Lab 1

Primitive Gates

Lab Workshop on ARM Based SOC Design

Name:	**Satvikkumar Patel**
Email:	Satvikpatel135@gmail.com
Date of Sub:	January 31, 2024

Contents

1 Primitive Gates **4**
1.1 OR gate . . . 4
1.2 NOR gate . . . 4
1.3 AND gate . . . 4
1.4 NAND gate . . . 4
1.5 NOT gate . . . 4
1.6 XOR gate . . . 5
2 Test Bench **6**
3 Output waveform **7**
4 Questions **8**

NATIONAL INSTITUTE OF ELECTRONICS AND INFORMATION TECHNOLOGY, CALICUT

NPTEL Lab Workshop Manual

Lab Workshop on ARM Based SOC Design

Contact Point for the Lab

Mr. Sreejeesh SG

Senior Technical Officer

Email: sreejeesh@nielit.gov.in

Ph.: 9447769756 (WhatsApp Preferred)

Introduction to Combinational Circuit Simulation Lab: 1

Logic gates

Introduction

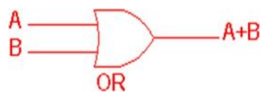
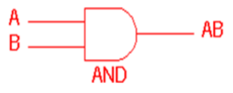
The purpose of this experiment is to simulate the behavior of several of the basic logic gates and you will connect several logic gates together to create simple digital model.

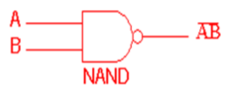
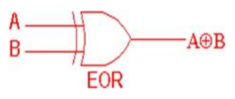
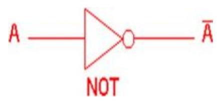
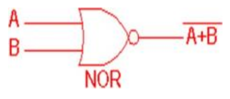
Software tools Requirement

Modelsim (Siemens)

Xilinx Vivado

Logic Gates and their Properties

Gate	Description	Truth Table			Logic Symbol
OR	The output is active high if any one of the input is in active high state, Mathematically, Q = A+B	A 0 0 1 1	B 0 1 0 1	Output Q 0 1 1	
	The output is active	A	B	Output	
AND	high only if both the inputs are in active high state, Mathematically,	0 0 1	0 1 0	Q 0 0	
	Q = A.B	1	1	0	

NAND	The output is active high only if any one of the input is in active low state, Mathematically, Q = (A.B)’	A 0 0 1 1	B 0 1 0 1	Output Q 1 1 1 0	
XOR	The output is active high only if any one of the input is in active high state, Mathematically, Q = A.B’ + B.A’	A 0 0 1 1	B 0 1 0 1	Output Q 0 1 1 0	
NOT	In this gate the output is opposite to the input state, Mathematically, Q = A^1	A 0 1		Output Q 1 0	
NOR	The output is active high only if both the inputs are in active low state, Mathematically, Q = (A+B)’	A 0 0 1 1	B 0 1 0 1	Output Q 1 0 0	

Describe the following basic logic gates in Verilog HDL and capture the Waveforms

Questions to answered after this lab

1. *What is meant by ports?*
2. *Write the different types of port modes.*
3. *What are different types of operators?*
4. *What is difference b/w <= and: = operators?*
5. *What is meant by simulation?*

1 Primitive Gates

1.1 OR gate

```
module OR_gate (A,B,Y);

parameter integer w = 8;
input wire [w-1:0] A,B;
output wire [w-1:0] Y;

assign Y = A | B;

endmodule
```

1.2 NOR gate

```
module NOR_gate (A,B,Y);

parameter integer w = 8;
input wire [w-1:0] A,B;
output wire [w-1:0] Y;

assign Y = !(A | B);

endmodule
```

1.3 AND gate

```
module AND_gate (A,B,Y);

parameter integer w = 8;
input wire [w-1:0] A,B;
output wire [w-1:0] Y;

assign Y = A & B;

endmodule
```

1.4 NAND gate

```
module NAND_gate (A,B,Y);

parameter integer w = 8;
input wire [w-1:0] A,B;
output wire [w-1:0] Y;

assign Y = !(A & B);

endmodule
```

1.5 NOT gate

```
module NOT_gate (A,Y);

parameter integer w = 8;
input wire [w-1:0] A;
output wire [w-1:0] Y;

assign Y = ~A ;

endmodule
```

1.6 XOR gate

```
module XOR_gate (A,B,Y);
parameter integer w = 8;
input wire [w-1:0] A,B;
output wire [w-1:0] Y;
assign Y = A ^ B;
endmodule
```

2 Test Bench

```
‘timescale 1ns/1ps
module tb_primitive_gates;

wire  A,B;
wire Y_OR, Y_NOR, Y_AND, Y_NAND, Y_NOT, Y_XOR;

OR_gate #(.w(1)) dut_OR (.A(A), .B(B), .Y(Y_OR));
NOR_gate #(.w(1)) dut_NOR (.A(A), .B(B), .Y(Y_NOR));
AND_gate #(.w(1)) dut_AND (.A(A), .B(B), .Y(Y_AND));
NAND_gate #(.w(1)) dut_NAND (.A(A), .B(B), .Y(Y_NAND));
NOT_gate #(.w(1)) dut_NOT (.A(A), .Y(Y_NOT));
XOR_gate #(.w(1)) dut_XOR (.A(A), .B(B), .Y(Y_XOR));

reg [1:0] count;
reg clk;

always #5 clk = ~clk;

always@(posedge clk)
begin
if (count < 2’b11)
  count =  count + 1;
else
  count = 2’b00;
end

assign A = count[1];
assign B = count[0];


initial
begin
clk = 0;
#200 $stop;
end


endmodule
```

