EXPERIMENT NO.-4

Title : To realize various configurations of shift registers and counters using D flipflop.

Lab Exercise:

1. Design a 4-bit serial in and serial out shift register using D flip flop.

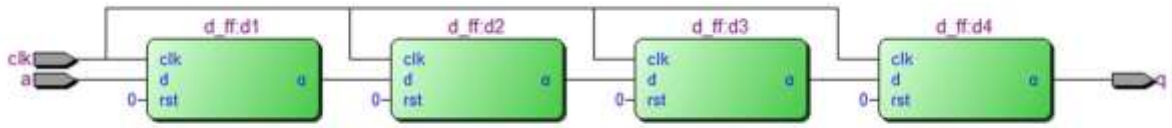
Code:

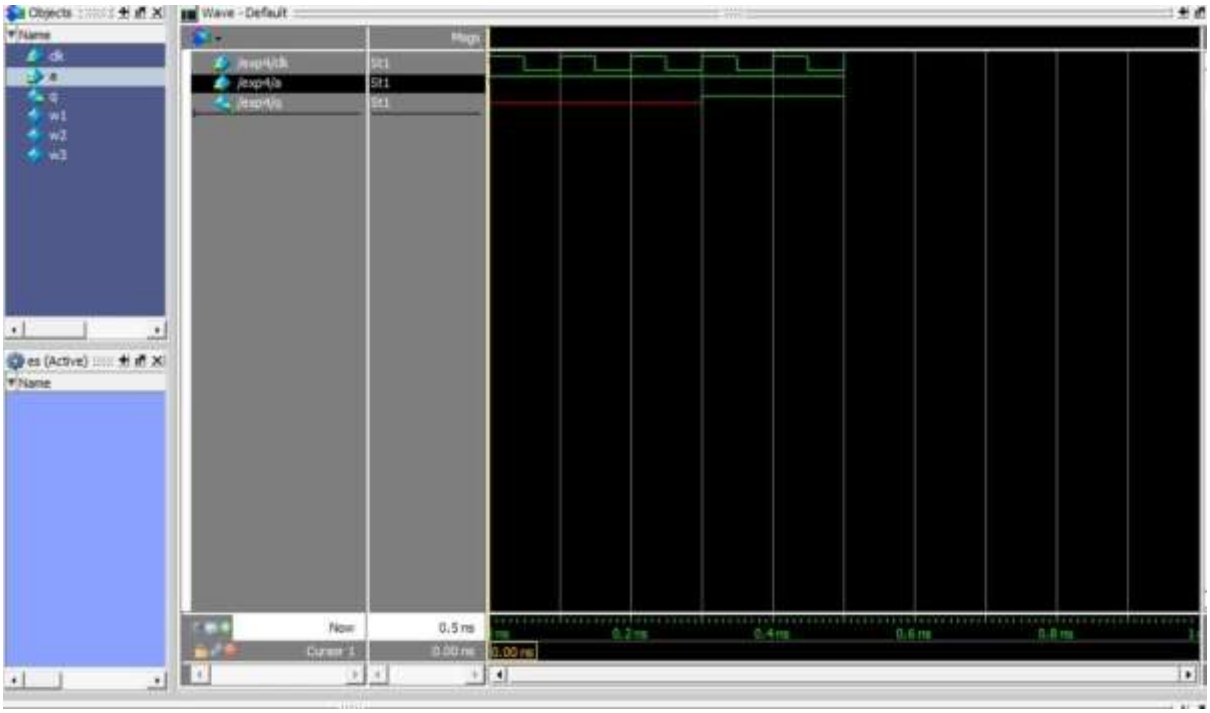
```
module exp4(input clk, input a, output q);
wire w1,w2,w3; d_ff
d1(clk,a,0,w1); d_ff
d2(clk,w1,0,w2); d_ff
d3(clk,w2,0,w3);
d_ff d4(clk,w3,0,q);
endmodule

module d_ff (
input       clk,
input         d,
input rst,
 output reg q);

 always @(posedge clk)
 begin
   if (rst)
q <= 1'b0;
else
q<=d;
 end
endmodule
```

