DLD

LAB REPORT

EXPERIMENT 11

Aim: To simulate the logic gates of 8bit shift registers

Tools used: vivado software

Truth table:

Ring counter(4bit):

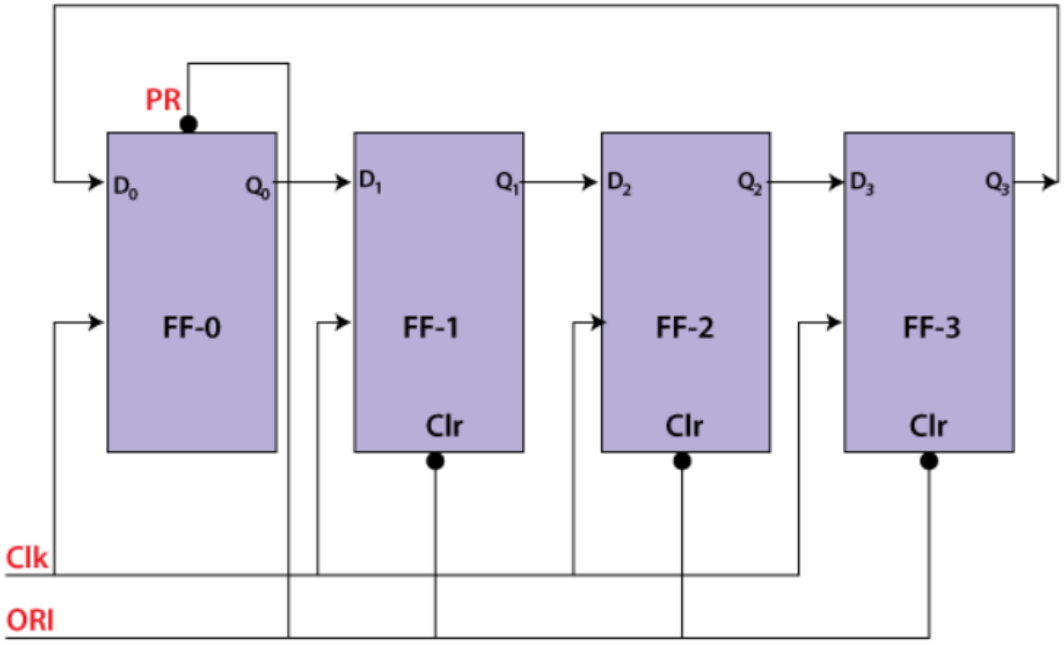
M	Clk	Q3	Q2	Q1	Q0
⌐	X	0	0	0	1
1	↓	0	0	1	0
1	↓	0	1	0	0
1	↓	1	0	0	0
1	↓	0	0	0	1
1	↓	0	0	1	0

Johnson counter(4bit):

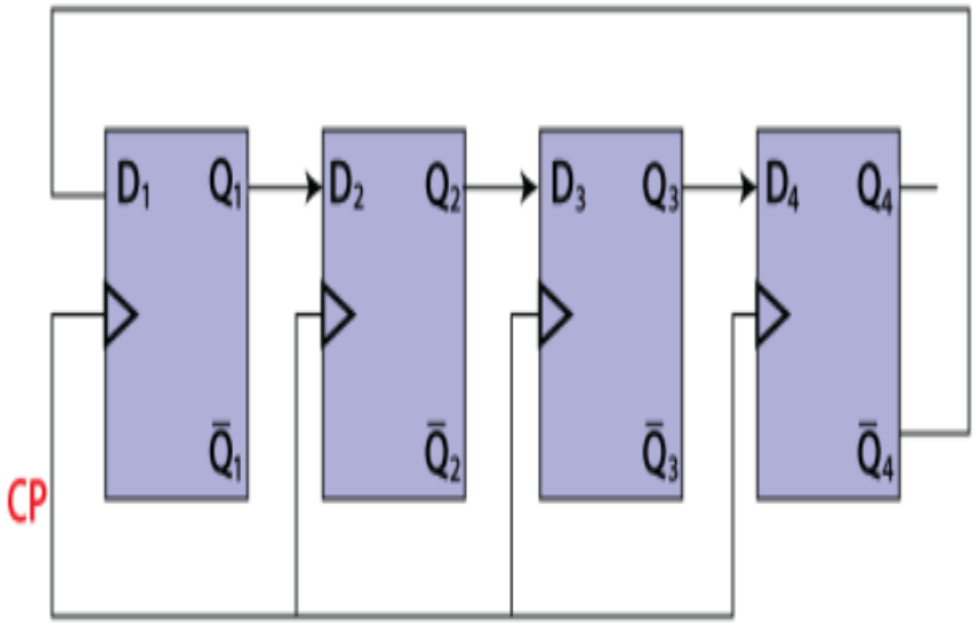
M	Clk	Q3	Q2	Q1	Q0
⌐	X	0	0	0	0
1	↓	0	0	0	1
1	↓	0	0	1	1
1	↓	0	1	1	1
1	↓	1	1	1	1
1	↓	1	1	1	0
1	↓	1	1	0	0
1	↓	1	0	0	0

