**Laboratory # 14**

**OPEN- ENDED LAB**

**Simulate Full Adder**

**Objective:**

Imagine a circuit in a lab with three inputs, A, B, and Cin, representing binary digits to be added. The full adder has two outputs, Sum (S) and Carry-out (Cout), which generate the result of the addition. Here's how it works:

* **Input A:** When A is set to 1, it represents the first binary digit to be added.
* **Input B:** When B is set to 1, it represents the second binary digit to be added.
* **Input Cin (Carry-in):** Cin represents any carry that might come from the previous addition stage.

Now, let's define the conditions for the outputs:

* **Sum (S):** S is 1 when the number of 1s among A, B, and Cin is odd (an odd parity function).
* **Carry-out (Cout):** Cout is 1 when two or more inputs among A, B, and Cin are 1 (a majority function).

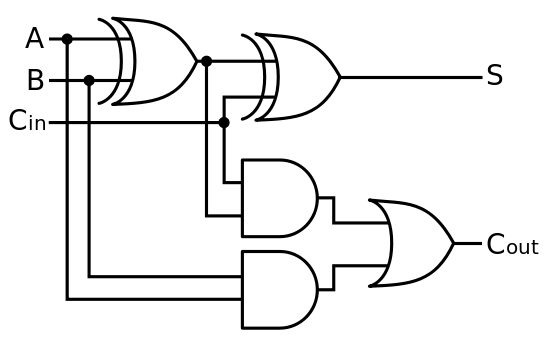
Trace and simulate a Full Adder using Verilog.

**Hardware/Software Required:**

IC 7486 (XOR gate), IC 7408 (AND gate), and IC 7432 (OR gate)

Verilog

**Diagram:**

****

*Full Adder Circuit*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **Cin** | **Cout** | **Sum** |
| 0 | 0 | 0 |  |  |
| 0 | 0 | 1 |  |  |
| 0 | 1 | 0 |  |  |
| 0 | 1 | 1 |  |  |
| 1 | 0 | 0 |  |  |
| 1 | 0 | 1 |  |  |
| 1 | 1 | 0 |  |  |
| 1 | 1 | 1 |  |  |

*Truth Table for Full Adder Circuit*

**Theory:**

The full adder accepts three inputs and generates a sum output and a carry output. The full adder must add the two input bits and a carry output of one half adder as a carry input for the other half adder.

**Steps to follow:**

**Step 1:**

* Write a code in Verilog module for Full Adder.

module fulladder (carry\_out, sum, a, b, carry\_in);

input a, b, carry\_in;

output carry\_out, sum;

wire c1, c2;

// First XOR gate

xor x1(sum, a, b);

// Second XOR gate with carry\_in

xor x2(sum, sum, carry\_in);

// First AND gate

and a1(c1, a, b);

// Second AND gate

and a2(c2, sum, carry\_in);

// OR gate for the final carry\_out

or o1(carry\_out, c1, c2);

endmodule

**Step 2:**

* Write a test bench code for Full Adder.

module fulladder\_tb();

reg a, b, carry\_in;

wire carry\_out, sum;

fulladder add1(carry\_out, sum, a, b, carry\_in);

initial

begin

a=0; b=0; carry\_in = 0;

#200;

a=0; b=1; carry\_in = 0;

#200;

a=1; b=0; carry\_in = 0;

#200;

a=1; b=1; carry\_in = 0;

#200;

// Test with carry\_in = 1

a=0; b=0; carry\_in = 1;

#200;

a=0; b=1; carry\_in = 1;

#200;

a=1; b=0; carry\_in = 1;

#200;

a=1; b=1; carry\_in = 1;

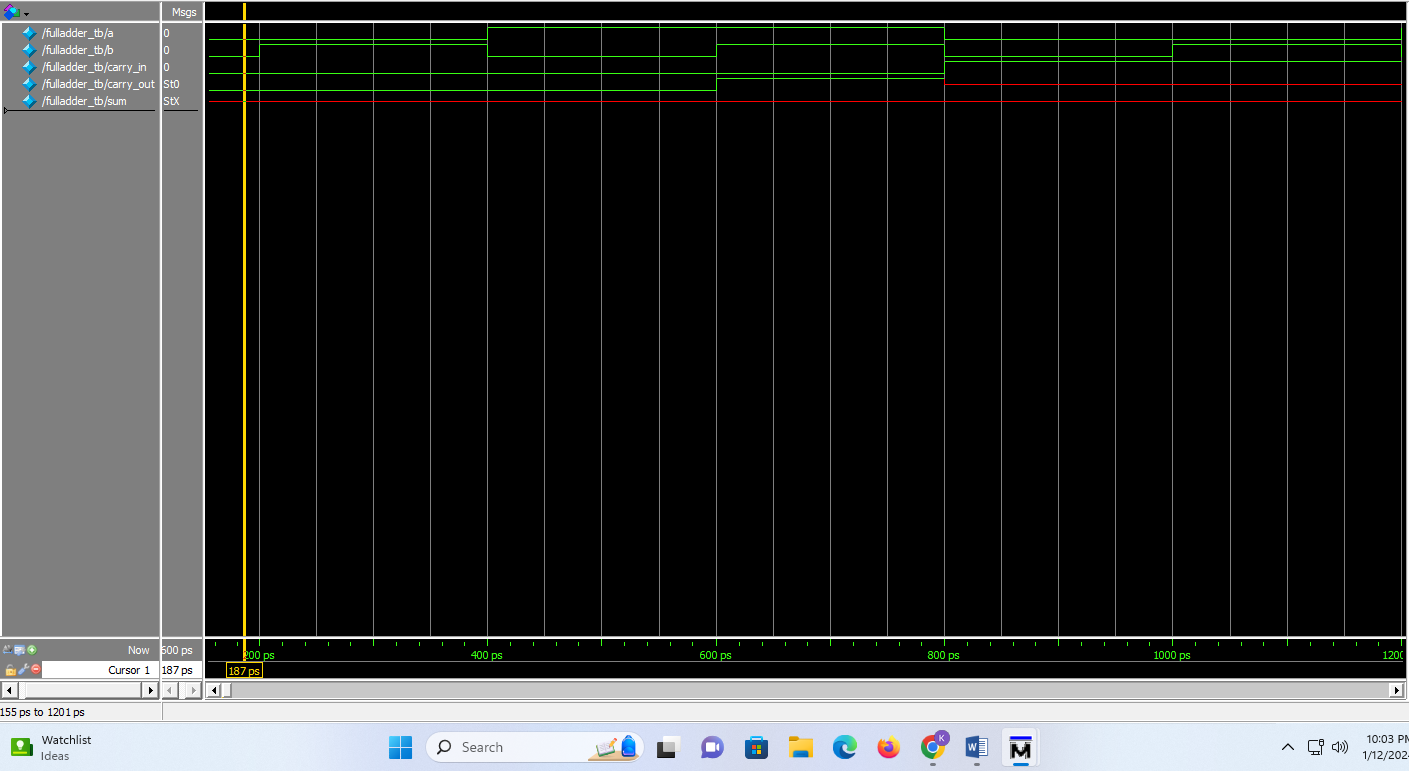
#200;

$finish;

end

endmodule

**Observation:**

****

**Conclusion:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Date: \_\_\_\_\_\_\_\_\_\_\_ Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_