INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI

Course No: **EE206P**

Name of the Lab: **Digital Systems Laboratory**

Academic Year/Semester: **2024/B.Tech 3^{rd} 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

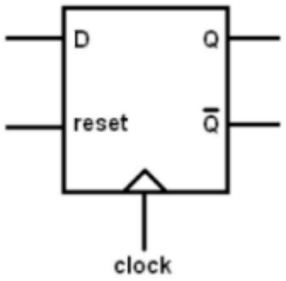
1 Problem Statement

The Problem Statements

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

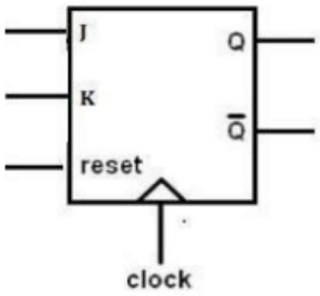
1. Write a Verilog HDL code for the D flip flop for the following options given below
   A. Without reset
   B. With asynchronous reset
   C. With synchronous reset

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



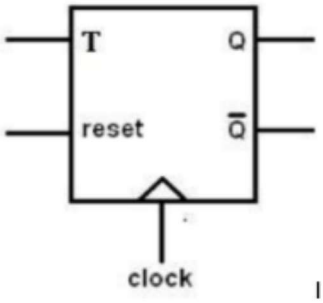
2. Write a Verilog HDL code for the JK flip flop for the following options given below
   A. Without reset
   B. With asynchronous reset
   C. With synchronous reset

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

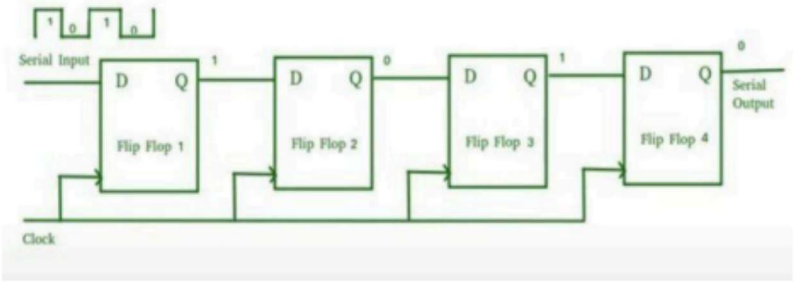


3. Write a Verilog HDL code for the T flip flop for the following options given below
   A. Without reset
   B. with asynchronous reset
   C. With synchronous reset

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

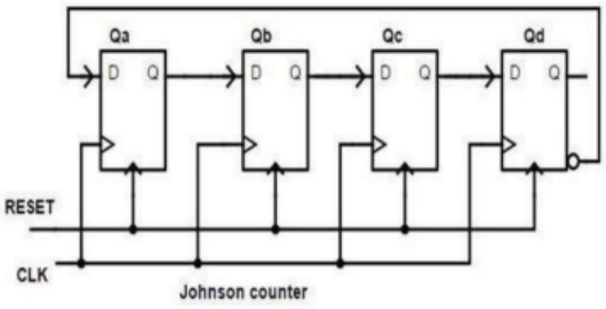


4. Write a Verilog HDL code for the 4-bit serial in and serial out shift register for transferring the input data which is shown in the figure and Verify the functionality of your implementation by simulating it using a test bench.



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 jhonson 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

```
module d_ff_without_r(clk,D,Q,Qb);
input clk;
input D;
output reg Q;
output Qb;

always@(posedge clk)
begin

Q <= D;

end

assign Qb = ~Q;

endmodule
```

TEST BENCH:

```
module TB_d_ff_without_r();
reg clk,D;
wire Q,Qb;
d_ff_without_r uut (.clk(clk),.D(D),.Q(Q),.Qb(Qb));
initial
begin
clk = 1'b0;
D =1;
repeat(10)
begin
#5
D = $random;
end
end
always
#5
clk = ~clk;
endmodule
```

b) D Flip Flop with asynchronous reset

```
module d_ff_with_ar(clk,D,Q,Qb,rst);
input clk;
input D;
input rst;
output reg Q;
output Qb;


always@(posedge clk or rst)

begin

if(rst == 1)
  Q <= 1'b0;
else
Q <= D;

end

assign Qb = ~Q;

endmodule
```

TEST BENCH:

```
module TB_d_ff_with_as();
reg clk;
reg D;
reg rst;
wire Q,Qb;

d_ff_with_ar uut (.clk(clk),.D(D),.Q(Q),.Qb(Qb),.rst(rst));

initial
begin
clk = 1'b0;
D =1;
rst = 1'b0;

```

```
repeat(10)
begin
#5
D = $random;
end

end

always
#5
clk = ~clk;
always
#5
rst = ~rst;

endmodule
```

c) D Flip Flop with synchronous reset

```
module d_ff_with_sr(clk,D,Q,Qb,rst);
input clk;
input D;
input rst;
output reg Q;
output Qb;


always@(posedge clk)
begin

if(rst == 1)
  Q <= 1'b0;
else
Q <= D;

end

assign Qb = ~Q;

endmodule
```

TEST BENCH:

```
module TB_d_ff_with_as();
reg clk;
reg D;
reg rst;
wire Q,Qb;

d_ff_with_sr uut (.clk(clk),.D(D),.Q(Q),.Qb(Qb),.rst(rst));

initial
begin
clk = 1'b0;
D =1;
rst = 1'b0;

repeat(10)
begin
#5
D = $random;
end

end

always
#5
clk = ~clk;
always
#5
rst = ~rst;

endmodule
```

