LAB 01

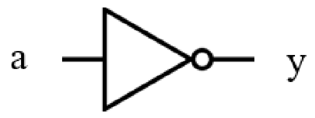
INVERTER AND XOR2

CECS 225 – DIGITAL LOGIC AND ASSEMBLY PROGRAMMING

Professor: Xiaolong Wu
Student: Thanh Nguyen _ ID: 026843815

I. Part 01: Inverter

1. Description:
   - The purpose of this function is performing logical negation on its input. In other words, if the input is 0, then the output will be 1. Similarly, an 1 input results in a 0 output.
   - Use logic gate (NOT) to do this function



   - Truth table:

a	**y = NOT a**
0	1
1	0

2. Verilog Codes:
   - Design Code

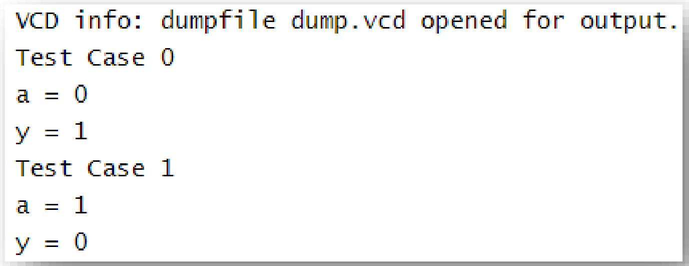
```
module inverter(a, y);
  input a;
  output y;

  assign y = ~a; // y = Not a

endmodule
```

   - Testbench

```
`timescale 1ns/1ps

module testbench();
  reg a1;
  wire y1;
  inverter inv1(a1, y1);

  initial
    begin
      // Dump waves
      $dumpfile("dump.vcd");
      $dumpvars(1, testbench);

      // Test Case a = 0
      $display("Test Case 0");
      a1 = 1'b0;
      $display("a = %b", a1);
      #1
      $display("y = %b", y1);
```

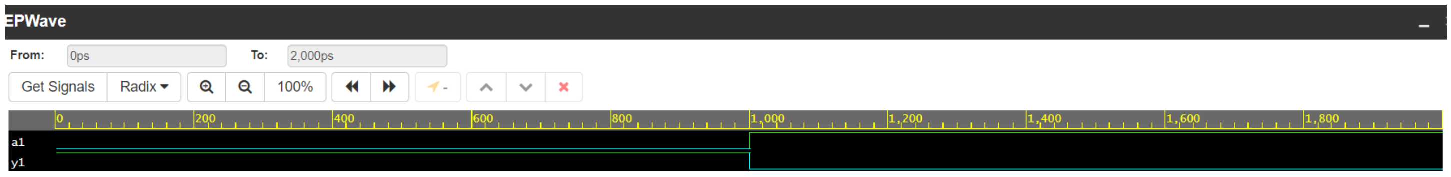
```
        // Test Case a = 1
        $display("Test Case 1");
        a1 = 1'b1;
        $display("a = %b", a1);
        #1
        $display("y = %b", y1);
      end
endmodule
```

3. Simulator Waveform:
   - Inputs and Outputs:



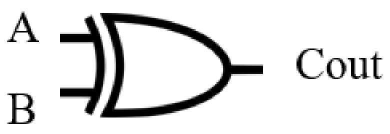
```
VCD info: dumpfile dump.vcd opened for output.
Test Case 0
a = 0
y = 1
Test Case 1
a = 1
y = 0
```

   - Waveform:



II. Part 02: Xor2

1. Description:
   - This function is used to implement an exculsive disjunction. Output will be a logical 1 if the two input values differ, i.e., its output is a logical 1 if either of its inputs are 1, but not at the same time (exculsively)
   - Use logic gate (XOR) for this function with two inputs A and B, and one output: Cout = A ⊕ B



   - Truth table

A	B	Cout = A XOR B

0	0	0
0	1	1
1	0	1
1	1	0

2. Verilog Codes:
   - Design Code

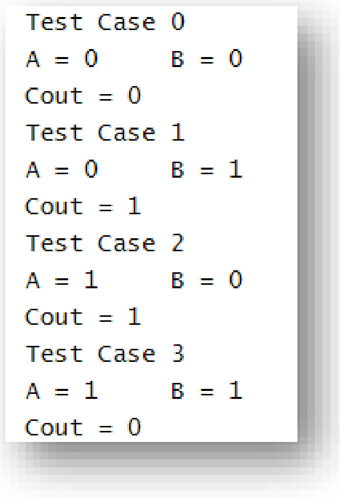
```
module XOR2(A, B, Cout);
  input A, B;
  output Cout;

  assign Cout = A ^ B; // Cout = A XOR B

endmodule
```

   - Testbench

```
`timescale 1ns/1ps

module testbench();
  reg A1, B1;
  wire Cout1;
  XOR2 Xor2_1(A1, B1, Cout1);

  initial
    begin
      //Dump waves
      $dumpfile("dump.vcd");
      $dumpvars(1, testbench);

      $display("Test Case 0");
      A1 = 1'b0; B1 = 1'b0;
      $display("A = %b", A1, "      B = %b", B1);
      #1
      $display("Cout = %b", Cout1);

      $display("Test Case 1");
      A1 = 1'b0; B1 = 1'b1;
      $display("A = %b", A1, "      B = %b", B1);
      #1
      $display("Cout = %b", Cout1);

      $display("Test Case 2");
      A1 = 1'b1; B1 = 1'b0;
      $display("A = %b", A1, "      B = %b", B1);
      #1
      $display("Cout = %b", Cout1);

      $display("Test Case 3");
      A1 = 1'b1; B1 = 1'b1;
      $display("A = %b", A1, "      B = %b", B1);
```

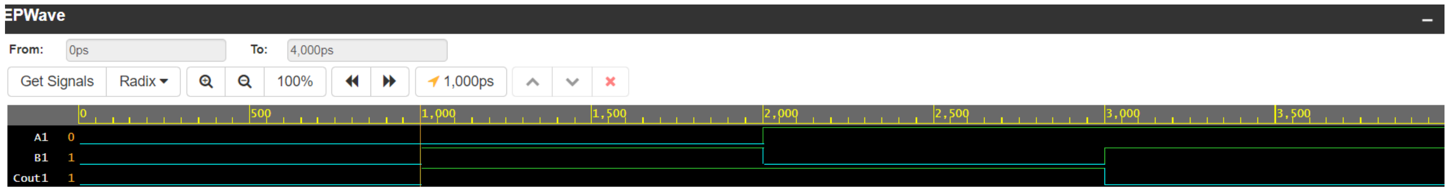
```
        #1
        $display("Cout = %b", Cout1);
    end

endmodule
```

3. Simulator Waveform:
   - Inputs and Outputs:



```
Test Case 0
A = 0     B = 0
Cout = 0
Test Case 1
A = 0     B = 1
Cout = 1
Test Case 2
A = 1     B = 0
Cout = 1
Test Case 3
A = 1     B = 1
Cout = 0
```

   - Waveform:



From: 0ps To: 4,000ps