DIGITAL ELECTRONICS AND COMPUTER ORGANIZATION LABORATORY

Dr. Anjad Badran

Experiment No. 2 - Comparators, Adders and Subtractors

Section 11

Islam Jihad Joma 1191375

T. Enas Jawabrah

1-3-2021

Abstract

In this experiment we have to build half subtractor and adder and full adder and subtractor and comparators with different number of bits for all circuits, we have to use both basic gates and some designed blocks, we have to simulate them via Proteus and find the truth table for each circuit.

Table of content:

We need some pieces but as we are in online learning so all we need is proteus and his software of different pieces.

(and, or, nand, xor, not) gates

(74LS85,74LS83) IC’s

Wires as well

Introduction (Theory)

In Comparator Circuit At least two numbers are required to perform any comparison. The simplest form of the comparator has two inputs. If the two inputs are called A and B, there are three possible outputs: A>B, A=B, and A<B, we use it to create a 4-bit comparator which is most used.

Digital computers perform a variety of information processing tasks. Among the functions encountered are the various arithmetic operations. The most basic arithmetic operation is the addition of two binary digits. Combinational circuit that performs the addition of two bits is called a half adder. One that performs the addition of three bits (two significant bits and previous carry) is a full adder. The names of the circuits stem from the fact that two half adders can be employed to implement a full adder.

In Half- and Full-Subtractor Circuits, binary subtraction is usually performed by using 2's complement. Two steps are required to obtain 2's complement. First, the subtrahend is inverted to 1's complement, i.e. a "1" to a "0" and a "0" to a "1". Secondly, a "1" is added to the least significant bit of the subtrahend in 1's complement. A half-subtractor performs the task if subtraction 1-bit at a time regardless of whether the minuend is greater or less than the subtrahend. "Borrow" from the previous subtraction is not taken into consideration. Full subtractor it's made the same as half subtractor but with some difference to get the carry from the previous bit.

Procedure (Discussion & Results)

First, I made 1 bit comparator using basic gates but in the inverse form and the truth table was as following:

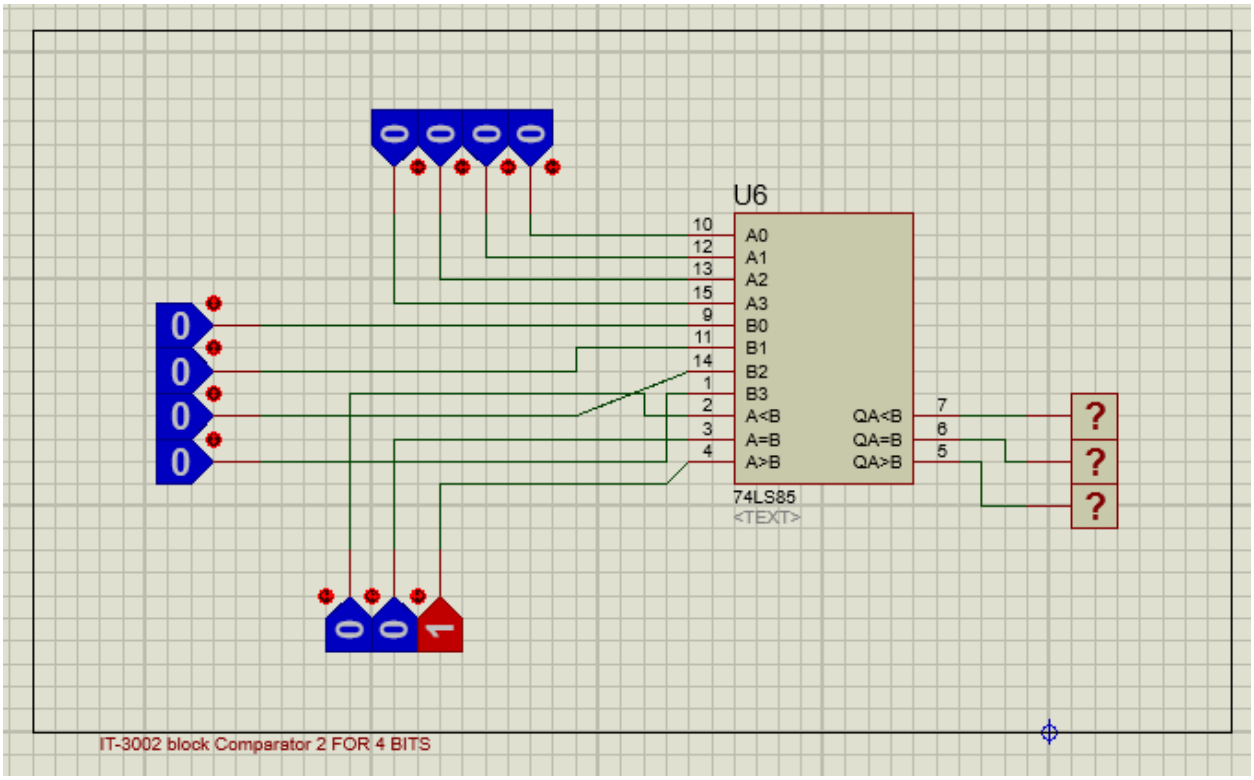
inputs		outputs		
A	B	A>B	A=B	A<B
0	0	1	0	1
0	1	1	1	0
1	0	0	1	1
1	1	1	0	1

