**Department of Electrical Engineering**

**Faculty Member:** Arshad Nazir **Dated:** 30/09/2021

**Semester:** 3rd **Section:** BEE 12C

**Group No.: 7**

**EE-221: Digital Logic Design**

Lab 2: Introduction to Verilog

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **PLO4/CLO4** | **PLO4/CLO4** | **PLO5/CLO5** | **PLO8/CLO6** | **PLO9/CLO7** |  |
| **Name** | **Reg. No** | **Viva / Lab Performance** | **Analysis of data in Lab Report** | **Modern Tool Usage** | **Ethics and Safety** | **Individual and Teamwork** | **Total marks Obtained** |
|  |  | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **25 Marks** |
| **Muhammad Adil Azeem** | 342459 |  |  |  |  |  |  |
| **Muhammad Umer** | 345834 |  |  |  |  |  |  |
| **Raahim Ahmad Waqar** | 342287 |  |  |  |  |  |  |
| **Saad Bakhtiar** | 341150 |  |  |  |  |  |  |
| **Syed Aun Ali Kazmi** | 342384 |  |  |  |  |  |  |

## Lab 2: Introduction to Verilog, Gate-level/Behavioral Modeling and Hardware Implementation of Basic Logic Circuit

This Lab has been divided into two parts.

In first part you will be introduced to Verilog and Gate-Level Modeling.

The next part is the hardware implementation of a Boolean function given to you.

**Objectives:**

* Understand HDL and compare it with normal programming languages.
* Simulate Basic Gates using Verilog with ModelSim
* Write stimulus using Verilog
* Derive algebraic expression for a Boolean function from the given schematics.
* Hardware Implementation of Logic Circuit

**Lab Instructions**

* This lab activity comprises three parts, namely Pre-lab, Lab tasks, and Post-Lab Viva session.
* The lab report will be uploaded on LMS three days before the scheduled lab date. The students will get a hard copy of the lab report, complete the Pre-lab task before coming to the lab and deposit it with the teacher/lab engineer for necessary evaluation. Alternately each group uploads a completed lab report on LMS for grading.
* The students failing to submit Pre-Lab will not be allowed to perform Lab work.
* The students will start lab task and demonstrate design steps separately for stepwise evaluation (course instructor/lab engineer will sign each step after ascertaining functional verification)
* Remember that a neat logic diagram with pins numbered coupled with a nicely patched circuit will simplify trouble-shooting.
* After the lab, students are expected to unwire the circuit and deposit back components before leaving.
* The students will complete a lab task and submit a complete report to the Lab Engineer before leaving the labs. The Verilog tutorial part is non-printable and for reference only.
* There are related questions at the end of this activity. Give complete answers.

**Pre-Lab Tasks: (2 marks)**

1. Read the manual Getting Started with Verilog and answer the following questions.
2. **HDL stands for:**

**Hardware Description Language**

1. **Two standard versions of HDL are**
2. **VHDL**
3. **Verilog HDL**
4. **Give the different levels of abstraction in Verilog HDL**

**Verilog Modeling Level**

1. **Gate Level Modeling** - using the basic gates and, or, Not, etc. to and their inter-connection to describe the whole system
2. **Dataflow Modeling** - defining the flow of data among different components of design using operators, no interconnection between gates needed.
3. **Behavioral Modeling** - defining the overall behavior of the system like if input is x, then output is y.
4. **Switch Level Modeling** - the transistor level of hardware is dealt with.

**Lab Tasks: (8 marks)**

1. **Lab Task 1: (2 marks)**

Model and simulate the basic gates i.e., NOT, AND & OR in Verilog (Gate level) using ModelSim. Compare the simulation waveform results with the truth table in the space given below.

**NOT Gate**

module task1(out, in1, in2);

    input in1, in2;

    output out;

    and a1(out, in1, in2);

endmodule

module test1;

    reg IN1,IN2;

    wire OUT;

    task1 t1(OUT,IN1,IN2);

    initial begin

        #100ps IN1 = 0; IN2 = 0;

        #100ps IN1 = 0; IN2 = 1;

        #100ps IN1 = 1; IN2 = 0;

        #100ps IN1 = 1; IN2 = 1;

    end

endmodule



**AND Gate**

module task1(out, in1, in2);

    input in1, in2;

    output out;

    and a1(out, in1, in2);

endmodule

module test1;

    reg IN1,IN2;

    wire OUT;

    task1 t1(OUT,IN1,IN2);

    initial begin

        #100ps IN1 = 0; IN2 = 0;

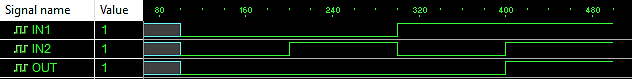
        #100ps IN1 = 0; IN2 = 1;

        #100ps IN1 = 1; IN2 = 0;

        #100ps IN1 = 1; IN2 = 1;

    end

endmodule



**OR Gate**

module task1(out, in1, in2);

    input in1, in2;

    output out;

    and a1(out, in1, in2);

endmodule

module test1;

    reg IN1,IN2;

    wire OUT;

    task1 t1(OUT,IN1,IN2);

    initial begin

        #100ps IN1 = 0; IN2 = 0;

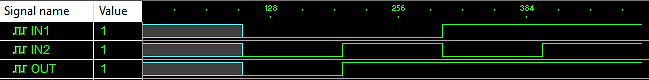
        #100ps IN1 = 0; IN2 = 1;

        #100ps IN1 = 1; IN2 = 0;

        #100ps IN1 = 1; IN2 = 1;

    end

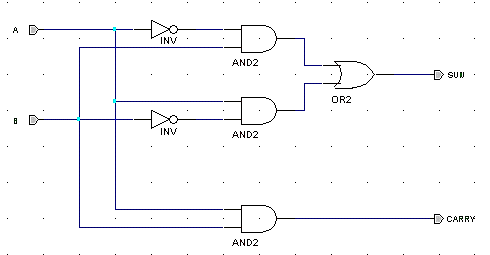
endmodule



1. **Lab Task 2: (2 marks)**

Write the Verilog Code using Gate Level modeling for the following circuit. List the code for design as well as stimulus below:

Simulate below circuit on Proteus and perform it on hardware.



module task2(sum, carry, a, b);

    input a, b;

    output sum, carry;

    wire w1, w2, w3, w4;

    not n1(w3, a);

    not n2(w4, b);

    and a1(w1, w3, b);

    and a2(w2, a, w4);

    or o1(sum, w1, w2);

    and a3(carry, a, b);

endmodule

module testbench;

    reg A, B;

    wire SUM, CARRY;

    task2 t2(SUM, CARRY, A, B);

    initial begin

        #100ps A = 0; B = 0;

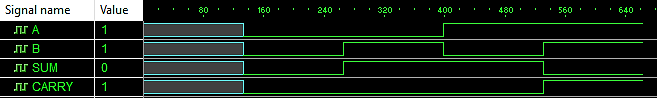
        #100ps A = 0; B = 1;

        #100ps A = 1; B = 0;

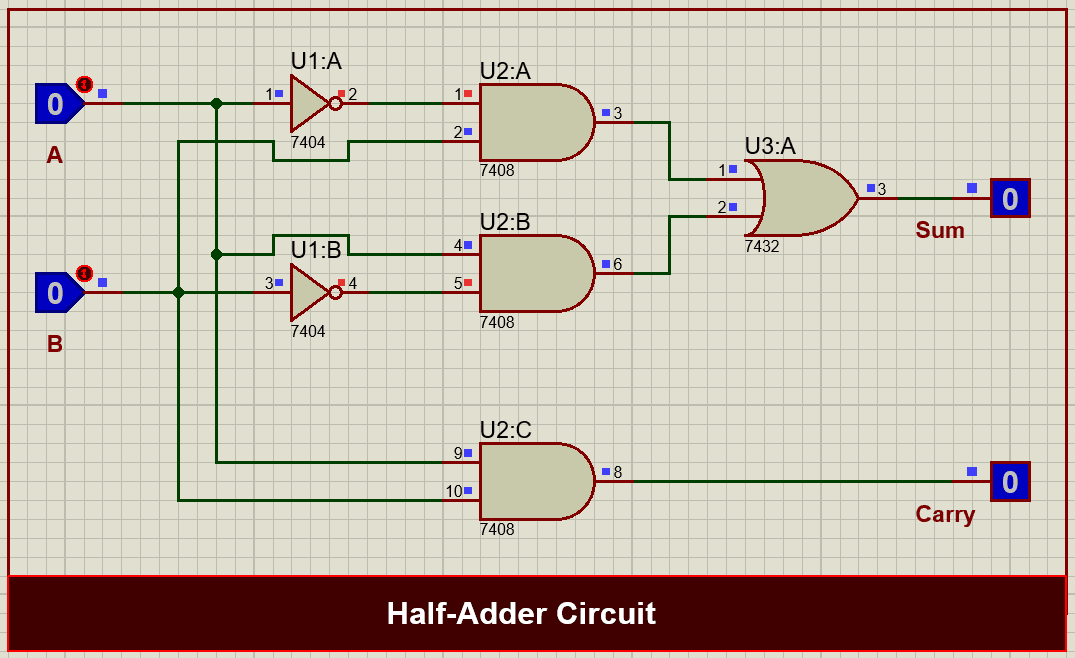
        #100ps A = 1; B = 1;

    end

endmodule

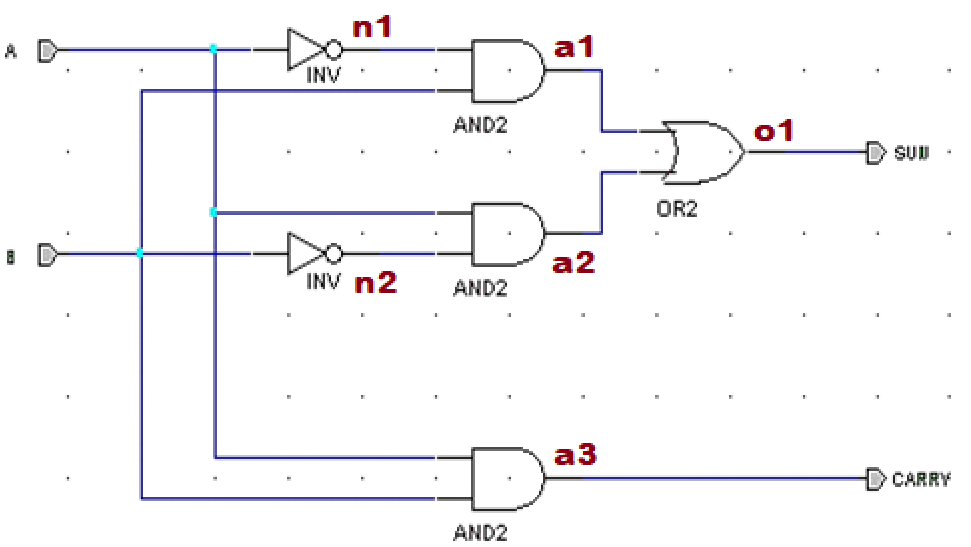


**Proteus Simulation**



1. **Lab Task 3: (2 marks)**

Label each gate output in the above circuit and derive algebraic expressions for SUM and Carry Out. Fill in the following truth table and determine the function performed by the circuit.



**Function Table**

|  |  |
| --- | --- |
| **Gates** | **Function** |
| n1 | ~A |
| n2 | ~B |
| a1 | ~A.B |
| a2 | A.~B |
| a3 | A.B |
| o1 | (~A.B) + (A.~B) |

* **Sum** = (~A.B) + (A.~B)
* **Carry Out** = A.B

**Truth Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **Sum** | **Carry Out** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

The given circuit is a half-adder circuit.

1. **Lab Task 4: (2 marks)**

After determining the function performed by the circuit given in Lab Task 2, write the Verilog description of the circuit in **dataflow**. Comment on the two different modeling levels you used to model the same circuit. (Paste snapshots of the codes and stimuluses below)

module halfadder(sum, carry, a, b);

    input a, b;

    output sum, carry;

    assign {carry, sum} = a + b;

endmodule

module testbench;

    reg A, B;

    wire SUM, CARRY;

    halfadder t4(SUM, CARRY, A, B);

    initial begin

        #100ps A = 0; B = 0;

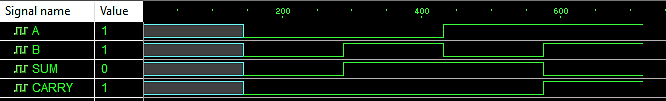
        #100ps A = 0; B = 1;

        #100ps A = 1; B = 0;

