Digital Logic Design Laboratory

Lab 5

Demultiplexers

Full name: Nguyễn Đình Ngọc Huy-EEEEIU22020

Lê Quang Thông- EEACIU22222

Student number: 2

Class: 5

Date: 17/12/2023

GRADING GUIDELINE FOR LAB REPORT

Number	Content		Score	Comment
1	**Format (max 9%)**			
	• Font type	Yes No		
	• Font size	Yes No		
	• Lab title	Yes No		
	• Page number	Yes No		
	• Table of contents	Yes No		
	• Header/Footer	Yes No		
	• List of figures (if exists)	Yes No		
	• List of tables (if exists)	Yes No		
	• Lab report structure	Yes No		
2	**English Grammar and Spelling (max 6%)**			
	• Grammar	Yes No		
	• Spelling	Yes No		
3	**Data and Result Analysis (max 85%)**			
Total Score				

Signature:

Date:

I. Objectives

In this laboratory, students will study:

- Understand and design a multiplexer.

- Use a demultiplexer and design/implement a circuit based on a function definition.
- Design combinational circuits using DEMUX.

II. Procedure

1. Design demultiplexer using logic gates

a. Design 1-to-2 demultiplexer using logic gates:

A 1-to-2 demultiplexer has I is the input, S is the selector input, and Y_1 and Y_2 are two outputs. When S = 0 then Y_0 = I but when S = 1 then Y_1 = I. The Figure 1 shows the illustration of DEMUX 1-2.

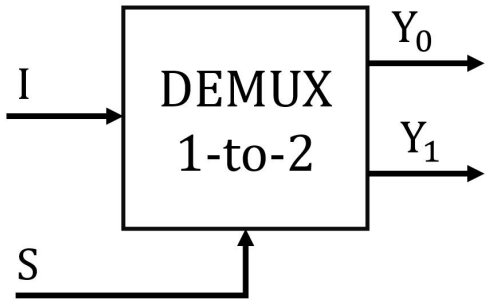


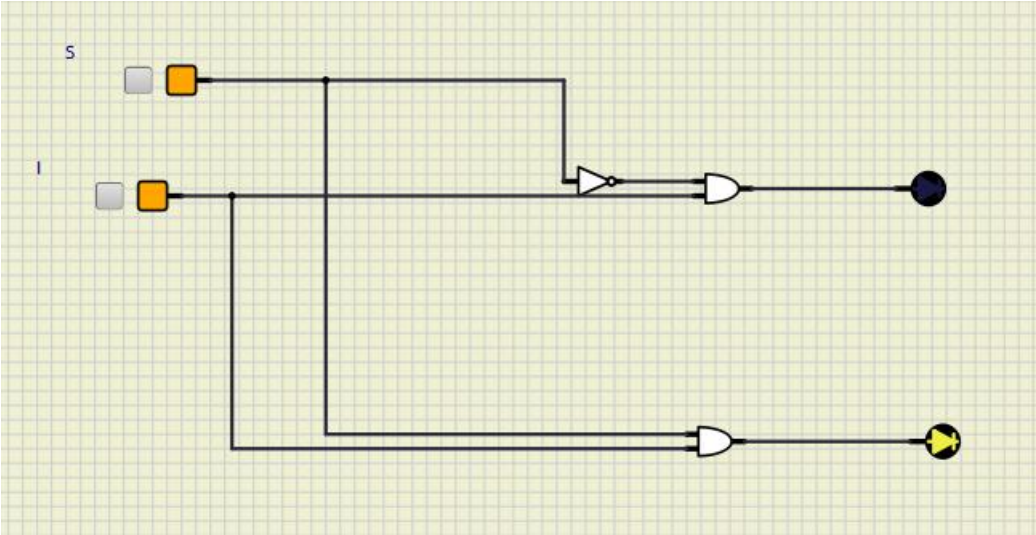
Figure 1. The illustration of DEMUX 1-2.

Built the truth table:

Input		Output	
S	**I**	**Y_0**	**Y_1**
0	**0**	**0**	**0**
0	**1**	**1**	**0**
1	**0**	**0**	**0**
1	**1**	**0**	**1**

The expressions: $Y0=\overline{S}.I$; $Y1=S.I$

Implement the circuit via simulation software and paste the result in here



Make comment on the results:

- When the input S is low I=Y0=0

-When the input S is low I=Y0=1

-When the input S is high I=Y1=0

-When the input S is high I=Y1=1

b. Design 1-to-4 DEMUX using logic gates.

