Bangladesh University of Engineering and Technology

Department of Electrical and Electronics Engineering

Course Number: EEE 304

Course Title: Digital Electronics Laboratory

Experiment Number: 04

Name of the Experiment(s):

Design, Simulation, and Test of Sequential Circuits Using Verilog and Implementation In FPGA

Prepared by: Tasmin Khan

Student ID: 1906044

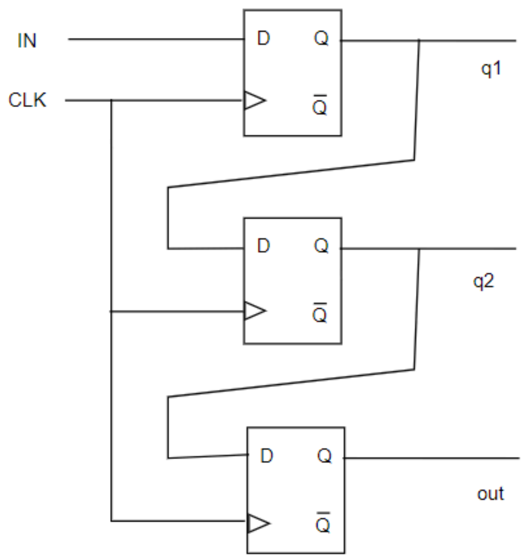
Level:3 Term:2

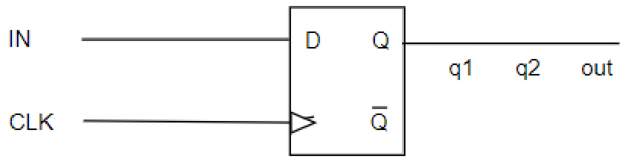
Report Task 1

Question: For the following code snippets, draw the conceptual circuit diagram of each code that it represents and explain the difference.

```
always @ (posedge clk)
begin
  q1 <= in;
  q2 <= q1;
  out <= q2;
end
```

```
always @ (posedge clk)
begin
  q1 = in;
  q2 = q1;
  out = q2;
end
```





Discussion:

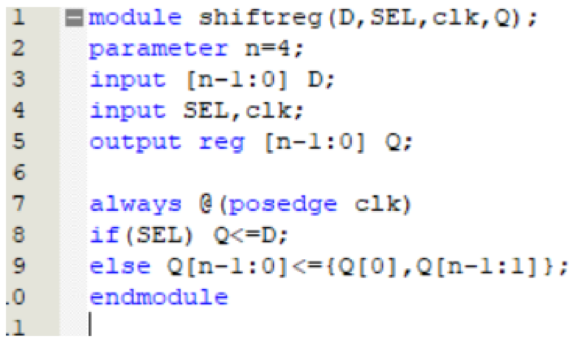
‘<=’ is non-blocking in nature. This means that in an `always` block, every line will be executed in parallel. Hence leading to implementation of sequential elements. In this way, out will store the previous value of q2 while q2 loads new value. This new value is the old one from q1 and q1 stores the value from IN. Thus this is a time dependent loop.

‘=’ is a blocking statement. In an `always` block, the line of code will be executed only after it's previous line has executed. Hence, they happen one after the other, just like combinational logic in a loop. In this way, out,q2,and q1 all will ultimately be assigned the value of IN and all of them willbe connected by a wire instead of creating flipflops.

