

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI

Course No: EE206P

Name of the Lab: Digital Systems Laboratory

Academic Year/Semester: 2024/B.Tech 3rd Sem

ASSIGNMENT NUMBER: 03

Implementing the FlipFlops, 4-bit serial in and serial out shift register and 4-bit Johnson counter using HDL Language

P MADHU SAI NAIK

EE23B039

PRATHAM C

EE23B041

Instructor:

Dr. Vikramkumar Pudi

Lab Assistant:

Mr. Kumar Bellikatti

11-Nov-2024



Contents

1 Problem Statement	2
2 Objectives of the Experiment	4
3 Design Approach	4
4 Implementation	5
5 Simulation Results and Discussions	21
6 Conclusion	32

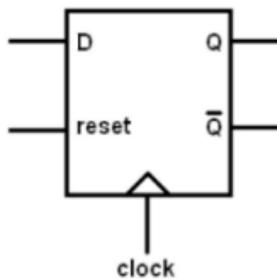
1 Problem Statement

The Problem Statements

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI
Digital Systems Lab (EE206P)
Assignment-03

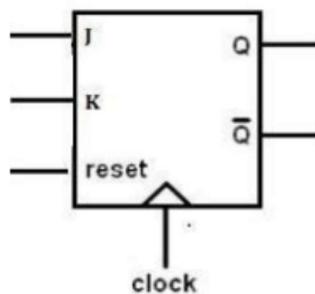
1. Write a Verilog HDL code for the D flip flop for the following options given below
 - A. Without reset
 - B. With asynchronous reset
 - C. With synchronous reset

Verify the functionality of your implementation by simulating it using a test bench.



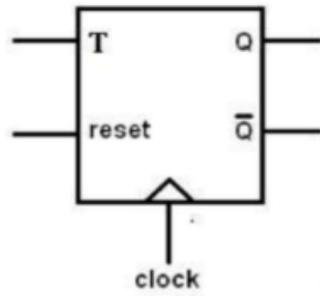
2. Write a Verilog HDL code for the JK flip flop for the following options given below
 - A. Without reset
 - B. With asynchronous reset
 - C. With synchronous reset

Verify the functionality of your implementation by simulating it using a test bench.

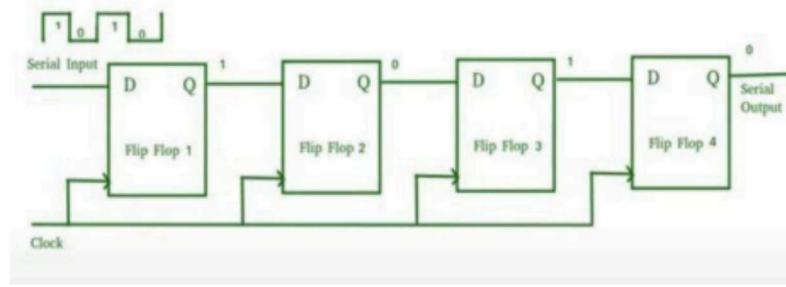


3. Write a Verilog HDL code for the T flip flop for the following options given below
 - A. Without reset
 - B. with asynchronous reset
 - C. With synchronous reset

Verify the functionality of your implementation by simulating it using a test bench.

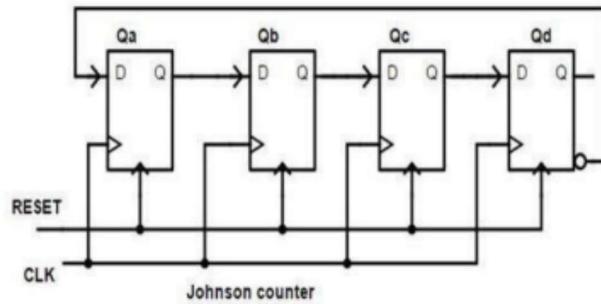


4. Write a Verilog HDL code for the 4-bit serial in and serial out shift register for transferring the input data which is shown in the figure and Verify the functionality of your implementation by simulating it using a test bench.



5. Write a Verilog HDL code for the 4-bit Johnson counter which is shown in the figure and Verify the functionality of your implementation by simulating it using a test bench.

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			



2 Objectives of the Experiment

- Verilog HDL code for D flip flop in 3 different conditions.
- Verilog HDL code for JK flip flop in 3 different conditions.
- Verilog HDL code for T flip flop in 3 different conditions.
- Verilog HDL code for 4 bit serial in and serial out shift register.
- Verilog HDL code for 4 bit Johnson counter.

3 Design Approach

To create a digital circuit model with various flip-flops and counters, along with corresponding test benches, the following components are required:

- D Flip-Flop: A model of a D flip-flop that captures data on the clock's rising edge.
- JK Flip-Flop: A JK flip-flop model with standard behavior based on J, K, and clock inputs.
- T Flip-Flop: A T flip-flop model that toggles its state with each clock pulse.
- 4-bit Serial In, Serial Out Shift Register: A 4-bit shift register that allows serial data input and output.
- 4-bit Johnson Counter: A 4-bit Johnson counter, also known as a twisted ring counter.

4 Implementation

I a) D Flip Flop without reset

```
1 module d_ff_without_r(clk,D,Q,Qb);
2   input clk;
3   input D;
4   output reg Q;
5   output Qb;
6
7   always@(posedge clk)
8   begin
9
10  Q <= D;
11
12 end
13
14 assign Qb = ~Q;
15
16 endmodule
```

TEST BENCH:

```
1 module TB_d_ff_without_r();
2   reg clk,D;
3   wire Q,Qb;
4   d_ff_without_r uut (.clk(clk),.D(D),.Q(Q),.Qb(Qb));
5   initial
6   begin
7     clk = 1'b0;
8     D = 1;
9     repeat(10)
10    begin
11      #5
12      D = $random;
13    end
14  end
15  always
16  #5
17  clk = ~clk;
18 endmodule
```

b) D Flip Flop with asynchronous reset

```
1 module d_ff_with_ar(clk,D,Q,Qb,rst);
2 input clk;
3 input D;
4 input rst;
5 output reg Q;
6 output Qb;
7
8
9 always@(posedge clk or rst)
10
11 begin
12
13 if(rst == 1)
14     Q <= 1'b0;
15 else
16     Q <= D;
17
18 end
19
20 assign Qb = ~Q;
21
22 endmodule
```

