DIGITAL SYSTEM DESIGN APPLICATION

EHB436E CRN: 11280

Salih Ömer Ongün
040220780

Experiment 1

Salih Ömer Ongün
040220780

AND GATE

Design Sources

AND gate design sources

```
`timescale 1ns / 1ps

module andgate
(
    input l1,
    input l2,
    output o
);

    assign o = l1 & l2;

endmodule
```

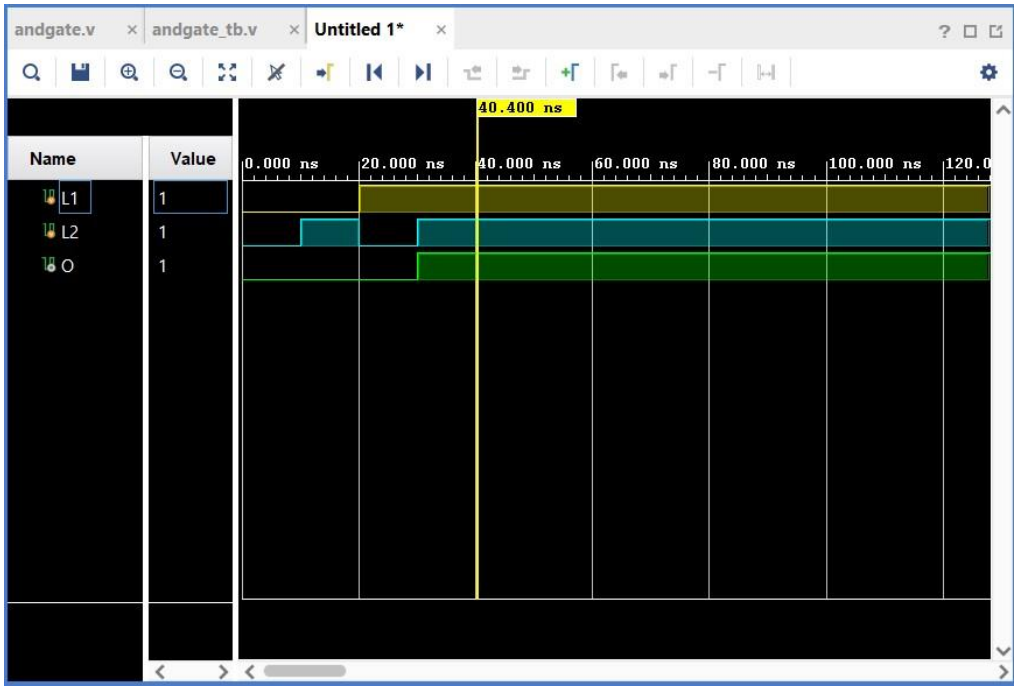
Simulation Sources

AND gate simulation sources

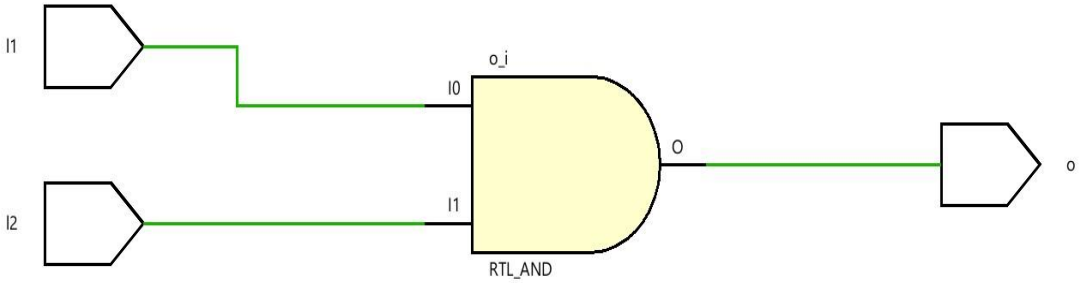
```
`timescale 1ns / 1ps


module andgate_tb();

    reg L1=0;
    reg L2=0;
    wire O;

    andgate uut(

        .l1(L1),
        .l2(L2),
        .o(O)
    );

    initial
    begin

        L1=0; L2=0;
        #10;
        L1=0; L2=1;
        #10;
        L1=1; L2=0;
        #10;
        L1=1; L2=1;
        #10;
    end

endmodule
```

