Lab Experiment #2
Realization of Digital circuits using behavioral and Switch level modelling

2.1 Objective: To realize the design of digital circuits in Verilog using behavioral and switch level modelling then simulating and synthesizing using EDA tools.

2.2 Software tools Requirement

Equipments:

Computer with Xilinx and Modelsim Software Specifications:

HP Computer P4 Processor – 2.8 GHz, 2GB RAM, 160 GB Hard Disk

Softwares: Synthesis tool: Xilinx ISE.

Simulation tool: ModelSim Simulator

2.3 Prelab Questions

(write pre lab Q & A in an A4 sheet)

1. Write the difference between initial and always block.
2. List the Reduction and Logical Operators.
3. Give the use of Blocking and Nonblocking statments.
4. Differentiate case, casex and casez statements.

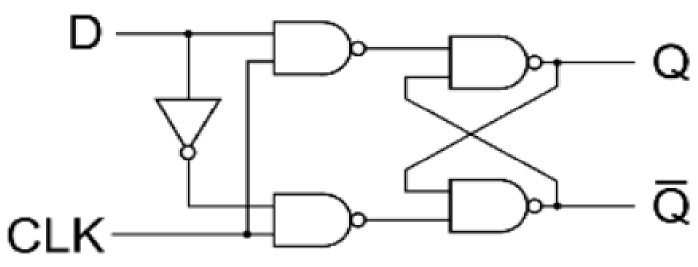
2.4.1 Problem 1: Write a Verilog code to implement Flip flops.

The following points should be taken care of:

1. Use If statement to design positive edge triggered D flip
2. Use case statement to design negative edge triggered T flip flop