II a) JK Flip Flop without reset

```
module jk_without_r(clk, J, K, Q, Qb);
    input clk;
    input J;
    input K;
    output reg Q;
    output Qb;

    always @(posedge clk) begin
        if (J==0 && K==0)
            Q <= Q;
        else if (J==0 && K==1)
            Q <= 0;
        else if (J==1 & K==0)
            Q <= 1;
        else
            Q <= ~Q;
    end

    assign Qb = ~Q;
endmodule
```

TEST BENCH:

```
module TB_jk_without_r();
    reg clk;
    reg J;
    reg K;
    wire Q;
    wire Qb;

    jk_without_r uut (.clk(clk), .J(J), .K(K), .Q(Q), .Qb(Qb));

    initial begin
        clk = 1'b0;
        J = 1'b0;
        K = 1'b0;

        #10 J = 1'b1; K = 1'b0;
        #10 J = 1'b0; K = 1'b1;
        #10 J = 1'b1; K = 1'b1;
        #10 J = 1'b0; K = 1'b0;
        #10 J = 1'b1; K = 1'b0;
        #10 J = 1'b1; K = 1'b1;
        #10 J = 1'b0; K = 1'b1;
        #10 J = 1'b0; K = 1'b0;
    end

    always #5 clk = ~clk;
endmodule
```

b) JK Flip Flop with Asynchronous reset

```
module jk_with_ar(clk, J, K, Q, Qb,rst);
    input clk;
    input J;
    input K;
    input rst;
    output reg Q;
    output Qb;

    always @(posedge clk or rst)
    begin
```

```
    if(rst == 1)
     Q <= 1'b0;
    else
        if (J==0 && K==0)
            Q <= Q;
        else if (J==0 && K==1)
            Q <= 0;
        else if (J==1 & K==0)
            Q <= 1;
        else
            Q <= ~Q;
    end

    assign Qb = ~Q;
endmodule
```

TEST BENCH:

```
module TB_jk_with_ar();
    reg clk;
    reg J;
    reg K;
    reg rst;
    wire Q;
    wire Qb;

    jk_with_ar uut (.clk(clk), .J(J), .K(K), .Q(Q), .Qb(Qb), .rst(rst))
       ;

    initial begin
        clk = 1'b0;
        rst = 1'b1;
        J = 1'b0;
        K = 1'b0;

        #10 rst = 1'b0;
        #10 J = 1'b1; K = 1'b0;
        #10 J = 1'b0; K = 1'b1;
        #10 J = 1'b1; K = 1'b1;
        #10 J = 1'b0; K = 1'b0;
```

```
        #10 J = 1'b1; K = 1'b0;
        #10 J = 1'b1; K = 1'b1;
        #10 J = 1'b0; K = 1'b1;
        #10 rst = 1'b1;
        #10 rst = 1'b0; J = 1'b1; K = 1'b1;
    end

    always #5 clk = ~clk;
endmodule
```

c) JK Flip Flop with synchronous reset

```
module jk_with_sr(clk, J, K, Q, Qb,rst);
    input clk;
    input J;
    input K;
    input rst;
    output reg Q;
    output Qb;

    always @(posedge clk)
    begin
    if(rst == 1)
     Q <= 1'b0;
    else
        if (J==0 && K==0)
            Q <= Q;
        else if (J==0 && K==1)
            Q <= 0;
        else if (J==1 & K==0)
            Q <= 1;
        else
            Q <= ~Q;
    end

    assign Qb = ~Q;
endmodule
```

TESTBENCH:

```
module TB_jk_with_sr();
    reg clk;
    reg J;
    reg K;
    reg rst;
    wire Q;
    wire Qb;

    jk_with_sr uut (.clk(clk), .J(J), .K(K), .Q(Q), .Qb(Qb), .rst(rst))
       ;

    initial begin
        clk = 1'b0;
        rst = 1'b1;
        J = 1'b0;
        K = 1'b0;

        #10 rst = 1'b0;
        #10 J = 1'b1; K = 1'b0;
        #10 J = 1'b0; K = 1'b1;
        #10 J = 1'b1; K = 1'b1;
        #10 J = 1'b0; K = 1'b0;
        #10 J = 1'b1; K = 1'b0;
        #10 J = 1'b1; K = 1'b1;
        #10 J = 1'b0; K = 1'b1;
        #10 rst = 1'b1;
        #10 rst = 1'b0; J = 1'b1; K = 1'b1;
    end

    always #5 clk = ~clk;
endmodule
```

III a) T FlipFlop Without reset

```
module t_without_r(clk,T,Q,Qb);
input clk;
input T;
output reg Q;
output Qb;

always@(posedge clk)
begin
 Q = T^Q;
end
assign Qb = ~Q;
endmodule
```

TEST BENCH:

```
module TB_t_without_r();
    reg clk;
    reg T;
    wire Q;
    wire Qb;

    t_without_r uut (.clk(clk), .T(T), .Q(Q), .Qb(Qb));

    initial begin
        clk = 1'b0;
        T = 1'b0;

        repeat(10) begin
            #10 T = $random;
        end
    end

    always #5 clk = ~clk;
endmodule
```

b) T FlipFlop with Asynchronous reset

```
module t_with_ar(clk,T,Q,Qb,rst);
input clk;
input T;
input rst;
output reg Q;
output Qb;

always@(posedge clk or rst)
begin
if(rst == 1)
Q = 1'b0;
else
 Q = T^Q;
end
assign Qb = ~Q;
endmodule
```

