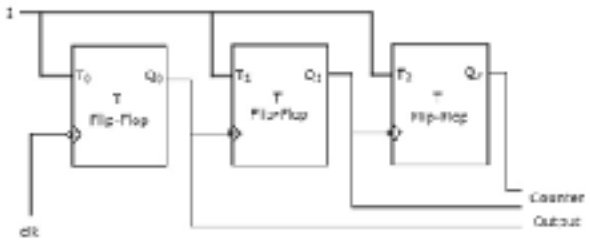
Amrita Vishwa Vidyapeetham
Amrita School of Computing, Bangalore
Department of Computer Science and Engineering
19CSE211 Computer Organization and Architecture
Lab Handout - 5
Sequential Circuits
Counters and Shift Registers

Exercise Problems

1. Write a verilog code to implement 3 bit asynchronous counter using T flip-flop

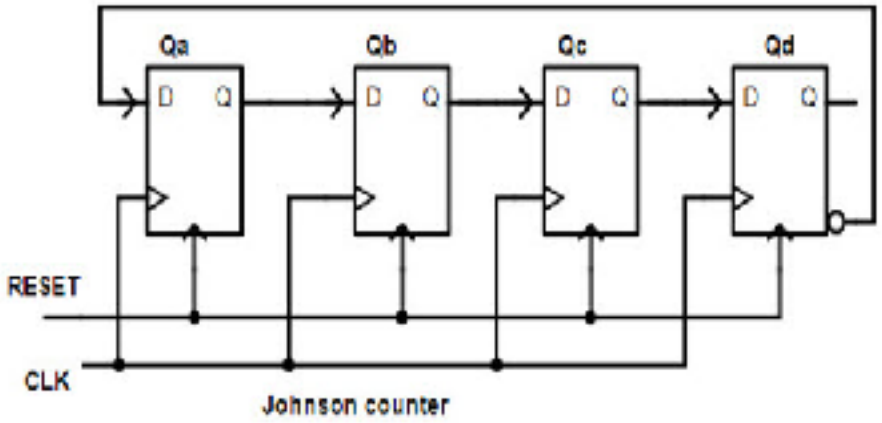


```
module tff(T, clk, Q, reset);

input T, clk, reset;
output Q;
reg Q;

always @(posedge clk)
begin
     if (reset)
          Q = 0;
     else begin
          if(T)
               Q = ~Q;
          else
               Q = Q;
     end

end
endmodule

module a_counter_3(T, Q, clk,
reset);

input [2:0] T;
input clk, reset;
output [2:0] Q;


tff i1(T[0], clk, Q[0], reset);
tff i2(T[1], ~Q[0], Q[1], reset);
tff i3(T[2], ~Q[1], Q[2], reset);
endmodule
```

```
module a_counter_3_tb;

reg [2:0] T;
reg clk, reset;
wire [2:0] Q;

a_counter_3 i1(T, Q, clk, reset);

always #5 clk = ~clk;

initial
begin
     T = 3'b000; reset = 1; clk = 1'b0;
     $monitor("Time=%f, T=%3b, clk=%b, Q=%3b",
$time, T, clk, Q);
     #10 reset =0;
     #10 T=3'b111;
     #80 $finish;
end

endmodule
```

2. Write a verilog code to implement 3-bit Johnson counter

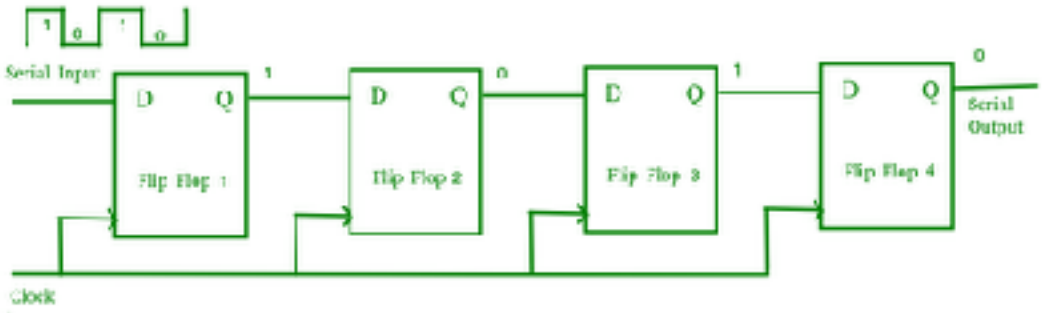
Q_A	Q_B	Q_C	Q_D
0	0	0	0
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1
repeat			




```
module dff(D, clk, Q, reset);

input D, clk, reset;
output Q;
reg Q;

always @(posedge clk)
if (reset) begin
    Q = 0;
end
else begin
    Q = D;
end

endmodule

module johnson_counter(reset,
clk, Q);

input reset, clk;
output [3:0] Q;

dff i1(~Q[0], clk, Q[3],
reset);
dff i2(Q[3], clk, Q[2], reset);
dff i3(Q[2], clk, Q[1], reset);
dff i4(Q[1], clk, Q[0], reset);

endmodule
```

```
module johnson_counter_tb;

reg reset, clk;
wire [3:0] Q;

johnson_counter i(reset, clk, Q);

always #5 clk = ~clk;

initial
begin
    clk=1'b0; reset=1'b1;
    $monitor("Time:%f, clk=%b,
Q=%4b", $time, clk, Q);
    #10 reset = 1'b0;
    #100 $finish;
end
endmodule
```

