Basic Logic Gates
Workshop

Lab 02

Table of contents

01

Basics Of DLD

Theory of Basics gates In DLD

Theory

NOT Gate:

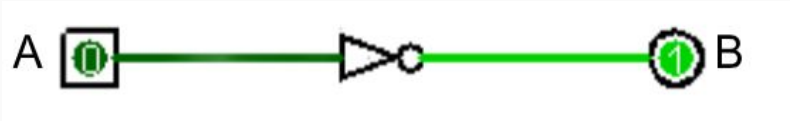
Definition:

The inverter performs basic logic gate operations called inversion. The purpose of an inverter is to change one logic level to the opposite level. When the high level is applied to the top of an inverter, the low level will appear at the output and vice versa.

Mathematical Expression:

$$B = \sim A$$

Diagram:



Truth Table:

Inputs	Outputs
A	**B**
0	1
1	0

Theory

AND Gate:

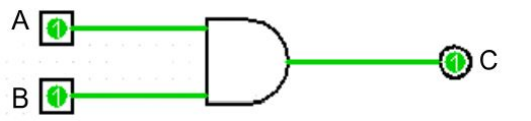
Definition:

The AND gate performs logical multiplication. The operation of the AND gate is such that the output is high only when all inputs are high and the output is low when any of the input is low.

Mathematical Expression:

$$C = A\&B$$

Diagram:



Truth Table:

Inputs		Outputs
A	**B**	**C**
0	0	0
0	1	0
1	0	0
1	1	1

Theory

OR Gate:

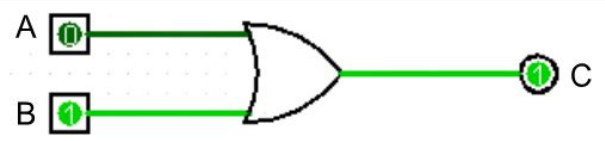
Definition:

The OR gate performs logical addition. The operation of the OR gate is such that the output is high only when one of its input is high and the output is low when both of its inputs are low.

Mathematical Expression:

$$C = A|B$$

Diagram:



Truth Table:

Inputs		Outputs
A	**B**	**C**
0	0	1
0	1	1
1	0	1
1	1	0

Theory

NAND Gate:

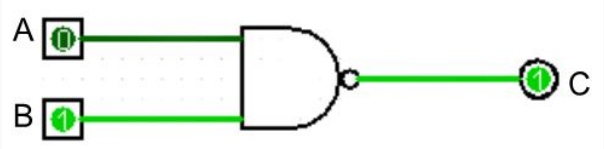
Definition:

The term NAND gate is derived from the AND gate. The operation of the NAND gate is such that the output is low only when all inputs are high and the output is high when any of the inputs are low.

Mathematical Expression:

$$C = \sim(A\&B)$$

Diagram:



Truth Table:

Inputs		Outputs
A	**B**	**C**
0	0	1
0	1	1
1	0	1
1	1	0

Theory

NOR Gate:

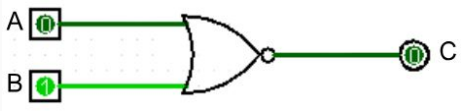
Definition:

The term NOR gate is derived from the OR gate. The operation of the NOR gate is such that the output is high only when all inputs are low and the output is low when any of the inputs are high.

Mathematical Expression:

$$C = \sim(A|B)$$

Diagram:



Truth Table:

Inputs		Outputs
A	**B**	**C**
0	0	0
0	1	0
1	0	0
1	1	1

Theory

XOR Gate:

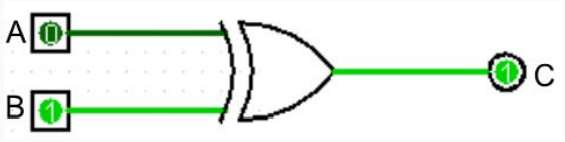
Definition:

The output of the XOR gate is high only when the inputs are at the opposite level.

Mathematical Expression:

$$C = A \oplus B$$

Diagram:



Truth Table:

Inputs		Outputs
A	**B**	**C**
0	0	0
0	1	1
1	0	1
1	1	0

Theory

XNOR Gate:

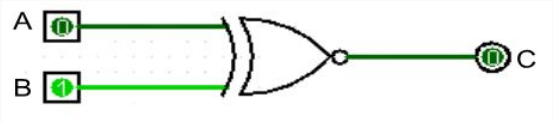
Definition:

The output of the XNOR gate is high only when the inputs are at the same level.

Mathematical Expression:

$$C = \sim(A \hat{} B)$$

Diagram:



Truth Table:

Inputs		Outputs
A	**B**	**C**
0	0	1
0	1	0
1	0	0
1	1	1

02

Verilog Coding

Implement Coding Of Gates with testbench.

Verilog Coding

Source Code Of *NOT* Gate

```
module top(
    input wire clk,
    input wire A,
    output wire led
);
    reg NOT;
    always @(posedge clk) begin
        if (A) begin
            NOT <= 0;
        end
        else begin
            NOT <= 1;
        end
    end
    assign led = NOT; // Connect NOT to the LED
endmodule
```

Verilog Coding

TestBench Code Of *NOT* Gate

```
module test;

    reg clk;
    reg A;
    wire led;

    // Instantiate the module to be tested
    top top_i (

        .clk(clk),
        .A(A),
        .led(led)
    );

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

initial begin
        // vcd dump
        $dumpfile("not.vcd");
        $dumpvars(0);
        // Initialize inputs
        clk = 0;
        A = 0;
        // Simulate for some clock cycles
        #10;
        A = 1;
        #10;
        A = 0;
        #10;
        $finish;
    end
endmodule
```