Simulation Wave

Behavioral simulation wave screenshot

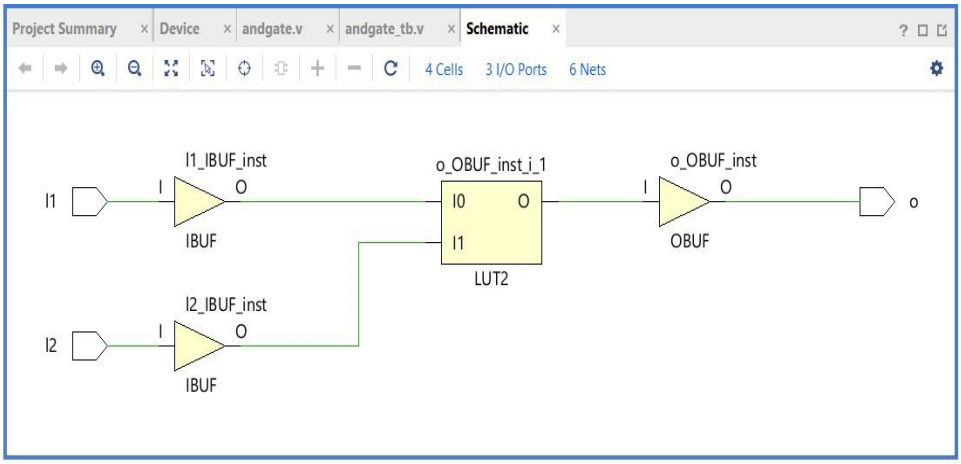


RTL Schematic



Technology Schematic

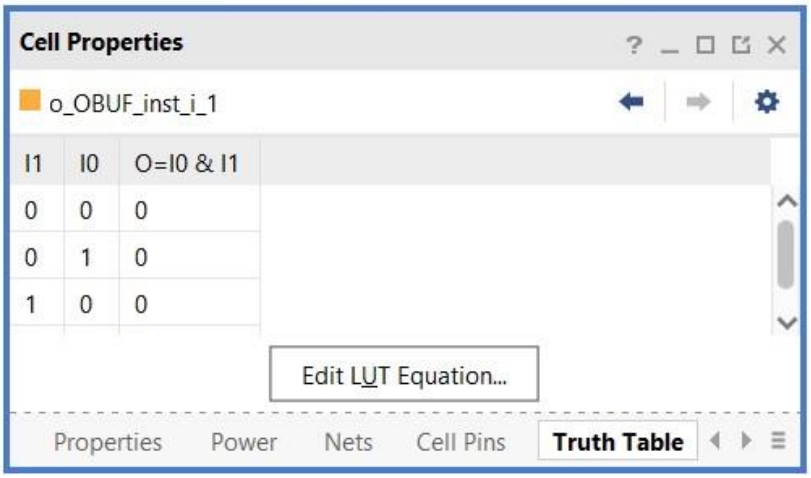
Technology Schematic of AND gate



There are not any gates in the structure of Fpga. Fpga implements gate functions with Look-Up Tables. If we look at the truth table of LUT2, we can see that LUT2 performs the gate function.

Synthesis Report

Truth Table of LUT2



Post-Synthesis Utilization Summary

Utilization **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	1	32600	0.01
IO	3	210	1.43

The delay of this path from “l1” to “o” is 6.738.

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Logic %	Net %	Requirement	Source Clock
Unconstrained Paths (1)													
(none) (1)													
Path 2	∞	3	2	1	l1	o	6.738	5.139	1.599	76.3	23.7	∞	input port clock

The delay of this path from “l2” to “o” is 6.732.