TEST BENCH:

```
module TB_t_with_ar();
    reg clk;
    reg T;
    reg rst;
    wire Q;
    wire Qb;

    t_with_ar uut (.clk(clk), .T(T), .Q(Q), .Qb(Qb), .rst(rst));

    initial begin
        clk = 1'b0;
        rst = 1'b1;
        T = 1'b0;

        #10 rst = 1'b0;
        repeat(10) begin
            #10 T = $random;
        end
        #10 rst = 1'b1;
        #10 rst = 1'b0;
```

```
        repeat(5) begin
            #10 T = $random;
        end
    end

    always #5 clk = ~clk;
endmodule
```

c) T Flip Flop with Synchronous reset

```
module t_with_sr(clk,T,Q,Qb,rst);
input clk;
input T;
input rst;
output reg Q;
output Qb;

always@(posedge clk)
begin
if(rst == 1)
Q = 1'b0;
else
 Q = T^Q;
end
assign Qb = ~Q;
endmodule
```

TEST BENCH:

```
module TB_t_with_sr();
    reg clk;
    reg T;
    reg rst;
    wire Q;
    wire Qb;

    t_with_sr uut (.clk(clk), .T(T), .Q(Q), .Qb(Qb), .rst(rst));

    initial begin
        clk = 1'b0;
        rst = 1'b1;
```

```
        T = 1'b0;

        #10 rst = 1'b0;
        repeat(10) begin
            #10 T = $random;
        end
        #10 rst = 1'b1;
        #10 rst = 1'b0;
        repeat(5) begin
            #10 T = $random;
        end
    end

    always #5 clk = ~clk;
endmodule
```

IV 4-bit serial in and serial out shift register

```
module siso_shift_register_4bit (
    input wire clk,
    input wire serial_in,
    output wire serial_out
);

reg [3:0] shift_reg;

initial begin
    shift_reg = 4'b1010;
end

always @(posedge clk) begin
    shift_reg <= {serial_in, shift_reg[3:1]};
end

assign serial_out = shift_reg[0];

endmodule
```

TEST BENCH:

```
module tb_siso_shift_register_4bit();

    reg clk;
    reg serial_in;
    wire serial_out;

    siso_shift_register_4bit uut (
        .clk(clk),
        .serial_in(serial_in),
        .serial_out(serial_out)
    );

    initial begin
        clk = 0;
        forever #10 clk = ~clk;
    end

    initial begin
        #20;
        repeat(10) begin
            serial_in = $random % 2;
            #20;
        end
        #100;
        $finish;
    end

    initial begin
        $monitor("Time = %0d, serial_in = %b, serial_out = %b,
           shift_reg = %b", $time, serial_in, serial_out, uut.shift_reg
           );
    end

endmodule
```

V 4-bit Johnson counter

```
module JohnsonCounter (
    input wire clk,
    input wire reset,
    output reg [3:0] Q
);

always @(posedge clk or posedge reset) begin
    if (reset)
        Q <= 4'b0000;
    else
        Q <= {~Q[0], Q[3:1]};
end

endmodule
```

TEST BENCH:

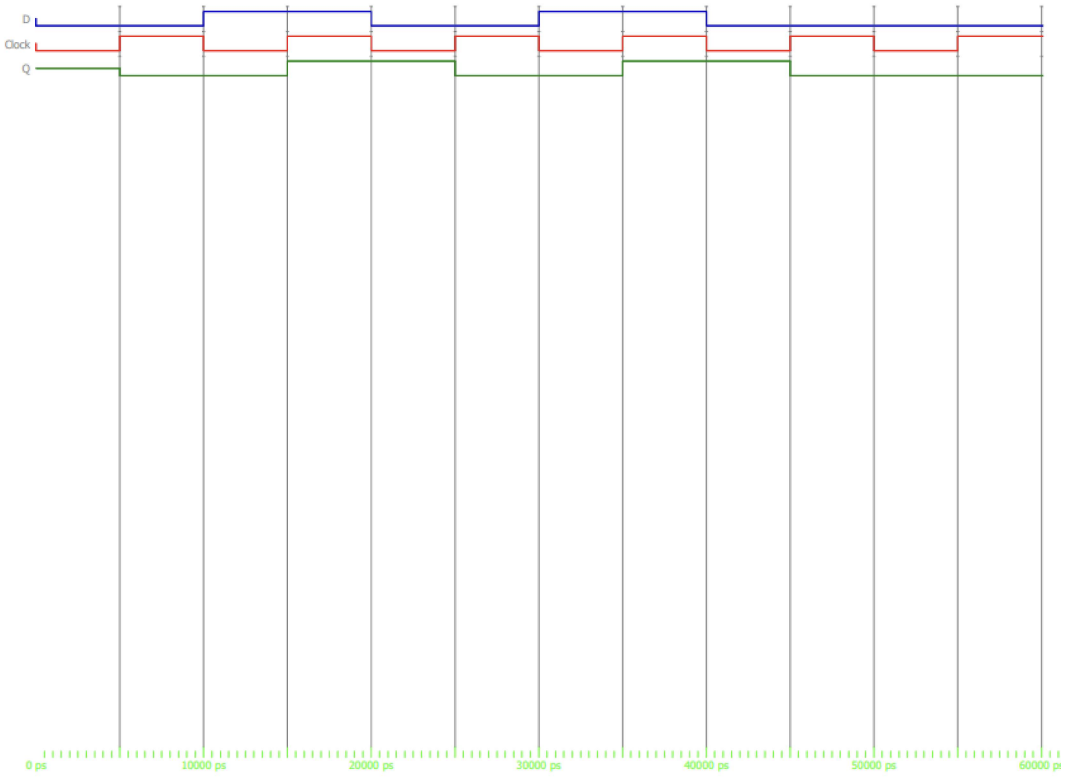
```
module tb_JohnsonCounter();

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

    JohnsonCounter uut (
        .clk(clk),
        .reset(reset),
        .Q(Q)
    );

    initial begin
        clk = 0;
        forever #10 clk = ~clk;
    end

    initial begin
        reset = 1;
        #20;
        reset = 0;
        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
```

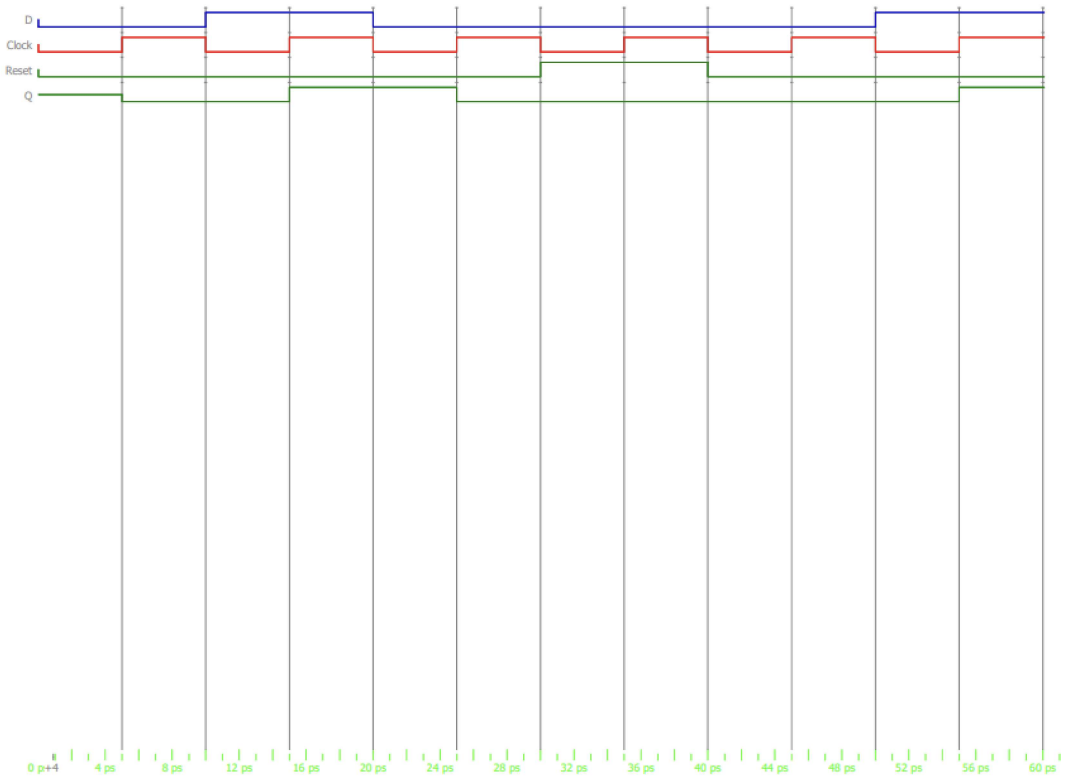
5 Simulation Results and Discussions

THE WAVEFORMS OF THE ABOVE HDL CODES

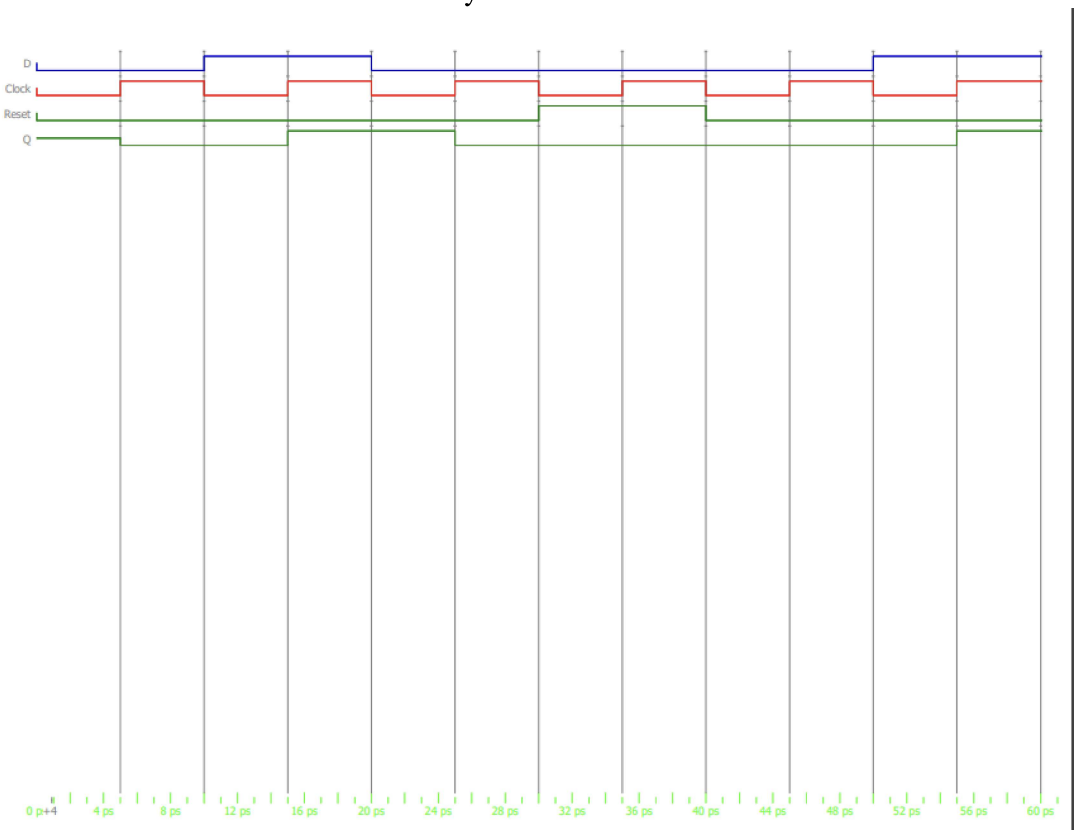
- D without reset



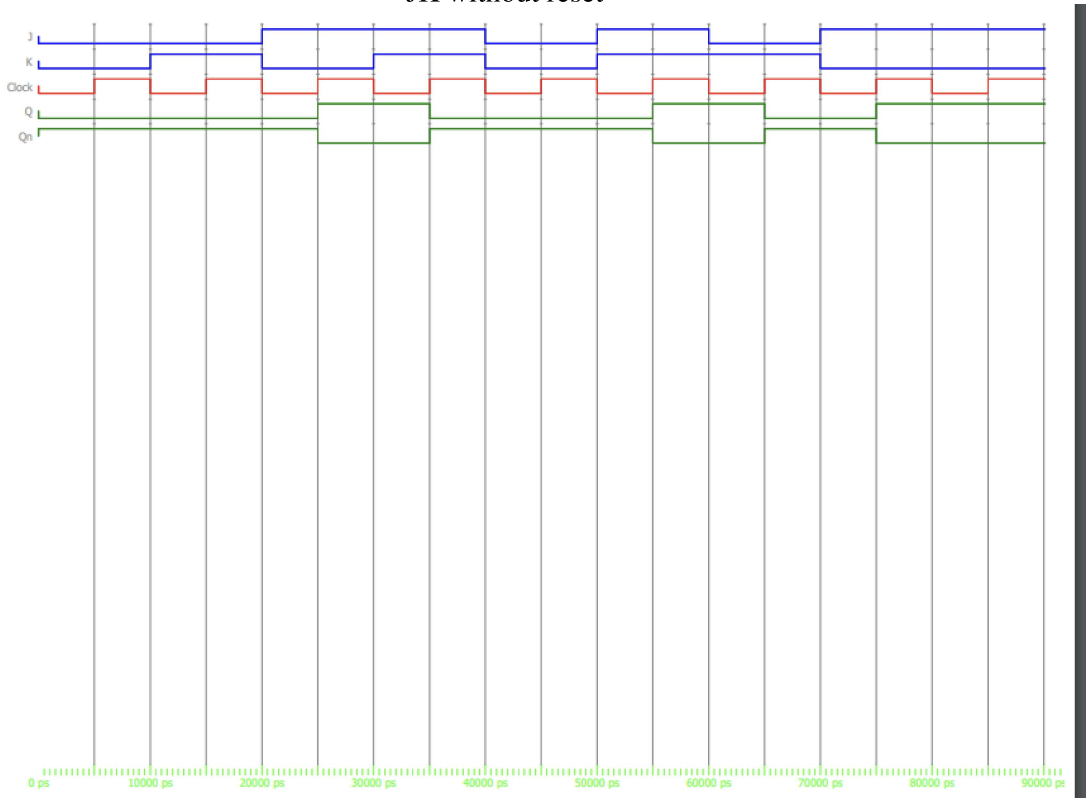
- D with Asynchronous reset



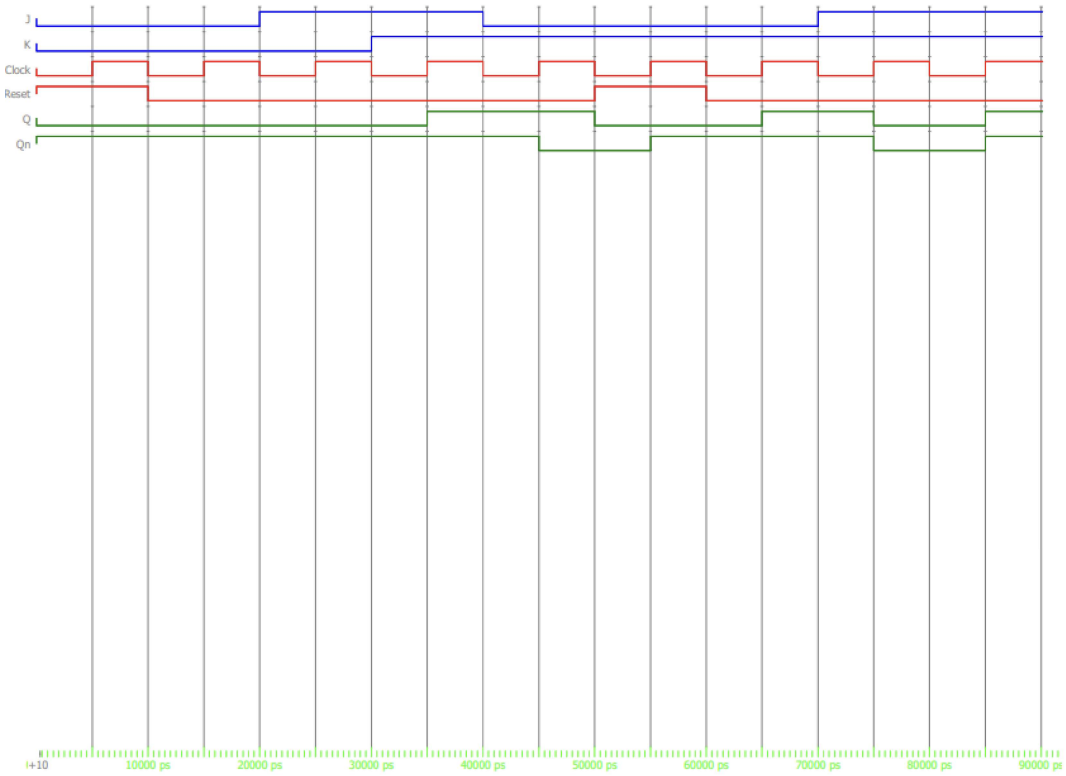
- D with Synchronous reset



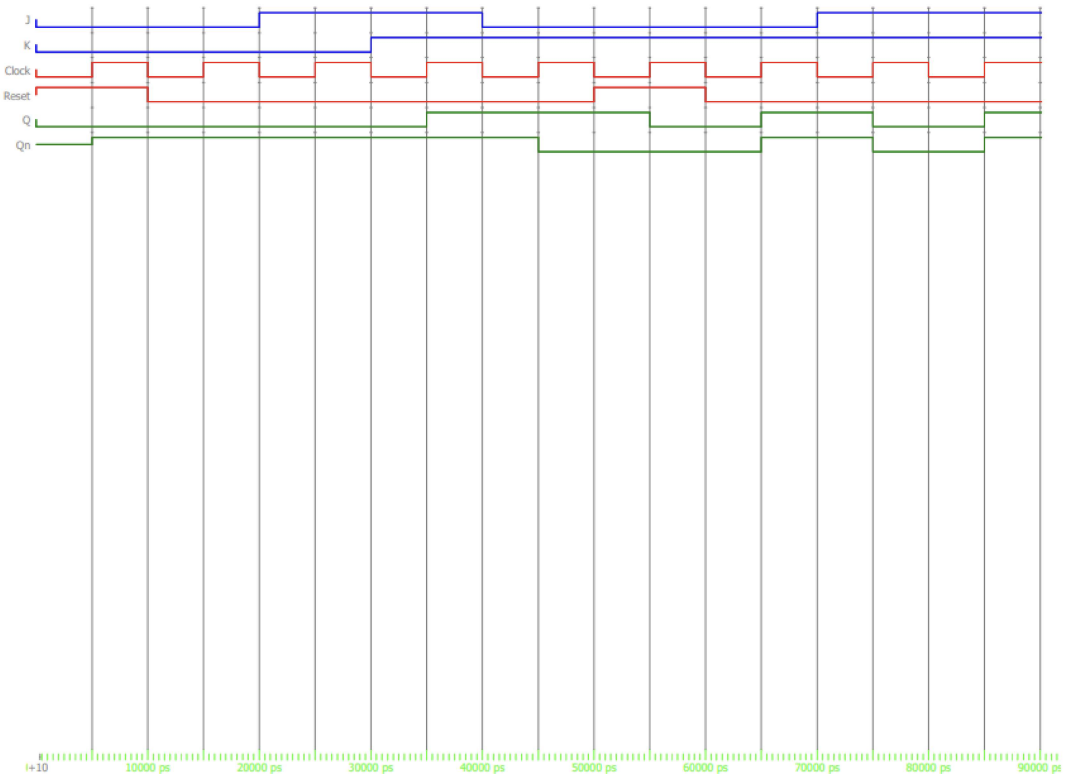
- JK without reset



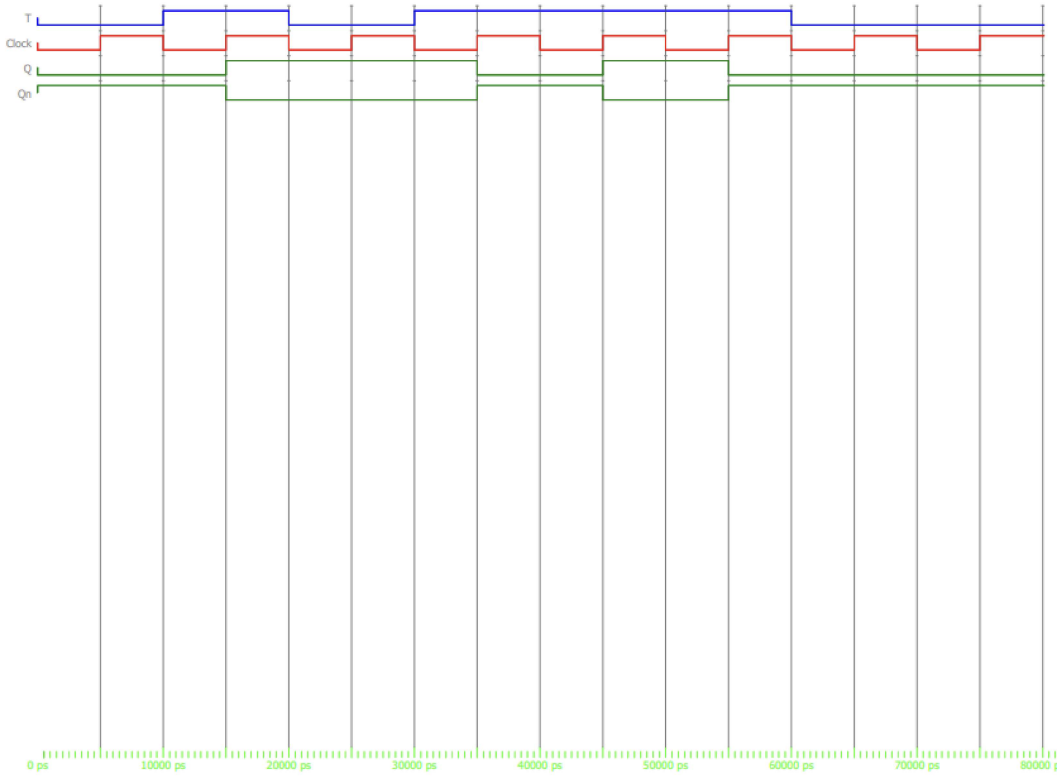
- JK with Asynchronous reset



