# Digital System Design Applications

# Experiment 1 Report

## EHB436E

## CRN: 10345

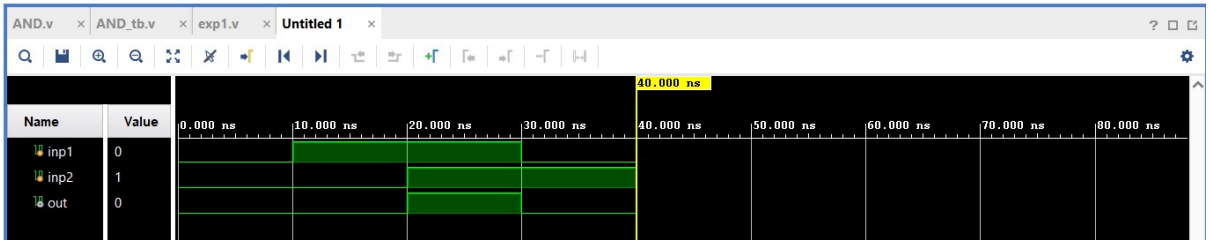
Hatice Nur Andı
040200203

**AND GATE**

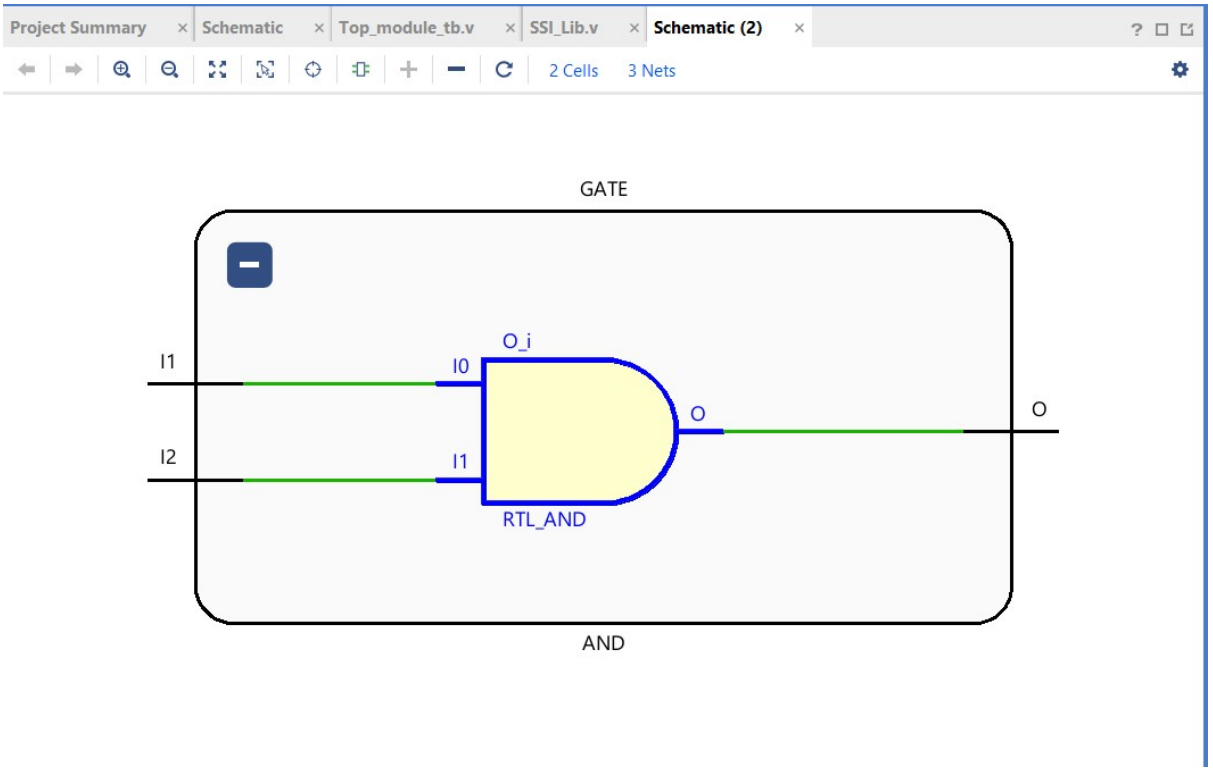
The first part of my report includes the verilog code, testbench code, behavioral simulation waves, RTL schematic, Technology Schematic, truth table, utilization summaries, combinational delays before and after implementation one by one.

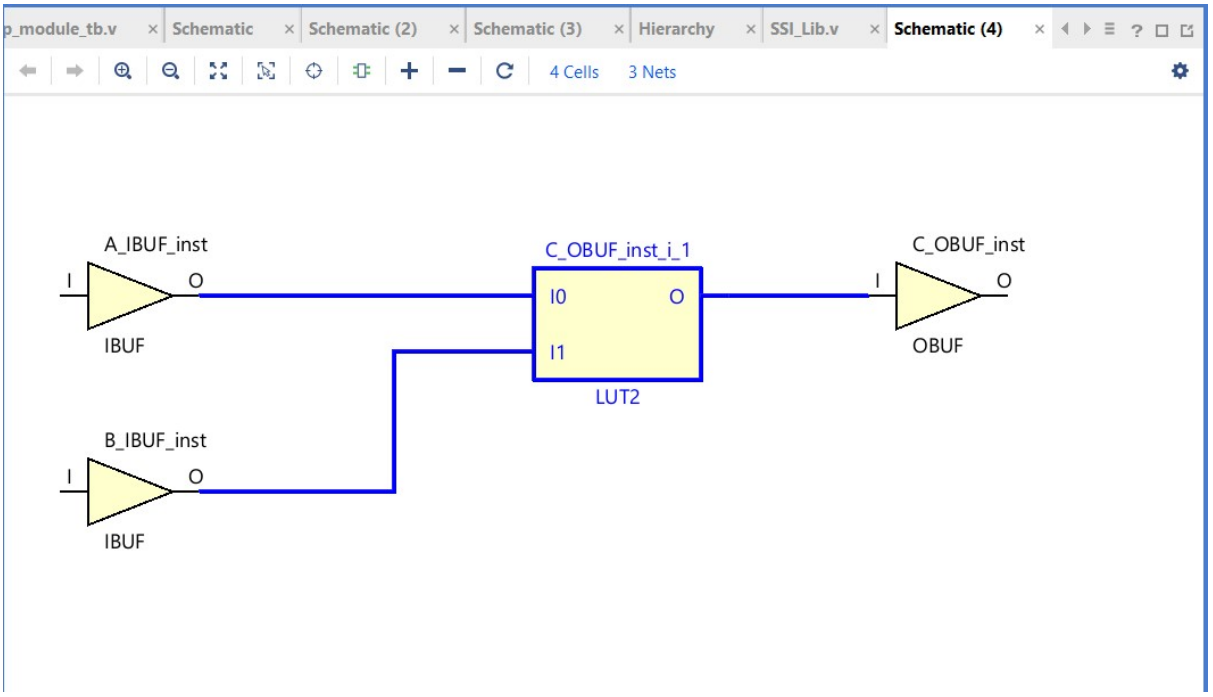
```verilog
`timescale 1ns / 1ps

module AND(
input I1,I2,
output O
    );

    assign O=I1&I2;

endmodule
```

```verilog
`timescale 1ns / 1ps

module AND_tb;

reg inp1,inp2;
wire out;

AND uut(.O(out), .I1(inp1), .I2(inp2));

initial
begin
inp1=0; inp2=0;
#10

inp1=1; inp2=0;
#10

inp1=1; inp2=1;
#10

inp1=0; inp2=1;
#10

$finish;
end
endmodule
```

