Report for CDA 3201L Lab #4

Combinational Logic Design with Higher Level Devices

Names:

- Quang Anh Nguyen - U31803401

Introduction

In this experiment, a full adder circuit was designed and tested using both hardware and Verilog HDL. The full adder adds three 1-bit inputs (a, b, c_{in}) and produces two outputs (sum and cout). First, the truth table and Boolean expressions were derived and used to build the circuit with XOR and NAND gates on a breadboard. Then, the same circuit was implemented and simulated in Verilog to verify the results.

Methods and Materials

1) Materials

- Breadboard: A breadboard (Fig. 1) is a punctured plastic board used to construct circuit prototypes. The many holes on a breadboard allow us to insert the terminals of different components (i.e., resistors, capacitors, integrated circuits, etc.) and connect them as needed with the use of external wires (or wire jumpers)

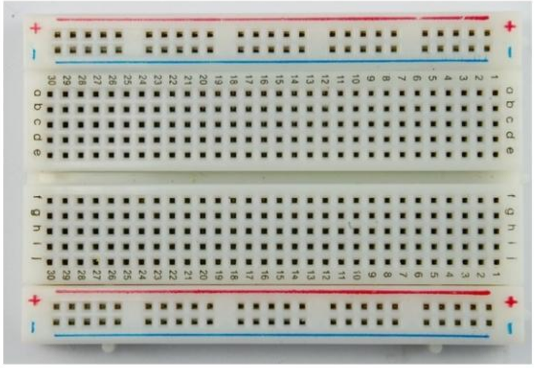
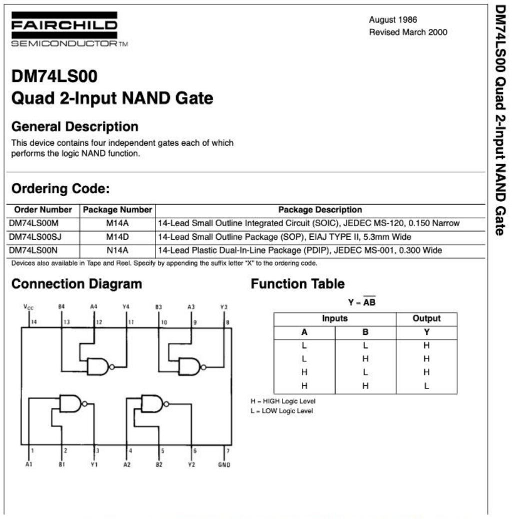


Figure 1 – Picture of a section of a breadboard

- Integrated Circuits (ICs): The 74LS86 IC was used to implement the XOR gates required for the sum output, and the 74LS00 IC was used to implement the NAND gates required for the carry output. The datasheets for these ICs were consulted to determine the pin configurations, number of gates per IC, and power supply connections.



FAIRCHILD SEMICONDUCTOR TM

August 1986
Revised March 2000

DM74LS00 Quad 2-Input NAND Gate

DM74LS00
Quad 2-Input NAND Gate

General Description

This device contains four independent gates each of which performs the logic NAND function.

Ordering Code:

Order Number	Package Number	Package Description
DM74LS00M	M14A	14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-120, 0.150 Narrow
DM74LS00SJ	M14D	14-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS00N	N14A	14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