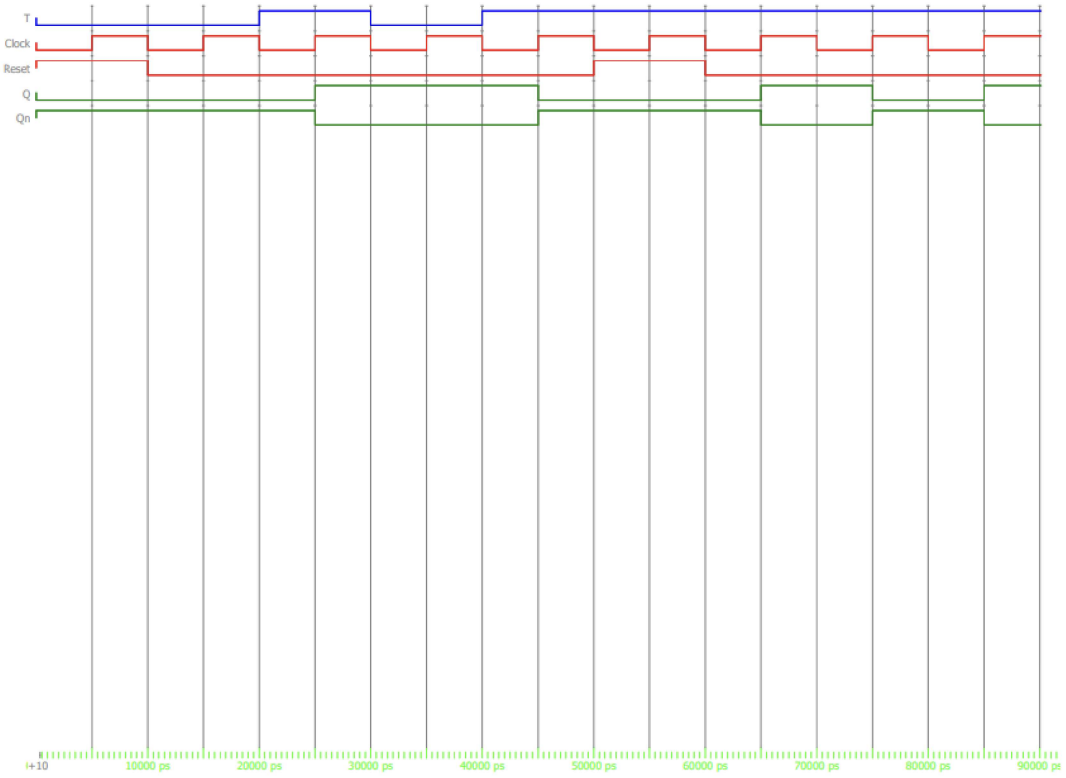
- JK with Synchronous reset



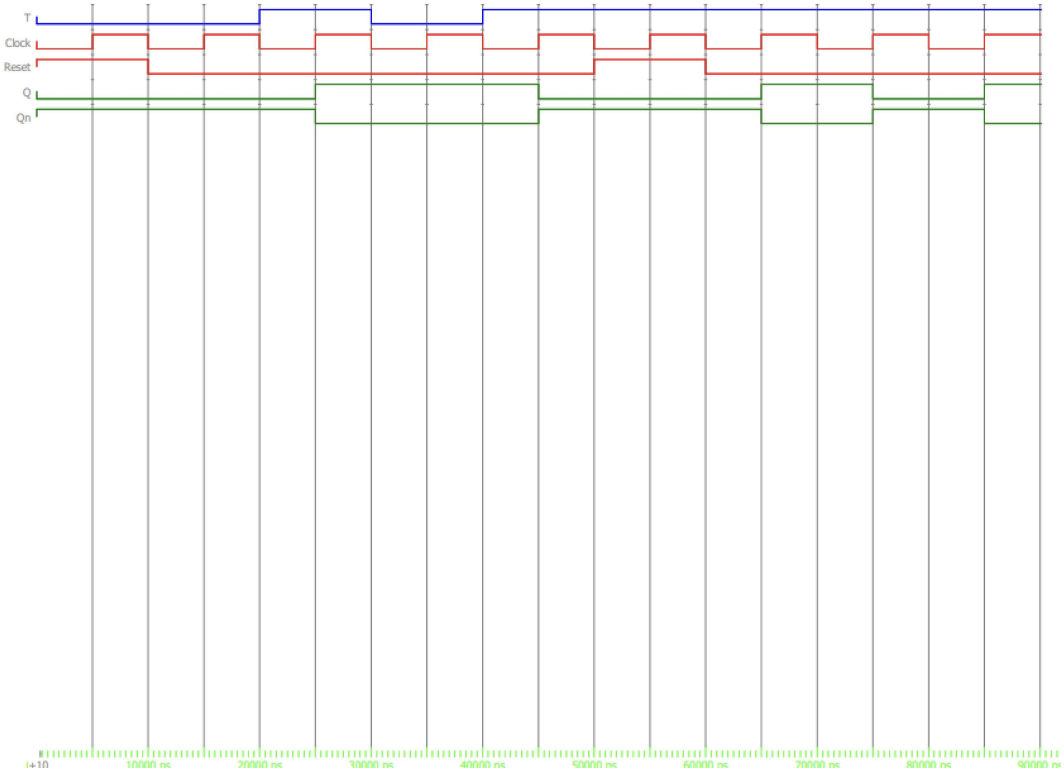
- T without reset



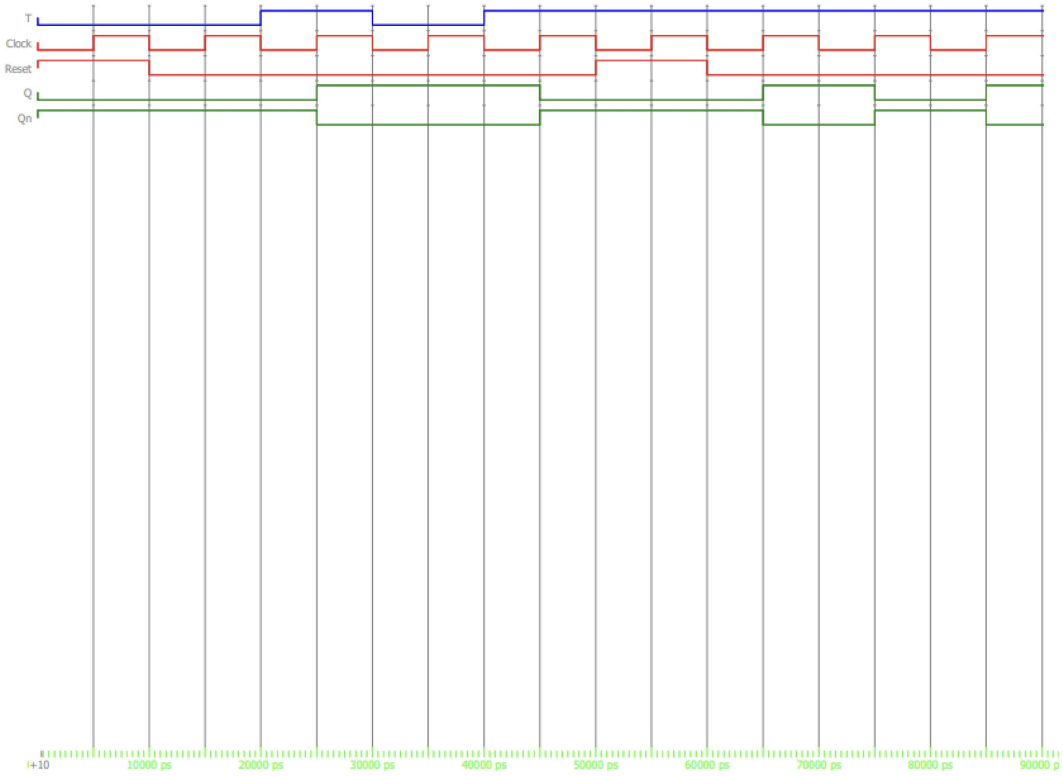