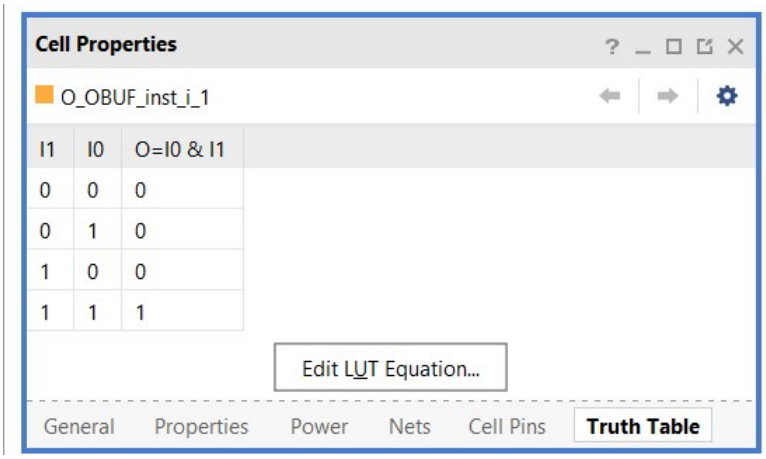



Verilog code, tesbench code and simulation wave



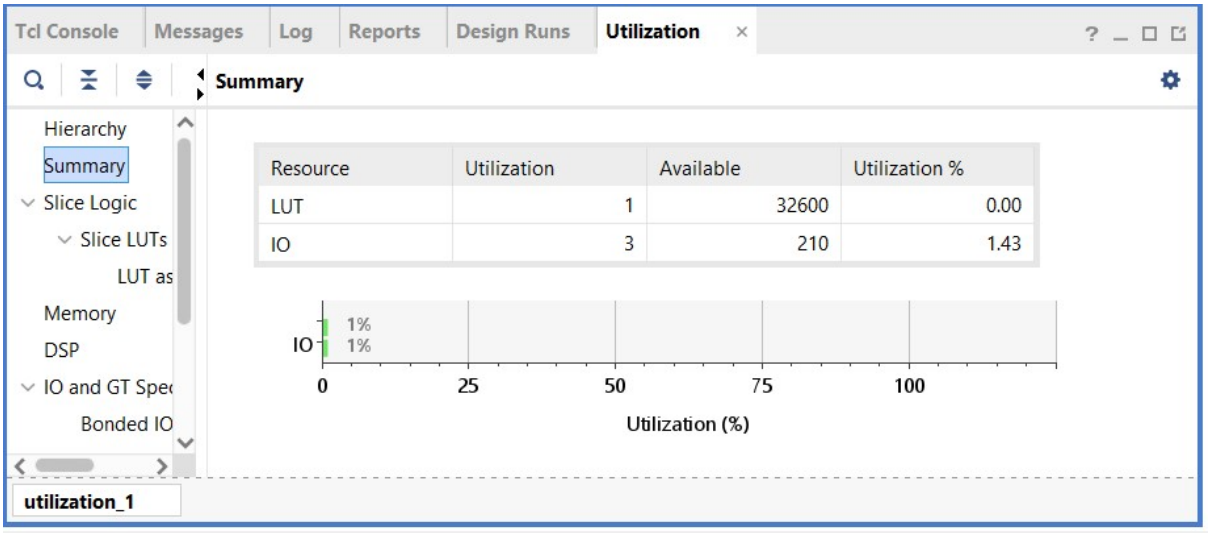




RTL and Technology schematic

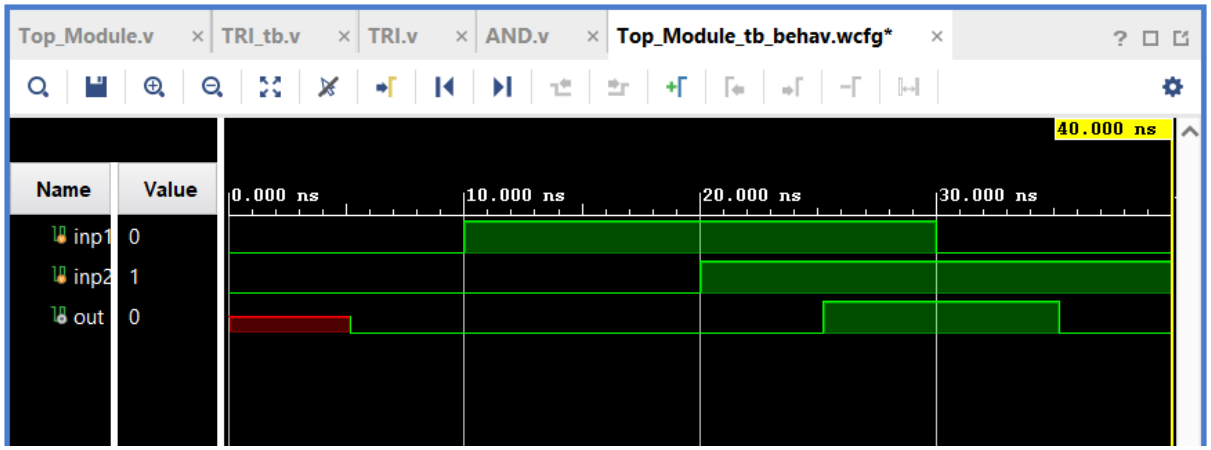



Truth Table

We can understand that RTL schematic and technology schematic have the same functionality from the truth table of LUT2. Truth table of LUT2 has the same functionality with the RTL_AND in the RTL schematic.




-Utilization Summary

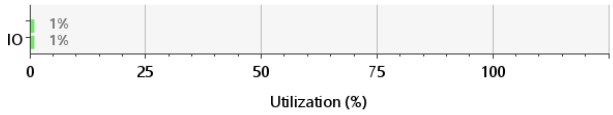



| From Port | To Port | Max Delay | Max Process Corner | Min Delay | Min Process Corner |
|---|---|---|---|---|---|
| I1 | O | 5.333 | SLOW | 2.074 | FAST |
| I2 | O | 5.333 | SLOW | 2.074 | FAST |

-Timing Report

**Summary**

| Resource | Utilization | Available | Utilization % |
|---|---|---|---|
| LUT | 1 | 32600 | 0.00 |
| IO | 3 | 210 | 1.43 |




**Hierarchy**

| Name | Slice LUTs (32600) | Slice (8150) | LUT as Logic (32600) | Bonded IOB (210) |
|---|---|---|---|---|
| AND | 1 | 1 | 1 | 3 |