3 Output waveform

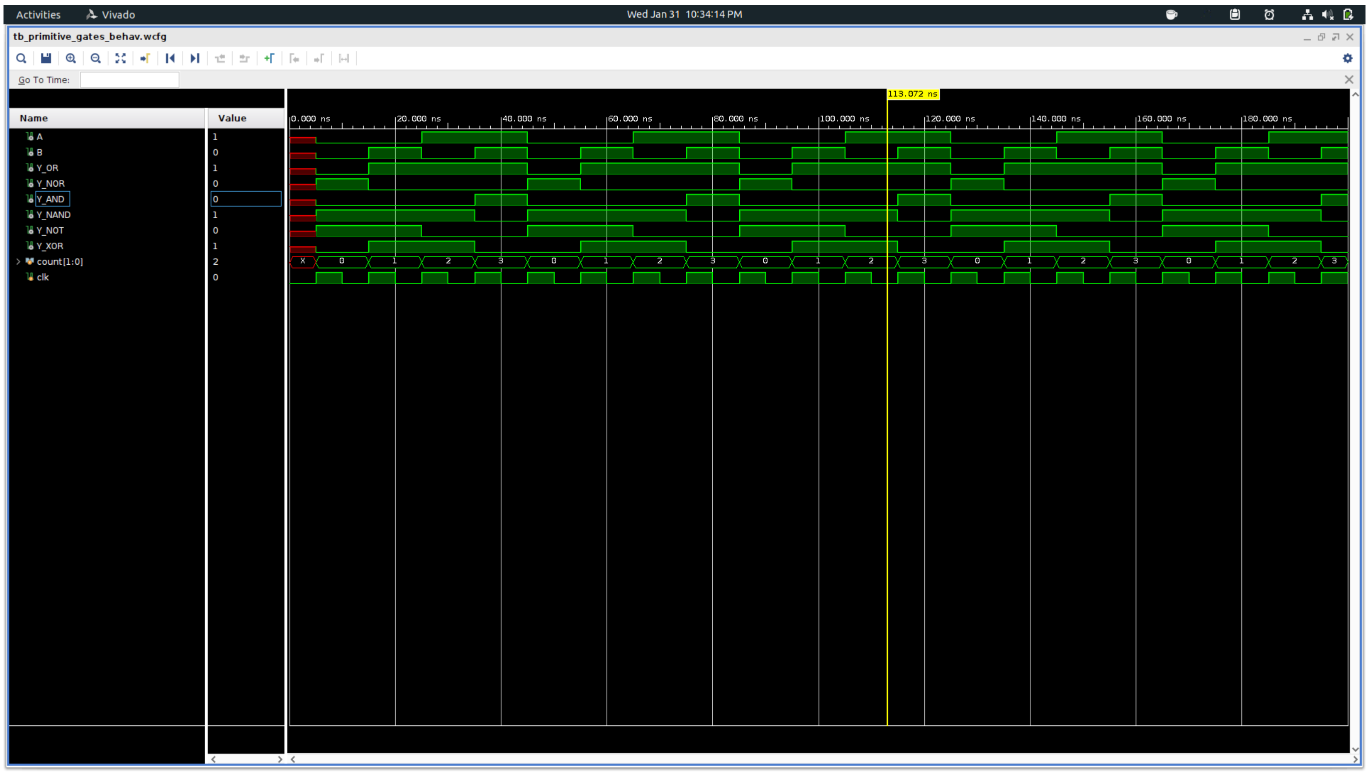


Figure 1: Output waveform in Vivado

4 Questions

1. **What is meant by ports?**

   Ports represent a module's *interface*, acting as connections to the external world or other modules. They enable data *input* and *output* for communication and interaction. Ports support *modularity* and *reusability* by encapsulating internal functionality while defining interaction points.

2. **Write the different types of port modes.**

   Verilog offers three main port modes:

   Input: Data flows only *into* the module through this port.
   Output: Data flows only *out of* the module through this port.
   Inout: Data can flow *both in and out* of the module through this port (bidirectional).

3. **What are different types of operators?**

   Verilog provides diverse operators for manipulating signals through logic and arithmetic operations:
   Logical:‘&&‘, ‘||‘(and, or)
   Relational:‘ $<$ ‘,‘ $>$ ‘,‘ $>=$ ‘,‘ $<=$ ‘,‘ $==$ ‘,‘ $===$ ‘$(less\ than, greater\ than, etc.)$
   Arithmetic:‘ $+$ ‘,‘ $-$ ‘,‘ $*$ ‘,‘$/$‘,‘%‘$(addition, subtraction, multiplication, etc.)$
   Shift:‘ $\ll$ ‘,‘ $\gg$ ‘$(left\ and\ right\ shift)$
   Bitwise:‘&‘,‘ $\sim (and, not)$

4. **What is difference b/w <= and := operators?**

   Both ‘<=‘ and ‘:=‘ assign values to signals, but they differ in usage:

   <= : Used within procedural blocks for *delayed assignment*, setting values on the next clock edge.
   := : Used outside procedural blocks for *immediate assignment*, setting values instantaneously.

5. **What is meant by simulation?**

   Verilog simulation is a virtual testing ground for your design. You run your code through a simulator, feeding it different inputs and observing outputs. This lets you check for errors, analyze timing, and ensure functionality before building actual hardware, saving time and resources.