- T with Asynchronous reset



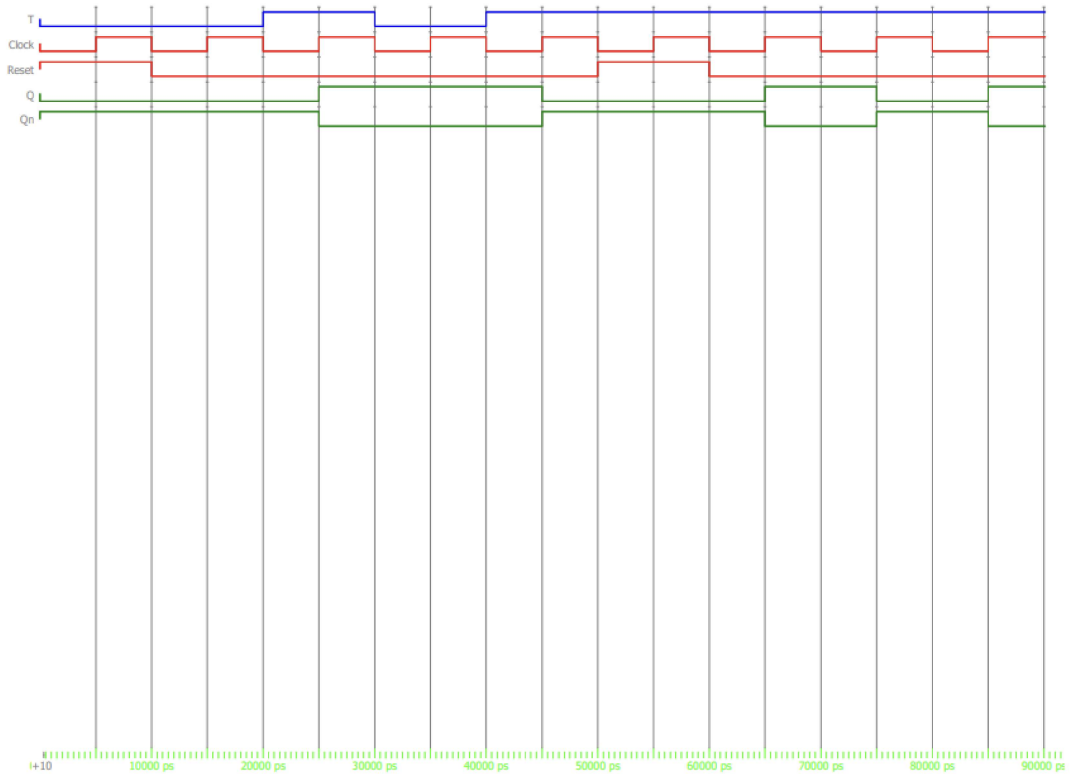
- T with Synchronous reset



- 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.