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Logic %	Net %	Requirement	Source Clock
Unconstrained Paths (1)													
(none) (1)													
Path 2	∞	3	2	1	l2	o	6.732	5.132	1.599	76.2	23.8	∞	input port clock

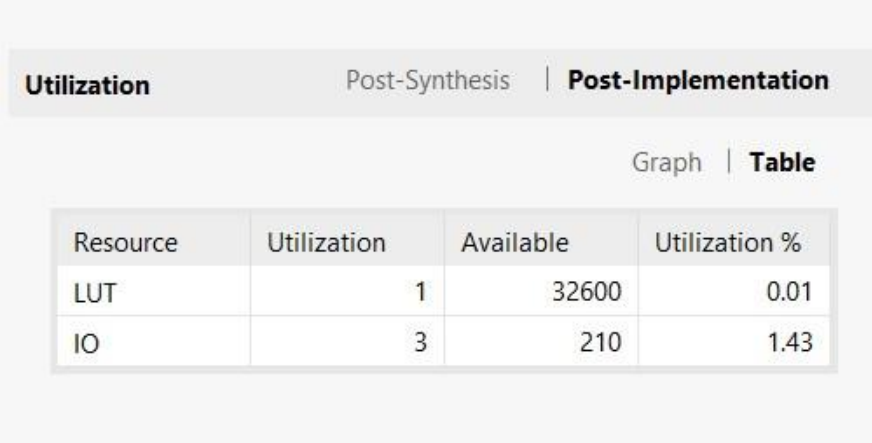
Maximum combinational path delay is 6.738.

Post-Synthesis Simulation File

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

(* NotValidForBitStream *)
module andgate
   (l1,
    l2,
    o);
  input l1;
  input l2;
  output o;

  wire l1;
  wire l1_IBUF;
  wire l2;
  wire l2_IBUF;
  wire o;
  wire o_OBUF;

initial begin

$sdf_annotate("andgate_tb_time_synth.sdf",,,,"tool_control");
end
  IBUF #(
    .CCIO_EN("TRUE"))
    l1_IBUF_inst
       (.I(l1),
        .O(l1_IBUF));
  IBUF #(
    .CCIO_EN("TRUE"))
    l2_IBUF_inst
       (.I(l2),
        .O(l2_IBUF));
  OBUF o_OBUF_inst
       (.I(o_OBUF),
        .O(o));
  LUT2 #(
    .INIT(4'h8))
    o_OBUF_inst_i_1
       (.I0(l1_IBUF),
        .I1(l2_IBUF),
        .O(o_OBUF));
endmodule
`ifndef GLBL
`define GLBL

`timescale  1 ps / 1 ps


module glbl ();

    parameter ROC_WIDTH = 100000;
    parameter TOC_WIDTH = 0;
    parameter GRES_WIDTH = 10000;
    parameter GRES_START = 10000;

//--------   STARTUP Globals --------
    wire GSR;
    wire GTS;
    wire GWE;
    wire PRLD;
    wire GRESTORE;
    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;
    reg GRESTORE_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;


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

    initial begin
    GRESTORE_int = 1'b0;
    #(GRES_START);
    GRESTORE_int = 1'b1;
    #(GRES_WIDTH);
    GRESTORE_int = 1'b0;
    end

endmodule
`endif
```

Implementation Report

Post-Implementation Utilization Summary



Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	1	32600	0.01
IO	3	210	1.43

The delay of this path from “l1” to “o” is 8.268.

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Logic %	Net %	Requirement	Source Clock
Unconstrained Paths (1)													
(none) (1)													
Path 2	∞	3	2	1	l1	o	8.268	5.139	3.129	62.2	37.8	∞	input port clock

The delay of this path from “l2” to “o” is 8.371.

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Logic %	Net %	Requirement	Source Clock
Unconstrained Paths (1)													
(none) (1)													
Path 2	∞	3	2	1	l2	o	8.371	5.132	3.238	61.3	38.7	∞	input port clock

Maximum combinational path delay is 8.371 .

Synthesis is a predictive phase, but implementation involves physical design. The delay after implementation provides more reliable results compared to synthesis.

Salih Ömer Ongün
040220780

OTHER GATES

Design Sources

OR gate

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

    assign O = I1 | I2;

endmodule
```