INPUTS			OUTPUTS		
B (SW2)	A (SW1)		F1 (L1)	F2 (L2)	F5 (L3)
0	0	A=B	0	0	1
0	1	A>B	1	0	0
1	0	A<B	0	1	0
1	1	A=B	0	0	1

Table 2.1

And by simulating it by proteus we got those results which are true for one bit comparator, I used basic gates in the all-experiments file diagram and gave it input bit and output bit and connected them with wires.

Then we made it bigger by make it a 4-bit comparator using 74LS85 block, and we got this table from simulating:



INPUT			OUTPUT		
A>B	A=B	A<B	A>B	A=B	A<B
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	0	0
1	1	1	0	1	0

Those results when A=B, here the last bit from the other block that is connected will change the equation because the previous 4 bits are equal.

INPUT			OUTPUT		
A>B	A=B	A<B	A>B	A=B	A<B
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	0	0
1	1	1	1	0	0

And those when A>B, as the 4 bits A is bigger than B then the last digit won’t change the compare sign because it’s a lower digit

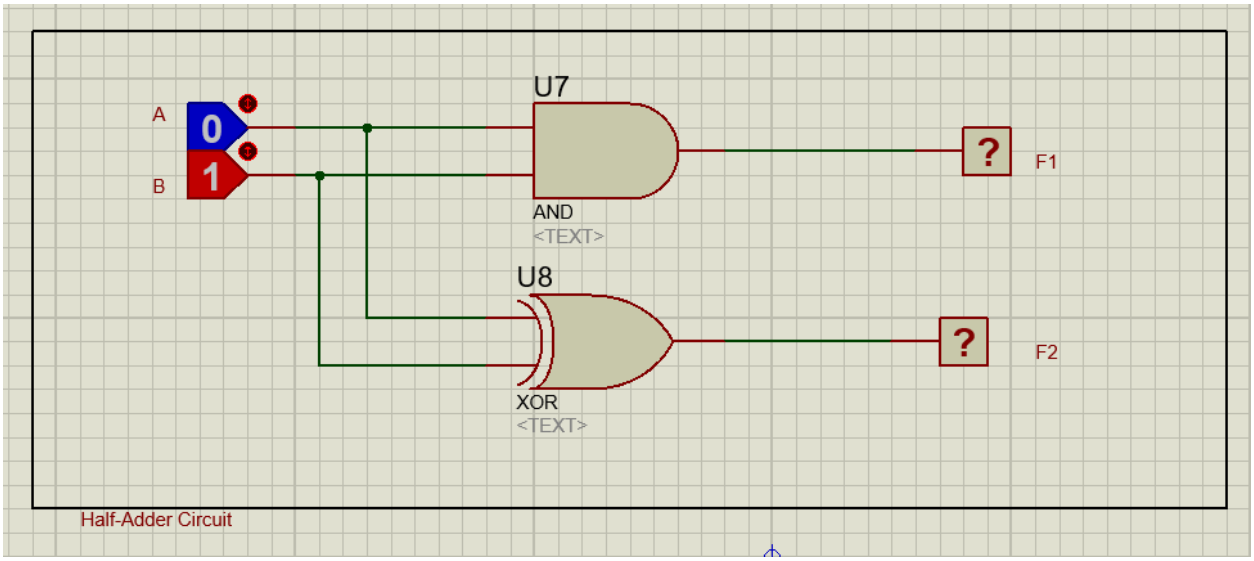
INPUT			OUTPUT		
A>B	A=B	A<B	A>B	A=B	A<B
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	0	0	1
1	0	1	0	0	1
1	1	1	0	0	1