Report Task 2

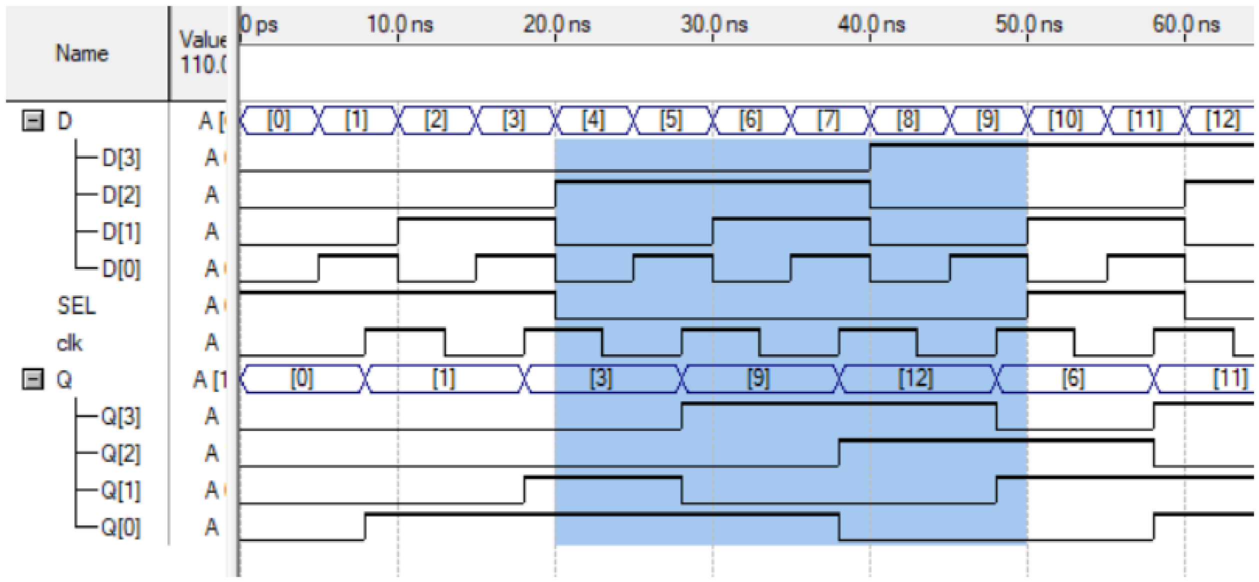
Question: Write the Verilog code of a n-bit end around shift register with provision to load data depending on a control signal ‘sel’ and also provide the waveforms.

Verilog code



```verilog
module shiftreg(D,SEL,clk,Q);
parameter n=4;
input [n-1:0] D;
input SEL,clk;
output reg [n-1:0] Q;

always @(posedge clk)
if(SEL) Q<=D;
else Q[n-1:0]<={Q[0],Q[n-1:1]};
endmodule
```

Waveform:



Discussion:

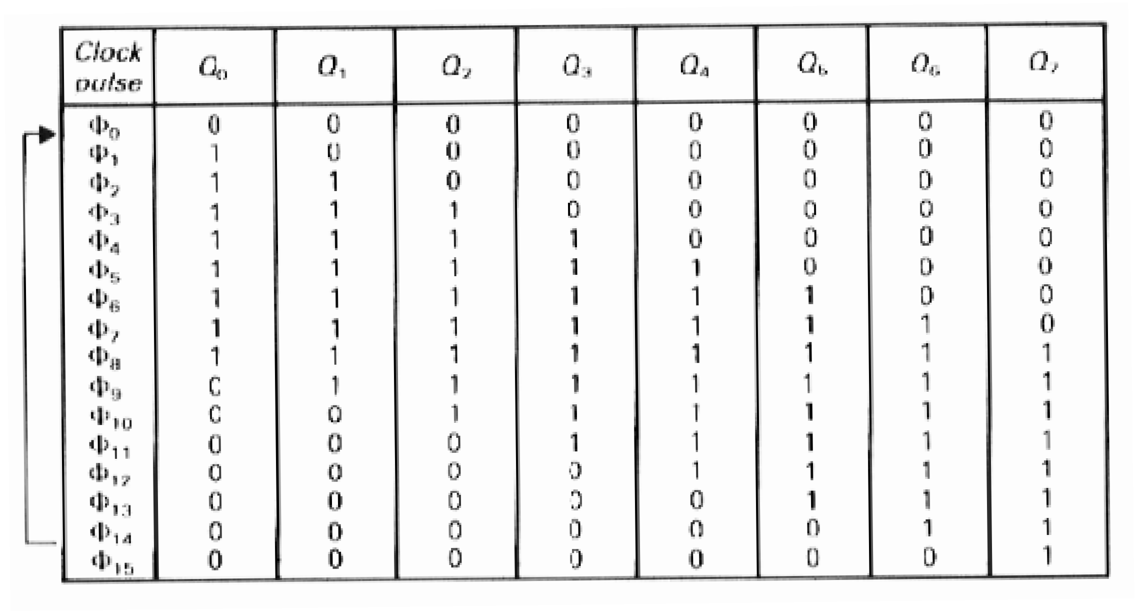
When the select input is high, with the positive edge of the clock pulse, the value from D is set to Q. This is seen in the white portion of the plot. After Q=3 is set, we made the select input zero in the darkened portion. It then right shifts the values by one bit.