NOT gate

```
module NOT
(
    input I,
    output O
);

    assign O = ~I ;

endmodule
```

NAND gate

```
module NAND
(
    input I1,
    input I2,
    output O
);
    reg O_reg;

    always@(*)
    begin

        O_reg = ~ ( I1 & I2);
    end

    assign O = O_reg;

endmodule
```

NOR gate

```
module NOR
(
    input I1,
    input I2,
    output O
);
    reg O_reg;

    always@(*)
    begin

        O_reg = ~ ( I1 | I2);
    end

    assign O = O_reg;

endmodule
```

EXOR gate

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

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


endmodule
```

EXNOR gate

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

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


endmodule
```

TRI gate

```
module TRI
(
    input I,
    input E,
    output O
);

    assign O = (E == 1'b1) ? I
: 1'bZ;

endmodule
```

Salih Ömer Ongün
040220780

TOP MODULE

Design Sources

Top_Module design source.

```
`timescale 1ns / 1ps

module Top_Module
(
 input [15:0]IN,
 output [7:0]OUT

);

    AND s1
    (
    .I1(IN[0]),
    .I2(IN[1]),
    .O(OUT[0])
    );

    OR s2
    (
    .I1(IN[2]),
    .I2(IN[3]),
    .O(OUT[1])
    );

    NOT s3
    (
    .I(IN[4]),
    .O(OUT[2])
    );
```

```
    NAND s4
    (
    .I1(IN[5]),
    .I2(IN[6]),
    .O(OUT[3])
    );

    NOR s5
    (
    .I1(IN[7]),
    .I2(IN[8]),
    .O(OUT[4])
    );

    EXOR s6
    (
    .I1(IN[9]),
    .I2(IN[10]),
    .O(OUT[5])
    );

    EXNOR s7
    (
    .I1(IN[11]),
    .I2(IN[12]),
    .O(OUT[6])
    );

    TRI s8
    (
    .I(IN[13]),
    .E(IN[14]),
    .O(OUT[7])
    );

endmodule
```