And those when B>A. as the 4 bits A is smaller than B then the last digit won’t change the compare sign because it’s a lower digit.

This is the truth table for half adder as I simulated it on proteus I got those results and they are equal to the theoretical part

Sum= A xor B

Carry= A and B



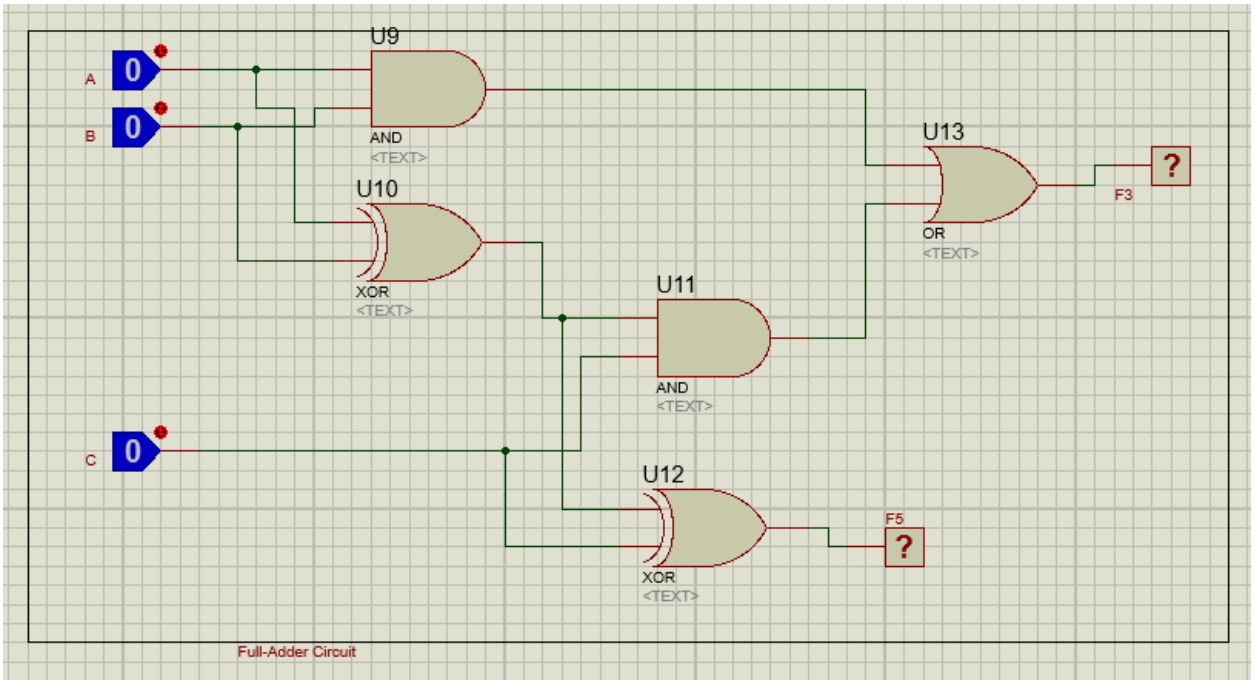
INPUTS		OUTPUTS	
SW1 (B)	SW0 (A)	CARRY (F1)	SUM (F2)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Table 2.3

In this truth table it's for full adder circuit which is made by proutus and the results are good, as shown in the table, the full adder add one bit to another and add the previous carry to add and put them in sum & carry

