يوسف أحمد محمد ابراهيم

Assignment 4

Extra

Module 1 :

- **RTL code :**

```
module mod1_LFSR(input clk, rst, set, output [3:0]out);

    reg q1, q2, q3, q4;
    always @(posedge clk, posedge set, posedge rst) begin
        if (rst && set) begin
            q1 <= 1;
            q2 <= 0;
            q3 <= 0;
            q4 <= 0;
        end
        else begin
            q1 <= q4;
            q2 <= q1 ^ q4;
            q3 <= q2;
            q4 <= q3;
        end
    end
    assign out = {q4, q3, q2, q1};
endmodule
```

- **testbench code :**

ssignments > 4_extra > 1_tb.v > mod1_LFSR_tb

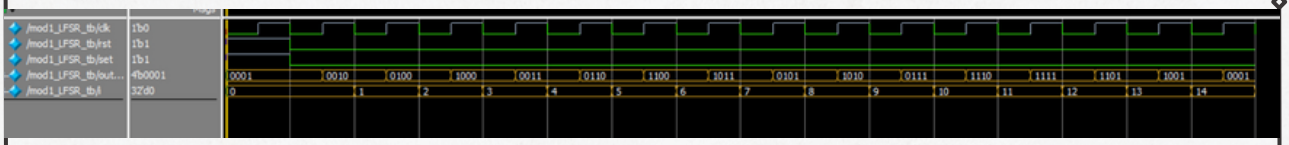
```
module mod1_LFSR_tb();
    reg clk, rst, set;
    reg [3:0]out_exp_sequence[14:0];
    wire [3:0]out_dut;

    mod1_LFSR dut(clk, rst, set, out_dut);

    initial begin
        clk = 0;
        forever
            #1 clk = ~clk;
    end

    integer i = 0;
    initial begin
        rst = 1;
        set = 1;
        out_exp_sequence[0] = 4'b0001;
        out_exp_sequence[1] = 4'b0010;
        out_exp_sequence[2] = 4'b0100;
        out_exp_sequence[3] = 4'b1000;
        out_exp_sequence[4] = 4'b0011;
        out_exp_sequence[5] = 4'b0110;
        out_exp_sequence[6] = 4'b1100;
        out_exp_sequence[7] = 4'b1011;
        out_exp_sequence[8] = 4'b0101;
        out_exp_sequence[9] = 4'b1010;
        out_exp_sequence[10] = 4'b0111;
        out_exp_sequence[11] = 4'b1110;
        out_exp_sequence[12] = 4'b1111;
        out_exp_sequence[13] = 4'b1101;
        out_exp_sequence[14] = 4'b1001;
        out_exp_sequence[15] = 4'b0001;
        @(negedge clk);
        rst = 0;
        set = 0;
        for (i = 0; i < 15; i = i + 1) begin
            if(out_dut != out_exp_sequence[i]) begin
                $display("ERROR i = %d dut = %d and exp = %d", i, out_dut, out_exp_sequence[i]);
            end
            @(negedge clk);
        end
        $stop;
    end
endmodule
```

- **waveforms (questasim)**



Module 2 :

- **RTL code :**

```
module mod2_param_adder #(
    parameter WIDTH = 4,
    parameter PIPELINE_ENABLE = 1,
    parameter USE_FULL_ADDER = 1
)(
    input  wire clk, rst, cin,
    input  wire [WIDTH-1:0] a, b,
    output wire [WIDTH-1:0] sum,
    output wire cout
);

    wire [WIDTH:0] comb_res;
    wire effective_cin = (USE_FULL_ADDER) ? cin : 1'b0;

    assign comb_res = a + b + effective_cin;

    generate
        if (PIPELINE_ENABLE) begin
            reg [WIDTH-1:0] sum_reg;
            reg cout_reg;
            always @(posedge clk) begin
                if (rst) begin
                    sum_reg  <= {WIDTH{1'b0}};
                    cout_reg <= 1'b0;
                end else begin
                    sum_reg  <= comb_res[WIDTH-1:0];
                    cout_reg <= comb_res[WIDTH];
                end
            end
            assign sum  = sum_reg;
            assign cout = cout_reg;
        end else begin
            assign sum  = comb_res[WIDTH-1:0];
            assign cout = comb_res[WIDTH];
        end
    endgenerate

endmodule
```

- **testbench code :**

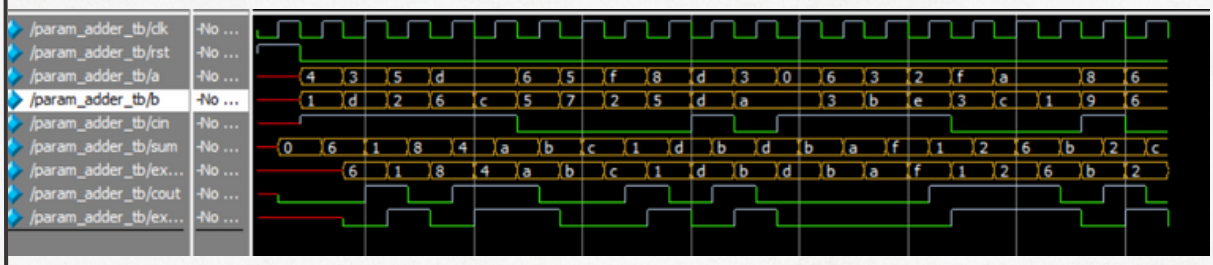
```
module param_adder_tb;
    reg         clk, rst;
    reg  [3:0]  a, b;
    reg         cin;
    wire [3:0]  sum;
    reg  [3:0]  exp_s;
    wire        cout;
    reg         exp_c;

    mod2_param_adder dut(clk, rst, cin, a, b, sum, cout);
    initial begin
        clk = 0;
        forever
            #1 clk = ~clk;
    end

    initial begin
        rst = 1;
        @(negedge clk);
        rst = 0;
        repeat (20) begin
            a = $random;
            b = $random;
            cin = $random;
            @(negedge clk);

            {exp_c, exp_s} = a + b + cin;
            if ({cout, sum} !== {exp_c, exp_s}) begin
                $display("ERROR");
                $stop;
            end
        end
        $stop;
    end
endmodule
```