Simulation Sources

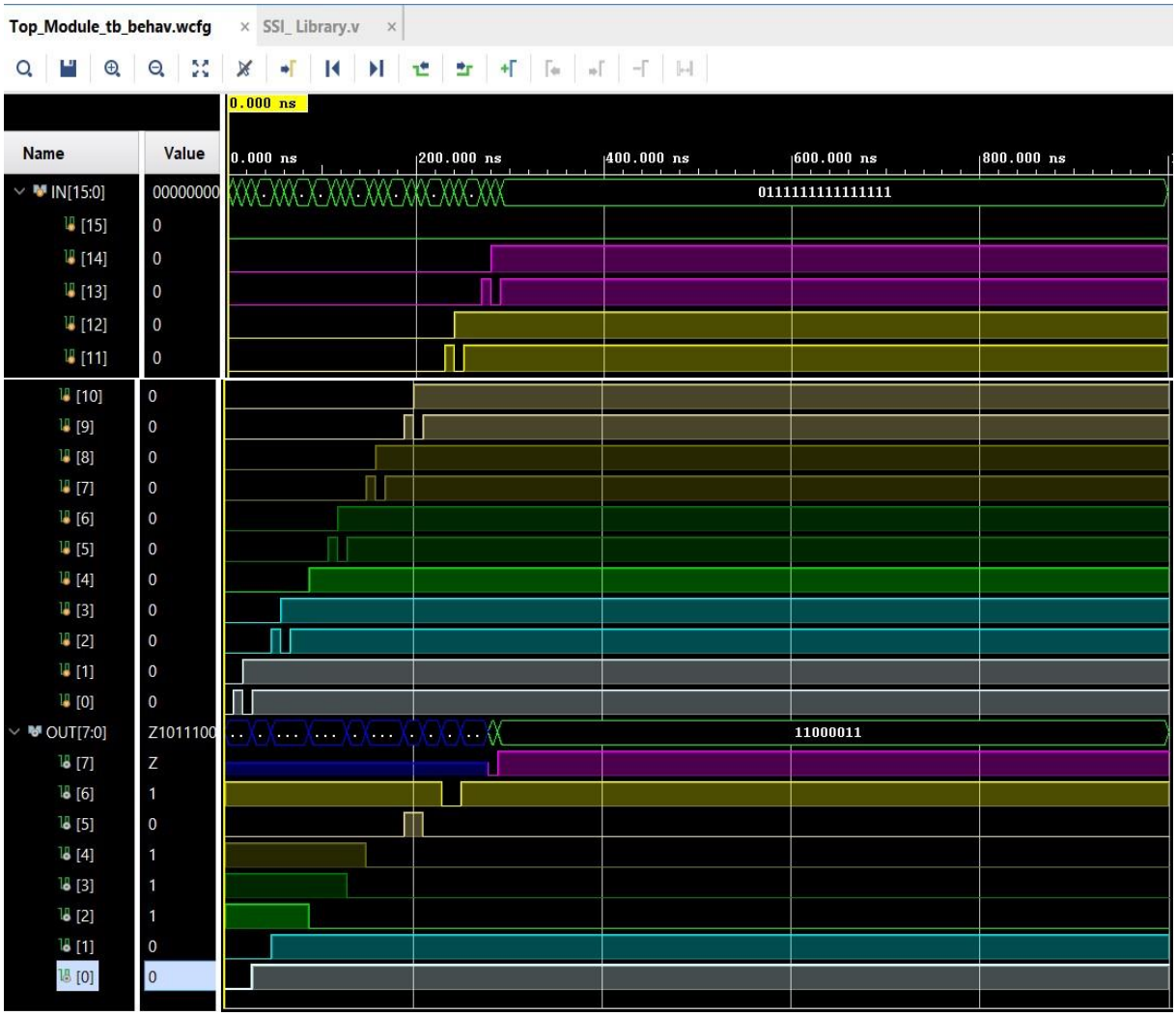
TOP_MODULE simulation sources

```
`timescale 1ns / 1ps

module Top_Module_tb();

    reg [15:0]IN=16'b0;
    wire [7:0]OUT;
    //integer i;
    //integer y;
    Top_Module uut(

        .IN(IN),
        .OUT(OUT)
    );

    initial
    begin

        IN[0]=0; IN[1]=0; //AND
        #10;
        IN[0]=1; IN[1]=0;
        #10;
        IN[0]=0; IN[1]=1;
        #10;
        IN[0]=1; IN[1]=1;
        #10;

        IN[2]=0; IN[3]=0; // OR
        #10;
        IN[2]=1; IN[3]=0;
        #10;
        IN[2]=0; IN[3]=1;
        #10;
        IN[2]=1; IN[3]=1;
        #10;

        IN[4]=0; //NOT
        #10;
        IN[4]=1;
        #10;

        IN[5]=0; IN[6]=0; // NAND
        #10;
        IN[5]=1; IN[6]=0;
        #10;
        IN[5]=0; IN[6]=1;
        #10;
        IN[5]=1; IN[6]=1;
        #10;
```

