Digital Systems : Assignment 4

Nimitt (22110169)

Pratham Sharda (2211203)

Toggle Flip Flop

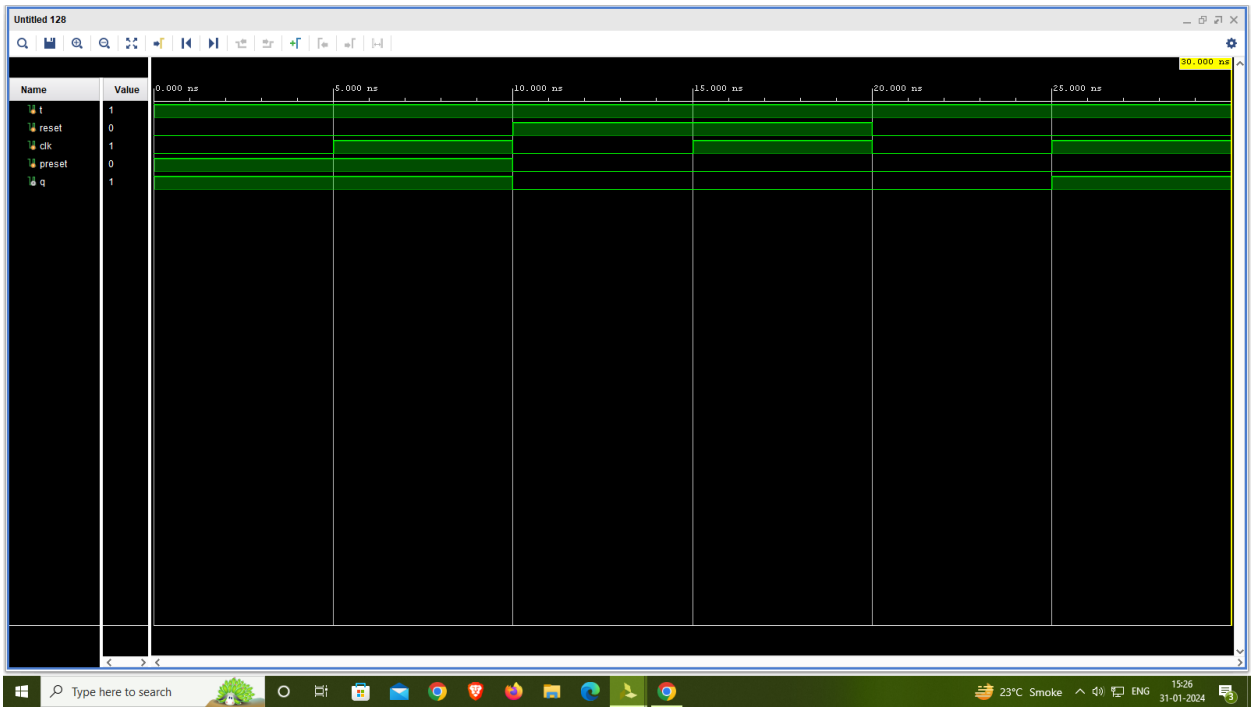
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 31.01.2024 07:12:01
// Design Name:
// Module Name: tff
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module tff(
    input clk,
    input reset,
    input t,
    input preset,
```

```
    output reg q
    );
    always @ (posedge clk, posedge reset, posedge preset) begin
        if (reset)
          q <= 0;
        else if (preset)
          q <= 1;
        else
            begin
            if (t == 1)
                q <= ~q;
            else
                q <= q;
            end
      end
endmodule
```

```
TestBench
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 31.01.2024 07:45:11
// Design Name:
// Module Name: tff_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
```

```
// Additional Comments:
//
//////////////////////////////////////////////////////////////////


module tff_tb();
    reg t, reset, clk, preset;
    wire q;
    tff uut(clk, reset,t, preset, q);
    initial
    begin
    clk = 0;
        forever #5 clk = ~clk;
    end
    initial
    begin
    t = 1; reset = 0; preset = 1;
    #10;
        t = 1; reset = 1; preset = 0;
    #10;
        t = 1; reset = 0; preset = 0;
    #10;

