 6;

```

```

//----- Input arbiter wires/regs -----
wire                                in_arb_in_reg_req;
wire                                in_arb_in_reg_ack;
wire                                in_arb_in_reg_rd_wr_L;
wire [`UDP_REG_ADDR_WIDTH-1:0]    in_arb_in_reg_addr;
wire [`CPCI_NF2_DATA_WIDTH-1:0]    in_arb_in_reg_data;
wire [UDP_REG_SRC_WIDTH-1:0]       in_arb_in_reg_src;

//----- output port lut wires/regs -----
wire [CTRL_WIDTH-1:0]              op_lut_in_ctrl;
wire [DATA_WIDTH-1:0]              op_lut_in_data;
wire                                op_lut_in_wr;
wire                                op_lut_in_rdy;

wire                                op_lut_in_reg_req;
wire                                op_lut_in_reg_ack;
wire                                op_lut_in_reg_rd_wr_L;
wire [`UDP_REG_ADDR_WIDTH-1:0]    op_lut_in_reg_addr;
wire [`CPCI_NF2_DATA_WIDTH-1:0]    op_lut_in_reg_data;
wire [UDP_REG_SRC_WIDTH-1:0]       op_lut_in_reg_src;

//----- passthrough wires/regs -----
wire [CTRL_WIDTH-1:0]              passthrough_in_ctrl;
wire [DATA_WIDTH-1:0]              passthrough_in_data;
wire                                passthrough_in_wr;
wire                                passthrough_in_rdy;

wire                                passthrough_in_reg_req;
wire                                passthrough_in_reg_ack;
wire                                passthrough_in_reg_rd_wr_L;
wire [`UDP_REG_ADDR_WIDTH-1:0]    passthrough_in_reg_addr;
wire [`CPCI_NF2_DATA_WIDTH-1:0]    passthrough_in_reg_data;
wire [UDP_REG_SRC_WIDTH-1:0]       passthrough_in_reg_src;

//----- output queues wires/regs -----
wire [CTRL_WIDTH-1:0]              oq_in_ctrl;
wire [DATA_WIDTH-1:0]              oq_in_data;
wire                                oq_in_wr;
wire                                oq_in_rdy;

wire                                oq_in_reg_req;
wire                                oq_in_reg_ack;
wire                                oq_in_reg_rd_wr_L;
wire [`UDP_REG_ADDR_WIDTH-1:0]    oq_in_reg_addr;
wire [`CPCI_NF2_DATA_WIDTH-1:0]    oq_in_reg_data;
wire [UDP_REG_SRC_WIDTH-1:0]       oq_in_reg_src;

//----- UDP register master wires/regs -----
wire                                udp_reg_req_in;
wire                                udp_reg_ack_in;
wire                                udp_reg_rd_wr_L_in;
wire [`UDP_REG_ADDR_WIDTH-1:0]    udp_reg_addr_in;
wire [`CPCI_NF2_DATA_WIDTH-1:0]    udp_reg_data_in;
wire [UDP_REG_SRC_WIDTH-1:0]       udp_reg_src_in;

```



```
//----- Connect the data path -----
```

```
input_arbiter
```

```
#(.DATA_WIDTH(DATA_WIDTH),  
  .CTRL_WIDTH(CTRL_WIDTH),  
  .UDP_REG_SRC_WIDTH (UDP_REG_SRC_WIDTH),  
  .STAGE_NUMBER(IN_ARB_STAGE_NUM))
```

```
input_arbiter
```

```
(  
  .out_data      (op_lut_in_data),  
  .out_ctrl      (op_lut_in_ctrl),  
  .out_wr        (op_lut_in_wr),  
  .out_rdy       (op_lut_in_rdy),
```

```
  // --- Interface to the input queues
```

```
  .in_data_0     (in_data_0),  
  .in_ctrl_0     (in_ctrl_0),  
  .in_wr_0       (in_wr_0),  
  .in_rdy_0      (in_rdy_0),
```

```
  .in_data_1     (in_data_1),  
  .in_ctrl_1     (in_ctrl_1),  
  .in_wr_1       (in_wr_1),  
  .in_rdy_1      (in_rdy_1),
```