Sum= A xor B xor Cin

C out= AB + BCin + ACin



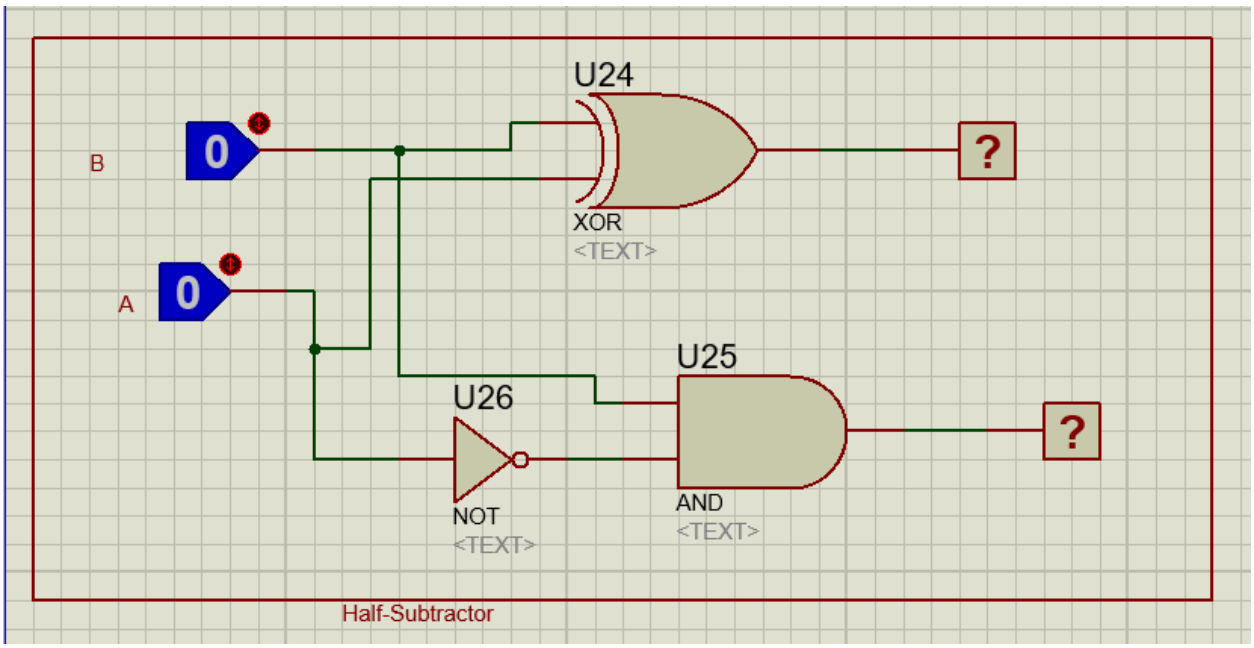
INPUTS			OUTPUTS	
A	B	C IN	SUM	C OUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0

1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

This is the truth table for half subtractor as I simulated it on proteus I got those results and they are equal to the theoretical part

