Lab Assignment 3

Mumuksh Jain – 22110160

Nishant Kumar – 22110170

Code for toggle:

```
`timescale 1ns / 1ps


module tff(
input clk,     //clock signal
input rst,     //reset signal
input t,       //toggle signal, will be 1 to toggle
output reg q,  //output bit
input en       //enable signal
);


always @ (posedge clk or negedge rst)
begin
  if (!en)       //if not enabled then do nothing
    q<=q;       //return the same thing
  else
    if (!rst)  //active low reset signal
     q <= 0;  //so reset when rst=0
    else
      if (t)  //toggle=1 means change output
        q <= ~q;
      else
        q <= q;
   end
endmodule
```

Testbench for toggle:

```
`timescale 1ns / 1ps

module tff_tb();
reg t,clk,rst;
wire q;

tff uut(clk,rst,t,q);

initial
begin
clk=0;
   forever #5 clk=~clk;
end

initial
begin
t=1;
rst=0;
clk=1;
#5;
rst=1;
#10;
t=0;
#2;
t=1;
#20;
rst=0;
#5;
rst=1;
```

```
t=0;
#16;
t=1;
#5;
$finish();
end
endmodule
```

Code for counter:

```
`timescale 1ns / 1ps

module synch_up(
input clk,      //clock signal
input wire rst, //reset signal
input t,        //toggle value, will be kept 1 as synchronous counter
input sel1,     //select signal for choosing b/w Up(0) and Down(1) counter
input sel2,     //select signal for choosing b/w BCD(1) and Binary(0)
output [3:0]q,  //4-bit output
input en        //enable signal
);

tff a1(.clk(clk),.rst(rst),.t(t),.q(q[0]),.en(en));
and(f1,q[0],q[0]);
and(e,q[0],~sel1);
and(f,~q[0],sel1);
or(g2,e,f);

and(g3,g2,~sel2);
and(o2,~f4,f1,sel2,~sel1);
or(g1,g3,o2);
or(g10,f2,f3,f4);
and(g11,g10,~q[0],sel1,sel2);
or(g22,g1,g11);

tff a2(.clk(clk),.rst(rst),.t(g22),.q(q[1]),.en(en));
and(f2,q[1],q[1]);
and(h,q[1],~sel1);
and(i,~q[1],sel1);
or(j1,h,i);

and(s1,g2,j1);

and(s10,f4,~f1,sel1,sel2);
and(s9,~f1,~f2,f3,sel1,sel2);
or(s11,s9,s10);
and(s12,sel1,sel2);
and(s13,~s12,s1);
and(s14,s12,s11);
or(s15,s13,s14);

tff a3(.clk(clk),.rst(rst),.t(s15),.q(q[2]),.en(en));

and(f3,q[2],q[2]);
and(k,q[2],~sel1);
and(l,~q[2],sel1);
or(m,k,l);

and(s4,s1,m);
and (o1,f4,f1,sel2);
or(h1,o1,s4);
```

```
and(h11,~f2,~f3,~f1,sel1,sel2);

and(h12,sel1,sel2);

and(h13,~h12,h1);

and(h14,h12,h11);

or(h15,h13,h14);


tff a4(.clk(clk),.rst(rst),.t(h15),.q(q[3]),.en(en));


and(f4,q[3],q[3]);


endmodule
```

Code for shift register:

```
`timescale 1ns / 1ps


module shfit(

input clk,            //clock signal

input [3:0]q,         //4 bit input

input [1:0]m,         //2 bit multiplexer input for 4 diff cases

input sir,            //shift insert right

input sil,            //shift insert left

input en,             //enable signal

output reg [3:0]qshift  //4 bit output

);


wire [3:0]q1;         //a temp variable which stores the incoming load

assign q1=q;          //necessary when parallel load required in future


always@(posedge clk)
```

```
begin
if (en)             //active low signal
   qshift=4'b0000;    //till the counter counts
else
   if(m==2'b00)       //for m=00, parallel load
      begin
      qshift[3]<=q1[3];
      qshift[0]<=q1[0];
      qshift[1]<=q1[1];
      qshift[2]<=q1[2];
      end
   else if (m==2'b10)  //for m=10, left shift
   begin
      qshift[0]<=sil;
      qshift[1]<=qshift[0];
      qshift[2]<=qshift[1];
      qshift[3]<=qshift[2];
   end

   else if(m==2'b01)   //for m=01, right shift
      begin
      qshift[3]<=sir;
      qshift[2]<=qshift[3];
      qshift[1]<=qshift[2];
      qshift[0]<=qshift[1];
   end
   else if(m==2'b11)   //for m=11, no shift
      qshift<=qshift;
end

endmodule
```

Testbench for shift register only:

```
`timescale 1ns / 1ps

module shift_tb();


reg clk;

reg [3:0]q;

reg sil;

reg sir;

reg [1:0]m;

reg en;

wire [3:0]qshift;


shfit uut(clk,q,m,sir,sil,en,qshift);


initial

begin

clk=0;

   forever #5 clk=~clk;

end


initial

begin

clk=1;

m=2'b00;

en=1;

sil=0;

sir=0;

q=4'b1001;

#10;
```

```
en=0;

#5;

#15;

m=2'b10;

sil=1;

#10;

sil=0;

#5;

sil=1;

#15;

m=2'b01;

sir=0;

#15;

sir=1;

#5;

sir=0;

#10;

m=2'b11;

#50;

m=2'b00;

#22;

$finish();

end


endmodule
```

Code for final top module:

```
`timescale 1ns / 1ps

module sync_sr(
input clk,
input rst,
input sel1,
input sel2,
output [3:0]q,
input en,
input t,
input [1:0]m,
input sil,
input sir,
output [3:0]qshift);

synch_up a1(.clk(clk),.rst(rst),.t(t),.sel1(sel1),.sel2(sel2),.q(q),.en(en));

shfit a2(.clk(clk),.q(q),.m(m),.sir(sir),.sil(sil),.en(en),.qshift(qshift));

endmodule
```

Final Testbench:

```
`timescale 1ns / 1ps


module sync_sr_tb();

reg clk;

wire [3:0]q;

reg rst;

reg t;

reg sil;

reg sir;

reg en;

reg [1:0]m;

reg sel1;

reg sel2;

wire [3:0]qshift;
```

```
sync_sr uut(clk,rst,sel1,sel2,q,en,t,m,sil,sir,qshift);



initial

begin

clk=0;

   forever #5 clk=~clk;

end


initial

begin

t=1;

m=2'b00;

rst=0;

clk=1;

//BCD Down Count

en=1;

sel1=1;

sel2=1;

sil=0;

sir=0;

#7;

rst=1;

#82;

//Counter stopped, shift register started

en=0;

m=2'b00;

#12;

m=2'b10;

sil=1;

#14;
```

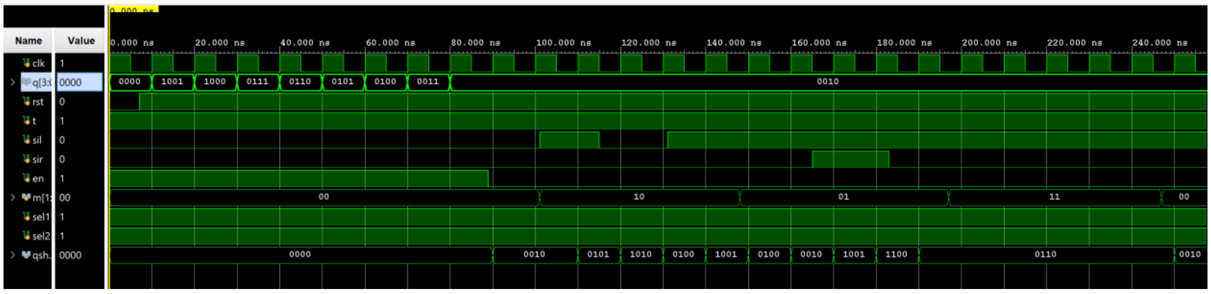
```
sil=0;

#16;

sil=1;

#17;

m=2'b01;

sir=0;

#17;

sir=1;

#18;

sir=0;

#14;

m=2'b11;

#50;

m=2'b00;

#22;

//BCD Up Counter

en=1;

sel1=0;

sel2=1;

sil=0;

sir=0;

#7;

rst=0;

#27;

rst=1;

#102;

//Counter stopped, shift register started

en=0;

m=2'b00;

#12;

m=2'b10;
```

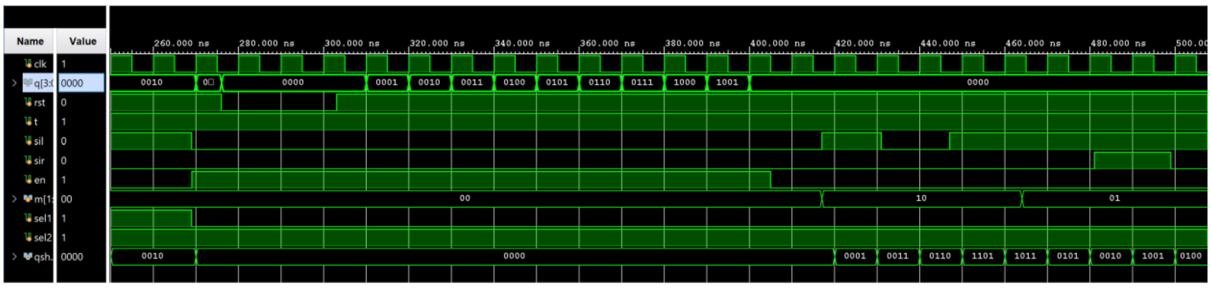
```
sil=1;

#14;

sil=0;

#16;

sil=1;

#17;

m=2'b01;

sir=0;

#17;

sir=1;

#18;

sir=0;

#14;

m=2'b11;

#50;

m=2'b00;

#22;

//Binary Down Counter

en=1;

sel1=1;

sel2=0;

sil=0;

sir=0;

#7;

rst=1;

#82;

//Counter stopped, shift register started

en=0;

m=2'b00;

#12;

rst=0;
```

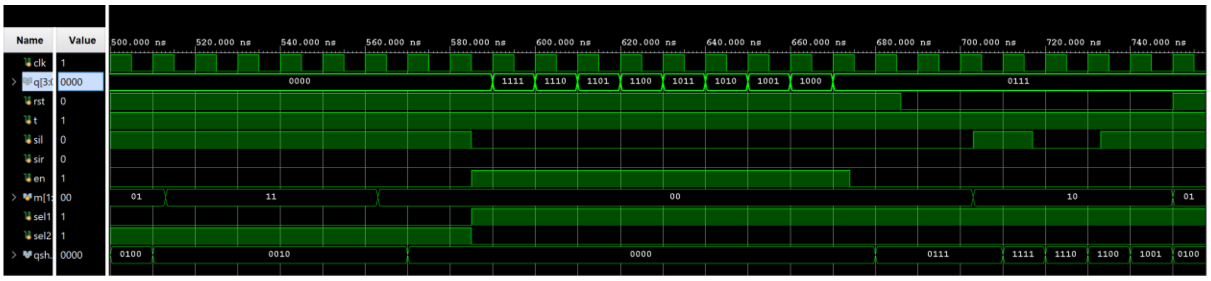
```
#17;
m=2'b10;
sil=1;
#14;
sil=0;
#16;
sil=1;
#17;
rst=1;
m=2'b01;
sir=0;
#17;
sir=1;
#18;
sir=0;
#14;
m=2'b11;
#50;
m=2'b00;
#22;
//Binary Up Counter
en=1;
sel1=0;
sel2=0;
sil=0;
sir=0;
#7;
rst=1;
#72;
//Counter stopped, shift register started
en=0;
```

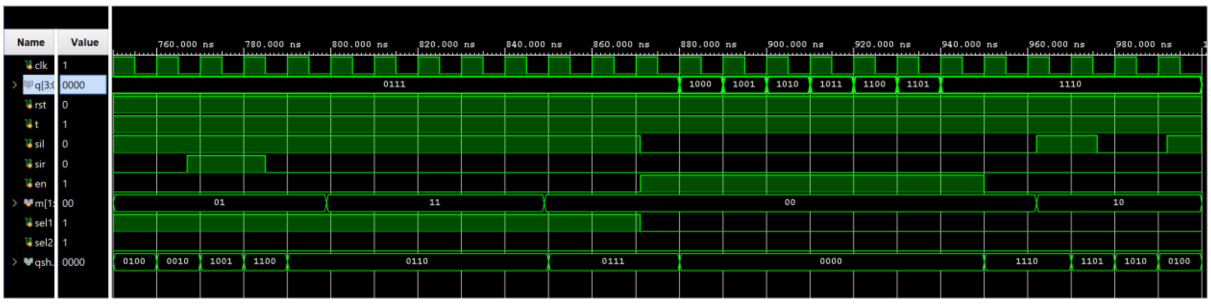
```
m=2'b00;

#12;

m=2'b10;

sil=1;

#14;

sil=0;

#16;

sil=1;

#17;

m=2'b01;

sir=0;

#17;

sir=1;

#18;

sir=0;

#14;

m=2'b11;

#40;

m=2'b00;

#22;

$finish();

end


endmodule
```

Simulation:


Name
Value
clk
q[3:
rst
t
sil
sir
en
m[1:
sel1
sel2
qsh.


Name
Value
clk
q[3:
rst
t
sil
sir
en
m[1:
sel1
sel2
qsh.


Name
Value
clk
q[3:
rst
t
sil
sir
en
m[1:
sel1
sel2
qsh.


Name
Value
clk
q[3:
rst
t
sil
sir
en
m[1:
sel1
sel2
qsh.