**Combinational Delays**

| From Port | To Port | Max Delay | Max Process Corner | Min Delay | Min Process Corner |
|---|---|---|---|---|---|
| I1 | O | 6.378 | SLOW | 2.105 | FAST |
| I2 | O | 6.470 | SLOW | 2.125 | FAST |

-Implementation Report for AND GATE

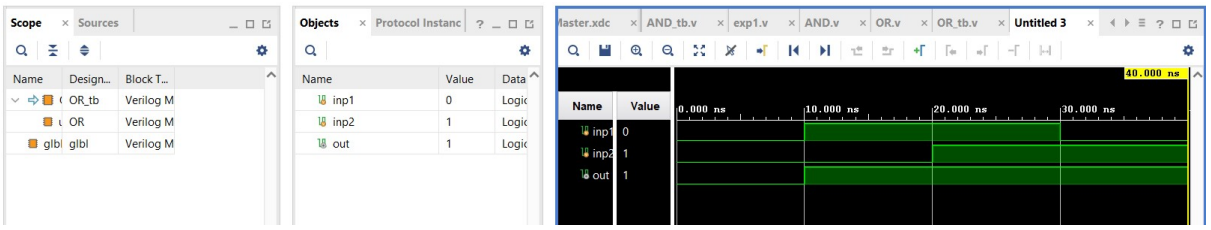
The maximum and minimum path delays increased. After the implementation, the physical placement of my LUTs on the board, their proximity to one another, and other factors result in delays.

The screenshots of codes,testbench codes, behavioral simulation wave, RTL schematic and technology schematic screenshots of gates and modules are represented after this part

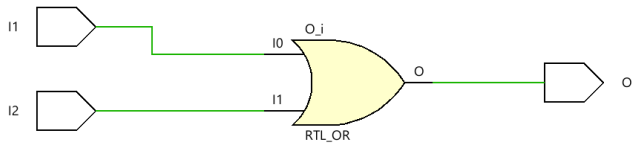
**OR GATE**

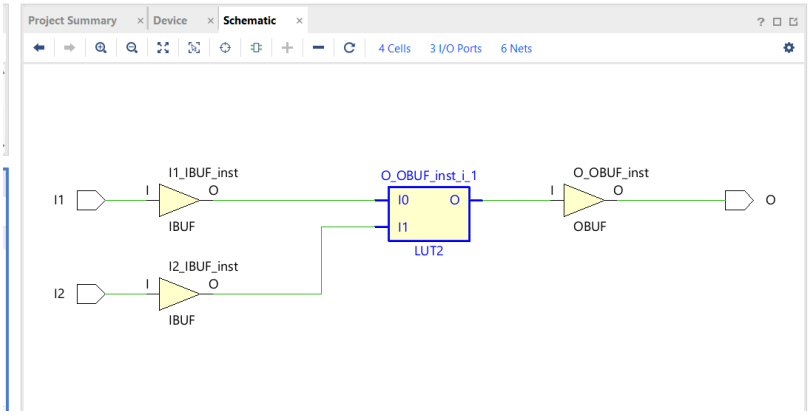
```
`timescale 1ns / 1ps

module OR(
input I1,I2,
output O
    );

    assign O=I1||I2;

endmodule
```

```
module OR_tb;

reg inp1,inp2;
wire out;

OR uut(.O(out), .I1(inp1), .I2(inp2));

initial
begin

inp1=0; inp2=0;
#10

inp1=1; inp2=0;
#10

inp1=1; inp2=1;
#10

inp1=0; inp2=1;
#10

$finish;

end
endmodule
```




Verilog Code, Testbench Code and Simulation Wave



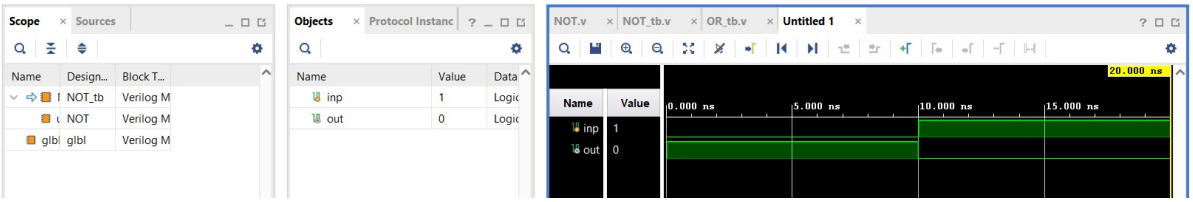




RTL and Technology Schematic

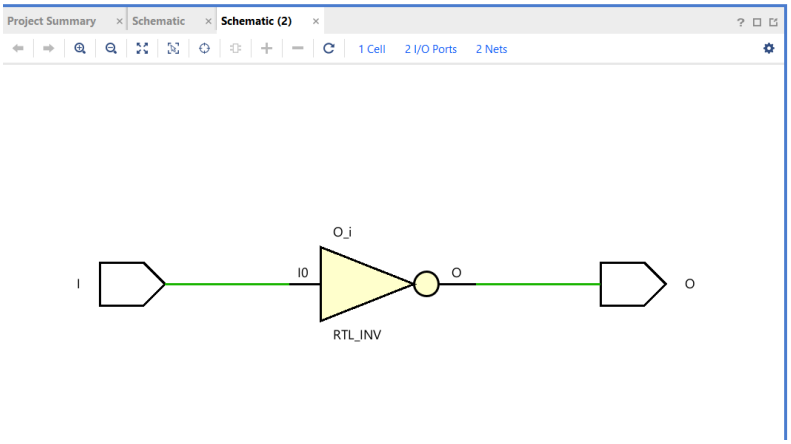
**NOT GATE**

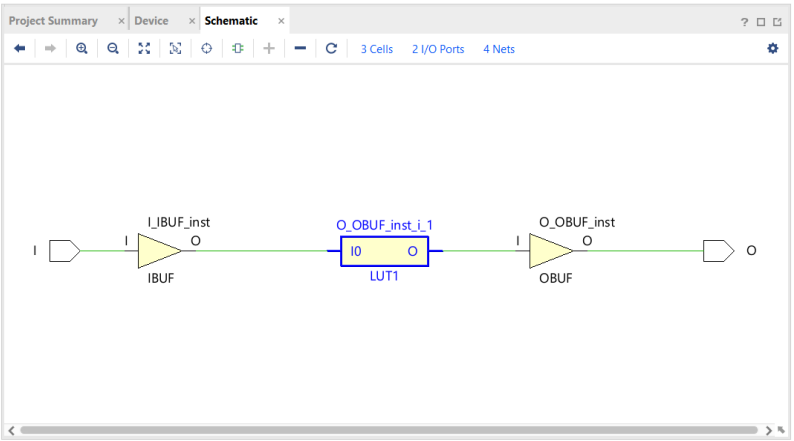
```
`timescale 1ns / 1ps

module NOT(

input I,
output O

    );

   assign O=~I;
endmodule
```

```
`timescale 1ns / 1ps

module NOT_tb;
reg inp;
wire out;

NOT uut(.I(inp), .O(out));

initial
begin

inp=0;
#10
inp=1;
#10
$finish;

end
endmodule
```




