

## 1. CLOCK DIVIDER.

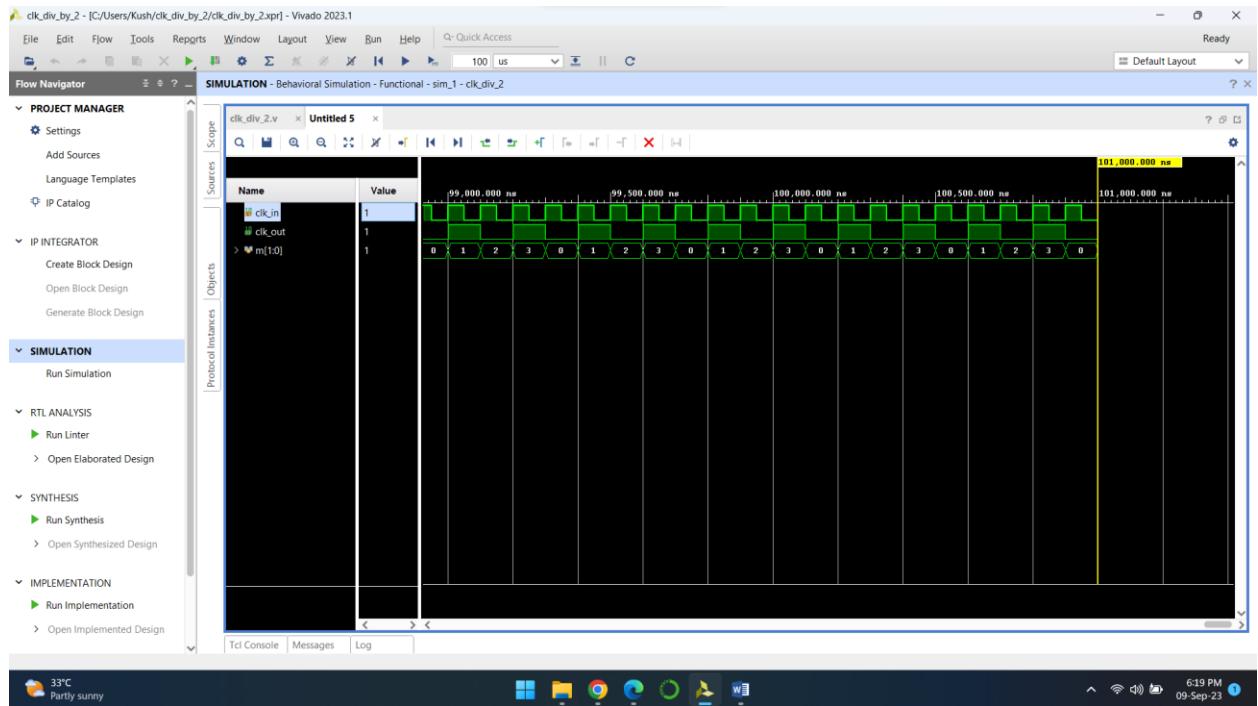
### CLOCK DIVIDER BY 2:

#### CODE:

The screenshot shows the Vivado 2023.1 interface with the project 'clk\_div\_by\_2' open. The 'SIMULATION' section is selected in the left sidebar. The main window displays the Verilog code for the 'clk\_div\_2' module. The code defines a module with an input 'clk\_in' and an output 'clk\_out'. It initializes 'm' to 0 and uses an always block to increment 'm' by 1 whenever the edge of 'clk\_in' is detected. The 'clk\_out' signal is assigned the value of 'm[0]'. The code is annotated with various comments starting with '///'.

```
// Module Name: clk_div_2
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
// Dependencies:
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
module clk_div_2(clk_in,clk_out);
    input clk_in;
    output clk_out;
    reg [1:0]m;
    initial m = 0;
    always @ (posedge clk_in)
        begin
            m=m+1;
        end
    assign clk_out = m[0];
endmodule
```

#### WAVEFORM:



## CLOCK DIVIDER 4: CODE:

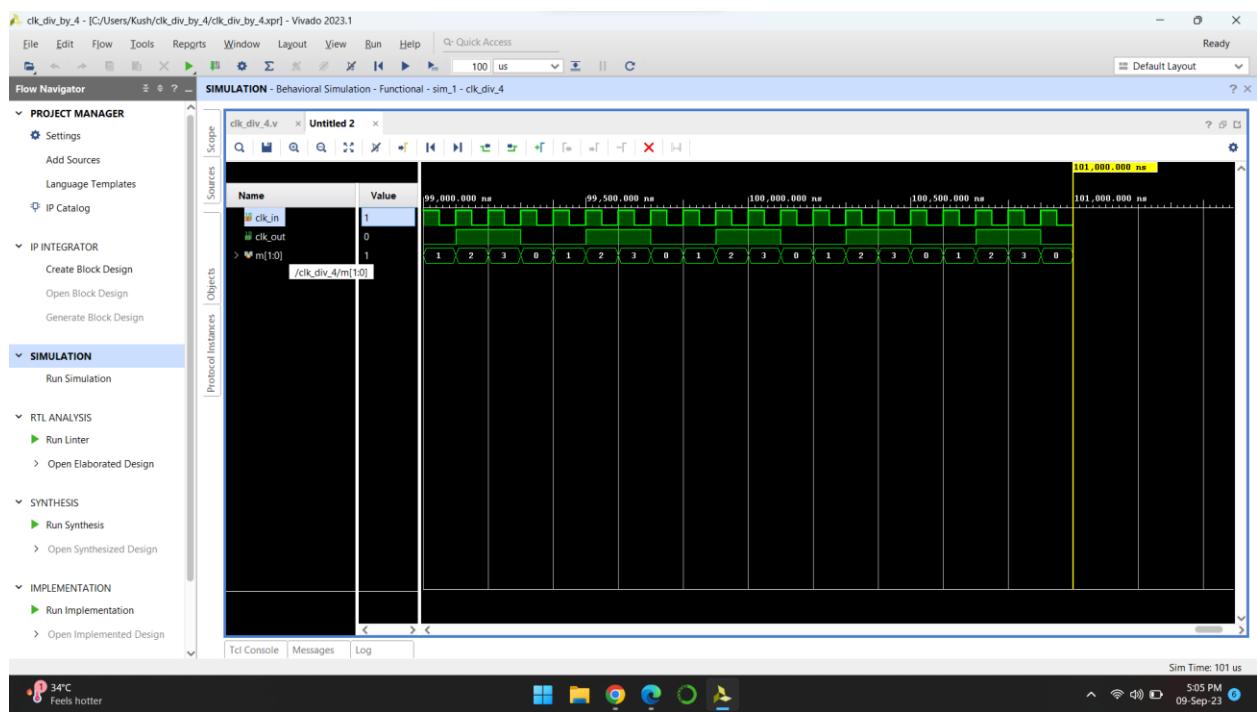
clk\_div\_4.v x Untitled 2

C:/Users/Kush/clk\_div\_by\_4/clk\_div\_by\_4.srsc/sources\_1/new/clk\_div\_4.v

Q | B | ← | → | X | // | █ | Q |

```
5 //  
6 // Create Date: 09/09/2023 04:55:35 PM  
7 // Design Name:  
8 // Module Name: clk_div_4  
9 // Project Name:  
10 // Target Devices:  
11 // Tool Versions:  
12 // Description:  
13 //  
14 // Dependencies:  
15 //  
16 // Revision:  
17 // Revision 0.01 - File Created  
18 // Additional Comments:  
19 //  
20 ///////////////////////////////////////////////////////////////////  
21  
22  
23 module clk_div_4(clk_in,clk_out);  
24 input clk_in;  
25 output clk_out;  
26 reg [1:0]m;  
27 initial m = 0;  
28 always @ (posedge clk_in)  
29 begin  
30 m=m+1;  
31 end  
32 assign clk_out = m[1];  
33  
34  
35 endmodule  
%e
```

#### WAVEFORM:

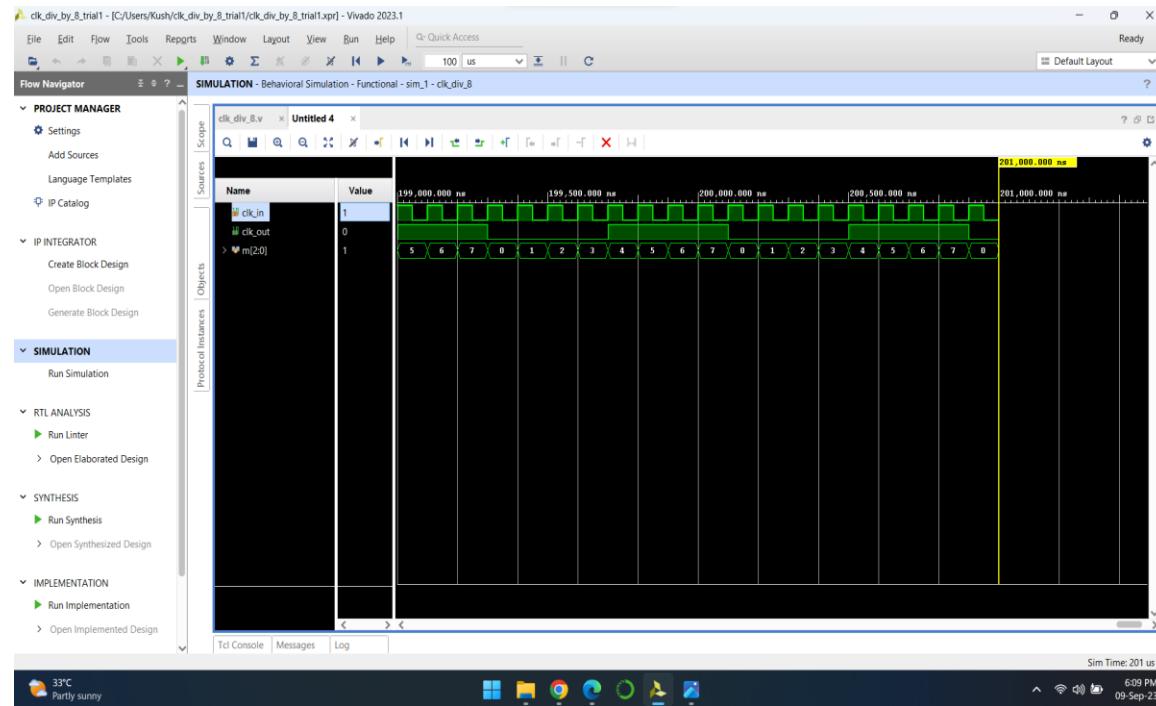


## CLOCK DIVIDER BY 8:

CODE:

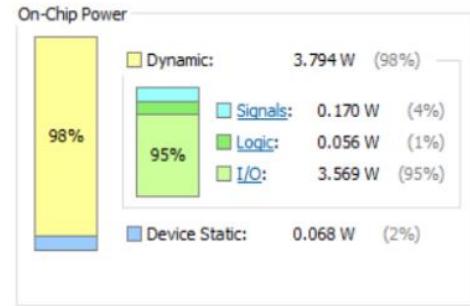
```
// Design Name:  
// Module Name: clk_div_8  
// Project Name:  
// Target Devices:  
// Tool Versions:  
// Description:  
// Dependencies:  
// Revision:  
// Revision 0.01 - File Created  
// Additional Comments:  
//  
///////////////////////////////////////////////////////////////////  
//  
module clk_div_8(clk_in,clk_out);  
    input clk_in;  
    output clk_out;  
    reg [2:0]m;  
    initial m = 0;  
    always @ (posedge clk_in)  
        begin  
            m=m+1;  
            end  
        assign clk_out = m[2];  
endmodule
```

WAVEFORM:



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

**Total On-Chip Power:** 3.862 W  
**Junction Temperature:** 43.5 °C  
 Thermal Margin: 56.5 °C (11.8 W)  
 Effective ΔJA: 4.8 °C/W  
 Power supplied to off-chip devices: 0 W  
 Confidence level: Low



## 2. JOHNSON COUNTER:

### CODE:

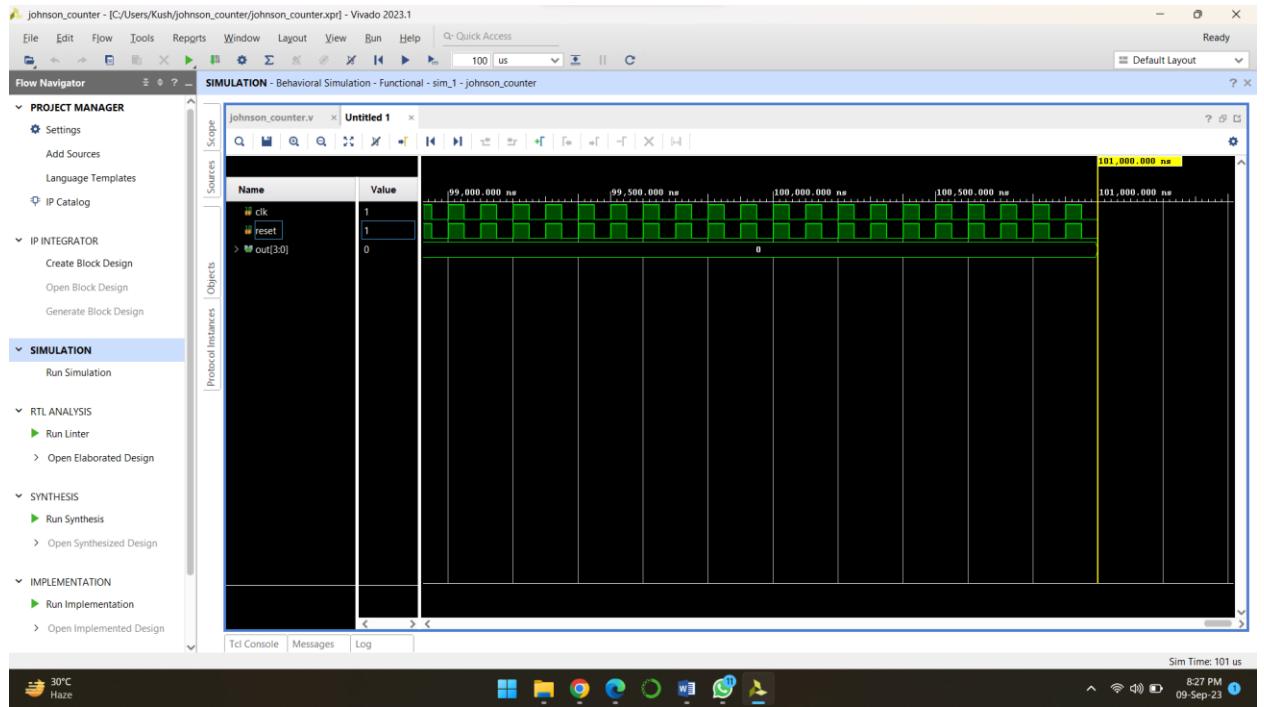
```

// Create Date: 09/09/2023 08:04:59 PM
// Design Name:
// Module Name: johnson_counter
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
// Dependencies:
// Revision:
// Revision 0.01 - File Created
// Additional comments:
// 
// 
// module johnson_counter(out,clk,reset);
//   input clk;
//   input reset;
//   output reg [3:0]out;
//   always @(posedge clk or posedge reset) begin
//     if (reset) begin
//       out = 4'b0000;
//     end else begin
//       out = {out[0], out[3:1]};
//     end
//   end
// endmodule

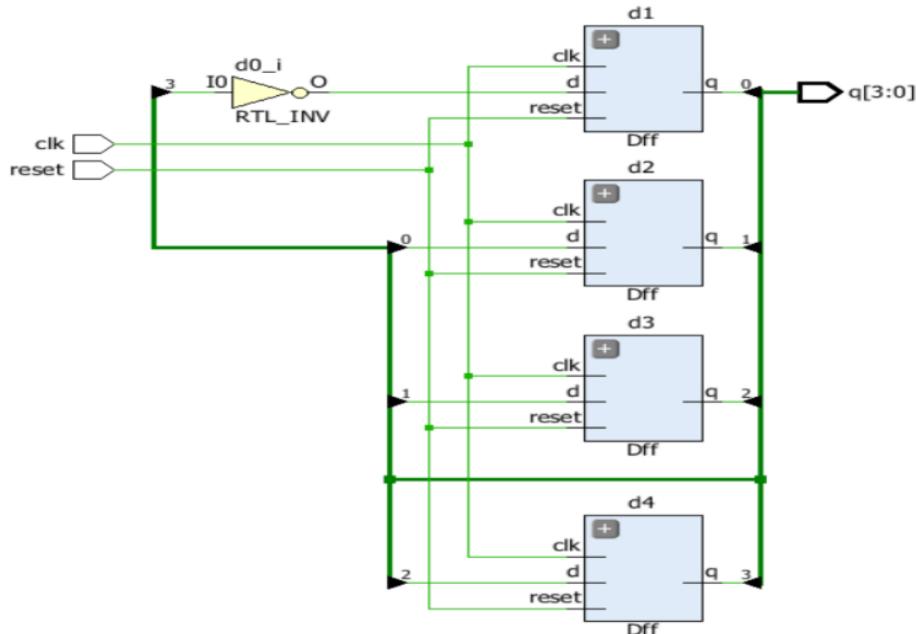
```

The screenshot shows the Vivado 2023.1 interface with the project manager open. The code for the Johnson Counter is displayed in the main editor window. The code is a Verilog module named 'johnson\_counter' with three ports: 'out', 'clk', and 'reset'. It includes a 'always' block that updates the 'out' variable based on the current state of 'out[0]' and the previous state of 'out[3:1]'.

## WAVEFORM:



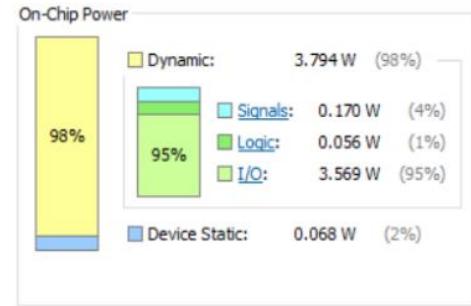
## RTL REPORT:



## POWER REPORT:

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

**Total On-Chip Power:** 3.862 W  
**Junction Temperature:** 43.5 °C  
 Thermal Margin: 56.5 °C (11.8 W)  
 Effective ΔJA: 4.8 °C/W  
 Power supplied to off-chip devices: 0 W  
 Confidence level: Low



### 3. RING COUNTER:

#### CODE:

```

// Dependencies:
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
// -----
// module ring_counter_tb(
// );
// reg clk;
// reg rst;
// wire [3:0]q;
//
// ring_counter ut(clk,rst,q);
//
// initial begin
// #0 clk=1'b0;
// #0 rst = 1'b0;
// end
// always
// #10 clk = ~clk;
// initial
// begin
// #10 rst =1'b1;
// #20 rst =1'b0;
// #500 $finish;
// end
// endmodule

```

The screenshot shows the Vivado IDE interface. The left sidebar displays the project structure under 'PROJECT MANAGER' and 'SIMULATION'. The main window shows the 'SIMULATION - Behavioral Simulation - Functional - sim\_1 - ring\_counter\_tb' tab, which contains the Verilog code for the ring counter testbench. The code includes dependencies, a module declaration, a reg declaration, a wire declaration, an initial block, an always block, and a \$finish statement. The status bar at the bottom right indicates a simulation time of 200530 ns.

#### TESTBENCH:

**RING COUNTER - [C:/Users/Kush/RING COUNTER/RING COUNTER.xpr] - Vivado 2023.1**

File Edit Flow Tools Reports Window Layout View Run Help Q-Quick Access

200 us

SIMULATION - Behavioral Simulation - Functional - sim\_1 - ring\_counter\_tb

Project Manager IP Integrator Simulation RTL Analysis Synthesis Implementation

**SIMULATION**

Run Simulation

RTL Codeline: RING\_Counter.v

```

// Description:
// Dependencies:
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//////////////////////////////////////////////////////////////////
module ring_counter(
    input clk,
    input rst,
    output reg [3:0] q
);
    //seq [3:0]q;
    always @(posedge clk)
        begin
            if(rst==1)
                q <= 4'b0001;
            else
                begin
                    q[0]<=q[3];
                    q[1]<=q[0];
                    q[2]<=q[1];
                    q[3]<=q[2];
                end
        end
endmodule

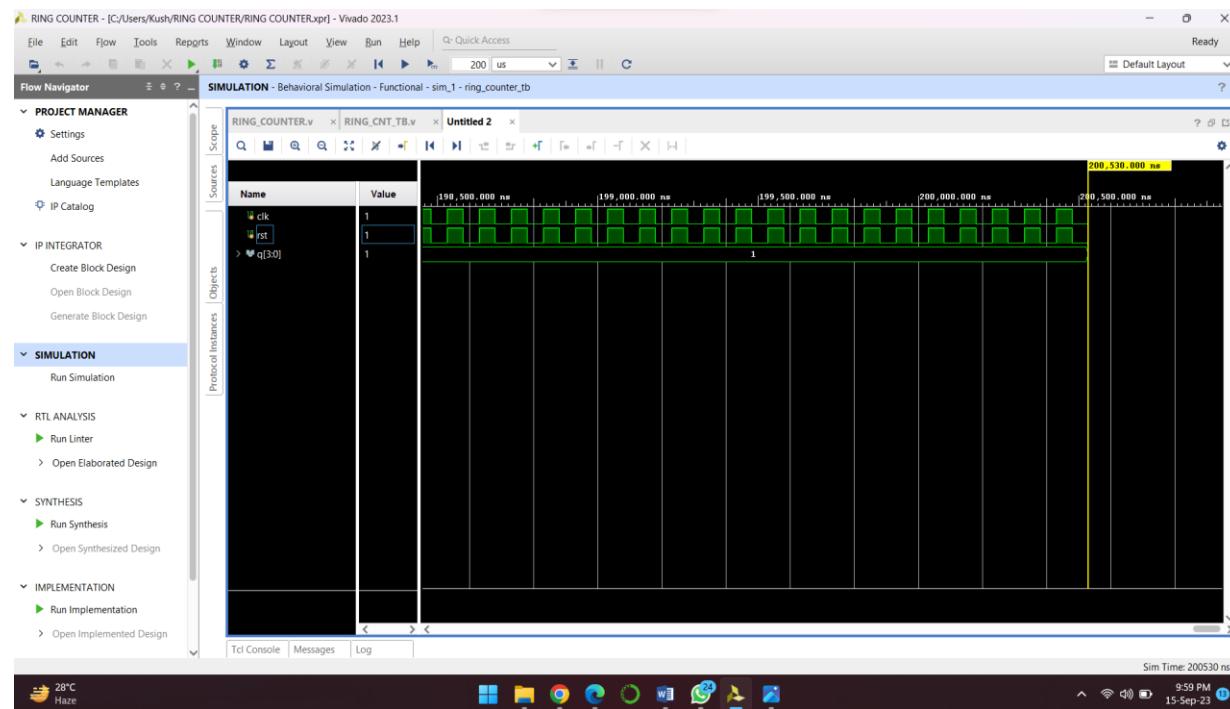
```

Tcl Console Messages Log

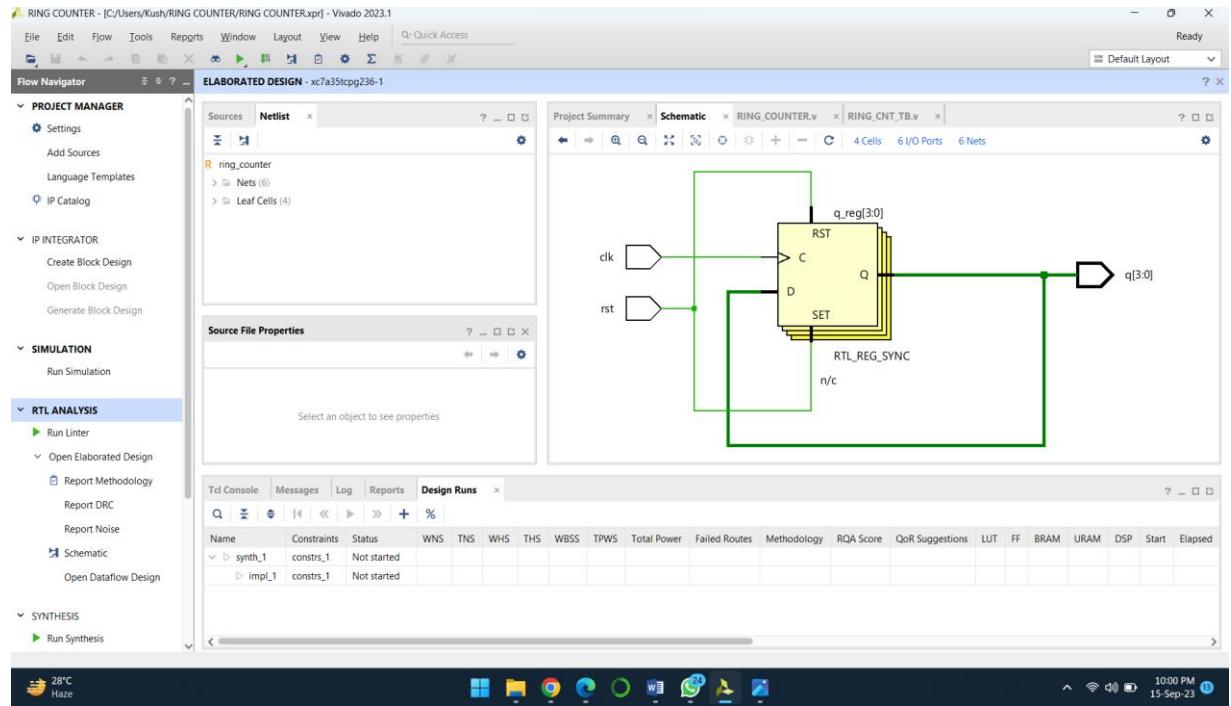
Sim Time: 200530 ns

28°C Haze

## WAVEFORM:



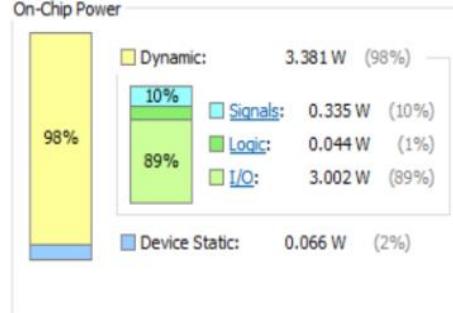
## RTL SCHEMATIC:



## POWER REPORT:

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

**Total On-Chip Power:** 3.448 W  
**Junction Temperature:** 41.5 °C  
 Thermal Margin: 58.5 °C (12.2 W)  
 Effective ΔJA: 4.8 °C/W  
 Power supplied to off-chip devices: 0 W  
 Confidence level: Low



## 4. 5 INPUT MAJORITY CIRCUIT

CODE:

**TESTBENCH:**

```

FIVE_IN_MAJ_CKT.v | FIVE_IN_MAJ_CKT_TB.v
C:/Users/Kush/FIVE_INP... Close CKT/FIVE_INPUT_MAJ_CKT.srs/sources_1/new/FIVE_IN_MAJ_CKT.v

10 // Target Devices;
11 // Tool Versions;
12 // Description;
13 //
14 // Dependencies;
15 //
16 // Revision;
17 // Revision 0.01 - File Created
18 // Additional Comments;
19 //
20 ///////////////////////////////////////////////////////////////////
21 //
22 //
23 module input_majority(
24   input [4:0] a,
25   output reg z
26 );
27   reg [3:0]count;
28   always @ (a) begin
29     count = 3'b000;
30     for (integer i = 0; i < 5; i = i + 1) begin
31       if (a[i] == 1'b1)
32         count = count + 1;
33     end
34     $display("%b",count);
35     if (count >= 3'b011)
36       z = 1;
37     else
38       z = 0;
39   end
40 endmodule

```

**WAVEFORM:**

```

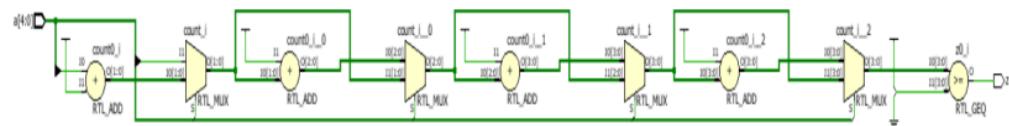
FIVE_IN_MAJ_CKT.v | FIVE_IN_MAJ_CKT_TB.v
C:/Users/Kush/FIVE_INP... Close CKT/FIVE_INPUT_MAJ_CKT.srs/sources_1/new/FIVE_IN_MAJ_CKT.v

11 // Tool Versions;
12 // Description;
13 //
14 // Dependencies;
15 //
16 // Revision;
17 // Revision 0.01 - File Created
18 // Additional Comments;
19 //
20 ///////////////////////////////////////////////////////////////////
21 //
22 //
23 module input_majority_tb;
24   wire a;
25   wire z;
26   // Instantiate the module under test
27   input_majority uut (
28     .a(a),
29     .z(z)
30   );
31   initial begin
32     a = 5'b00000;
33     #10 a = 5'b00001;
34     #10 a = 5'b0011;
35     #10 a = 5'b0111;
36     #10 a = 5'b1111;
37     #10 a = 5'b00011;
38     #10
39     $finish; // Finish simulation
40   end
41 endmodule

```



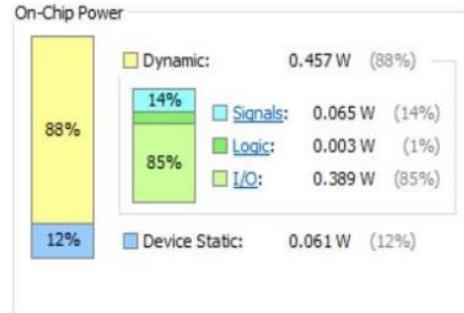
### RTL SCHEMATIC:



### POWER REPORT:

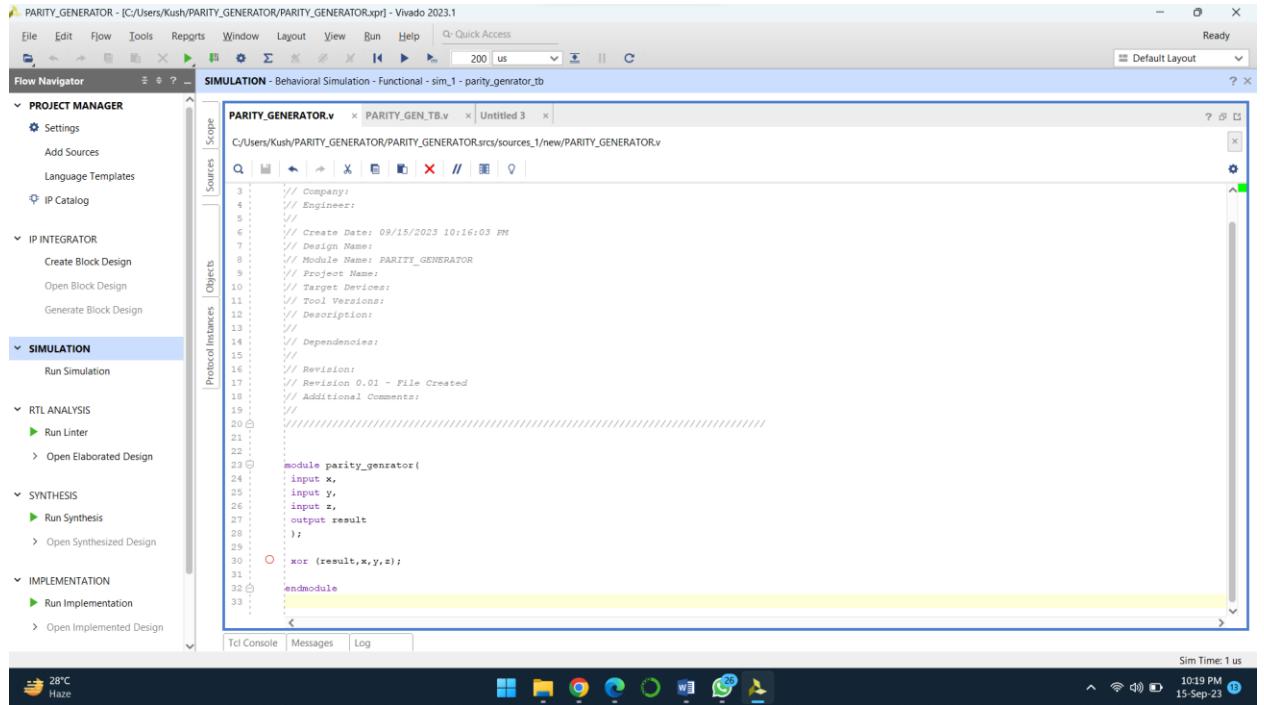
Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