Connection Diagram

Function Table

$Y = \overline{AB}$

Inputs		Output
A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

H = HIGH Logic Level
L = LOW Logic Level

Figure 2 - Example of a datasheet for the 74LS00 IC

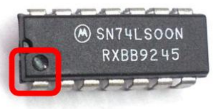
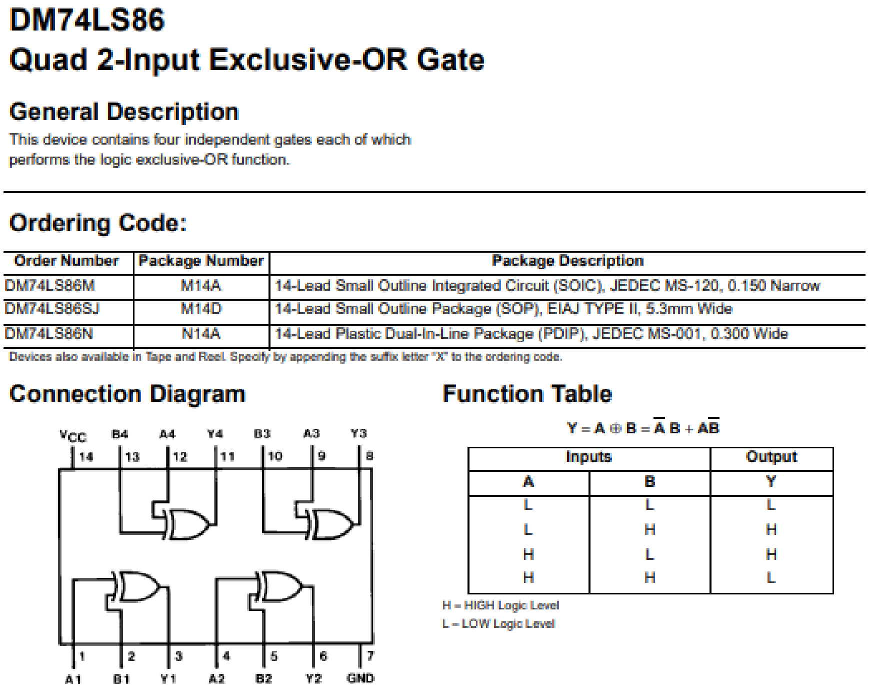


Figure 3 - A picture of a 74LS00 IC, where the first pin (engraved with a dot) is highlighted



DM74LS86
Quad 2-Input Exclusive-OR Gate

General Description

This device contains four independent gates each of which performs the logic exclusive-OR function.

Ordering Code:

Order Number	Package Number	Package Description
DM74LS86M	M14A	14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-120, 0.150 Narrow
DM74LS86SJ	M14D	14-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS86N	N14A	14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

Connection Diagram

Function Table

$Y = A \oplus B = \overline{A}\,B + A\overline{B}$

Inputs		Output
A	B	Y
L	L	L
L	H	H
H	L	H
H	H	L

H = HIGH Logic Level
L = LOW Logic Level

- Wiring: Wire jumper kit (Fig 4) consists of wires of different lengths intended to connect different components on a breadboard. You will also need to make connections from/to the power supply to/from the breadboard. For this purpose, you should use the alligator clips (Fig. 5) available in the lab.

Figure 4 - The wire jumper kit

Figure 5 – Alligator clips

- LEDs & Resistors: LEDs will be used to indicate the state of a circuit's output(s). When the logic state of the output connected to a LED is "1", the LED will emit light. On the other hand, the LED will stay off whenever the logic state of the output connected to it is "0". Importantly, a resistor (Fig. 7) should be connected in series with the LED (i.e., between its negative terminal and the ground node of the voltage supply. The resistor will act as a current limiter to prevent the LED from being accidentally burnt.

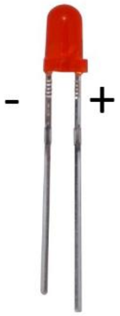


Figure 6 – A LED with correct polarity (long leg: +)