TEST BENCH:

```
1 module TB_d_ff_with_as();
2 reg clk;
3 reg D;
4 reg rst;
5 wire Q,Qb;
6
7 d_ff_with_ar uut (.clk(clk),.D(D),.Q(Q),.Qb(Qb),.rst(rst));
8
9 initial
10 begin
11     clk = 1'b0;
12     D = 1;
13     rst = 1'b0;
14
```

```

15 || repeat(10)
16 begin
17 #5
18 D = $random;
19 end
20
21 end
22
23 always
24 #5
25 clk = ~clk;
26 always
27 #5
28 rst = ~rst;
29
30 endmodule

```

c) D Flip Flop with synchronous reset

```

1 module d_ff_with_sr(clk,D,Q,Qb,rst);
2 input clk;
3 input D;
4 input rst;
5 output reg Q;
6 output Qb;
7
8
9 always@(posedge clk)
10 begin
11
12 if(rst == 1)
13   Q <= 1'b0;
14 else
15   Q <= D;
16
17 end
18
19 assign Qb = ~Q;
20
21 endmodule

```

TEST BENCH:

```
1 module TB_d_ff_with_as();
2 reg clk;
3 reg D;
4 reg rst;
5 wire Q,Qb;
6
7 d_ff_with_sr uut (.clk(clk),.D(D),.Q(Q),.Qb(Qb),.rst(rst));
8
9 initial
10 begin
11 clk = 1'b0;
12 D =1;
13 rst = 1'b0;
14
15 repeat(10)
16 begin
17 #5
18 D = $random;
19 end
20
21 end
22
23 always
24 #5
25 clk = ~clk;
26 always
27 #5
28 rst = ~rst;
29
30 endmodule
```

II a) JK Flip Flop without reset

```
1 module jk_without_r(clk, J, K, Q, Qb);
2   input clk;
3   input J;
4   input K;
5   output reg Q;
6   output Qb;
7
8   always @ (posedge clk) begin
9     if (J==0 && K==0)
10       Q <= Q;
11     else if (J==0 && K==1)
12       Q <= 0;
13     else if (J==1 & K==0)
14       Q <= 1;
15     else
16       Q <= ~Q;
17   end
18
19   assign Qb = ~Q;
20 endmodule
```

TEST BENCH:

```
1 module TB_jk_without_r();
2   reg clk;
3   reg J;
4   reg K;
5   wire Q;
6   wire Qb;
7
8   jk_without_r uut (.clk(clk), .J(J), .K(K), .Q(Q), .Qb(Qb));
9
10 initial begin
11   clk = 1'b0;
12   J = 1'b0;
13   K = 1'b0;
14
15   #10 J = 1'b1; K = 1'b0;
16   #10 J = 1'b0; K = 1'b1;
17   #10 J = 1'b1; K = 1'b1;
18   #10 J = 1'b0; K = 1'b0;
19   #10 J = 1'b1; K = 1'b0;
20   #10 J = 1'b1; K = 1'b1;
21   #10 J = 1'b0; K = 1'b1;
22   #10 J = 1'b0; K = 1'b0;
23 end
24
25 always #5 clk = ~clk;
26 endmodule
```

b) JK Flip Flop with Asynchronous reset

```
1 module jk_with_ar(clk, J, K, Q, Qb, rst);
2   input clk;
3   input J;
4   input K;
5   input rst;
6   output reg Q;
7   output Qb;
8
9   always @(posedge clk or rst)
10 begin
```

```

11      if(rst == 1)
12          Q <= 1'b0;
13      else
14          if (J==0 && K==0)
15              Q <= Q;
16          else if (J==0 && K==1)
17              Q <= 0;
18          else if (J==1 & K==0)
19              Q <= 1;
20          else
21              Q <= ~Q;
22      end
23
24      assign Qb = ~Q;
25 endmodule

```

TEST BENCH:

```

1 module TB_jk_with_ar();
2     reg clk;
3     reg J;
4     reg K;
5     reg rst;
6     wire Q;
7     wire Qb;
8
9     jk_with_ar uut (.clk(clk), .J(J), .K(K), .Q(Q), .Qb(Qb), .rst(rst))
10    ;
11
11 initial begin
12     clk = 1'b0;
13     rst = 1'b1;
14     J = 1'b0;
15     K = 1'b0;
16
17     #10 rst = 1'b0;
18     #10 J = 1'b1; K = 1'b0;
19     #10 J = 1'b0; K = 1'b1;
20     #10 J = 1'b1; K = 1'b1;
21     #10 J = 1'b0; K = 1'b0;

```

```

22      #10 J = 1'b1; K = 1'b0;
23      #10 J = 1'b1; K = 1'b1;
24      #10 J = 1'b0; K = 1'b1;
25      #10 rst = 1'b1;
26      #10 rst = 1'b0; J = 1'b1; K = 1'b1;
27  end
28
29  always #5 clk = ~clk;
30 endmodule

```

c) JK Flip Flop with synchronous reset

```

1 module jk_with_sr(clk, J, K, Q, Qb, rst);
2   input clk;
3   input J;
4   input K;
5   input rst;
6   output reg Q;
7   output Qb;
8
9   always @ (posedge clk)
10 begin
11   if(rst == 1)
12     Q <= 1'b0;
13   else
14     if (J==0 && K==0)
15       Q <= Q;
16     else if (J==0 && K==1)
17       Q <= 0;
18     else if (J==1 & K==0)
19       Q <= 1;
20     else
21       Q <= ~Q;
22   end
23
24   assign Qb = ~Q;
25 endmodule