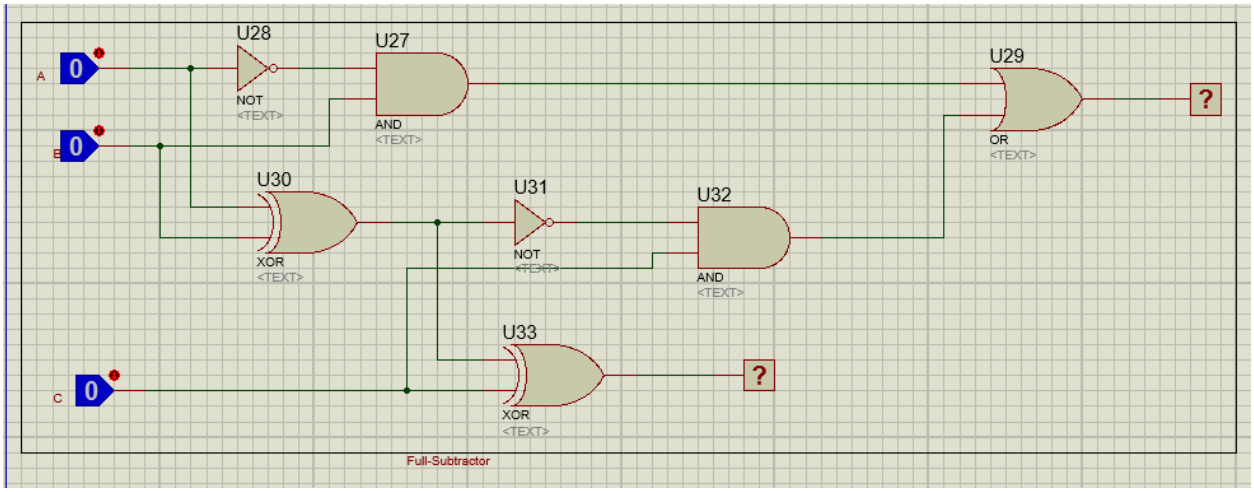
DIFFERENCE= A xor B

BORROW= A'B



A	B	DIFFERENCE	BORROW
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

In this truth table it's for full subtractor circuit which is made by proutus and the results are good, as shown in the table, the full subtractor subtract one bit from other and check the previous carry to add and put them in DIFFRANCE & BORROW



Difference= A'B'Cin + AB'Cin'+ A'BCin' + ABCin

Borrow= A'Cin + A'B + BCin

Inputs			Outputs			
C	A	B	F1	F2	F3	F5
0	0	1				
0	0	0				
0	1	1				
0	1	0				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Table 2.7

Here is the 3 in 1 circuit (full adder + full subtractor BCD adder)

If the Cin is =0 it will work as full adder and if it was =1 it will work as full subtractor and the final result using the second IC block will give us the BCD adder

Here is the full adder truth table

INPUT								OUTPUT				
Y3	Y2	Y1	Y0	X3	X2	X1	X0	Σ3	Σ2	Σ1	Σ0	F1(CARRY)
0	0	0	0	0	0	0	0					
0	0	0	0	0	0	0	1					
0	0	0	0	0	1	1	0					
0	0	0	0	1	0	0	1					
0	0	0	0	1	1	1	1					
0	0	0	1	0	0	1	1					
0	0	0	1	0	1	1	0					
0	0	0	1	1	0	0	0					
0	0	1	1	0	1	1	0					
0	1	0	0	1	0	0	0					
0	1	0	0	1	1	1	1					
1	0	0	0	0	1	1	1					
1	0	0	1	1	0	0	1					
1	0	1	0	1	0	1	1					

And here is the truth table of full subtractor

INPUT								OUTPUT				
X3	X2	X1	X0	Y3	Y2	Y1	Y0	F1	F11	F10	F9	F8
0	1	0	0	0	1	0	0	1	0	0	0	0
0	1	0	0	0	0	1	1	1	0	0	0	0
1	0	0	0	0	0	1	1	1	0	1	0	1
1	0	0	0	0	0	0	1	1	0	1	1	1
1	0	0	1	1	0	0	0	1	0	0	0	1
1	0	0	1	0	1	1	1	1	0	0	1	0
1	0	1	0	0	1	1	0	1	0	1	0	0
1	0	1	0	0	1	0	1	1	0	1	0	1
1	0	1	1	1	0	1	0	1	0	0	0	1
1	1	1	1	1	0	1	0	1	0	1	0	1

Table 2.8

And here is the truth table of BCD adder

INPUT								OUTPUT (U9)					LAST (U12)					
X3	X2	X1	X0	Y3	Y2	Y1	Y0	F1	F11	F10	F9	F8	F2	F3	F7	F6	F5	F4
0	0	0	0	0	0	0	0											
0	0	0	1	0	0	1	1											
0	0	1	1	0	1	0	0											
0	0	1	0	0	0	1	0											
0	0	1	0	1	0	0	0											
0	0	1	1	0	1	1	0											
0	1	0	0	0	0	1	0											
0	1	0	0	0	1	0	1											
0	1	0	0	0	1	1	0											
0	1	0	1	0	1	1	0											
0	1	1	0	0	1	1	1											
0	1	1	1	1	0	0	0											
0	1	1	1	1	0	0	1											
1	0	0	0	1	0	0	1											
1	0	0	1	1	0	0	1											
1	0	1	0	1	0	1	0											
1	0	1	0	1	0	1	1											
1	0	1	0	1	1	0	0											
1	0	1	1	1	1	1	0											
1	1	1	1	1	1	1	1											

Table 2.6

1.6 Problem

A 4-input , 3-output circuit that compares 2-bit unsigned numbers and output a (1) on one of the three output lines according to weather the first number is greater than , equal to , or less than the other number , you can only use two 4X1 multiplexers .