**Total On-Chip Power:** 0.518 W  
**Junction Temperature:** 27.5 °C  
 Thermal Margin: 72.5 °C (15.1 W)  
 Effective  $\delta$ JA: 4.8 °C/W  
 Power supplied to off-chip devices: 0 W  
 Confidence level: Low



## 5. PARITY GENERATOR:

### CODE:



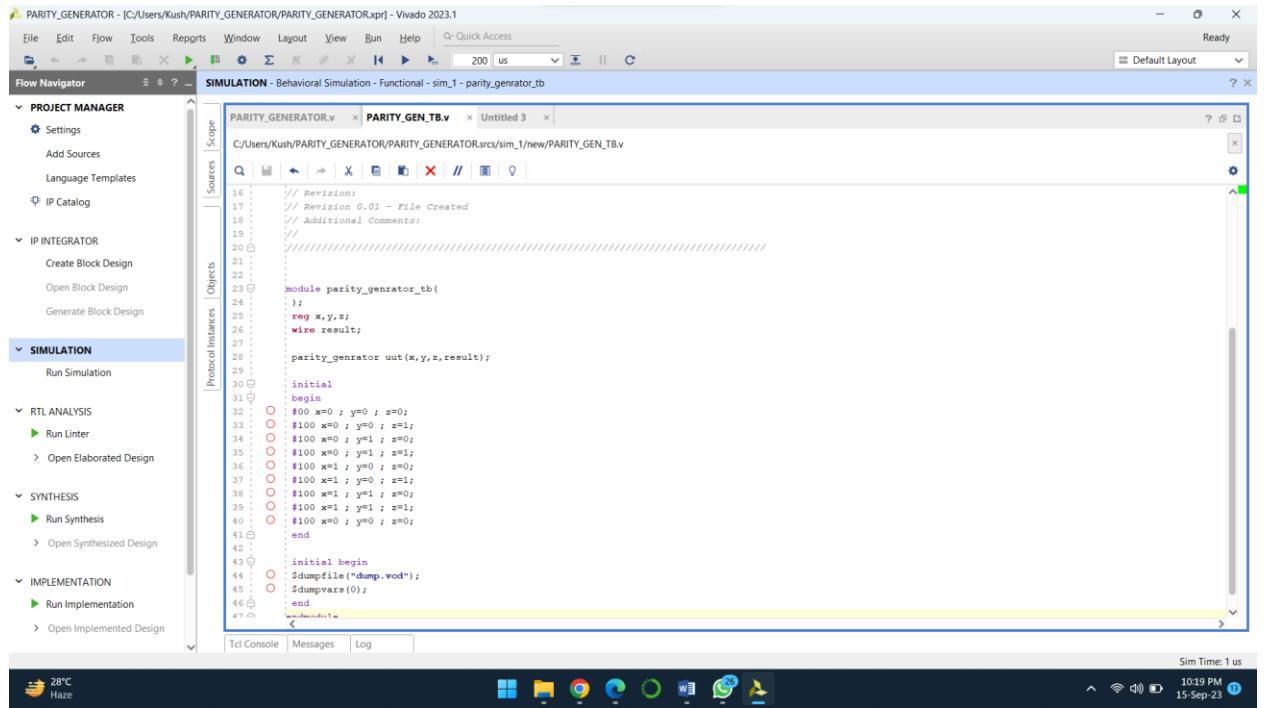
```

PARITY_GENERATOR.v  x PARITY_GEN_TB.v  x Untitled 3  x
C:/Users/Kush/PARITY_GENERATOR/PARITY_GENERATOR/srcs/sources_1/new/PARITY_GENERATOR.v

3 // Company:
4 // Engineer:
5 //
6 // Create Date: 09/15/2023 10:16:03 PM
7 // Design Name:
8 // Module Name: PARITY_GENERATOR
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21 //
22 //
23 module parity_generator(
24     input x,
25     input y,
26     input z,
27     output result
28 );
29 
30     xor (result,x,y,z);
31 
32 endmodule
33 
34

```

### TESTBENCH:



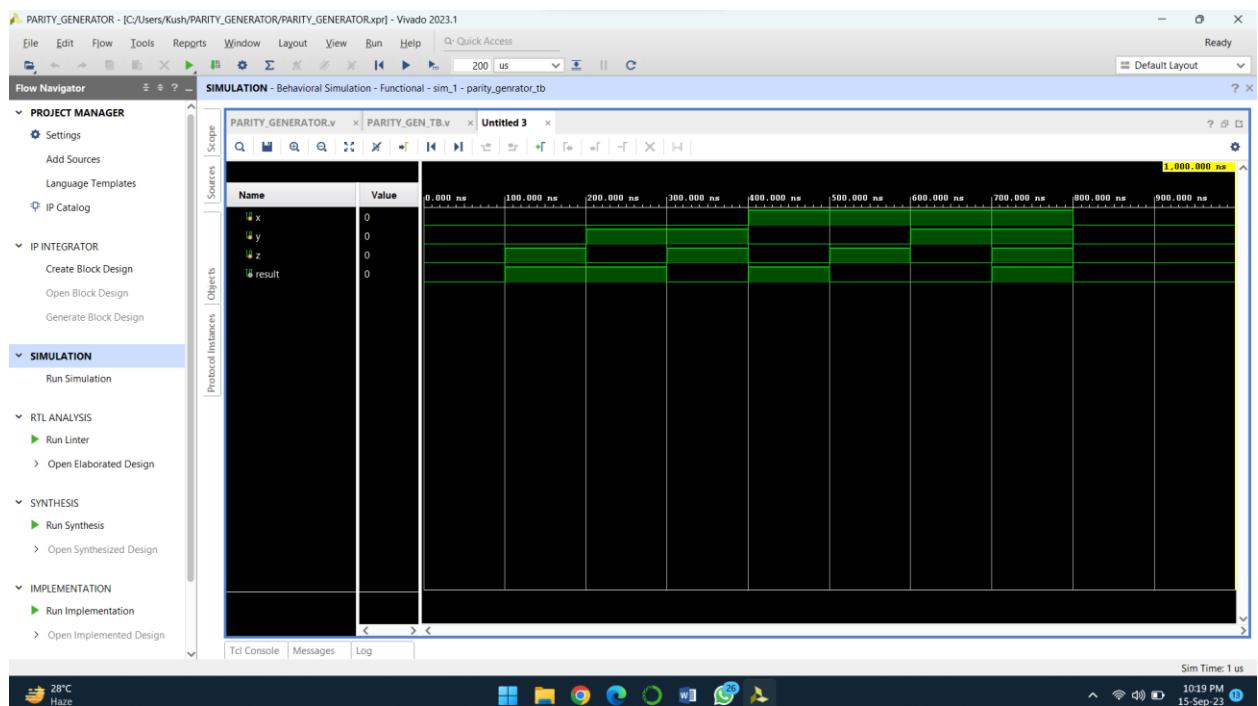
```

PARITY_GENERATOR.v  x PARITY_GEN_TB.v  x Untitled 3  x
C:/Users/Kush/PARITY_GENERATOR/PARITY_GENERATOR/srcs/sim_1/new/PARITY_GEN_TB.v

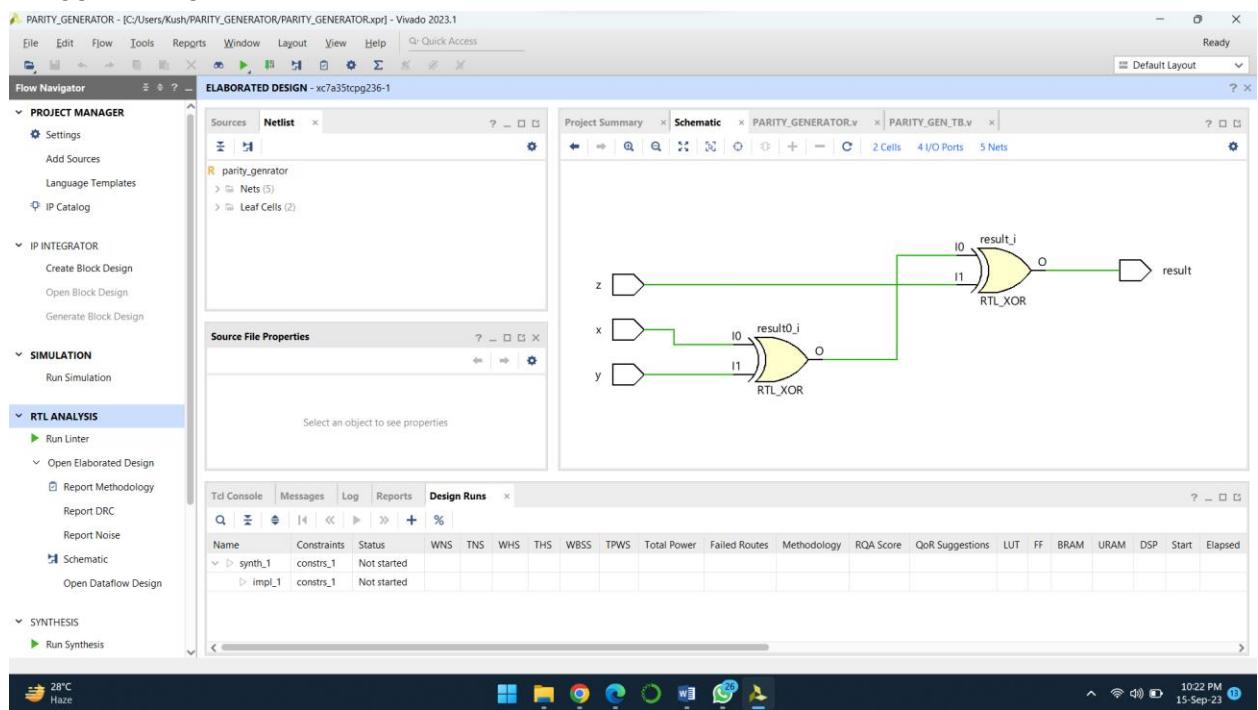
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //
21 //
22 //
23 module parity_generator_tb();
24     reg x,y,z;
25     wire result;
26 
27     parity_generator uut(x,y,z,result);
28 
29     initial
30     begin
31         $00 x=0 ; y=0 ; z=0;
32         $100 x=0 ; y=0 ; z=1;
33         $0100 x=0 ; y=1 ; z=0;
34         $010 x=0 ; y=1 ; z=1;
35         $100 x=1 ; y=0 ; z=0;
36         $100 x=1 ; y=0 ; z=1;
37         $100 x=1 ; y=1 ; z=0;
38         $100 x=1 ; y=1 ; z=1;
39         $100 x=0 ; y=0 ; z=0;
40         $100 x=0 ; y=0 ; z=1;
41     end
42 
43     initial begin
44         $dumpfile("dump.vcd");
45         $dumpvars(0);
46     end
47 end

```

### WAVEFORM:



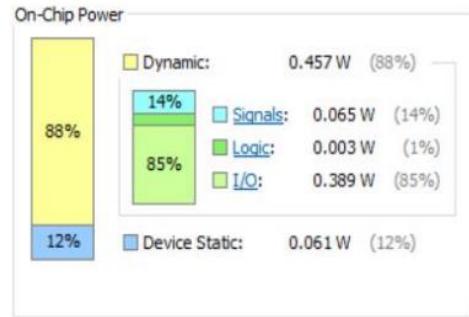
## RTL SCHEMATIC:



## POWER REPORT:

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

**Total On-Chip Power:** 0.518 W  
**Junction Temperature:** 27.5 °C  
 Thermal Margin: 72.5 °C (15.1 W)  
 Effective ΔJA: 4.8 °C/W  
 Power supplied to off-chip devices: 0 W  
 Confidence level: Low



## 6. BINARY TO ONE HOT ENCODER:

### CODE:

```

1: `timescale 1ns / 1ps
2: // Company:
3: // Engineer:
4: //
5: // Create Date: 09/15/2023 10:25:09 PM
6: // Design Name: BINARY_TO_ONE_HOT_ENCODER
7: // Module Name: BINARY_TO_ONE_HOT_ENCODER
8: // Project Name:
9: // Target Devices:
10: // Tool Versions:
11: // Description:
12: //
13: // Dependencies:
14: //
15: // Revision:
16: // Revision 0.01 - File Created
17: // Additional Comments:
18: //
19: // 
20: ///////////////////////////////////////////////////////////////////
21: //
22: //
23: module Binary_too_One_Hot_encoder(
24:     input [3:0] a,
25:     output [15:0] b
26: );
27:     assign b = 1'b<>a;
28: endmodule
29: 
```

### TESTBENCH:

**ELABORATED DESIGN - xc7a35tcpg236-1**

```

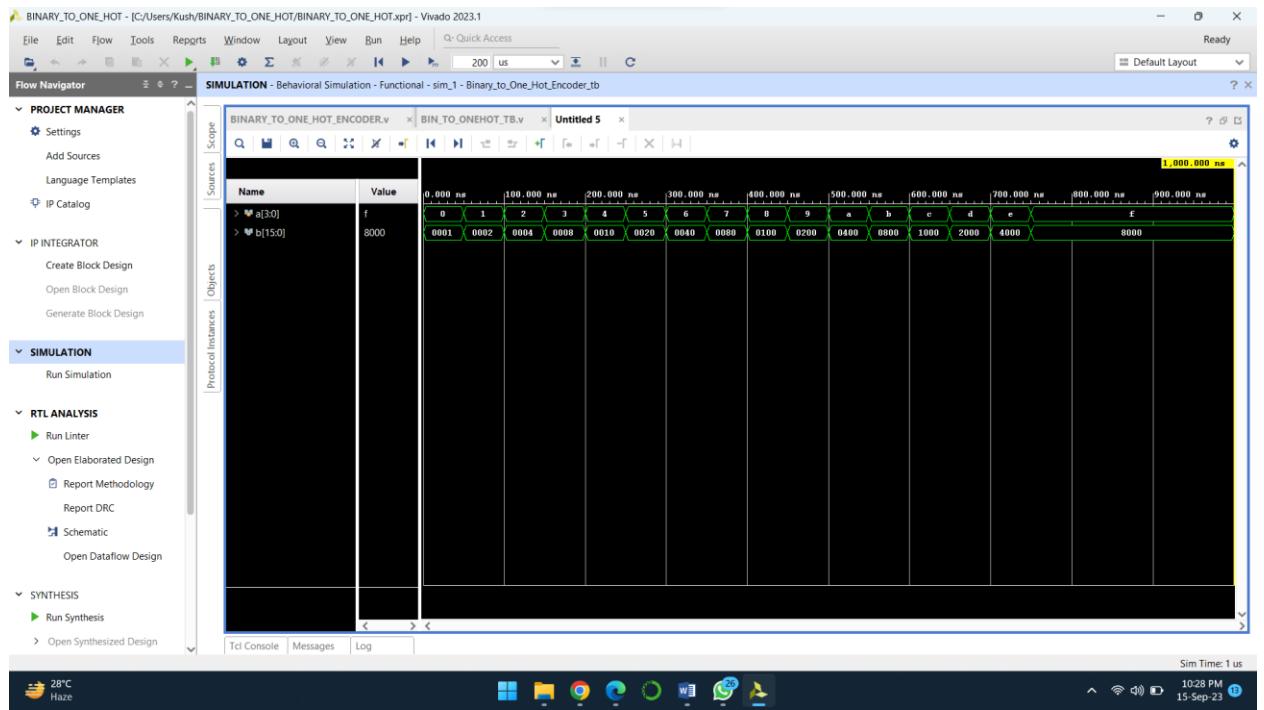
module Binary_to_One_Hot_Encoder_tb;
  reg [3:0] a;
  wire [15:0] b;

  Binary_to_One_Hot_encoder uut(a,b);

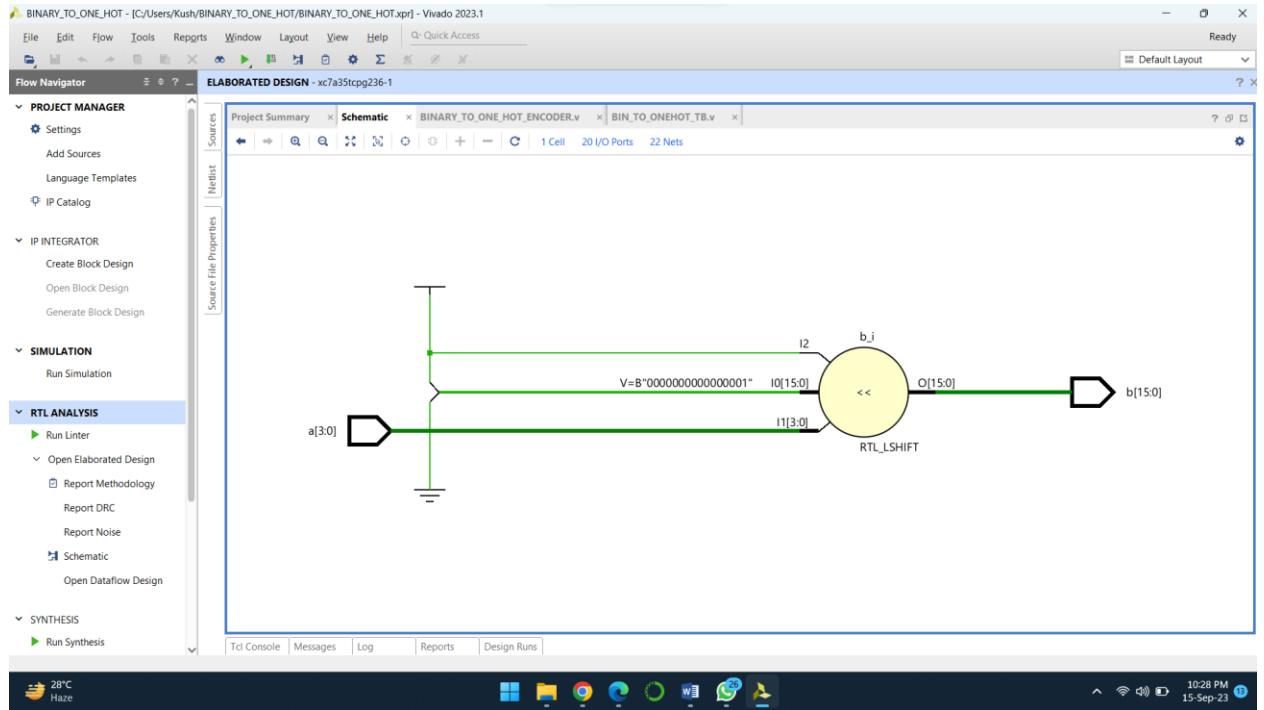
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0);
  end
  initial begin
    a=4'b0000;
    #50 a=4'b0001;
    #50 a=4'b0110;
    #50 a=4'b0111;
    #50 a=4'b1100;
    #50 a=4'b1101;
    #50 a=4'b1110;
    #50 a=4'b1111;
    #50 a=4'b0000;
    #50 a=4'b0001;
    #50 a=4'b0100;
    #50 a=4'b0101;
    #50 a=4'b1100;
    #50 a=4'b1101;
    #50 a=4'b1110;
    #50 a=4'b1111;
  end
endmodule

```

## WAVEFORM:



## RTL REPORT:

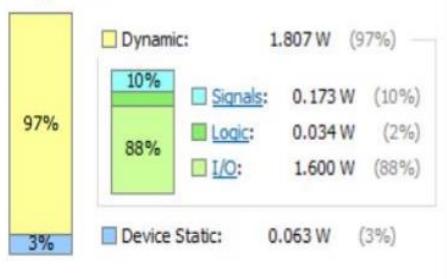


## POWER REPORT:

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

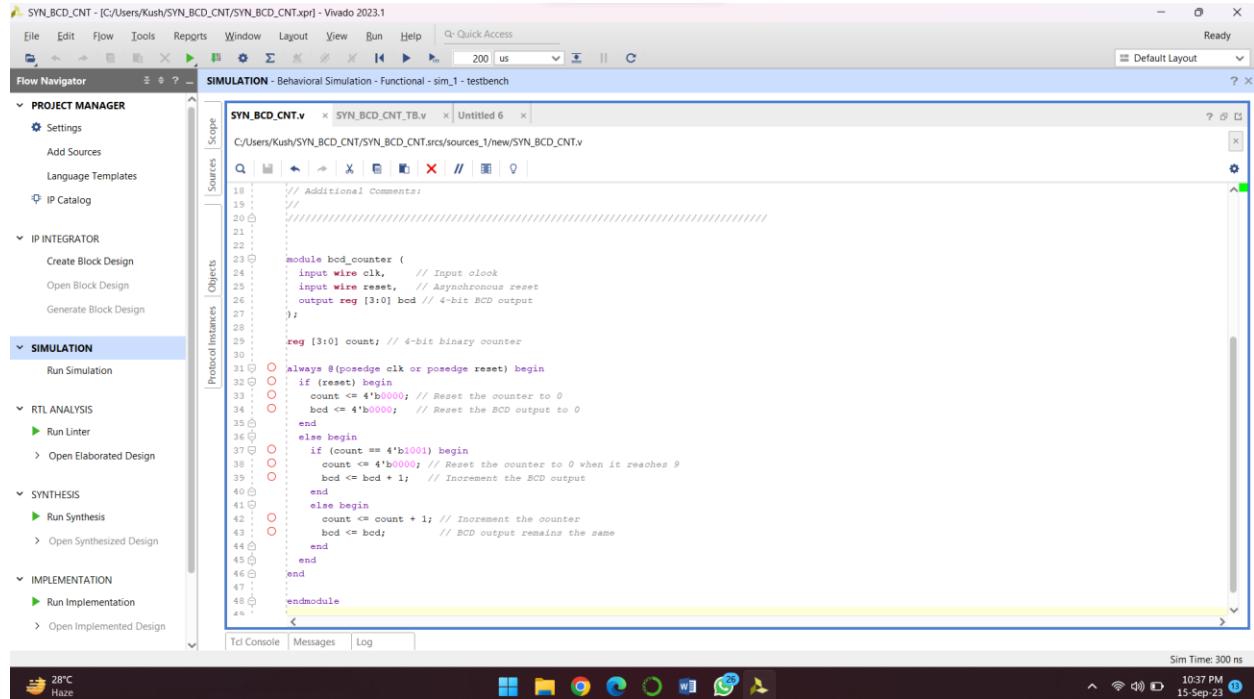
**Total On-Chip Power:** 1.87 W  
**Junction Temperature:** 33.9 °C  
 Thermal Margin: 66.1 °C (13.7 W)  
 Effective  $\theta_{JA}$ : 4.8 °C/W  
 Power supplied to off-chip devices: 0 W  
 Confidence level: Low

On-Chip Power



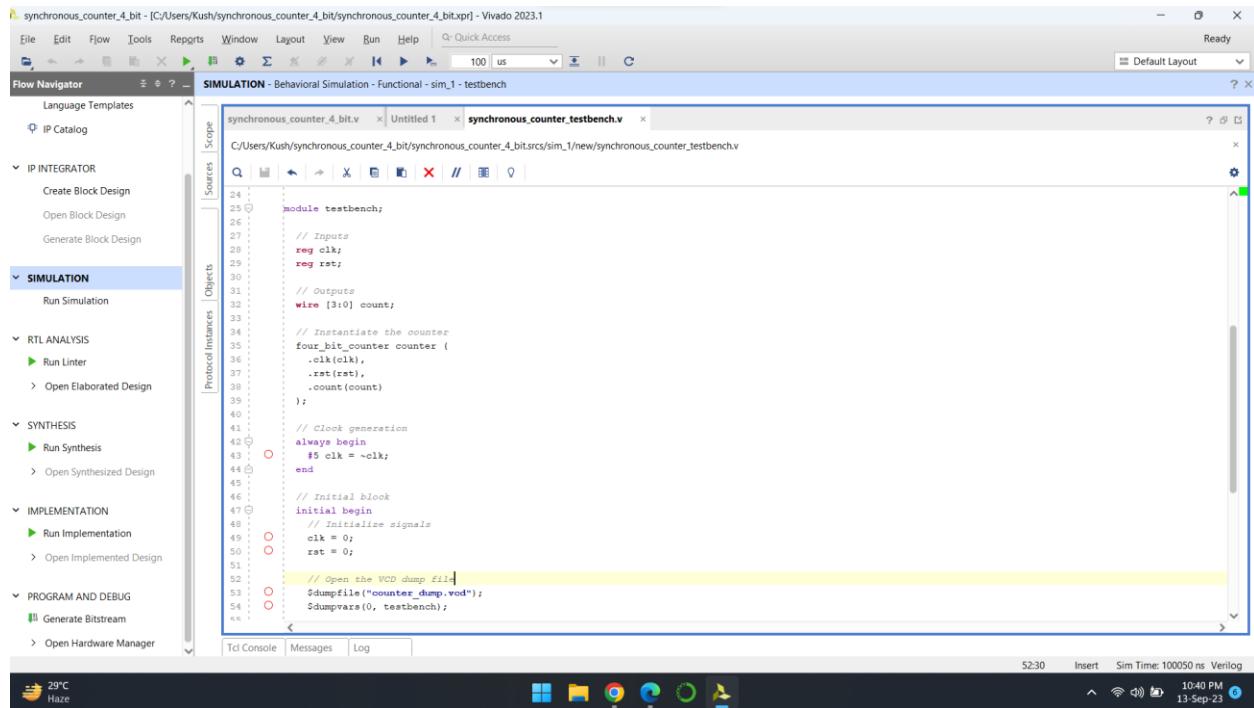
## 7. 4-Bit Synchronous BCD Counter:

### CODE:



```
// Additional Comments:  
//  
//  
module bcd_counter (  
    input wire clk, // Input clock  
    input wire reset, // Asynchronous reset  
    output reg [3:0] bcd // 4-bit BCD output  
);  
  
reg [3:0] count; // 4-bit binary counter  
  
always @ (posedge clk or posedge reset) begin  
    if (reset) begin  
        count <= 4'b0000; // Reset the counter to 0  
        bcd <= 4'b0000; // Reset the BCD output to 0  
    end  
    else begin  
        if (count == 4'b1001) begin  
            count <= 4'b0000; // Reset the counter to 0 when it reaches 9  
            bcd <= bcd + 1; // Increment the BCD output  
        end  
        else begin  
            count <= count + 1; // Increment the counter  
            bcd <= bcd; // BCD output remains the same  
        end  
    end  
endmodule
```

### TESTBENCH:



```
module testbench;  
// Inputs  
reg clk;  
reg rst;  
  
// Outputs  
wire [3:0] count;  
  
// Instantiate the counter  
four_bit_counter counter (  
    .clk(clk),  
    .rst(rst),  
    .count(count)  
);  
  
// Clock generation  
always begin  
    #5 clk = ~clk;  
end  
  
// Initial block  
initial begin  
    // Initialize signals  
    clk = 0;  
    rst = 0;  
  
    // Open the VCD dump file  
    $dumpfile("counter_dump.vcd");  
    $dumparam(0, testbench);  
end
```

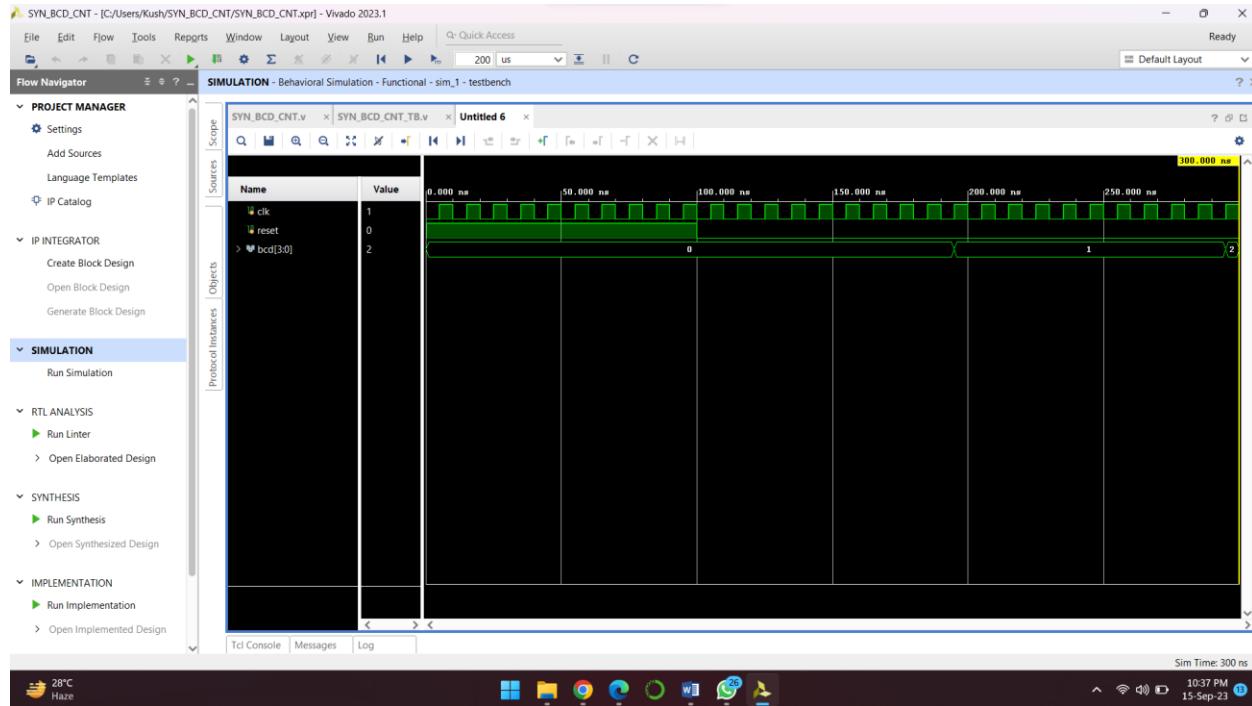
The screenshot shows the Vivado 2023.1 interface with the following details:

- Title Bar:** synchronous\_counter\_4\_bit - [C:/Users/Kush/synchronous\_counter\_4\_bit/synchronous\_counter\_4\_bit.xpr] - Vivado 2023.1
- Menu Bar:** File, Edit, Fjow, Tools, Repgts, Window, Layout, View, Run, Help, Quick Access
- Toolbars:** Flow Navigator, Language Templates, IP Catalog, IP INTEGRATOR, Create Block Design, Open Block Design, Generate Block Design, SIMULATION (Run Simulation), RTL ANALYSIS (Run Linter, Open Elaborated Design), SYNTHESIS (Run Synthesis, Open Synthesized Design), IMPLEMENTATION (Run Implementation, Open Implemented Design), PROGRAM AND DEBUG (Generate Bitstream, Open Hardware Manager).
- Simulator View:** SIMULATION - Behavioral Simulation - Functional - sim\_1 - testbench. The code window displays the following Verilog testbench script:

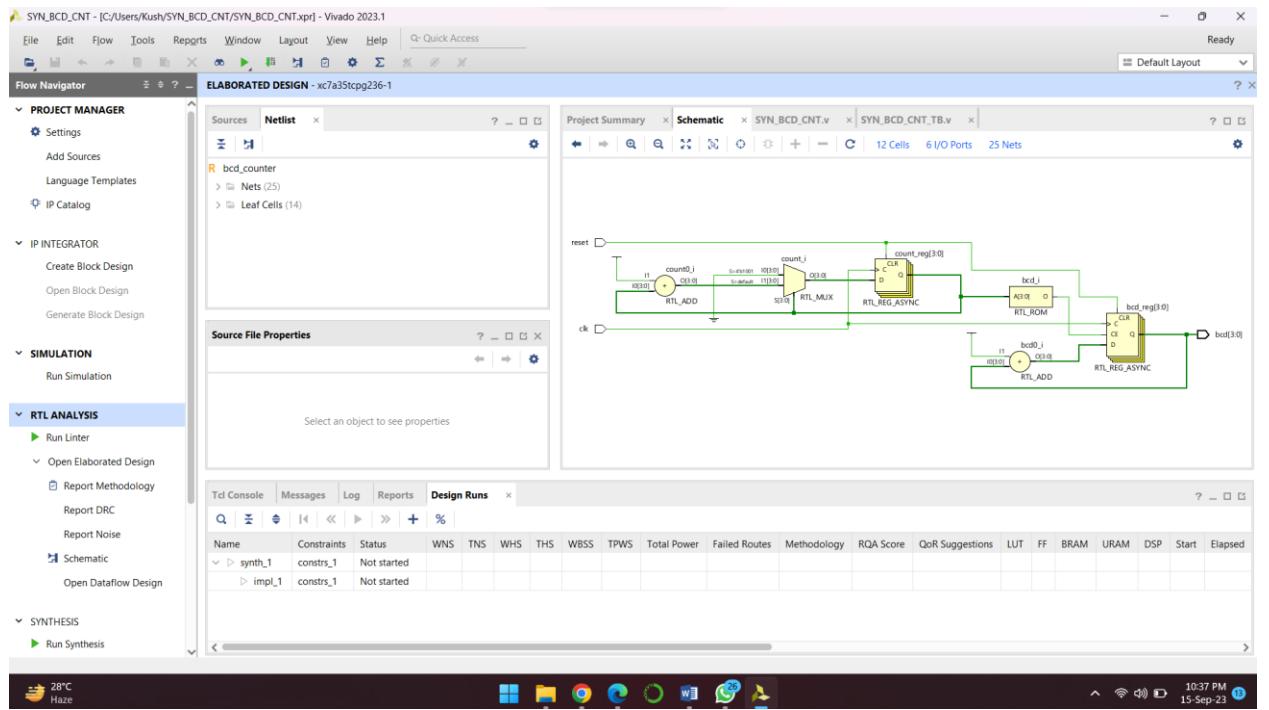
```
synchronous.counter_4_bit.v * Untitled 1 * synchronous_counter_testbench.v
C:/Users/Kush/synchronous_counter_4_bit/synchronous_counter_4_bit.srcs/sim_1/new/synchronous_counter_testbench.v

45 // Initial block
46 initial begin
47     // Initialize signals
48     clk = 0;
49     rst = 0;
50
51     // Open the VCD dump file
52     $dumpfile("counter_dump.vcd");
53     $dumpvars(0, testbench);
54
55     // Apply reset
56     rst = 1;
57     #10 rst = 0;
58
59     // Perform some clock cycles and observe the counter
60     #20;
61     $display("Counter Value: %b", count);
62
63     #10;
64     $display("Counter Value: %b", count);
65
66     #10;
67     $display("Counter Value: %b", count);
68
69     // Finish simulation
70     $finish;
71 end
72
73
74 endmodule
```
- Bottom Status Bar:** 29°C Haze, 10:40 PM, 13-Sep-23, Insert, Verilog.