Verilog Code, Testbench Code and Simulation Wave



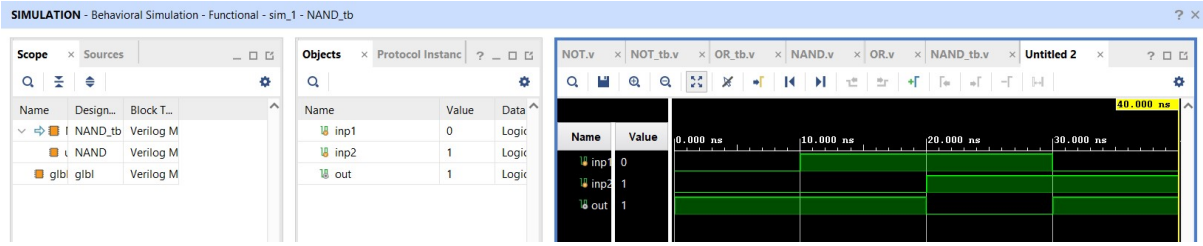




RTL and Technology Schematic

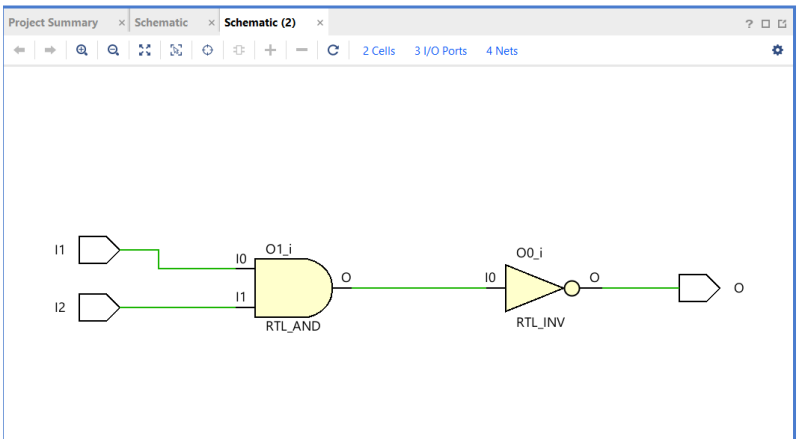
**NAND GATE**

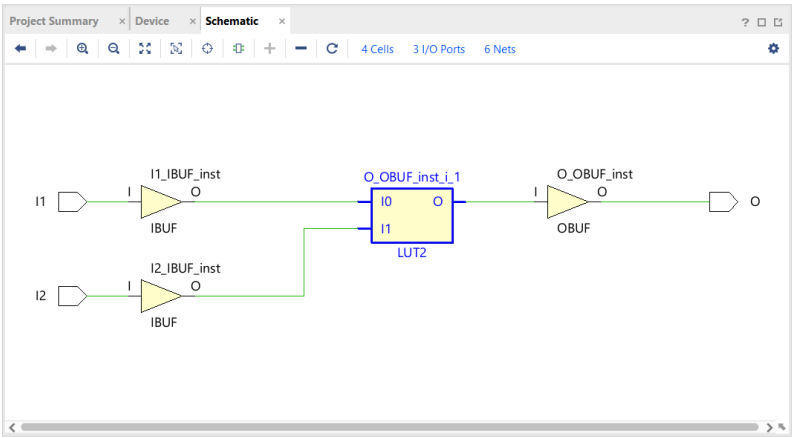
```
1  `timescale 1ns / 1ps
2
3  module NAND(
4  input I1,I2,
5  output reg O
6      );
7
8  always @ (I1 or I2)
9    begin
0        O =~(I1&I2);
1    end
2  endmodule
3
```

```
`timescale 1ns / 1ps

module NAND_tb;

reg inp1,inp2;
wire out;

NAND uut(.O(out), .I1(inp1), .I2(inp2));

initial
begin


inp1=0; inp2=0;
#10

inp1=1; inp2=0;
#10

inp1=1; inp2=1;
#10

inp1=0; inp2=1;
#10
$finish;

end
endmodule
```




Verilog Code, Testbench Code and Simulation Wave



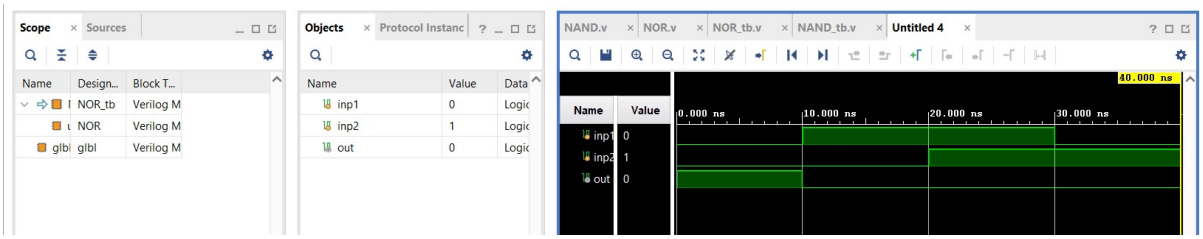




RTL and Technology Schematic

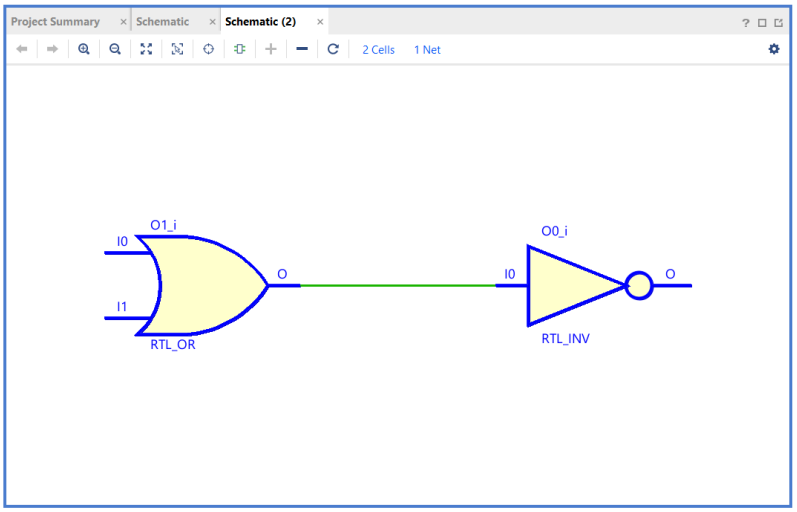
**NOR GATE**

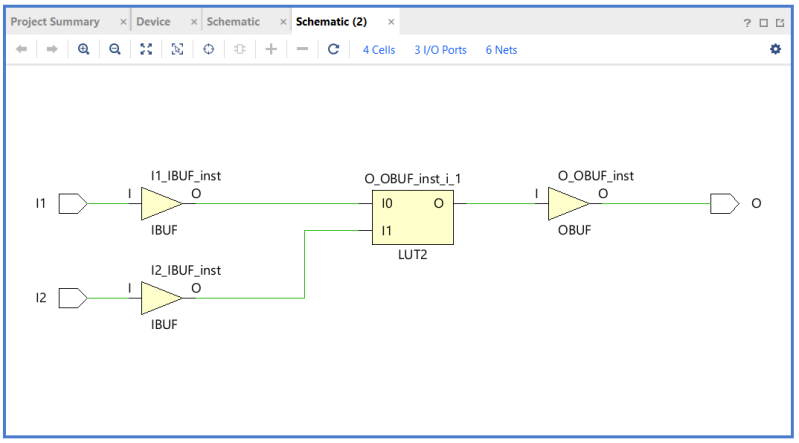
```verilog
`timescale 1ns / 1ps
module NOR(
input I1,I2,
output reg O
    );

always @ (I1 or I2)
  begin
      O =~(I1||I2);
  end
endmodule
```

```verilog
`timescale 1ns / 1ps
module NOR_tb;

reg inp1,inp2;
wire out;

NOR uut(.O(out), .I1(inp1), .I2(inp2));

initial
begin

inp1=0; inp2=0;
#10

inp1=1; inp2=0;
#10

inp1=1; inp2=1;
#10

inp1=0; inp2=1;
#10
$finish;

end
endmodule
```




