**Digital Design and Computer Organization Laboratory**

**3rd Semester, Academic Year 2024**

Date: 17/10/2024

| | | |
|---|---|---|
| Name: Rithvik Rajesh Matta | SRN: PES2UG23CS485 | Section: H |

Week#____9_____ Program Number: ____9__

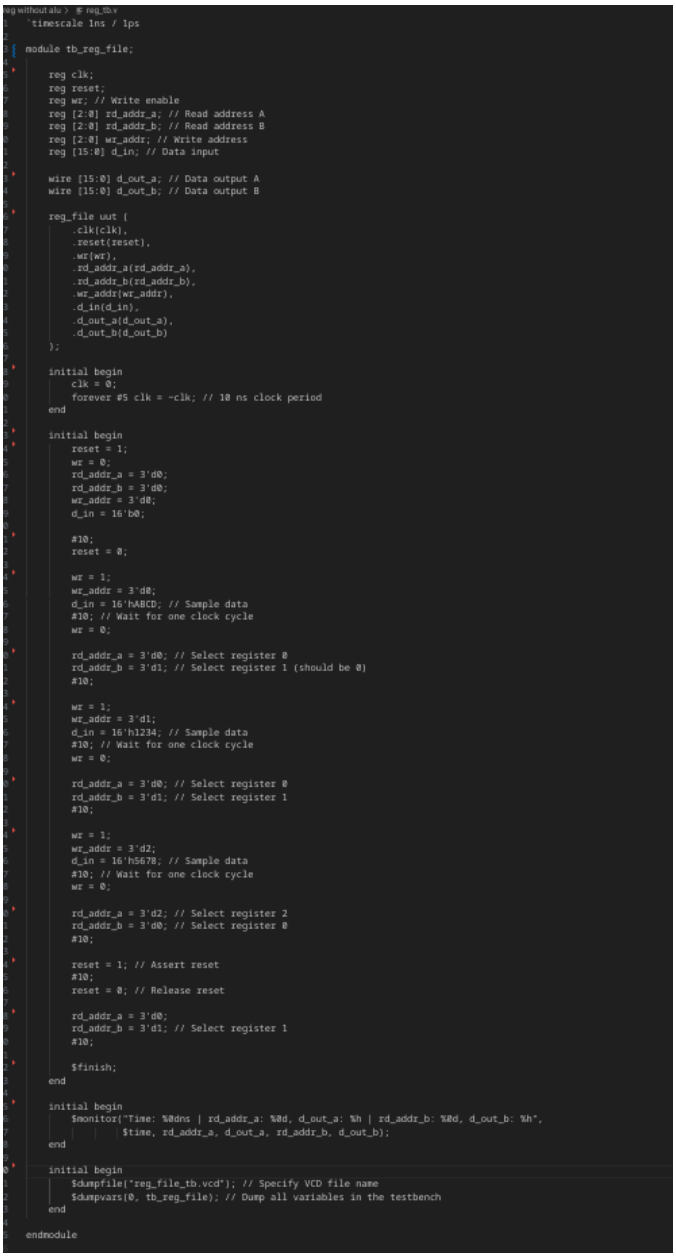
Aim of the Experiment:

CONSTRUCT A REGISTER FILE (without alu), FROM WHICH TWO 16-BIT VALUES CAN BE READ, AND TO WHICH ONE 16-BIT VALUE WRITTEN, EVERY CLOCK CYCLE.GENERATE THE VVP OUTPUT AND SIMULATION WAVEFORM USING GTKWAVE. VERIFY THE OUTPUT AND WAVEFORM WITH THE TRUTH TABLE