```

TESTBENCH:

```
1 module TB_jk_with_sr();
2     reg clk;
3     reg J;
4     reg K;
5     reg rst;
6     wire Q;
7     wire Qb;
8
9     jk_with_sr uut (.clk(clk), .J(J), .K(K), .Q(Q), .Qb(Qb), .rst(rst))
10    ;
11
12 initial begin
13     clk = 1'b0;
14     rst = 1'b1;
15     J = 1'b0;
16     K = 1'b0;
17
18     #10 rst = 1'b0;
19     #10 J = 1'b1; K = 1'b0;
20     #10 J = 1'b0; K = 1'b1;
21     #10 J = 1'b1; K = 1'b1;
22     #10 J = 1'b0; K = 1'b0;
23     #10 J = 1'b1; K = 1'b0;
24     #10 J = 1'b1; K = 1'b1;
25     #10 J = 1'b0; K = 1'b1;
26     #10 rst = 1'b1;
27     #10 rst = 1'b0; J = 1'b1; K = 1'b1;
28
29     always #5 clk = ~clk;
30 endmodule
```

III a) T FlipFlop Without reset

```
1 module t_without_r(clk,T,Q,Qb);
2   input clk;
3   input T;
4   output reg Q;
5   output Qb;
6
7   always@(posedge clk)
8   begin
9     Q = T^Q;
10  end
11 assign Qb = ~Q;
12 endmodule
```

TEST BENCH:

```
1 module TB_t_without_r();
2   reg clk;
3   reg T;
4   wire Q;
5   wire Qb;
6
7   t_without_r uut (.clk(clk), .T(T), .Q(Q), .Qb(Qb));
8
9   initial begin
10   clk = 1'b0;
11   T = 1'b0;
12
13   repeat(10) begin
14     #10 T = $random;
15   end
16 end
17
18 always #5 clk = ~clk;
19 endmodule
```

b) T FlipFlop with Asynchronous reset

```
1 module t_with_ar(clk,T,Q,Qb,rst);
2 input clk;
3 input T;
4 input rst;
5 output reg Q;
6 output Qb;
7
8 always@(posedge clk or rst)
9 begin
10 if(rst == 1)
11 Q = 1'b0;
12 else
13 Q = T^Q;
14 end
15 assign Qb = ~Q;
16 endmodule
```

TEST BENCH:

```
1 module TB_t_with_ar();
2     reg clk;
3     reg T;
4     reg rst;
5     wire Q;
6     wire Qb;
7
8     t_with_ar uut (.clk(clk), .T(T), .Q(Q), .Qb(Qb), .rst(rst));
9
10 initial begin
11     clk = 1'b0;
12     rst = 1'b1;
13     T = 1'b0;
14
15     #10 rst = 1'b0;
16     repeat(10) begin
17         #10 T = $random;
18     end
19     #10 rst = 1'b1;
20     #10 rst = 1'b0;
```

```

21         repeat(5) begin
22             #10 T = $random;
23         end
24     end
25
26     always #5 clk = ~clk;
27 endmodule

```

c) T Flip Flop with Synchronous reset

```

1 module t_with_sr(clk,T,Q,Qb,rst);
2 input clk;
3 input T;
4 input rst;
5 output reg Q;
6 output Qb;
7
8 always@(posedge clk)
9 begin
10 if(rst == 1)
11 Q = 1'b0;
12 else
13 Q = T^Q;
14 end
15 assign Qb = ~Q;
16 endmodule

```

TEST BENCH:

```

1 module TB_t_with_sr();
2     reg clk;
3     reg T;
4     reg rst;
5     wire Q;
6     wire Qb;
7
8     t_with_sr uut (.clk(clk), .T(T), .Q(Q), .Qb(Qb), .rst(rst));
9
10 initial begin
11     clk = 1'b0;
12     rst = 1'b1;

```

```

13      T = 1'b0;
14
15      #10 rst = 1'b0;
16      repeat(10) begin
17          #10 T = $random;
18      end
19      #10 rst = 1'b1;
20      #10 rst = 1'b0;
21      repeat(5) begin
22          #10 T = $random;
23      end
24  end
25
26  always #5 clk = ~clk;
27 endmodule

```

IV 4-bit serial in and serial out shift register

```

1 module siso_shift_register_4bit (
2     input wire clk,
3     input wire serial_in,
4     output wire serial_out
5 );
6
7 reg [3:0] shift_reg;
8
9 initial begin
10     shift_reg = 4'b1010;
11 end
12
13 always @(posedge clk) begin
14     shift_reg <= {serial_in, shift_reg[3:1]};
15 end
16
17 assign serial_out = shift_reg[0];
18
19 endmodule

```

TEST BENCH:

```
1 module tb_siso_shift_register_4bit();
2
3     reg clk;
4     reg serial_in;
5     wire serial_out;
6
7     siso_shift_register_4bit uut (
8         .clk(clk),
9         .serial_in(serial_in),
10        .serial_out(serial_out)
11    );
12
13     initial begin
14         clk = 0;
15         forever #10 clk = ~clk;
16     end
17
18     initial begin
19         #20;
20         repeat(10) begin
21             serial_in = $random % 2;
22             #20;
23         end
24         #100;
25         $finish;
26     end
27
28     initial begin
29         $monitor("Time = %0d, serial_in = %b, serial_out = %b,
30                 shift_reg = %b", $time, serial_in, serial_out, uut.shift_reg
31 );
32 endmodule
```

V 4-bit Johnson counter

```
1 module JohnsonCounter (
2     input wire clk,
3     input wire reset,
4     output reg [3:0] Q
5 );
6
7     always @ (posedge clk or posedge reset) begin
8         if (reset)
9             Q <= 4'b0000;
10        else
11            Q <= {~Q[0], Q[3:1]};
12    end
13
14 endmodule
```

TEST BENCH:

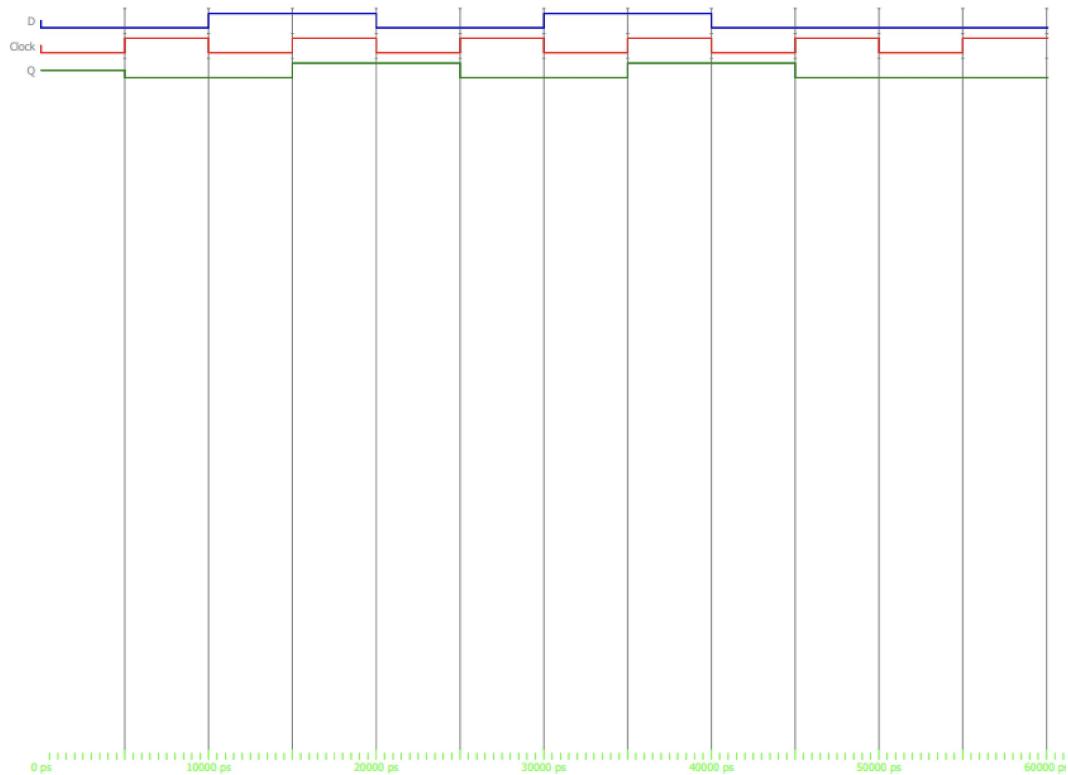
```
1 module tb_JohnsonCounter();
2
3     reg clk;
4     reg reset;
5     wire [3:0] Q;
6
7     JohnsonCounter uut (
8         .clk(clk),
9         .reset(reset),
10        .Q(Q)
11    );
12
13     initial begin
14         clk = 0;
15         forever #10 clk = ~clk;
16     end
17
18     initial begin
19         reset = 1;
20         #20;
21         reset = 0;
22         repeat(10) begin
```

```
23         reset = $random % 2;
24         #20;
25     end
26     #100;
27     $finish;
28 end
29
30 initial begin
31     $monitor("Time = %0d, reset = %b, Q = %b", $time, reset, Q);
32 end
33
34 endmodule
```