Verilog Code, Testbench Code and Simulation Wave



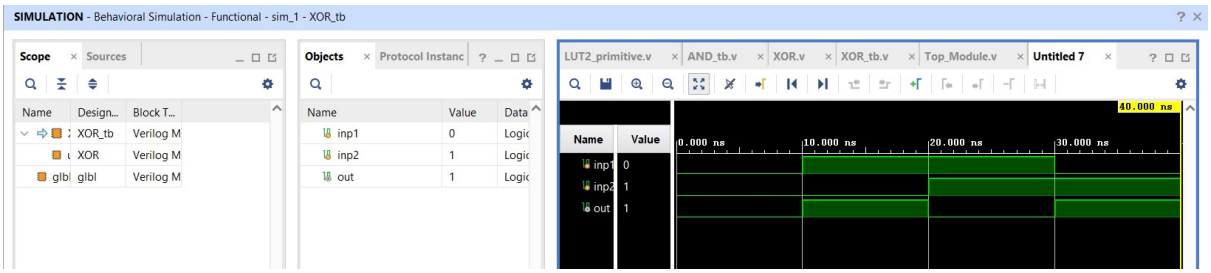




RTL and Technology Schematic

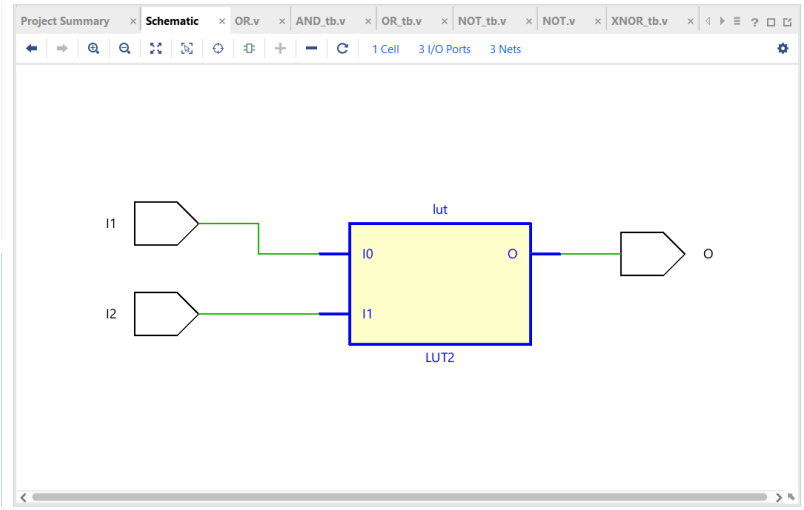
**XOR**

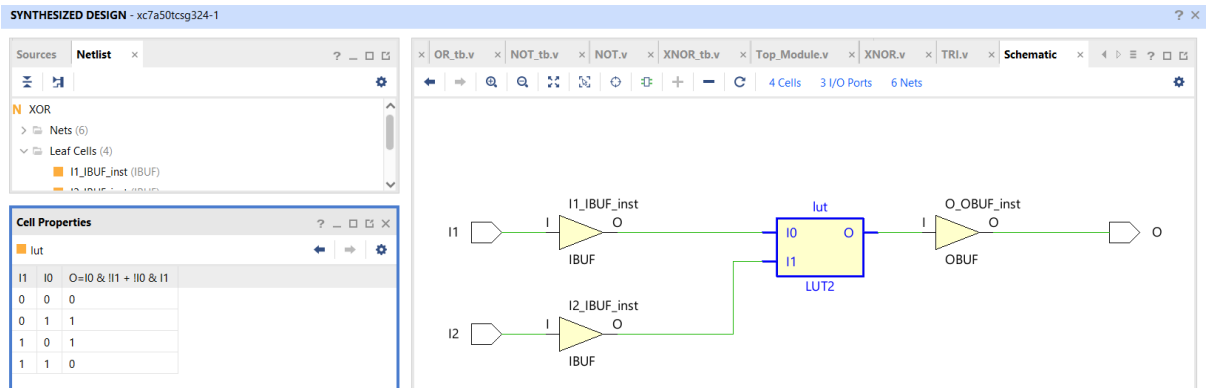
```
`timescale 1ns / 1ps
module XOR(
input I1,I2,
output O);

LUT2# (.INIT(4'b0110))lut(.O(O),.I0(I1),.I1(I2));



endmodule
```

```
`timescale 1ns / 1ps
//////////////////////
module XOR_tb;

reg inp1,inp2;
wire out;

XOR uut(.O(out), .I1(inp1), .I2(inp2));

initial
begin
inp1=0; inp2=0;
#10

inp1=1; inp2=0;
#10

inp1=1; inp2=1;
#10

inp1=0; inp2=1;
#10

$finish;
end
endmodule
```