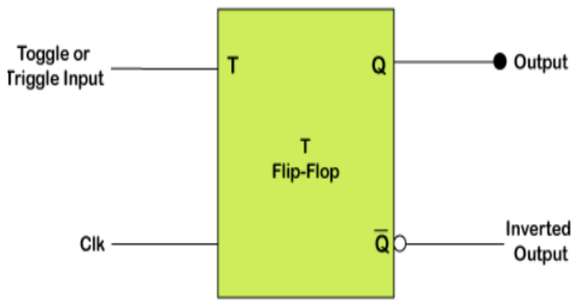
Logic Diagram – Problem 1.1

Q	D	Q(T+1)
0	0	0
0	1	1
1	0	0
1	1	1

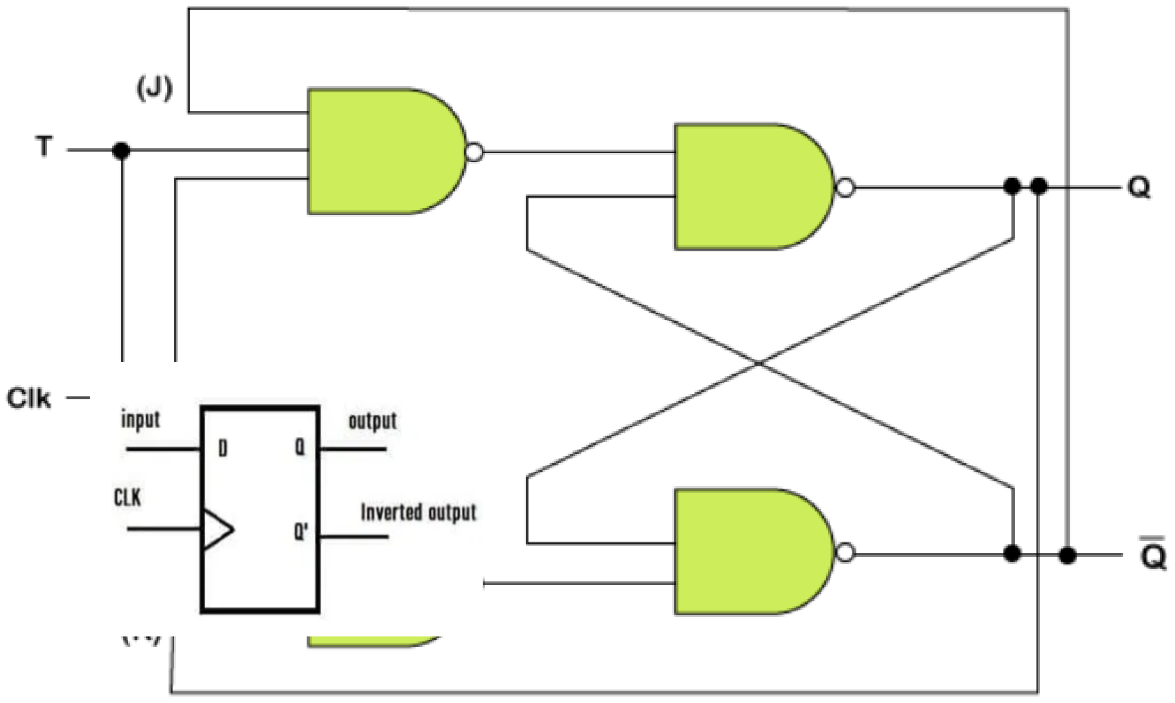


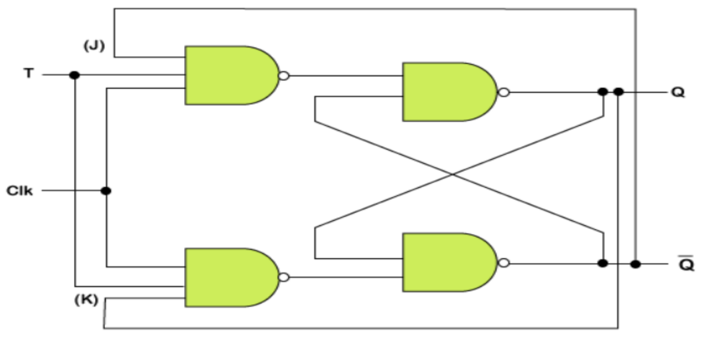
D flip flop

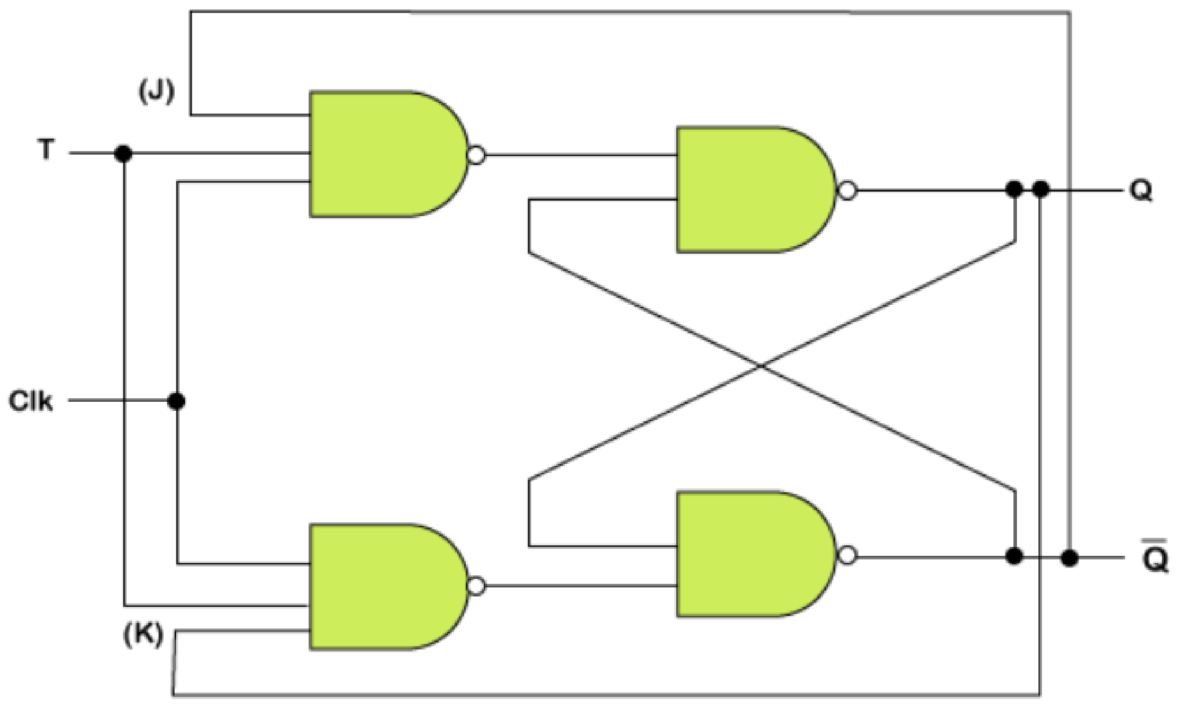
Logic Diagram – Problem 1.2



T	Q	Q'
0	0	1
0	1	0
1	0	1
1	1	0







Verilog Code - Problem 1

1.(a) POSITIVE EDGE TRIGGERED D FLIPFLOP USING IF STATEMENT

```
module dff(q, qbar, d, clk, clear);
   output q;
   output qbar;
   input d;
   input clk;
   input clear;
            reg q, qbar;
             always@(posedge clk or posedge clear)
begin
if(clear== 1)
begin
q <= 0;
qbar <= 1;
end
else
begin
q <= d;
qbar = !d;
end
end
endmodule
```

// TEST BENCH

```
module dff_tb_v;

            // Inputs
            reg d;
            reg clk;
            reg clear;

            // Outputs
            wire q;
```

```
        wire qbar;

        // Instantiate the Unit Under Test (UUT)
        dff uut (
                .q(q),
                .qbar(qbar),
                .d(d),
                .clk(clk),
                .clear(clear)
        );

        initial begin
                // Initialize Inputs
                d = 0;clk = 0;clear = 1;

                // Wait 100 ns for global reset to finish
