PES UNIVERSITY

100 Feet Ring Road, Banashankari Stage III, Dwaraka Nagar, Bengaluru, Karnataka 560085

Mini project submitted in partial fulfillment of Requirement for the award of engineering in third semester computer science and engineering

Digital Design & Computer Organization Lab

Title:- **Serial 2's Complement with a Shift Register and a flip–flop.**

Submitted By:-

BHAVANA S

PES1UG21CS807

UNDER THE GUIDANCE OF

Dr. NAGEGOWDA K S

Group with :-

BHAVANA S	**PES1UG21CS807**
NIKHIL RAVALLU	**PES1UG21CS830**
PRIYA GAWLI	**PES1UG21CS829**
NANDANA C L	**PES1UG21CS822**

ABSTRACT OF THE PROJECT:

As we see in the diagram below there is a serial input and serial output connected to the shift register in the circuit diagram. The clock of the register is connected to the clock signal of the D Flip-Flop. The serial output is being fed back to the input of the OR gate and its output P is fed to the input of the Flip-Flop and the output of the Flip flop is fed to the input of the input of the OR gate. We also have a reset signal to the Flip-Flop which is 0 initially and whenever the reset signal is 1 the value is resetted. The output of the Flip-Flop is also fed to the input of the XNOR gate and the other input is from the serial output. And at last the output of the XNOR is the input of the Shift Register and is also known as the Serial Input which eventually leads to the shift in the 2' s compliment of the shift register and the process repeats.

Flip flop and shift registers triggered at same frequency. Thus, both produce the outputs at the same time.