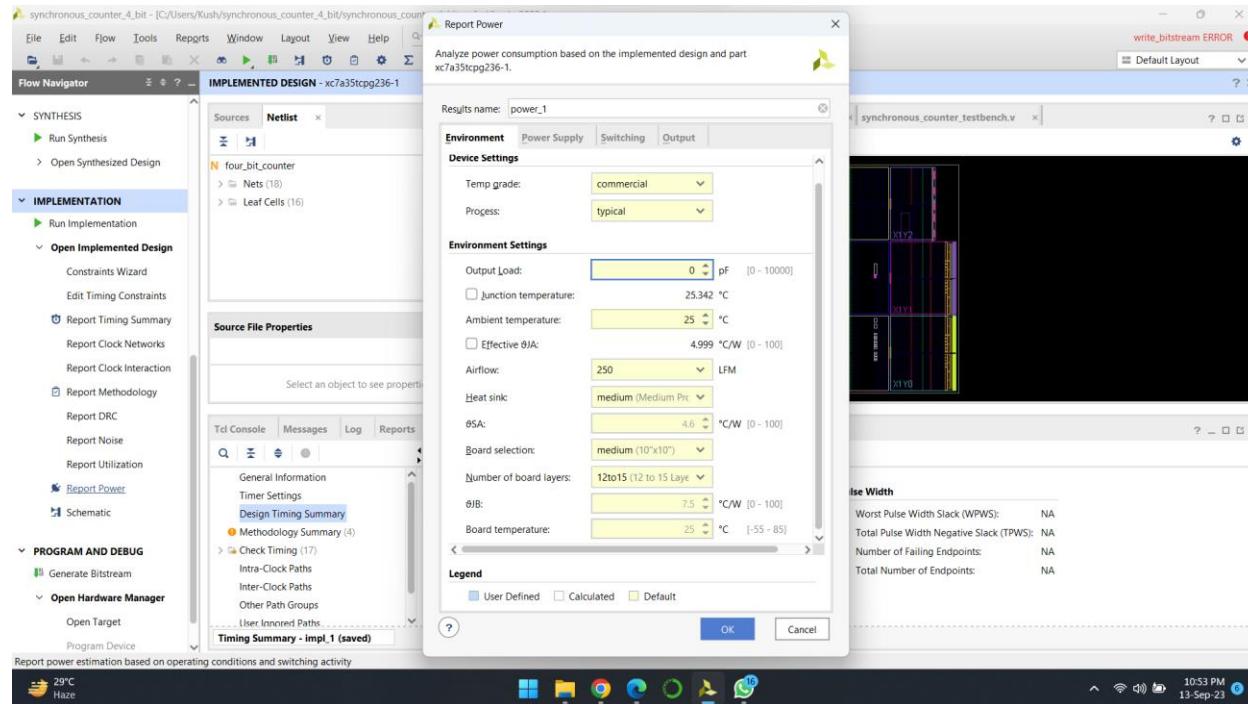
## WAVEFORM:

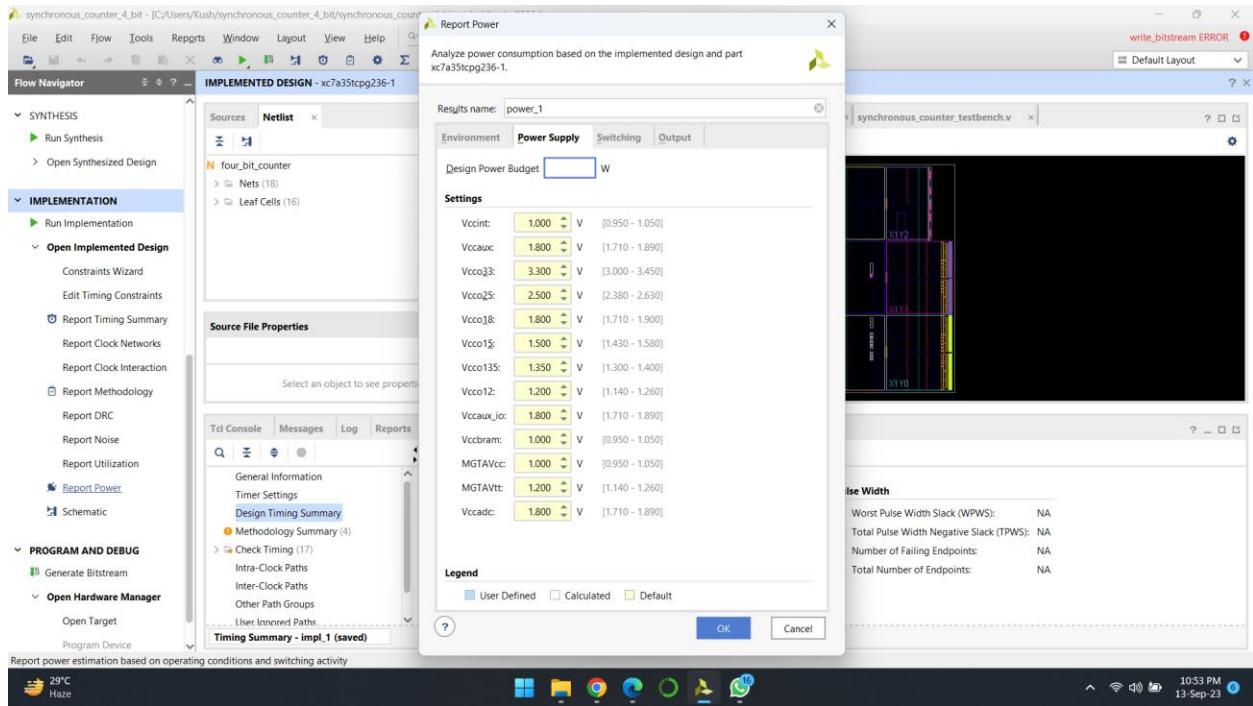


## RTL SCHEMATIC:



## POWER REPORT:





## 8. 4-BIT LOOK AHEAD ADDER:

CODE:

The screenshot shows the Vivado 2023.1 interface with the following windows:

- Flow Navigator**: Shows project structure with sections like SYNTHESIS, IMPLEMENTATION, and PROGRAM AND DEBUG.
- IMPLEMENTED DESIGN - xc7a35tcpg236-1**: A code editor window displaying Verilog code for a 4-bit look-ahead adder. The code includes comments and logic for generating intermediate carry signals and summing them to produce the final sum and carry-out.

```

15 // Revision:
16 // Revision 0.01 - File Created
17 // Additional Comments:
18 /**
19  * module carry_lookahead_adder (
20  *   input wire [3:0] A, // 4-bit input A
21  *   input wire [3:0] B, // 4-bit input B
22  *   input wire Cin, // Carry-in
23  *   output wire [3:0] Sum, // 4-bit output Sum
24  *   output wire Cout // Carry-out
25  * );
26  *
27  *   // Generate intermediate carry signals
28  *   wire [3:0] G, P;
29  *   assign G = A & B;
30  *   assign P = A ^ B;
31  *
32  *   // Generate Sum and Carry-out signals
33  *   assign Sum[0] = A[0] ^ B[0] ^ Cin;
34  *   assign Sum[1] = P[0] ^ Cin;
35  *   assign Sum[2] = P[1] ^ (G[0] & Cin);
36  *   assign Sum[3] = P[2] ^ (G[1] & Cin);
37  *
38  *   assign Cout = G[2] | (G[1] & P[2]) | (G[0] & P[1]) | (A[3] & B[3] & Cin);
39  *
40  * );
41  *
42  * endmodule
43  */

```

## TESTBENCH:

```

module testbench;
    // Inputs
    reg [3:0] A;
    reg [3:0] B;
    reg Cin;

    // Outputs
    wire [3:0] Sum;
    wire Cout;

    // Instantiate the Carry Look-Ahead Adder
    carryLookahead_adder uut (
        .A(A),
        .B(B),
        .Cin(Cin),
        .Sum(Sum),
        .Cout(Cout)
    );

    // Clock generation (not used in this testbench)
    reg clk;
    always begin
        #5 clk = ~clk;
    end

    // Initial block
    initial begin
        // Initialize inputs
        A = 4'b0000;
        B = 4'b0000;
        Cin = 0;
    end

```

```

#5 clk = ~clk;
end

// Initial block
initial begin
    // Initialize inputs
    A = 4'b0000;
    B = 4'b0000;
    Cin = 0;

    // Open the VCD dump file
    $dumpfile("CLA_dump.vcd");
    $dumpvars(0, testbench);

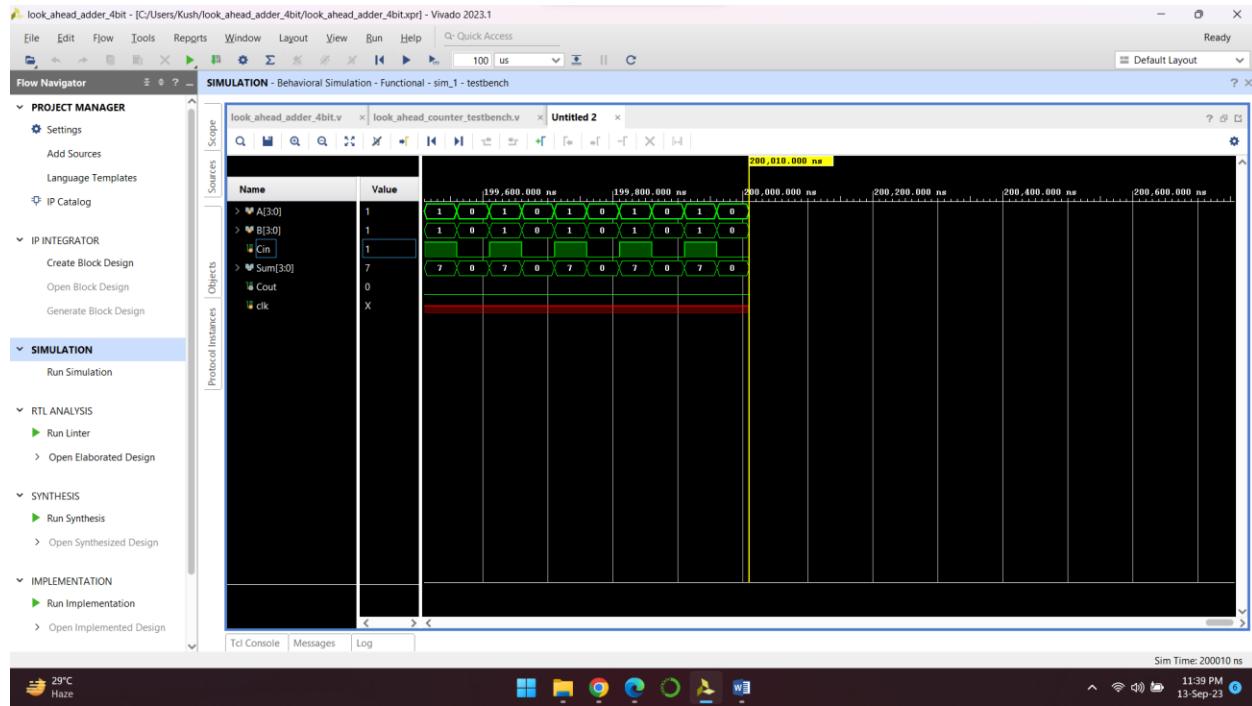
    // Apply test vectors
    A = 4'b1101;
    B = 4'b1010;
    Cin = 0;

    // Simulate for a few time units
    #10;

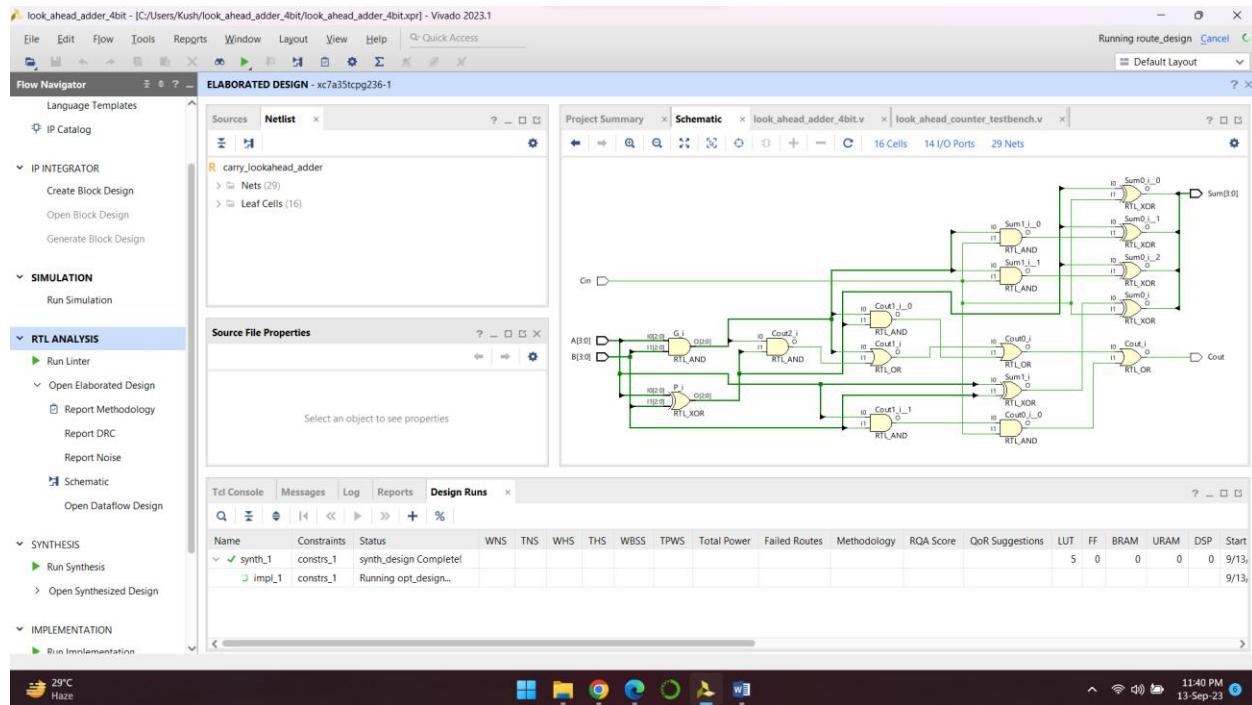
    // Display results
    $display("Inputs: A=%b, B=%b, Cin=%b", A, B, Cin);
    $display("Sum: %b, Cout: %b", Sum, Cout);
    $display("Finish simulation");
    $finish;
end
endmodule

```

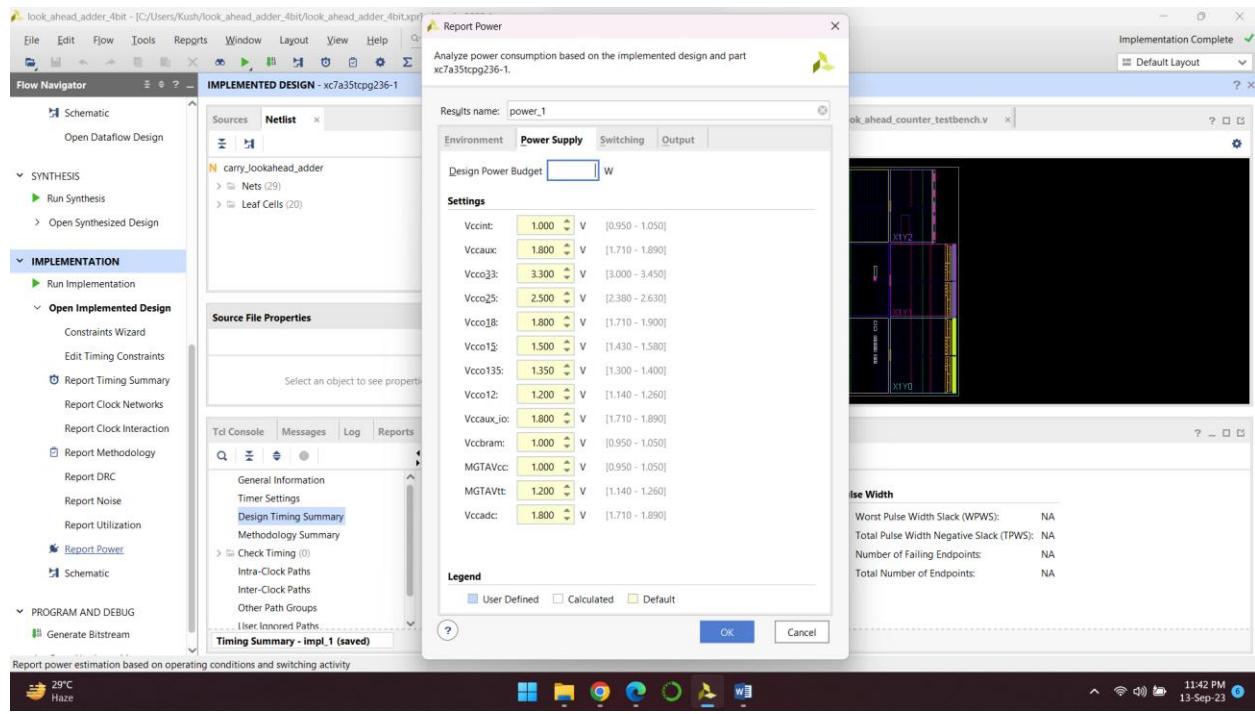
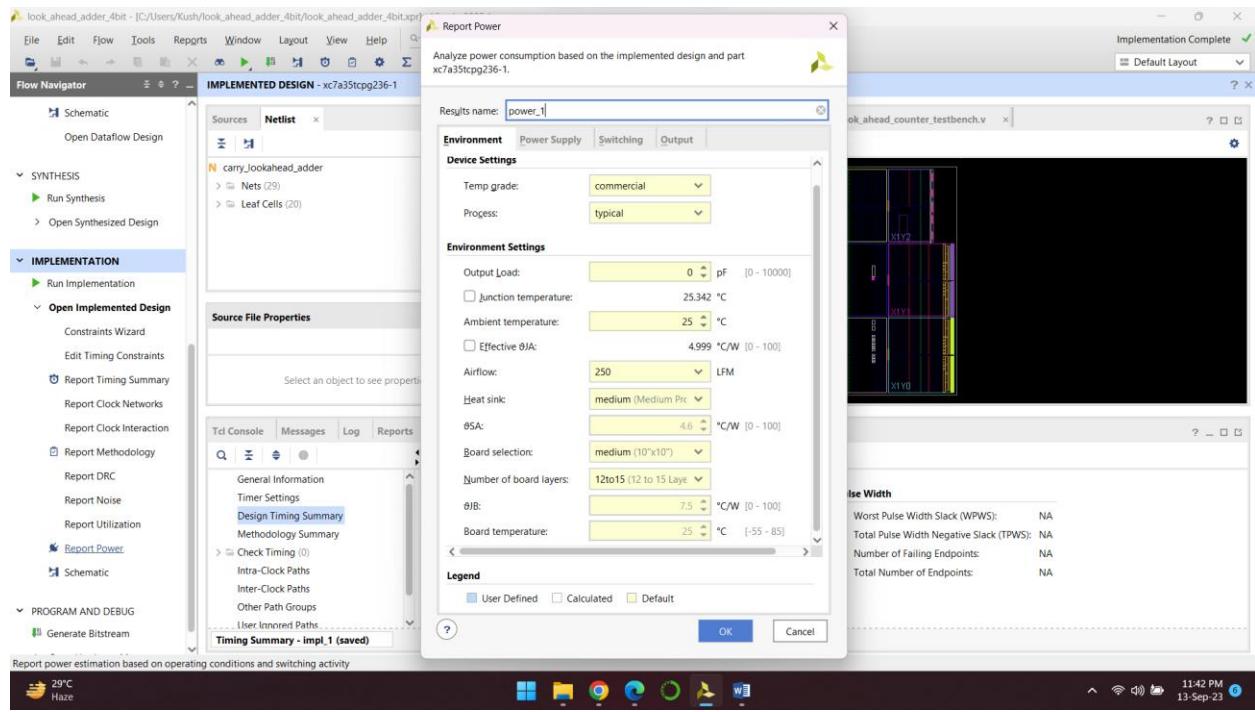
## WAVEFORM:



## RTL SYNTHESIS:



## POWER REPORT:



## 9. N-BIT COMPARATOR:

CODE:

The screenshot shows the Vivado 2023.1 interface with the project "N\_BIT\_CMP" open. The left sidebar displays the Flow Navigator with various simulation, synthesis, and implementation options. The main window shows the behavioral simulation of the "N\_BIT\_CMP\_TB.v" testbench. The code editor displays the Verilog source code for the "n\_bit\_comparator" module, which takes two n-bit inputs (a and b) and outputs reg greater, equal, and lesser. The code includes a parameter "n=3" and logic for all three cases: a>b, a=b, and a<b. The simulation results are shown in the waveform viewer below, with a sim time of 300 ns.

```
21 : 
22 : 
23 : module n_bit_comparator(
24 :   input [n-1:0] a,
25 :   input [n-1:0] b,
26 :   output reg greater,
27 :   output reg equal,
28 :   output reg lesser
29 : );
30 :   parameter n=3 ;
31 :   always @ (a,b)
32 :   begin
33 :     if(a>b)
34 :       begin
35 :         greater = 1;
36 :         equal = 0;
37 :         lesser = 0;
38 :       end
39 :     else if(a==b)
40 :       begin
41 :         greater = 0;
42 :         equal = 1;
43 :         lesser = 0;
44 :       end
45 :     else if(a<b)
46 :       begin
47 :         greater = 0;
48 :         equal = 0;
49 :         lesser = 1;
50 :       end
51 :     end
52 :   endmodule
```

This screenshot is identical to the one above, showing the Vivado 2023.1 interface with the project "N\_BIT\_CMP" open. The Flow Navigator and simulation results are the same, displaying the behavioral simulation of the "N\_BIT\_CMP\_TB.v" testbench. The code editor shows the Verilog source code for the "n\_bit\_comparator" module, which takes two n-bit inputs (a and b) and outputs reg greater, equal, and lesser. The code includes a parameter "n=3" and logic for all three cases: a>b, a=b, and a<b. The simulation results are shown in the waveform viewer below, with a sim time of 300 ns.

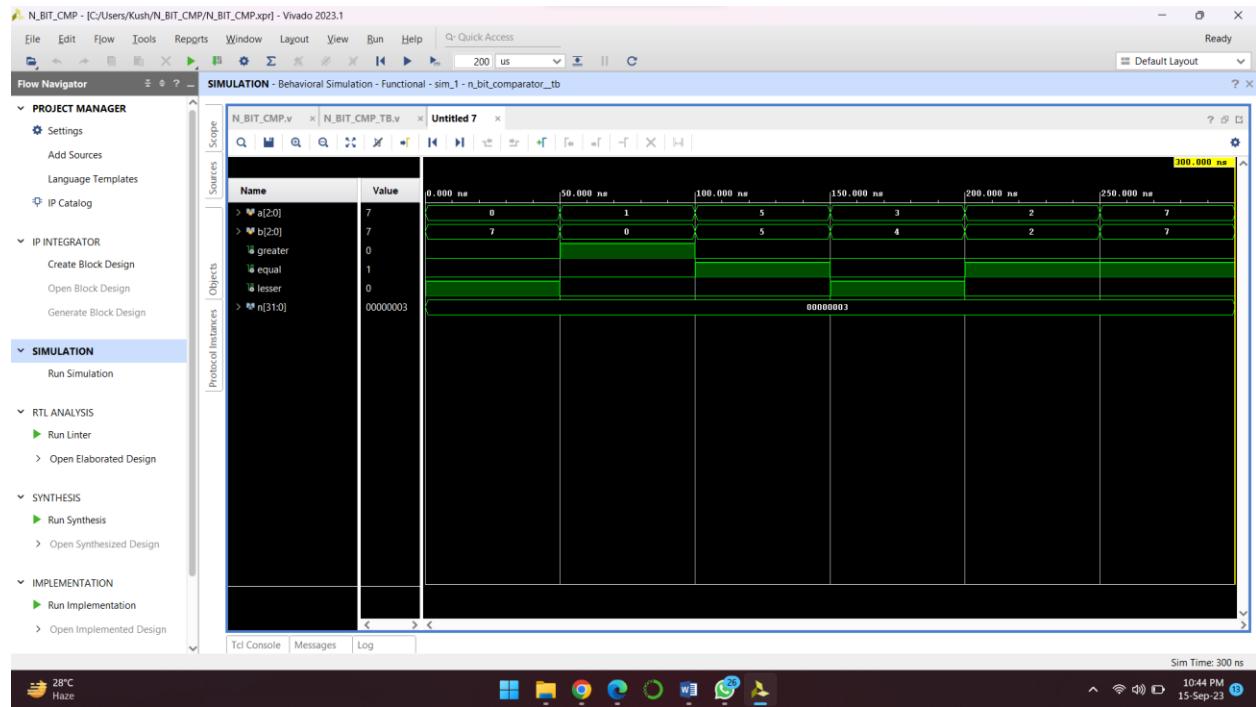
```
25 : 
26 : 
27 : module n_bit_comparator(
28 :   input [n-1:0] a,
29 :   output reg greater,
30 :   output reg equal,
31 :   output reg lesser
32 : );
33 :   parameter n=3 ;
34 :   always @ (a,b)
35 :   begin
36 :     if(a>b)
37 :       begin
38 :         greater = 1;
39 :         equal = 0;
40 :         lesser = 0;
41 :       end
42 :     else if(a==b)
43 :       begin
44 :         greater = 0;
45 :         equal = 1;
46 :         lesser = 0;
47 :       end
48 :     else if(a<b)
49 :       begin
50 :         greater = 0;
51 :         equal = 0;
52 :         lesser = 1;
53 :       end
54 :     end
55 :   endmodule
```

## TESTBENCH:

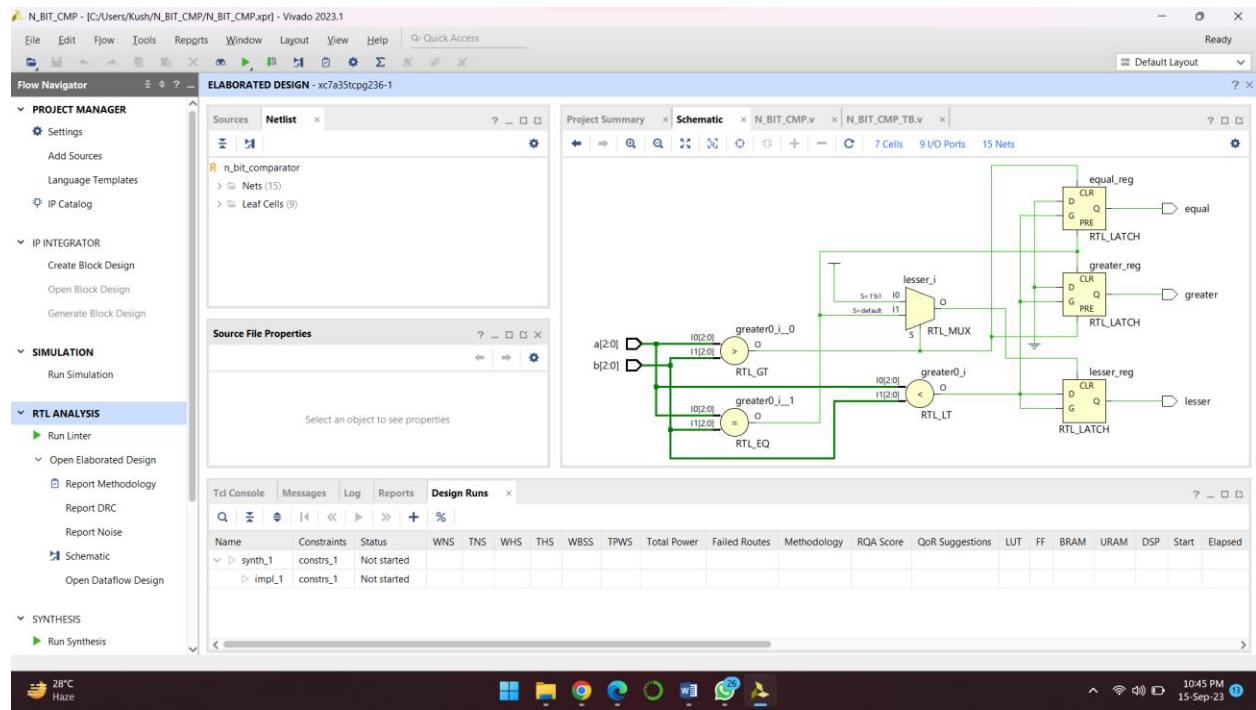
The screenshot shows the Vivado 2023.1 interface with the "SIMULATION - Behavioral Simulation - Functional - sim\_1 - n\_bit\_comparator\_tb" window active. The code editor displays the testbench source code (N\_BIT\_CMP\_TB.v) which includes a testbench for an n-bit comparator. The code defines a module n\_bit\_comparator\_tb with parameters n=3, inputs a and b, and outputs greater, equal, and lesser. It contains an initial block with various test cases and ends with a \$finish command. The simulation console at the bottom shows the current time as 10:44 PM on 15-Sep-23.

```
18 : // Additional Comments:
19 :
20 : //////////////////////////////////////////////////////////////////
21 :
22 : module n_bit_comparator_tb(
23 : );
24 : parameter n=3;
25 : reg [(n-1):0]a;
26 : reg [(n-1):0]b;
27 : wire greater,equal,lesser;
28 :
29 : n_bit_comparator uut(a,b,greater,equal,lesser);
30 :
31 : initial begin
32 : end
33 : initial begin
34 : $00 a=3'b000; b=3'b111;
35 : $00 a=3'b001; b=3'b000;
36 : $00 a=3'b101; b=3'b011;
37 : $00 a=3'b111; b=3'b000;
38 : $00 a=3'b101; b=3'b101;
39 : $00 a=3'b111; b=3'b111;
40 : $00 $finish;
41 : end
42 :
43 : initial begin
44 : $dumpfile("dump.vcd");
45 : $dumpvars(0);
46 : end
47 : endmodule
48 :
49 : a=
```

## WAVEFORM:



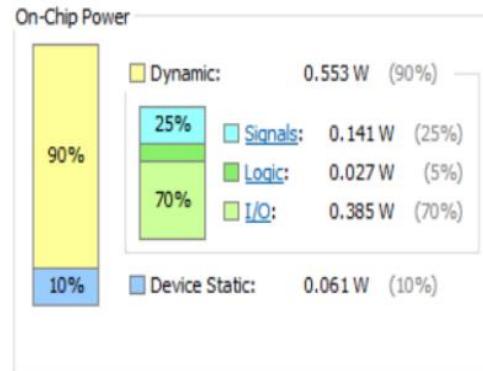
## RTL SCHEMATIC:



## POWER REPORT:

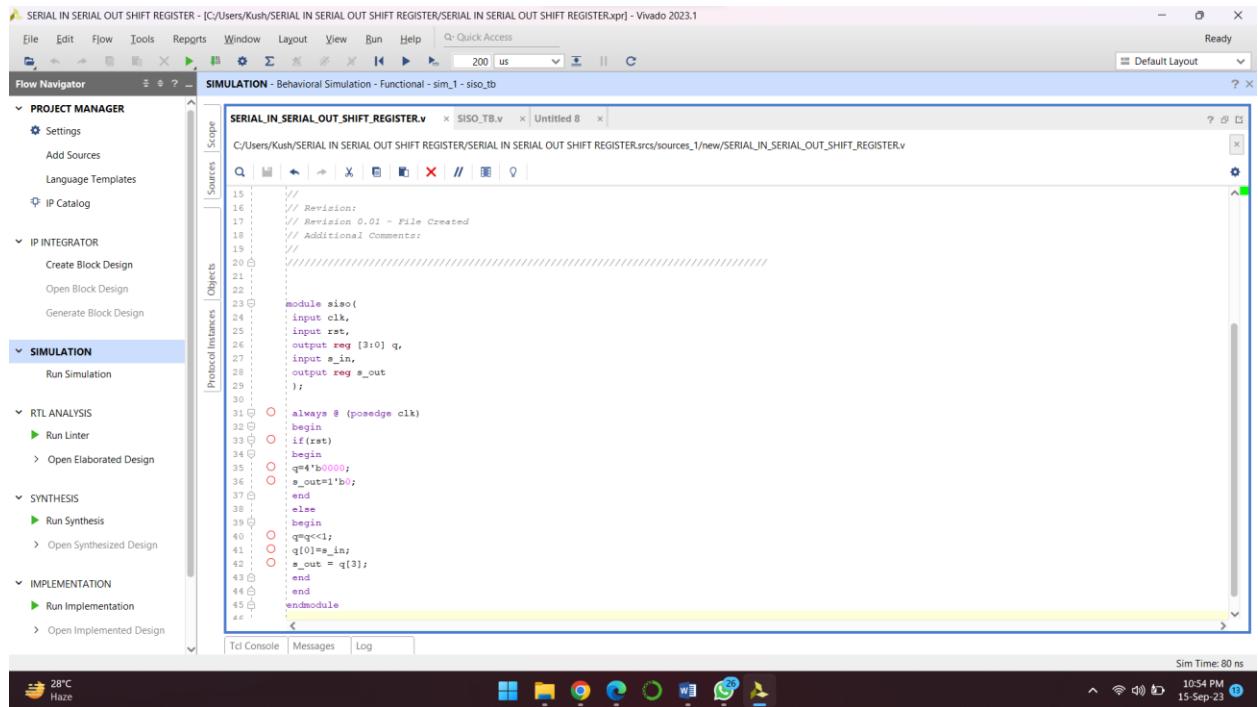
Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

**Total On-Chip Power:** 0.614 W  
**Junction Temperature:** 27.9 °C  
 Thermal Margin: 72.1 °C (15.0 W)  
 Effective  $\Delta T_{JA}$ : 4.8 °C/W  
 Power supplied to off-chip devices: 0 W  
 Confidence level: Low



## 10. SERIAL IN SERIAL OUT SHIFT REGISTER:

### CODE:



```
// Revision: 0.01 - File Created
// Additional Comments:
//
`ifndef __SISO_SV
`define __SISO_SV

module siso(
    input clk,
    input rst,
    output reg [3:0] q,
    input s_in,
    output reg s_out
);

    always @ (posedge clk)
        begin
            if(rst)
                begin
                    q<=4'b0000;
                    s_out<=1'b0;
                end
            else
                begin
                    if(s_in<=1)
                        s_out<=q[0];
                    else
                        s_out = q[3];
                end
        end
endmodule
`endif
```

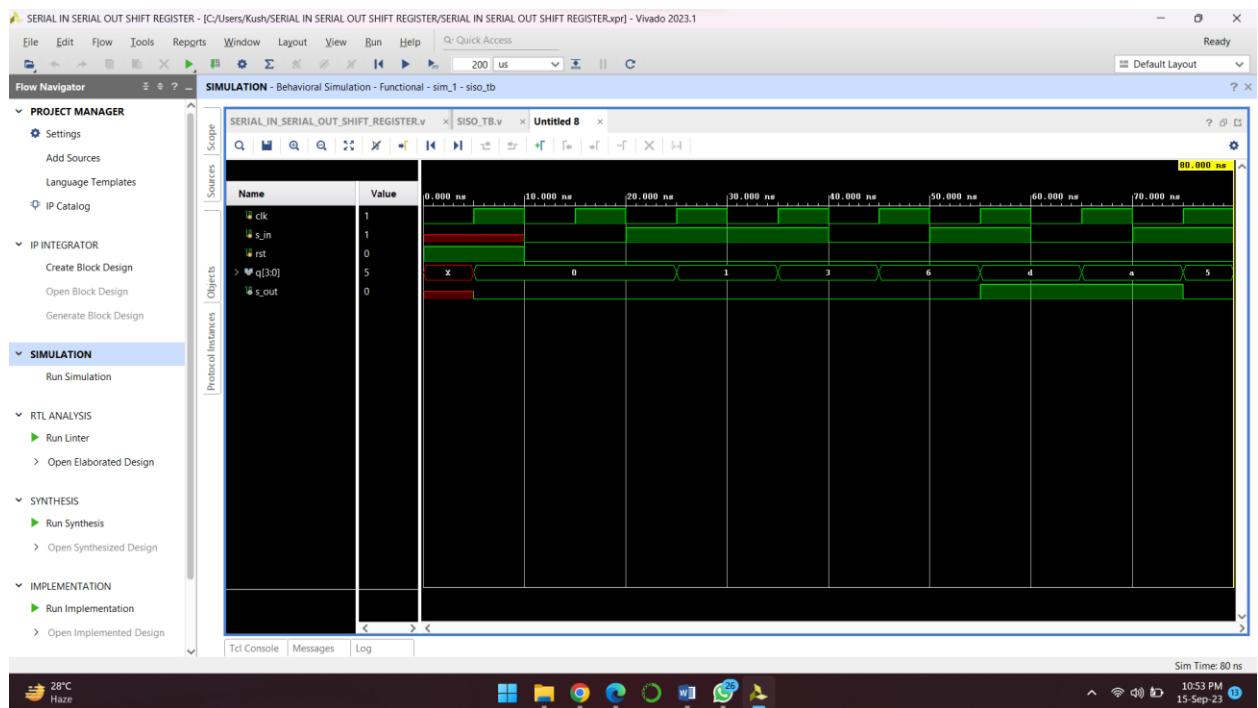
## TESTBENCH:

```

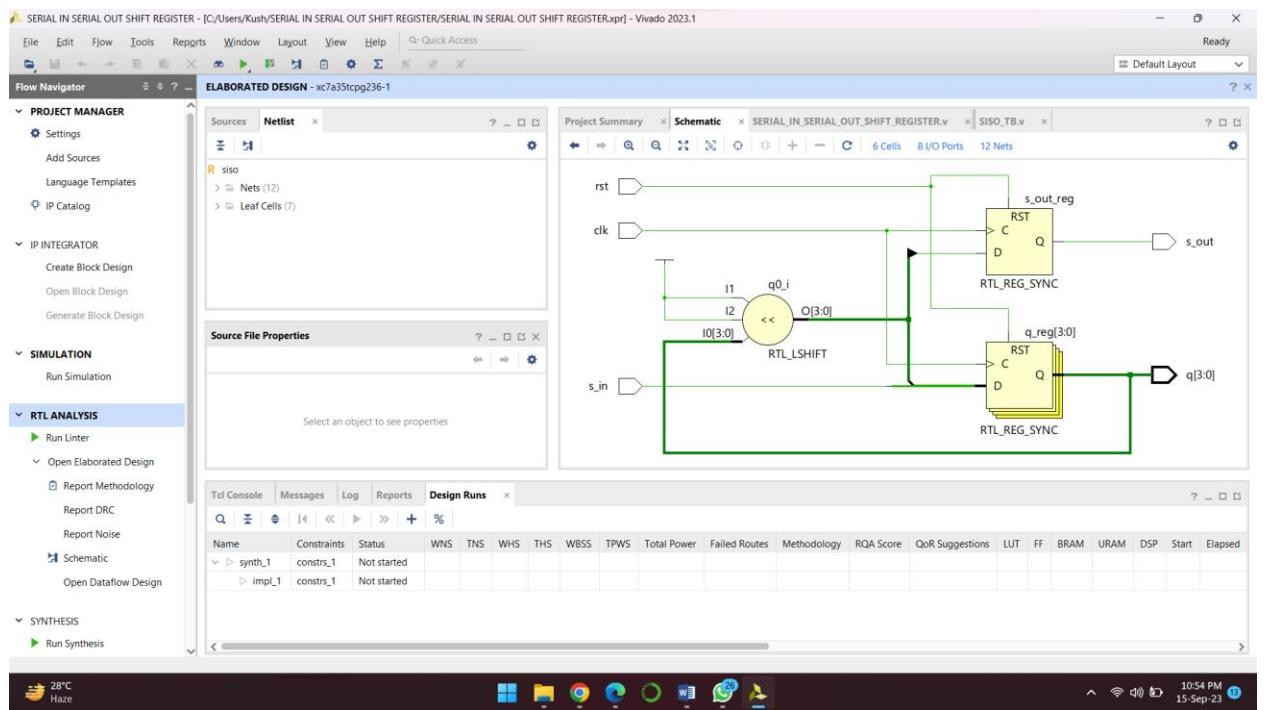
module siso_tb();
    reg clk,s_in;
    reg rst;
    wire [3:0]q;
    wire s_out;
    siso uut(clk,rst,q,s_out);
    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end
    initial
    begin
        #00 rst = 1'b1;
        #10 rst = 1'b0;
        #500 $finish;
    end
    initial begin
        #10 s_in = 1'b0;
        #10 s_in = 1'b1;
        #10 s_in = 1'b1;
        #10 s_in = 1'b0;
        #10 s_in = 1'b1;
        #10 s_in = 1'b0;
        #10 s_in = 1'b1;
        #10 s_in = 1'b0;
        #10 s_in = 1'b1;
        #10 $finish;
    end
end

```

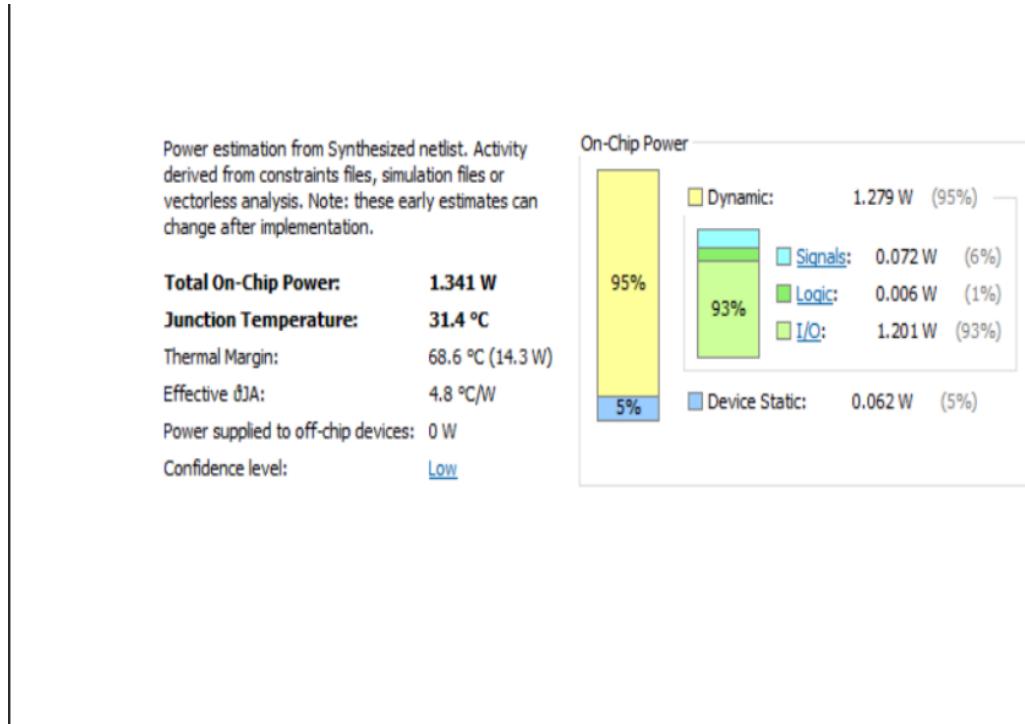
## WAVEFORM:



## RTL SCHEMATIC:



## POWER REPORT:



## 11. SERIAL IN PARALLEL OUT SHIFT REGISTER:

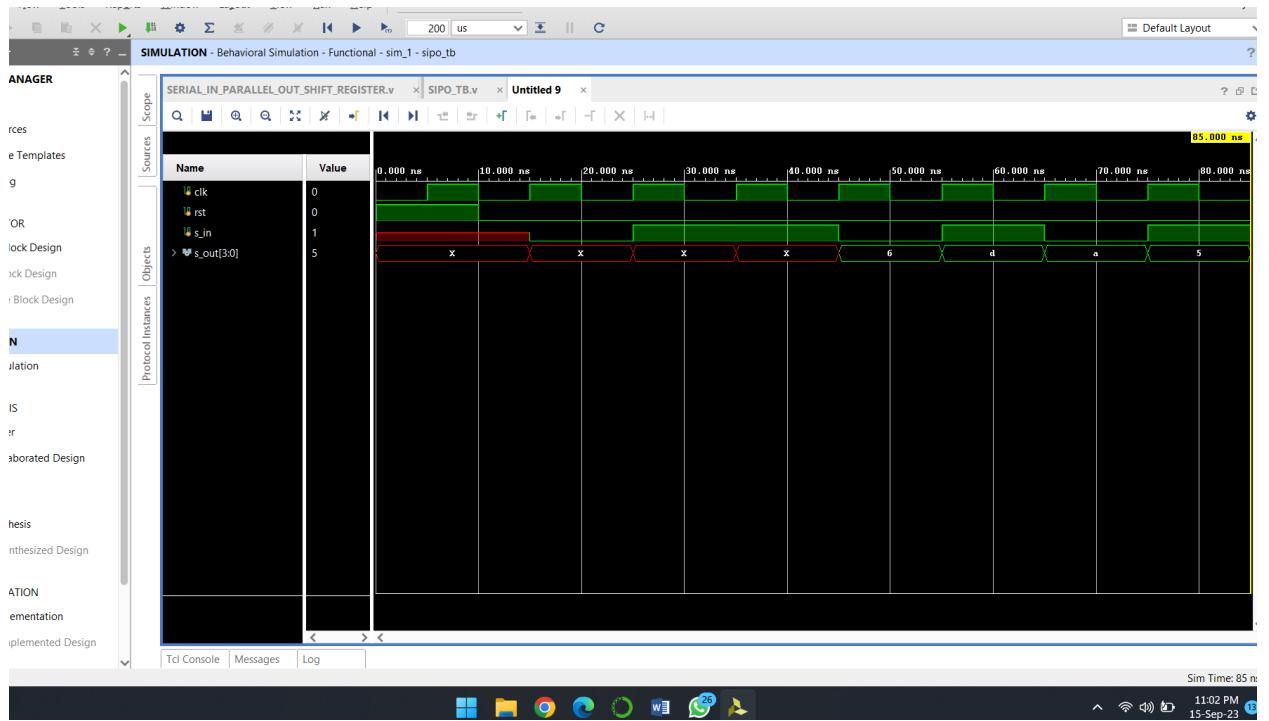
CODE:

```
SERIAL_IN_PARALLEL_OUT_SHIFT_REGISTER.v // SIMULATION - Behavioral Simulation - Functional - sim_1 - sipo_tb
C:/Users/Kush/SERIAL_IN_PARALLEL_OUT_SHIFT_REGISTER/SERIAL_IN_PARALLEL_OUT_SHIFT_REGISTER.srs/sources_1/new/SERIAL_IN_PARALLEL_OUT_SHIFT_REGISTER.v
module DFF(
    input clk,
    input rst,
    input d,
    output reg q
);
    always @ (posedge clk)
        begin
            if(rst)
                q=0;
            else if(clk)
                q = d ;
        end
endmodule
module sipo(
    input clk,
    input rst,
    input [3:0] s_in,
    output [3:0] s_out
);
    Dff d1(clk,reset,s_in,s_out[0]);
    Dff d2(clk,reset,s_out[0],s_out[1]);
    Dff d3(clk,reset,s_out[1],s_out[2]);
    Dff d4(clk,reset,s_out[2],s_out[3]);
endmodule
```

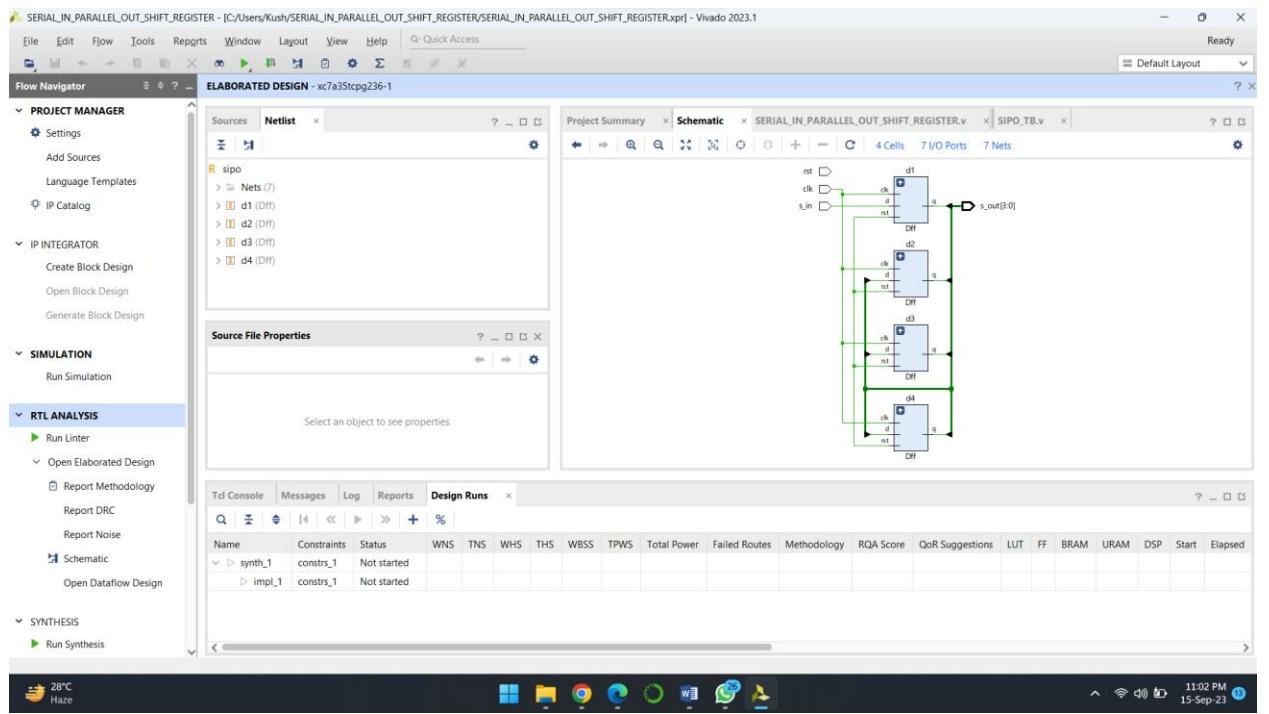
TESTBENCH:

```
SERIAL_IN_PARALLEL_OUT_SHIFT_REGISTER.v // SIMULATION - Behavioral Simulation - Functional - sim_1 - sipo_tb
C:/Users/Kush/SERIAL_IN_PARALLEL_OUT_SHIFT_REGISTER/SERIAL_IN_PARALLEL_OUT_SHIFT_REGISTER.srs/sources_1/new/SIPO_TB.v
reg s_in;
wire [3:0] s_out;
sipto uut(clk,rst,s_in,s_out);
initial begin
    clk = 1'b0;
    forever #5 clk = ~clk;
    end
initial begin
    rst=1'b1;
    #10 rst = 1'b0;
end
always @ (posedge clk,s_in)
begin
    #10 s_in = 1'b0;
    #10 s_in = 1'b1;
    #10 s_in = 1'b1;
    #10 s_in = 1'b0;
    #10 s_in = 1'b1;
    #10 s_in = 1'b0;
    #10 $finish;
end
endmodule
```

## WAVEFORM:



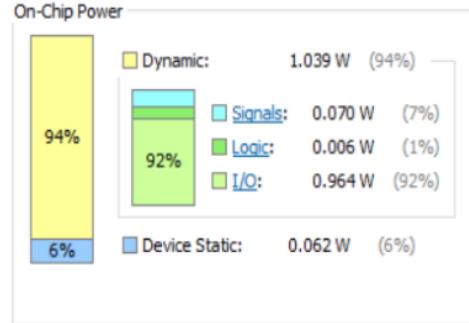
## RTL SCHEMATIC:



## POWER REPORT:

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

**Total On-Chip Power:** 1.101 W  
**Junction Temperature:** 30.3 °C  
 Thermal Margin: 69.7 °C (14.5 W)  
 Effective ΔJA: 4.8 °C/W  
 Power supplied to off-chip devices: 0 W  
 Confidence level: Low



## 12. PARALLEL IN PARALLEL OUT REGISTER:

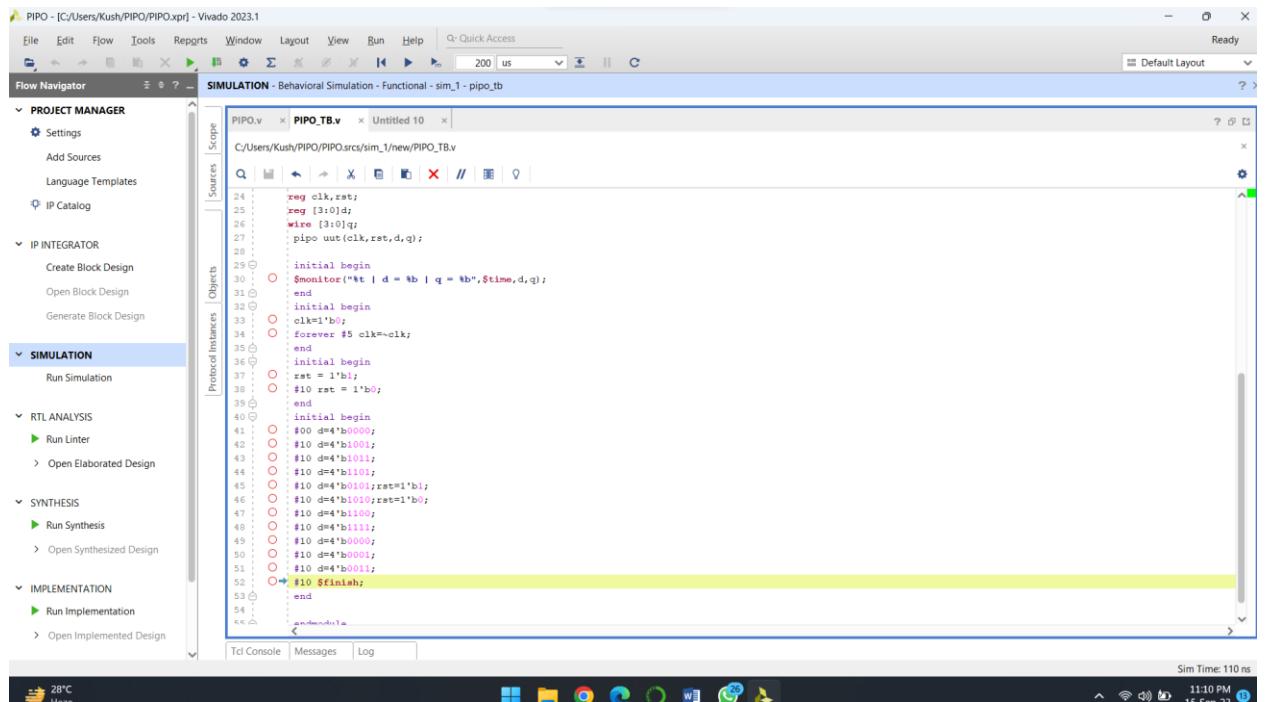
### CODE:

```

// Design Name:
// Module Name: PIP0
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
// Dependencies:
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//////////////////////////////////////////////////////////////////
module pipo(
    input clk,
    input rst,
    input [3:0] d,
    output reg [3:0] q
);
    always @ (posedge clk,d)
        begin
            if(rst)
                q<=0000;
            else
                q=d;
        end
endmodule

```

## TESTBENCH:



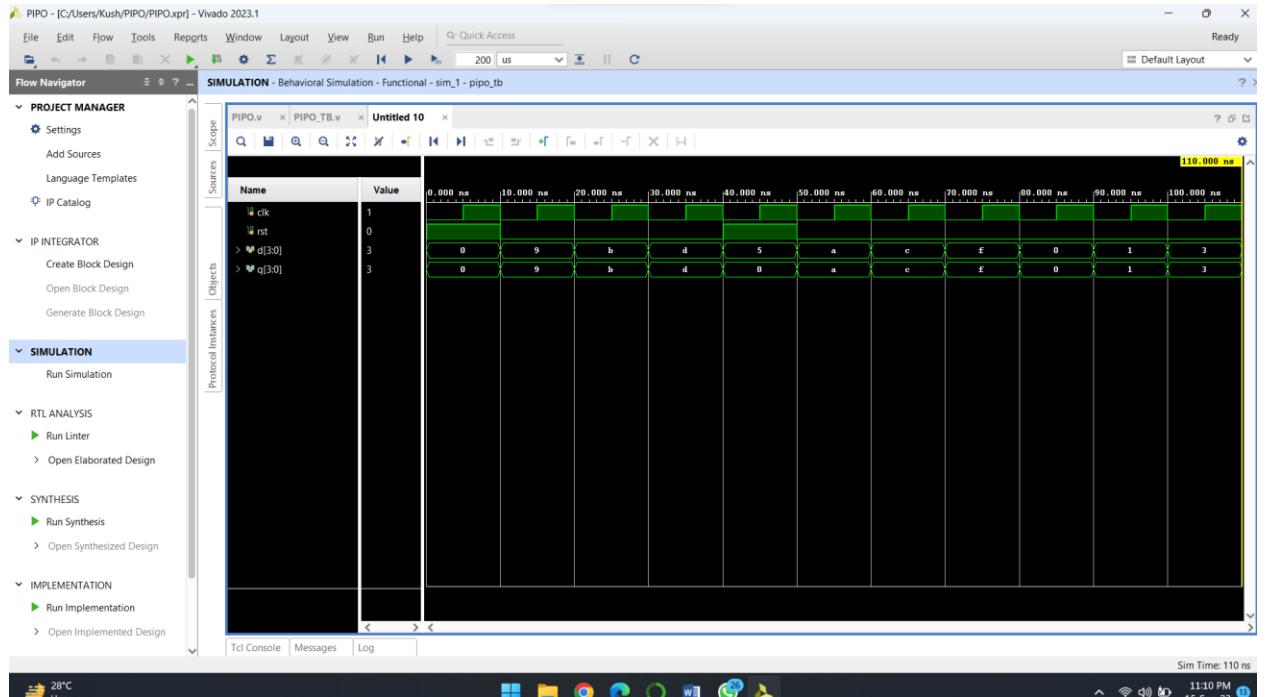
The screenshot shows the Vivado 2023.1 interface with the project 'PIPO' open. The 'SIMULATION' section is selected in the left sidebar. The main window displays the source code for 'PIPO\_TB.v'. The code includes an initial block for monitoring transitions on the 'd' signal, followed by several initial blocks for setting up the 'clk', 'rst', and 'd' signals. The final part of the code contains a 'finish' statement. The status bar at the bottom indicates a simulation time of 110 ns.

```

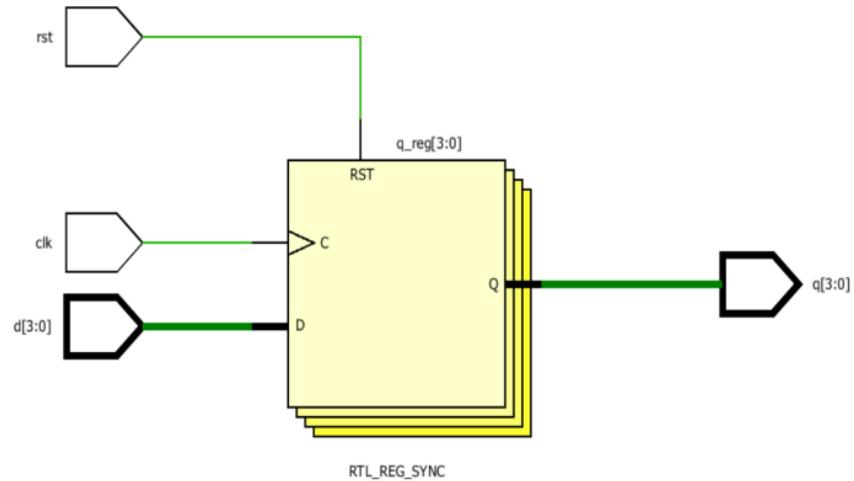
reg clk,rst;
reg [3:0]d;
wire [3:0]q;
pipe#(1) p;
initial begin
$monitor("%t | d = %b | q = %b",$time,d,q);
end
initial begin
clk=1'b0;
forever #5 clk=~clk;
end
initial begin
rst = 1'b1;
#10 rst = 1'b0;
end
initial begin
#10 d=d4'b0000;
#10 d=d4'b1001;
#10 d=d4'b0111;
#10 d=d4'b1101;
#10 d=d4'b1010;rst=1'b1;
#10 d=d4'b1010;rst=1'b0;
#10 d=d4'b1100;
#10 d=d4'b1111;
#10 d=d4'b0000;
#10 d=d4'b0001;
#10 d=d4'b0011;
#10 $finish;
end
endmodule

```

## WAVEFORM:



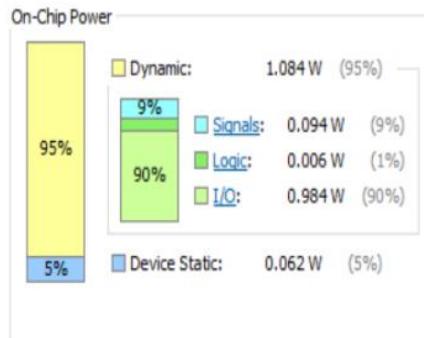
## RTL SCHEMATIC:



## POWER REPORT:

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

**Total On-Chip Power:** 1.146 W  
**Junction Temperature:** 30.5 °C  
Thermal Margin: 69.5 °C (14.5 W)  
Effective  $\Delta T_A$ : 4.8 °C/W  
Power supplied to off-chip devices: 0 W  
Confidence level: Low



### 13. PARALLEL IN SERIAL OUT REGISTER:

CODE:

```

module DFF(
    input clk,
    input rst,
    input d,
    output reg q
);
begin
    if(rst)
        q<='1'b;
    else if(clk)
        q=d;
end
endmodule

module shift_register(
    input d,
    input si,
    input ld,
    output q );
    assign q = (si & ld) | (d & ~ld);
endmodule

module piso(
    input clk,
    input ld,rst,
    input [3:0] d,
    output out
);
    wire w1,w2,w3;
    wire q1,q2,q3;
    Dff d1(clk,rst,d[0],w1);
    shift_register s1(d[1],w1,ld,q1);
    Dff d2(clk,rst,q1,w2);
    shift_register s2(d[2],w2,ld,q2);
    Dff d3(clk,rst,q2,w3);
    shift_register s3(d[3],w3,ld,q3);
    Dff d4(clk,rst,q3,out);
endmodule

```

```

begin
    if(rst)
        q<='1'b;
    else if(clk)
        q=d;
end
endmodule

module shift_register(
    input d,
    input si,
    input ld,
    output q );
    assign q = (si & ld) | (d & ~ld);
endmodule

module piso(
    input clk,
    input ld,rst,
    input [3:0] d,
    output out
);
    wire w1,w2,w3;
    wire q1,q2,q3;
    Dff d1(clk,rst,d[0],w1);
    shift_register s1(d[1],w1,ld,q1);
    Dff d2(clk,rst,q1,w2);
    shift_register s2(d[2],w2,ld,q2);
    Dff d3(clk,rst,q2,w3);
    shift_register s3(d[3],w3,ld,q3);
    Dff d4(clk,rst,q3,out);
endmodule

```

## TESTBENCH:

The screenshot shows the Vivado 2023.1 interface with the following details:

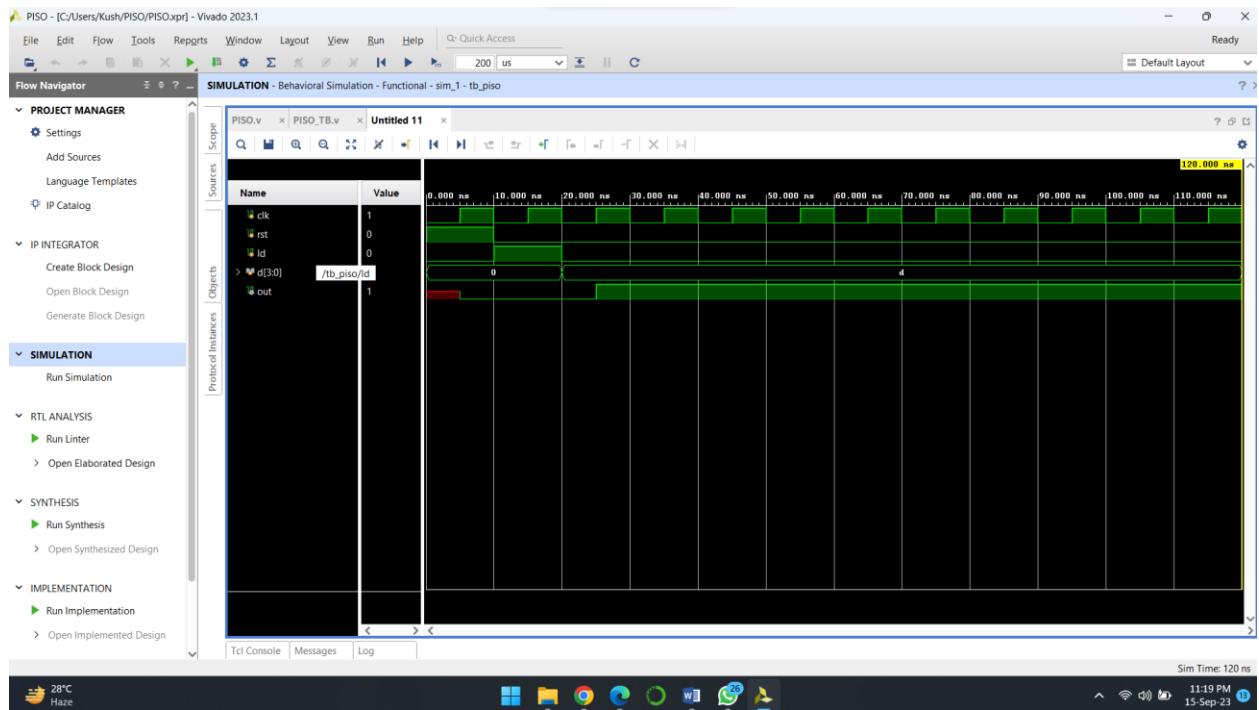
- Title Bar:** PISO - [C:/Users/Kush/PISO/PISO.xpr] - Vivado 2023.1
- Menu Bar:** File, Edit, Flow, Tools, Reports, Window, Layout, View, Run, Help, Q- Quick Access
- Toolbar:** Includes icons for New, Open, Save, Import, Export, Run, Stop, and Help.
- Flow Navigator:** Shows the current flow as "SIMULATION - Behavioral Simulation - Functional - sim\_1 - tb\_piso".
- Project Manager:** Settings, Add Sources, Language Templates, IP Catalog.
- IP Integrator:** Create Block Design, Open Block Design, Generate Block Design.
- Simulation (selected):** Run Simulation.
- RTL Analysis:** Run Linter, Open Elaborated Design.
- Synthesis:** Run Synthesis, Open Synthesized Design.
- Implementation:** Run Implementation, Open Implemented Design.
- Editor:** Displays the Verilog code for the tb\_piso testbench. The code instantiates a piso module, generates a 50 MHz clock, and initializes inputs for clk, rst, id, and d.
- Tcl Console, Messages, Log:** Buttons at the bottom of the editor.
- System Icons:** Weather (28°C), battery, signal strength, and system status.
- Bottom Status Bar:** Sim Time: 120 ns, Date: 15-Sep-23, and a "Default Layout" button.