Output:


d_ff:d1
d_ff:d2
d_ff:d3
d_ff:d4
clk
a
clk
d
rst
q
0


Objects
Wave - Default
Now
Cursor 1

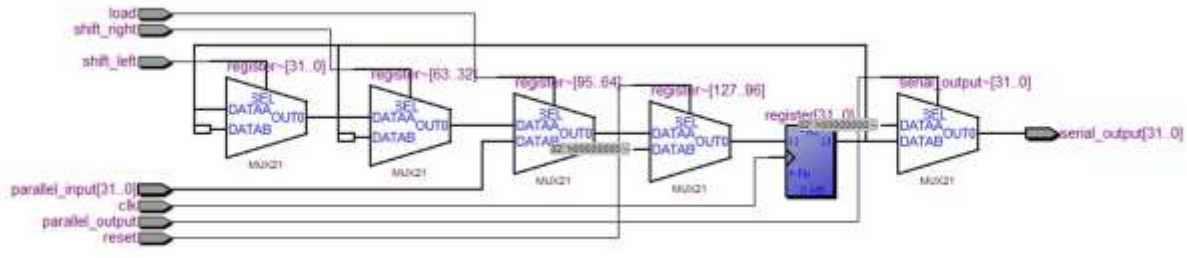
2. Implement a 32-bit shift register having 4 control inputs (load, shift-right, shift-left, paralleloutput).

Code:

```
module exp4( input clk,
input reset, input load,
input shift_right, input
shift_left, input
parallel_output, input
[31:0] parallel_input,
output [31:0] serial_output
);
reg [31:0] register; always
@(posedge clk) begin
if (reset) begin register
<= 32'b0;
end else begin if (load)
begin register <=
parallel_input; end
else if (shift_right) begin register <=
{register[0], register [31:1]}; end
else if (shift_left) begin register <=
{register [30:0], register [31]}; end end
end

assign serial_output =(parallel_output) ? register: 32'b0;
 endmodule
```

Output:



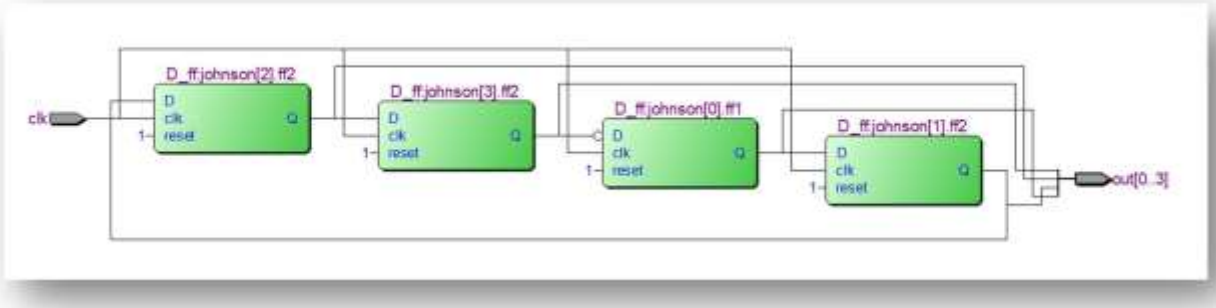
3. Design a 4-bit Synchronous Johnson counter.

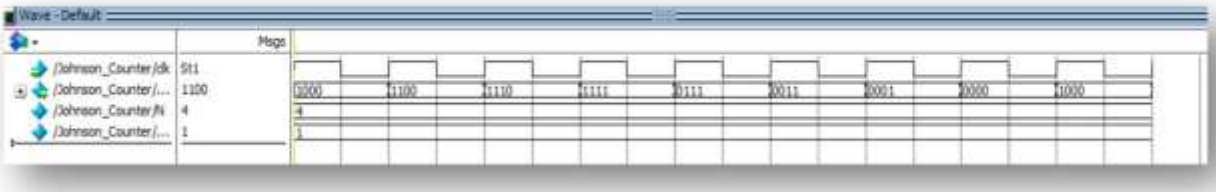
Code:

```
module Johnson_Counter(clk,out);
parameter N = 4; input clk; output
[0:N-1]out; parameter reset=1;
generate genvar ; for(i=0;i<N;i=i+1)
begin:johnson if(i==0)
D_ff ff1 (~out[N-1],clk,reset,out[i]);
else
D_ff ff2 (out[i-1],clk,reset,out[i]);  end
endgenerate
endmodule

module D_ff(D,clk,reset,Q);
input D,clk,reset; output reg
Q=0; always@(posedge clk)
begin if(~reset) Q = 0; else
Q = D;
end
 endmodule
```

Output:





Post-Lab Exercise:

1.Design an n-bit ring counter using a D flip flop.