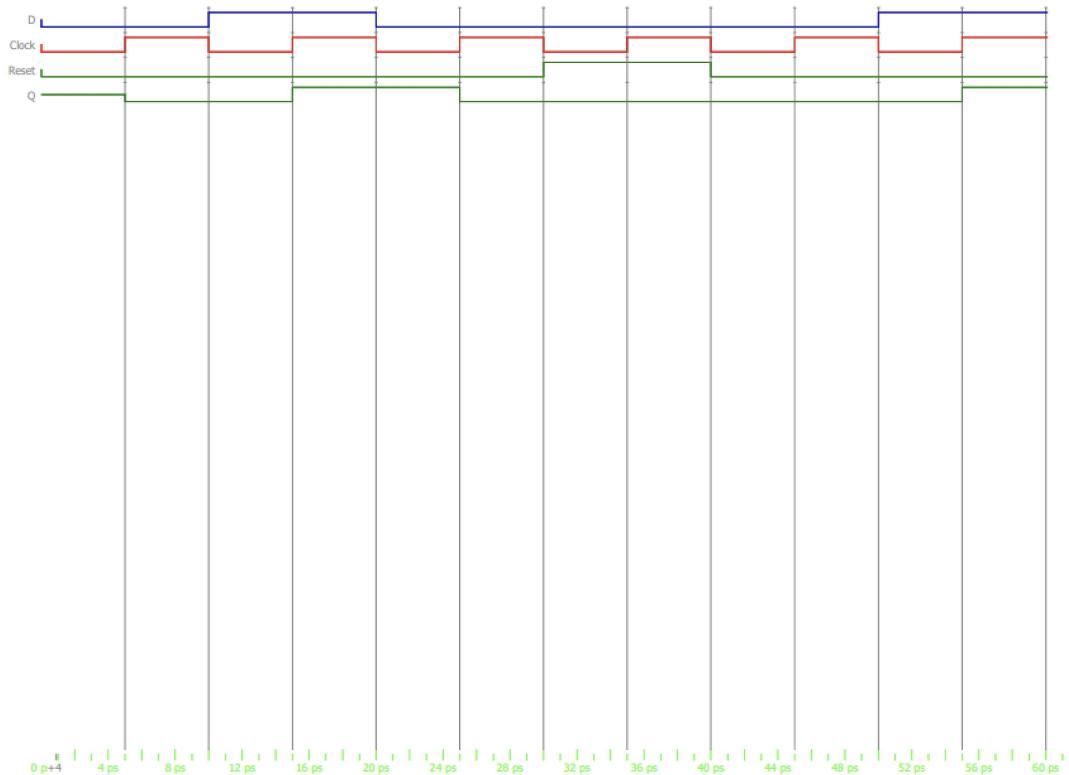
5 Simulation Results and Discussions

THE WAVEFORMS OF THE ABOVE HDL CODES

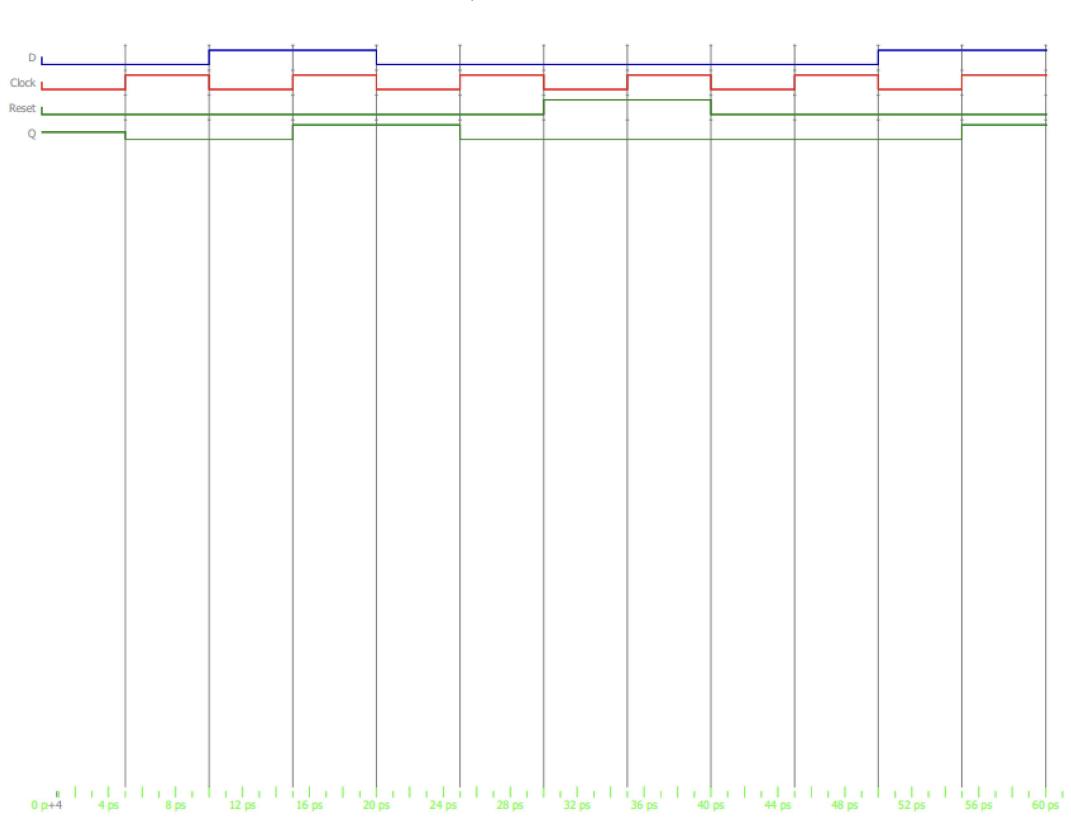
- D without reset



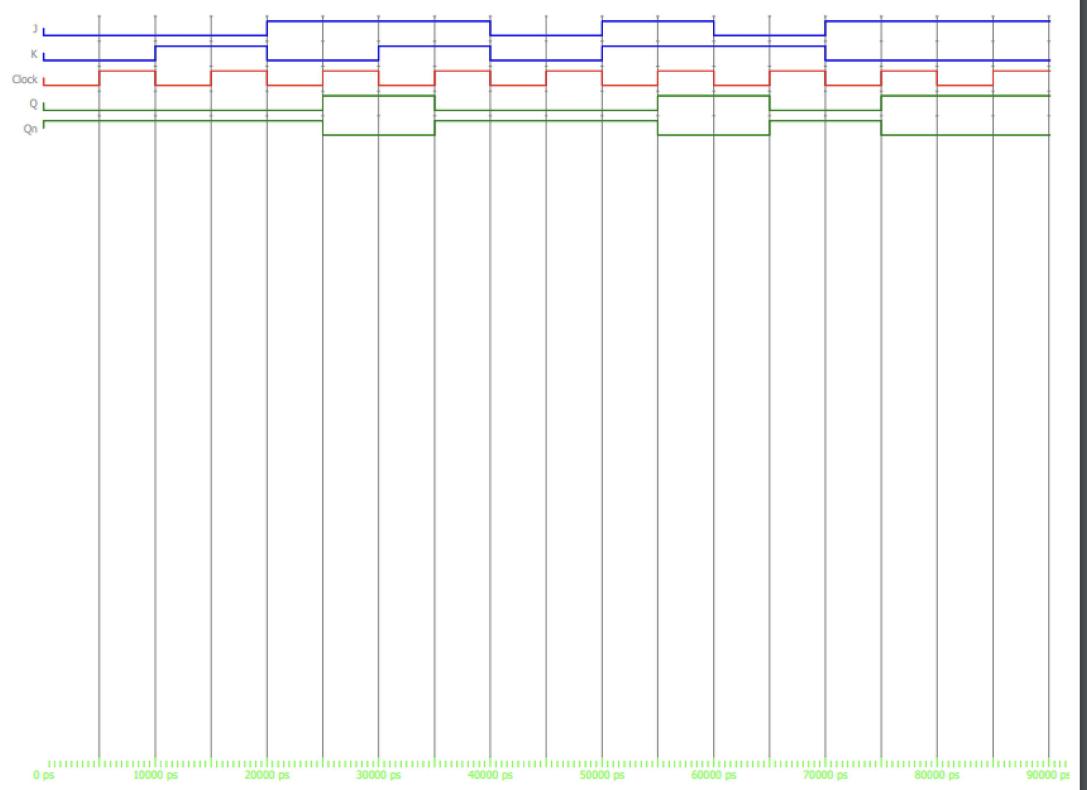
- D with Asynchronous reset



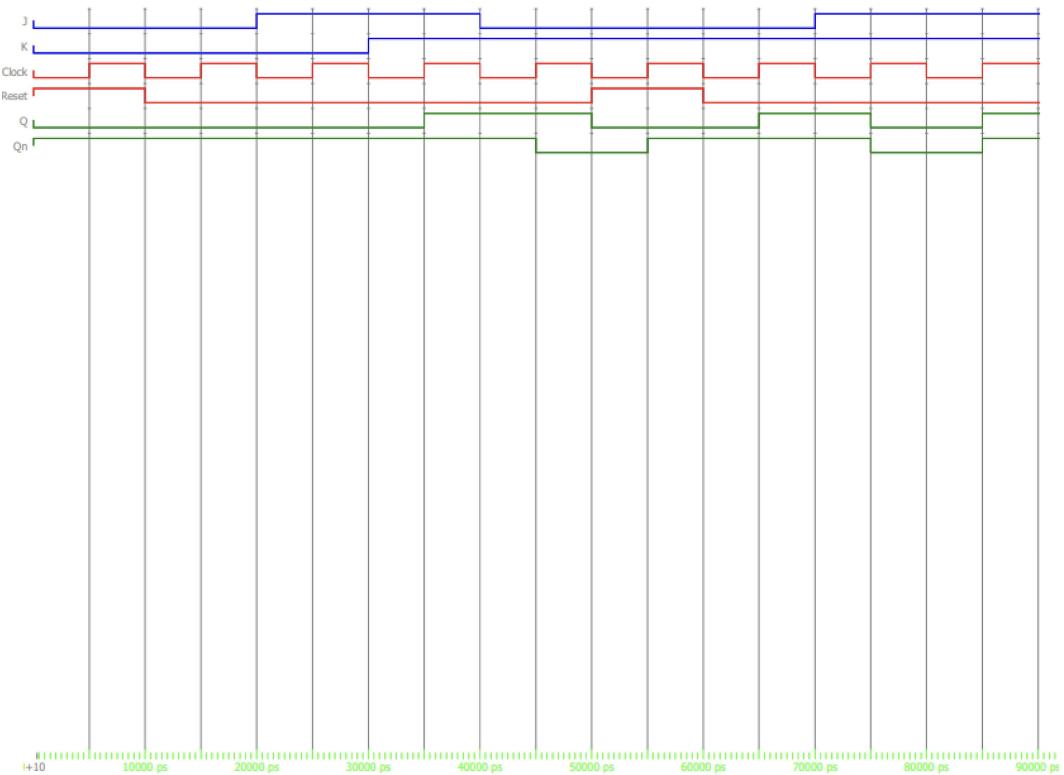