Diagrams:

Ring counter:



Johnson counter:



Codes:

Ring counter:

```
module ring_counter(clock,reset,count_out);
input clock;
input reset;
output [3:0]count_out;
reg [3:0]count_temp;
always@(negedge(clock),posedge(reset))
begin
if(reset == 1'b1)
begin
 count_temp = 4'b0001;
end
else if(clock == 1'b1)
begin
 count_temp = {count_temp[2:0],count_temp[3]};
end
end
assign count_out = count_temp;
endmodule
```

Testbench code:

```
module tb_ring;
reg clock;
reg reset;
wire [3:0]count_out;
ring_counter e3(clock,reset,count_out);
initial clock = 0;
always #10 clock = ~clock;
initial begin
reset = 1;
#50
reset = 0;
end
endmodule
```

Johnson counter:

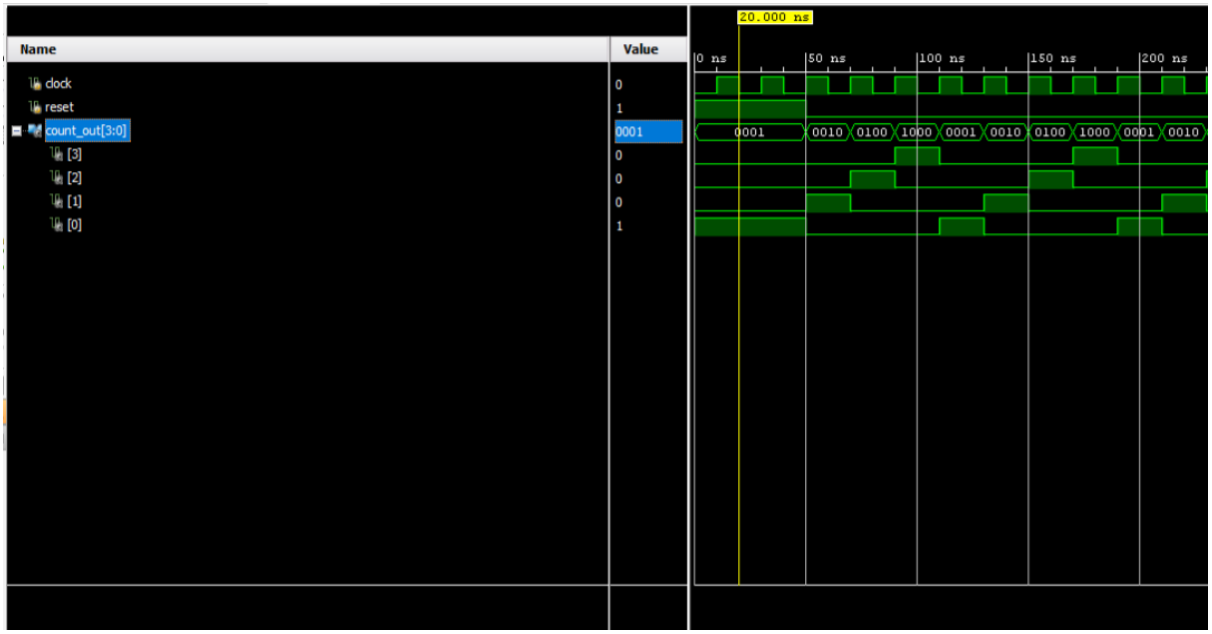
```
23 module johnson_counter(clk,reset,Q);
24 input clk,reset;
25 output [3:0] Q;
26 wire d0,d1,d2,d3;
27 not n1(d0,Q[3]);
28 buf b1(d1,Q[0]);
29 buf b11(d2,Q[1]);
30 buf b12(d3,Q[2]);
31 dff A(clk,reset,d0,Q[0]);
32 dff B(clk,reset,d1,Q[1]);
33 dff C(clk,reset,d2,Q[2]);
34 dff D(clk,reset,d3,Q[3]);
35 endmodule
36 module dff(clk,clr,D,Q);
37 input clk,clr,D;
38 output reg Q;
39 always@(negedge clk,posedge clr)
40 begin
41 if(clr)
42 Q<=0;
43 else
44 Q<=D;
45 end
46 endmodule
```