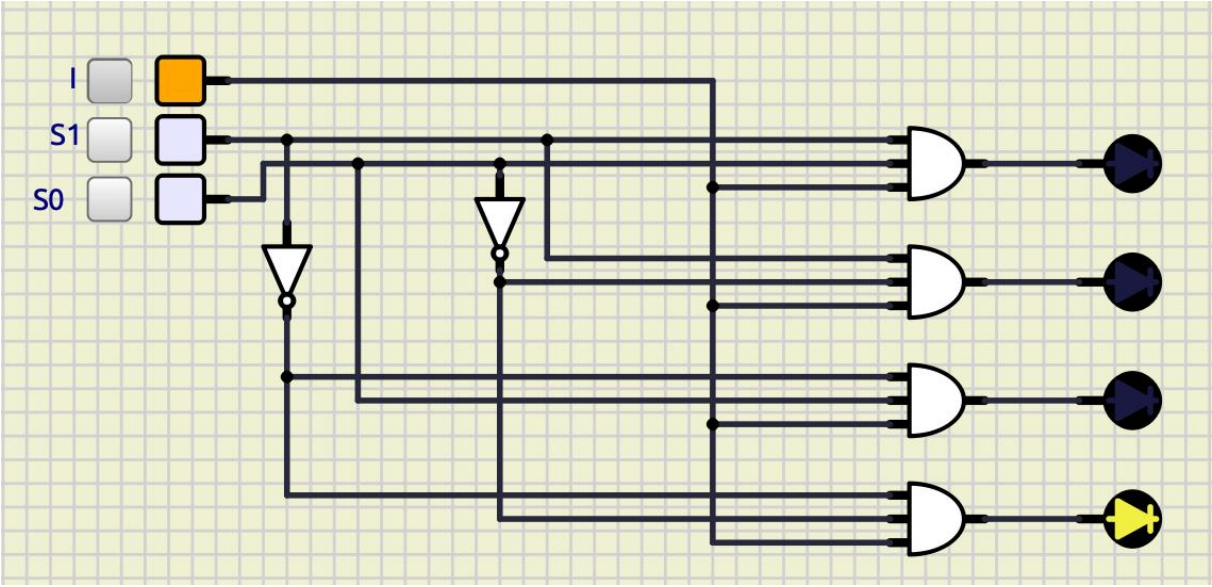
Build the circuit. The inputs S0, S1, I, are driven by 6 switches. The outputs Y0, Y1, Y2, Y3 are connected to LED.

Input			Output			
S_1	S_0	I	Y_0	Y_1	Y_2	Y_3
0	0	0	**0**	0	0	0
0	0	1	**1**	0	0	0
0	1	0	0	**0**	0	0
0	1	1	0	**1**	0	0
1	0	0	0	0	**0**	0
1	0	1	0	0	**1**	0
1	1	0	0	0	0	**0**
1	1	1	0	0	0	**1**

Table 1

The expressions: 0= $.\overline{S1}.\overline{S2}$; 1= $.\overline{S1}$. 0 ; $Y2=I.S1.\overline{S0}$; $Y3=I.S1.S0$

Implement the circuit via simulation software and paste the result in here



Make comment on the results:

-When the input S1=0 ; S0=0 and I=0, so Y0=0

-When the input S1=0 ; S0=0 and I=1, so Y0=1

-When the input S1=0 ; S0=1 and I=0, so Y1=0

-When the input S1=0 ; S0=1 and I=1, so Y1=1

-When the input S1=1 ; S0=0 and I=0, so Y2=0

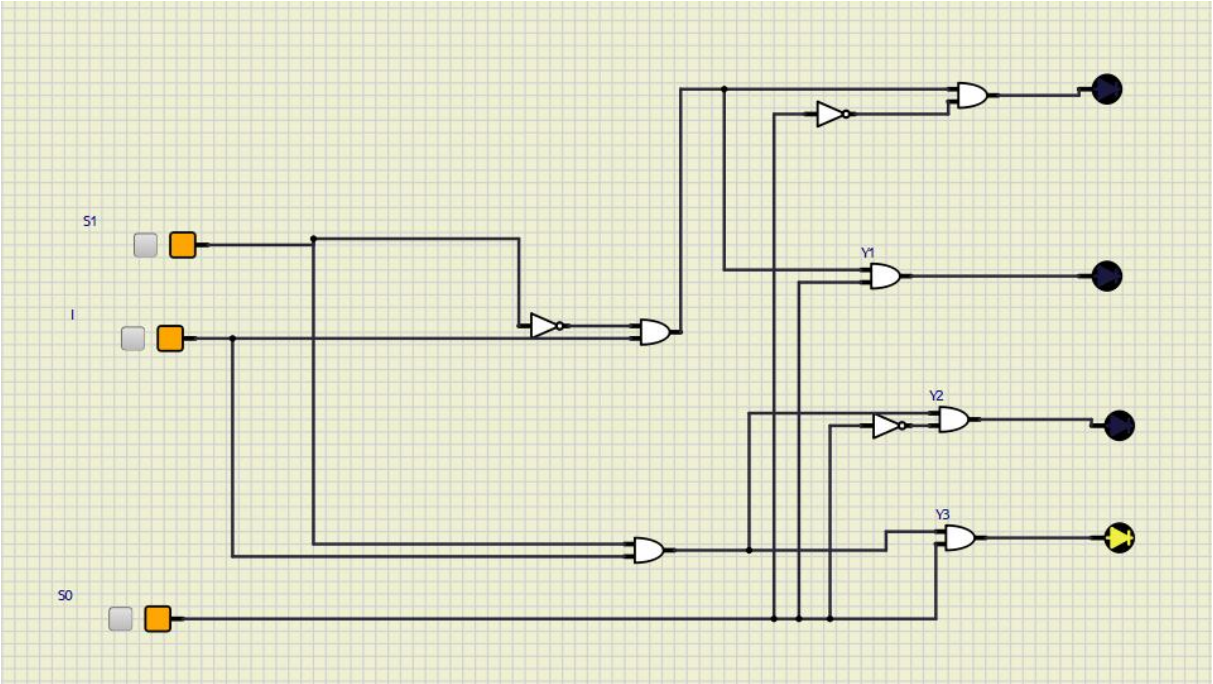
-When the input S1=1 ; S0=0 and I=1, so Y2=1

-When the input S1=1 ; S0=1 and I=0, so Y3=0

-When the input S1=1 ; S0=1 and I=1, so Y3=1

c. Design 1-to-4 DEMUX using 3 DEMUX 1-2.

Implement the circuit via simulation software and paste the result in here



Make comment on the results:

The provided truth table demonstrates a 1-to-4 DEMUX's functionality, and it corresponds to a 1-to-4 DEMUX implemented with logic gates. This is achieved by using three 1-to-2 DEMUXes. The first DEMUX accepts the most significant bit of the selection input, with its outputs feeding the other two DEMUXes. These DEMUXes then process the least significant selection bit. The combination of their outputs forms the 1-to-4 DEMUX output lines, routing the input data depending on the selection inputs.

2. Investigate IC 1-to-8 DeMultiplexer (74HC238)

Construct the circuit as below:

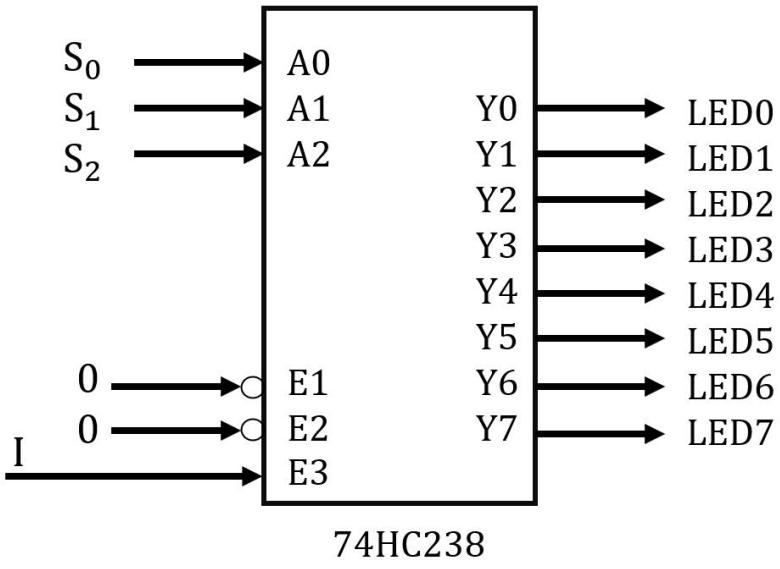


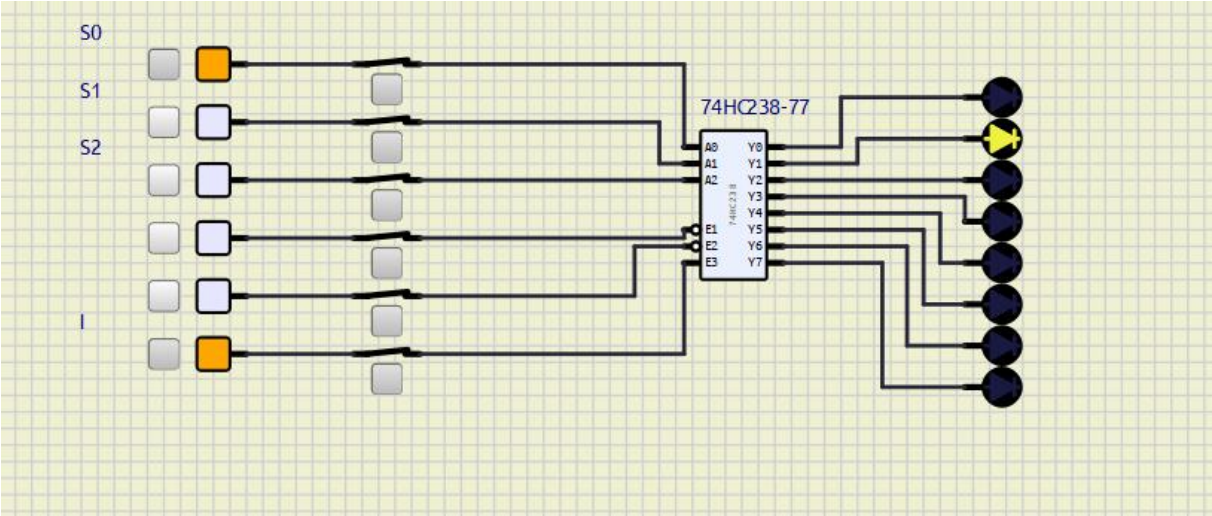
Figure 2. IC 1-to-8 DeMultiplexer (74HC238)

- 8 outputs are connected by using LEDs.
- The inputs are controlled by switches.

- Observe the results and fulfill the truth table

INPUT			OUTPUT							
S2	**S1**	**S0**	**Y0**	**Y1**	**Y2**	**Y3**	**Y4**	**Y5**	**Y6**	**Y7**
0	0	0	I	0	0	0	0	0	0	0
0	0	1	0	I	0	0	0	0	0	0
0	1	0	0	0	I	0	0	0	0	0
0	1	1	0	0	0	I	0	0	0	0
1	0	0	0	0	0	0	I	0	0	0
1	0	1	0	0	0	0	0	I	0	0
1	1	0	0	0	0	0	0	0	I	0
1	1	1	0	0	0	0	0	0	0	I

Implement the circuit via simulation software and paste the result in here



Briefly describe the operation of the IC:

The 74HC238 1-to-8 Demultiplexer IC uses three binary inputs (S2, S1, S0) to choose and activate one of its eight outputs (Y0 - Y7). Each combination of binary inputs corresponds to one active output, with all other outputs remaining inactive. This allows the IC to route a single input signal to one of eight distinct output lines.

3. Design 1-bit Full Subtractor

a. Using logic gates

Construct the circuit as below:

Three inputs are A, B, Bin. Two outputs are D and Bout.