                #100 d = 0;clear = 0;
        #100 d = 1;clear = 0;
                #100 d = 1;clear = 1;

                                // Add stimulus here

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

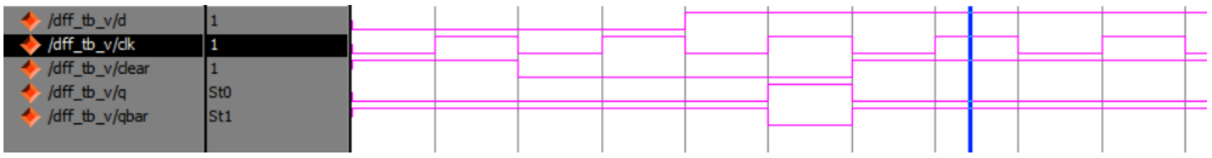
1.2) NEGATIVE EDGE TRIGGERED T FLIPFLOP USING CASE STATEMENT

```
module tffcase(q, clk, clr, t);

  output q;

  input clk;

  input clr;

  input t;

        reg q;
initial

        q=0;

        always@(negedge clk)

        begin

        case({clr,t})
        2'b10: q=0;


        2'b00: q=q;


        2'b01: q=~q;

        endcase

        end

endmodule
```

// TEST BENCH

```
module tffcase_tb_v;

        // Inputs

        reg clk; reg clr; reg t;

        //


        Outputs

        wire q;

        // Instantiate the Unit Under Test

        (UUT) tffcase uut

        (.q(q),.clk(clk),.clr(clr),.t(t)); initial begin

                // Initialize Inputs

                clk = 0; clr = 1; t =

                0;

                // Wait 100 ns for global reset to

                finish #100; clr=0;

                #100; clr=0; t=1;

                // Add stimulus here

        end
  always

        #50 clk=~clk;