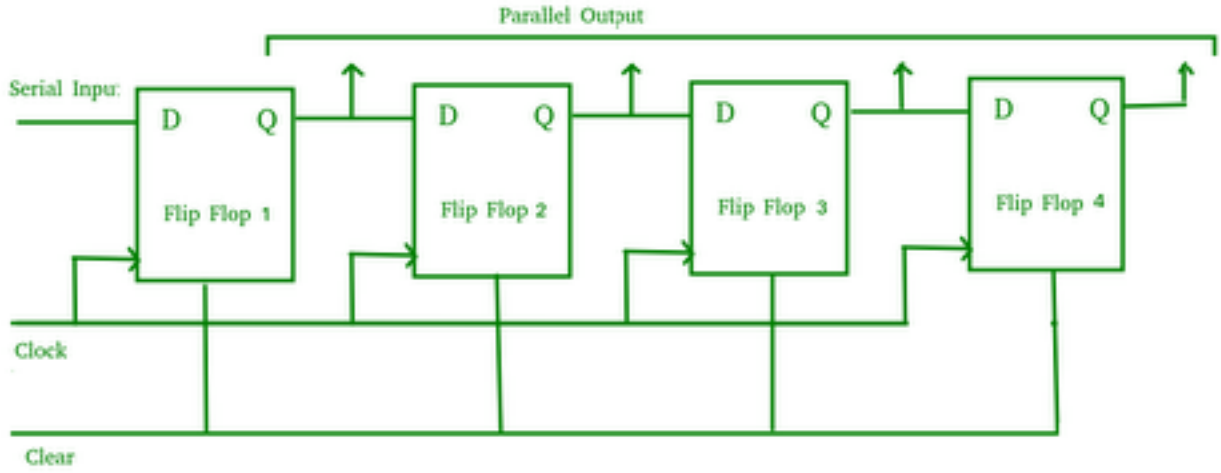
3. Write a verilog code to implement 4 bit Serial In Serial Out shift registers




```
module dff(D, clk, Q);

input D, clk;
output Q;
reg Q;

always @(posedge clk)
begin
    Q = D;
end

endmodule

module siso(in, out, clk);
input in, clk;
output out;
wire [2:0] Q;

dff i1(in, clk, Q[0]);
dff i2(Q[0], clk, Q[1]);
dff i3(Q[1], clk, Q[2]);
dff i4(Q[2], clk, out);

endmodule
```

```
module siso_tb;
reg in, clk;
wire out;

siso i(in, out, clk);

always #5 clk = ~clk;

initial
begin
    in = 1'b0; clk = 1'b0;
    $monitor("Time:%f, in:%b, clk:
%b, out:%b",$time, in, clk, out);
    #10 in = 1'b1;
    #10 in = 1'b0;
    #10 in = 1'b1;
    #10 in = 1'b0;
    #60 $finish;
end

endmodule
```

4. Write a verilog code to implement 4 bit Serial In parallel Out shift registers



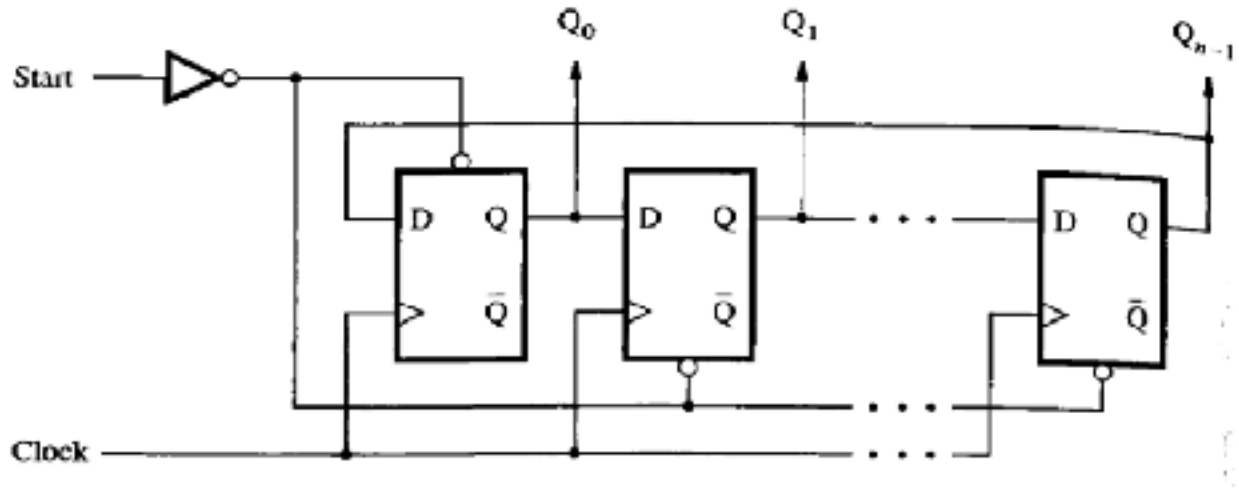

```
module dff(D, clk, Q);

input D, clk;
output Q;
reg Q;

always @(posedge clk)
begin
    Q = D;
end

endmodule

module sipo(in, out, clk);
input in, clk;
output [3:0] out;

dff i1(in, clk, out[0]);
dff i2(out[0], clk, out[1]);
dff i3(out[1], clk, out[2]);
dff i4(out[2], clk, out[3]);

endmodule
```

```
module sipo_tb;
reg in, clk;
wire [3:0] out;

sipo i(in, out, clk);

always #5 clk = ~clk;

initial
begin
    in = 1'b0; clk = 1'b0;
    $monitor("Time:%f, in:%b,
clk:%b, out:%4b",$time, in, clk,
out);
    #10 in = 1'b1;
    #10 in = 1'b0;
    #10 in = 1'b1;
    #10 in = 1'b0;
    #60 $finish;
end

endmodule
```

Assignment

1. Write a verilog code to implement 3-bit Ring counter.

Hint: Preset signal should be added in the D flip flop module



2. Write a verilog code to implement 4 bit parallel In parallel Out shift registers