The screenshot shows the Vivado 2023.1 interface with the following details:

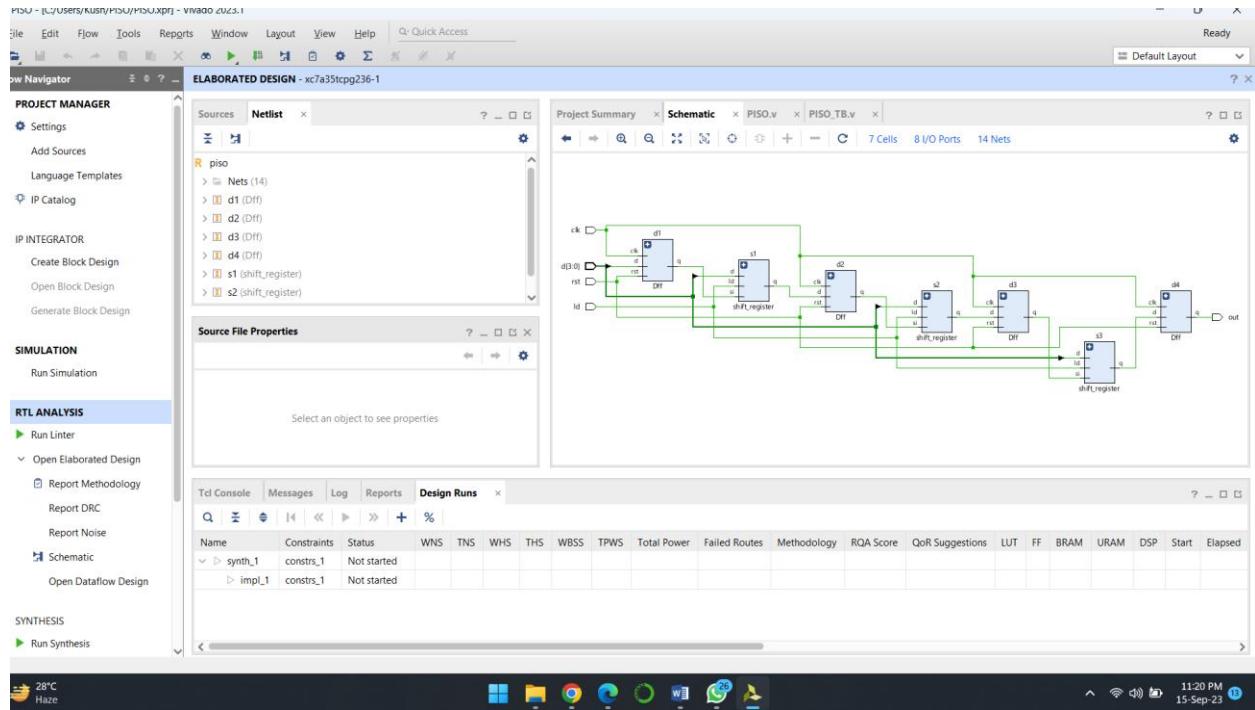
- Title Bar:** PISO - [C:/Users/Kush/PISO/PISO.xpr] - Vivado 2023.1
- Menu Bar:** File, Edit, Flow, Tools, Reports, Window, Layout, View, Run, Help
- Quick Access:** Q- Quick Access, 200 us, Default Layout
- Flow Navigator:** PROJECT MANAGER, IP Catalog, SIMULATION (selected), RTL ANALYSIS, SYNTHESIS, IMPLEMENTATION.
- SIMULATION View:** Behavioral Simulation - Functional - sim\_1\_tb\_piso
- Code Editor:** PISO.v, PISO\_TB.v, Untitled 11. The code is as follows:

```
42:      #5 clk = ~clk; // Toggle the clock every 5 time units (10 ns)
43: end
44:
45: // Initial block
46: initial begin
47:   // Initialize inputs
48:   clk = 0;
49:   rst = 0;
50:   id = 0;
51:   d = 4'b0000;
52:
53:   // Apply reset
54:   rst = 1;
55:   #10 rst = 0;
56:
57:   // Load the shift register
58:   id = 1;
59:   #10 id = 0;
60:
61:   // Input data
62:   d = 4'b1111; // Example input data
63:
64:   // Monitor output and display it using $monitor
65:   $monitor("Time=%t: out=%b", $time, out);
66:
67:   // Simulate for 100 time units (500 ns)
68:   #100 $finish;
69: end
70:
71: endmodule
```
- Tcl Console, Messages, Log:** These tabs are visible at the bottom of the editor.
- System Tray:** 28°C Haze, 11:19 PM, 15-Sep-23, 120 ns, Ready.

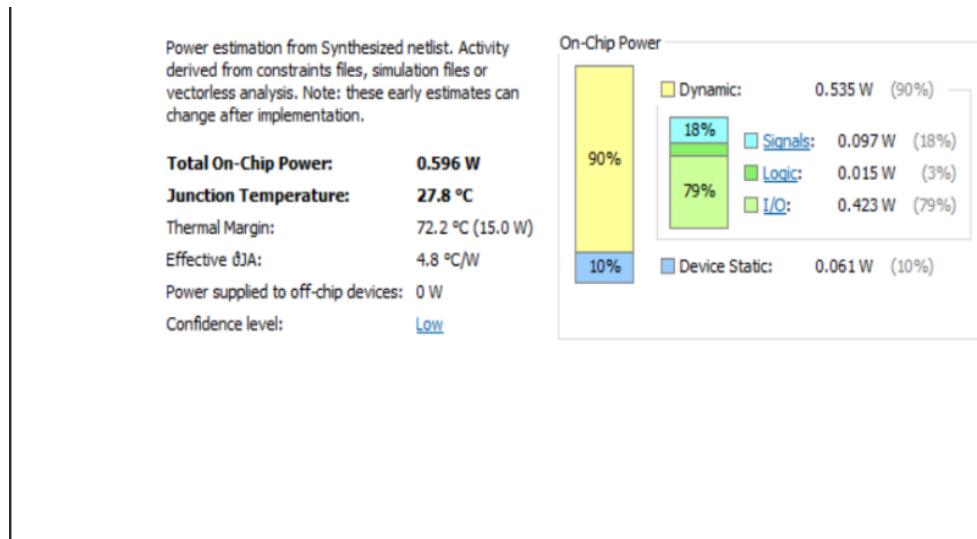
## WAVEFORM:



## RTL SCHEMATIC:



## POWER REPORT:



## 14. BIDIRECTION SHIFT REGISTER:

CODE:

**SYNTHESIZED DESIGN - xc7a35tcpg236-1**

```

module bidirectional_shift_register (
    input wire clk,           // Clock input
    input wire rst,            // Reset input
    input wire shift_left,     // Shift left control input
    input wire shift_right,    // Shift right control input
    input wire parallel_in,    // Parallel load input
    input wire [3:0] data_in,   // Data input (4-bit data)
    output wire [3:0] data_out // Data output (4-bit data)
);

reg [3:0] shift_reg; // 4-bit shift register

always @(posedge clk or posedge rst) begin
    if (rst) begin
        // Reset the shift register to all zeros
        shift_reg <= 4'b0000;
    end else if (parallel_in) begin
        // Parallel load data when parallel_in is asserted
        shift_reg <= data_in;
    end else if (shift_left) begin
        // Shift left when shift_left is asserted
        shift_reg <= (shift_reg[2:0], 1'b0);
    end else if (shift_right) begin
        // Shift right when shift_right is asserted
        shift_reg <= (1'b0, shift_reg[3:1]);
    end
end

assign data_out = shift_reg;
endmodule

```

## TESTBENCH:

**SYNTHESIZED DESIGN - xc7a35tcpg236-1**

```

timescale 1ns/1ns

module tb;

// Inputs
reg clk;
reg rst;
reg shift_left;
reg shift_right;
reg parallel_in;
reg [3:0] data_in;

// Outputs
wire [3:0] data_out;

// Instantiate the bidirectional shift register
bidirectional_shift_register uut (
    .clk(clk),
    .rst(rst),
    .shift_left(shift_left),
    .shift_right(shift_right),
    .parallel_in(parallel_in),
    .data_in(data_in),
    .data_out(data_out)
);

// Clock generation
always begin
    #5 clk = ~clk;
end

```

bidirectional\_shift\_register - [C:/Users/Kush/bidirectional\_shift\_register/bidirectional\_shift\_register.xpr] - Vivado 2023.1

File Edit Flow Tools Reports Window Layout View Help Quick Access

Implementation Complete ✓ Default Layout

**SYNTHESIZED DESIGN - xc7a35tcpg236-1**

Sources | Netlist | Project Summary | Device | bidirectional\_shift\_register.v | bidirectional\_shift\_register\_testbench.v

```

initial begin
    // Initialize signals
    clk = 0;
    rst = 0;
    shift_left = 0;
    shift_right = 0;
    parallel_in = 0;
    data_in = 4'b0000;

    // Open the VCD dump file
    $dumpfile("Shift_Register_dump.vcd");
    $dumpvars(0, testbench);

    // Apply parallel load data
    parallel_in = 1;
    data_in = 4'b101;
    #10;
    parallel_in = 0;

    // Shift data to the left
    shift_left = 1;
    #10;
    shift_left = 0;

    // Shift data to the right
    shift_right = 1;
    #10;
    shift_right = 0;

    // Display the output
    $display("Data Out: %b", data_out);
end

```

Tcl Console Messages Log Reports Design Runs

29°C Haze 12:14 AM 14-Sep-23

bidirectional\_shift\_register - [C:/Users/Kush/bidirectional\_shift\_register/bidirectional\_shift\_register.xpr] - Vivado 2023.1

File Edit Flow Tools Reports Window Layout View Help Quick Access

Implementation Complete ✓ Default Layout

**SYNTHESIZED DESIGN - xc7a35tcpg236-1**

Sources | Netlist | Project Summary | Device | bidirectional\_shift\_register.v | bidirectional\_shift\_register\_testbench.v

```

data_in = 4'b0000;

// Open the VCD dump file
$dumpfile("Shift_Register_dump.vcd");
$dumpvars(0, testbench);

// Apply parallel load data
parallel_in = 1;
data_in = 4'b101;
#10;
parallel_in = 0;

// Shift data to the left
shift_left = 1;
#10;
shift_left = 0;

// Shift data to the right
shift_right = 1;
#10;
shift_right = 0;

// Display the output
$display("Data Out: %b", data_out);

// Finish simulation
$finish;
end

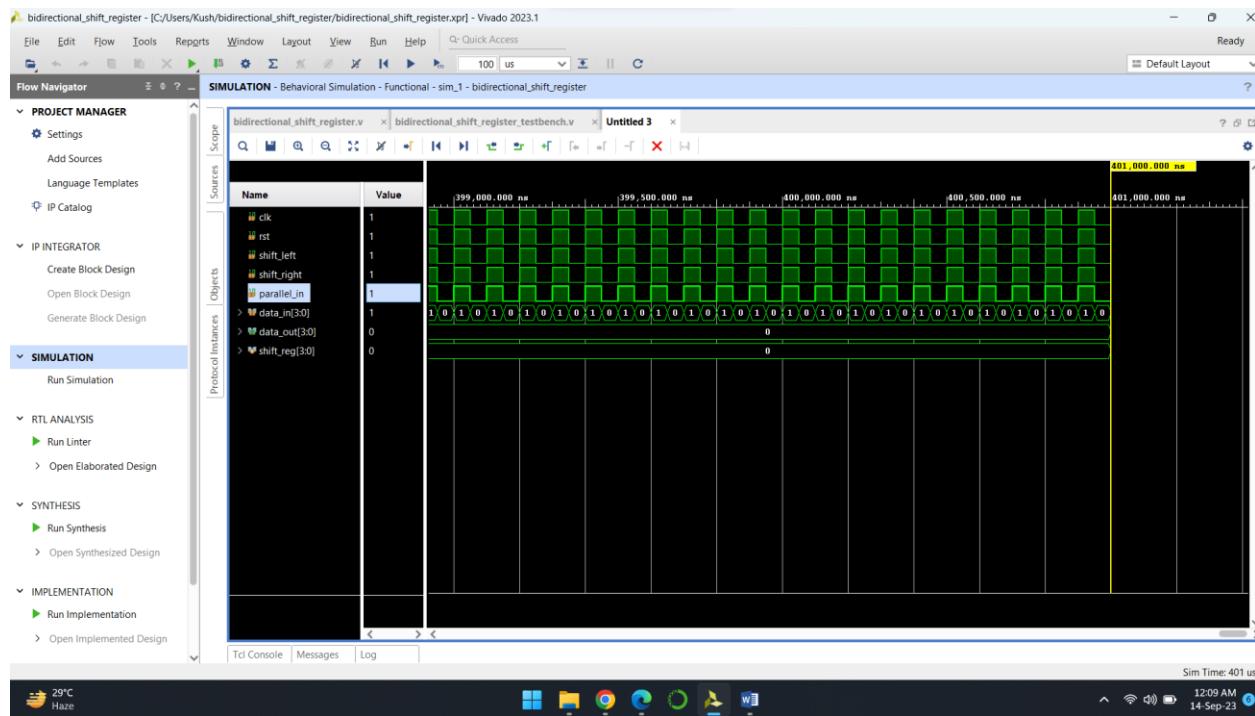
endmodule

```

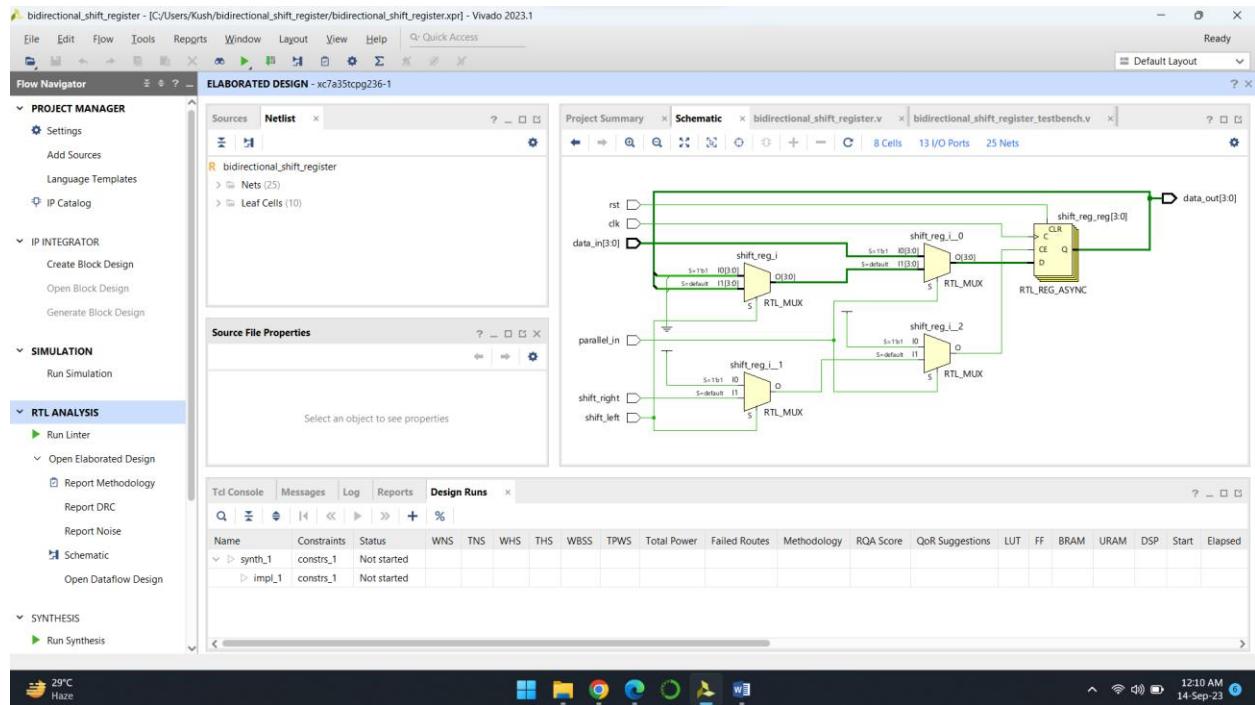
Tcl Console Messages Log Reports Design Runs

29°C Haze 12:15 AM 14-Sep-23

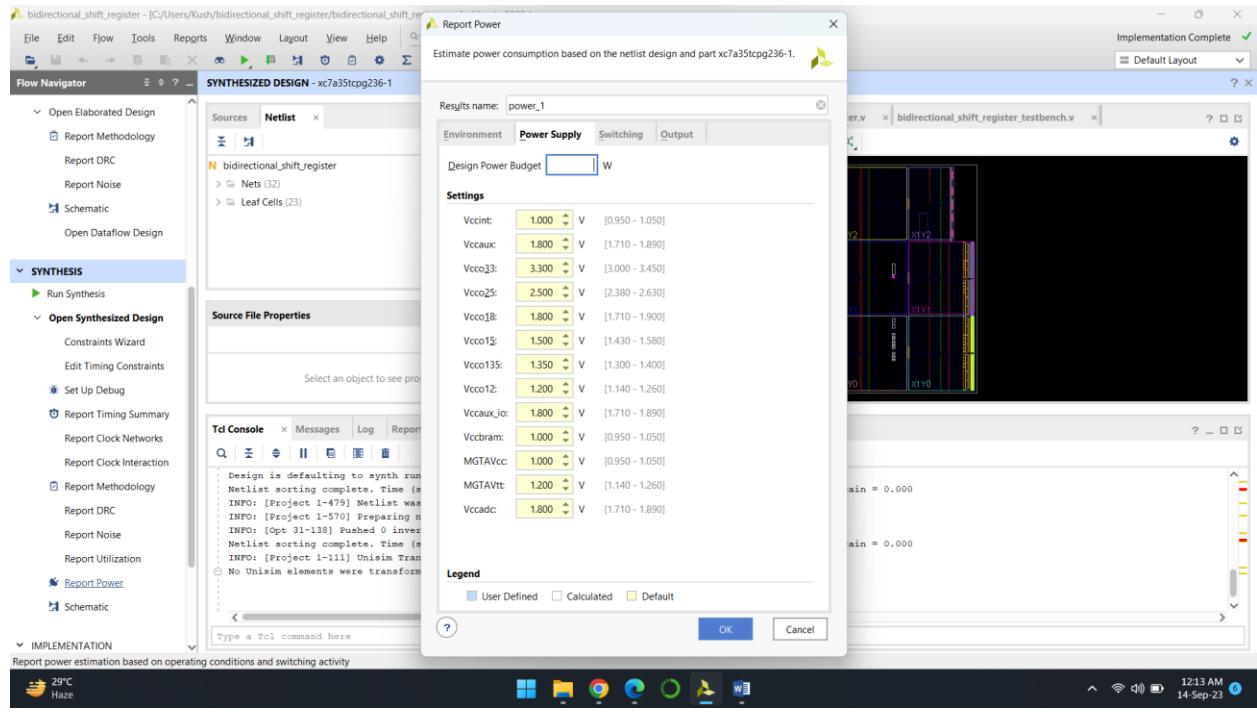
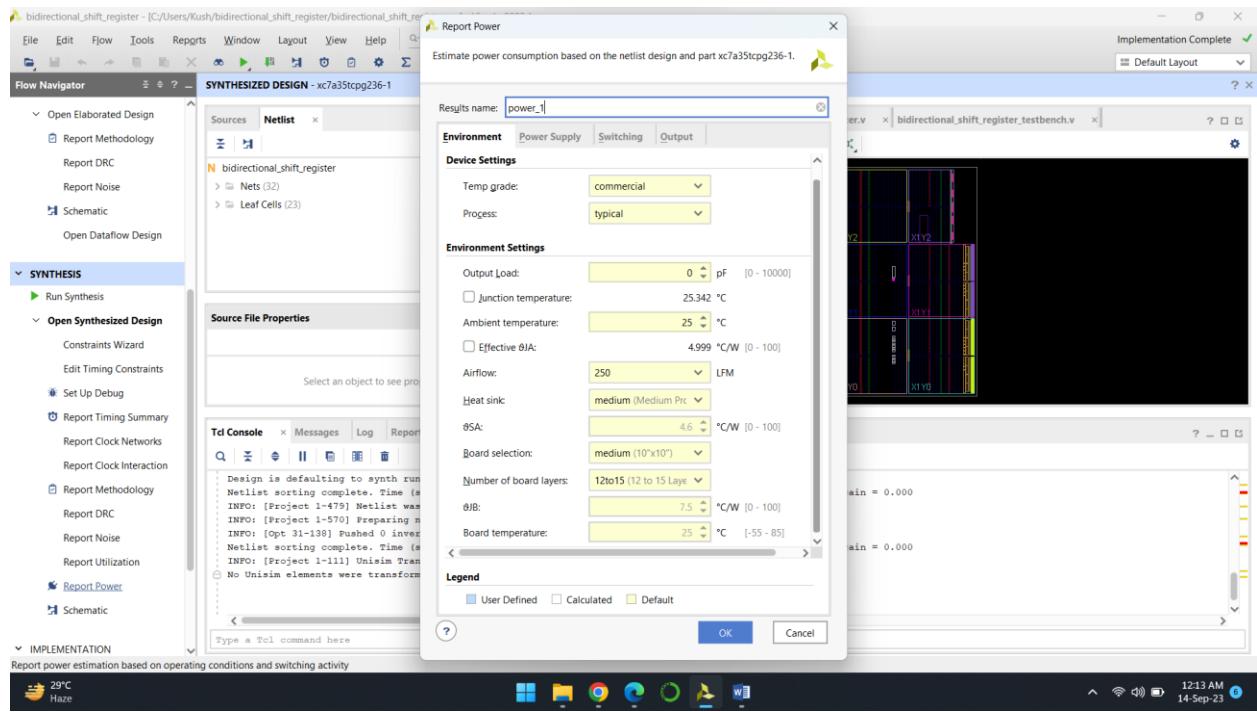
**WAVEFORM:**



## RTL SCHEMATIC:



## POWER REPORT:



## 15. PRBS SEQUENCE GENERATOR:

CODE:

**PRBS SEQUENCE GENERATOR - [C:/Users/Kush/PRBS SEQUENCE GENERATOR/PRBS SEQUENCE GENERATOR.xprj] - Vivado 2023.1**

```

// Tool Versions:
// Description:
// Dependencies:
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
///////////////////////////////////////////////////////////////////
module prbs_generator(
    input clk,
    input reset,
    output reg prbs);
    reg [3:0] lfsr; // 4-bit LFSR register
    always @(posedge clk or posedge reset)
        begin
            if (reset)
                lfsr <= 4'b001; // Initial value (non-zero)
            else
                lfsr[3:0] <= (lfsr[2:0], lfsr[3] ^ lfsr[0]); // LFSR feedback logic
            prbs <= lfsr[3]; // PRBS output is the most significant bit of the LFSR
        end
endmodule

```

Sim Time: 50 ns      8:02 PM 14-Sep-23

## TESTBENCH:

**PRBS SEQUENCE GENERATOR - [C:/Users/Kush/PRBS SEQUENCE GENERATOR/PRBS SEQUENCE GENERATOR.xprj] - Vivado 2023.1**

```

module tb_prbs_generator;
    reg clk;
    reg reset;
    wire prbs;

    // Instantiate the PRBG generator module
    prbs_generator uut (
        .clk(clk),
        .reset(reset),
        .prbs(prbs)
    );

    // Clock generation
    always begin
        #5 clk = ~clk; // Toggle the clock every 5 time units
    end

    // Initializations
    initial begin
        clk = 0;
        reset = 0;
        // Apply reset and wait a few clock cycles
        reset = 1;
        #10 reset = 0;
    end

    // Monitor PRBS output and display it using $monitor
    initial begin
        $monitor("Time=%t PRBS=%b", $time, prbs);
        // Run the simulation for a reasonable number of clock cycles
    end
endmodule

```

Sim Time: 50 ns      8:02 PM 14-Sep-23

PRBS SEQUENCE GENERATOR - [C:/Users/Kush/PRBS SEQUENCE GENERATOR/PRBS SEQUENCE GENERATOR.xpr] - Vivado 2023.1

**Flow Navigator**

- PROJECT MANAGER
- IP INTEGRATOR
- SIMULATION
- RTL ANALYSIS
- SYNTHESIS
- IMPLEMENTATION

**SIMULATION - Behavioral Simulation - Functional - sim\_1 - tb\_prbs\_generator**

**prbs\_sequence\_generator.v**

```

30: prbs_generator uut (
31:   .clk(clk),
32:   .reset(reset),
33:   .prbs(prbs)
34: );
35:
36: // Clock generation
37: always begin
38:   #5 clk = ~clk; // Toggle the clock every 5 time units
39: end
40:
41: // Initializations
42: initial begin
43:   clk = 0;
44:   reset = 0;
45:   // Apply reset and wait a few clock cycles
46:   #10 reset = 1;
47:   #10 reset = 0;
48: end
49:
50: // Monitor PRBS output and display it using $monitor
51: initial begin
52:   $monitor("Time=%t PRBS=%b", $time, prbs);
53:   // Run the simulation for a certain number of clock cycles
54:   #50;
55:   // End simulation
56:   $finish;
57: end
58:
59: endmodule
60:

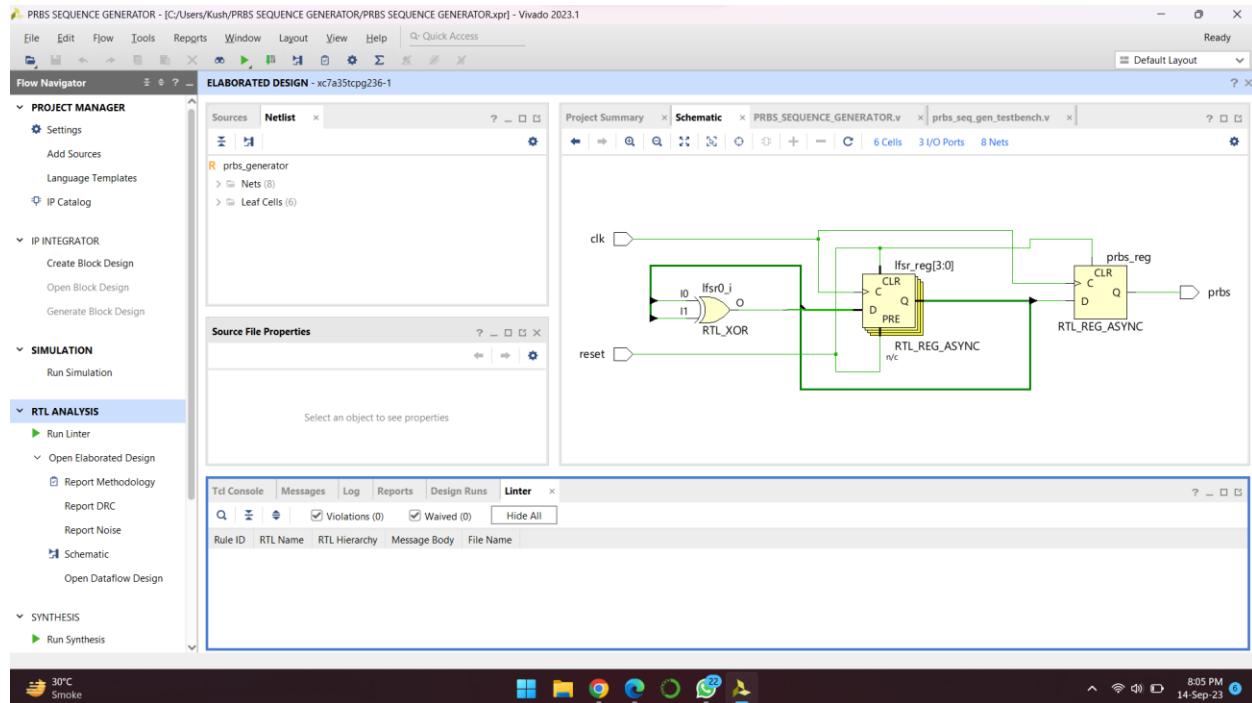
```

Tcl Console | Messages | Log

Sim Time: 50 ns

8:03 PM 14-Sep-23

## RTL SCHEMATIC:



## POWER REPORT:

The screenshot shows the Xilinx Vivado IDE interface with two 'Report Power' dialog boxes open.

**Top Dialog (Environment Settings):**

- Results name: power\_1
- Environment tab selected.
- Device Settings: Temp grade: commercial, Process: typical.
- Properties panel: Select an object to see properties.
- Output Load: 0 pF [0 - 10000].
- Junction temperature: 25.342 °C.
- Ambient temperature: 25 °C.
- Effective θJA: 4.999 °C/W [0 - 100].
- Airflow: 250 lFM.
- Heat sink: medium (Medium Prc).
- θSA: 4.6 °C/W [0 - 100].
- Board selection: medium (10" x 10").
- Number of board layers: 12to15 (12 to 15 Layr).
- θJB: 7.5 °C/W [0 - 100].
- Board temperature: 25 °C [-55 - 85].
- Legend: User Defined, Calculated, Default.

