**INTRODUCTION TO**

**VERILOG**

**LAB # 0****6**

**Fall 2023**

**CSE-304L**

**Computer Organization & Architecture Lab**

Submitted by: **AIMAL KHAN**

Registration No.: **21PWCSE1996**

Class Section: **A**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”



Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Dr. Bilal Habib**

Friday, November 10, 2023

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**ASSESSMENT RUBRICS COA LABS**

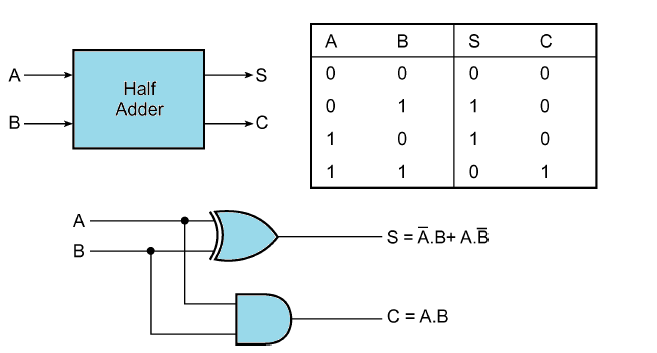
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LAB REPORT ASSESSMENT** | | | | |
| **Criteria** | **Excellent** | **Average** | **Nill** | **Marks Obtained** |
| 1. **Objectives of Lab** | All objectives of lab are properly covered  [Marks 10] | Objectives of lab are partially covered  [Marks 5] | Objectives of lab are not shown  [Marks 0] |  |
| 1. **MIPS instructions with**   **Comments and proper indentations.** | All the instructions are well written with comments explaining the code and properly indented  [Marks 20] | Some instructions are missing are poorly commented code  [Marks 10] | The instructions are not properly written  [Marks 0] |  |
| 1. **Simulation run without error and warnings** | The code is running in the simulator without any error and warnings  [Marks 10] | The code is running but with some warnings or errors.  [Marks 5] | The code is written but not running due to errors  [Marks 0] |  |
| 1. **Procedure** | All the instructions are written with proper procedure  [Marks 20] | Some steps are missing  [Marks 10] | steps are totally missing  [Marks 0] |  |
| 1. **OUTPUT** | Proper output of the code written in assembly  [Marks 20] | Some of the outputs are missing  [Marks 10] | No or wrong output  [Marks 0] |  |
| 1. **Conclusion** | Conclusion about the lab is shown and written  [Marks 20] | Conclusion about the lab is partially shown  [Marks 10] | Conclusion about the lab is not shown [Marks0]  [Marks 0] |  |
| 1. **Cheating** |  |  | Any kind of cheating will lead to 0 Marks |  |
| Total Marks Obtained: \_\_\_\_\_\_\_\_\_\_  Instructor Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | |

**Introduction to Verilog**

Objectives:

* Introduction to Verilog
* Introduction to hardware design, Model Sim and testing the hardware design.

Tasks:

**Task 1**: Implement half adder in Verilog using gate level modeling.

Block and circuit diagram of half adder.

**DUT Code:**

module HalfAdder(input1, input2, sum, carry);

input input1, input2;

output sum, carry;

xor xorGate(sum, input1, input2);

and andGate(carry, input1, input2);

endmodule;

**Test Code:**

module simulateHalfAdder();

reg input1, input2;

wire sum, carry;

HalfAdder ha(input1, input2, sum, carry);

initial

begin

$display("I1, I2, Sum, Carry");

input1 = 0;

input2 = 0;

#10

$display("%b, %b, %b, %b", input1, input2, sum, carry);

input1 = 0;

input2 = 1;

#10

$display("%b, %b, %b, %b", input1, input2, sum, carry);

input1 = 1;

input2 = 0;

#10

$display("%b, %b, %b, %b", input1, input2, sum, carry);

input1 = 1;

input2 = 1;

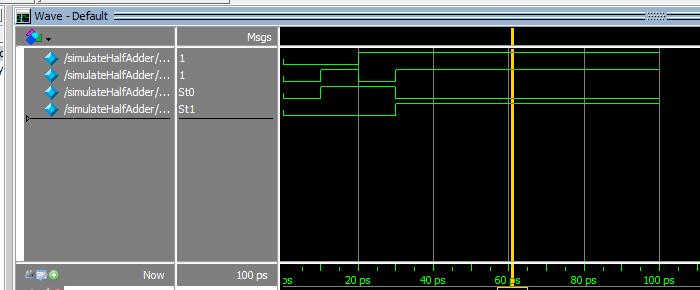
#10

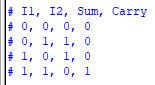
$display("%b, %b, %b, %b", input1, input2, sum, carry);

end

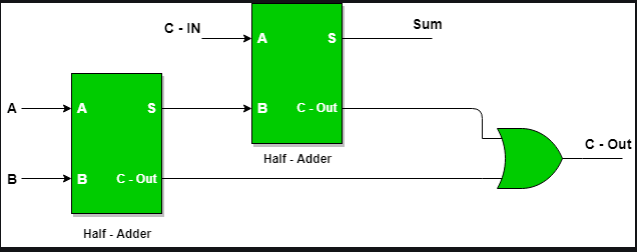
endmodule;

**Output:**

****

****

**Task 2**: Implement full adder using two half adder. *(use the above half adder to create full adder)*



**DUT Code:**

module FullAdder(input1, input2, carryIn, sum, carryOut);

input input1, input2, carryIn;

output sum, carryOut;

wire sum1, carry1, carry2;

HalfAdder ha1(input1, input2, sum1, carry1);

HalfAdder ha2(carryIn, sum1, sum, carry2);

or orGate(carryOut, carry1, carry2);

endmodule;

**Test Code:**

module simulateFullAdder();

reg input1, input2, carryIn;

wire sum, carryOut;

FullAdder fa(input1, input2, carryIn, sum, carryOut);

initial

begin

$display("I1, I2, Cin, Sum, Cout");

input1 = 0;

input2 = 0;

carryIn = 0;

#10

$display("%b, %b, %b, %b, %b", input1, input2, carryIn, sum, carryOut);

input1 = 1;

input2 = 0;

carryIn = 0;

#10

$display("%b, %b, %b, %b, %b", input1, input2, carryIn, sum, carryOut);

input1 = 0;

input2 = 1;

carryIn = 0;

#10

$display("%b, %b, %b, %b, %b", input1, input2, carryIn, sum, carryOut);

input1 = 1;

input2 = 1;

carryIn = 0;

#10

$display("%b, %b, %b, %b, %b", input1, input2, carryIn, sum, carryOut);

input1 = 0;

input2 = 0;

carryIn = 1;

#10

$display("%b, %b, %b, %b, %b", input1, input2, carryIn, sum, carryOut);

input1 = 1;

input2 = 0;

carryIn = 1;

#10

$display("%b, %b, %b, %b, %b", input1, input2, carryIn, sum, carryOut);

input1 = 0;

input2 = 1;

carryIn = 1;

#10

$display("%b, %b, %b, %b, %b", input1, input2, carryIn, sum, carryOut);

input1 = 1;

input2 = 1;

carryIn = 1;

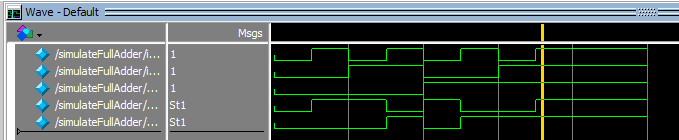
#10

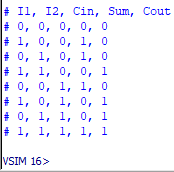
$display("%b, %b, %b, %b, %b", input1, input2, carryIn, sum, carryOut);

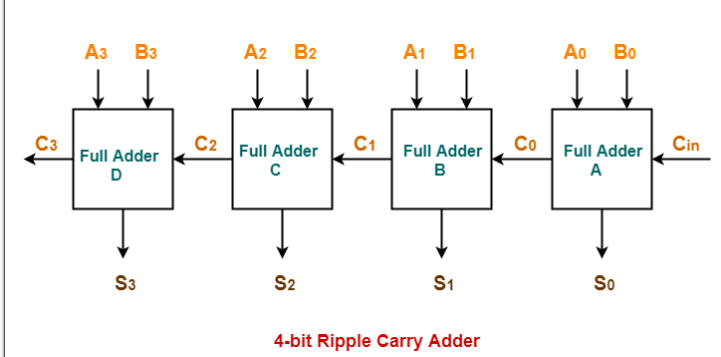
end

endmodule

**Output:**

****

****

**Task 3**: Write a Verilog code for 4 bit ripple carry adder.

**DUT Code:**

module FourBitAdder(A0, B0, A1, B1, A2, B2, A3, B3, Cin, S0, S1, S2, S3, Cout);

input A0, B0, A1, B1, A2, B2, A3, B3, Cin;

output S0, S1, S2, S3, Cout;

wire C0, C1, C2;