- **waveforms (questasim)**



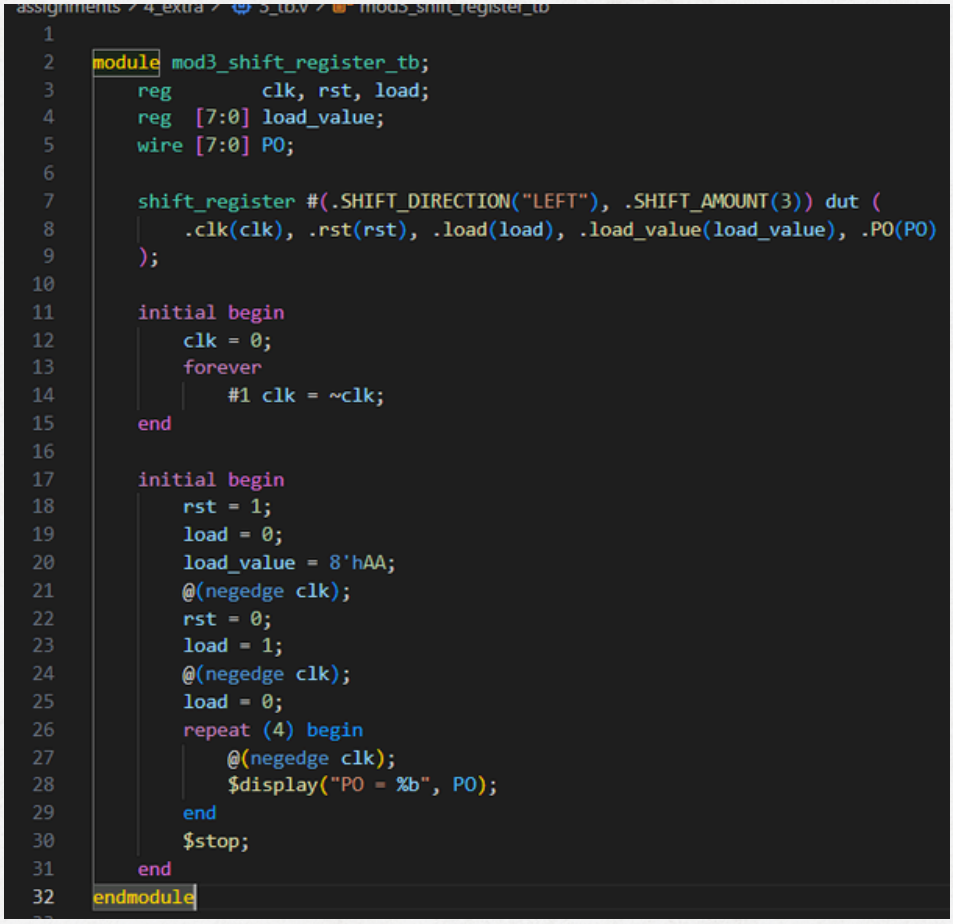
Module 3 :

- **RTL code :**

```
module mod3_shift_register #(
    parameter SHIFT_DIRECTION = "LEFT",
    parameter SHIFT_AMOUNT    = 1
)(
    input  wire        clk,
    input  wire        rst,
    input  wire        load,
    input  wire [7:0]  load_value,
    output reg  [7:0]  PO
);
    always @(posedge clk or posedge rst) begin
        if (rst) begin
            PO <= 8'b0;
        end else if (load) begin
            PO <= load_value;
        end else begin
            if (SHIFT_DIRECTION == "LEFT") begin
                PO <= PO << SHIFT_AMOUNT;
            end else begin
                PO <= PO >> SHIFT_AMOUNT;
            end
        end
    end
endmodule

```

- **testbench code :**



```
assignments > 4_extra > 5_tb.v > mod3_shift_register_tb

module mod3_shift_register_tb;
    reg        clk, rst, load;
    reg  [7:0] load_value;
    wire [7:0] PO;

    shift_register #(.SHIFT_DIRECTION("LEFT"), .SHIFT_AMOUNT(3)) dut (
        .clk(clk), .rst(rst), .load(load), .load_value(load_value), .PO(PO)
    );

    initial begin
        clk = 0;
        forever
            #1 clk = ~clk;
    end

    initial begin
        rst = 1;
        load = 0;
        load_value = 8'hAA;
        @(negedge clk);
        rst = 0;
        load = 1;
        @(negedge clk);
        load = 0;
        repeat (4) begin
            @(negedge clk);
            $display("PO = %b", PO);
        end
        $stop;
    end
endmodule
```

- **waveforms (questasim)**