If we right shift 3(0011) we get 9(1001), If we right shift 9(1001) we get 12(1100). Right shifting 12(1100) finally outputs 6(0110). The select value afterwards is made 1 again.
Then It starts to imitate the values of D with the clock as it stores 11.

Report Task 3

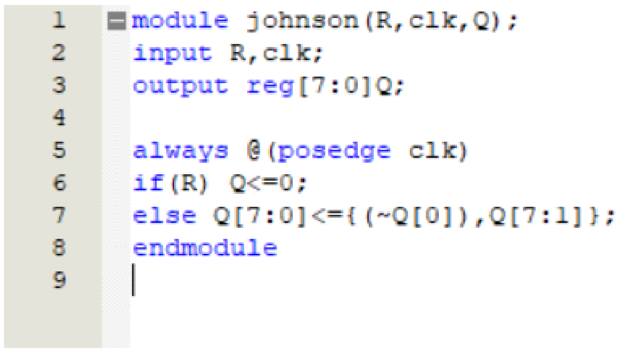
Question: Write Verilog code of an eight-bit Johnson counter and show the timing simulation that shows the counting sequence.

Truth table for 8 bit Johnson counter:



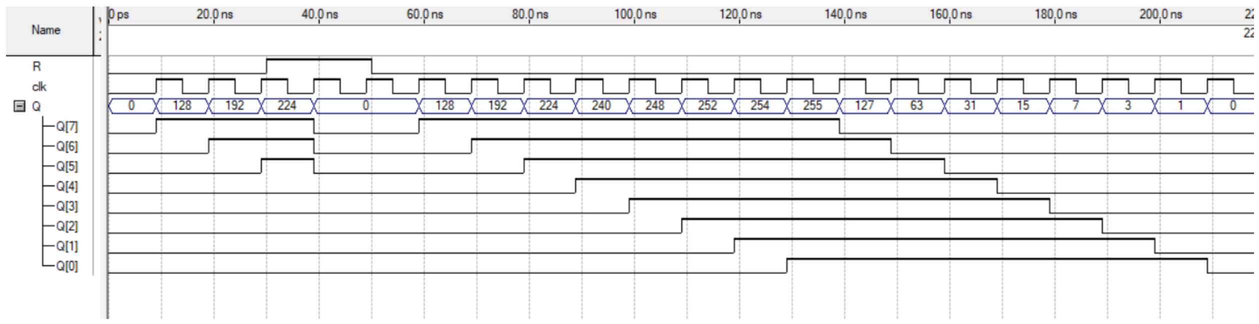
Clock pulse	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
Φ_0	0	0	0	0	0	0	0	0
Φ_1	1	0	0	0	0	0	0	0
Φ_2	1	1	0	0	0	0	0	0
Φ_3	1	1	1	0	0	0	0	0
Φ_4	1	1	1	1	0	0	0	0
Φ_5	1	1	1	1	1	0	0	0
Φ_6	1	1	1	1	1	1	0	0
Φ_7	1	1	1	1	1	1	1	0
Φ_8	1	1	1	1	1	1	1	1
Φ_9	C	1	1	1	1	1	1	1
Φ_{10}	C	0	1	1	1	1	1	1
Φ_{11}	0	0	0	1	1	1	1	1
Φ_{12}	0	0	0	0	1	1	1	1
Φ_{13}	0	0	0	0	0	1	1	1
Φ_{14}	0	0	0	0	0	0	1	1
Φ_{15}	0	0	0	0	0	0	0	1

Verilog code:



```
module johnson(R,clk,Q);
input R,clk;
output reg[7:0]Q;

always @(posedge clk)
if(R) Q<=0;
else Q[7:0]<={(~Q[0]),Q[7:1]};
endmodule

```

Waveform:



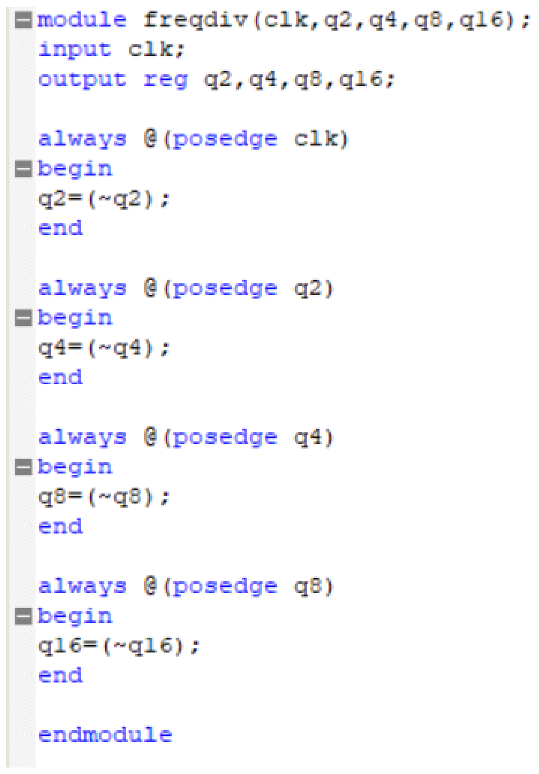
Discussion:

By the nature of the Johnson counter, it shifts one value from the LSB to MSB after toggling it. As expected, the similar nature is shown in the waveform. For the portion when reset is high, all the values are zero for Q. As reset goes down, the counter starts to work with clock pulses.

<u>Report Task 4</u>

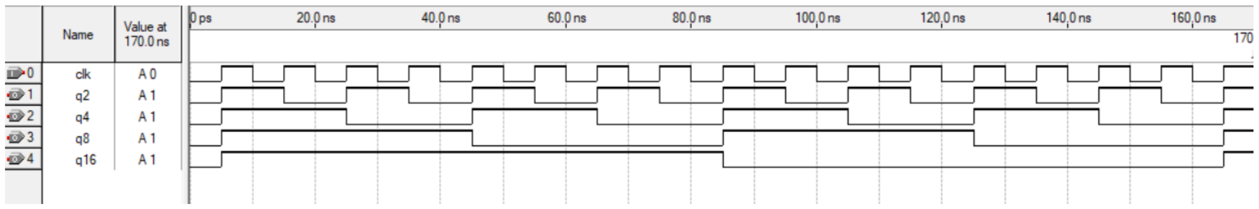
Question: Write Verilog code that will divide the input clock frequency by 16.

Verilog code:



```
module freqdiv(clk,q2,q4,q8,q16);
input clk;
output reg q2,q4,q8,q16;

always @(posedge clk)
begin
q2=(~q2);
end

always @(posedge q2)
begin
q4=(~q4);
end

always @(posedge q4)
begin
q8=(~q8);
end

always @(posedge q8)
begin
q16=(~q16);
end

endmodule
```

Waveform:



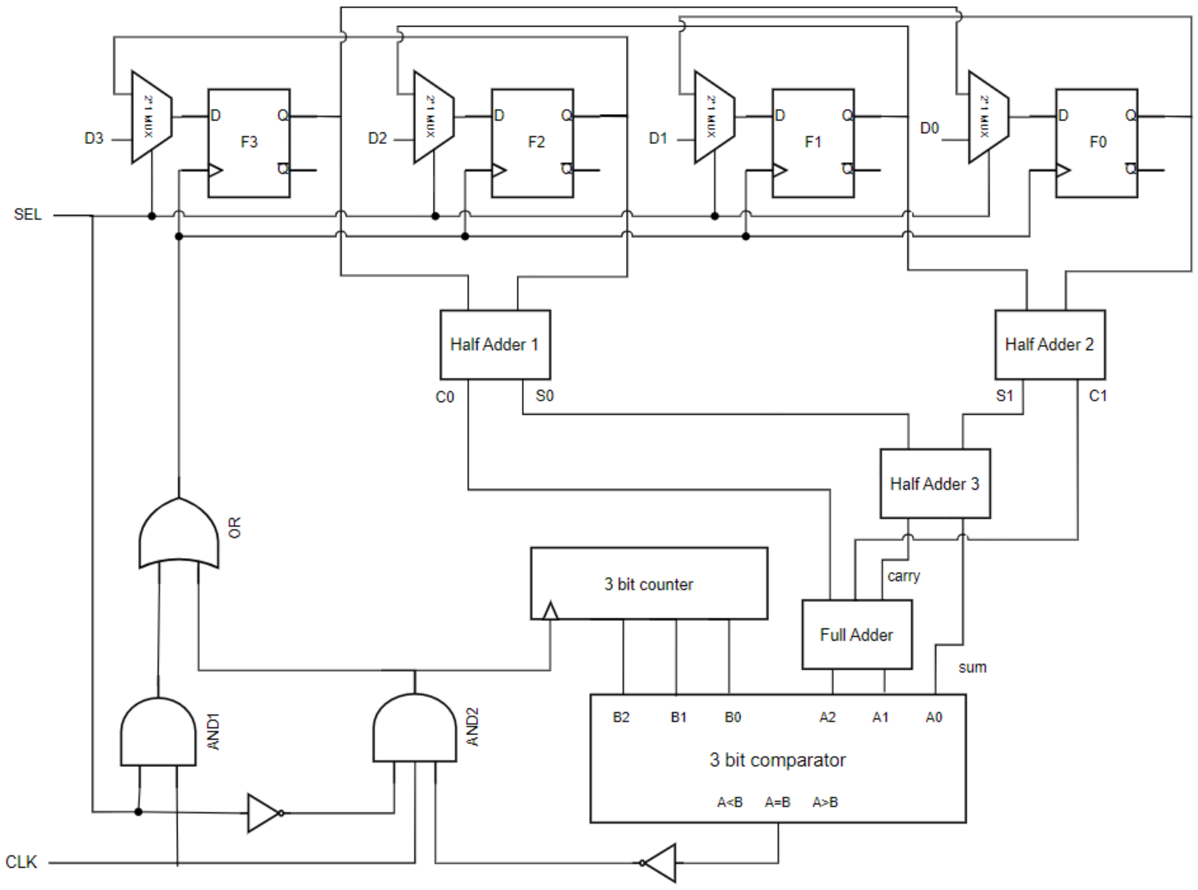
Discussion:

Here we used the positive edge of the signals to trigger a second signal and thus essentially making the frequency half. To achieve 1/16 of frequency, we needed to do the same 4 times. It would have worked the same if we used the negative edge of the clock as well.

Report Task 5

Question: Propose a digital scheme which will detect the number of 1's in a 4-bit number and it will do end around left shift total number of 1s' times. e. g. If the number is 0101, it will do end around left shift twice. If it is 1011, it will end around left shift thrice. This is NOT a Verilog problem. You will have to draw logic circuit.

Circuit diagram:



Discussion:

Here the four flip flops can store data or do a left shift with a clock pulse.

First, to load data, we set SEL=1 This enables the MUX to pass the values D3,D2,D1 and D0. Through the AND1 and OR gate, a clock pulse passes to load these data to the respective flip flops. AND2 gate is a 3 input gate, here having SEL high will prevent the clock pulse passing to the counter. So the count remains zero when we load data.

SEL value is set to low after loading. Immediately, the output from each flip flop is summed to produce a 3 bit binary number (A2 A1 A0) that is essentially the total number of 1's in the four Q values. Half Adder1 sums the values from F3 and F2 while Half Adder2 sums the values from F1 and F0. These then output two two bit values (C0 S0) and (C1 S1).
This is shown in the truth table below

F3	F2	Half Adder 1		F1	F0	Half Adder 2	
Q3	**Q2**	**C0**	**S0**	**Q1**	**Q0**	**C1**	**S1**
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1
1	0	0	1	1	0	0	1
1	1	1	0	1	1	1	0

Afterwards we implemented the Full Adder and Half Adder 3 where carry out from Half adder goes to the Full Adder as Carry in.

Sum 0		Sum 1		Full Adder			Half Adder 3	
C0	**S0**	**C1**	**S1**	**Carry in**	**A2**	**A1**	**A0**	**Carry out**
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	0	0
0	1	0	0	0	0	0	1	0
1	0	0	0	0	0	1	0	0
0	1	0	1	1	0	1	0	1
0	1	1	0	0	0	1	1	0
1	0	0	1	0	0	1	1	0
1	0	1	0	0	1	0	0	0

FInally, the (A2 A1 A0) is set in the 3 bit comparator to compare values coming from the up counter. Up counter gets its clock input from the AND2 gate which passes the clock as long as SEL=0 and (A=B)=0

For example, if we have a total of three 1's in the Q values, three clock pulses will pass which will count three in the up counter. As soon as (A=B)=1, it inputs a zero value in the AND2 gate stopping the fourth clock pulse from passing. Thus the Q values left shift exactly 3 times.