FullAdder faA(A0, B0, Cin, S0, C0);

FullAdder faB(A1, B1, C0, S1, C1);

FullAdder faC(A2, B2, C1, S2, C2);

FullAdder faD(A3, B3, C2, S3, Cout);

endmodule;

**Test Code:**

module simulateFourBitAdder();

reg A0, B0, A1, B1, A2, B2, A3, B3, Cin;

wire S0, S1, S2, S3, Cout;

FourBitAdder fba(A0, B0, A1, B1, A2, B2, A3, B3, Cin, S0, S1, S2, S3, Cout);

initial

begin

$display("A0, B0, A1, B1, A2, B2, A3, B3, Cin, S0, S1, S2, S3, Cout");

Cin = 0;

A0 = 0;

B0 = 0;

A1 = 0;

B1 = 0;

A2 = 0;

B2 = 0;

A3 = 0;

B3 = 0;

#10

$display("%b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b", A0, B0, A1, B1, A2, B2, A3, B3, Cin, S0, S1, S2, S3, Cout);

Cin = 1;

A0 = 0;

B0 = 0;

A1 = 0;

B1 = 0;

A2 = 0;

B2 = 1;

A3 = 0;

B3 = 0;

#10

$display("%b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b", A0, B0, A1, B1, A2, B2, A3, B3, Cin, S0, S1, S2, S3, Cout);

Cin = 1;

A0 = 1;

B0 = 0;

A1 = 1;

B1 = 0;

A2 = 1;

B2 = 0;

A3 = 1;

B3 = 0;

#10

$display("%b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b", A0, B0, A1, B1, A2, B2, A3, B3, Cin, S0, S1, S2, S3, Cout);

Cin = 0;

A0 = 1;

B0 = 1;

A1 = 1;

B1 = 0;

A2 = 1;

B2 = 1;

A3 = 1;

B3 = 0;

#10

$display("%b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b", A0, B0, A1, B1, A2, B2, A3, B3, Cin, S0, S1, S2, S3, Cout);

Cin = 0;

A0 = 0;

B0 = 0;

A1 = 1;

B1 = 1;

A2 = 1;

B2 = 0;

A3 = 1;

B3 = 1;

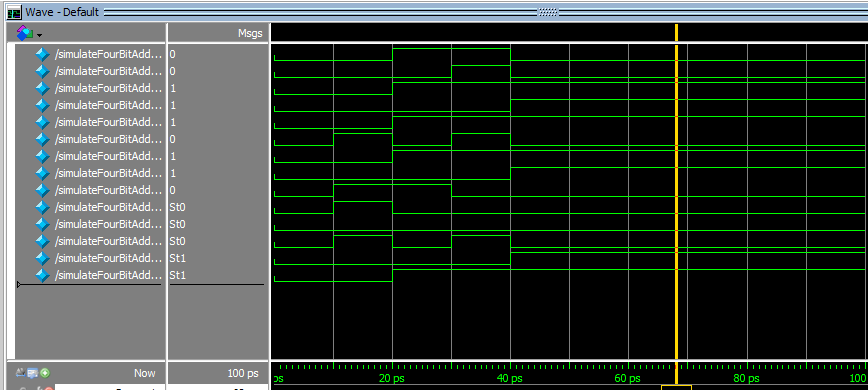
#10

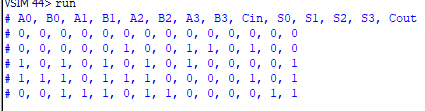
$display("%b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b, %b", A0, B0, A1, B1, A2, B2, A3, B3, Cin, S0, S1, S2, S3, Cout);

end

endmodule;

**Output:**

****

****

Reference:

To view my codes, please refer to my [GitHub Account.](https://github.com/aimalexe/DCSE/tree/main/semester_5_(fall-23)/computer_organization_and_architechure_lab/lab_reports/)

Conclusion:

In conclusion, I have learnt the basic of Verilog, model Sim, designing of hardware and the testing the hardware for its functionality. I can now create my own piece of hardware from basic gates and then use my hardware in designing other hardware components.

The End.