I. Verilog Code Screenshot

wk reg without alu > reg.v

```v
module dfrl_16 (
    input wire clk,
    input wire reset,
    input wire load,
    input wire [15:0] in,
    output reg [15:0] out
);
    always @(posedge clk or posedge reset) begin
        if (reset) begin
            out <= 16'b0; // Reset the output to zero
        end else if (load) begin
            out <= in; // Load input to output
        end
    end
endmodule

module reg_file (
    input wire clk,
    input wire reset,
    input wire wr, // Write enable
    input wire [2:0] rd_addr_a, // Read address A
    input wire [2:0] rd_addr_b, // Read address B
    input wire [2:0] wr_addr, // Write address
    input wire [15:0] d_in, // Data input
    output wire [15:0] d_out_a, // Data output A
    output wire [15:0] d_out_b // Data output B
);
    wire [15:0] dout[7:0]; // Internal outputs for 8 registers
    wire [7:0] load; // Load signals for each register

    dfrl_16 dfrl_16_0(clk, reset, load[0], d_in, dout[0]);
    dfrl_16 dfrl_16_1(clk, reset, load[1], d_in, dout[1]);
    dfrl_16 dfrl_16_2(clk, reset, load[2], d_in, dout[2]);
    dfrl_16 dfrl_16_3(clk, reset, load[3], d_in, dout[3]);
    dfrl_16 dfrl_16_4(clk, reset, load[4], d_in, dout[4]);
    dfrl_16 dfrl_16_5(clk, reset, load[5], d_in, dout[5]);
    dfrl_16 dfrl_16_6(clk, reset, load[6], d_in, dout[6]);
    dfrl_16 dfrl_16_7(clk, reset, load[7], d_in, dout[7]);

    assign load = {8{wr}} & (1 << wr_addr);

    assign d_out_a = (rd_addr_a == 3'd0) ? dout[0] :
                     (rd_addr_a == 3'd1) ? dout[1] :
                     (rd_addr_a == 3'd2) ? dout[2] :
                     (rd_addr_a == 3'd3) ? dout[3] :
                     (rd_addr_a == 3'd4) ? dout[4] :
                     (rd_addr_a == 3'd5) ? dout[5] :
                     (rd_addr_a == 3'd6) ? dout[6] :
                     (rd_addr_a == 3'd7) ? dout[7] : 16'b0;

    assign d_out_b = (rd_addr_b == 3'd0) ? dout[0] :
                     (rd_addr_b == 3'd1) ? dout[1] :
                     (rd_addr_b == 3'd2) ? dout[2] :
                     (rd_addr_b == 3'd3) ? dout[3] :
                     (rd_addr_b == 3'd4) ? dout[4] :
                     (rd_addr_b == 3'd5) ? dout[5] :
                     (rd_addr_b == 3'd6) ? dout[6] :
                     (rd_addr_b == 3'd7) ? dout[7] : 16'b0;
endmodule

```

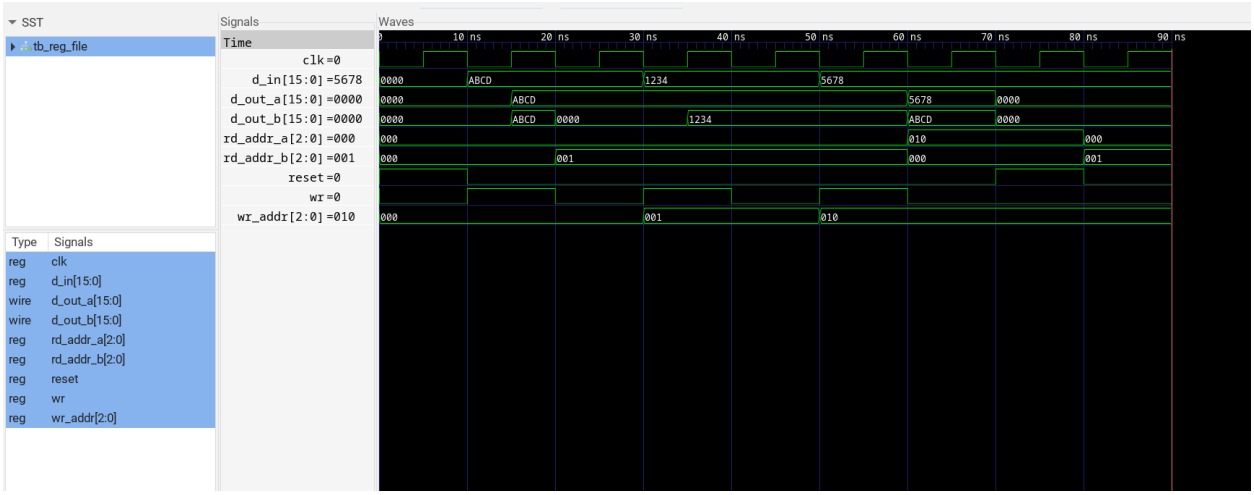


```
reg without alu > reg_tb.v
`timescale 1ns / 1ps

module tb_reg_file;

    reg clk;
    reg reset;
    reg wr; // Write enable
    reg [2:0] rd_addr_a; // Read address A
    reg [2:0] rd_addr_b; // Read address B
    reg [2:0] wr_addr; // Write address
    reg [15:0] d_in; // Data input

    wire [15:0] d_out_a; // Data output A
    wire [15:0] d_out_b; // Data output B

    reg_file uut (
        .clk(clk),
        .reset(reset),
        .wr(wr),
        .rd_addr_a(rd_addr_a),
        .rd_addr_b(rd_addr_b),
        .wr_addr(wr_addr),
        .d_in(d_in),
        .d_out_a(d_out_a),
        .d_out_b(d_out_b)
    );

    initial begin
        clk = 0;
        forever #5 clk = ~clk; // 10 ns clock period
    end

    initial begin
        reset = 1;
        wr = 0;
        rd_addr_a = 3'd0;
        rd_addr_b = 3'd0;
        wr_addr = 3'd0;
        d_in = 16'b0;

        #10;
        reset = 0;

        wr = 1;
        wr_addr = 3'd0;
        d_in = 16'hABCD; // Sample data
        #10; // Wait for one clock cycle
        wr = 0;

        rd_addr_a = 3'd0; // Select register 0
        rd_addr_b = 3'd1; // Select register 1 (should be 0)
        #10;

        wr = 1;
        wr_addr = 3'd1;
        d_in = 16'h1234; // Sample data
        #10; // Wait for one clock cycle
        wr = 0;

        rd_addr_a = 3'd0; // Select register 0
        rd_addr_b = 3'd1; // Select register 1
        #10;

        wr = 1;
        wr_addr = 3'd2;
        d_in = 16'h5678; // Sample data
        #10; // Wait for one clock cycle
        wr = 0;

        rd_addr_a = 3'd2; // Select register 2
        rd_addr_b = 3'd0; // Select register 0
        #10;

        reset = 1; // Assert reset
        #10;
        reset = 0; // Release reset

        rd_addr_a = 3'd0;
        rd_addr_b = 3'd1; // Select register 1
        #10;

        $finish;
    end

    initial begin
        $monitor("Time: %0dns | rd_addr_a: %0d, d_out_a: %h | rd_addr_b: %0d, d_out_b: %h",
                 $time, rd_addr_a, d_out_a, rd_addr_b, d_out_b);
    end

    initial begin
        $dumpfile("reg_file_tb.vcd"); // Specify VCD file name
        $dumpvars(0, tb_reg_file); // Dump all variables in the testbench
    end

endmodule
```