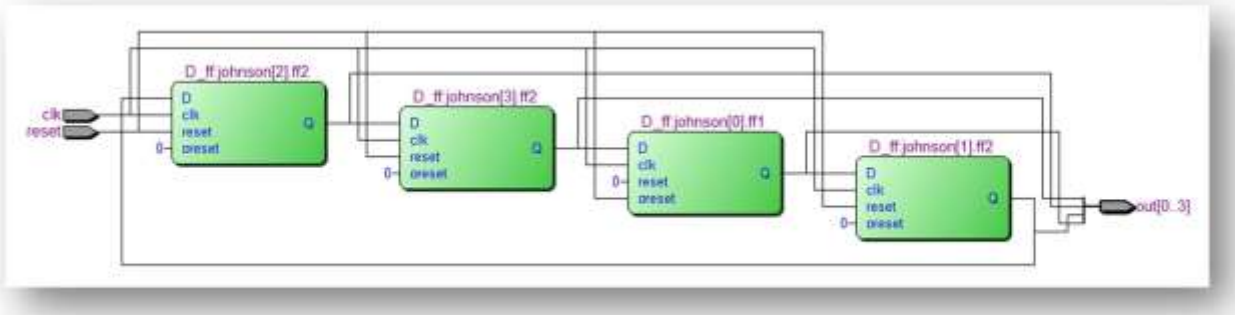
Code:

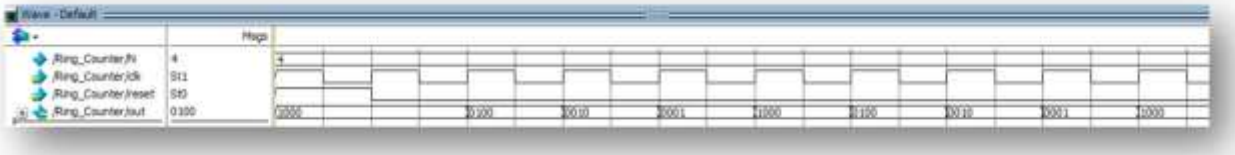
```
module Ring_Counter(clk,reset,out);
parameter N = 4; input clk,reset;
output [0:N-1]out; generate genvar
i; for(i=0;i<N;i=i+1) begin:Ring
if(i==0)
D_ff ff1 (out[N-1],clk,1'b0,reset,out[i]);
else
D_ff ff2 (out[i-1],clk,reset,1'b0,out[i]);
 end
endgenerate
endmodule

// D Flip flop module module
D_ff(D,clk,reset,preset,Q); input
D,clk,reset,preset;
output reg Q;

always@(posedge clk)
begin case
({preset,reset}) 2'b00 :
Q = D;
2'b01 : Q = 1'b0;
2'b10 : Q = 1'b1;
2'b11 : Q = 1'bx;
endcase
end
endmodule
```

Output:





2.Design and realize a 32-bit barrel shifter.

Code:

```
module Barrel_Shifter(data_in,data_out,clk,shift_amount);
parameter shift = 5 ; parameter N = 2**shift; input [N-1:0]
data_in; output wire [N-1:0] data_out; input [shift-1:0]
shift_amount; input clk; wire [N-1:0] mid[shift:0]; assign
mid[0] = data_in; generate genvar i,j;
for(i=0;i<shift;i=i+1)  // decides levels begin:loop1
for(j=0;j<N;j=j+1)   // decide no. of mux in 1 level begin:loop2 if (j<2**i)
mux_2_1 f2(shift_amount[i],{mid[i][j],1'b0},mid[i+1][j],clk); else
mux_2_1 f5(shift_amount[i],{mid[i][j],mid[i][j-(2**i)]},mid[i+1][j],clk);
end end endgenerate assign data_out = mid[shift]; endmodule

module mux_2_1(s,data,out,clk);
input clk,s; input
[1:0] data; output
out;
 assign out = ~s&data[0] | s&data[1];
endmodule
```

Output: