CS223-003 (DIGITAL DESIGN)

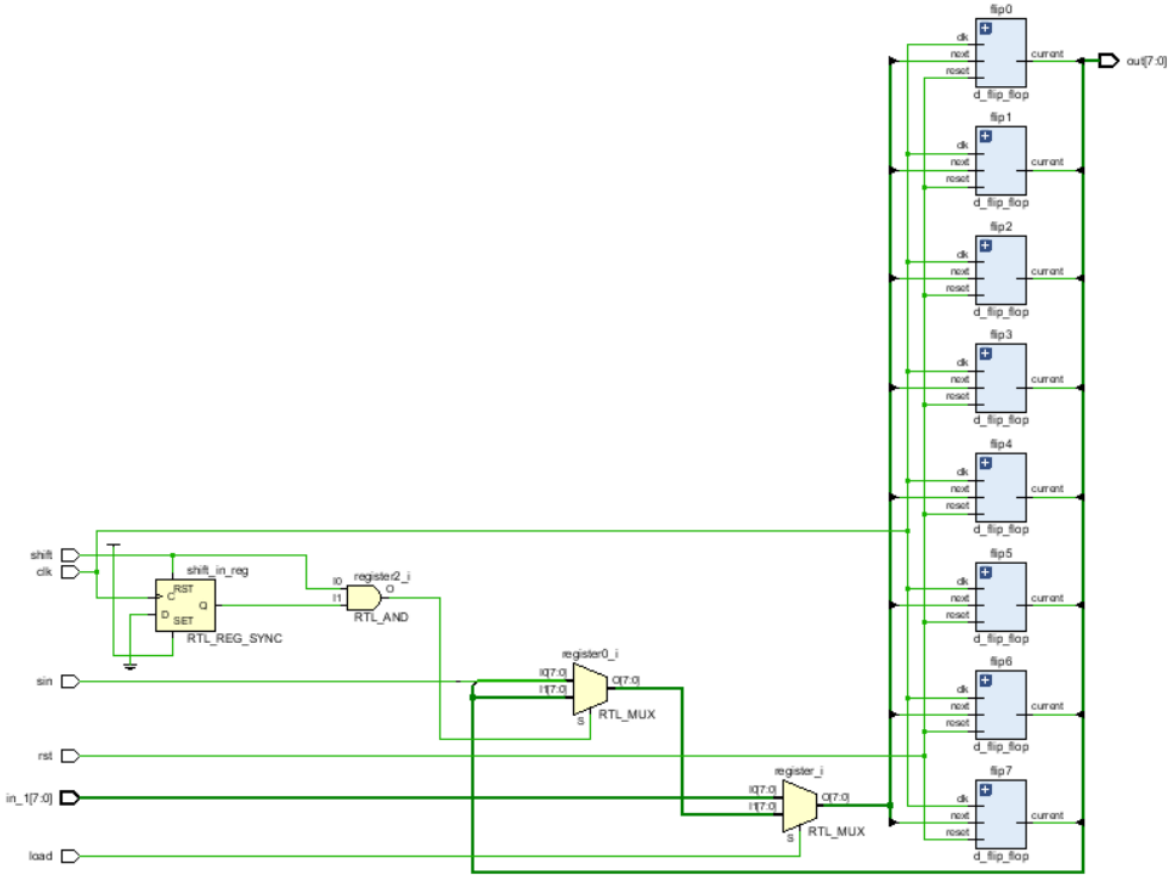
LAB 5

NAME: SERTAÇ DERYA

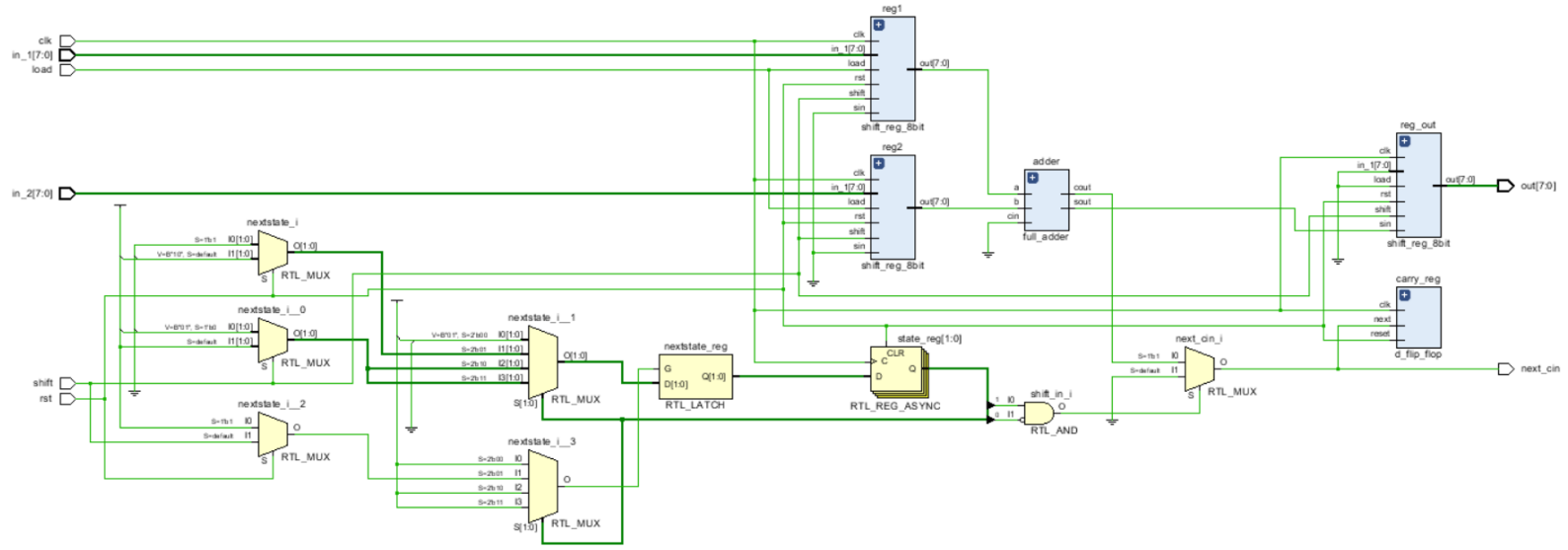
ID: 22003208

DATE: 5.12.2022

Circuit Schematic for Shift Register



Circuit Schematic for Shift Register



SystemVerilog Module for D Flip Flop

```
module d_flip_flop(
    output logic current,
    input logic next,
    input logic clk,
    input logic reset
    );
    always_ff @(posedge clk)
        if(reset) current <= 0;
        else current <= next;
endmodule
```

SystemVerilog Module for Shift Register

```
module shift_reg_8bit(
    input logic [7:0]in_1,
    input logic load,
    input logic clk,
    input logic shift,
    input logic rst,
    input logic sin,
    output logic [7:0]out
    );
    logic [7:0]register;
    logic shift_in;
    assign register = load ? in_1 : ((shift & shift_in) ? {sin, out[7:1]} :
out);
    d_flip_flop flip0(out[0], register[0], clk, rst);
    d_flip_flop flip1(out[1], register[1], clk, rst);
    d_flip_flop flip2(out[2], register[2], clk, rst);
    d_flip_flop flip3(out[3], register[3], clk, rst);
    d_flip_flop flip4(out[4], register[4], clk, rst);
    d_flip_flop flip5(out[5], register[5], clk, rst);
    d_flip_flop flip6(out[6], register[6], clk, rst);
    d_flip_flop flip7(out[7], register[7], clk, rst);
    always_ff @(posedge clk)
    begin
        if(shift) shift_in <= 0;
        else if(~shift) shift_in <= 1;
    end
endmodule
```

SystemVerilog TestBench for Shift Register

```
module shifter_sim( );
    logic [7:0]in_1;
    logic load;
    logic clk;
    logic shift;
    logic rst;
    logic sin;
    logic [7:0]out;
    shift_reg_8bit dut(in_1, load, clk, shift, rst, sin, out);
    always begin
        clk = 0; #5; clk = 1; #5;
    end
    initial begin
        in_1 = 8'b01100110; shift = 1; #10;
        shift = 0; #10; shift = 1; #10;
        shift = 0; #10; shift = 1; #10;
        shift = 0; #10; shift = 1; #10;
    end
endmodule
```