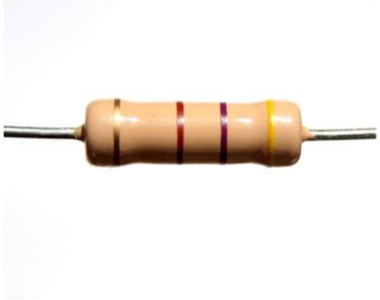
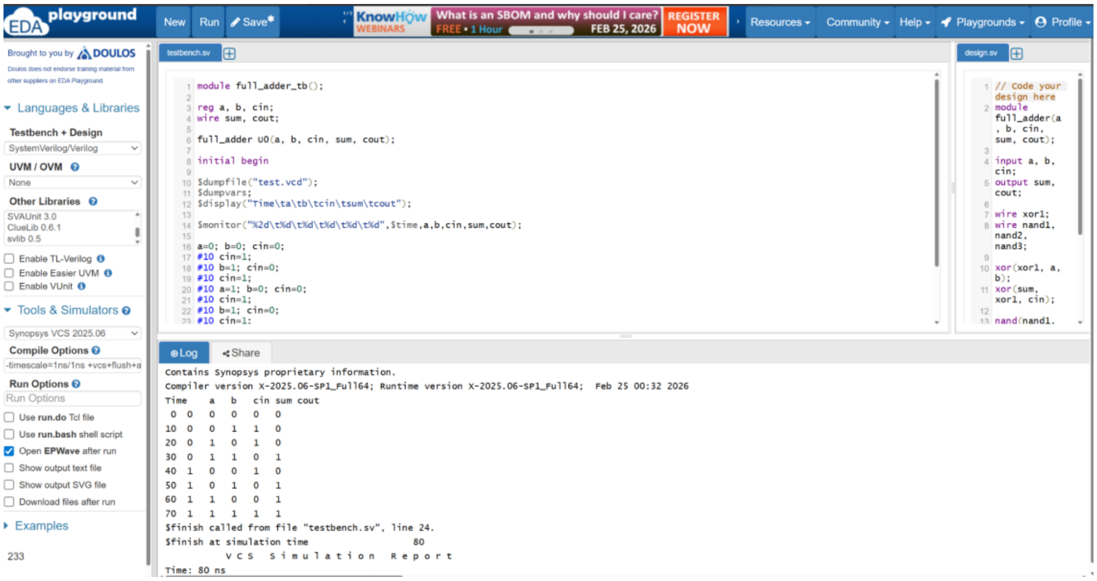


Figure 7 – A 470 ohm resistor

- EDA Playground: An online Verilog simulation tool used to write, run, and test Verilog code. It allows users to simulate digital circuits and view console output and waveform results to verify circuit functionality.



EDA playground
New
Run
Save*
KnowHow WEBINARS What is an SBOM and why should I care? FREE • 1 Hour FEB 25, 2026 REGISTER NOW
Resources
Community
Help
Playgrounds
Profile
Brought to you by DOULOS
Doulos does not endorse training material from other suppliers on EDA Playground.
Languages & Libraries
Tools & Simulators
Examples
233
testbench.sv
design.sv
Log
Share

Testbench + Design: SystemVerilog/Verilog
UVM / OVM: None
Other Libraries: SVAUnit 3.0, ClueLib 0.6.1, svlib 0.5
□ Enable TL-Verilog
□ Enable Easier UVM
□ Enable VUnit
Synopsys VCS 2025.06
Compile Options: -timescale=1ns/1ns +vcs+flush+a
Run Options
□ Use run.do Tcl file
□ Use run.bash shell script
☑ Open EPWave after run
□ Show output text file
□ Show output SVG file
□ Download files after run

testbench.sv

```
module full_adder_tb();

reg a, b, cin;
wire sum, cout;

full_adder U0(a, b, cin, sum, cout);

initial begin

$dumpfile("test.vcd");
$dumpvars;
$display("Time\ta\tb\tcin\tsum\tcout");

$monitor("%2d\t%d\t%d\t%d\t%d\t%d",$time,a,b,cin,sum,cout);

a=0; b=0; cin=0;
#10 cin=1;
#10 b=1; cin=0;
#10 cin=1;
#10 a=1; b=0; cin=0;
#10 cin=1;
#10 b=1; cin=0;
#10 cin=1;
```

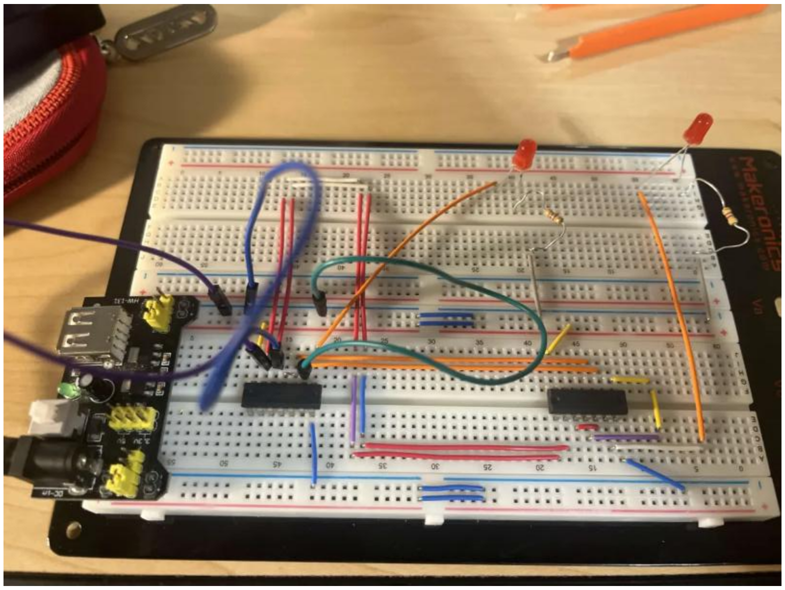
design.sv

```
// Code your design here
module full_adder(a, b, cin, sum, cout);

input a, b, cin;
output sum, cout;

wire xor1;
wire nand1, nand2, nand3;

xor(xor1, a, b);
xor(sum, xor1, cin);

nand(nand1,
```

```
Contains Synopsys proprietary information.
Compiler version X-2025.06-SP1_Full64; Runtime version X-2025.06-SP1_Full64;  Feb 25 00:32 2026
Time     a    b    cin  sum  cout
 0   0    0    0    0    0
10   0    0    1    1    0
20   0    1    0    1    0
30   0    1    1    0    1
40   1    0    0    1    0
50   1    0    1    0    1
60   1    1    0    0    1
70   1    1    1    1    1
$finish called from file "testbench.sv", line 24.
$finish at simulation time                    80
           V C S   S i m u l a t i o n   R e p o r t
Time: 80 ns
```

2) Methods:

- The truth table for the full adder was created using the three inputs (a, b, cin) and two outputs (sum, cout).
- A K-map was used to simplify the Boolean expressions for sum and c_{out}.
- The circuit was designed using XOR gates for the sum output and NAND gates for the cout output.
- The circuit was constructed on a breadboard using the 74LS86 XOR IC and 74LS00 NAND IC.
- LEDs were connected to the outputs to observe and verify the circuit functionality for all input combinations.
- The same full adder circuit was implemented in Verilog using structural modeling in EDA Playground.
- A testbench was written to apply different input combinations and simulate the circuit.
- The console output and waveform were observed to verify that the Verilog simulation matched the expected results from the hardware implementation.