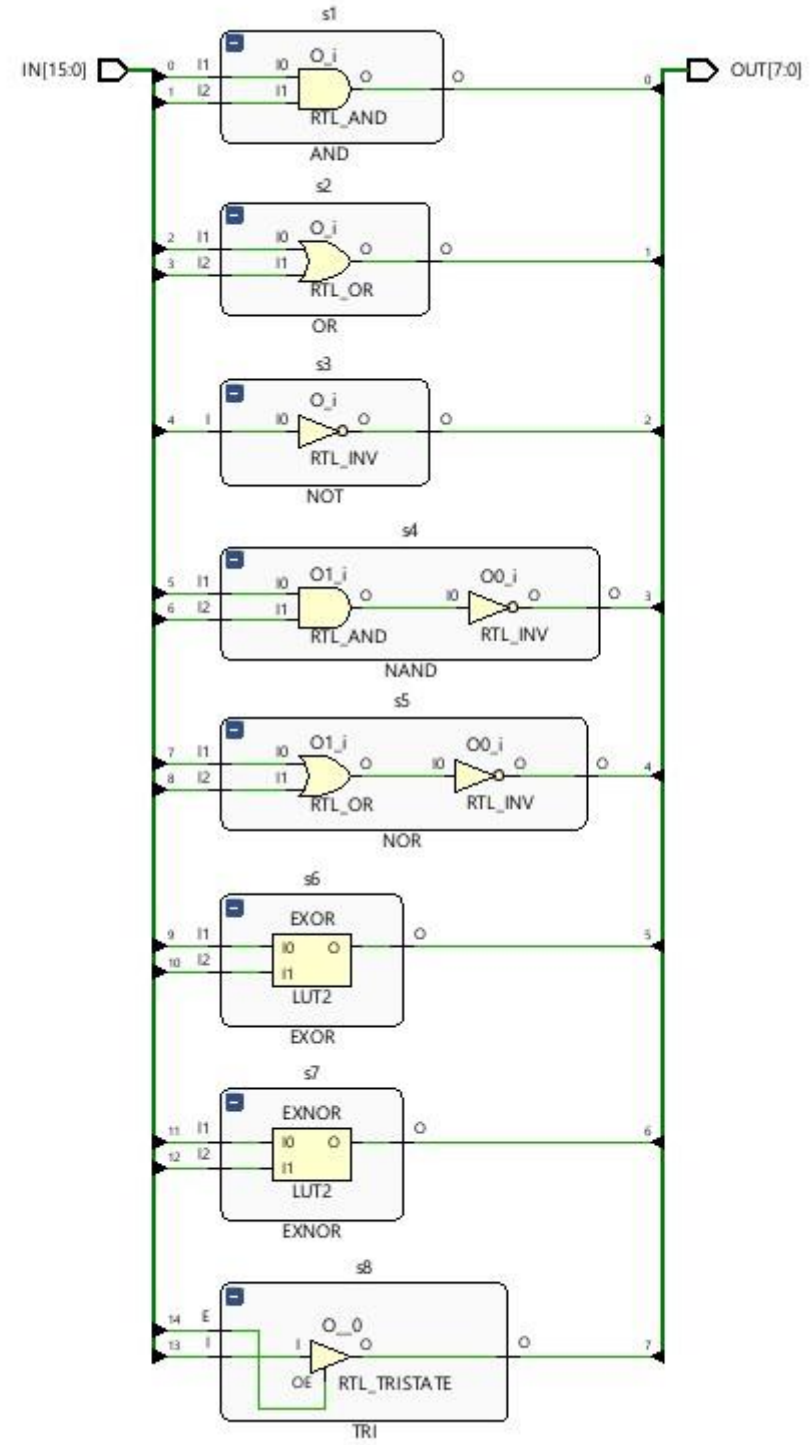
```
        IN[7]=0; IN[8]=0; // NOR
        #10;
        IN[7]=1; IN[8]=0;
        #10;
        IN[7]=0; IN[8]=1;
        #10;
        IN[7]=1; IN[8]=1;
        #10;

        IN[9]=0; IN[10]=0; //EXOR
        #10;
        IN[9]=1; IN[10]=0;
        #10;
        IN[9]=0; IN[10]=1;
        #10;
        IN[9]=1; IN[10]=1;
        #10;

        IN[11]=0; IN[12]=0; //EXNOR
        #10;
        IN[11]=1; IN[12]=0;
        #10;
        IN[11]=0; IN[12]=1;
        #10;
        IN[11]=1; IN[12]=1;
        #10;

        IN[13]=0; IN[14]=0; //TRI
        #10;
        IN[13]=1; IN[14]=0;
        #10;
        IN[13]=0; IN[14]=1;
        #10;
        IN[13]=1; IN[14]=1;
        #10;

    end
endmodule
```