SystemVerilog Module for Serial Adder

```
module serial_adder(
    input logic [7:0]in_1,
    input logic [7:0]in_2,
    input logic shift,
    input logic load,
    input logic rst,
    input logic clk,
    output logic [7:0]out,
    output logic next_cin
    );
    typedef enum logic[1:0] {S0,S1,S2,S3} statetype;
    statetype state, nextstate;
    logic [7:0]out1;
    logic [7:0]out2;
    logic shift_in;
    logic load_in;
    logic cin;
    logic cout;
    logic sout;
    //logic next_cin;

    always_ff @(posedge clk, posedge rst)
    begin
        if(rst) state <= S0;
        else state <= nextstate;
    end

    always_comb
    begin
        case(state)
            S0: nextstate=S1;
            S1: if(rst) nextstate = S0;
                else if(shift) nextstate = S2;
            S2: if(~shift)nextstate=S1;
                else if(shift) nextstate = S3;
            S3: if(~shift)nextstate=S1;
                else nextstate = S3;
            default: nextstate=S0;
        endcase
    end
    assign shift_in = state[1] & ~state[0];
    shift_reg_8bit reg1(in_1, load, clk, shift, rst, 0, out1);
    shift_reg_8bit reg2(in_2, load, clk, shift, rst, 0, out2);
    full_adder adder(out1[0], out2[0], cin, sout, cout);
    assign next_cin = shift_in ? cout : cin;
    d_flip_flop carry_reg(cin, next_cin, clk, rst);
    shift_reg_8bit reg_out(0 ,0 , clk, shift, rst, sout, out);
endmodule
```

SystemVerilog TestBench for Serial Adder

```
module serial_adder_sim();
    logic [7:0]in_1 = 8'b0;
    logic [7:0]in_2 = 8'b0;
    logic shift;
    logic load;
    logic rst;
    logic clk;
    logic [7:0] out;
    logic [7:0] out1;
    logic [7:0] out2;
    serial_adder dut(in_1, in_2, shift, load, rst, clk, out, out1, out2);
    always
        begin
        clk = 0; shift = 0; #10;
        clk = 1; shift = 1; #10;
        end
    initial
        begin
        in_1 = 8'b01001010; in_2 = 8'b00110011; load = 0; #5;
        load = 1; #5;
        end
endmodule
```