Lab 4

Ashmitha Murali

Prof. Alex Yang

Verilog HDL Lab EE46L

8th October 2025

1. I have designed all the gates using both continuous assignment and always block. I have tested using different possible input combinations including x/z inputs. Below is the output waveform for all gates.

a1 is input for the not gate and (a,b) are the two inputs for the 2 bit gates. Name of the output is appended with “\_asg” for continuous assignment and “\_alw” for always block outputs.

Waveform:

A screenshot of a computer screen

AI-generated content may be incorrect.

**Code used:**  
`timescale 1ns/1ps

//Not gate

module not1\_assign(input wire a, output wire y);

assign y = ~a;

endmodule

module not1\_always(input wire a, output reg y);

always @\* y = ~a;

endmodule

//AND gate

module and2\_assign(input wire a, b, output wire y);

assign y = a & b;

endmodule

module and2\_always(input wire a, b, output reg y);

always @\* y = a & b;

endmodule

//OR gate

module or2\_assign(input wire a, b, output wire y);

assign y = a | b;

endmodule

module or2\_always(input wire a, b, output reg y);

always @\* y = a | b;

endmodule

//NAND Gate

module nand2\_assign(input wire a, b, output wire y);

assign y = ~(a & b);

endmodule

module nand2\_always(input wire a, b, output reg y);

always @\* y = ~(a & b);

endmodule

//NOR gate

module nor2\_assign(input wire a, b, output wire y);

assign y = ~(a | b);

endmodule

module nor2\_always(input wire a, b, output reg y);

always @\* y = ~(a | b);

endmodule

//XOR gate

module xor2\_assign(input wire a, b, output wire y);

assign y = a ^ b;

endmodule

module xor2\_always(input wire a, b, output reg y);

always @\* y = a ^ b;

endmodule

//XNOR gate

module xnor2\_assign(input wire a, b, output wire y);

assign y = ~(a ^ b);

endmodule

module xnor2\_always(input wire a, b, output reg y);

always @\* y = ~(a ^ b);

endmodule

**Testbench used:**

`timescale 1ns/1ps

module tb\_rtl\_gates\_2inputs;

reg a1;

wire y1\_asg, y1\_alw;

reg a, b;

wire y\_and\_asg, y\_and\_alw;

wire y\_or\_asg, y\_or\_alw;

wire y\_nand\_asg, y\_nand\_alw;

wire y\_nor\_asg, y\_nor\_alw;

wire y\_xor\_asg, y\_xor\_alw;

wire y\_xnor\_asg, y\_xnor\_alw;

// DUTs

not1\_assign u\_not1\_assign(.a(a1), .y(y1\_asg));

not1\_always u\_not1\_always(.a(a1), .y(y1\_alw));

and2\_assign u\_and2\_assign (.a(a), .b(b), .y(y\_and\_asg ));

and2\_always u\_and2\_always (.a(a), .b(b), .y(y\_and\_alw ));

or2\_assign u\_or2\_assign (.a(a), .b(b), .y(y\_or\_asg ));

or2\_always u\_or2\_always (.a(a), .b(b), .y(y\_or\_alw ));

nand2\_assign u\_nand2\_assign(.a(a), .b(b), .y(y\_nand\_asg));

nand2\_always u\_nand2\_always(.a(a), .b(b), .y(y\_nand\_alw));

nor2\_assign u\_nor2\_assign (.a(a), .b(b), .y(y\_nor\_asg ));

nor2\_always u\_nor2\_always (.a(a), .b(b), .y(y\_nor\_alw ));

xor2\_assign u\_xor2\_assign (.a(a), .b(b), .y(y\_xor\_asg ));

xor2\_always u\_xor2\_always (.a(a), .b(b), .y(y\_xor\_alw ));

xnor2\_assign u\_xnor2\_assign(.a(a), .b(b), .y(y\_xnor\_asg));

xnor2\_always u\_xnor2\_always(.a(a), .b(b), .y(y\_xnor\_alw));

// Dump file for waveform viewing

initial begin

$dumpfile("rtl\_gates\_2inputs\_tb.vcd");

$dumpvars(0, tb\_rtl\_gates\_2inputs);

end

integer i;

initial begin

// Test for 2-input gates

for (i = 0; i < 4; i = i + 1) begin

{a, b} = i[1:0];

#5;

end

//Test for X and Z

a = 1'bx; b = 1'b0; #5;

a = 1'b1; b = 1'bx; #5;

a = 1'bz; b = 1'b1; #5;

a = 1'b0; b = 1'bz; #5;

// Test for 1-bit NOT

a1 = 0; #5;

a1 = 1; #5;

a1 = 1'bx; #5;

a1 = 1'bz; #5;

a1 = 0; #5;

$finish;

end

endmodule