    $finish();
    end
endmodule
```

Simultation Results :



Counter

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 31.01.2024 07:08:44
// Design Name:
// Module Name: Counter
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
```

```
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////


module Counter(
    input clk,
    input reset,
    output [3:0] count,
    input up_down,
    input bin_bcd,
    input preset
    );



     and(cond1,count[3],~count[0], ~count[2],count[1], bin_bcd);
     or(cond2,cond1,reset);
     and(cond10, preset, bin_bcd);
     or(cond11, cond2, cond10);




     and(cond4,count[3],count[1],count[0],count[2], bin_bcd);
     or(cond6, cond4, preset);
     and(cond9, preset, ~bin_bcd);
     or(cond7, cond4, cond11);


    tff a1(.t(1),.clk(clk),.reset(cond2), .preset(cond6), .q(cou


    and(g1,~up_down,count[0]); //  Up Counter
    and(g2,up_down,~count[0]); // Down Counter
```

```
    or(f1,g1,g2);


    tff a2(.t(f1),.clk(clk),.reset(cond7),.preset(cond9), .q(cou


    and(h1,g1,count[1]);
    and(h2,g2,~count[1]);
    or(f2,h1,h2);


    tff a3(.t(f2),.clk(clk),.reset(cond7), .preset(cond9), .q(co


    and(i1,h1,count[2]);
    and(i2,h2,~count[2]);
    or(f3,i1,i2);


    tff a4(.t(f3),.clk(clk),.reset(cond2),. preset(cond6), .q(co

endmodule
```

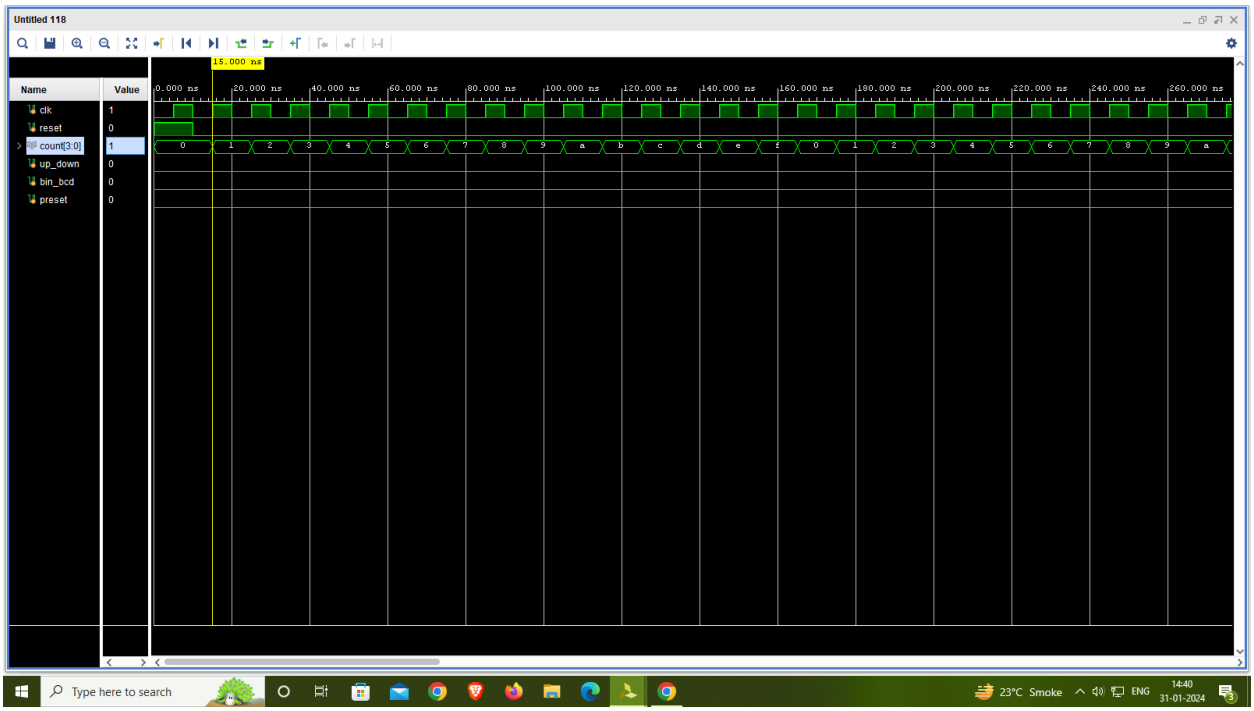
```
TestBench
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 31.01.2024 07:20:07
// Design Name:
// Module Name: Counter_tb
// Project Name:
// Target Devices:
// Tool Versions:
```

```
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////


module Counter_tb();

    reg clk;
    reg reset;
    wire [3:0] count;
    reg up_down;
    reg bin_bcd;
    reg preset;
    Counter uut( clk,reset,count, up_down, bin_bcd, preset);

    initial begin
    clk = 0;
    forever #5 clk = ~clk;
    end

    initial begin

    // Test Case 1
//    reset = 1; up_down = 0; bin_bcd = 0; preset = 0;
//    #10;
//    reset = 0; up_down = 0; bin_bcd = 0; preset = 0;
//    #10;

    // Test Case 2
//    reset = 1; up_down = 0; bin_bcd = 1; preset = 0;
```

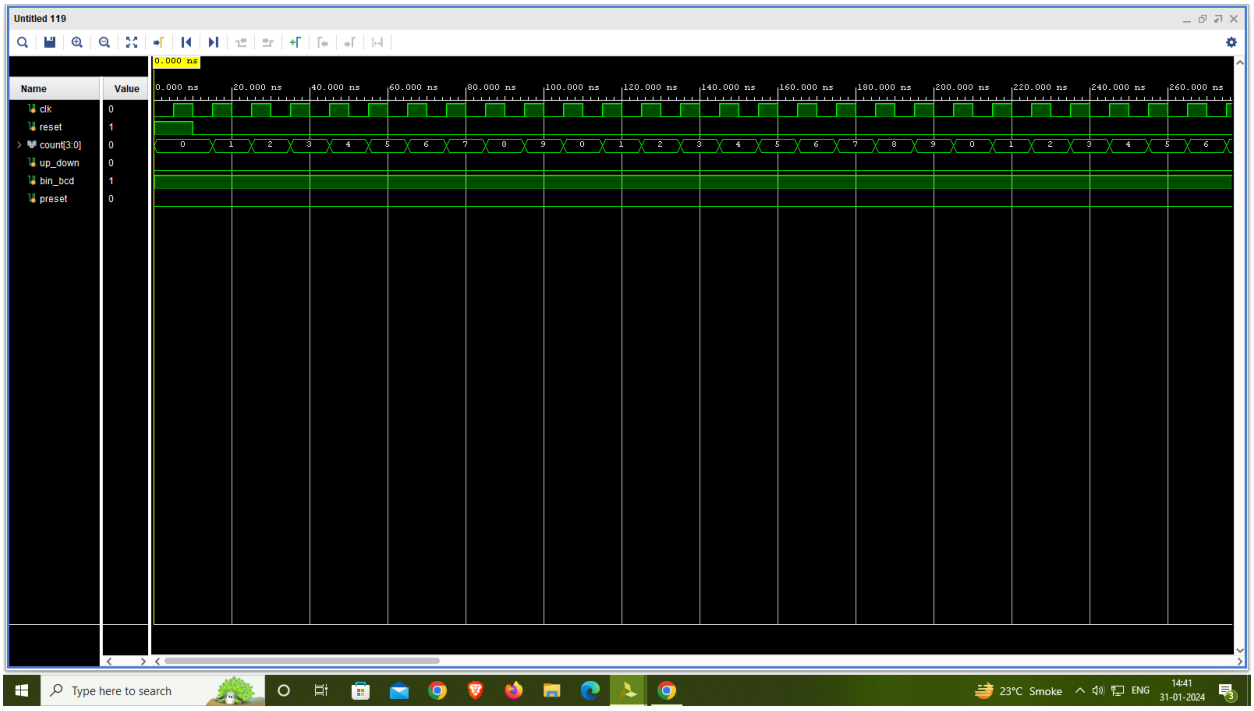
```
//    #10;
//    reset = 0; up_down = 0; bin_bcd = 1; preset = 0;
//    #10;

    // Test Case 3
//    reset = 0; up_down = 1; bin_bcd = 0; preset = 1;
//    #10;
//    reset = 0; up_down = 1; bin_bcd = 0; preset = 0;
//    #10;

    // Test Case 4
    reset = 0; up_down = 1; bin_bcd = 1; preset = 1;
    #10;
    reset = 0; up_down = 1; bin_bcd = 1; preset = 0;
    #10;
    end

endmodule
```

Simulation Results :

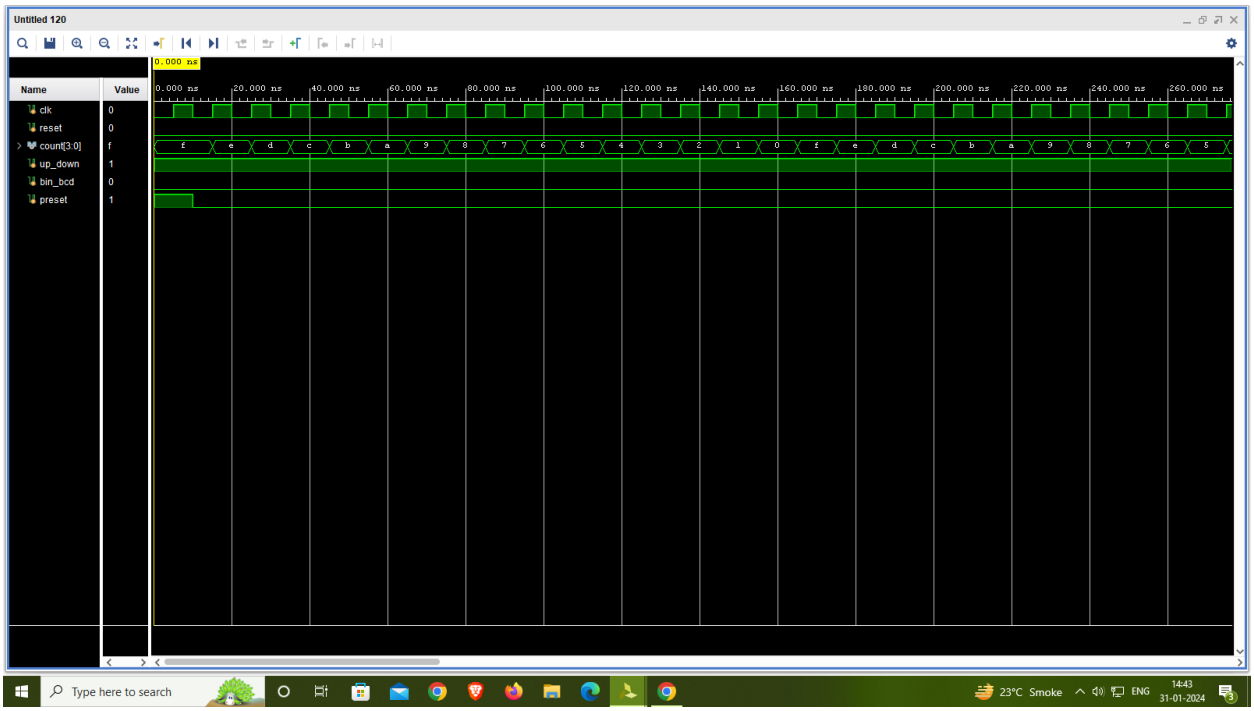
1. Binary Up Counter



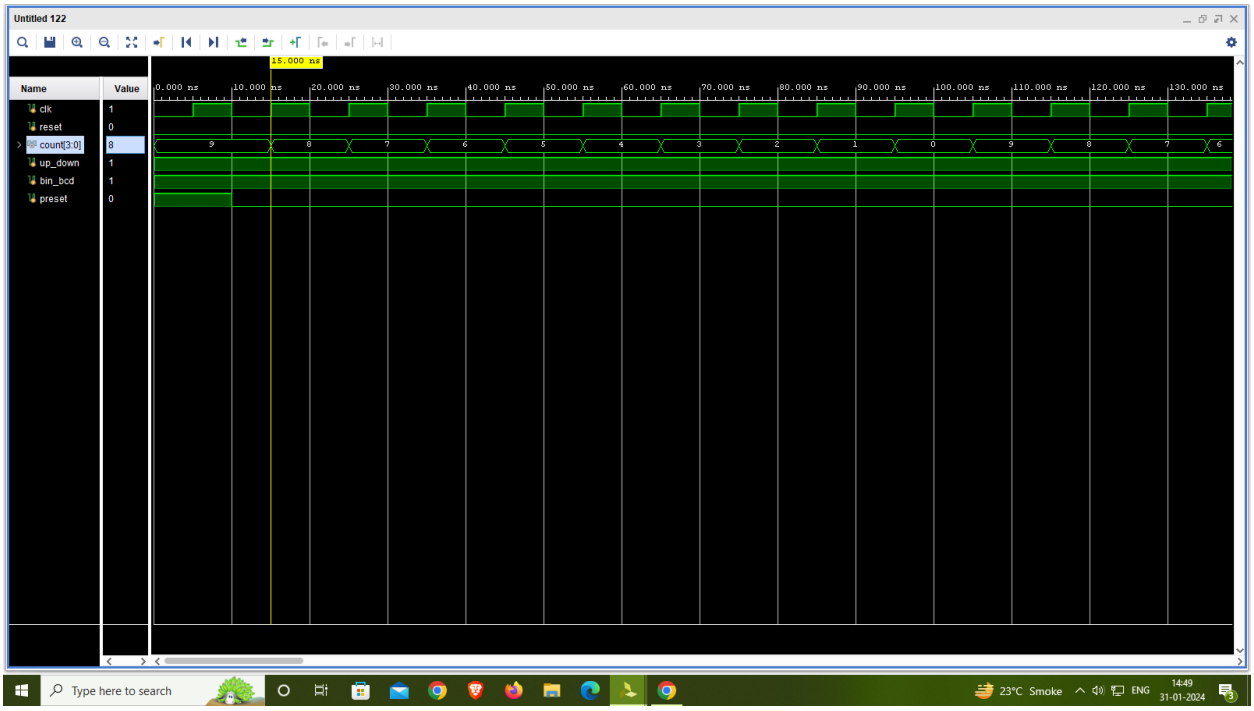
2. BCD Up Counter



3. Binary Down Counter



4. BCD Down Counter



We can see this matches the expected Output.

Shift Register

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 31.01.2024 07:00:11
// Design Name:
// Module Name: ShiftRegister
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module ShiftRegister(
    input [3:0] D,
    input S,
    input [1:0] Shift,
    input Clk,
    output reg [3:0] Q
    );
    always @(posedge Shift,negedge Shift, posedge Clk)
    begin
        if (~Shift[1] & ~Shift[0])            // Parallel Input
            Q <= D;
        else if (~Shift[1] & Shift[0])
```

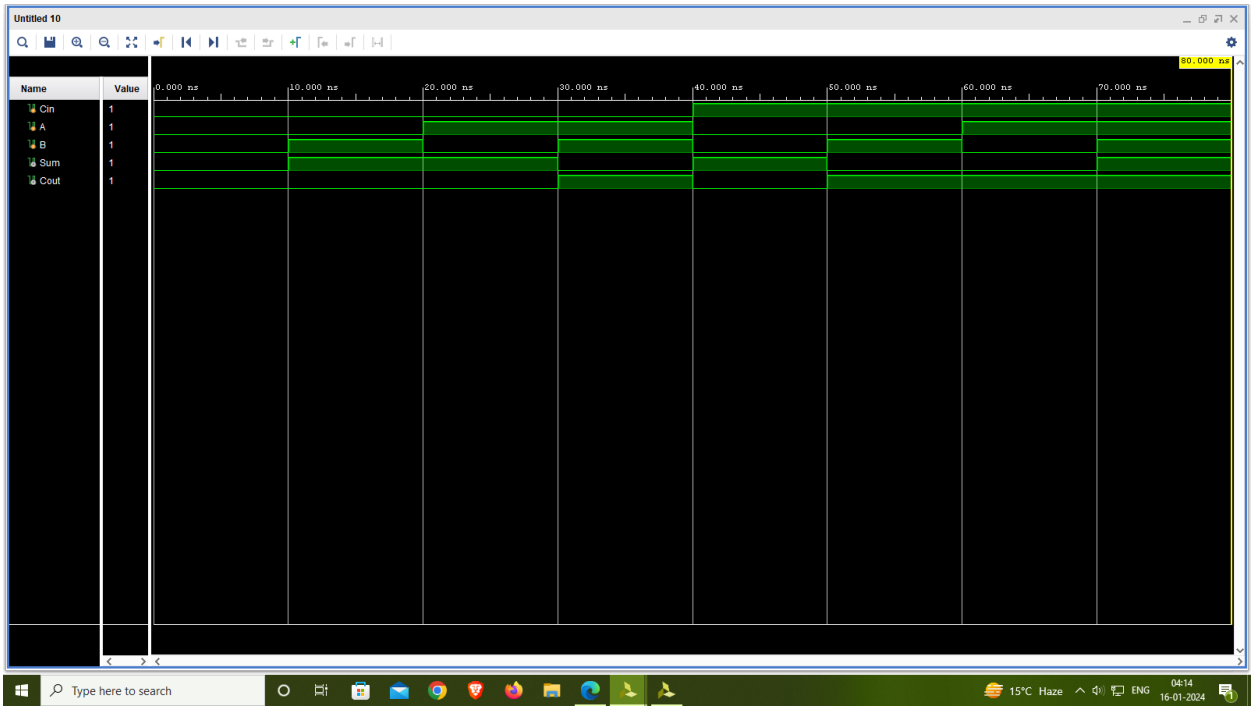
```
            begin                                   // Left Shift Re
                Q[3] <= Q[2];
                Q[2] <= Q[1];
                Q[1] <= Q[0];
                Q[0] <= S;
            end
        else if (Shift[1] & ~Shift[0])
            begin
                Q[0] <= Q[1];
                Q[1] <= Q[2];
                Q[2] <= Q[3];
                Q[3] <= S;                          // RightSl
            end
        else                                    // No Shift
            Q <= Q;
    end
endmodule
```

```
TestBench
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 31.01.2024 10:22:54
// Design Name:
// Module Name: ShiftRegister_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
```

```
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////


module ShiftRegister_tb();
    reg [3:0] D;
    reg S;
    reg [1:0] Shift;
    reg Clk;
    wire [3:0] Q;

    ShiftRegister uut(D, S, Shift, Clk, Q);

    initial
    begin
        Clk = 1;
        forever #5 Clk = ~Clk;
    end

    initial
    begin
    D = 4'b0000; Shift = 2'b00;  S = 0;
    #10;
    D = 4'b0000; Shift = 2'b01;  S = 1;
    #10;
    D = 4'b0000; Shift = 2'b01;  S = 1;
    #10;
    D = 4'b0000; Shift = 2'b01;  S = 1;
    #10;
    D = 4'b0000; Shift = 2'b01;  S = 1;
    #10;
    D = 4'b0000; Shift = 2'b10;  S = 0;
    #10;
    D = 4'b0000; Shift = 2'b10;  S = 0;
```

```
    #10;
    D = 4'b0000; Shift = 2'b10;  S = 0;
    #10;
    D = 4'b0000; Shift = 2'b10;  S = 0;
    #10;
    D = 4'b1111; Shift = 2'b00;  S = 0;
    #10;
    $finish();
    end
endmodule
```

Simulation Results :



Top Module

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
```

```
// Engineer:
//
// Create Date: 31.01.2024 10:40:19
// Design Name:
// Module Name: top
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////


module top(
    input S,
    input [2:0] Shift,
    input Clk,
    input reset,
    input preset,
    input up_down,
    input bin_bcd,
    output [3:0] OUT
    );
    wire [3:0] out;
    Counter c(.clk(Clk), .reset(reset), .count(out), .up_down(u
    ShiftRegister sr(.D(out), .S(S), .Shift(Shift), .Clk(Clk),
endmodule
```

```
TestBench
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 31.01.2024 12:12:07
// Design Name:
// Module Name: top_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////


module top_tb();
    reg S;
    reg [2:0] Shift;
    reg Clk;
    reg reset;
    reg preset;
    reg up_down;
    reg bin_bcd;
    wire [3:0] OUT;

    top uut(S, Shift, Clk, reset, preset, up_down, bin_bcd, OUT
    initial
```

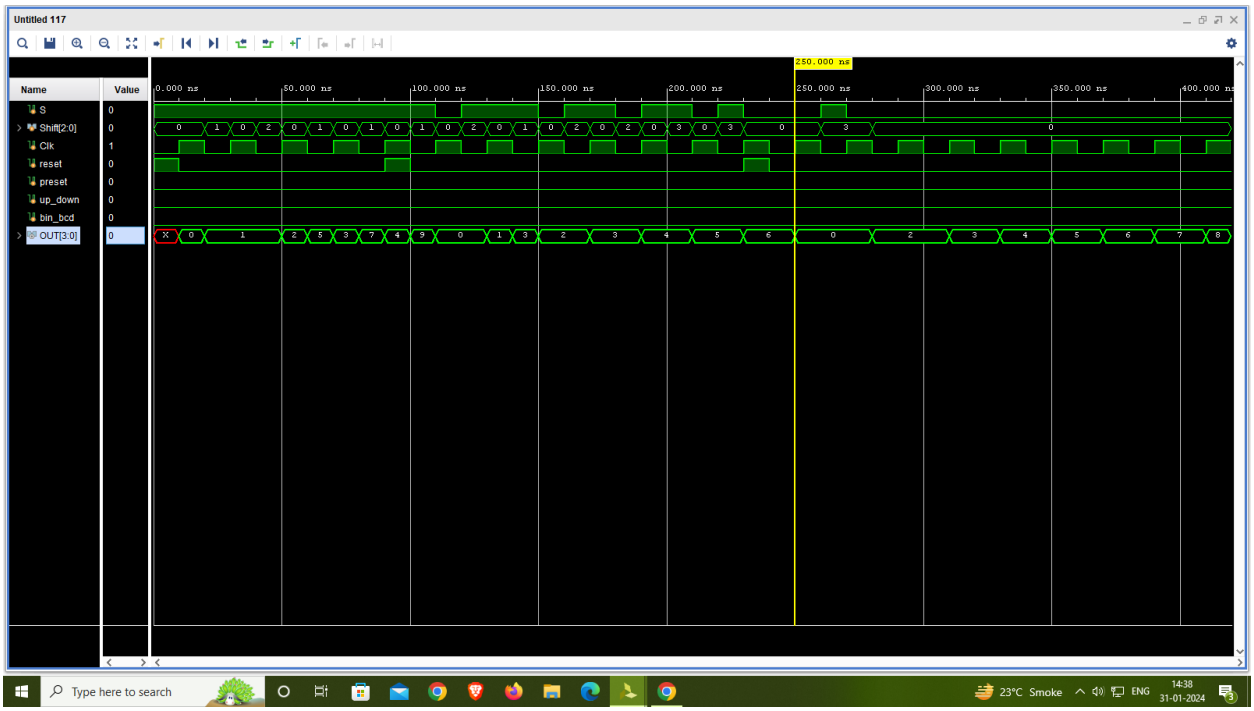
```
    begin
        Clk = 0;
        forever #10 Clk = ~Clk;
    end

    initial
    begin
        // Test Case 1 Binary Up
        S = 1; Shift = 0; reset = 1; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 1; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 2; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 1; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 1; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 0; reset = 1; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 1; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 2; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
```

```
        S = 1; Shift = 1; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 2; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 2; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 3; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 3; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 1; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 1; Shift = 3; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 3; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
```

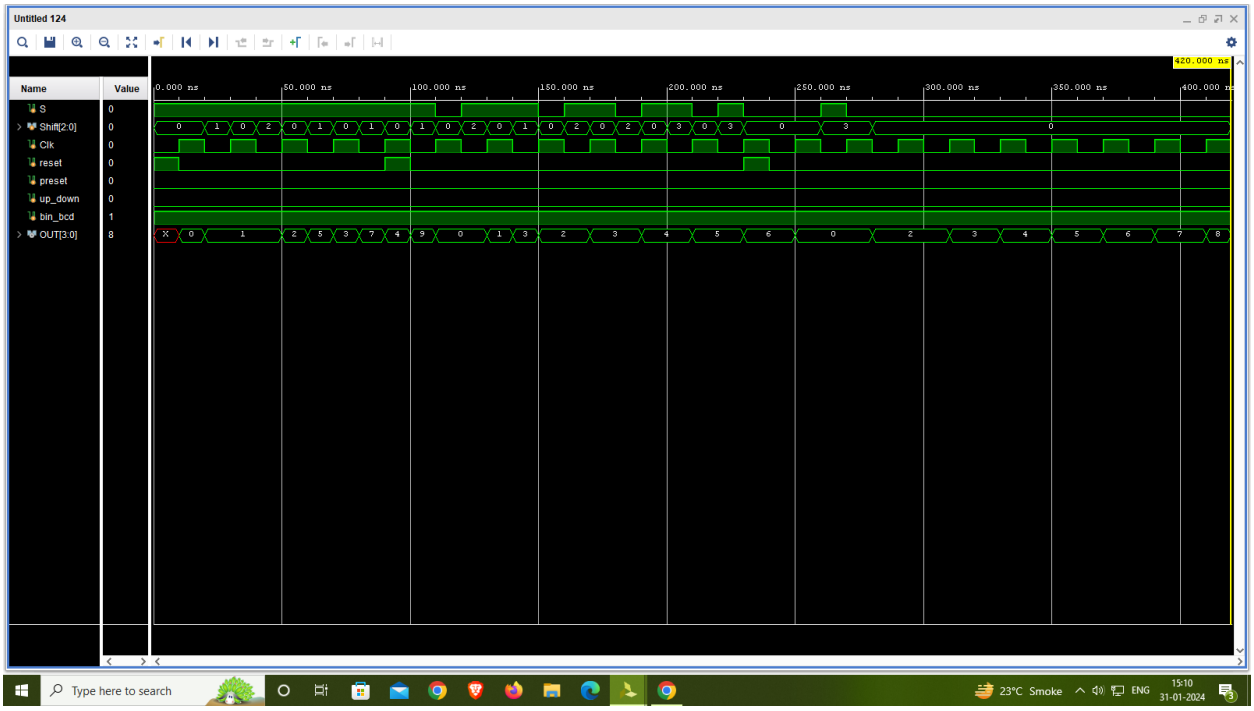
```
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;
        S = 0; Shift = 0; reset = 0; preset = 0; up_down = 0; bi
        #10;

        // Similarly Done for other modes of counter
                // Uncomment the corresponding Test Case

    $finish();
    end
endmodule
```

Simulation Results :