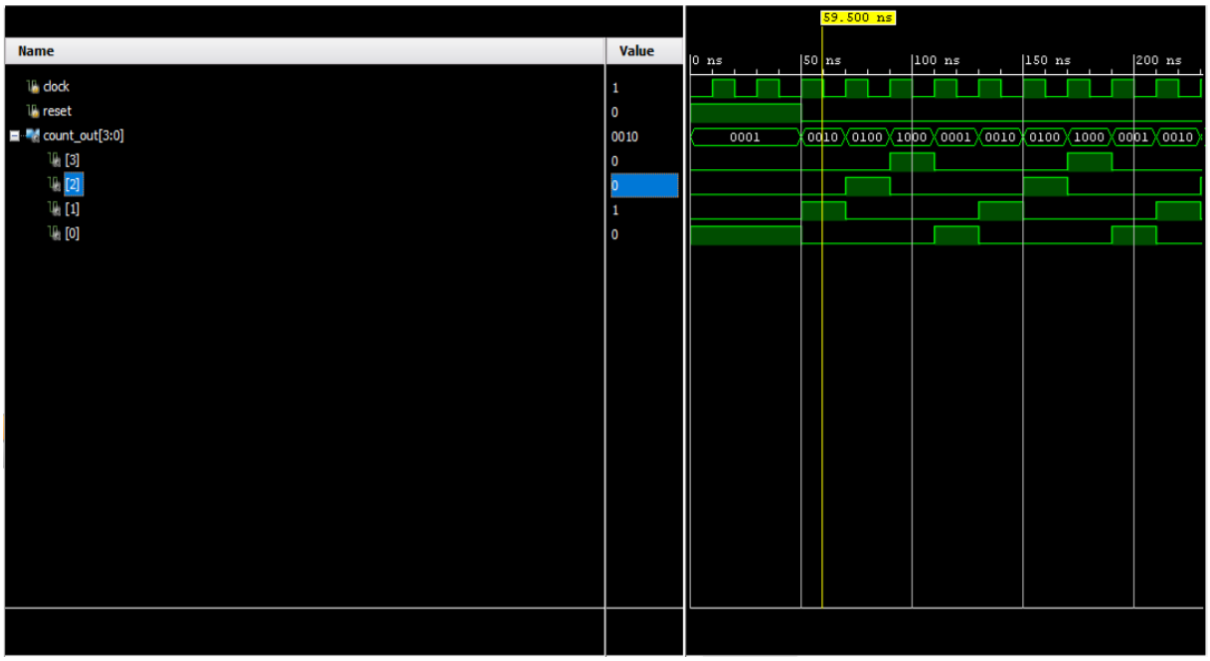
Testbench code:

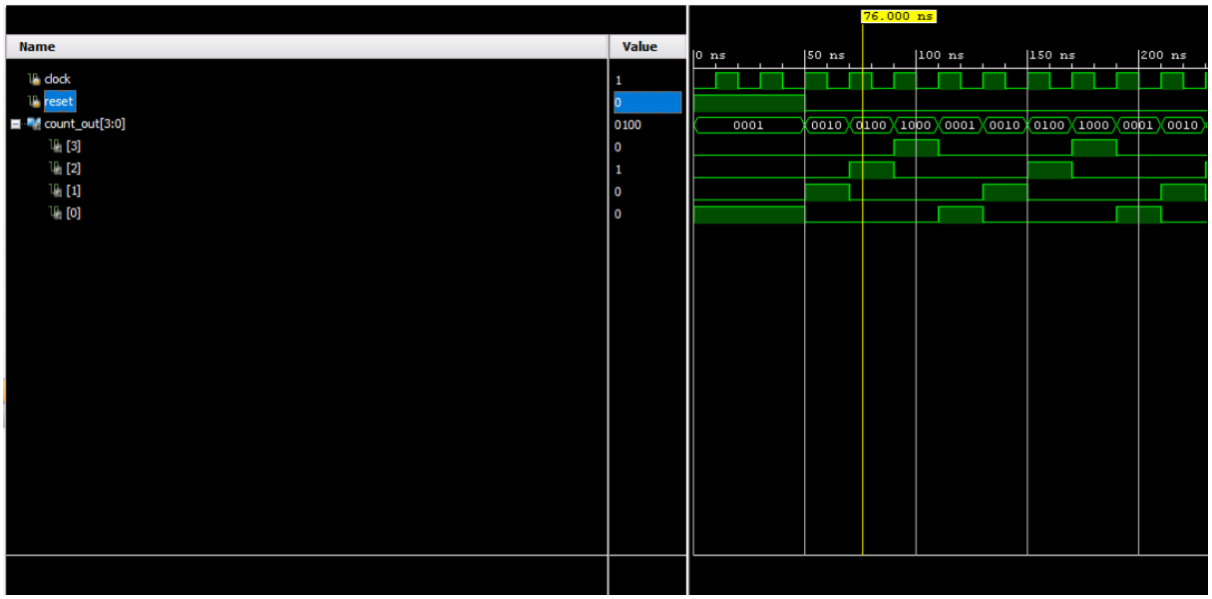
```
23 module tb_johnson_counter;
24 reg clk;
25 reg reset;
26 wire [3:0]Q;
27 johnson_counter_15 e3(clk,reset,Q);
28 initial clk = 0;
29 always #10 clk = ~clk;
30 initial begin
31 reset = 1;
32 #50
33 reset = 0;
34 end
35 endmodule
36
```

Results:

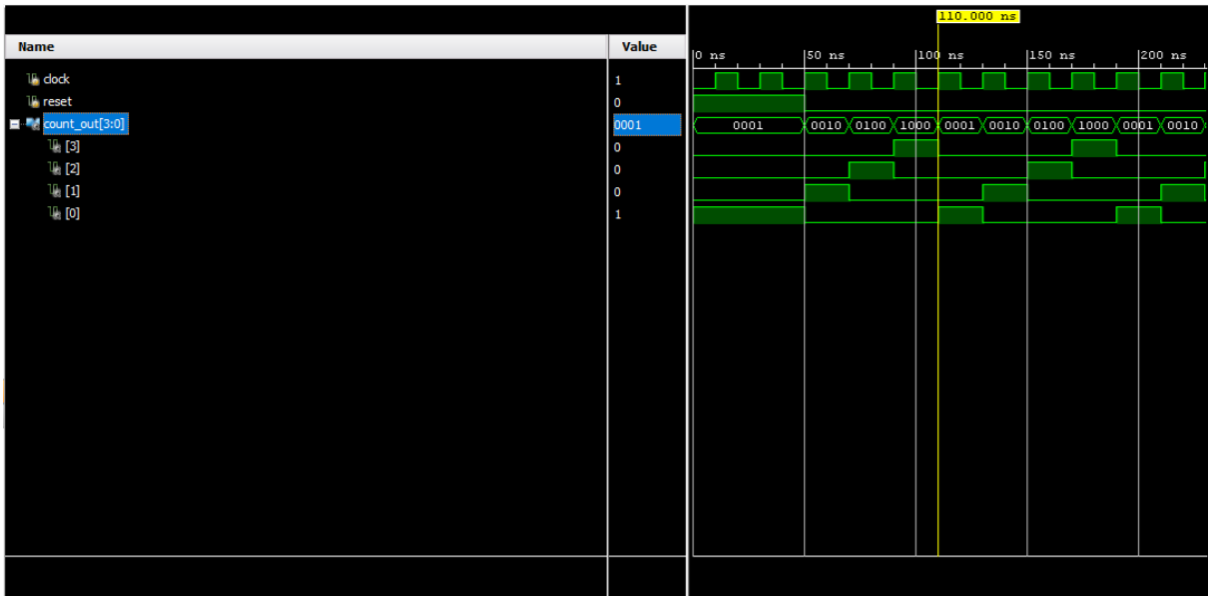
Ring counter 4 bit:

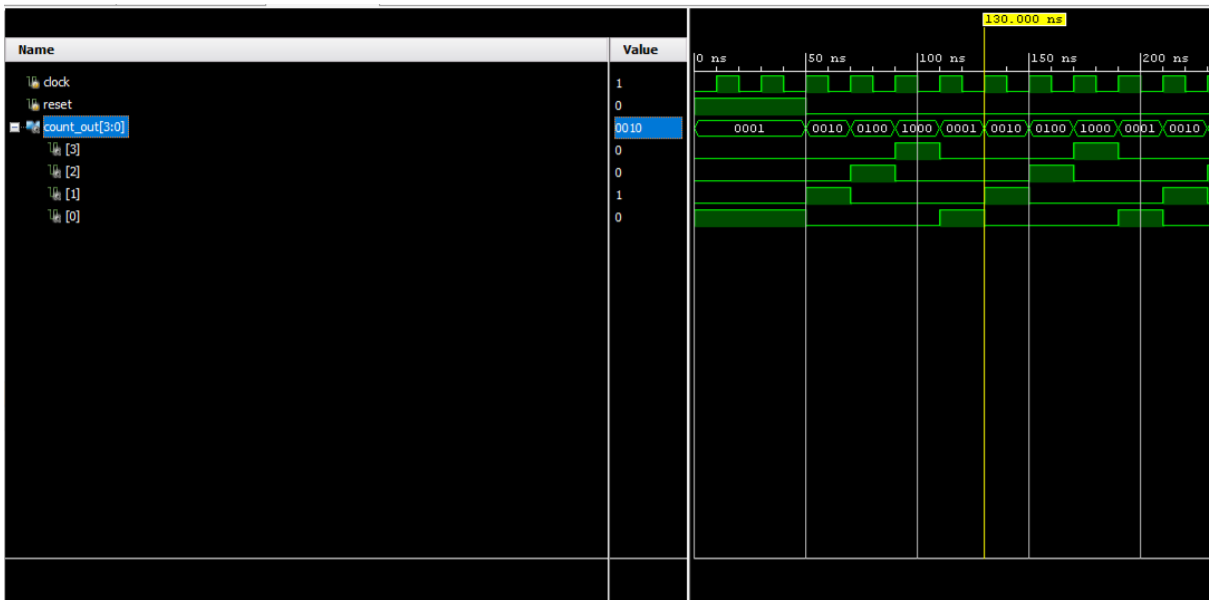




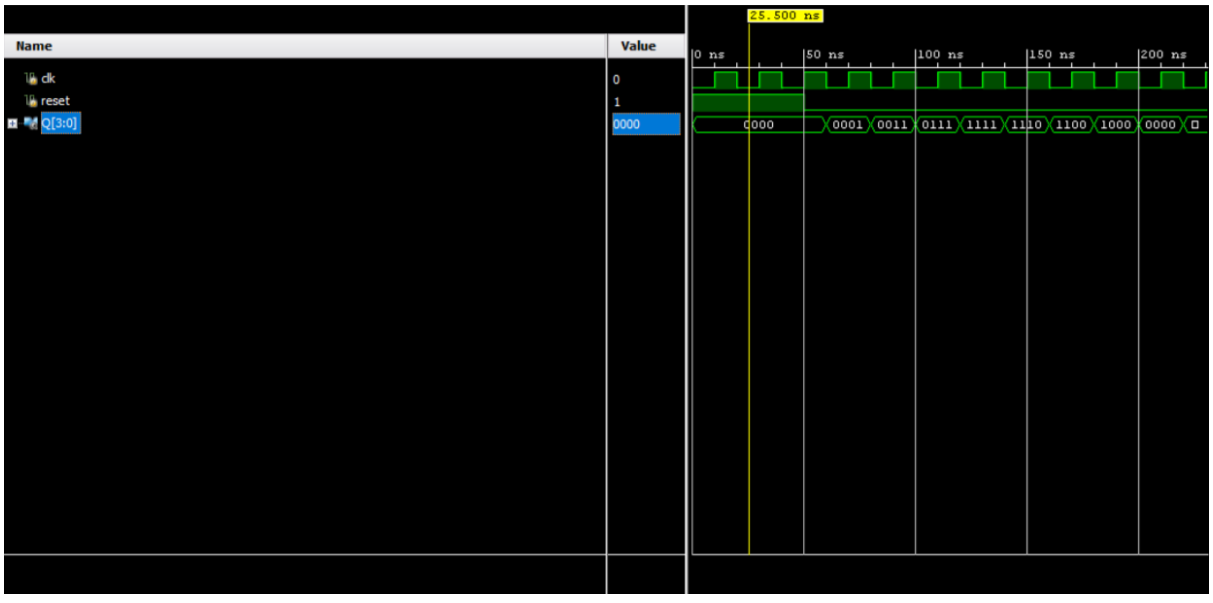

Name
Value
clock
reset
count_out[3:0]
[3]
[2]
[1]
[0]
1
0
0100
0
1
0
0
76.000 ns
0 ns
50 ns
100 ns
150 ns
200 ns
0001
0010
0100
1000
0001
0010
0100
1000
0001
0010


Name
Value
clock
reset
count_out[3:0]
[3]
[2]
[1]
[0]
1
0
1000
1
0
0
0
92.000 ns
0 ns
50 ns
100 ns
150 ns
200 ns
0001
0010
0100
1000
0001
0010
0100
1000
0001
0010


Name
Value
clock
reset
count_out[3:0]
[3]
[2]
[1]
[0]
1
0
0001
0
0
0
1
110.000 ns
0 ns
50 ns
100 ns
150 ns
200 ns
0001
0010
0100
1000
0001
0010
0100
1000
0001
0010