• D with Synchronous reset



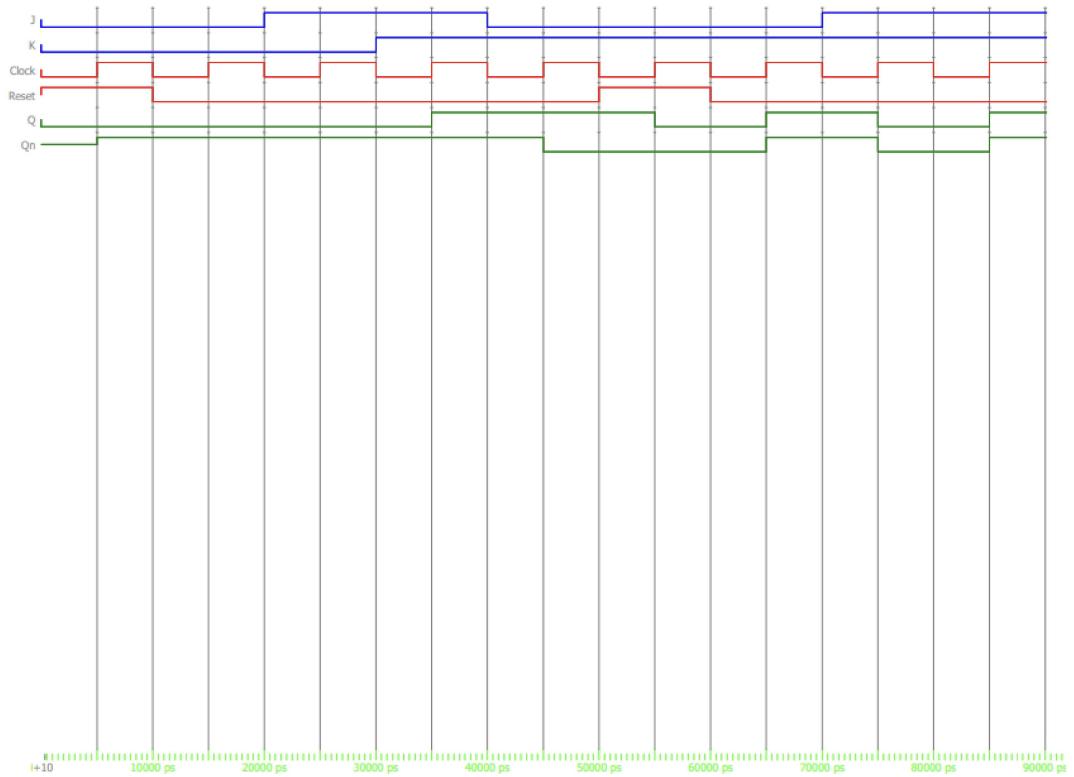
• JK without reset



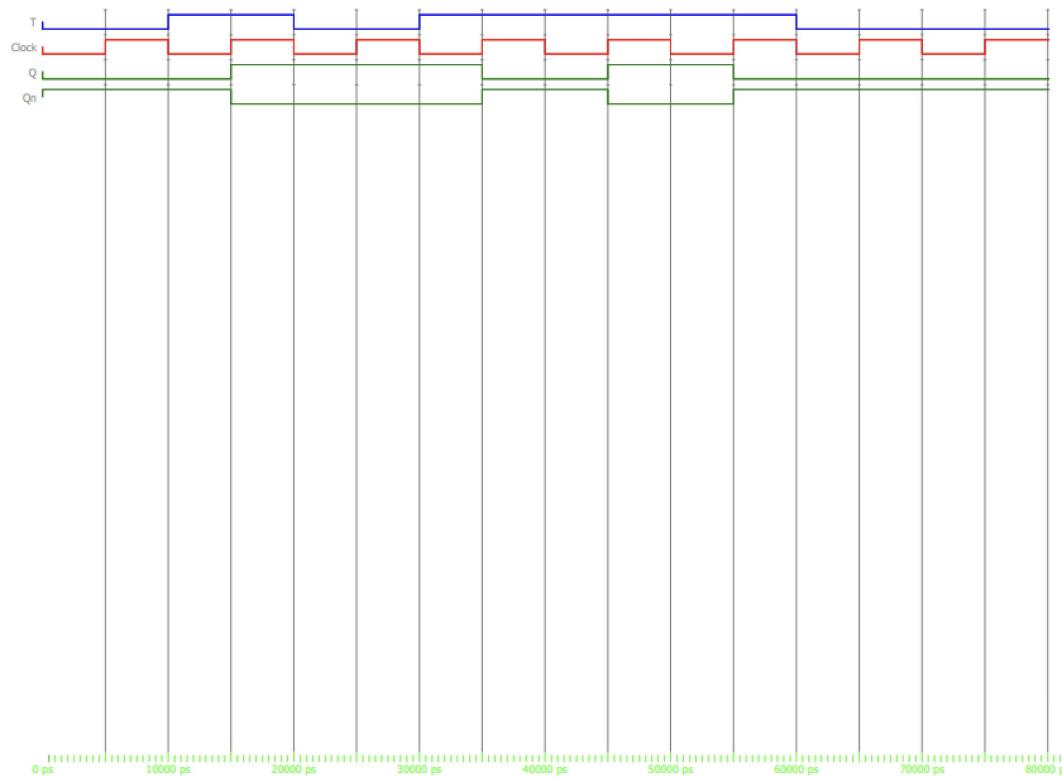
• JK with Asynchronous reset



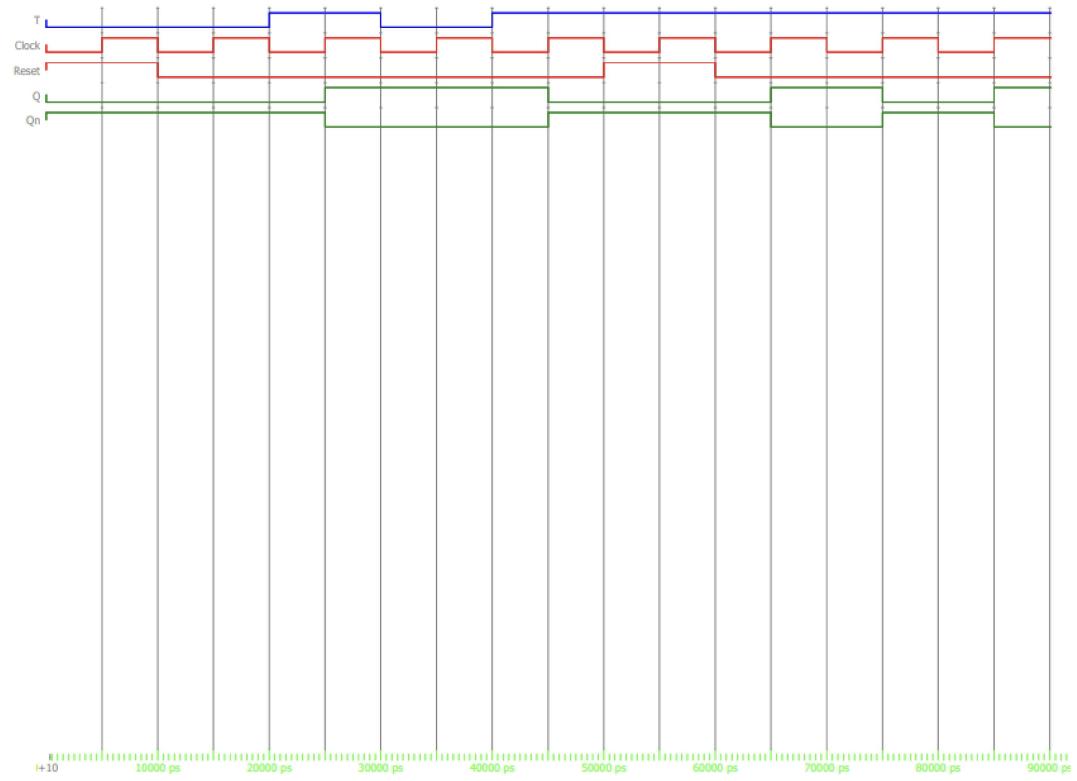
• JK with Synchronous reset



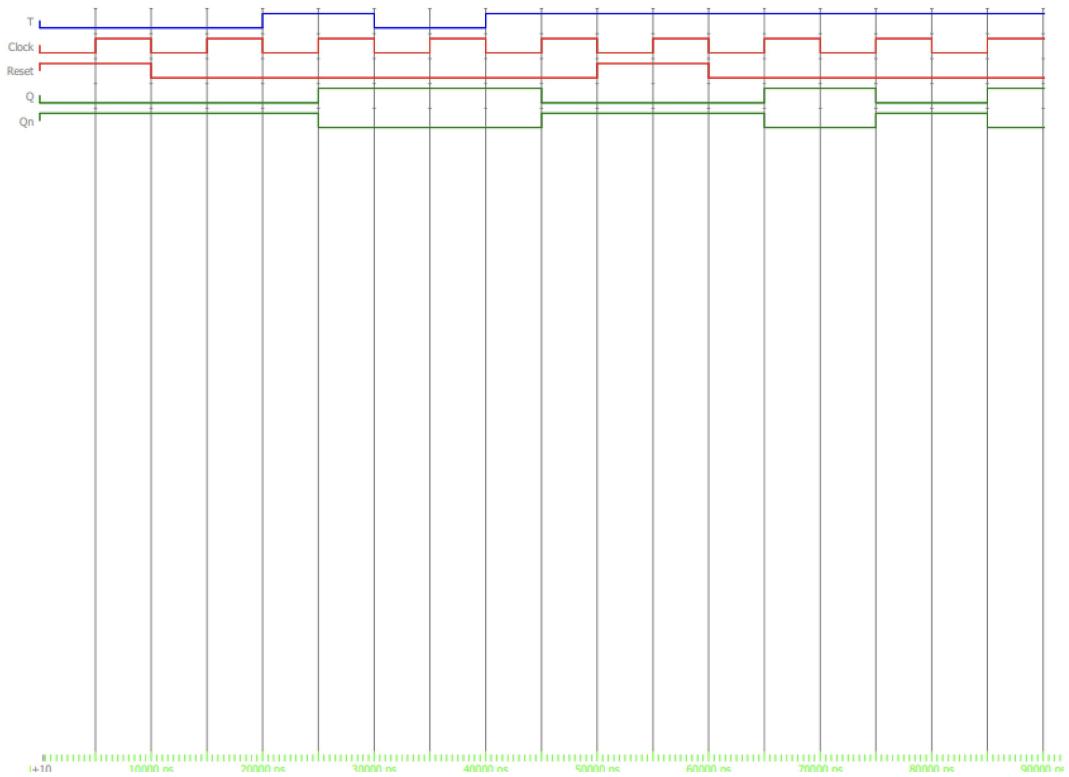