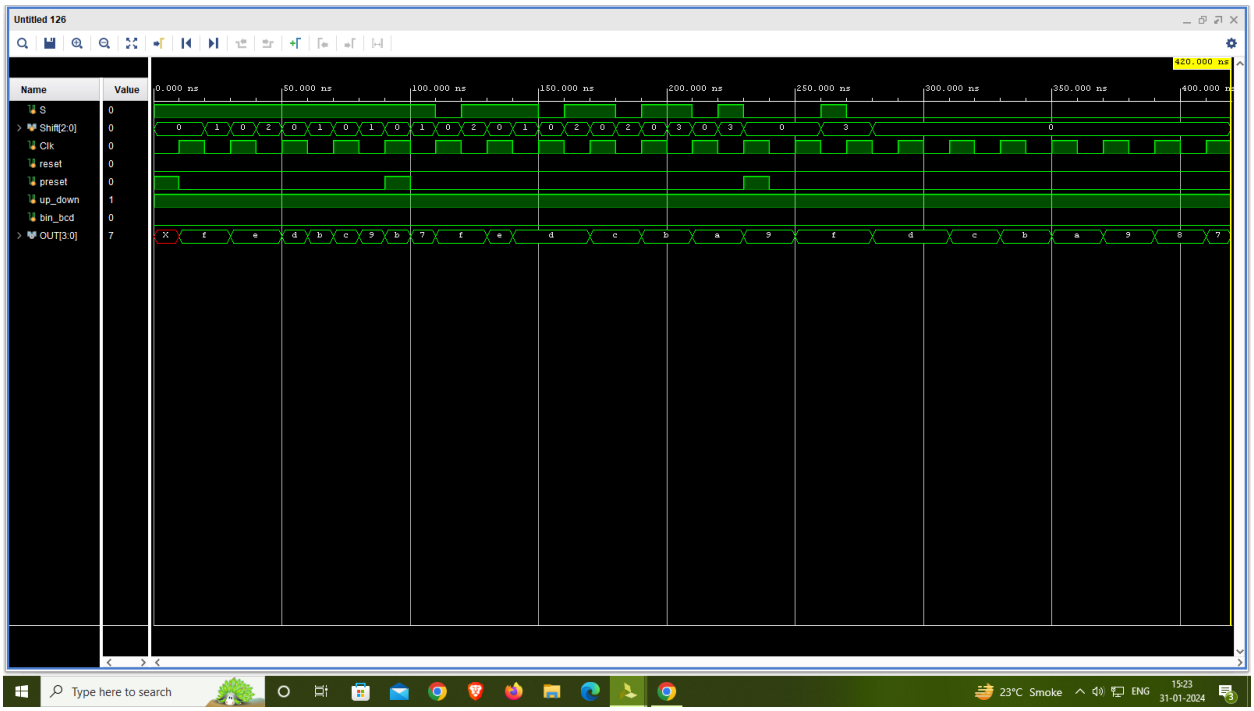
1. Binary Up Counter



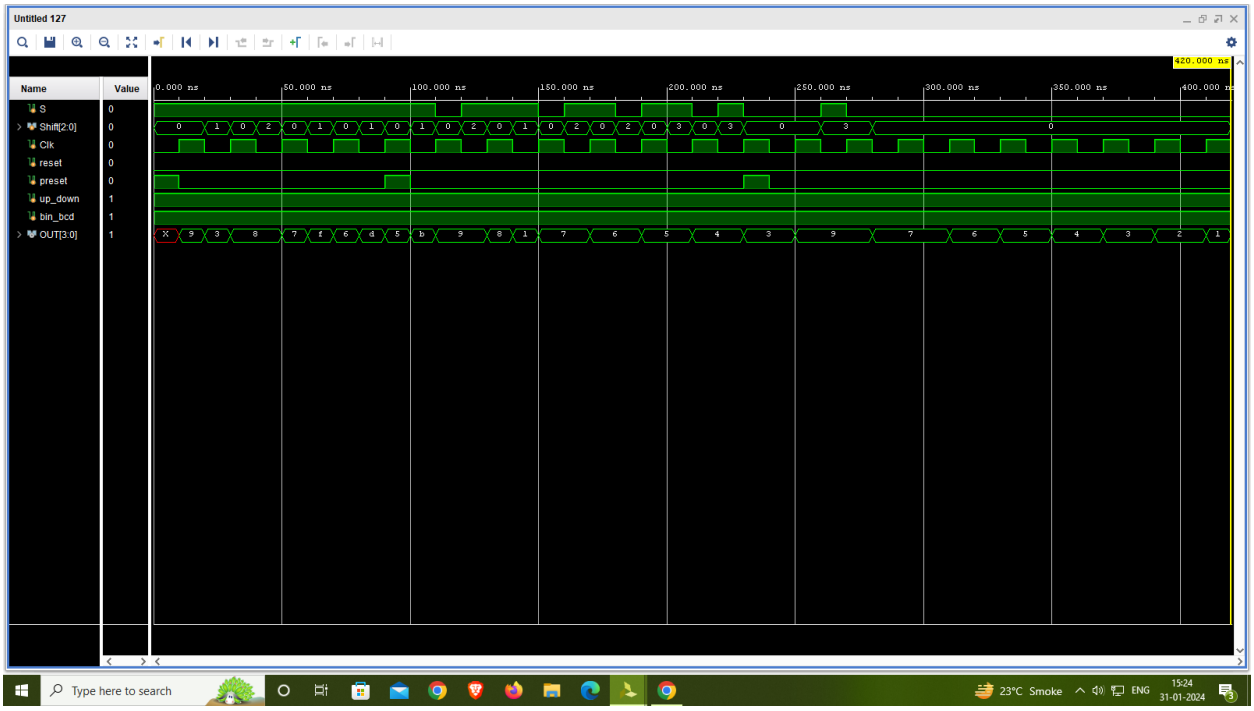
2. BCD Up Counter



3. Binary Down Counter



4. BCD Down Counter



We can see this matches the expected Output as we load the counter data and shift it in register.