Build the truth table and the expressions

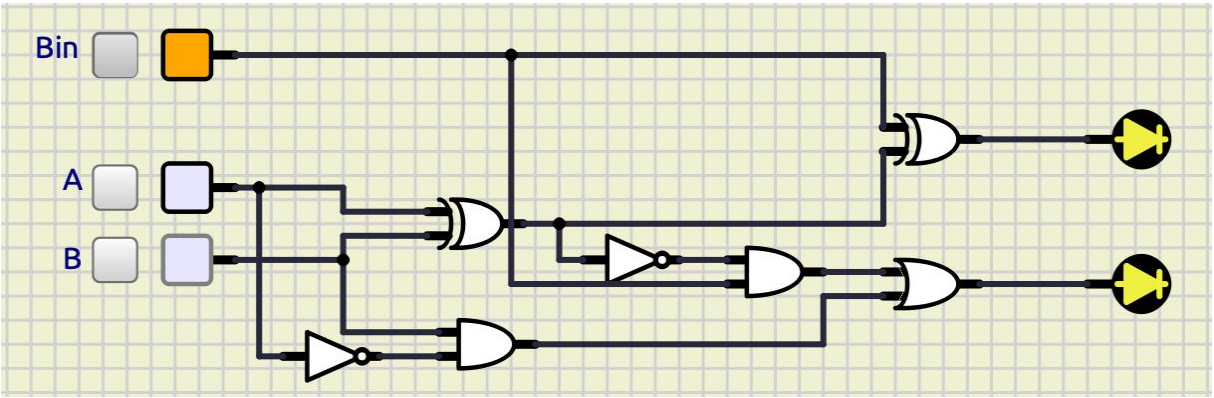
Input			Output	
A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

The simplified expressions:

$$D = \sum m(1, 2, 4, 7)$$

$$B_{out} = f(A, B, B_{in}) = \sum m(1, 2, 3, 7)$$

Implement the circuit via simulation software and paste the result in here



Make comment on the results

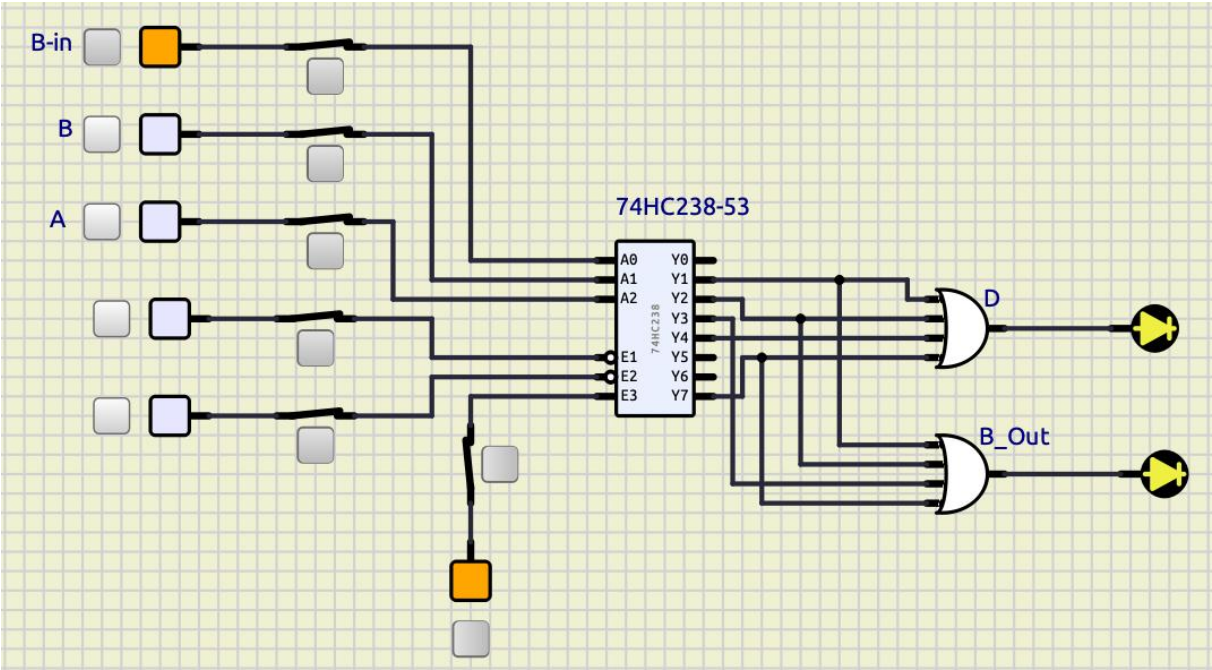
- The Difference output, D, is true (1) when the number of 1s in the inputs is odd (A, B, Bin = 1,2,4,7).