```
  .in_data_2     (in_data_2),  
  .in_ctrl_2     (in_ctrl_2),  
  .in_wr_2       (in_wr_2),  
  .in_rdy_2      (in_rdy_2),
```

```
  .in_data_3     (in_data_3),  
  .in_ctrl_3     (in_ctrl_3),  
  .in_wr_3       (in_wr_3),  
  .in_rdy_3      (in_rdy_3),
```

```
  .in_data_4     (in_data_4),  
  .in_ctrl_4     (in_ctrl_4),  
  .in_wr_4       (in_wr_4),  
  .in_rdy_4      (in_rdy_4),
```

```
  .in_data_5     (in_data_5),  
  .in_ctrl_5     (in_ctrl_5),  
  .in_wr_5       (in_wr_5),  
  .in_rdy_5      (in_rdy_5),
```

```
  .in_data_6     (in_data_6),  
  .in_ctrl_6     (in_ctrl_6),  
  .in_wr_6       (in_wr_6),  
  .in_rdy_6      (in_rdy_6),
```

```
  .in_data_7     (in_data_7),  
  .in_ctrl_7     (in_ctrl_7),  
  .in_wr_7       (in_wr_7),  
  .in_rdy_7      (in_rdy_7),
```

```

// --- Register interface
.reg_req_in      (in_arb_in_reg_req),
.reg_ack_in      (in_arb_in_reg_ack),
.reg_rd_wr_L_in  (in_arb_in_reg_rd_wr_L),
.reg_addr_in     (in_arb_in_reg_addr),
.reg_data_in     (in_arb_in_reg_data),
.reg_src_in      (in_arb_in_reg_src),

.reg_req_out     (op_lut_in_reg_req),
.reg_ack_out     (op_lut_in_reg_ack),
.reg_rd_wr_L_out (op_lut_in_reg_rd_wr_L),
.reg_addr_out    (op_lut_in_reg_addr),
.reg_data_out    (op_lut_in_reg_data),
.reg_src_out     (op_lut_in_reg_src),

// --- Misc
.reset          (reset),
.clk            (clk)
);

```

output_port_lookup

```

#(.DATA_WIDTH(DATA_WIDTH),
  .CTRL_WIDTH(CTRL_WIDTH),
  .UDP_REG_SRC_WIDTH (UDP_REG_SRC_WIDTH),
  .INPUT_ARBITER_STAGE_NUM(IN_ARB_STAGE_NUM),
  .STAGE_NUM(OP_LUT_STAGE_NUM),
  .NUM_OUTPUT_QUEUES(NUM_OUTPUT_QUEUES),
  .NUM_IQ_BITS(NUM_IQ_BITS))

```

output_port_lookup

```

(.out_data      (passthrough_in_data),
 .out_ctrl      (passthrough_in_ctrl),
 .out_wr        (passthrough_in_wr),
 .out_rdy       (passthrough_in_rdy),

```

// --- Interface to the rx input queues

```

.in_data      (op_lut_in_data),
.in_ctrl      (op_lut_in_ctrl),
.in_wr        (op_lut_in_wr),
.in_rdy       (op_lut_in_rdy),

```

// --- Register interface

```

.reg_req_in      (op_lut_in_reg_req),
.reg_ack_in      (op_lut_in_reg_ack),
.reg_rd_wr_L_in  (op_lut_in_reg_rd_wr_L),
.reg_addr_in     (op_lut_in_reg_addr),
.reg_data_in     (op_lut_in_reg_data),
.reg_src_in      (op_lut_in_reg_src),

.reg_req_out     (passthrough_in_reg_req),
.reg_ack_out     (passthrough_in_reg_ack),
.reg_rd_wr_L_out (passthrough_in_reg_rd_wr_L),
.reg_addr_out    (passthrough_in_reg_addr),
.reg_data_out    (passthrough_in_reg_data),
.reg_src_out     (passthrough_in_reg_src),

```

// --- Misc