**Bottom Dialog (Power Supply Settings):**

- Results name: power\_1
- Power Supply tab selected.
- Design Power Budget: 0 W.
- Settings section:
  - Vccint: 1.000 V [0.950 - 1.050]
  - Vccaux: 1.800 V [1.710 - 1.890]
  - Vcco33: 3.300 V [3.000 - 3.450]
  - Vcco25: 2.500 V [2.380 - 2.630]
  - Vcco18: 1.800 V [1.710 - 1.900]
  - Vcco15: 1.500 V [1.430 - 1.580]
  - Vcco13: 1.350 V [1.300 - 1.400]
  - Vcco12: 1.200 V [1.140 - 1.260]
  - Vccauxlo: 1.800 V [1.710 - 1.890]
  - Vccbram: 1.000 V [0.950 - 1.050]
  - MGTAVcc: 1.000 V [0.950 - 1.050]
  - MGTAVlt: 1.200 V [1.140 - 1.260]
  - Vccadc: 1.800 V [1.710 - 1.890]
- Legend: User Defined, Calculated, Default.

## 17. 8-BIT ADDER/SUBTRACTOR:

**CODE:**

The screenshot shows the Vivado 2023.1 interface with the following details:

- Title Bar:** adder\_subtractor\_8bit - [C:/Users/Kush/adder\_subtractor\_8bit/adder\_subtractor\_8bit.xpr] - Vivado 2023.1
- Implementation Status:** Implementation Complete
- Flow Navigator:** Report DRC, Report Noise, Schematic, Open Dataflow Design, SYNTHESIS (selected), Run Synthesis, Open Synthesized Design, Constraints Wizard, Edit Timing Constraints, Set Up Debug, Report Timing Summary, Report Clock Networks, Report Clock Interaction, Report Methodology, Report DRC, Report Noise, Report Utilization, Report Power, Schematic, IMPLEMENTATION (selected), Run Implementation, Open Implemented Design.
- Synthesis Project:** SYNTHEZIZED DESIGN - x7e35tcpq236-1
- Source File:** adder\_subtractor\_8bit.v (opened)
- Code Content:** Verilog code for an 8-bit adder-subtractor module. The code includes tool versions, revision information, and a module definition with various input and output wires, assignments for addition and subtraction, and an overflow signal.
- Bottom Navigation:** Tcl Console, Messages, Log, Reports, Design Runs.

## TESTBENCH:

The screenshot shows the Vivado 2023.1 interface with the following details:

- Top Bar:** File, Edit, Flow, Tools, Reports, Window, Layout, View, Help, Quick Access.
- Implementation Status:** Implementation Complete, Default Layout.
- Flow Navigator (Left Sidebar):**
  - SYNTHESIS:** Run Synthesis, Open Synthesized Design (selected), Constraints Wizard, Edit Timing Constraints, Set Up Debug, Report Timing Summary, Report Clock Networks, Report Clock Interaction, Report Methodology (DRC, Noise, Utilization, Power).
  - IMPLEMENTATION:** Run Implementation, Open Implemented Design.
- Central Area:** SYNTHESIZED DESIGN - xc7a35tcpg236-1. Project Summary, Device, adder\_subtractor\_8bit, adder\_subtractor\_8bit\_tb.v.
  - Sources:** Testbench script (adder\_subtractor\_8bit\_tb.v) containing Verilog code for a 8-bit adder/subtractor.
  - Properties:** Properties tab for the testbench.
- Bottom Navigation:** Tcl Console, Messages, Log, Reports, Design Runs.
- System Tray:** 30°C, Smoke, Icons for File Explorer, Task View, Edge, Google Chrome, File History, Taskbar, Network, Battery, Volume, and a yellow icon.
- Bottom Right:** 11:58 PM, 14-Sep-23.

Vivado 2023.1 - Implementation Complete

**SYNTHESIZED DESIGN - xc7a35tcpg236-1**

**Sources**

```

A = $b'00101010;
B = $b'11001100;
subtract = 0;
$10;

// Test case 2: Subtraction
A = $b'11001100;
B = $b'00101010;
subtract = 1;
$10;

// Test case 3: Overflow during addition
A = $b'11111111;
B = $b'00000000;
subtract = 0;
$10;

// Test case 4: Overflow during subtraction
A = $b'00000000;
B = $b'00000000;
subtract = 1;
$10;

$finish;
end

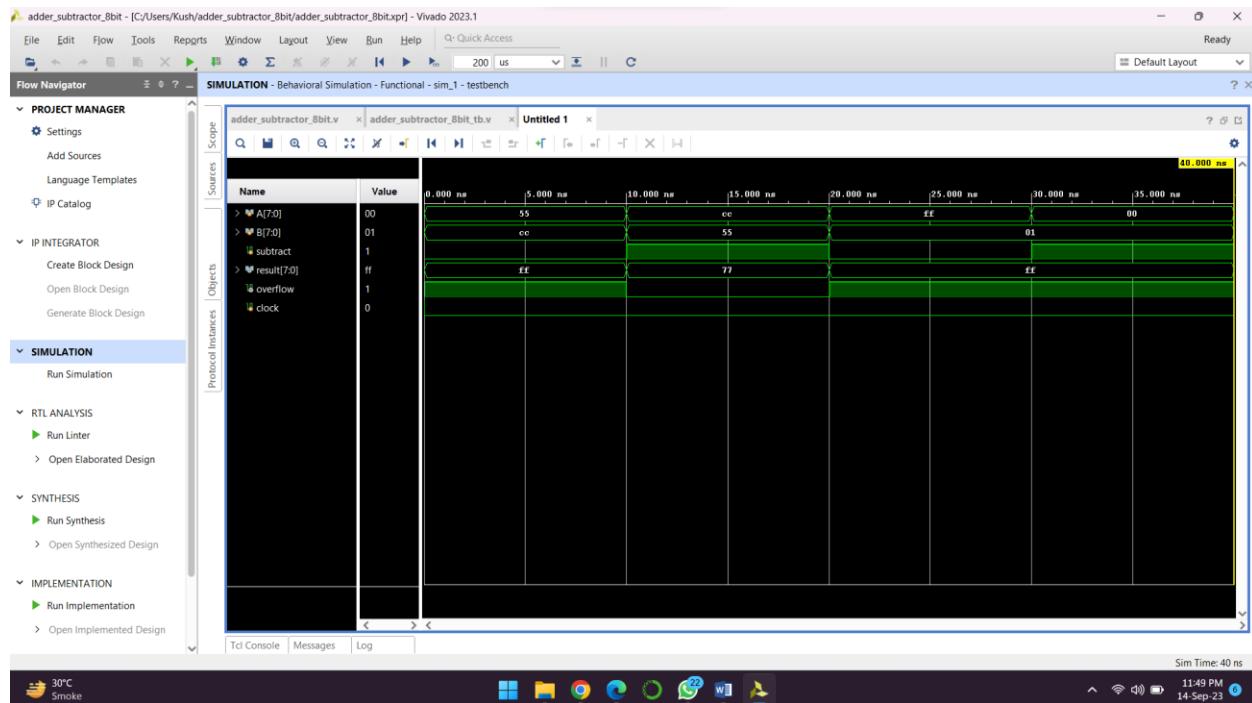
// Clock generation
always #5 clock = ~clock;
endmodule

```

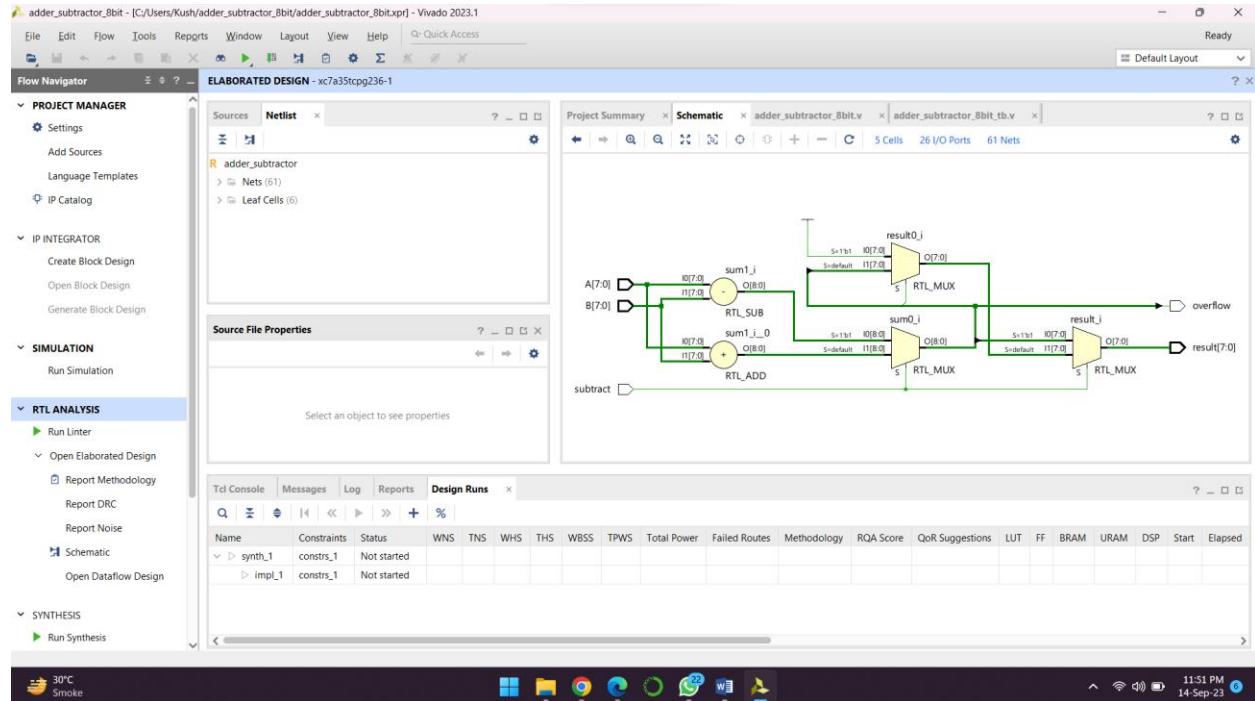
Tcl Console | Messages | Log | Reports | Design Runs

30°C Smoke 11:58 PM 14-Sep-23

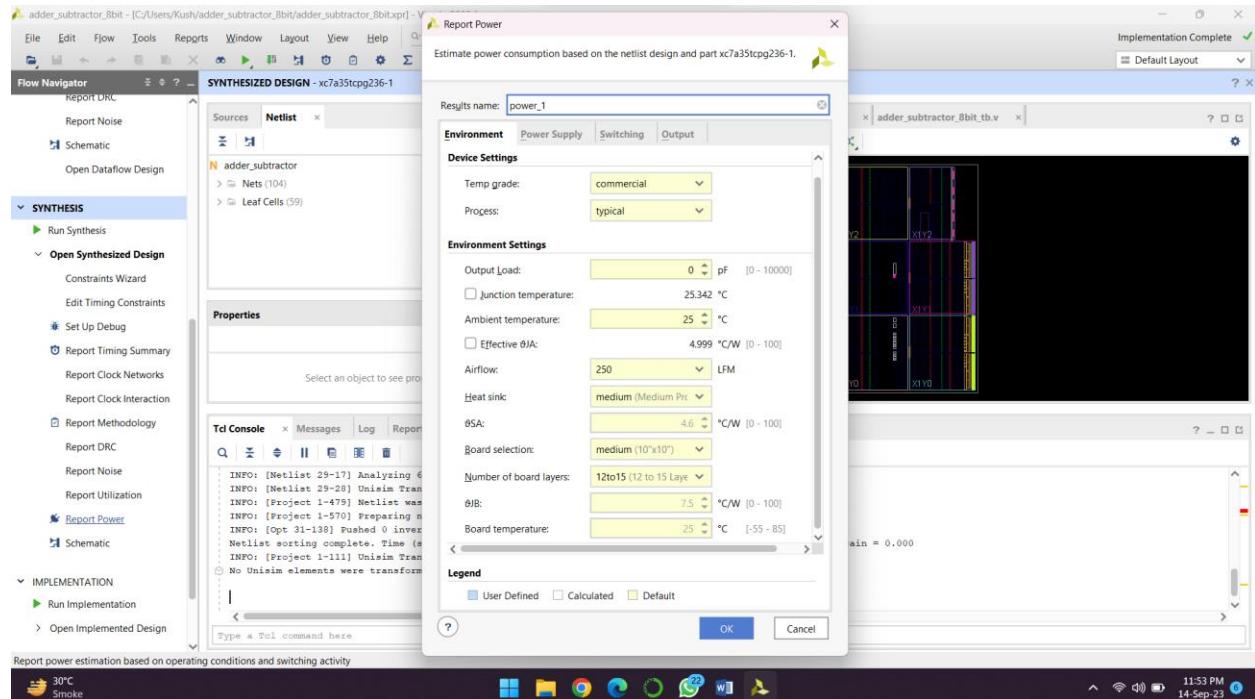
## WAVEFORM:

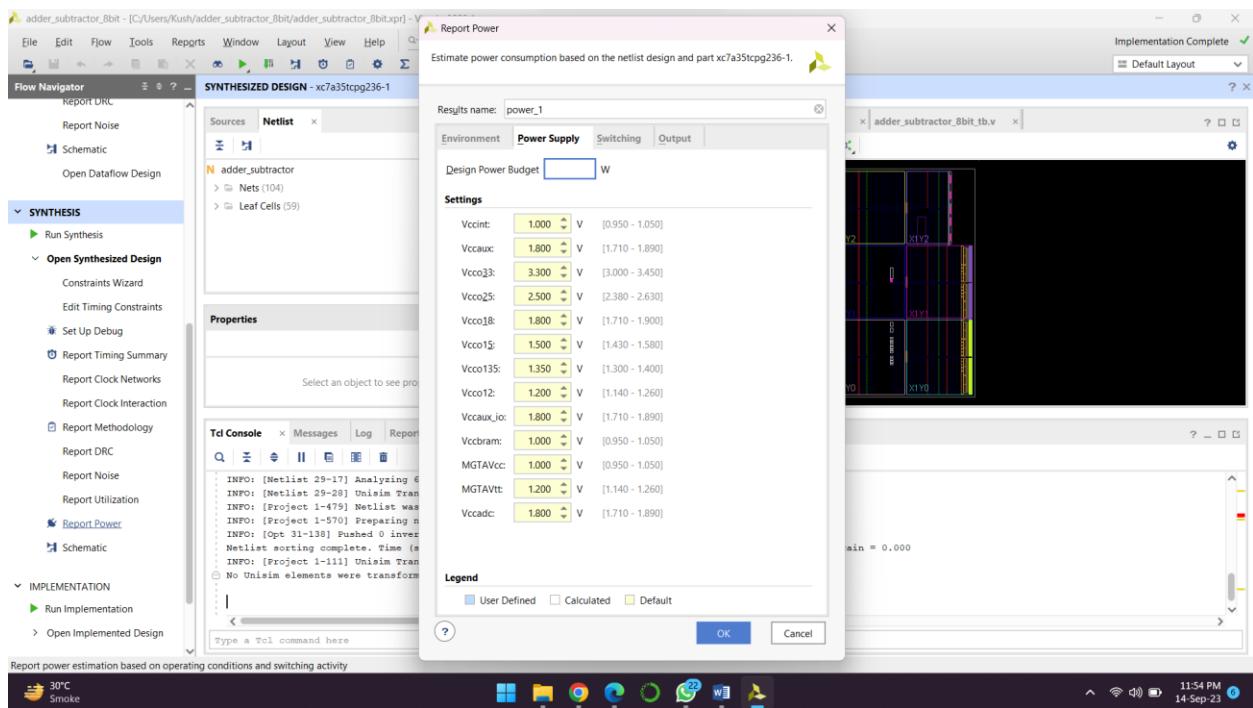


## RTL SCHMETIC:



## POWER REPORT:





## 18. 4-BIT MULTIPLIER:

CODE:

**FOUR\_BIT\_MUX - [C:/Users/Kush/FOUR\_BIT\_MUX/FOUR\_BIT\_MUX.xpr] - Vivado 2023.1**

File Edit Flow Tools Reports Window Layout View Help Q- Quick Access

IMPLEMENTED DESIGN - xc7a35tcpg236-1

Flow Navigator Sources Netlist Project Summary Device FOUR\_BIT\_MUX.v FOUR\_BIT\_MUX\_TB.v

Schematic Report Methodology Report DRC Report Noise Open Dataflow Design

SYNTHESIS Run Synthesis Open Synthesized Design

IMPLEMENTATION Run Implementation Open Implemented Design

- Constraints Wizard
- Edit Timing Constraints
- Report Timing Summary
- Report Clock Networks
- Report Clock Interaction
- Report Methodology
- Report DRC
- Report Noise
- Report Utilization
- Report Power

Tcl Console Messages Log Reports Design Runs Power DRC Timing

```

9 // Project Name;
10 // Target Devices;
11 // Tool Versions;
12 // Description;
13 //
14 // Dependencies;
15 //
16 // Revision;
17 // Revision 0.01 - File Created
18 // Additional Comments;
19 //
20 ///////////////////////////////////////////////////////////////////
21 //
22 module mux_4bit (
23     input wire [3:0] data0, // 4-bit input data 0
24     input wire [3:0] data1, // 4-bit input data 1
25     input wire [3:0] data2, // 4-bit input data 2
26     input wire [3:0] data3, // 4-bit input data 3
27     input wire [1:0] sel, // 2-bit select input
28     output wire [3:0] y   // 4-bit output
29 );
30
31 assign y = (sel == 2'b00) ? data0 :
32           (sel == 2'b01) ? data1 :
33           (sel == 2'b10) ? data2 :
34           data3;
35
36 endmodule
37
38
39 
```

30°C Smoke 12:27 AM 15-Sep-23

## TESTBENCH:

**FOUR\_BIT\_MUX - [C:/Users/Kush/FOUR\_BIT\_MUX/FOUR\_BIT\_MUX.xpr] - Vivado 2023.1**

File Edit Flow Tools Reports Window Layout View Help Q- Quick Access

IMPLEMENTED DESIGN - xc7a35tcpg236-1

Flow Navigator Sources Netlist Project Summary Device FOUR\_BIT\_MUX.v FOUR\_BIT\_MUX\_TB.v

Schematic Report Methodology Report DRC Report Noise Open Dataflow Design

SYNTHESIS Run Synthesis Open Synthesized Design

IMPLEMENTATION Run Implementation Open Implemented Design

- Constraints Wizard
- Edit Timing Constraints
- Report Timing Summary
- Report Clock Networks
- Report Clock Interaction
- Report Methodology
- Report DRC
- Report Noise
- Report Utilization
- Report Power

Tcl Console Messages Log Reports Design Runs Power DRC Timing

```

23 module testbench;
24
25     // Parameters
26     reg [3:0] data0, data1, data2, data3;
27     reg [1:0] sel;
28     wire [3:0] y;
29
30     // Instantiate the 4-bit MUX module
31     mux_4bit uut (
32         .data0(data0),
33         .data1(data1),
34         .data2(data2),
35         .data3(data3),
36         .sel(sel),
37         .y(y)
38     );
39
40     // Clock generation
41     reg clock = 0;
42     always #5 clock = ~clock;
43
44     // Test stimulus
45     initial begin
46         $monitor("Time=%t: data0=%b, data1=%b, data2=%b, data3=%b, sel=%b, y=%b", $time, data0, data1, data2, data3, sel, y);
47
48         // Test case 1: Select data0
49         data0 = 4'b1101;
50         data1 = 4'b1010;
51         data2 = 4'b0110;
52         data3 = 4'b1001;
53         sel = 2'b00;
54         #10;
55     end
56 
```

30°C Smoke 12:27 AM 15-Sep-23

**FOUR\_BIT\_MUX - [C:/Users/Kush/FOUR\_BIT\_MUX/FOUR\_BIT\_MUX.xpr] - Vivado 2023.1**

File Edit Flow Tools Reports Window Layout View Help Quick Access

Implementation Complete ✓ Default Layout

**IMPLEMENTED DESIGN - xc7a35tcpg236-1**

Project Summary Device FOUR\_BIT\_MUX.v FOUR\_BIT\_MUX\_TB.v

C:/Users/Kush/FOUR\_BIT\_MUX/FOUR\_BIT\_MUX/srcs/sim\_1/new/FOUR\_BIT\_MUX\_TB.v

Sources | Netlist

```

initial begin
    $monitor("Time=%t: data0=%h, data1=%h, data2=%h, data3=%h, sel=%b, y=%h", $time, data0, data1, data2, data3, sel, y);

    // Test case 1: Select data0
    data0 = 4'b1101;
    data1 = 4'b1010;
    data2 = 4'b0110;
    data3 = 4'b1001;
    sel = 2'b00;
    #10;

    // Test case 2: Select data1
    sel = 2'b01;
    #10;

    // Test case 3: Select data2
    sel = 2'b10;
    #10;

    // Test case 4: Select data3
    sel = 2'b11;
    #10;

    $finish;
end

// Clock generation
always #5 clock = ~clock;

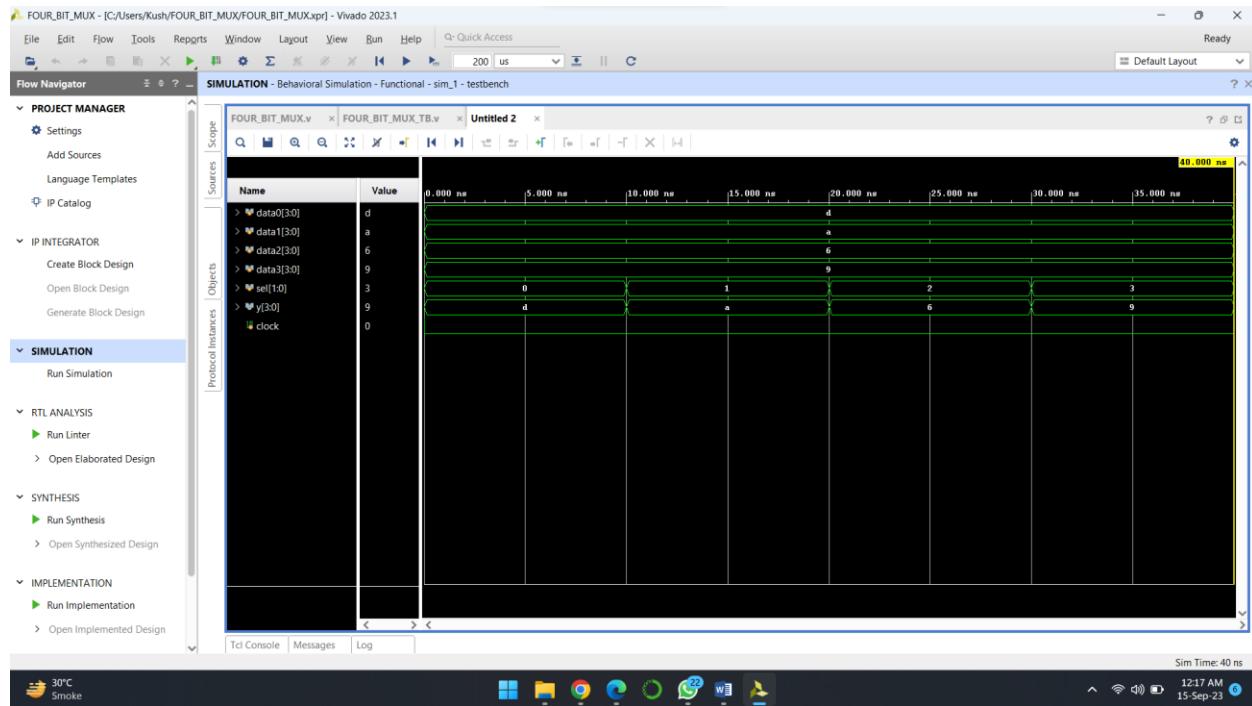
endmodule

```

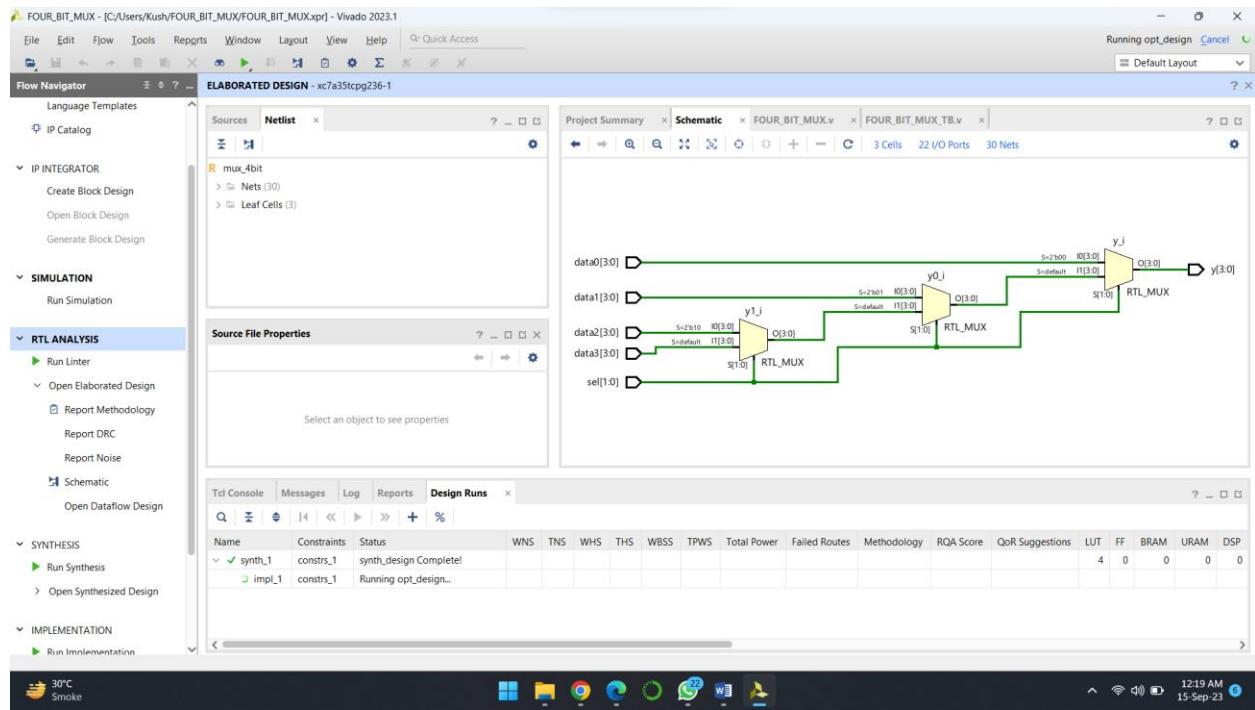
Tcl Console Messages Log Reports Design Runs Power DRC Timing

30°C Smoke 12:27 AM 15-Sep-23

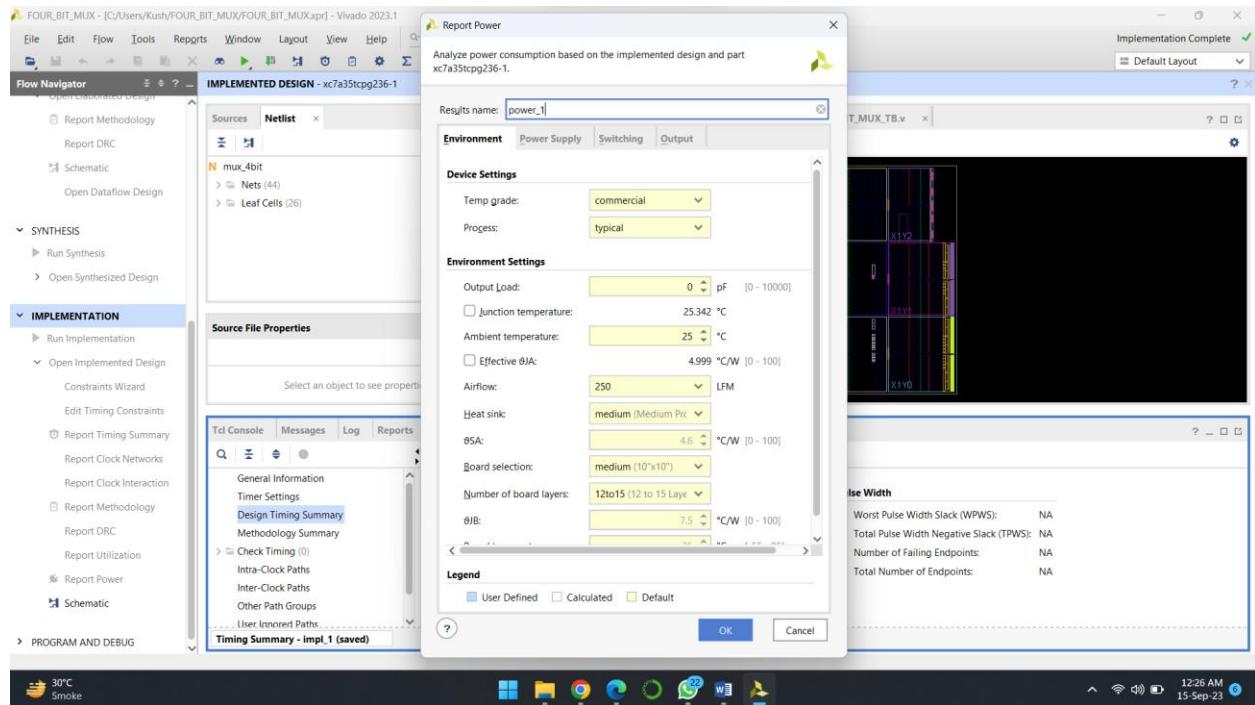
## WAVEFORM:

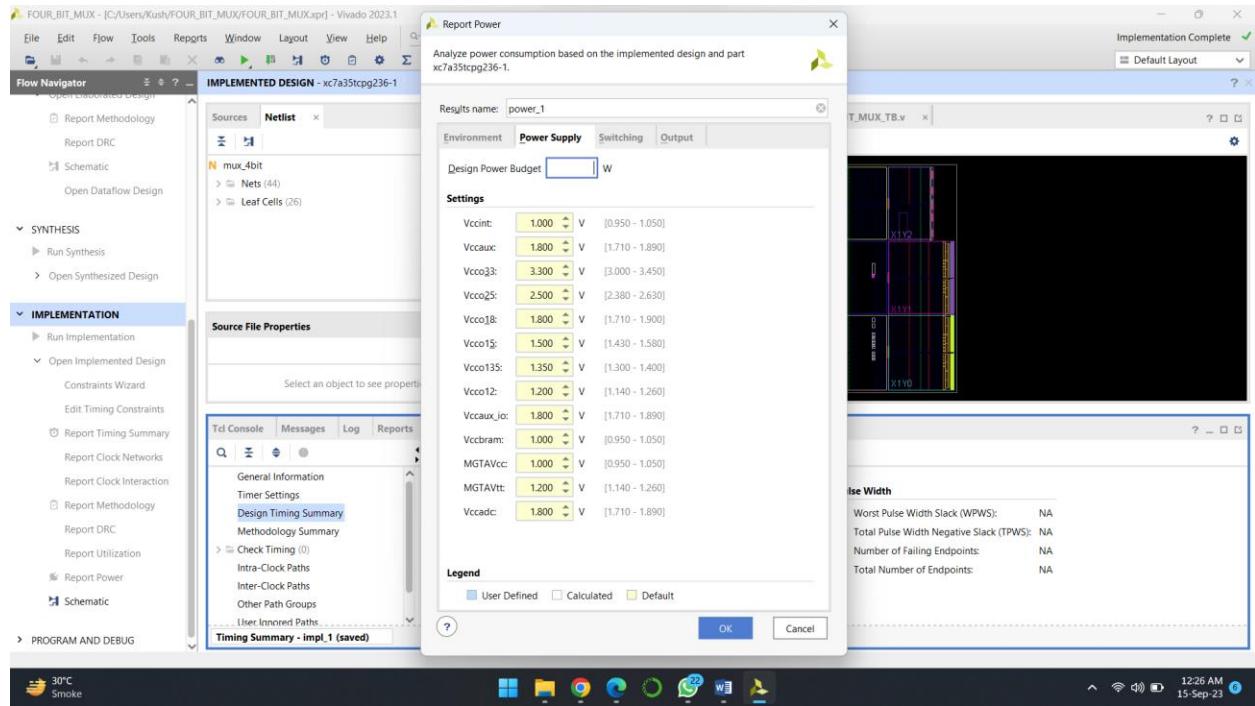


## RTL SCHEMATIC:



## POWER REPORT:





## 19. FIXED POINT DIVISION:

**CODE:**

```

module fixed_point_division (
    input wire [7:0] numerator,      // 8-bit input numerator
    input wire [7:0] denominator,    // 8-bit input denominator
    output wire [7:0] quotient     // 8-bit output quotient
);
    reg [15:0] remainder;          // 16-bit remainder
    reg [7:0] temp_quotient;      // 8-bit temporary quotient
    reg [3:0] shift_count;        // 4-bit shift counter
    reg [1:0] sign;

    always @ (numerator, denominator) begin
        remainder <= numerator;
        temp_quotient <= 0;
        shift_count <= 0;
        sign <= 0;

        // Perform division
        while (shift_count < 8) begin
            if (remainder >= denominator) begin
                remainder <= remainder - denominator;
                temp_quotient <= temp_quotient + (1 << (7 - shift_count));
            end
            remainder <= remainder << 1;
            shift_count <= shift_count + 1;
        end
        assign quotient = temp_quotient;
    end
endmodule