Verilog Code, Testbench Code and Simulation Wave



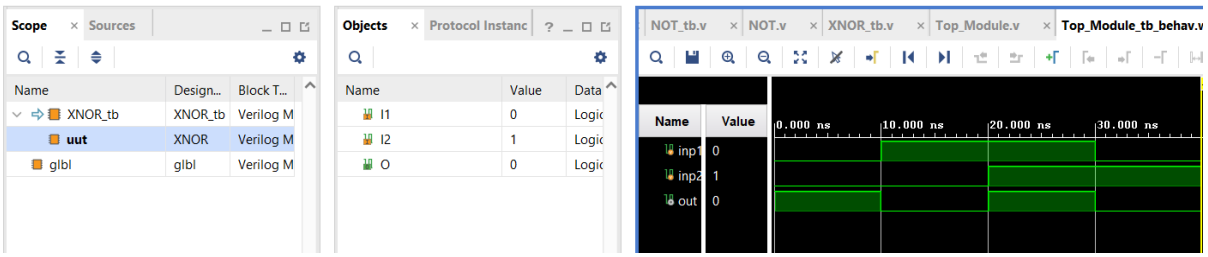




RTL and Technology Schematic

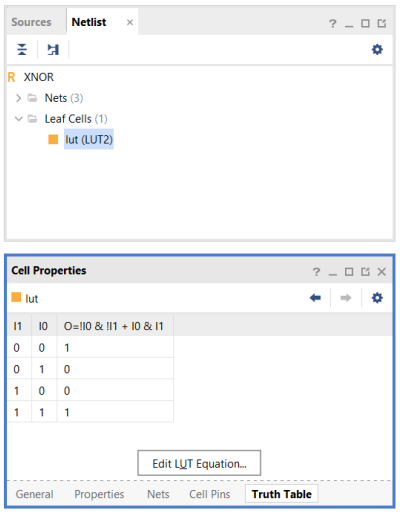
**XNOR GATE**

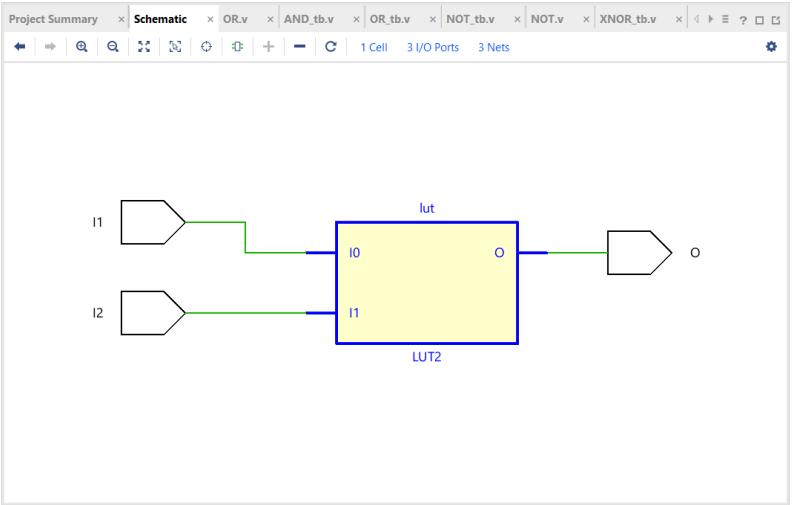
```
`timescale 1ns / 1ps
module XNOR(
input I1,I2,
output O);

LUT2# (.INIT(4'b1001))lut(.O(O),.I0(I1),.I1(I2));
endmodule
```

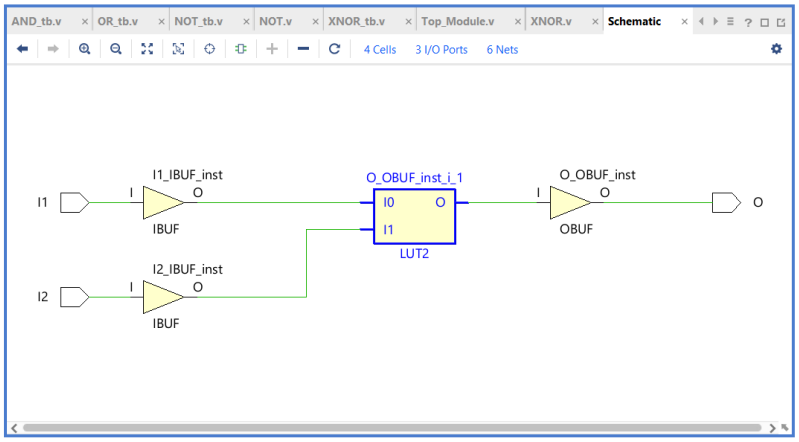
```
`timescale 1ns / 1ps
module XNOR_tb;

reg inp1,inp2;
wire out;

XNOR uut(.O(out), .I1(inp1), .I2(inp2));

initial
begin
inp1=0; inp2=0;
#10

inp1=1; inp2=0;
#10

inp1=1; inp2=1;
#10

inp1=0; inp2=1;
#10

$finish;
end
endmodule
```