endmodule
```

Waveforms – Problem 1



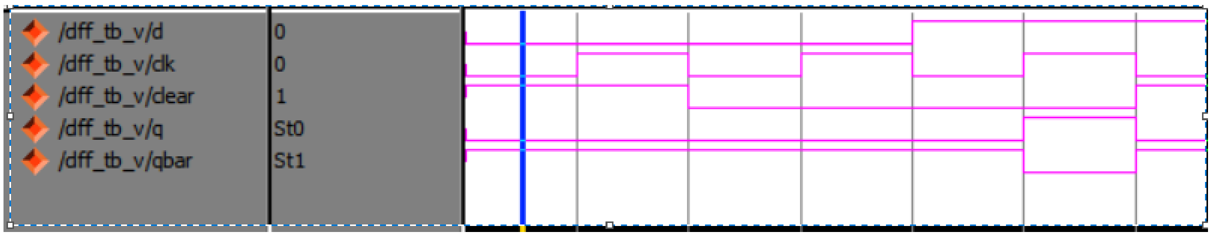


Fig 1.1 D Flipflop using IF statement

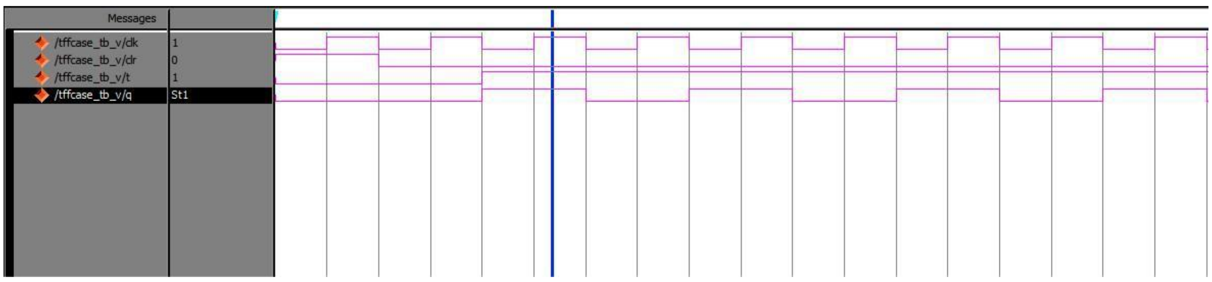


Fig 1.2 T Flipflop using CASE statement

2.4.2 Problem 2: Write a Verilog code to implement Counters.

The following points should be taken care of:

1. Use behavioral model to design Up-Down Counter. When mode ='1' do up counting and for mode='0' do down counting.
2. Use behavioral model to design Mod-N counter.
3. Design SISO register using behavioral model.

Logic Diagram – Problem 2

Verilog Code - Problem 2

2.(1) UP DOWN COUNTER USING BEHAVIOURAL MODEL

```
module updowncntr(q, clr, clk, mod);

  output reg [3:0] q;

  input clr;

  input clk;

  input mod;

        always@(posedge clk)

        begin

        case({clr,mod})

        2'b11 : q=0;

        2'b10 : q=0;

        2'b01 : q=q+1;

        2'b00 : q=q-1;

        endcase

        end
```

```
endmodule
```

// TEST BENCH

```
module updowncntr_tb_v;

        // Inputs

        reg clr; reg clk; reg mod;

        // Outputs

        wire [3:0] q;

        // Instantiate the Unit Under Test (UUT)

        updowncntr uut

        (.q(q),.clr(clr),.clk(clk),.mod(mod)); initial begin

                // Initialize Inputs

                clr = 1; clk = 0; mod = 1;

                // Wait 100 ns for global reset to

                finish #100; clr=0;

                #1000; mod=0;

                // Add stimulus here

        end

  always

        #50 clk=~clk;

Endmodule
```

2.(2) Mod N-COUNTER USING BEHAVIOURAL MODEL

```
module modN_ctr
# (parameter N = 10,
parameter WIDTH = 4)

( input   clk,
input   rstn,
output  reg[WIDTH-1:0] out);

always @ (posedge clk) begin
if (rstn) begin
out <= 0;
end else begin
if (out == N-1)
out <= 0;
else
out <= out + 1;
end
end
```

```
endmodule
```

// TEST BENCH

```
module modncounter_tb_v;

// Inputs
reg clk;
reg rstn;

// Outputs
wire [3:0] out;

// Instantiate the Unit Under Test (UUT)
modN_ctr uut (
.clk(clk),
.rstn(rstn),
.out(out)
);

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

// Wait 100 ns for global reset to finish
#100 rstn=0;

// Add stimulus here

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

2.(3) SISO REGISTER USING BEHAVIOURAL MODEL

```
module siso_register(shift_out, shift_in, clk);
input  shift_in;
input clk;
output  shift_out;
reg       shift_out;
reg  [2:0] data;
always @(posedge clk)
begin
data[0] <= shift_in ;
data[1] <= data[0];
data[2] <= data[1];
shift_out <= data[2];

end

endmodule
```