# II. Verilog VVP Output Screen Shot

```
rithvikmatta@penguin:~/PES/DDCO-sem-3/wk reg without alu$ iverilog -o test reg.v reg_tb.v
rithvikmatta@penguin:~/PES/DDCO-sem-3/wk reg without alu$ vvp test
VCD info: dumpfile reg_file_tb.vcd opened for output.
Time: 0ns | rd_addr_a: 0, d_out_a: 0000 | rd_addr_b: 0, d_out_b: 0000
Time: 15ns | rd_addr_a: 0, d_out_a: abcd | rd_addr_b: 0, d_out_b: abcd
Time: 20ns | rd_addr_a: 0, d_out_a: abcd | rd_addr_b: 1, d_out_b: 0000
Time: 35ns | rd_addr_a: 0, d_out_a: abcd | rd_addr_b: 1, d_out_b: 1234
Time: 60ns | rd_addr_a: 2, d_out_a: 5678 | rd_addr_b: 0, d_out_b: abcd
Time: 70ns | rd_addr_a: 2, d_out_a: 0000 | rd_addr_b: 0, d_out_b: 0000
Time: 80ns | rd_addr_a: 0, d_out_a: 0000 | rd_addr_b: 1, d_out_b: 0000
rithvikmatta@penguin:~/PES/DDCO-sem-3/wk reg without alu$ gtkwave reg_file_tb.vcd

GTKWave Analyzer v3.3.118 (w)1999-2023 BSI

[0] start time.
[90000] end time.
GTKWAVE | Touch screen detected, enabling gestures.
Exiting.
```

## III. GTKWAVE Screenshot




## IV. Output Table (Truth Table)

| rd_addr_a | d_out_a | rd_addr_b | d_out_b |
|---|---|---|---|
| 0 | 0000 | 0 | 0000 |
| 0 | abcd | 0 | abcd |
| 0 | abcd | 1 | 0000 |
| 0 | abcd | 1 | 1234 |
| 2 | 5678 | 0 | abcd |

| | | | |
|---|---|---|---|
| 2 | 0000 | 0 | 0000 |
| 0 | 0000 | 1 | 0000 |

If found plagiarized, I will abide with the disciplinary action of the University.

Signature:

Name: Rithvik Rajesh Matta

SRN: PES2UG23CS485

Section: H

Date: 17-10-2024