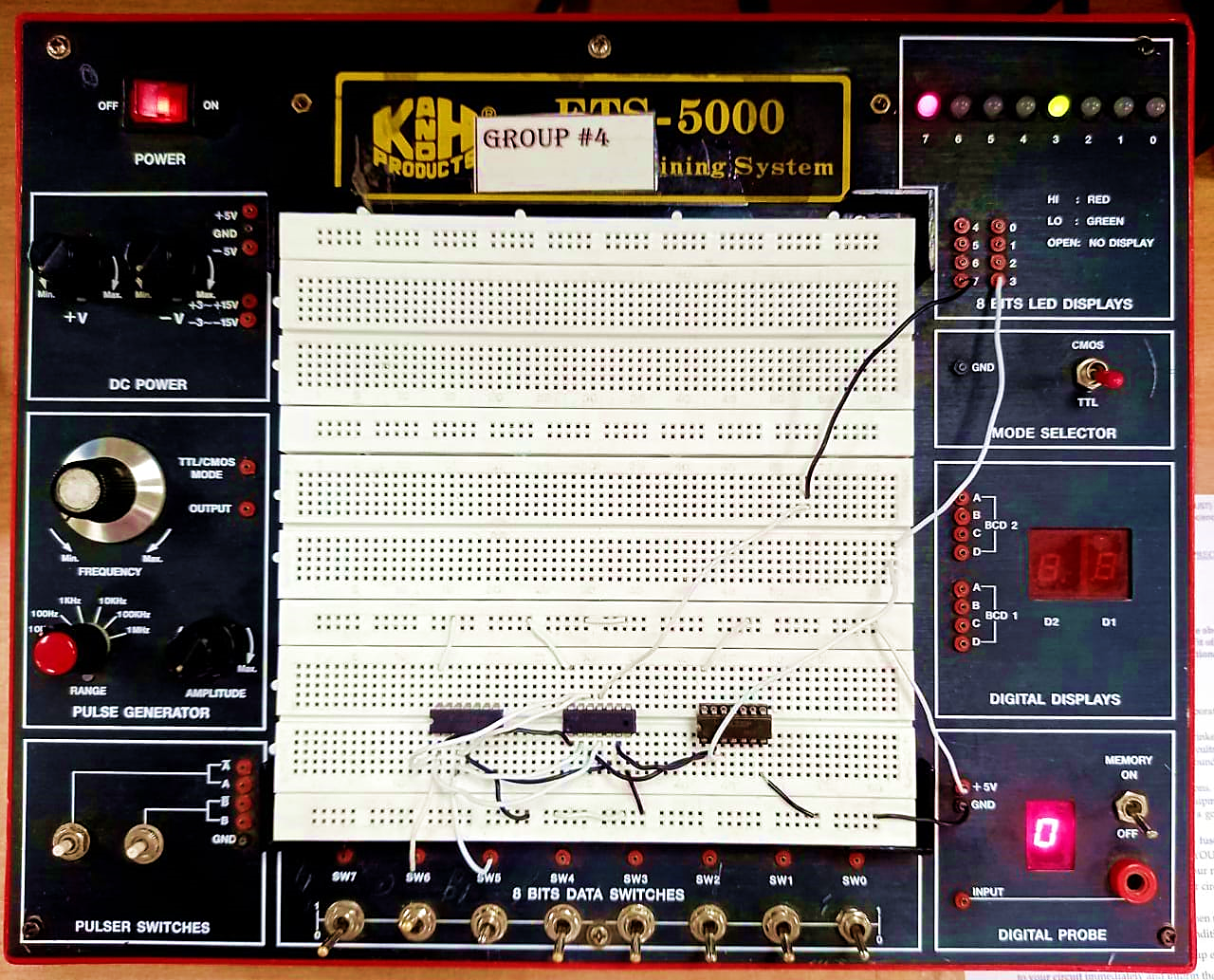
        #100ps A = 1; B = 1;

    end

endmodule



**Hardware Implementation**



The LEDs lit up according to the truth table filled above when passing different combinations of the switches SW5 and SW6, i.e., (0, 0), (0, 1), (1, 0), (1, 1). The LED at position 3 represents the **“SUM”** of the circuit whereas the LED at position 7 represents the **“CARRY OUT”**.

**Observations/Comments:**

We used gate-level and dataflow abstraction levels to write the code for a Half-Adder Circuit. In gate-level modeling, we used pre-defined gates to define the circuit. We just instantiated these gates in our modules. On the other hand, in data-flow modeling, we directly assigned the output i.e. {Carry, Sum} the required design using *“assign”* operator and we saw that there was no requirement to interconnect any gates to execute the Half-Adder Circuit.