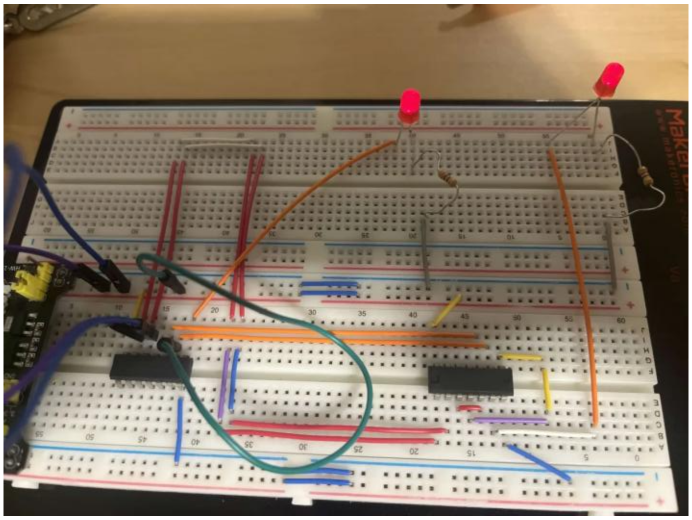


Circuit image

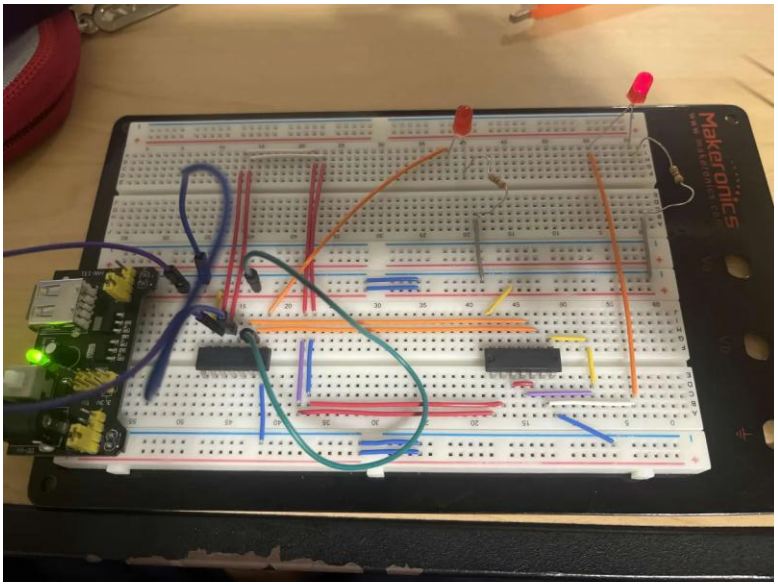
Results

The full adder circuit was successfully implemented on the breadboard using the 74LS86 XOR IC and 74LS00 NAND IC. LEDs were used to display the sum and c_{out} outputs for different input combinations of a, b, and c_{in}. The outputs matched the expected values from the truth table. The Verilog simulation in EDA Playground also produced console output and waveform results that were consistent with the hardware implementation.

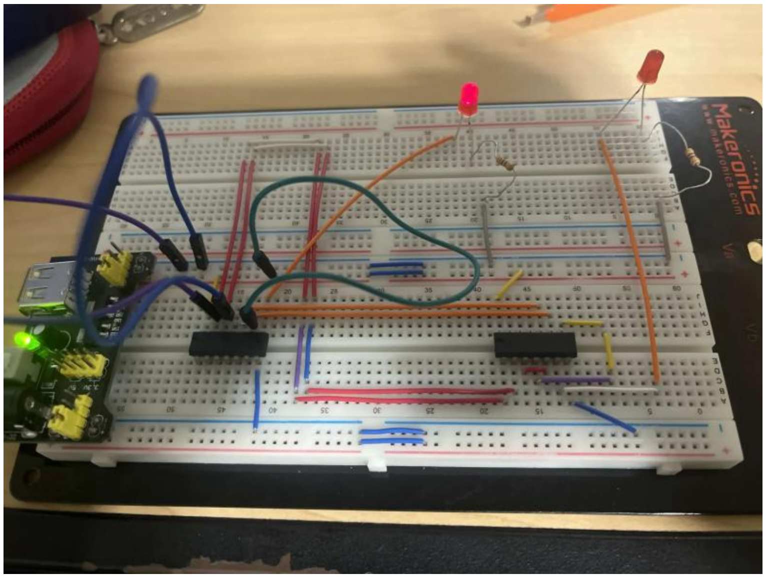
- Case a = 1, b = 1, c = 1:



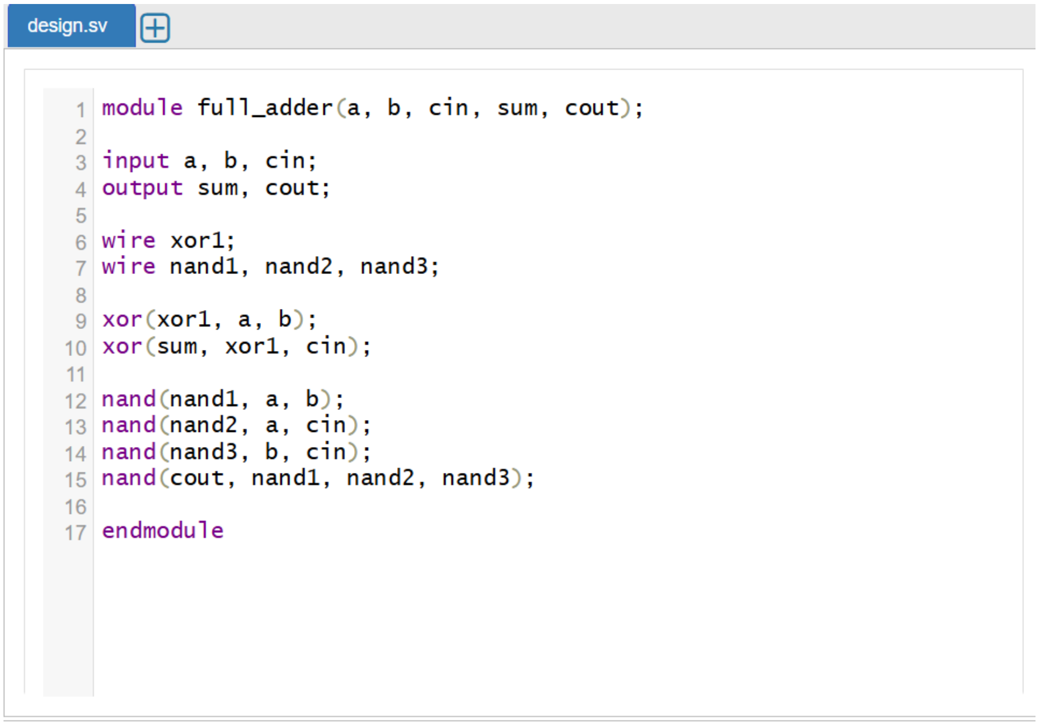
- Case a = 1, b = 0, c = 1:



- Case a = 0, b = 0, c = 1:

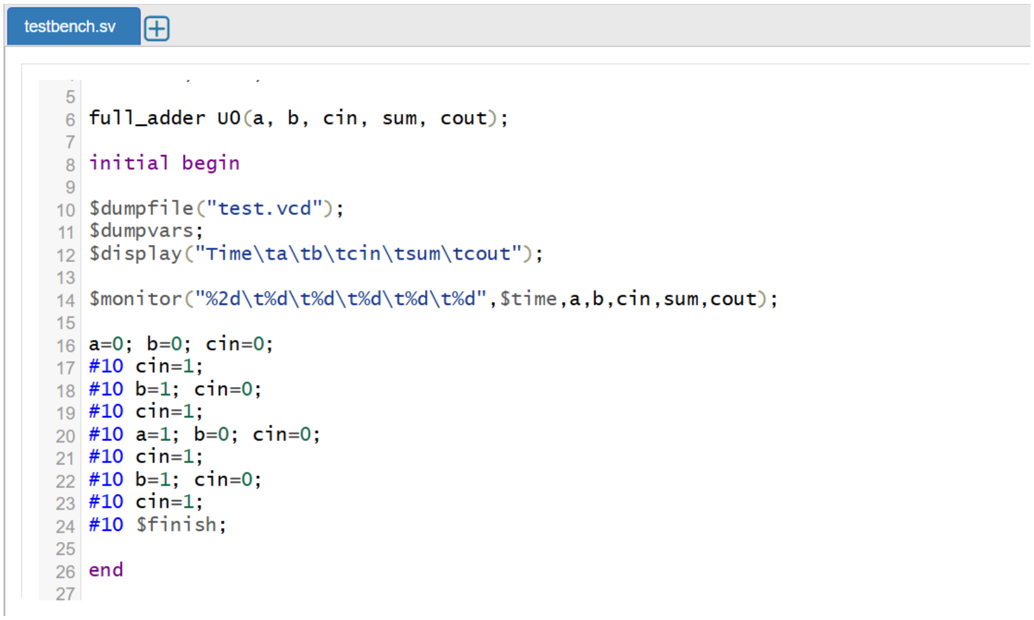


- Verilog Code Result and Waveform:



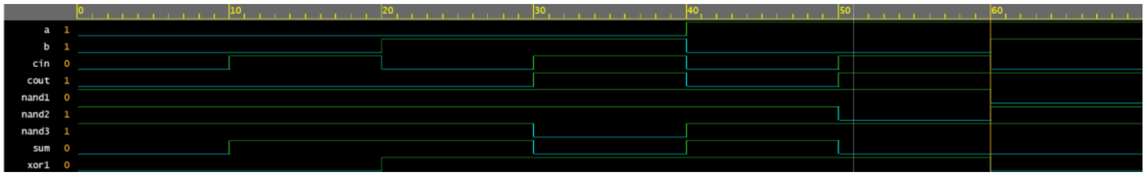
```
module full_adder(a, b, cin, sum, cout);

input a, b, cin;
output sum, cout;

wire xor1;
wire nand1, nand2, nand3;

xor(xor1, a, b);
xor(sum, xor1, cin);

nand(nand1, a, b);
nand(nand2, a, cin);
nand(nand3, b, cin);
nand(cout, nand1, nand2, nand3);

endmodule
```

Design code

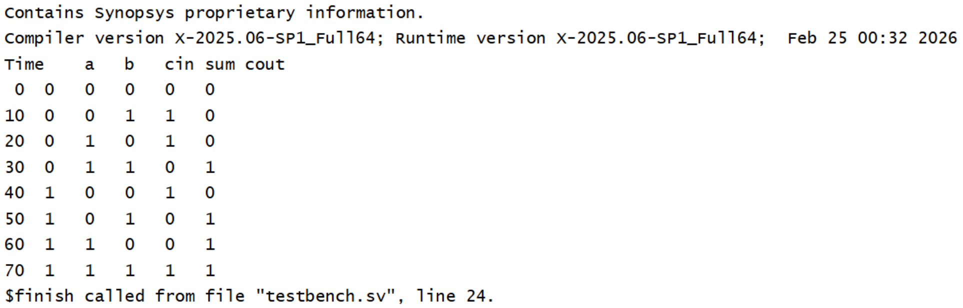


```

full_adder U0(a, b, cin, sum, cout);

initial begin

$dumpfile("test.vcd");
$dumpvars;
$display("Time\ta\tb\tcin\tsum\tcout");

$monitor("%2d\t%d\t%d\t%d\t%d\t%d",$time,a,b,cin,sum,cout);

a=0; b=0; cin=0;
#10 cin=1;
#10 b=1; cin=0;
#10 cin=1;
#10 a=1; b=0; cin=0;
#10 cin=1;
#10 b=1; cin=0;
#10 cin=1;
#10 $finish;

end

```

Testbench



Waveform



```
Contains Synopsys proprietary information.
Compiler version X-2025.06-SP1_Full64; Runtime version X-2025.06-SP1_Full64;  Feb 25 00:32 2026
Time    a   b   cin sum cout
 0  0   0   0   0   0
10  0   0   1   1   0
20  0   1   0   1   0
30  0   1   1   0   1
40  1   0   0   1   0
50  1   0   1   0   1
60  1   1   0   0   1
70  1   1   1   1   1
$finish called from file "testbench.sv", line 24.
```

Result

Lab Assignment Question

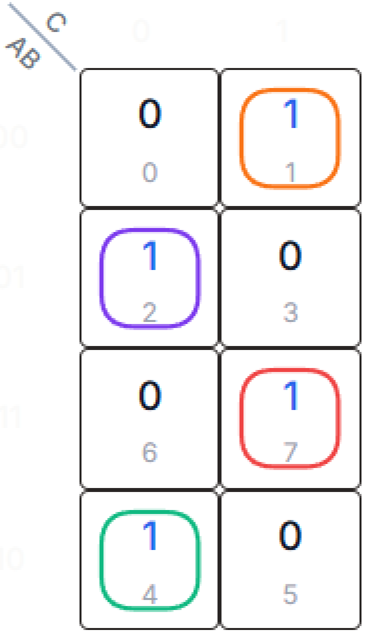
<u>Q1</u>:

Truth table:

a	b	c_{in}	Sum	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

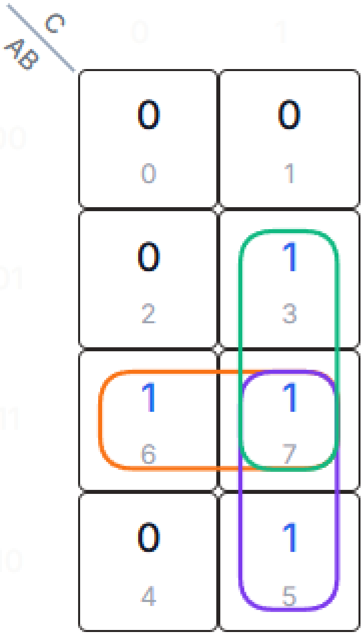
K-maps:

- Sum: Minterms f = Σm(1,2,4,7). MSOP = $\bar{a}\bar{b}c_{in} + \bar{a}b\overline{c_{in}} + a\bar{b}\overline{c_{in}} + abc_{in}$



AB \ C	0	1
00	0 0	1 1
01	1 2	0 3
11	0 6	1 7
10	1 4	0 5

- C$_{out}$: Minterms f = Σm(3,5,6,7). MSOP = $ab + bc_{in} + ac_{in}$



AB \ C	0	1
00	0 0	0 1
01	0 2	1 3
11	1 6	1 7
10	0 4	1 5

__Q2__: Sum: MSOP = $\bar{a}\bar{b}c_{in} + \bar{a}b\overline{c_{in}} + a\bar{b}\overline{c_{in}} + abc_{in} = c_{in}\left(\bar{a}\bar{b} + ab\right) + \overline{c_{in}}\left(a\bar{b} + \bar{a}b\right)$

$= c_{in}\overline{(a \oplus b)} + \overline{c_{in}}\,(a \oplus b)$

$= a \oplus b \oplus c_{in}$

C$_{out}$ = $\overline{\overline{ab}\,.\,\overline{ac_{in}}.\,\overline{bc_{in}}} = \overline{\overline{\overline{\overline{ab}\,.\,\overline{ac_{in}}}}.\,\overline{bc_{in}}}$

<u>Q3</u>:

c_{out} = 1 when at least two inputs are 1:

a	b	c_{in}
0	1	1
1	0	1
1	1	0
1	1	1

From truth table analysis, carry occurs when both a and b are 1, or one of (a XOR b) and $c_{in}$is 1

⇨ C$_{out}$ = $(a \oplus b).c_{in} + ab = \overline{\overline{(a \oplus b).c_{in}}.\overline{ab}}$ (1 XOR gate and 3 NAND gates)

Discussion

- The experimental results matched the expected truth table of a full adder. The sum output correctly represented the XOR of the three inputs, and the carry-out (cout) output correctly indicated a carry when at least two inputs were 1. The breadboard implementation functioned properly, and the LEDs displayed the correct output combinations for all input cases. Additionally, the Verilog simulation produced the expected console output and waveform results. These findings confirm that both the hardware implementation and the Verilog design were correct and functionally equivalent.
- A minor difficulty was encountered during the wiring process. Without a clear visualization, optimizing the circuit in Part B was a little challenging, particularly when simplifying and arranging the connections on the breadboard.

Conclusion

In this lab, a full adder circuit was successfully designed, implemented, and verified using both hardware and Verilog HDL. The truth table and Boolean expressions were derived using Karnaugh maps, and the circuit was implemented using XOR and NAND gates. The breadboard implementation and Verilog simulation both produced correct outputs for all input combinations. This lab helped reinforce the understanding of combinational logic circuits, Boolean algebra, and Verilog structural modeling.