• T without reset



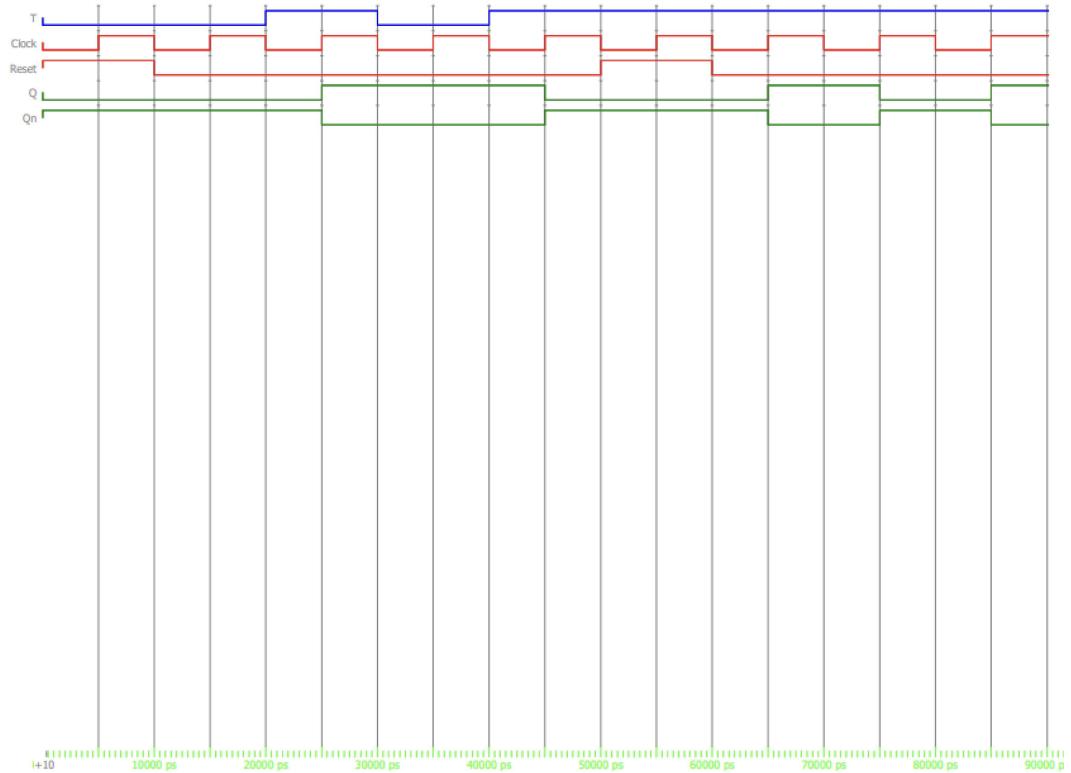
- T with Asynchronous reset



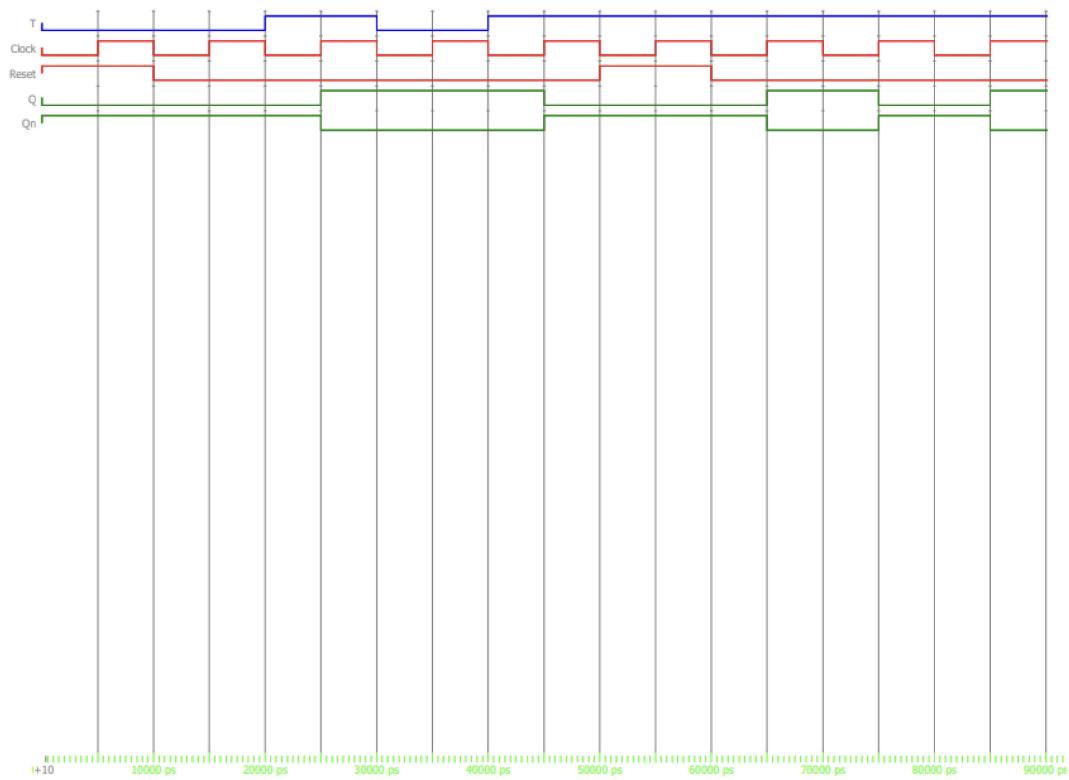
• T with Synchronous reset



- 4-bit Serial In and Serial Out Shift Register



• 4-bit Johnson Counter



6 Conclusion

In this lab, we designed D, JK, and T flip-flops, as well as shift registers and Johnson counters, using Verilog. Simulations verified that they functioned correctly, though some potential issues were observed, such as parameter mismatches and wiring errors. Despite these challenges, the lab was a success.