```

## TESTBENCH:

FIXED POINT DIVISION - [C:/Users/Kush/FIXED POINT DIVISION/FIXED POINT DIVISION.xpr] - Vivado 2023.1

Synthesis Failed

Default Layout

**PROJECT MANAGER - FIXED POINT DIVISION**

```

module tb_fixed_point_division;
    // Parameters
    reg [7:0] numerator;
    reg [7:0] denominator;
    wire [7:0] quotient;

    // Instantiate the fixed-point division module
    fixed_point_division uut (
        .numerator(numerator),
        .denominator(denominator),
        .quotient(quotient)
    );

    // Clock generation
    reg clock = 0;
    always #5 clock = ~clock;

    // Test stimulus
    initial begin
        $monitor("Time=%t: Numerator=%h, Denominator=%h, Quotient=%h", $time, numerator, denominator, quotient);
        // Test case 1: Divide 10 by 2
        numerator = 8'h0A;
        denominator = 8'h02;
        #10;
        // Test case 2: Divide 15 by 3
        numerator = 8'h0F;
        denominator = 8'h03;
        #10;
        // Test case 3: Divide 7 by 4
        numerator = 8'h07;
        denominator = 8'h04;
        #10;
        $finish;
    end
endmodule

```

Tcl Console Messages Log Reports Design Runs

23:1 Insert Verilog

12:55 AM 15-Sep-23

FIXED POINT DIVISION - [C:/Users/Kush/FIXED POINT DIVISION/FIXED POINT DIVISION.xpr] - Vivado 2023.1

Synthesis Failed

Default Layout

**PROJECT MANAGER - FIXED POINT DIVISION**

```

.quotient(quotient);
);

// Clock generation
reg clock = 0;
always #5 clock = ~clock;

// Test stimulus
initial begin
    $monitor("Time=%t: Numerator=%h, Denominator=%h, Quotient=%h", $time, numerator, denominator, quotient);
    // Test case 1: Divide 10 by 2
    numerator = 8'h0A;
    denominator = 8'h02;
    #10;
    // Test case 2: Divide 15 by 3
    numerator = 8'h0F;
    denominator = 8'h03;
    #10;
    // Test case 3: Divide 7 by 4
    numerator = 8'h07;
    denominator = 8'h04;
    #10;
    $finish;
end
endmodule

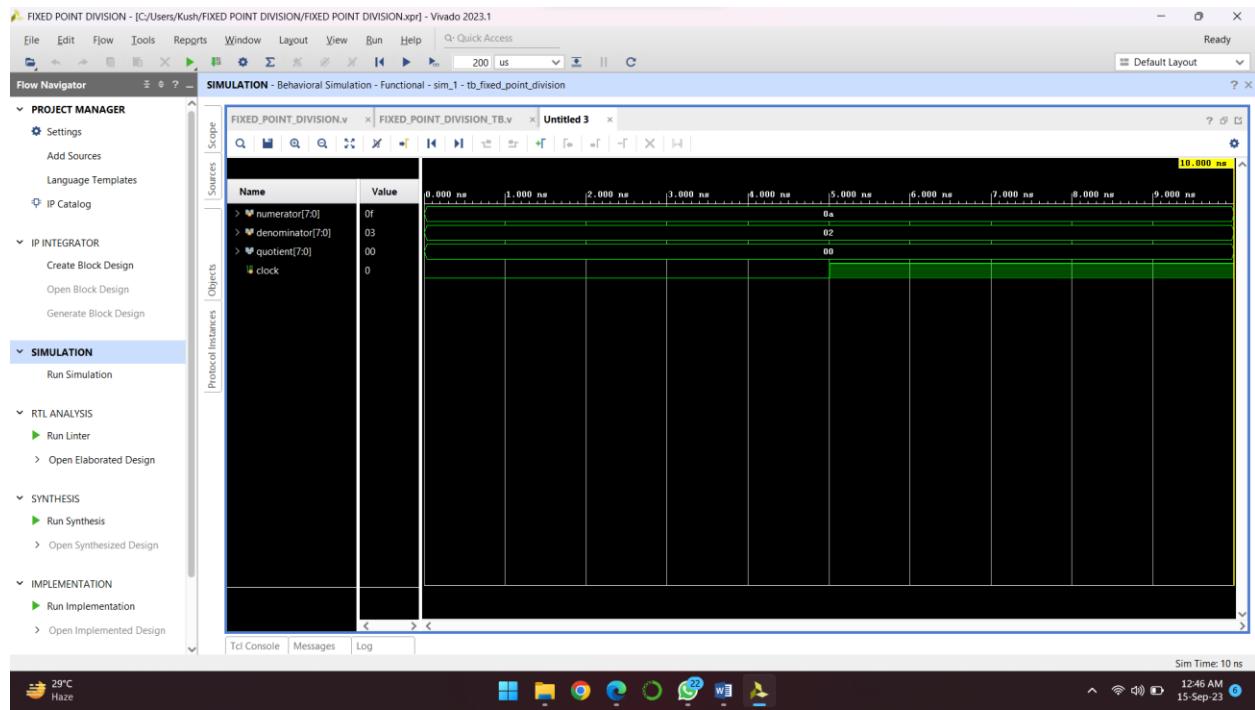
```

Tcl Console Messages Log Reports Design Runs

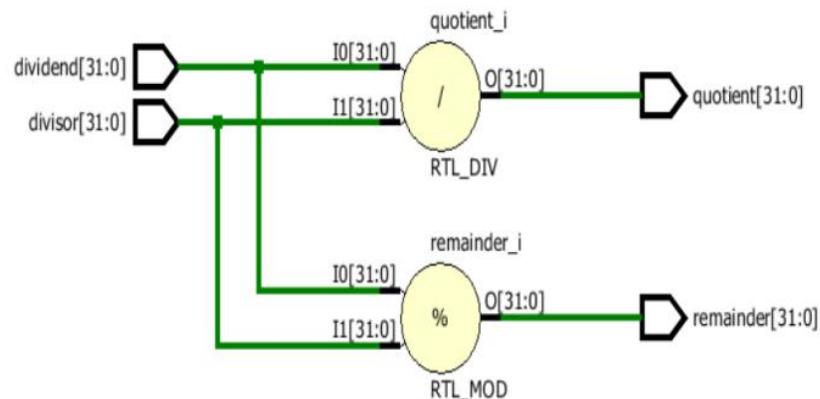
23:1 Insert Verilog

12:55 AM 15-Sep-23

## WAVEFORM:



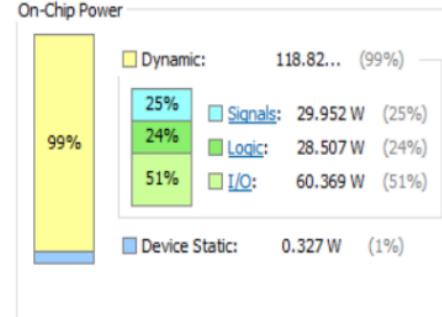
## RTL SCHEMATIC:



## POWER REPORT:

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

**Total On-Chip Power:** **119.155 W**  
**Junction Temperature:** **125.0 °C**  
 Thermal Margin: **-494.4 °C (-103.3 W)**  
 Effective ΔJA: **4.8 °C/W**  
 Power supplied to off-chip devices: **0 W**  
 Confidence level: **Low**



## 20. MASTER SLAVE JK FLIP FLOP:

CODE:

```

11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module JK_FlipFlop.MasterSlave (
24     input wire J, // Input J
25     input wire K, // Input K
26     input wire CLK, // Clock input
27     input wire CLR, // Clear input
28     output wire Q, // Output Q
29     output wire Qn // Output Q' (complement of Q)
30 );
31
32 reg Q_next; // Next state of Q
33
34 always @ (posedge CLK or posedge CLR) begin
35     if (CLR) begin
36         Q_next <= 1'b0; // Clear the flip-flop
37     end else begin
38         case ({J, K})
39             2'b00: Q_next <= Q_next; // No change
40             2'b01: Q_next <= 1'b1; // Clear (K=1)
41             2'b10: Q_next <= 1'b1; // Set (J=1)
42             2'b11: Q_next <= ~Q_next; // Toggle (J=K=1)
43         endcase
44     end
45 end
46
47 assign Q = Q_next;
48 assign Qn = ~Q_next;
49
50 endmodule
51 <
```

## TESTBENCH:

MASTERSLAVE JK FLIPFLOP - [C:/Users/Kush/MASTERSLAVE JK FLIPFLOP/MASTERSLAVE JK FLIPFLOP.xpr] - Vivado 2023.1

SIMULATION - Behavioral Simulation - Functional - sim\_1 - testbench

```

MASTERSLAVE_JK.v | MASTERSLAVE_JK_TB.v | Untitled 1 |
C:/Users/Kush/MASTERSLAVE JK FLIPFLOP/MASTERSLAVE JK FLIPFLOP.srsc/sim_1/new/MASTERSLAVE_JK_TB.v

19 : //////////////////////////////////////////////////////////////////
20 : module testbench();
21 :   // Inputs
22 :   reg J, K, CLR, Qn;
23 :   // Outputs
24 :   wire Q, Qn;
25 :
26 :
27 :
28 :
29 :
30 :
31 :   // Instantiate the JK Flip-Flop
32 :   JK_FlipFlop_MasterSlave jk_ff (
33 :     .J(J),
34 :     .K(K),
35 :     .CLR(CLR),
36 :     .Q(Q),
37 :     .Qn(Qn)
38 :   );
39 :
40 :   // Clock generation
41 :   reg clk = 0;
42 :   always begin
43 :     #5 clk = ~clk;
44 :   end
45 :
46 :   // Initialize Inputs
47 :   initial begin
48 :     J = 0;
49 :     K = 0;
50 :     CLR = 0;
51 :     CLK = 0;
52 :   end

```

Tcl Console | Messages | Log

Sim Time: 40 ns

29°C Haze

MASTERSLAVE JK FLIPFLOP - [C:/Users/Kush/MASTERSLAVE JK FLIPFLOP/MASTERSLAVE JK FLIPFLOP.xpr] - Vivado 2023.1

SIMULATION - Behavioral Simulation - Functional - sim\_1 - testbench

```

MASTERSLAVE_JK.v | MASTERSLAVE_JK_TB.v | Untitled 1 |
C:/Users/Kush/MASTERSLAVE JK FLIPFLOP/MASTERSLAVE JK FLIPFLOP.srsc/sim_1/new/MASTERSLAVE_JK_TB.v

43 :
44 :   // Initialize Inputs
45 :   initial begin
46 :     J = 0;
47 :     K = 0;
48 :     CLR = 0;
49 :     CLK = 0;
50 :
51 :
52 :   // Monitor the signals
53 :   $monitor("%time%#b J=%b K=%b CLR=%b CLK=%b Q=%b Qn=%b", $time, J, K, CLR, CLK, Q, Qn);
54 :
55 :   // Test case 1: Set
56 :   J = 1;
57 :   K = 0;
58 :   CLR = 0;
59 :   #10;
60 :
61 :   // Test case 2: Toggle
62 :   J = 1;
63 :   K = 1;
64 :   CLR = 0;
65 :   #10;
66 :
67 :   // Test case 3: Clear
68 :   J = 0;
69 :   K = 1;
70 :   CLR = 1;
71 :   #10;
72 :
73 :   // Test case 4: No Change
74 :   J = 0;
75 :   K = 0;
76 :

```

Tcl Console | Messages | Log

Sim Time: 40 ns

29°C Haze

MASTERSLAVE JK FLIPFLOP - [C:/Users/Kush/MASTERSLAVE JK FLIPFLOP/MASTERSLAVE JK FLIPFLOP.xpr] - Vivado 2023.1

File Edit Flow Tools Reports Window Layout View Run Help Q: Quick Access

200 us Default Layout

**Flow Navigator**

- PROJECT MANAGER
  - Settings
  - Add Sources
  - Language Templates
  - IP Catalog
- IP INTEGRATOR
  - Create Block Design
  - Open Block Design
  - Generate Block Design
- SIMULATION**
  - Run Simulation
- RTL ANALYSIS
  - Run Linter
  - Open Elaborated Design
- SYNTHESIS
  - Run Synthesis
  - Open Synthesized Design
- IMPLEMENTATION
  - Run Implementation
  - Open Implemented Design

**SIMULATION - Behavioral Simulation - Functional - sim\_1 - testbench**

MASTERSLAVE\_JK.v x MASTERSLAVE\_JK\_TB.v x Untitled 1 x

C:/Users/Kush/MASTERSLAVE JK FLIPFLOP/MASTERSLAVE JK FLIPFLOP.srscs/sim\_1/new/MASTERSLAVE\_JK\_TB.v

```

59: // Test case 2: Toggle
60: J = 1;
61: K = 1;
62: CLR = 0;
63: #10;
64:
65: // Test case 3: Clear
66: J = 0;
67: K = 1;
68: CLR = 1;
69: #10;
70:
71: // Test case 4: No Change
72: J = 0;
73: K = 0;
74: CLR = 0;
75: #10;
76:
77: // Terminate simulation
78: $finish;
79: end
80:
81:
82:
83: // Clock generation
84: always begin
85: #5 CLR = ~clk;
86: end
87:
88: endmodule
89:
90:
91:
92:
93:
94:
95:
96:
97:
98:
99:

```

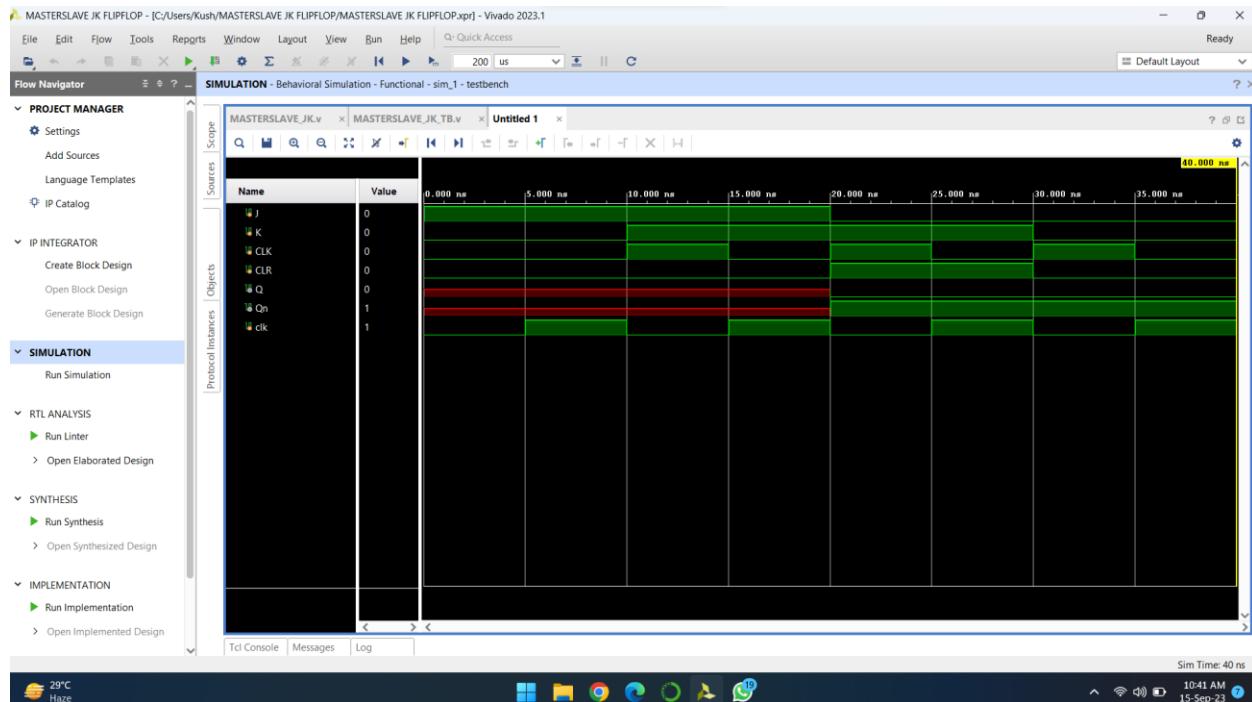
Tcl Console Messages Log

Sim Time: 40 ns

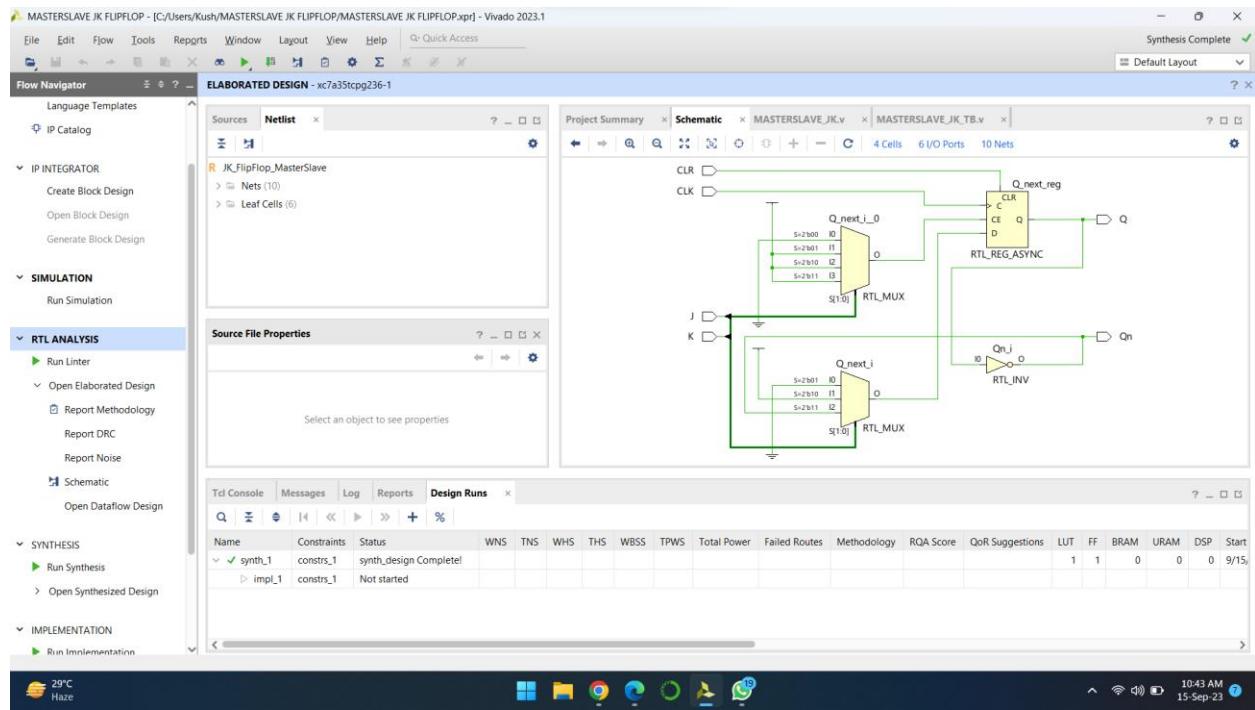
29°C Haze

10:42 AM 15-Sep-23

## WAVEFORM:



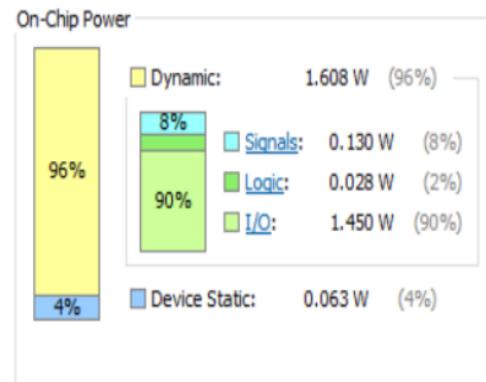
## RTL SCHEMATIC:



## POWER REPORT:

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

**Total On-Chip Power:** 1.67 W  
**Junction Temperature:** 33.0 °C  
 Thermal Margin: 67.0 °C (13.9 W)  
 Effective  $\Delta T$ : 4.8 °C/W  
 Power supplied to off-chip devices: 0 W  
 Confidence level: Low



## 21. POSITIVE EDGE DETECTOR:

CODE:

The screenshot shows a code editor window with the following details:

- Tab bar: `posedge_detct.v` (active), `posedge_tb.v`, Untitled 1.
- File path: C:/Users/DELL/positive edge detector/positive edge detector.srccs/sources\_1/new/posedge\_detct.v
- Toolbar icons: Q, file, back, forward, search, etc.
- Code content:

```
1 | module edge_detector(
2 |   input  data,
3 |   input  clock,
4 |   output edge_detect
5 | );
6 |   reg data_d;
7 |
8 |   always @ (posedge clock) begin
9 |     data_d <= data;
10|   end
11|   assign edge_detect = data & ~data_d;
12| endmodule
```

The code defines a Verilog module named `edge_detector`. It has three ports: `data` (input), `clock` (input), and `edge_detect` (output). Inside the module, there is a register `data_d` and an `always` block that updates `data_d` to the value of `data` whenever the `clock` signal rises. The output `edge_detect` is assigned the value of `data` ANDed with the complement of `data_d`.

### TESTBENCH:

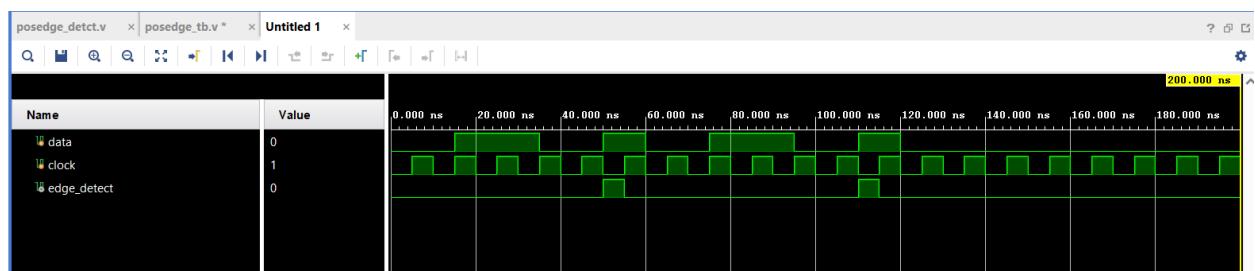
The screenshot shows a text editor window with three tabs at the top: "posedge\_detct.v", "posedge\_tb.v \*", and "Untitled 1". The current file is "posedge\_tb.v". The code is as follows:

```
posedge_detct.v  x  posedge_tb.v *  x  Untitled 1  x
C:/Users/DELL/positive edge detector/positive edge detector.srcs/sim_1/new/posedge_tb.v

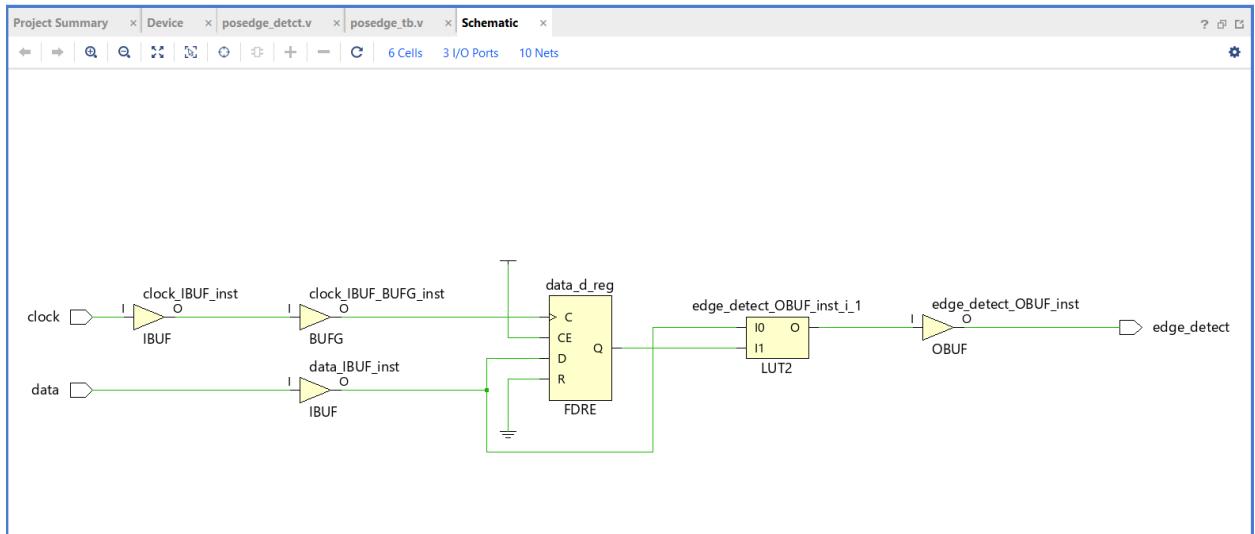
Q | F | ← | → | X | D | B | X | // | E | ? |

1 module edge_tb;
2
3     reg data;
4     reg clock;
5     wire edge_detect;
6     edge_detector uut (
7         .data(data),
8         .clock(clock),
9         .edge_detect(edge_detect)
10    );
11 initial begin
12     data = 0;
13     clock = 0;
14     #15 data = 1;
15     #20 data= 0;
16     #15 data = 1;
17     #10 data = 0;
18     #15 data = 1;
19     #20 data= 0;
20     #15 data = 1;
21     #10 data = 0;
22 end
23
24 always #5 clock=~clock;
25 initial begin
26     $monitor("Data =%b, Edge_detect=%b ", data,edge_detect);
27     #200 $finish;
28 end
29
30 endmodule
```

### WAVEFORM:



## RTL SCHEMATIC:



## POWER REPORT:

## Report Power

X

Estimate power consumption based on the netlist design and part  
xc7vx485tffg1157-1.



Results name: power\_1

X

### Environment

### Power Supply

### Switching

### Output

#### Device Settings

Temp grade:

commercial

▼

Process:

typical

▼

#### Environment Settings

Output Load:

0

▼

pF

[0 - 10000]

Junction temperature:

25.336 °C

Ambient temperature:

25

▼

°C

Effective θJA:

1.398 °C/W [0 - 100]

Airflow:

250

▼

LFM

Heat sink:

medium (Medium Prc

▼

θSA:

2.4

▼

°C/W [0 - 100]

Board selection:

medium (10"x10")

▼

Number of board layers:

12to15 (12 to 15 Laye

▼

θJB:

2.3

▼

°C/W [0 - 100]

Board temperature:

25

▼

°C

[-55 - 85]

#### Legend

User Defined

Calculated

Default

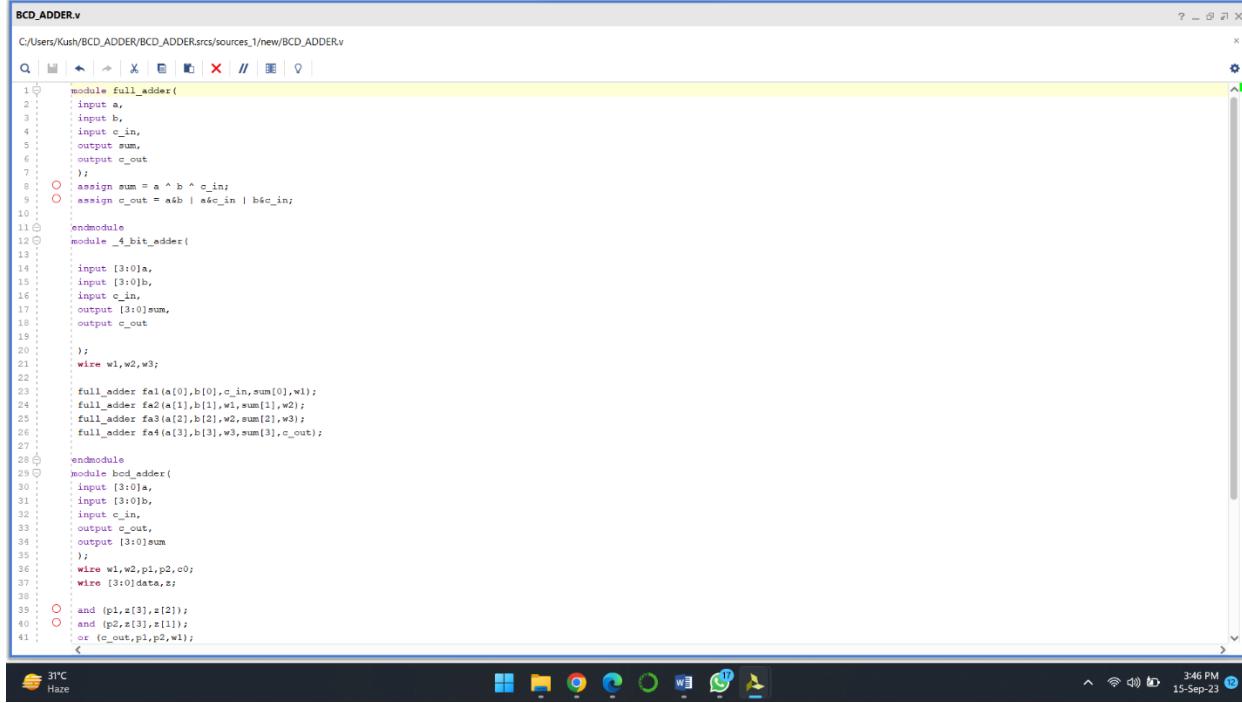


OK

Cancel

## 22. BCD ADDER:

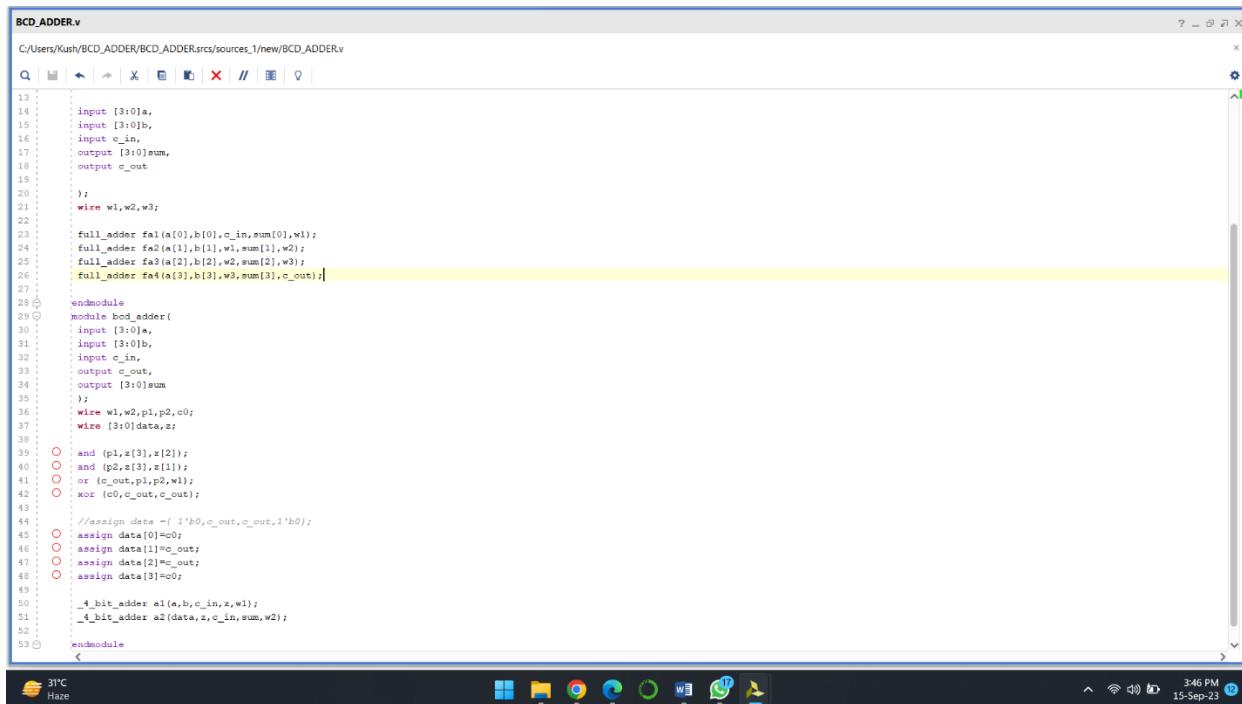
CODE:



```
BCD_ADDER.v
C:/Users/Kush/BCD_ADDER/BCD_ADDER.srs/sources_1/new/BCD_ADDER.v

1 module full_adder(
2     input a,
3     input b,
4     input c_in,
5     output sum,
6     output c_out
7 );
8     assign sum = a ^ b ^ c_in;
9     assign c_out = a&b | a&c_in | b&c_in;
10 endmodule
11 module _4_bit_adder(
12     input [3:0]a,
13     input [3:0]b,
14     input c_in,
15     output [3:0]sum,
16     output c_out
17 );
18     wire w1,w2,w3;
19
20     full_adder fa1(a[0],b[0],c_in,sum[0],w1);
21     full_adder fa2(a[1],b[1],w1,sum[1],w2);
22     full_adder fa3(a[2],b[2],w2,sum[2],w3);
23     full_adder fa4(a[3],b[3],w3,sum[3],c_out);
24 endmodule
25 module bcd_adder(
26     input [3:0]a,
27     input [3:0]b,
28     input c_in,
29     output c_out,
30     output [3:0]sum
31 );
32     wire w1,w2,p1,p2,c0;
33     wire [3:0]data,z;
34
35     assign (p1,z[3],z[2]);
36     assign (p2,z[3],z[1]);
37     assign (c_out,p1,p2,w1);
38     assign (c_out,p1,p2,w1);
39 endmodule