Verilog Code, Testbench Code and Simulation Wave



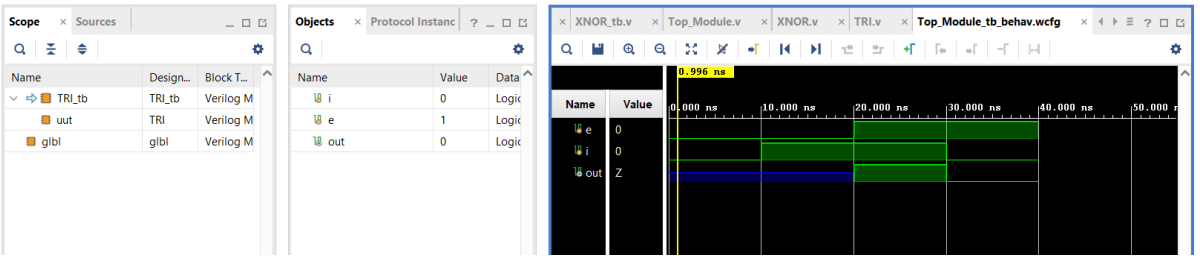




RTL and Technology Schematic

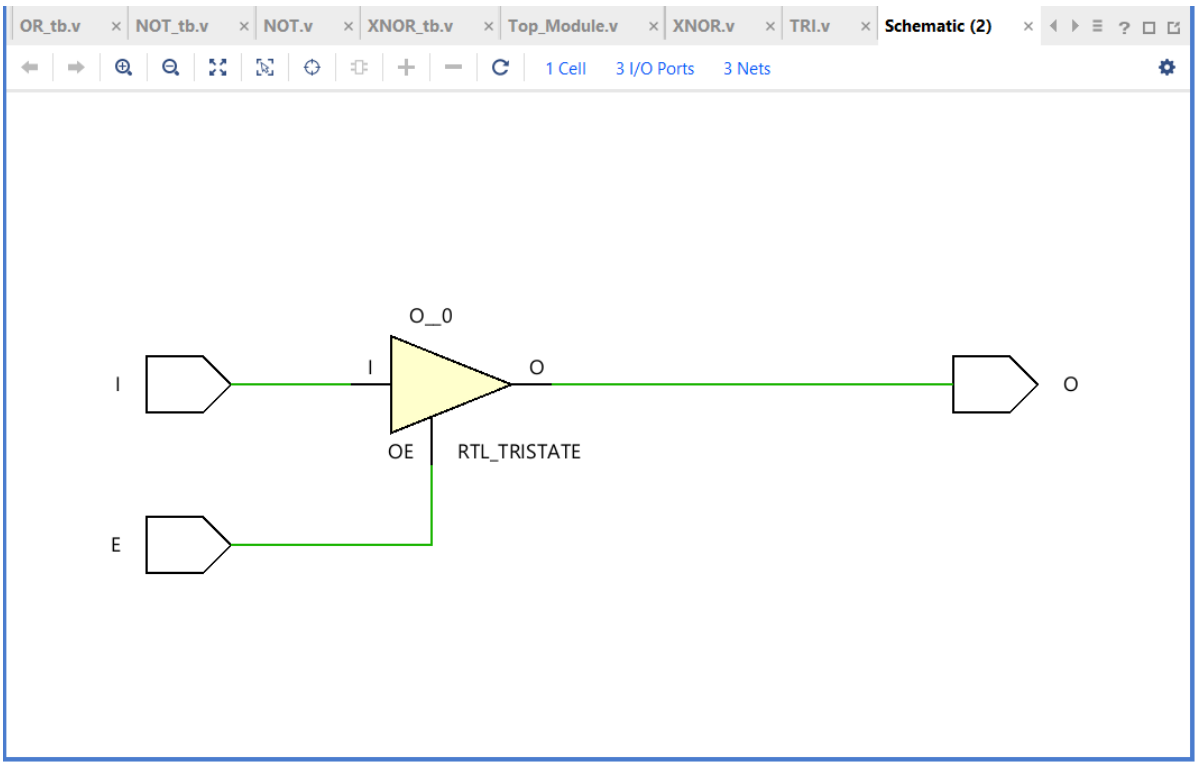
**TRI GATE**

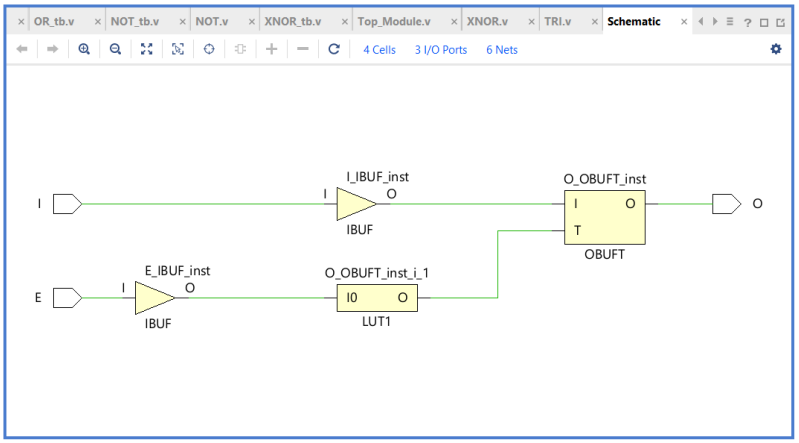
```
`timescale 1ns / 1ps
module TRI(
input I,E,
output O
    );

assign O = E ? I :1'bZ;

endmodule
```

```
`timescale 1ns / 1ps
module TRI_tb;

reg i,e;
wire out;

TRI uut(.O(out), .I(i), .E(e));

initial
begin
i=0; e=0;
#10

i=1; e=0;
#10

i=1; e=1;
#10

i=0; e=1;
#10

$finish;

end
endmodule
```