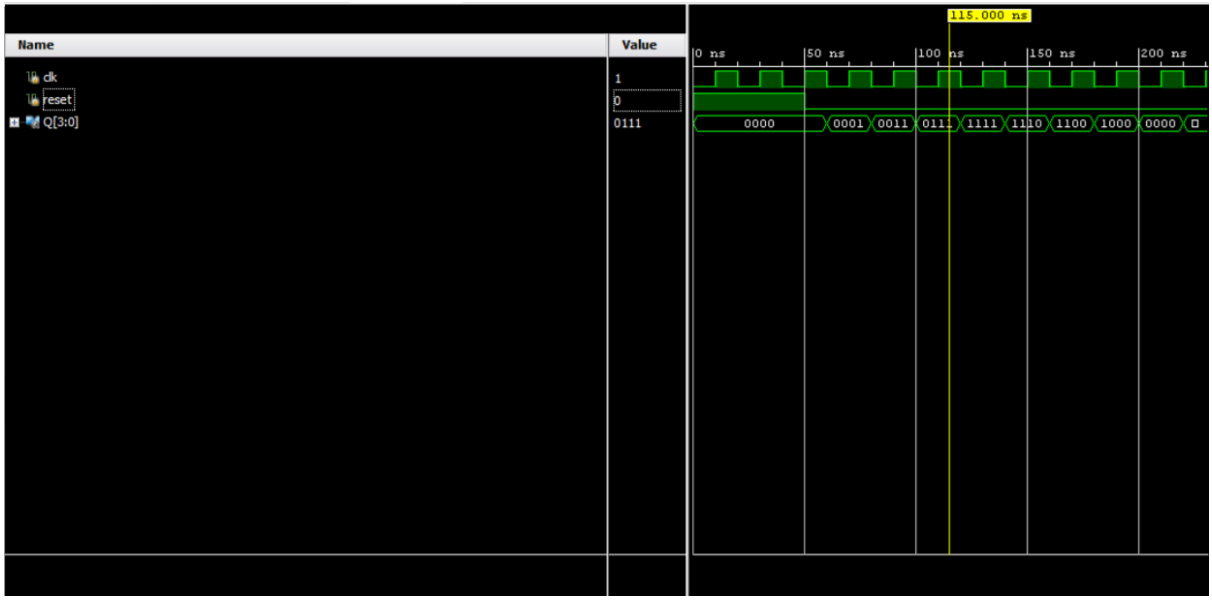


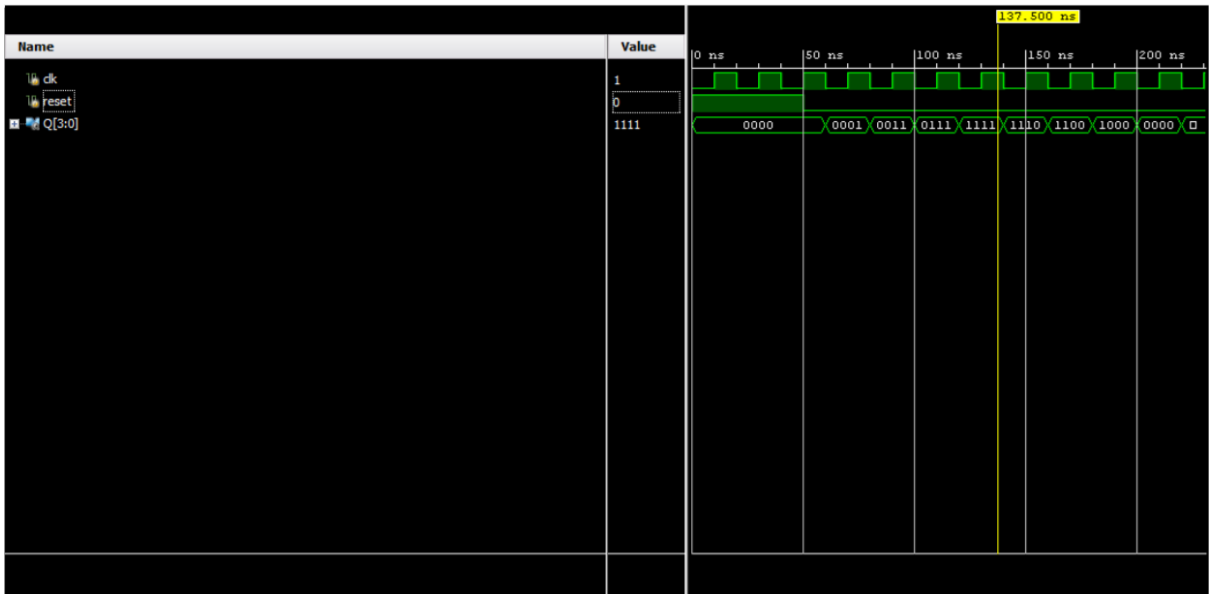
Johnson counter 4bit:

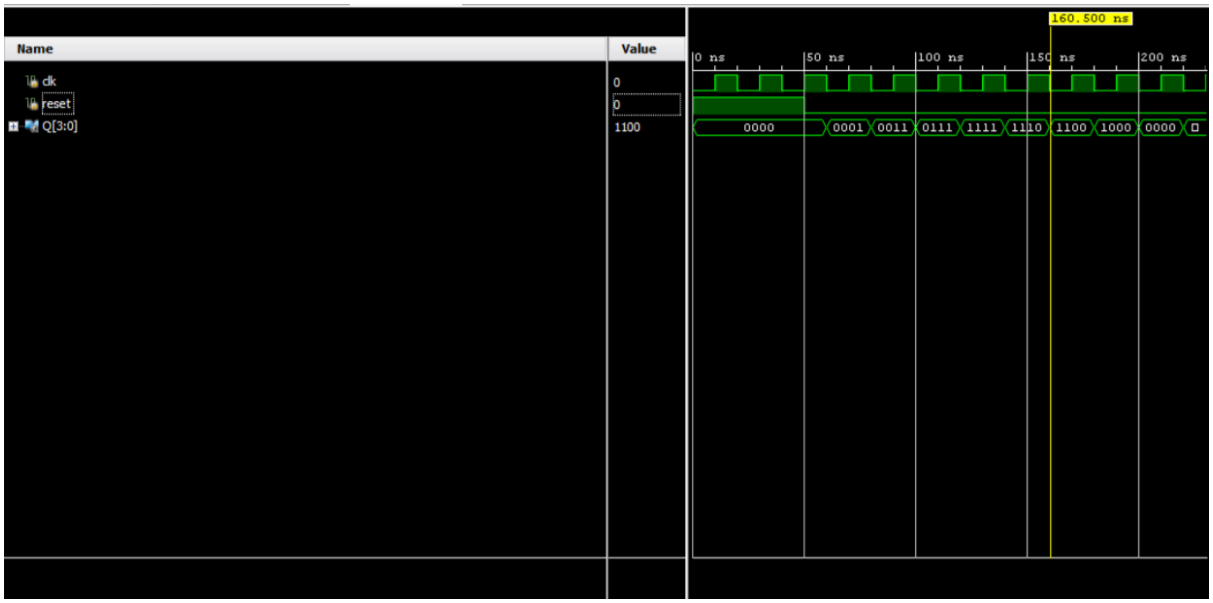


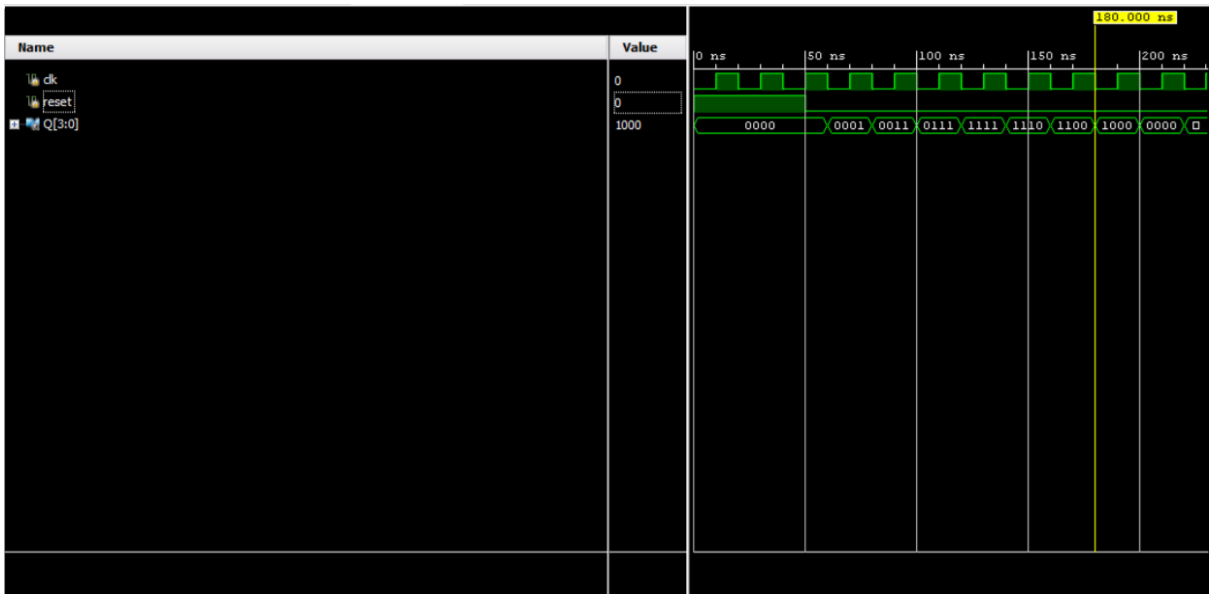

Name
Value
clk
reset
Q[3:0]
1
0
0001
72.000 ns
0 ns
50 ns
100 ns
150 ns
200 ns
0000
0001
0011
0111
1111
1110
1100
1000
0000


Name
Value
clk
reset
Q[3:0]
1
0
0011
91.500 ns
0 ns
50 ns
100 ns
150 ns
200 ns
0000
0001
0011
0111
1111
1110
1100
1000
0000


Name
Value
clk
reset
Q[3:0]
1
0
0111
115.000 ns
0 ns
50 ns
100 ns
150 ns
200 ns
0000
0001
0011
0111
1111
1110
1100
1000
0000


Name
Value
clk
reset
Q[3:0]
1
0
1111
137.500 ns
0 ns
50 ns
100 ns
150 ns
200 ns
0000
0001
0011
0111
1111
1110
1100
1000
0000


Name
Value
clk
reset
Q[3:0]
0
0
1100
160.500 ns
0 ns
50 ns
100 ns
150 ns
200 ns
0000
0001
0011
0111
1111
1110
1100
1000
0000


Name
Value
clk
reset
Q[3:0]
0
0
1000
180.000 ns
0 ns
50 ns
100 ns
150 ns
200 ns
0000
0001
0011
0111
1111
1110
1100
1000
0000

Conclusion:

Now im able to design the logic and the timing diagrams and codes for ring and Jonhson counters (4-bit)