# DIGITAL SYSTEM DESIGN APPLICATION – EHB 436E

**Experiment I**

**Yiğit Bektaş GÜRSOY**

**040180063**

**Class Lecturer: Sıddıka Berna Örs Yalçın**

**Class Assistant:**
**Serdar Duran**
**Yasin Fırat Kula**
**Mehmet Onur Demirtürk**

## 1. AND GATE

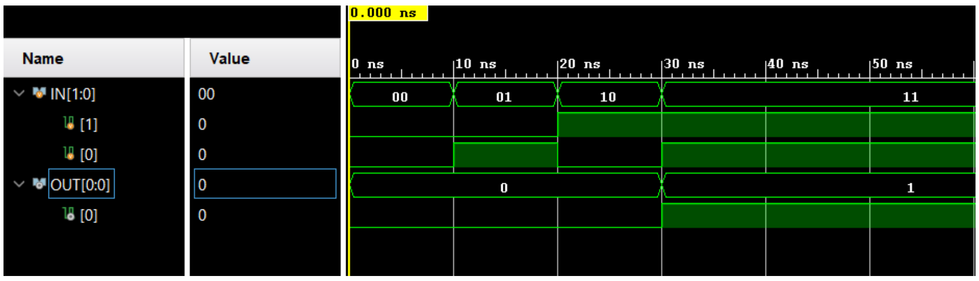
- Verilog code,testbench code and behavioral simulation wave screenshots.

### VERILOG CODE

```
/* AND GATE */
module AND_gate (
    input  I1,
    input  I2,
    output  O
);
    assign O = I1 & I2;
endmodule
```

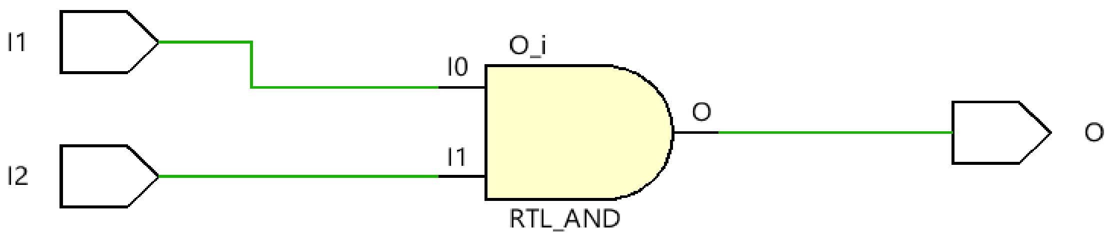
```
module Top_Module_tb();
    reg [1:0]IN;
    wire [0:0]OUT;
    Top_Module DUT(.IN(IN),
                   .OUT(OUT)
                   );
    initial
        //AND GATE
        begin
            IN[0]=0;
            IN[1]=0;
        #10
            IN[0]=1;
            IN[1]=0;
        #10
            IN[0]=0;
            IN[1]=1;
        #10
            IN[0]=1;
            IN[1]=1;
        end
endmodule
```

### BEHAVIORAL SIMULATION

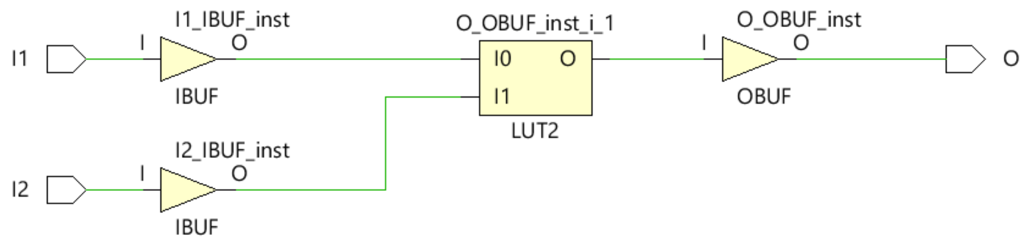


As you can see, it outputs 1 only when two inputs are "1", and outputs 0 when one of the 2 inputs is 0. As a result of this simulation, we can see that we have designed and door correctly.

- **RTL Schematic**



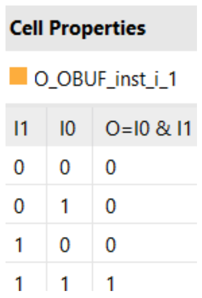
- **Technology Schematic**



➢ The accuracy of the above technology and RTL schematics can be checked with the following truth table. When checked, it seems that they both give the same output and the output values of our schematics overlap with each other.

- **Synthesis Report**

➢ Truth Table of the LUT

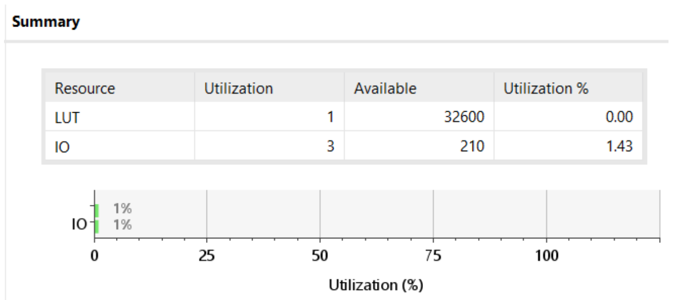


**Cell Properties**

🟧 O_OBUF_inst_i_1

| I1 | I0 | O=I0 & I1 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

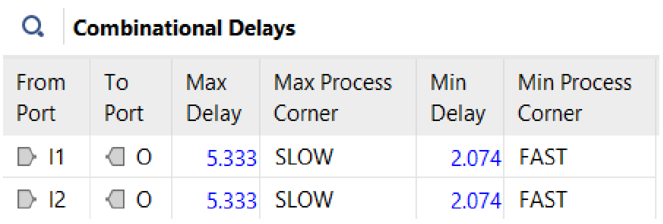
➢ Usage of the FPGA resources (utilization summary)



**Summary**

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

➢ Combinational path delays



**Combinational Delays**

| 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 |

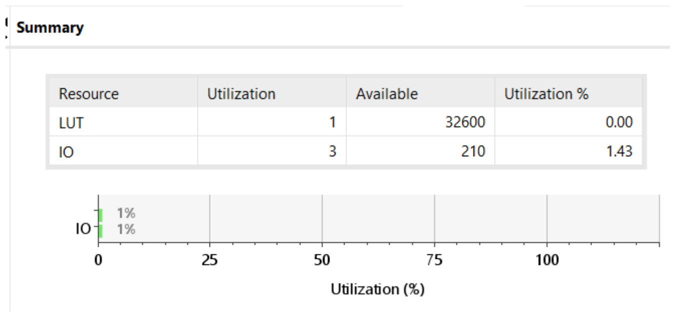
➢ Maximum combinational path delay is 5.3333.

- Post-synthesis simulation model

```
`timescale 1 ps / 1 ps
`define XIL_TIMING

(* NotValidForBitStream *)
module AND_gate
   (I1,
    I2,
    O);
  input I1;
  input I2;
  output O;

  wire I1;
  wire I1_IBUF;
  wire I2;
  wire I2_IBUF;
  wire O;
  wire O_OBUF;

initial begin

$sdf_annotate("Top_Module_tb_time_synth.sdf",,,,
"tool_control");
end
  IBUF I1_IBUF_inst
       (.I(I1),
        .O(I1_IBUF));
  IBUF I2_IBUF_inst
       (.I(I2),
        .O(I2_IBUF));
  OBUF O_OBUF_inst
       (.I(O_OBUF),
        .O(O));
  LUT2 #(
    .INIT(4'h8))
    O_OBUF_inst_i_1
       (.I0(I1_IBUF),
        .I1(I2_IBUF),
        .O(O_OBUF));
endmodule
`ifndef GLBL
`define GLBL
`timescale  1 ps / 1 ps

module glbl ();

    parameter ROC_WIDTH = 100000;
    parameter TOC_WIDTH = 0;

//--------   STARTUP Globals --------------
    wire GSR;
    wire GTS;
    wire GWE;
    wire PRLD;
    tri1 p_up_tmp;
    tri (weak1, strong0) PLL_LOCKG = p_up_tmp;

    wire PROGB_GLBL;
    wire CCLKO_GLBL;
    wire FCSBO_GLBL;
    wire [3:0] DO_GLBL;
    wire [3:0] DI_GLBL;

    reg GSR_int;
    reg GTS_int;
    reg PRLD_int;

//--------   JTAG Globals --------------
    wire JTAG_TDO_GLBL;
    wire JTAG_TCK_GLBL;
    wire JTAG_TDI_GLBL;
    wire JTAG_TMS_GLBL;
    wire JTAG_TRST_GLBL;

    reg JTAG_CAPTURE_GLBL;
    reg JTAG_RESET_GLBL;
    reg JTAG_SHIFT_GLBL;
    reg JTAG_UPDATE_GLBL;
    reg JTAG_RUNTEST_GLBL;

    reg JTAG_SEL1_GLBL = 0;
    reg JTAG_SEL2_GLBL = 0 ;
    reg JTAG_SEL3_GLBL = 0;
    reg JTAG_SEL4_GLBL = 0;

    reg JTAG_USER_TDO1_GLBL = 1'bz;
    reg JTAG_USER_TDO2_GLBL = 1'bz;
    reg JTAG_USER_TDO3_GLBL = 1'bz;
    reg JTAG_USER_TDO4_GLBL = 1'bz;

    assign (strong1, weak0) GSR = GSR_int;
    assign (strong1, weak0) GTS = GTS_int;
    assign (weak1, weak0) PRLD = PRLD_int;

    initial begin
    GSR_int = 1'b1;
    PRLD_int = 1'b1;
    #(ROC_WIDTH)
    GSR_int = 1'b0;
    PRLD_int = 1'b0;
    end

    initial begin
    GTS_int = 1'b1;
    #(TOC_WIDTH)
    GTS_int = 1'b0;
    end

endmodule
`endif
```

- **Implementation Report:**

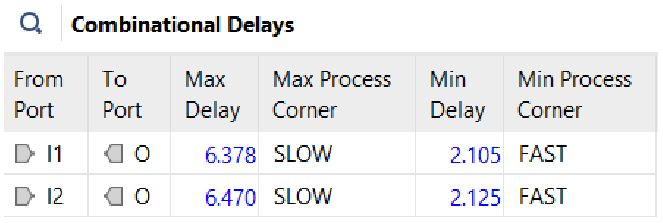
  - Usage of the FPGA resources (utilization summary)



**Summary**

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

  - Combinational path delays



**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 |

  - Maximum combinational path delay is 6470ns.

  - Implementation result was higher than synthesis result. The reason for this is that the circuit that we get output from as a result of the implementation is our circuit that makes the mapping on the FPGA. When we synthesize, the components are not processed on the FPGA. In this direction, the implementation results give more realistic results in delay times.

## 2. OTHER GATES

- Verilog code,testbench code and behavioral simulation wave screenshots.

### VERILOG CODES

```
/* OR GATE */
module OR_gate (
    input  I1,
    input  I2,
    output O
);
    assign O = I1 | I2;
endmodule

/* NOT GATE */
module NOT_gate (
    input I,
    output O
);
    assign O = ~I;
endmodule


module NAND_gate (
    input I1,
    input I2,
    output reg  O
);
    always@*
    begin
        assign O = ~( I1 &
I2);
    end
endmodule

/* NOR gate */
module NOR_gate (
    input I1,
    input I2,
    output reg O
);
    always@*
    begin
        assign O = ~( I1 |
I2);
    end
endmodule
```

```
module EXOR_gate(
    input I1,
    input I2,
    output O
    );

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

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

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

module TRI(
    input I,
    input E,
    output O
    );
    assign O = ( E == 1'b1) ?
I : 1'bZ;
endmodule
```

## TOP MODULE

```
module Top_Module(
    input [14:0]IN,
    output [7:0]OUT
    );

    AND_gate U1(
    .I1(IN[0]),
    .I2(IN[1]),
    .O(OUT[0]));

    OR_gate U2(
    .I1(IN[2]),
    .I2(IN[3]),
    .O(OUT[1]));

    NOT_gate U3(
    .I(IN[4]),
    .O(OUT[2]));

    NAND_gate U4(
    .I1(IN[5]),
    .I2(IN[6]),
    .O(OUT[3]));
```

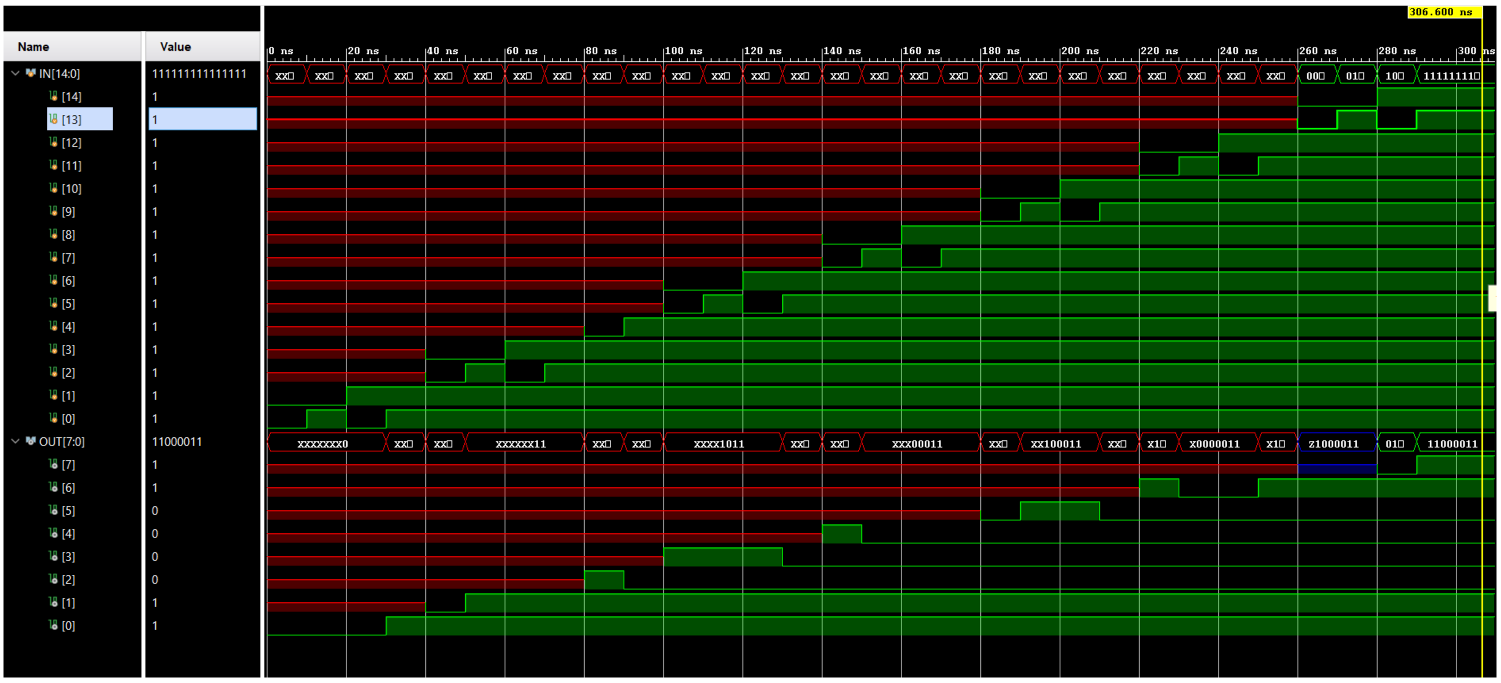
```
  NOR_gate U5(
    .I1(IN[7]),
    .I2(IN[8]),
    .O(OUT[4]));

    EXOR_gate U6(
    .I1(IN[9]),
    .I2(IN[10]),
    .O(OUT[5]));

    EXNOR_gate U7(
    .I1(IN[11]),
    .I2(IN[12]),
    .O(OUT[6]));

    TRI U8(
    .I(IN[13]),
    .E(IN[14]),
    .O(OUT[7]));

    endmodule
```

## TOP MODULE TEST BENCH

```
timescale 1ns / 1ps
module Top_Module_tb();
    reg [14:0]IN;
    wire [7:0]OUT;
    Top_Module DUT(.IN(IN),
                .OUT(OUT)
                );
    initial
        //AND GATE
        begin
            IN[0]=0;IN[1]=0;
        #10
            IN[0]=1;IN[1]=0;
        #10
            IN[0]=0;IN[1]=1;
        #10
            IN[0]=1;IN[1]=1;
        //OR GATE
        #10
            IN[2]=0;IN[3]=0;
        #10
            IN[2]=1;IN[3]=0;
        #10
            IN[2]=0;IN[3]=1;
        #10
            IN[2]=1;IN[3]=1;
        //NOT GATE
        #10
            IN[4]=0;
        #10
            IN[4]=1;
        //NAND GATE
        #10
            IN[5]=0;IN[6]=0;
        #10
            IN[5]=1;IN[6]=0;
        #10
            IN[5]=0;IN[6]=1;
        #10
            IN[5]=1;IN[6]=1;
```

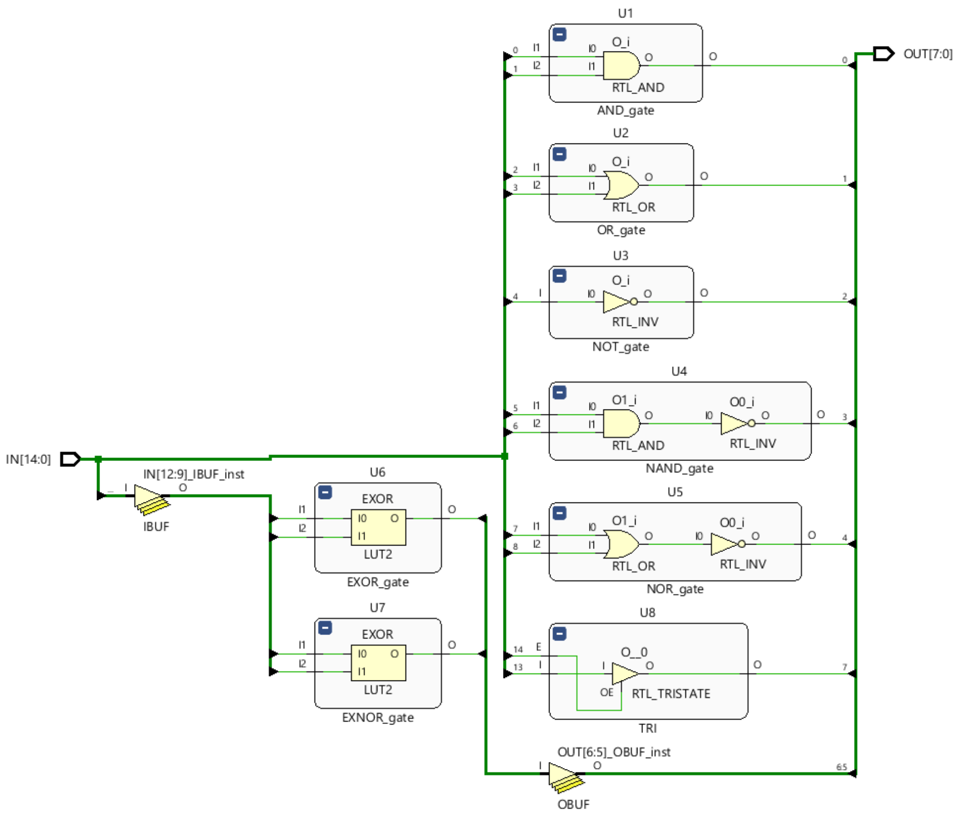
```
//NOR GATE
        #10
            IN[7]=0;IN[8]=0;
        #10
            IN[7]=1;IN[8]=0;
        #10
            IN[7]=0;IN[8]=1;
        #10
            IN[7]=1;IN[8]=1;
        //EXOR GATE
        #10
            IN[9]=0;IN[10]=0;
        #10
            IN[9]=1;IN[10]=0;
        #10
            IN[9]=0;IN[10]=1;
        #10
            IN[9]=1;IN[10]=1;

        //EXNOR GATE
        #10
            IN[11]=0;IN[12]=0;
        #10
            IN[11]=1;IN[12]=0;
        #10
            IN[11]=0;IN[12]=1;
        #10
            IN[11]=1;IN[12]=1;
        //TRI GATE
        #10
            IN[13]=0;IN[14]=0;
        #10
            IN[13]=1;IN[14]=0;
        #10
            IN[13]=0;IN[14]=1;
        #10
            IN[13]=1;IN[14]=1;
        end
endmodule
```

# BEHAVIORAL SIMULATION

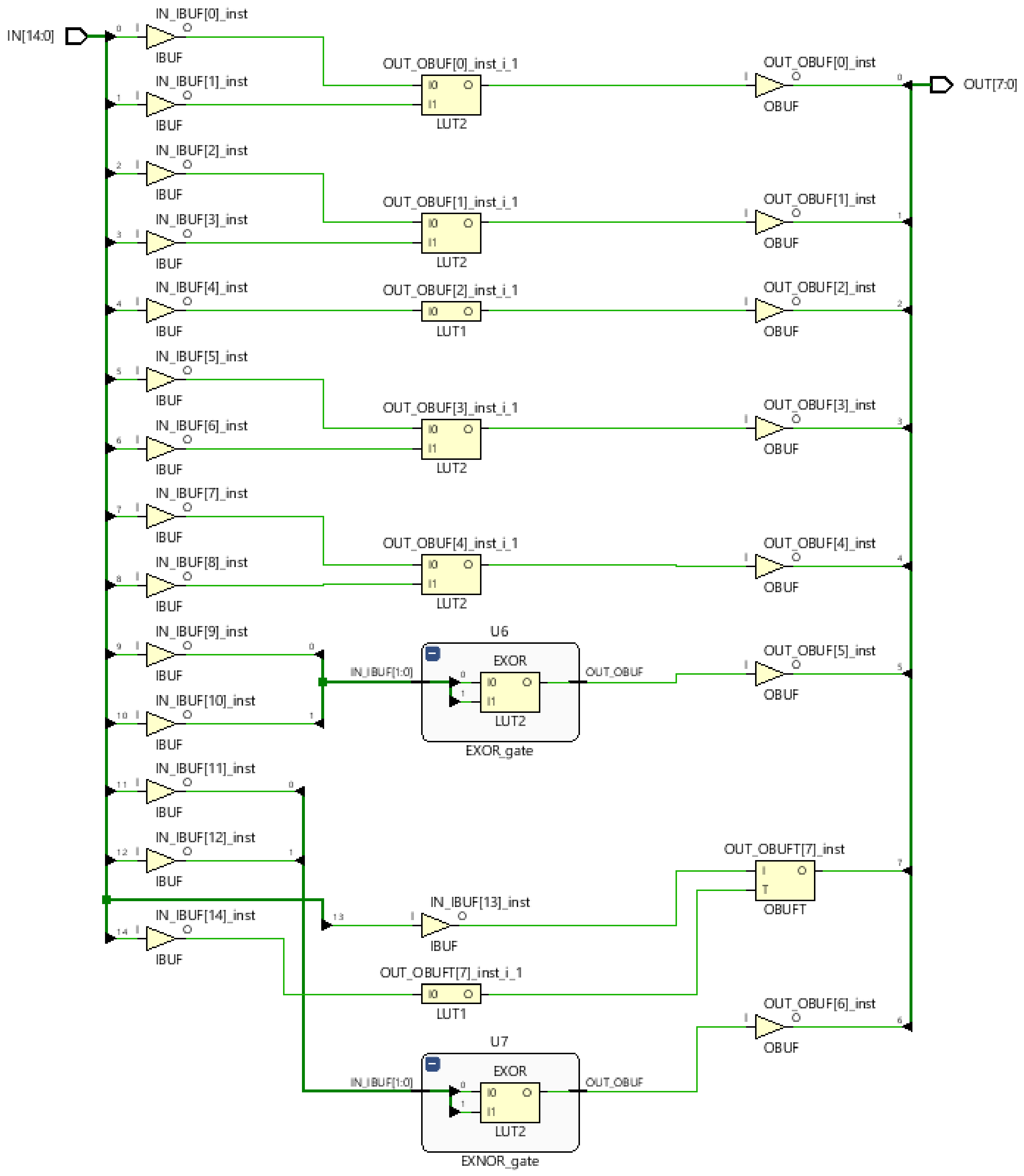


➢ As seen in the simulation results, all of our gates give the correct result. All doors have been tested with 10 second intervals.

- ## RTL Schmematic



- **Technology schematic**

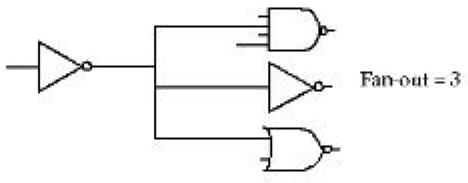


- **Look-up Table (LUT):**

  - Lookup table is a table that generates data based on input values. It is frequently used in simulation programs. It is an array that holds values that need to be calculated. The lookup table can be filled in manually when the program is written, or the simulation program can fill it with values while calculating the lookup table. When values are needed later, the program can save CPU resources by searching for them.

- **Fan-in and fan-out:**

  - Fan-in: Fan input is a term used to describe the maximum number of inputs to a logic gate (logic circuit). For example, let's have a circuit like the image below. Suppose we have a 3-input NAND gate. Based on what has been said, the fan-in value is 3.