```

.clk                (clk),
.reset              (reset));

passthrough
#(.DATA_WIDTH(DATA_WIDTH),
  .CTRL_WIDTH(CTRL_WIDTH),
  .UDP_REG_SRC_WIDTH (UDP_REG_SRC_WIDTH),
  .OP_LUT_STAGE_NUM(OP_LUT_STAGE_NUM),
  .NUM_OUTPUT_QUEUES(NUM_OUTPUT_QUEUES),
  .STAGE_NUM(OQ_STAGE_NUM),
  .SRAM_ADDR_WIDTH(SRAM_ADDR_WIDTH))
passthrough
(.out_data          (oq_in_data),
 .out_ctrl          (oq_in_ctrl),
 .out_wr            (oq_in_wr),
 .out_rdy           (oq_in_rdy),

// --- Interface to the rx input queues
.in_data            (passthrough_in_data),
.in_ctrl            (passthrough_in_ctrl),
.in_wr              (passthrough_in_wr),
.in_rdy             (passthrough_in_rdy),

// --- Register interface
.reg_req_in         (passthrough_in_reg_req),
.reg_ack_in         (passthrough_in_reg_ack),
.reg_rd_wr_L_in     (passthrough_in_reg_rd_wr_L),
.reg_addr_in        (passthrough_in_reg_addr),
.reg_data_in        (passthrough_in_reg_data),
.reg_src_in         (passthrough_in_reg_src),

.reg_req_out        (oq_in_reg_req),
.reg_ack_out        (oq_in_reg_ack),
.reg_rd_wr_L_out    (oq_in_reg_rd_wr_L),
.reg_addr_out       (oq_in_reg_addr),
.reg_data_out       (oq_in_reg_data),
.reg_src_out        (oq_in_reg_src),

// --- Misc
.clk                (clk),
.reset              (reset));

ids #(
  .DATA_WIDTH(DATA_WIDTH),
  .CTRL_WIDTH(CTRL_WIDTH),
  .UDP_REG_SRC_WIDTH (UDP_REG_SRC_WIDTH)
// .INPUT_ARBITER_STAGE_NUM(IN_ARB_STAGE_NUM),
// .NUM_OUTPUT_QUEUES(NUM_OUTPUT_QUEUES),
// .NUM_IQ_BITS(NUM_IQ_BITS)
) ids (
// --- data path interface
.out_data          (oq_in_data),
.out_ctrl          (oq_in_ctrl),
.out_wr            (oq_in_wr),
.out_rdy           (oq_in_rdy),

```

```

.in_data                (passthrough_in_data),
.in_ctrl                (passthrough_in_ctrl),
.in_wr                 (passthrough_in_wr),
.in_rdy                 (passthrough_in_rdy),

// --- Register interface
.reg_req_in            (passthrough_in_reg_req),
.reg_ack_in            (passthrough_in_reg_ack),
.reg_rd_wr_L_in        (passthrough_in_reg_rd_wr_L),
.reg_addr_in           (passthrough_in_reg_addr),
.reg_data_in           (passthrough_in_reg_data),
.reg_src_in            (passthrough_in_reg_src),

.reg_req_out           (oq_in_reg_req),
.reg_ack_out           (oq_in_reg_ack),
.reg_rd_wr_L_out       (oq_in_reg_rd_wr_L),
.reg_addr_out          (oq_in_reg_addr),
.reg_data_out          (oq_in_reg_data),
.reg_src_out           (oq_in_reg_src),

// --- Misc
.clk                   (clk),
.reset                 (reset)
);

```

output_queues

```

#(.DATA_WIDTH(DATA_WIDTH),
  .CTRL_WIDTH(CTRL_WIDTH),
  .UDP_REG_SRC_WIDTH (UDP_REG_SRC_WIDTH),
  .OP_LUT_STAGE_NUM(OP_LUT_STAGE_NUM),
  .NUM_OUTPUT_QUEUES(NUM_OUTPUT_QUEUES),
  .STAGE_NUM(OQ_STAGE_NUM),
  .SRAM_ADDR_WIDTH(SRAM_ADDR_WIDTH))

```

output_queues

```

(// --- data path interface

```

```

.out_data_0            (out_data_0),
.out_ctrl_0            (out_ctrl_0),
.out_wr_0              (out_wr_0),
.out_rdy_0             (out_rdy_0),

.out_data_1            (out_data_1),
.out_ctrl_1            (out_ctrl_1),
.out_wr_1              (out_wr_1),
.out_rdy_1             (out_rdy_1),