```

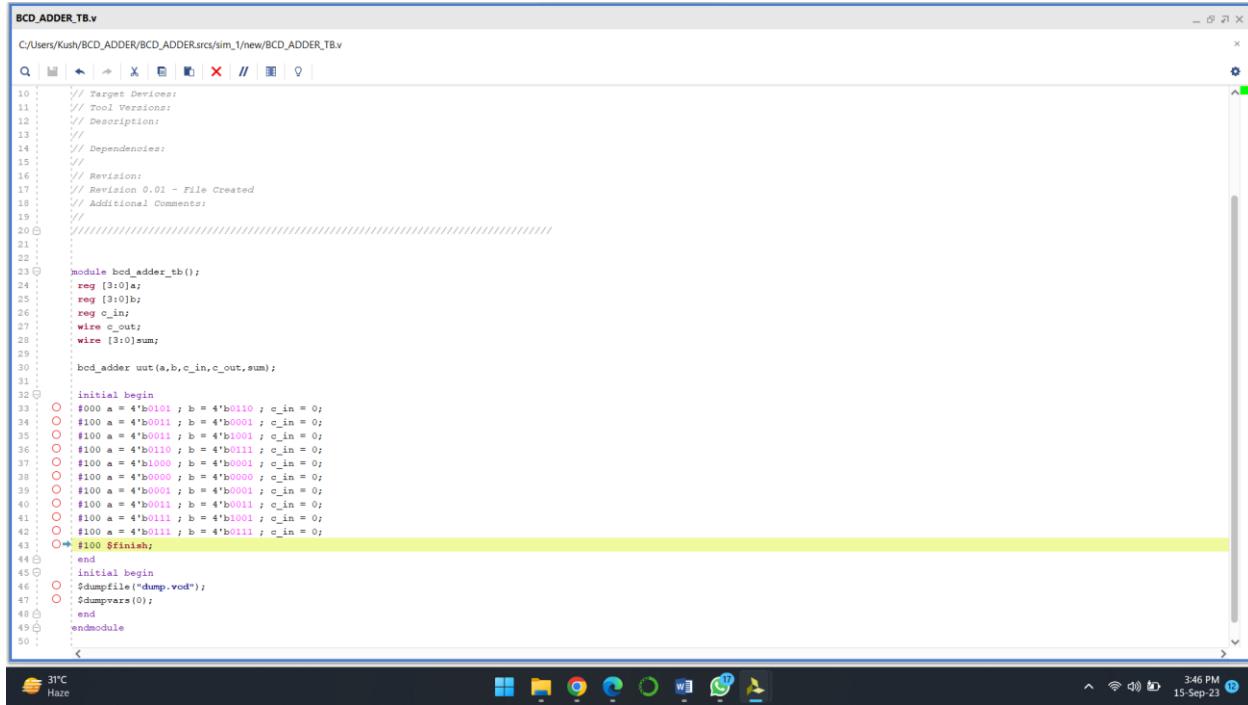


```
BCD_ADDER.v
C:/Users/Kush/BCD_ADDER/BCD_ADDER.srs/sources_1/new/BCD_ADDER.v

13 module full_adder(
14     input [3:0]a,
15     input [3:0]b,
16     input c_in,
17     output [3:0]sum,
18     output c_out
19 );
20     wire w1,w2,w3;
21
22     full_adder fa1(a[0],b[0],c_in,sum[0],w1);
23     full_adder fa2(a[1],b[1],w1,sum[1],w2);
24     full_adder fa3(a[2],b[2],w2,sum[2],w3);
25     full_adder fa4(a[3],b[3],w3,sum[3],c_out);
26 endmodule
27 module bcd_adder(
28     input [3:0]a,
29     input [3:0]b,
30     input c_in,
31     output c_out,
32     output [3:0]sum
33 );
34     wire w1,w2,p1,p2,c0;
35     wire [3:0]data,z;
36
37     assign (p1,z[3],z[2]);
38     assign (p2,z[3],z[1]);
39     assign (c_out,p1,p2,w1);
40     assign (c_out,p1,p2,w1);
41     xor (c0,c_out,c_out);
42
43     //assign data = ( 1'b0,c_out,c_out,1'b0 );
44     assign data[0]=0;
45     assign data[1]=c_out;
46     assign data[2]=c_out;
47     assign data[3]=0;
48
49     _4_bit_adder a1(a,b,c_in,z,w1);
50     _4_bit_adder a2(data,z,c_in,sum,w2);
51
52 endmodule

```

## TESTBENCH:

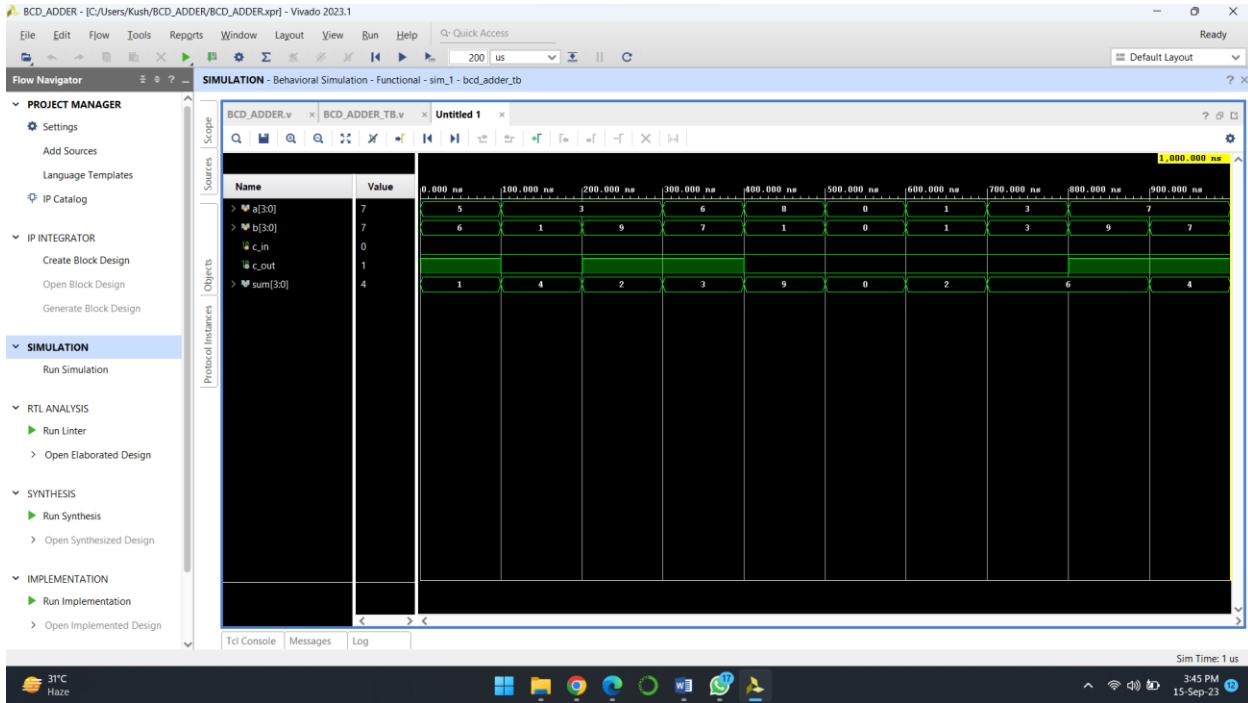


```
BCD_ADDER_TB.v
C:/Users/Kush/BCD_ADDER/BCD_ADDER.srsc/sim_1/new/BCD_ADDER_TB.v

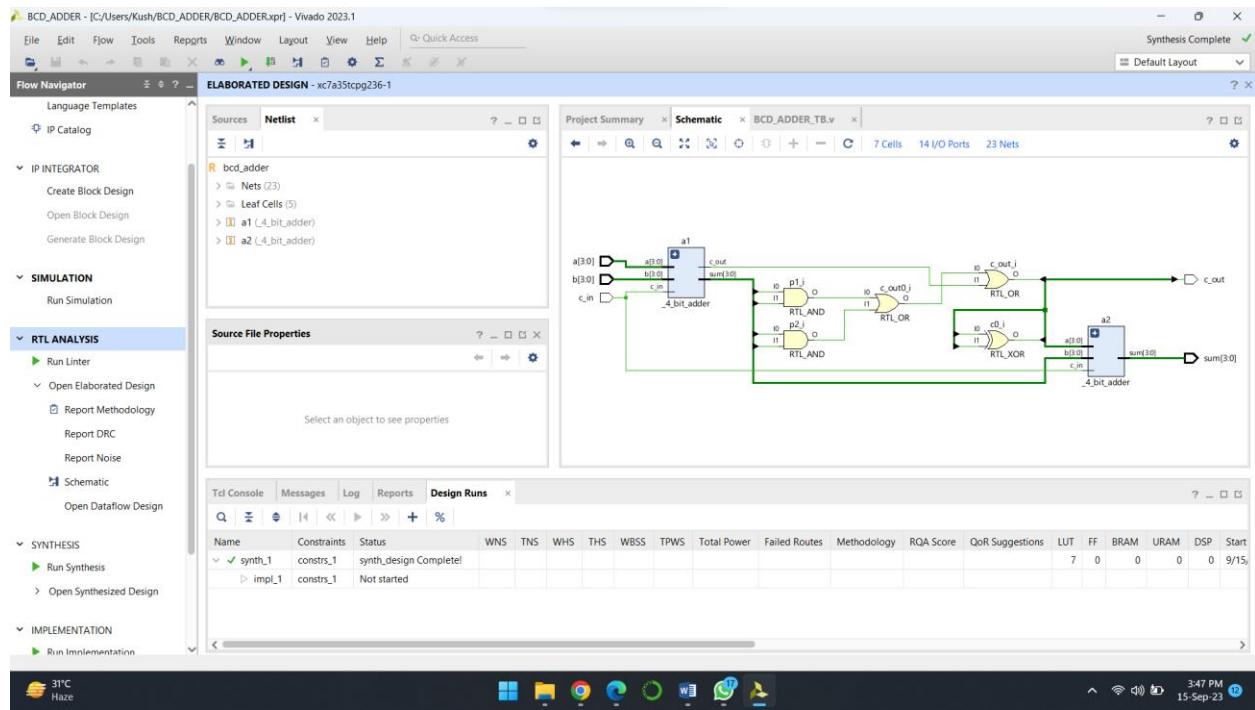
10 : // Target Devices:
11 : // Tool Versions:
12 : // Description:
13 : //
14 : // Dependencies:
15 : //
16 : // Revision:
17 : // Revision 0.01 - File Created
18 : // Additional Comments:
19 :
20 ///////////////////////////////////////////////////////////////////
21 :
22 //
23 module bcd_adder_tb();
24     reg [3:0]a;
25     reg [3:0]b;
26     reg c_in;
27     wire c_out;
28     wire [3:0]sum;
29
30     bcd_adder uut(a,b,c_in,c_out,sum);
31
32 initial begin
33     $000 a = 4'b0101 ; b = 4'b0110 ; c_in = 0;
34     $100 a = 4'b0011 ; b = 4'b0001 ; c_in = 0;
35     $100 a = 4'b0101 ; b = 4'b1001 ; c_in = 0;
36     $100 a = 4'b1101 ; b = 4'b0111 ; c_in = 0;
37     $100 a = 4'b1000 ; b = 4'b0001 ; c_in = 0;
38     $100 a = 4'b0000 ; b = 4'b0000 ; c_in = 0;
39     $100 a = 4'b0001 ; b = 4'b0001 ; c_in = 0;
40     $100 a = 4'b0101 ; b = 4'b0011 ; c_in = 0;
41     $100 a = 4'b1111 ; b = 4'b1001 ; c_in = 0;
42     $100 a = 4'b1111 ; b = 4'b1111 ; c_in = 0;
43     #100 $finish;
44 end
45 initial begin
46     $dumpfile("dmp.vcd");
47     $dumpvars(0);
48 end
49 endmodule
50

```

## WAVEFORM:



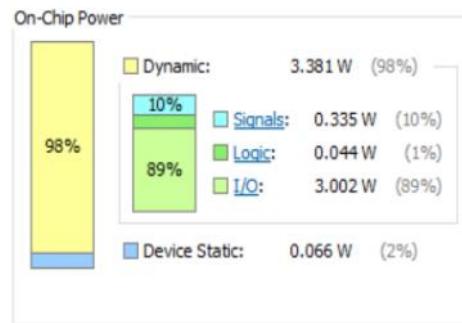
## RTL SCHEMATIC:



## POWER REPORT:

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

**Total On-Chip Power:** 3.448 W  
**Junction Temperature:** 41.5 °C  
 Thermal Margin: 58.5 °C (12.2 W)  
 Effective ΔTJA: 4.8 °C/W  
 Power supplied to off-chip devices: 0 W  
 Confidence level: LOW



## 23. 4-BIT CARRY SELECT ADDER:

CODE:

```
module four_bit_adder(x,y,cin,sum,cout);
input [3:0] x,y;
input cin;
output [3:0] sum;
output cout;
wire c0,c1,c2;
full_adder fa0(.a(x[0]),.b(y[0]),.c(cin),.s(sum[0]),.co(c0));
full_adder fa1(.a(x[1]),.b(y[1]),.c(c0),.s(sum[1]),.co(c1));
full_adder fa2(.a(x[2]),.b(y[2]),.c(c1),.s(sum[2]),.co(c2));
full_adder fa3(.a(x[3]),.b(y[3]),.c(c2),.s(sum[3]),.co(cout));
endmodule

module BCD_adder(S, C, A, B, C0);
input [3:0] A, B;
input C0;
output [3:0] S;
output C;
wire C1, C2, C3, C4, C5;
wire [3:0]X, Z;
and (C1, Z[3], Z[2]);
and (C2, Z[3], Z[1]);
or (C, C3, C1,C2);
xor (C5, C, C);
assign X[2] = C;
assign X[1] = C;
assign X[3] = C5;
assign X[0] = C5;
four_bit_adder F_1 ( A, B, C0,Z,C3);
four_bit_adder F_2 (X, Z, C0,S,C4);
endmodule
```

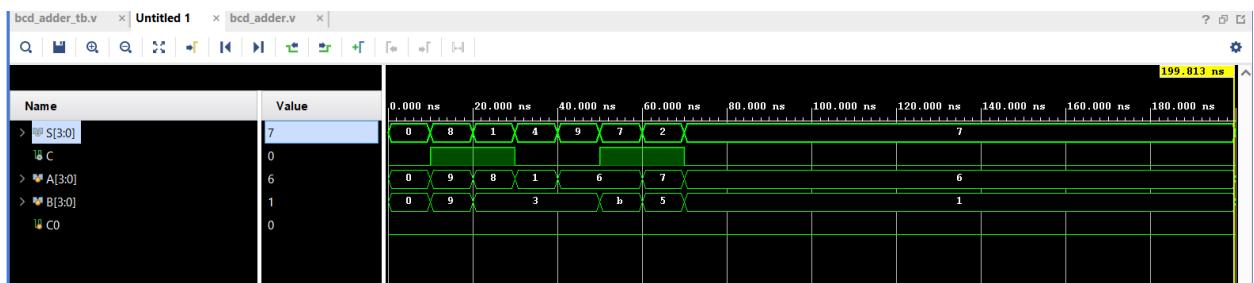
TESTBENCH:

```

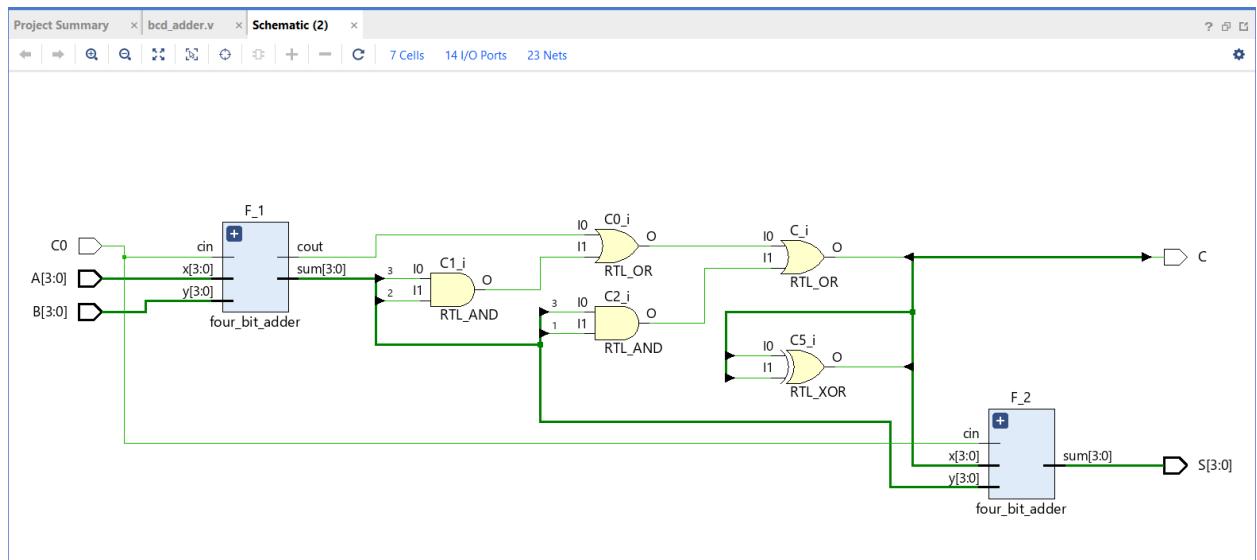
module BCD_adder_tb;
wire [3:0] S;
wire C;
reg [3:0]A, B;
reg C0;
BCD_adder F1(S, C, A, B, C0);
initial
begin
A[3:0] = 4'b0000; B = 4'b0000; C0 = 1'b0;
#10 A[3:0] = 4'b1001; B = 4'b1001; C0 = 1'b0;
#10 A[3:0] = 4'b1000; B = 4'b0011; C0 = 1'b0;
#10 A[3:0] = 4'b0001; B = 4'b0011; C0 = 1'b0;
#10 A[3:0] = 4'b0110; B = 4'b0011; C0 = 1'b0;
#10 A[3:0] = 4'b0110; B = 4'b1011; C0 = 1'b0;
#10 A[3:0] = 4'b0111; B = 4'b0101; C0 = 1'b0;
#10 A[3:0] = 4'b0110; B = 4'b0001; C0 = 1'b0;
end
initial
begin
$monitor("A=%b| B=%b| Cin=%b| BCD Sum=%b| Cout=%b",A,B,C0,S,C);
#200 $finish;
end
endmodule

```

### WAVEFORM:



## RTL SCHEMATIC:



## POWER REPORT:

Report Power X

Estimate power consumption based on the netlist design and part xc7vx48tff1157-1.

Results name:

**Environment** **Power Supply** **Switching** **Output**

**Device Settings**

Temp grade:  ▼

Process:  ▼

**Environment Settings**

Output Load:  pF [0 - 10000]

Junction temperature: 25.336 °C

Ambient temperature:  °C

Effective θJA: 1.398 °C/W [0 - 100]

Airflow:  LFM

Heat sink:  ▼

θSA:  °C/W [0 - 100]

Board selection:  ▼

Number of board layers:  ▼

θJB:  °C/W [0 - 100]

Board temperature:  °C [-55 - 85]

**Legend**

User Defined  Calculated  Default

? OK Cancel

## 24. MOORE FSM 1010 SEQUENCE DETECTOR:

CODE:

```
module moore_fsm(din, reset, clk, y);
input din;
input clk;
input reset;
output reg y;
reg [2:0] cst, nst;
localparam S0 = 3'b000,
           S1 = 3'b001,
           S2 = 3'b010,
           S3 = 3'b100,
           S4 = 3'b101;
always @(cst or din)
begin
  case (cst)
    S0: if (din == 1'b1)
        begin
          nst = S1;
          y=1'b0;
        end
      else nst = cst;
    S1: if (din == 1'b0)
        begin
          nst = S2;
          y=1'b0;
        end
      else
        begin
          nst = cst;
          y=1'b0;
        end
  endcase
  assign y = nst;
end
endmodule
```

```
s2: if (din == 1'b1)
      begin
        nst = s3;
        y=1'b0;
      end
      else
      begin
        nst = s0;
        y=1'b0;
      end
s3: if (din == 1'b0)
      begin
        nst = s4;
        y=1'b0;
      end
      else
      begin
        nst = s1;
        y=1'b0;
      end
s4: if (din == 1'b0)
      begin
        nst = s1;
        y=1'b1;
      end
      else
      begin
        nst = s3;
        y=1'b1;
      end
    default: nst = s0;

end
always@(posedge clk)
begin
  if (reset)
    cst <= s0;
  else
    cst <= nst;
end
endmodule
```

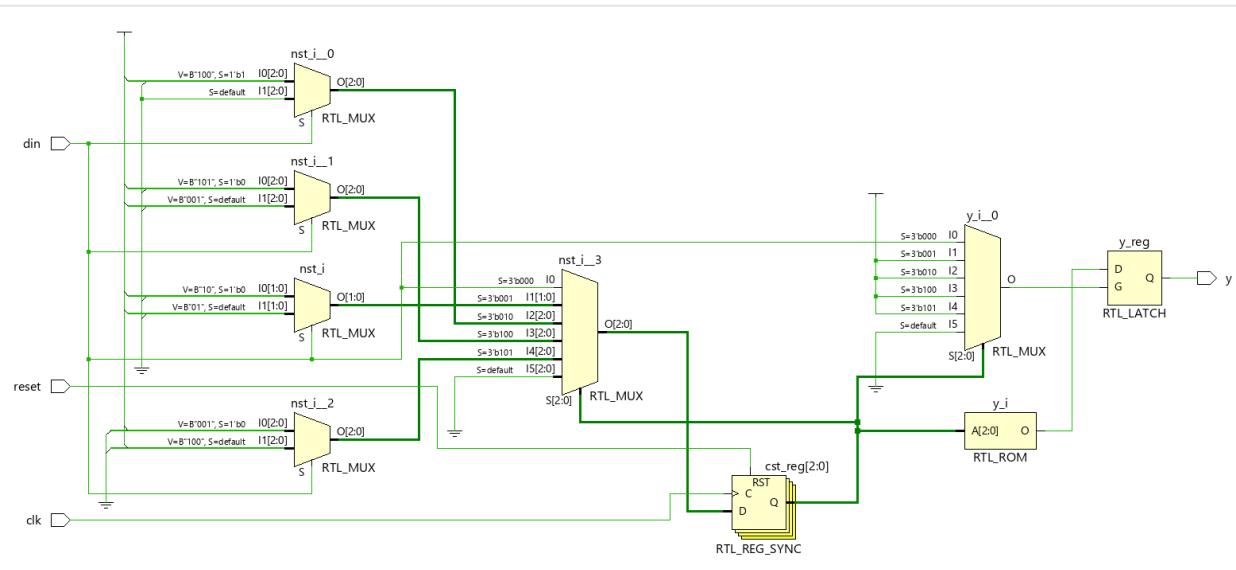
### TESTBENCH:

```
module moore_fsm_tb;
reg din,clk,reset;
wire y;
moore_fsm m1(din, reset, clk, y);
initial
begin
reset=0;clk=0;din=0;
$monitor($time, , "c=%b",clk,, "y=%b",y,, "r=%b",reset,, "d=%b",din);
#10 din=1;
#10 din=1;
#10 din=1;
#10 din=0;
#10 din=1;
#10 din=0;
#10 din=1;
#10 din=0;
#10 din=1;
#10 din=0;
#10 din=1;
end
always
#5 clk=~clk;
initial
#100 $finish ;
endmodule
```

### WAVEFORM:



## RTL SCHEMATIC:



## POWER REPORT:

Report Power X

Estimate power consumption based on the netlist design and part **xc7vx485tffn1157-1**.

Results name: **power\_1**

**Environment** **Power Supply** **Switching** **Output**

**Device Settings**

Temp grade: **commercial** ▾  
Process: **typical** ▾

**Environment Settings**

Output Load: **0** pF [0 - 10000]  
 Junction temperature: **25.336** °C  
Ambient temperature: **25** °C  
 Effective θJA: **1.398** °C/W [0 - 100]  
Airflow: **250** LFM  
Heat sink: **medium (Medium Prc)** ▾  
θSA: **2.4** °C/W [0 - 100]  
Board selection: **medium (10"x10")** ▾  
Number of board layers: **12to15 (12 to 15 Laye** ▾  
θJB: **2.3** °C/W [0 - 100]  
Board temperature: **25** °C [-55 - 85]

**Legend**

User Defined  Calculated  Default

? OK Cancel

## 25. N:1 MUX:

CODE:

The screenshot shows a text editor window with three tabs at the top: "n1\_mux.v" (selected), "n1\_mux\_tb.v", and "Untitled 3". The main area displays Verilog code for an N:1 MUX module. The code defines a module with inputs "in" and "sel", and an output "y". It uses a case statement to select one of the 16 inputs based on the value of "sel". The code is color-coded, with keywords like "module", "parameter", "input", "output", "always", "begin", "case", and "default" in blue; identifiers like "n1\_mux", "in", "sel", and "y" in black; and binary values in green. The code is as follows:

```
1 module n_1mux(in,sel,y);
2 parameter n=16;
3 input [n-1:0] in;
4 input [3:0] sel;
5 output reg y;
6
7 always @(in or sel)
8 begin
9 case(sel)
10 4'b0000: y= in[0];
11 4'b0001: y= in[1];
12 4'b0010: y= in[2];
13 4'b0011: y= in[3];
14 4'b0100: y= in[4];
15 4'b0101: y= in[5];
16 4'b0110: y= in[6];
17 4'b0111: y= in[7];
18 4'b1000: y= in[8];
19 4'b1001: y= in[9];
20 4'b1010: y= in[10];
21 4'b1011: y= in[11];
22 4'b1100: y= in[12];
23 4'b1101: y= in[13];
24 4'b1110: y= in[14];
25 4'b1111: y= in[n-1];
26 default: y=4'b0000;
27 endcase
28 end
29 endmodule
```

TESTBENCH:

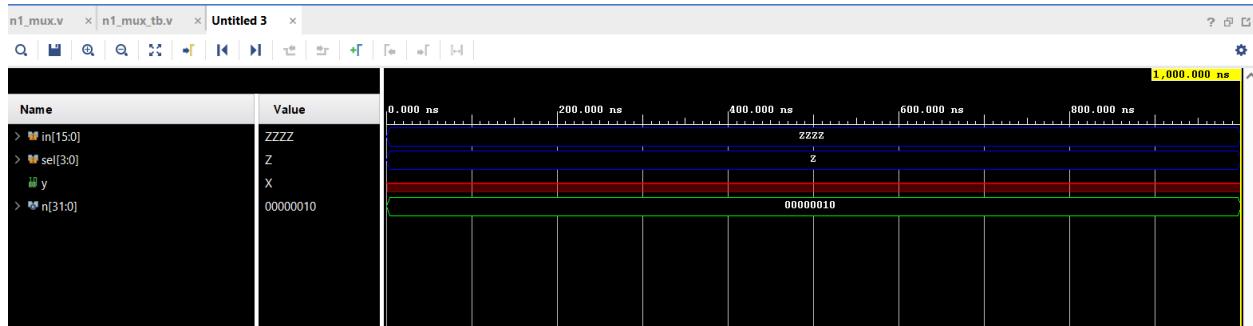
```
n1_mux.v  x  n1_mux_tb.v  x  Untitled 3  x
C:/Users/DELL/n1 mux/n1 mux.srccs/sim_1/new/n1_mux_tb.v

Q | H | ← | → | X | D | C | X | // | M | ? |

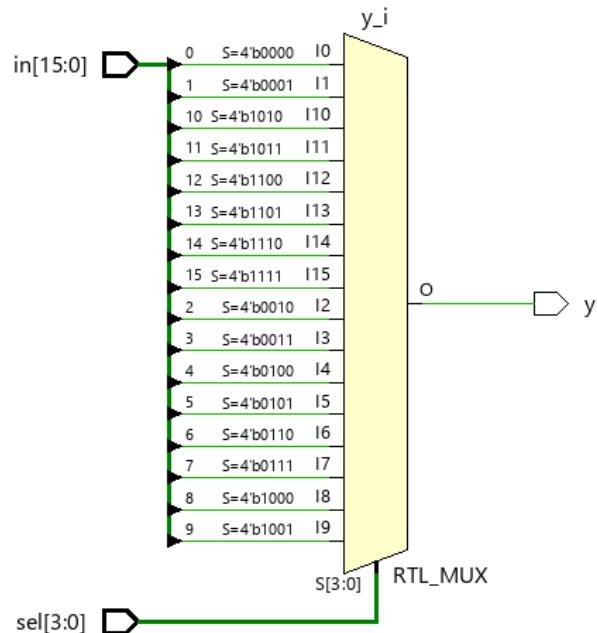
1 module n_1mux_tb;
2     reg [15:0] in;
3     reg [3:0] sel;
4     wire y;
5     n_1mux uut (
6         .in(in),
7         .sel(sel),
8         .y(y)
9     );
10    initial begin
11        in=16'b1000000000000000; sel=4'b0000;
12        #10 in=16'b0100000000000010; sel=4'b0001;
13        #10 in=16'b0010000000000100; sel=4'b0010;
14        #10 in=16'b0001000000000000; sel=4'b0011;
15        #10 in=16'b00001000000011000; sel=4'b0100;
16        #10 in=16'b0000010000000000; sel=4'b0101;
17        #10 in=16'b0000001000000000; sel=4'b0110;
18        #10 in=16'b0000000111100000; sel=4'b0111;
19        #10 in=16'b0000000010000000; sel=4'b1000;
20        #10 in=16'b0000000001000000; sel=4'b1001;
21        #10 in=16'b00000011000100000; sel=4'b1010;
22        #10 in=16'b0000000000100000; sel=4'b1011;
23        #10 in=16'b0000000000010000; sel=4'b1100;
24        #10 in=16'b0000000000001000; sel=4'b1101;
25        #10 in=16'b000000000000010; sel=4'b1110;
26        #10 in=16'b1000000000000001; sel=4'b1111;
27    end
28
29    initial
30    begin
31        $monitor("in=%b | sel=%b | out=%b",in,sel,y);
32    end

```

WAVEFORM:



RTL SCHEMATIC:



## POWER REPORT:

Report Power X

Estimate power consumption based on the netlist design and part **xc7vx485tfffa1157-1**.

Results name: **power\_1**

**Environment** **Power Supply** **Switching** **Output**

**Device Settings**

Temp grade: **commercial** ▾  
Process: **typical** ▾

**Environment Settings**

Output Load: **0 pF** [0 - 10000]  
 Junction temperature: **25.336 °C**  
Ambient temperature: **25 °C**  
 Effective  $\theta_{JA}$ : **1.398 °C/W [0 - 100]**  
Airflow: **250 LFM**  
Heat sink: **medium (Medium Prc)** ▾  
 $\theta_{SA}$ : **2.4 °C/W [0 - 100]**  
Board selection: **medium (10"x10")** ▾  
Number of board layers: **12to15 (12 to 15 Laye** ▾  
 $\theta_{JB}$ : **2.3 °C/W [0 - 100]**

**Legend**  
 User Defined  Calculated  Default

? OK Cancel



