**Basic designs:**

**AND:**

module nivi(

input a,

input b,

output y

);

and(y,a,b);

Endmodule

**TEST bench:**

`timescale 1ns / 1ps

module tb\_nivi;

// Inputs

reg a;

reg b;

// Output

wire y;

// Instantiate the Unit Under Test (UUT)

nivi uut (

.a(a),

.b(b),

.y(y)

);

initial begin

// Monitor output

$display("Time\ta b | y");

$display("----------------");

a = 0; b = 0; #10;

$display("%0t\t%b %b | %b", $time, a, b, y);

a = 0; b = 1; #10;

$display("%0t\t%b %b | %b", $time, a, b, y);

a = 1; b = 0; #10;

$display("%0t\t%b %b | %b", $time, a, b, y);

a = 1; b = 1; #10;

$display("%0t\t%b %b | %b", $time, a, b, y);

$finish;

end

**OR:**

module or\_gate(

input x,y,

output z

);

or(z,x,y);

Endmodule

**TEST\_bench:**

`timescale 1ns/1ps

module or\_gate\_tb;

reg x, y;

wire z;

// Instantiate the module

or\_gate uut (

.x(x),

.y(y),

.z(z)

);

initial begin

// Create VCD dump

$dumpfile("or\_gate.vcd");

$dumpvars(0, or\_gate\_tb);

// Test all combinations

x = 0; y = 0; #10;

x = 0; y = 1; #10;

x = 1; y = 0; #10;

x = 1; y = 1; #10;

$finish;

end

endmodule

**XOR-gate:**

module xor\_gate(

input a,

input b,

output y

);

assign y = a ^ b;

Endmodule

**Test-bench:**

`timescale 1ns / 1ps

module xor\_gate\_tb;

// Testbench signals

reg a, b;

wire y;

// Instantiate the XOR gate

xor\_gate uut (

.a(a),

.b(b),

.y(y)

);

initial begin

$display("A B | Y");

$display("---------");

// Test all combinations

a = 0; b = 0; #10;

$display("%b %b | %b", a, b, y);

a = 0; b = 1; #10;

$display("%b %b | %b", a, b, y);

a = 1; b = 0; #10;

$display("%b %b | %b", a, b, y);

a = 1; b = 1; #10;

$display("%b %b | %b", a, b, y);

$finish;

end

endmodule