- The Borrow out output, B_out, is true (1) when at least two of the inputs are 1 (A, B, Bin = 1,2,3,7).

The Full Subtractor performs subtraction on bits, including borrowing. From the truth table and the expressions given, it's evident that the Full Subtractor is working as expected. The Borrow out is high when at least two inputs are high which signifies a borrow in binary subtraction. The Difference output is high when the number of 1's in the inputs is odd which is a characteristic of XOR function, a crucial function in binary subtraction.

b. 1-to-8 DeMultiplexer (74HC238)

Implement the circuit via simulation software and paste the result in here



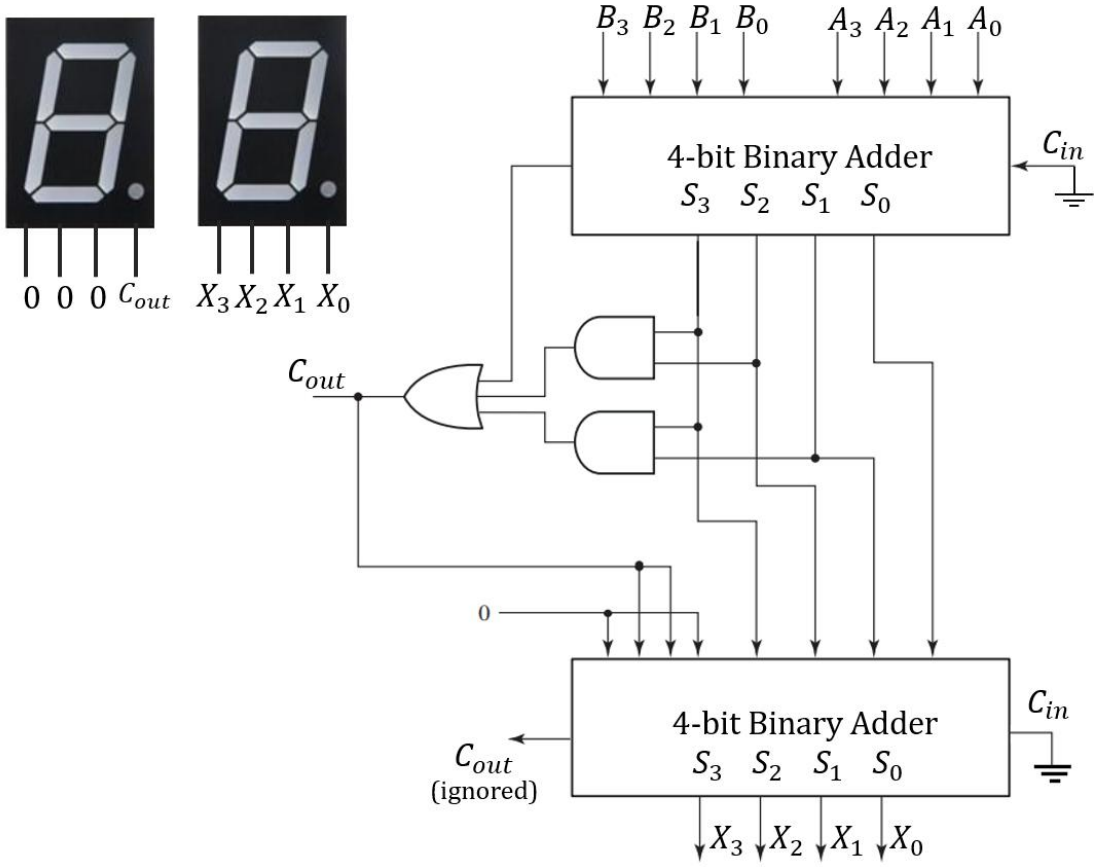
Make comment on the results

A 1-bit Full Subtractor can be designed using a 1-to-8 Demultiplexer (74HC238) by using the subtractor inputs (A, B, Bin) as the demultiplexer's selection inputs. The eight demultiplexer outputs (Y0 - Y7) are then mapped to represent the possible states of the subtractor's outputs (D, Bout). This design process involves careful mapping of logic functions to achieve the correct subtractor outputs based on the input combinations.