Verilog Code, Testbench Code and Simulation Wave



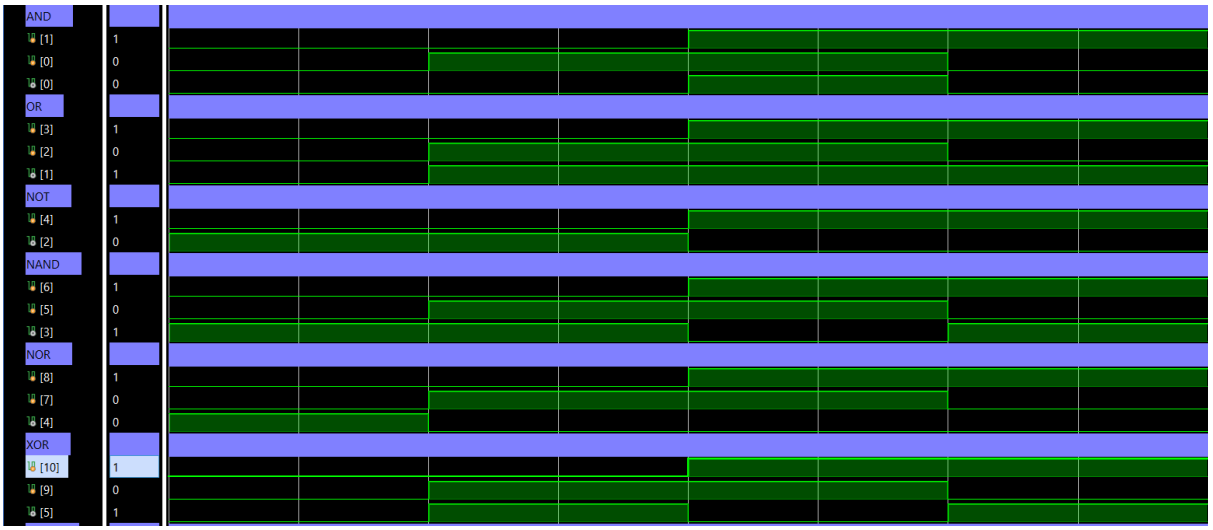



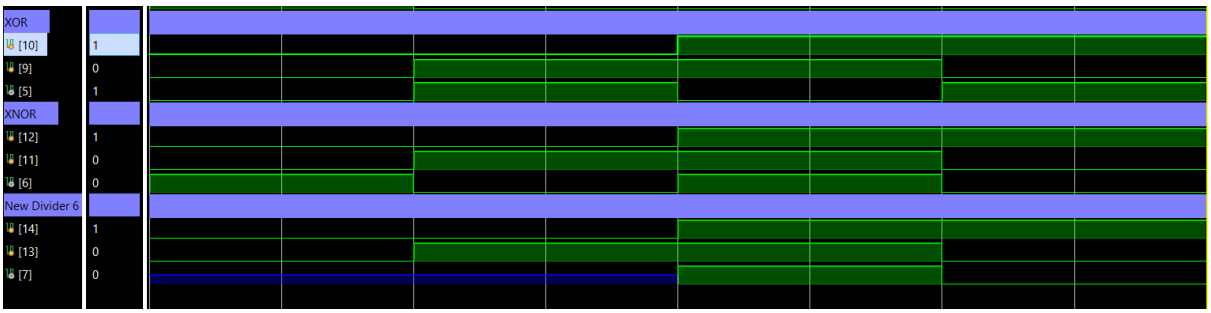

RTL and Technology Schematic

**TOP_MODULE**

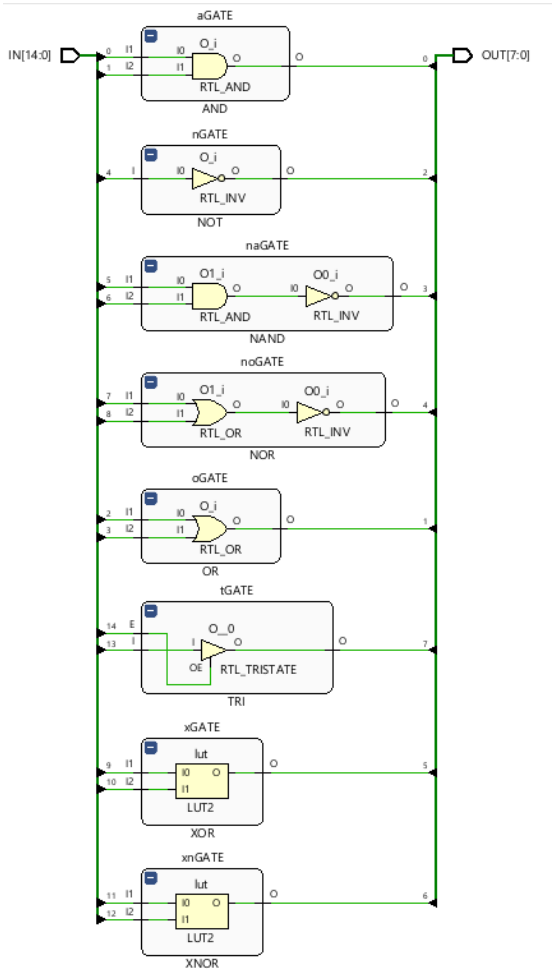
```
module Top_Module(

input [14:0] IN,
output [7:0] OUT

    );

AND aGATE(.O(OUT[0]), .I1(IN[0]), .I2(IN[1]));

OR oGATE(.O(OUT[1]), .I1(IN[2]), .I2(IN[3]));

NOT nGATE(.O(OUT[2]), .I(IN[4]));

NAND naGATE(.O(OUT[3]), .I1(IN[5]), .I2(IN[6]));

NOR noGATE(.O(OUT[4]), .I1(IN[7]), .I2(IN[8]));

XOR xGATE(.O(OUT[5]), .I1(IN[9]), .I2(IN[10]));

XNOR xnGATE(.O(OUT[6]), .I1(IN[11]), .I2(IN[12]));

TRI tGATE(.O(OUT[7]), .I(IN[13]), .E(IN[14]));

endmodule
```

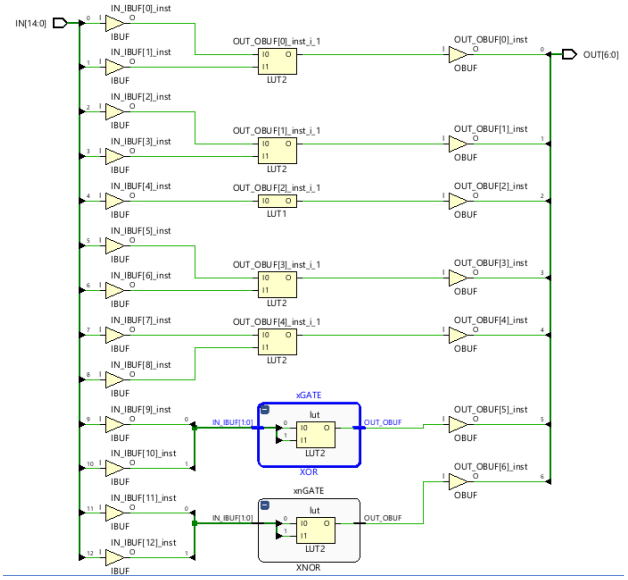
```
module Top_Module_tb;
reg [14:0] inp;
wire [7:0] out;

Top_Module uut(.OUT(out), .IN(inp));

initial
begin

inp=16'b0000000000000000;
#10

inp=16'b010101010100101;
#10

inp=16'b1111111111111111;
#10

inp=16'b101010101011010;
#10
$finish;
end
```






Verilog Code, Testbench Code and Simulation Wave

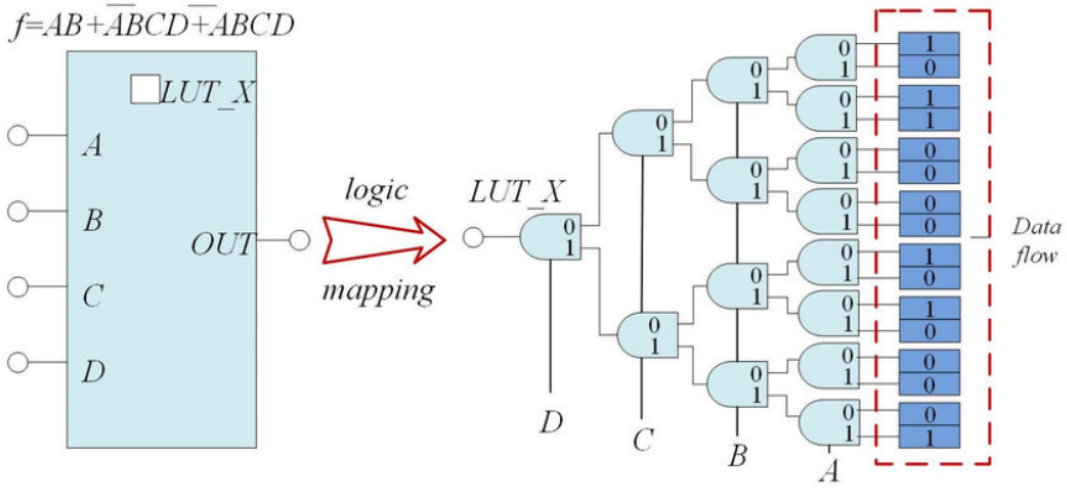



RTL and Technology Schematic

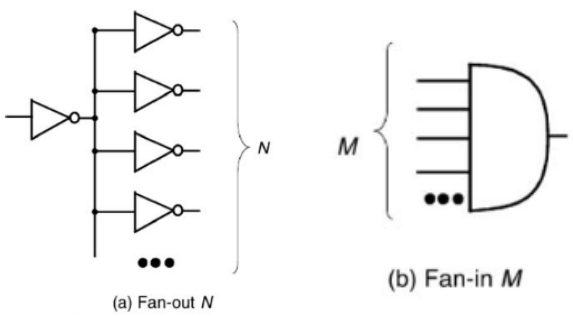
**LOOK UP TABLE**

A Look-Up Table is a configurable memory element in an FPGA that stores output values for all possible combinations of input signals. LUTs are implemented as arrays of memory cells, where each cell corresponds to a unique input combination. These cells store precomputed output values, enabling rapid signal processing and logic implementation. When an FPGA is programmed, the LUTs are configured to implement specific logic functions. Input signals are used as addresses to access the corresponding output values stored in the LUT. This process allows FPGAs to perform complex logic operations, making them highly adaptable for various applications. LUTs can implement any combinational logic function, making them essential for tasks like arithmetic operations, data manipulation, and conditional branching.




**FAN-OUT FAN-IN**

The fan-in is defined as the maximum number of inputs that a logic gate can accept. If the number of input exceeds, the output will be undefined or incorrect. It is specified by the manufacturer and is provided in the data sheet. The fan-out is defined as the maximum number of inputs (load) that can be connected to the output of a gate without degrading the normal operation. Fan Out is calculated from the amount of current available in the output of a gate and the amount of current needed in each input of the connecting gate. It is specified by the manufacturer and is provided in the data sheet. Exceeding the specified maximum load may cause a malfunction because the circuit will not be able to supply the demanded power.



**SETUP and HOLD TIME DELAY**

As a signal travels down a wire, it can change from a 0->1 or 1->0. An input to a Flip-Flop needs to be stable (not changing) in order for an FPGA design to work properly. The input must be stable for some small amount of time prior to being sampled by the clock. This amount of time is called setup time. Setup time is the amount of time required for the input to a Flip-Flop to be stable before a clock edge. Hold time is similar to setup time, but it deals with events after a clock edge occurs. Hold time is the minimum amount of time required for the input to a Flip-Flop to be stable after a clock edge.