.out_data_2            (out_data_2),
.out_ctrl_2            (out_ctrl_2),
.out_wr_2              (out_wr_2),
.out_rdy_2             (out_rdy_2),

.out_data_3            (out_data_3),
.out_ctrl_3            (out_ctrl_3),
.out_wr_3              (out_wr_3),
.out_rdy_3             (out_rdy_3),

```

```

.out_data_4      (out_data_4),
.out_ctrl_4      (out_ctrl_4),
.out_wr_4        (out_wr_4),
.out_rdy_4       (out_rdy_4),

.out_data_5      (out_data_5),
.out_ctrl_5      (out_ctrl_5),
.out_wr_5        (out_wr_5),
.out_rdy_5       (out_rdy_5),

.out_data_6      (out_data_6),
.out_ctrl_6      (out_ctrl_6),
.out_wr_6        (out_wr_6),
.out_rdy_6       (out_rdy_6),

.out_data_7      (out_data_7),
.out_ctrl_7      (out_ctrl_7),
.out_wr_7        (out_wr_7),
.out_rdy_7       (out_rdy_7),

    // --- Interface to the previous module
.in_data         (oq_in_data),
.in_ctrl         (oq_in_ctrl),
.in_rdy          (oq_in_rdy),
.in_wr           (oq_in_wr),

    // --- Register interface
.reg_req_in      (oq_in_reg_req),
.reg_ack_in      (oq_in_reg_ack),
.reg_rd_wr_L_in  (oq_in_reg_rd_wr_L),
.reg_addr_in     (oq_in_reg_addr),
.reg_data_in     (oq_in_reg_data),
.reg_src_in      (oq_in_reg_src),

.reg_req_out     (udp_reg_req_in),
.reg_ack_out     (udp_reg_ack_in),
.reg_rd_wr_L_out (udp_reg_rd_wr_L_in),
.reg_addr_out    (udp_reg_addr_in),
.reg_data_out    (udp_reg_data_in),
.reg_src_out     (udp_reg_src_in),

    // --- SRAM sm interface
.wr_0_addr       (wr_0_addr),
.wr_0_req        (wr_0_req),
.wr_0_ack        (wr_0_ack),
.wr_0_data       (wr_0_data),
.rd_0_ack        (rd_0_ack),
.rd_0_data       (rd_0_data),
.rd_0_vld        (rd_0_vld),
.rd_0_addr       (rd_0_addr),
.rd_0_req        (rd_0_req),

    // --- Misc
.clk              (clk),
.reset            (reset));

```

```

//-----
//
// --- User data path register master
//
//     Takes the register accesses from core,
//     sends them around the User Data Path module
//     ring and then returns the replies back
//     to the core
//
//-----

udp_reg_master #(
    .UDP_REG_SRC_WIDTH (UDP_REG_SRC_WIDTH)
) udp_reg_master (
    // Core register interface signals
    .core_reg_req          (reg_req),
    .core_reg_ack          (reg_ack),
    .core_reg_rd_wr_L      (reg_rd_wr_L),

    .core_reg_addr         (reg_addr),

    .core_reg_rd_data      (reg_rd_data),
    .core_reg_wr_data      (reg_wr_data),

    // UDP register interface signals (output)
    .reg_req_out           (in_arb_in_reg_req),
    .reg_ack_out           (in_arb_in_reg_ack),
    .reg_rd_wr_L_out       (in_arb_in_reg_rd_wr_L),

    .reg_addr_out          (in_arb_in_reg_addr),
    .reg_data_out          (in_arb_in_reg_data),

    .reg_src_out           (in_arb_in_reg_src),

    // UDP register interface signals (input)
    .reg_req_in            (udp_reg_req_in),
    .reg_ack_in            (udp_reg_ack_in),
    .reg_rd_wr_L_in        (udp_reg_rd_wr_L_in),

    .reg_addr_in           (udp_reg_addr_in),
    .reg_data_in           (udp_reg_data_in),

    .reg_src_in            (udp_reg_src_in),

    //
    .clk                   (clk),
    .reset                 (reset)
);

```

```
endmodule // user_data_path
```

5. Extra: Link to the GitHub Repo

- https://github.com/yuezhenglingluan/USC_EE533_lab6