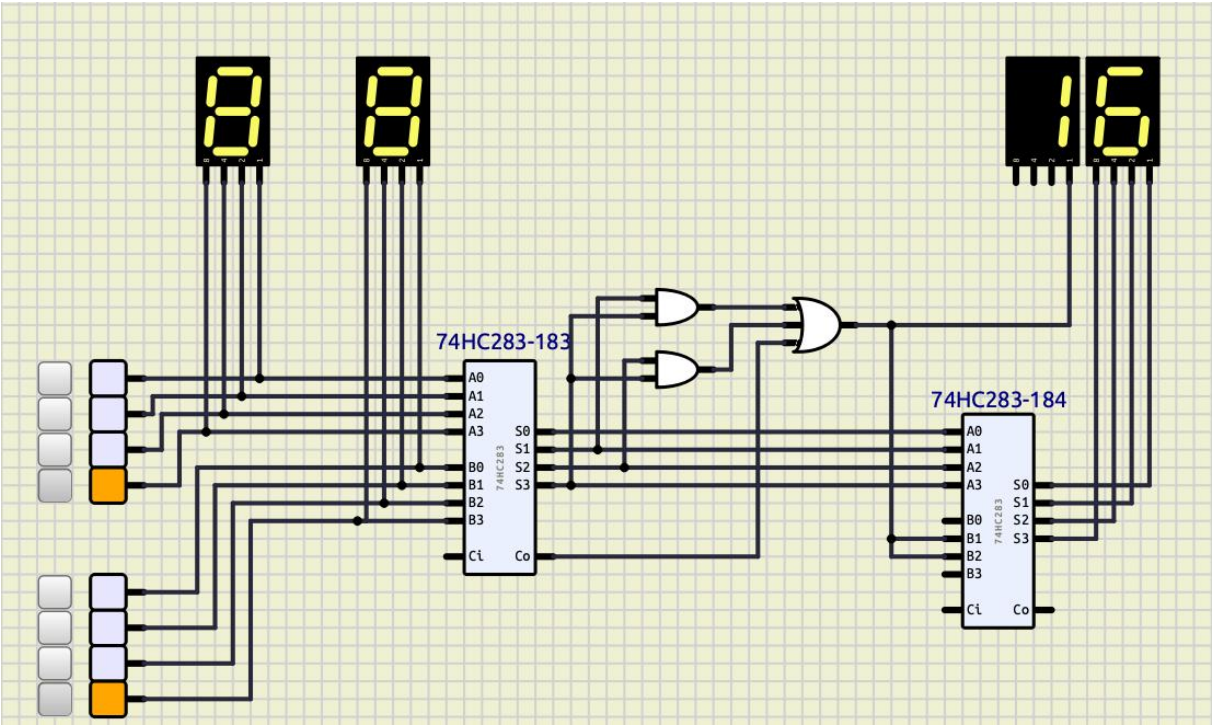
4. Design 4-bit Full Adder using 74HC283 and display to BCD Seg

Construct the circuit as below:

Four inputs for A(A3, A2, A1, A0) and B(B3, B2, B1, B0). The outputs are display by BCD 7seg



Implement the circuit via simulation software and paste the result in here



Make comment on the results

The 4-bit full adder is designed using the 74HC283 IC, which adds two 4-bit binary numbers and outputs the sum and a carry-out. The full adder works on the principle of cascading, i.e., the carry-out of the lower bit is the carry-in for the next higher bit. The first carry in is grounded as it is the least significant bit.

The outputs of the full adder are in binary form which can't be directly displayed on the 7-segment display. Hence, a Binary to BCD converter is used to change the binary output to BCD format. The BCD format is easier to display on a 7-segment display as each digit of a decimal number is represented separately.

The BCD output is then fed into a BCD to 7-segment decoder/driver like the 74HC47. This IC takes the BCD input and converts it into signals that can drive the 7-segment display. The 7-segment display lights up in the shape of the decimal digit it represents.

It's important to note that a standard 4-bit full adder can add numbers up to 15 (1111 in binary). If the circuit is displaying numbers up to 16, it might be interpreting the carry-out as a fifth bit. This could be an expected behavior if the circuit is designed to handle 5-bit inputs, or could indicate a need for adjustments in the circuit design.

In conclusion, the 4-bit Full Adder using a 74HC283 IC and BCD 7-segment display effectively demonstrates the principle of digital addition and the conversion of binary numbers to a human-readable decimal format.