CIRCUIT DIAGRAM

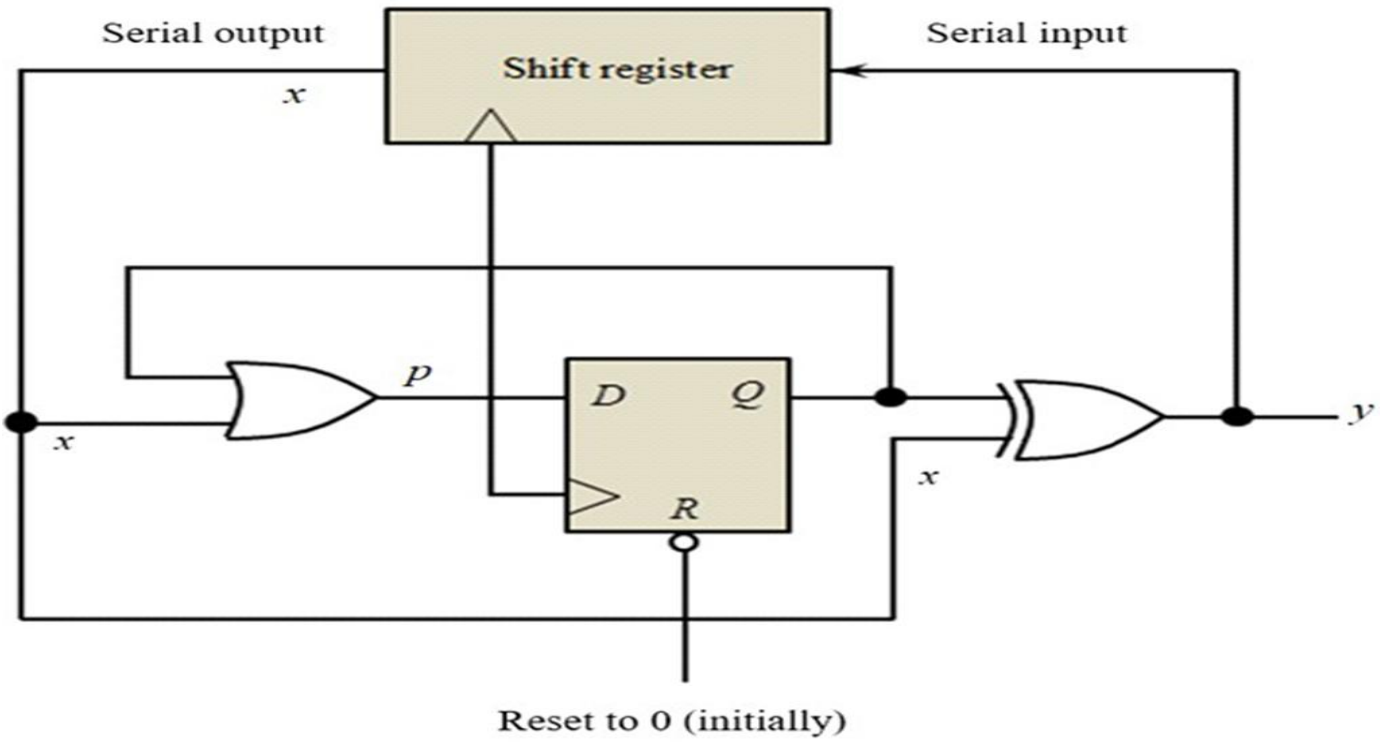

Figure 1

VERILOG CODE USED:

```
//used : = for blocking combintionl crkts and <= for non blocking sequential circuits

module xor2 (input wire i0, i1, output wire o);

  //selective complementer using xor and i1 as control

  assign o = i0^i1;

endmodule


module or2 (input wire i0, i1, output wire o);

  //oring for dflip flop input

  assign o = i0|i1;

endmodule


module dfrl (input clk,reset_b,shift_control,inputLSB,output wire out);

  reg df_out;

  wire df_data;


  //calculating the input to the dfrl

  or2 orgate(inputLSB,df_out,df_data);

  //reset and load flip flop

  //the flip flop is read in the posedge but written in neg edge clk cycle hence letting us save        an entire clock cycle

  always@(posedge clk,negedge reset_b)//sensitive to either pos clk or neg rest_b edge

  if(~reset_b) begin
```

```
    df_out <= 0;//reset the flip flop
  end
  else begin
    if(shift_control) begin
      df_out <= df_data;//supply the content to the output of dfrl
    end
  end
  assign out = df_out;
endmodule


module df (input clk, in, output wire out);
  reg df_out;
  always@(posedge clk) df_out <= in;//positive edge triggered DFF
  assign out = df_out;
endmodule


module shiftRegArr(input clk,load,shift_bool,seriallyInpBit,reset_b,input[7:0] data,output wire[7:0] outWires);
  //8 bit shift register with reset and load
  reg [7:0] myShiftRegs;
  always @ (posedge clk, negedge reset_b) //pos edge triggered
  if(reset_b == 0) begin
    myShiftRegs <= 0;
  end
  else begin
```

```
    if(load) myShiftRegs = data;//load the input data to be complemented serially

    else if(shift_bool) begin

      myShiftRegs <= {seriallyInpBit,myShiftRegs[7:1]};//inserting the xor's output at msb along with bits 1:7 of input since bit 0/LSB is gone for complementing

    end

  end

  assign outWires =  myShiftRegs;//store it back in the shiftWires because they need to be serially inputted with twos complement acc to question

endmodule



module Serial_twosComplementer(output y, input [7: 0] data, input load, shift_control, Clock, reset_b);



    wire[7:0] shiftwires;//wires passed to shiftRegArr to store the input data and serially get the twos complement of the individual bits

    wire Q;//stores the output from the dfrl which tells wheather to invert or not ie becomes 1 after seing the first 1 bit in input data

    wire inputLSB = shiftwires[0];//this will store the LSB of the input data that will be          complemented



    //instantiating the needed modules below:

    xor2 comp(inputLSB,Q,y);//based on Q(output) recieved from the dffrl it will either complement the inputBit or not and send output via y

    dfrl dffQ(Clock,reset_b,shift_control,inputLSB,Q);//supplies Q for selective inverting after seeing first 1 bit supplied

    shiftRegArr shiftReg0(Clock,load,shift_control,y,reset_b,data,shiftwires[7:0]);

endmodule
```

TEST BENCH FILE

```
module Serial_twosComplementer_Testbench();
    wire y;//this will be the output of the xor selective inverter
    reg [7: 0] data;//the input data
    reg load, shift_control, Clock, reset_b;//control signals
    reg [7: 0] twos_comp;//used to store the pure final output only.

    Serial_twosComplementer stc0 (y, data, load, shift_control, Clock, reset_b);

    always @ (posedge Clock, negedge reset_b)//concurrent execution
        if (reset_b == 0)
            twos_comp <= 0;
        else if (shift_control == 1 && load == 0) //shift the bits
            twos_comp <= {y, twos_comp[7: 1]};//concating the wires:y ie output bit of xor with the bits 1:7 of the
going to be output wire

        //all the below "Iniial" blocks will run at the same time concurrently
        initial #200 $finish;//this will finish after 200 time units of delay

        initial begin
            $dumpfile("test.vcd");
            $dumpvars(0,Serial_twosComplementer_Testbench);
        end
```

```
    initial begin//the statements listed under begin block execute sequentially one after the    other since '='
is used
        Clock = 0;
        while (1) begin //run an infinite loop to generate clk until simulation finishes
            #5 Clock = ~Clock; //rising clock edge at 5,10,15,20...
          end
        end
    initial begin
        #2 reset_b = 0;//reset the flip flops at 2nd time unit
        #4 reset_b = ~reset_b;//make reset_b bool -tive at 6th time unit
        end
    //each timing is absolute to when the group started to execte
    initial begin
        data = 8'h5A;//1011010 in binary
        //data = 8'h33;//00110011 in binary
        //data = 8'b1011010;//binary form
        #20 load = 1;//load the data into the shift registers
        #10 load = 0;//after 10s set load to false
        #20 shift_control = 1;//after 20 seconds shift one bit
        begin repeat (8+1) @ (posedge Clock);//The first posedge will trigger the for loop
                                    and afer 8 posedge the loop is terminated
          //after 8 pos edges of clock cycle we make shift control false to stop shifting of bits since there are only
8 bits
          shift_control = 0;
        end
```

```
    end

endmodule
```

OUTPUT