// TEST BENCH

```
module siso_tb_v;

// Inputs
reg shift_in;
reg clk;

// Outputs
wire shift_out;

// Instantiate the Unit Under Test (UUT)
siso_register uut (
.shift_out(shift_out),
.shift_in(shift_in),
.clk(clk)
);

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

// Wait 100 ns for global reset to finish

#100      shift_in = 0;
#100      shift_in = 1;
#100      shift_in = 0;
// Add stimulus here

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

Waveforms – Problem 2

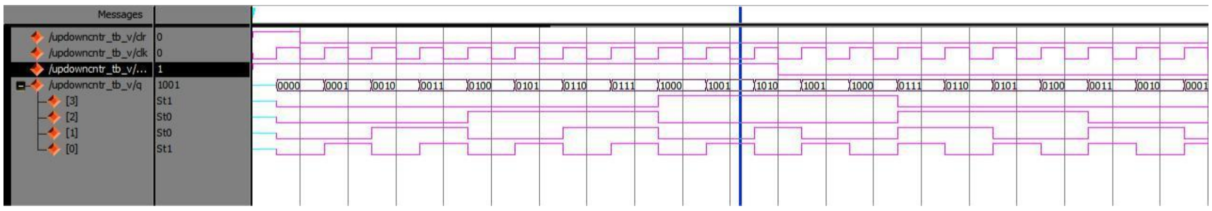


Fig 2.(1) Up Down counter using behavioral modelling

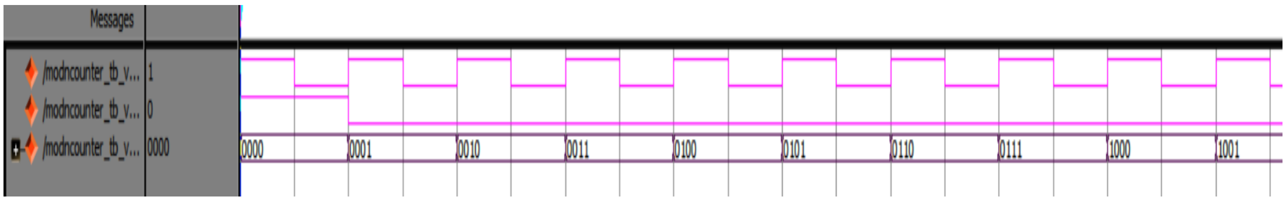


Fig 2.(2) Mod-N counter using behavioral modelling

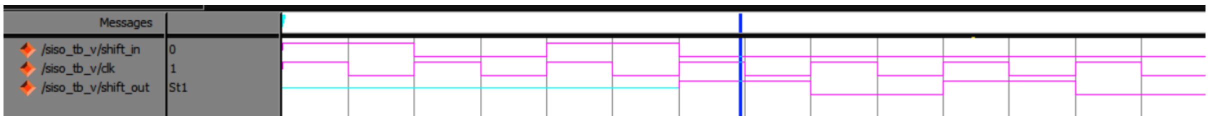


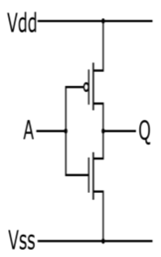
Fig 2.(3)SISO register using behavioral modelling

2.4.3:

1. Design inverter logic using verilog switch level modeling and verify the simulation result using test bench.

2. Design two input CMOS NAND logic using verilog switch level modeling and verify the simulation result using testbench.
3. Design 2:1 Mux using CMOS switches and write verilog coding using switch level modeling and verify the simulation result.

Logic Diagram-Problem 1



2.4.3 Verilog Code - Problem 1

CMOS Inverter