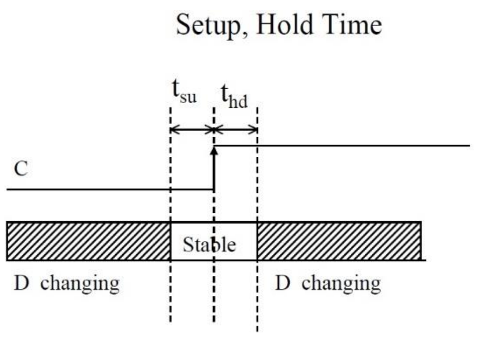


  - Fan-out: The maximum number of loads that a logic gate can drive from the same integrated family is called , Fan Out, the output capacity. The maximum fan out of an output measures the load-driving capacity.

- **Setup time and hold time:**

  - Setup time:  Setup time is defined as the minimum time before the active edge of the clock that must be constant for data to be retained correctly. For a flip-flop or any sequential item to hold data reliably, it needs time to pass before the clock edge arrives for the data to remain constant. This time is known as the setup time.

  - Hold time: Retention time is defined as the minimum time after the active edge of the clock during which data must be stable. The data needs some time to remain stable after the clock edge comes to capture data losslessly. This is the holding time.


Setup, Hold Time
$t_{su}$
$t_{hd}$
C
Stable
D changing
D changing