Course Name: Computer Architecture	**Course Code:** CMPE-421L
Assignment Type: Lab	**Dated:** 4th September 2023
Semester: 7th	**Session:** 2020
Lab/Project/Assignment #: 1	**CLOs to be covered: CLO 1**
Lab Title: Design & Verification of Basic Combinational Logic	**Teacher Name:** Engr. Afeef Obaid

Lab Evaluation:

CLO1	Understand the design and simulation of basic digital circuits with HDL and HDL simulation tools.					
Levels (Marks)	**Level1**	**Level2**	**Level3**	**Level4**	**Level5**	**Level6**
(5)						
					Total	**/10**

Rubrics for Current Lab Evaluation

Scale	Marks	Level	Rubric
Excellent	**9-10**	L1	Submitted all lab tasks, BONUS task, have good understanding.
Very Good	**7-8**	L2	Submitted the lab tasks but have good understanding
Good	**5-6**	L3	Submitted the lab tasks but have weak understanding.
Basic	**3-4**	L4	Submitted the lab tasks but have no understanding.
Barely Acceptable	**1-2**	L5	Submitted only one lab task.
Not Acceptable	**0**	L6	Did not attempt

LAB No 1

Lab Goals/Objectives:

By reading this manual, students will be able to:

- Learn about basic HDL commands
- Create a system verilog module to implement AND, NOR and XOR gates.
- Simulate and verify gate behaviors.

Equipment Required: Computer system with ModelSim or Xcelium, installed on it

Introduction to System Verilog

Introduction

Welcome to the System Verilog lab series! In these labs, you'll learn the fundamentals of System Verilog, a hardware description and verification language widely used in the field of digital design and verification. Whether you are a student aiming to understand digital systems or a professional looking to enhance your verification skills, this series will provide you with hands-on experience and practical knowledge.

Lab Setup

Before we dive into the labs, let's ensure you have the necessary tools set up on your computer. You will need the following:

1. A System Verilog simulator (e.g., ModelSim, Xcelium, etc).
2. A text editor or integrated development environment (IDE) for writing System Verilog code.
3. (Optional) A waveform viewer for visualizing simulation results.

For now, instead of installing the tools in our computers we will be using an online platform known as EDA Playground equipped with many digital design tools for beginners.
Please make an account here: https://edaplayground.com/ with your university emails.

Description

In this exercise, you will start with a simple combinational logic design. You'll design a 2-input AND gate using System Verilog and simulate its behavior. This exercise will help you get familiar with the basic syntax of System Verilog.

Instructions

1. Open the home page of EDA playground. There will be two windows by default design.sv and testbench.sv.
2. Create a new system verilog module named "and_gate" with two input ports (A and B) and one output port (Y).
3. Compile the design to be sure that there are no syntax errors.

design.sv

```
module and_gate
(
    input wire A,
    input wire B,
    output wire Y
);

assign Y = A & B;

endmodule
```

4. Once you've written your system Verilog design code, the next step is to simulate your design using the chosen simulator.
5. Write a testbench module for your design which is in our case is an AND gate.

testbench.sv

```
module tb_and_gate;

  // Define signals for connecting to the DUT (Device Under Test)
  logic A;
  logic B;
  logic Y;

  // Instantiate the AND gate module
  and_gate dut
  (
    .A(A),
    .B(B),
    .Y(Y)
  );

  // Initialize the signals
  initial
  begin
    A = 0; // Set input A to logic 0
    B = 0; // Set input B to logic 0

    // Apply test vectors
    $display("Input A\tInput B\tOutput Y");
    $display("----------------------------");

    // Test Case 1: A=0, B=0
    A = 0;
```

```
    B = 0;
    #10; // Wait for a simulation time of 10 time units
    $display("%b\t%b\t%b", A, B, Y);

    // Test Case 2: A=0, B=1
    A = 0;
    B = 1;
    #10; // Wait for a simulation time of 10 time units
    $display("%b\t%b\t%b", A, B, Y);

    // Test Case 3: A=1, B=0
    A = 1;
    B = 0;
    #10; // Wait for a simulation time of 10 time units
    $display("%b\t%b\t%b", A, B, Y);

    // Test Case 4: A=1, B=1
    A = 1;
    B = 1;
    #10; // Wait for a simulation time of 10 time units
    $display("%b\t%b\t%b", A, B, Y);

    // End the simulation
    $finish;
  end

endmodule
```

6. Compile your design and testbench modules.
7. Run the simulation.

Troubleshooting and Debugging

Common Debugging Techniques

Debugging system verilog code is an essential skill for any digital design or verification engineer. Here are some common debugging techniques:

1. Display/Monitor Statements: Insert display/monitor statements in your testbench to display signal values during simulation.
2. Waveform Viewing: Use a waveform viewer to visualize the behavior of your signals and modules. Add the following initial block in your testbench to generate waveform file.

```
initial
begin
    $dumpfile("tb_module_name.vcd");
    $dumpvars(0, tb_module_name);
end
```

3. Assertion-Based Verification: Implement assertions in your testbench to check for expected behaviors and detect issues early.

Review Questions and Lab Tasks

Review Question 1
What is the purpose of a testbench in System Verilog, and why is it important in the verification process?

Lab Task 1
Design a 4-input OR gate using system verilog and create a testbench to verify its functionality. Provide the system verilog code for both the gate and the testbench.

Lab Task 2
Design a 3-input NOR gate using system verilog and create a testbench to verify its functionality. Provide the system verilog code for both the gate and the testbench.

Lab Task 3
Design a 2-input XOR gate using system verilog and create a testbench to verify its functionality. Provide the system verilog code for both the gate and the testbench.