Simulation Wave

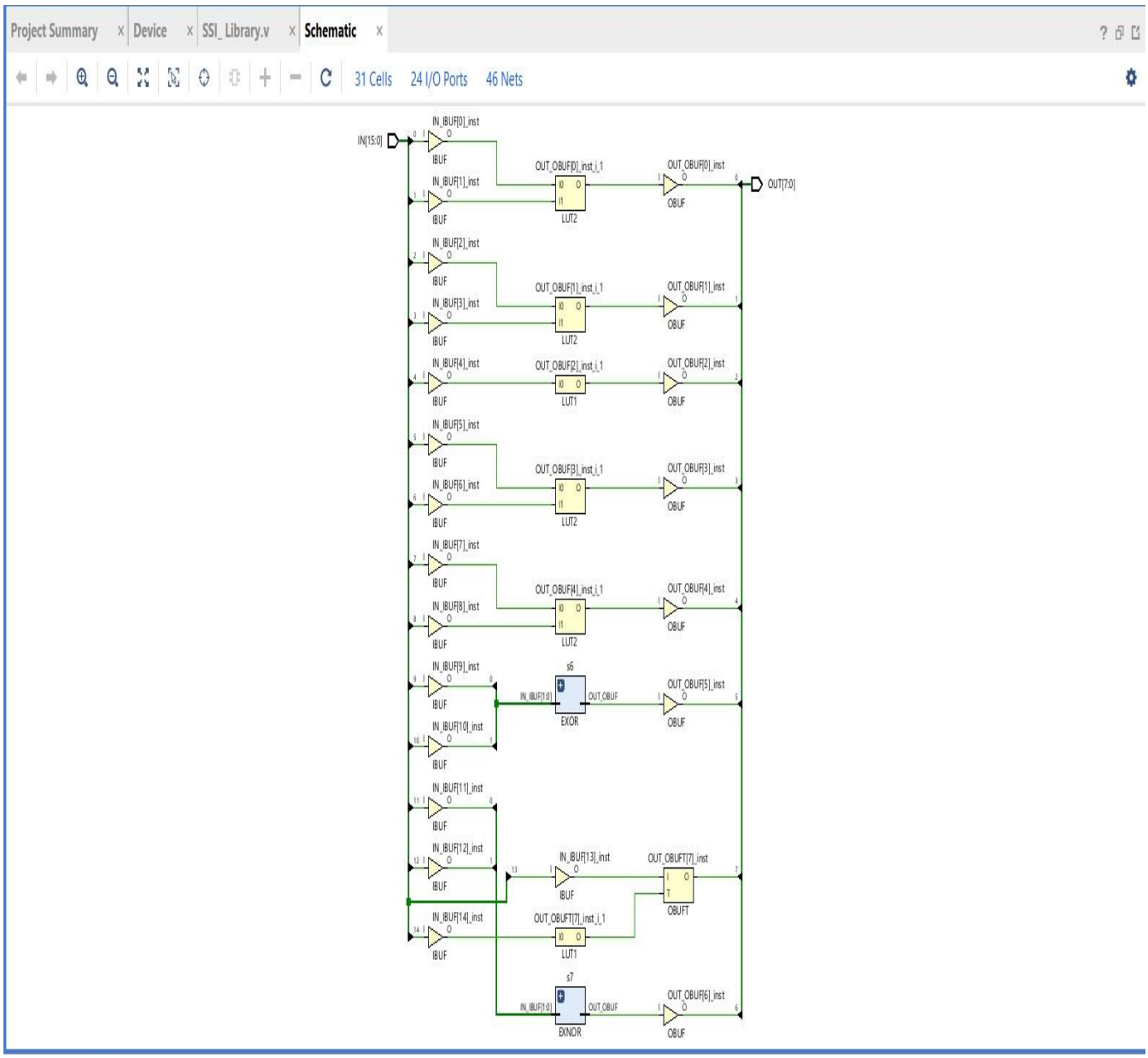
Behavioral simulation wave screenshot



RTL Schematic



Technology Schematic



Research

Fpga Resources

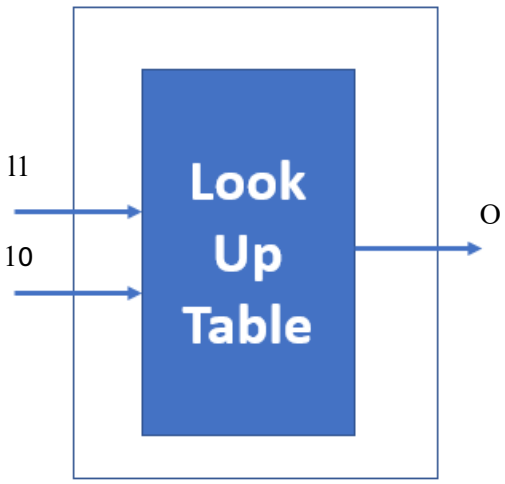
Look-Up Tables: Look-Up Tables work like truth tables. They are designed according to the user's functions. They provide a specific output based on the user's input.

Flip-Flops: Flip-Flops are basic parts of logic circuits. They store data and provide them with circuits according to clock signal. Flip-Flops help users design circuits with clock signals.

DSP Blocks: DSP blocks are an important part of FPGA. They include basic elements such as multipliers, adders, and subtractors. DSP blocks are used for signal processing work.

Look-Up Tables

Look-Up Tables are a basic part of FPGA. They provide a specific output based on the user's inputs. These inputs are set according to design codes. It implements the functions of these gates with Look-Up Tables.



Cell Properties

o_OBUF_inst_i_1

I1	I0	O=I0 & I1
0	0	0
0	1	0
1	0	0

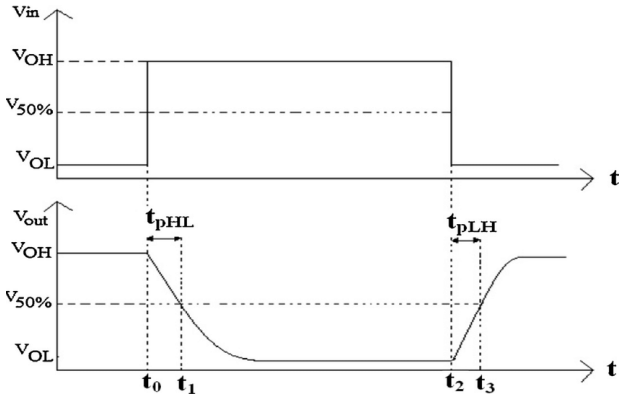
Look-Up Tables AND gate implementation

There is a Look-Up Table above that acts as a basic and gate. The truth table is generated based on the user's design code. It produces output in accordance with the values in the truth table based on the inputs received during use.

Glitches and Hazards

"Glitch" is a momentary faulty circuit that occurs in logic circuits. Hazards are design problems that cause glitches in logic circuits.

Logic gates contain CMOS transistors. Each transistor introduces a certain delay in the circuit flow. Although these delays are not visible to the naked eye, they can cause simulations to produce erroneous outputs on the order of nanoseconds or less.



Propagation Delay

Static 1 Hazards: This occurs when the output drops to 0 for a short time, even though it is expected to remain 1 at all times.

Static 0 Hazards: Static 0 hazards is when the output is 1 for a short time when it should always be 0.

Dynamic Hazards: Dynamic hazards cause the output to change several times during a value change.

Timing analysis is the most important stage to avoid glitches. Circuits which include more than paths can cause glitches if they do not have the same delay. Adding delays to fast paths is one solution to avoid glitches. Adding extra redundant gates can produce delays on the path.

SOURCES

https://www.maven-silicon.com/blog/digital-electronics-glitches-and-hazards/

https://www.researchgate.net/figure/Input-and-output-voltage-waveforms-of-CMOS-inverter-and-definitions-of-propagation-delay_fig3_283037145

https://hardwarebee.com/overview-of-lookup-tables-in-fpga-design/