```
module inverter(out,data);
output out;
input data;
supply1 vdd;
supply0 gnd;
pmos p1(out,vdd,data);
nmos n1(out,gnd,data);
endmodule
```

CMOS Inverter_TB

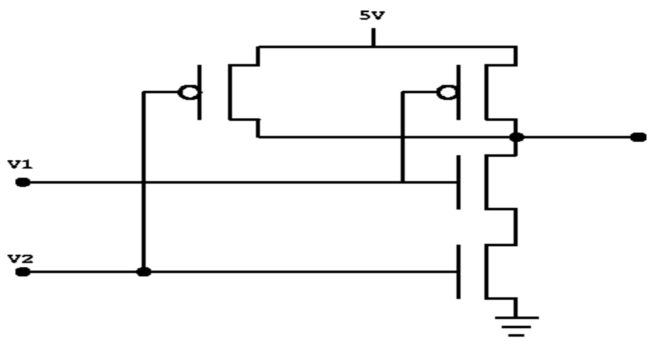
```
module inverter_tb_v;
reg data;
inverter uut (.out(out), .data(data));
initial begin
data = 0;
#100 data = 1;
end
endmodule
```

6.4 Waveforms – Problem 1

Fig 6.4.1 CMOS inverter

.

Logic Diagram-Problem 2



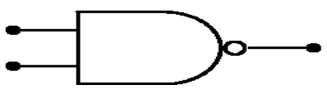


Fig Two input NAND

2.4.3 Verilog Code - Problem 2

CMOS NAND Logic

```
module nandcmos(out,a,b);
wire t;
output out;
input a,b;
supply1 vdd;
supply0 gnd;
pmos m1(out,vdd,a);
pmos m2(out,vdd,b);
nmos m3(out,t,a);
nmos m4(t,gnd,b);
endmodule
```

CMOS NAND Logic_TB

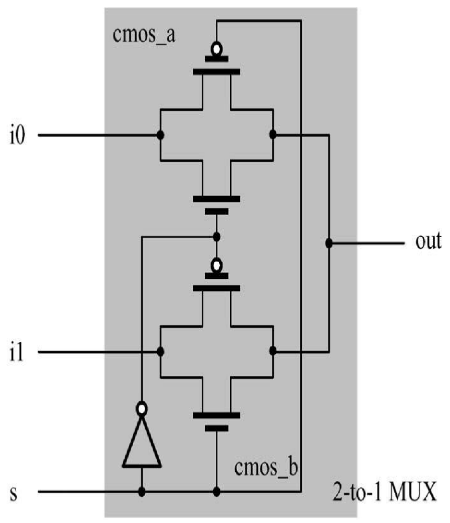
```
module nandcmos_tb_v;
reg a; reg b;
wire out;
nandcmos uut (
.out(out),  .a(a), .b(b)
);
initial begin
a = 0;  b = 0;
#100    a = 0;  b = 1;
#100    a = 1;  b = 0;
#100    a = 1;  b = 1;
end
```

```
endmodule
```

2.4.3 Waveforms – Problem 2

Fig 2.4.3 Two input NAND

Logic Diagram-Problem 3



2:1 Mux USING cmos

2.4.3 Verilog Code - Problem 3

CMOS Mux 2X1

```
module cmos_mux(out,s,i1,i0);
output out;
input s,i0,i1;
wire sbar;
not n1(sbar,s);
cmos(oout,i0,sbar,s);
cmos(out,i1,s,bar);
endmodule
```

CMOS Mux 2X1_TB

```
module cmos_mux_tb_v;
reg s;reg i1;reg i0;
wire out;
cmos_mux uut (.out(out), .s(s),
.i1(i1),.i0(i0));
initial begin
s = 0;i1 = 1;i0 = 0; #100;
s = 0;i1 = 0;i0 = 1; #100;
s = 1;i1 = 0;i0 = 1; #100;
s = 1;i1 = 1;i0 = 1;
end
endmodule
```

2.4.3 Waveforms – Problem 3

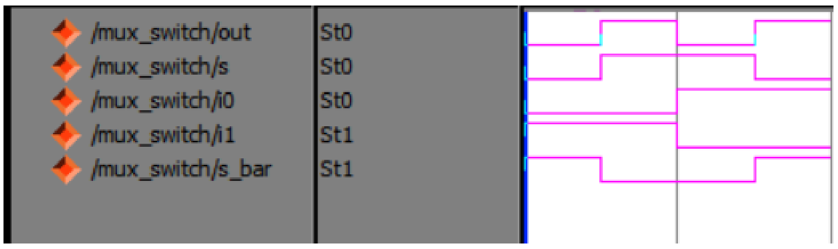


Fig 2.4.3 2X1 Multiplexer using CMOS

2.5 Post Lab :

Write a Verilog HDL Code to implement a SIPO and PIPO shift registers.

2.6 Result:

Thus, the design of Digital circuits using behavioral and Switch level modelling is simulated in Verilog and synthesized using EDA tools.