Verilog Coding

Source Code Of *AND* Gate

```
module top(
    input wire clk,
    input wire A,
    input wire B,     // Input B for AND gate
    output wire led
);
    reg AND;
    always @(posedge clk) begin
        if (A & B) begin
            AND <= 1;  // Set AND gate output to 1 if A and B are both 1
        end
        else begin
            AND <= 0;  // Set AND gate output to 0 otherwise
        end
    end
    assign led = AND; // Connect AND gate output to the LED
endmodule
```

Verilog Coding

TestBench Code Of *AND* Gate

```
module test;

    reg clk;
    reg A;
    wire led;

    // Instantiate the module to be tested
    top top_i (
        .clk(clk),
        .A(A),
        .led(led)
    );

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

initial begin
        // vcd dump
        $dumpfile("not.vcd");
        $dumpvars(0);
        // Initialize inputs
        clk = 0;
        A = 0;

        // Simulate for some clock cycles
        #10;
        A = 1;
        #10;
        A = 0;
        #10;
        $finish;
    end
endmodule
```

03

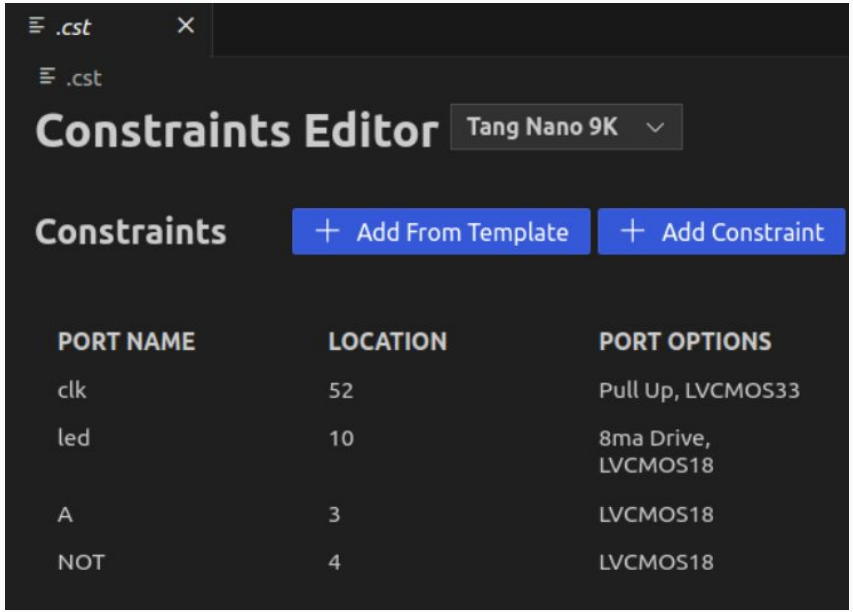
FPGA Porting.

Writing Constraint file.

FPGA Porting.

Constraint file:

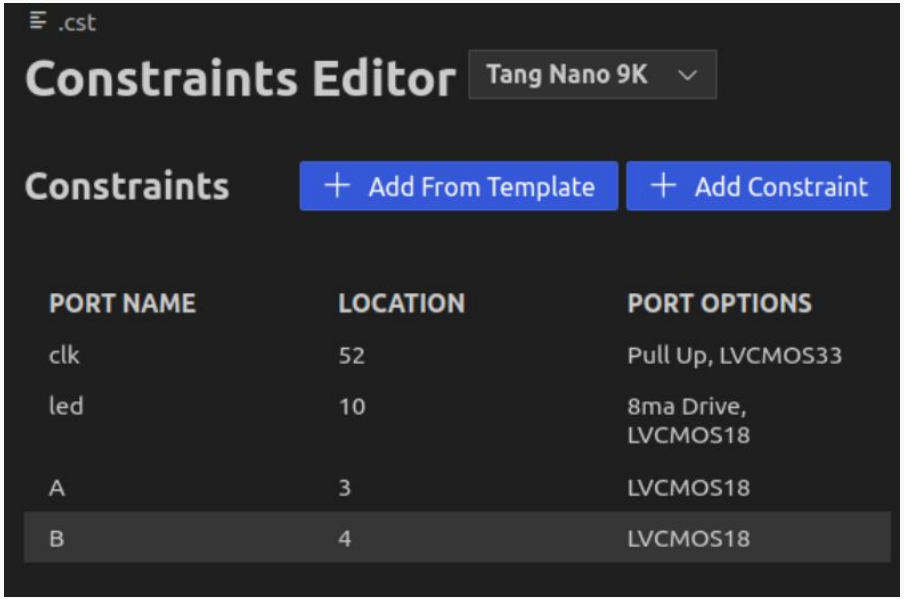
not.cst



FPGA Porting

Constraint file:

And.cst



04

Exercise.

Exercise Task.

Tasks

a. **Implement OR gates in verilog.**
b. **Implement NAND and NOR gates in verilog.**
c. **Implement XOR and XNOR gates in verilog.**

Testimonial

Author:

Abdul Muheet Ghani, Research Assistant at MERL-UITU.

Under The Supervision Of:

Dr.Ali Ahmed (Team Lead MERL).
Sajjad Ahmed (Research Associate)

Sponsored By:

Edmund from Symbiotic EDA. for sponsoring FPGA.

Thanks:

Lushay Lab (They've provided crucial resources and guidance throughout the project), Also thanks to Slidesgo.

Future Work

This is version 1.0 of our course. We will continue this training and very soon release further versions. For any questions or to stay connected with us.

Github: Abdul Muheet Ghani.
Gmail: Dr.Ali Ahmed. Sajjad Ahmed. Abdul Muheet Ghani.