A1	A0	B1	B0	A>B	A<B	A==B
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	1
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	1	0	1	0
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	0	1

0

B1+B0

~(B1+B0)

B1

~(A>B + A<B)

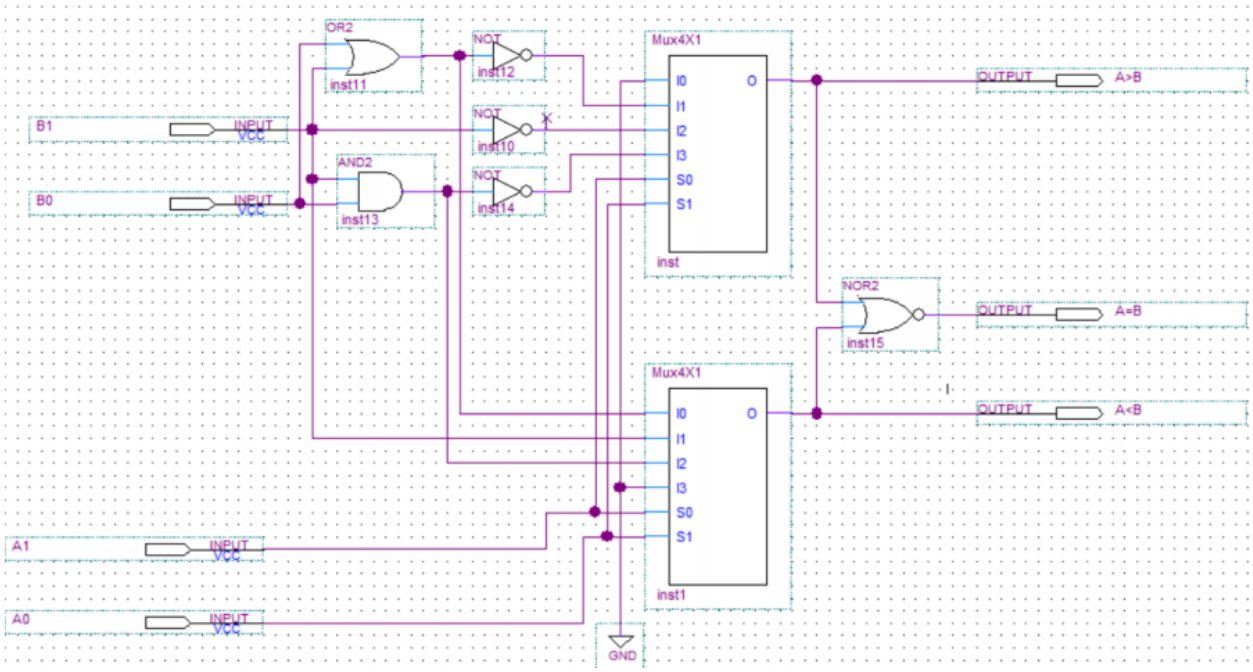
~B1

B1B0

~(B1B0)

0

Table 4 (truth table of the 2-bit comparator



Conclusion

In summarize, in this experiment we made a 1 bit comparator and extend it to make 4 bit comparator, we create half adder and half subtractor and make it more comprehensive by create full adder and full subtractor, and we extend them for 4 bits adder subtractor and used their blocks for creating the BCD adder as this circuit depends on the full adder and subtractor of the bits using a (key bit// 0 for add & 1 for subtract), all those circuits with their truth tables and Boolean equations.

References

Half Subtractor : Circuit Design, Truth Table & Its Applications (elprocus.com)

Full Subtractor Circuit Design - Theory, Truth Table, K-Map & Applications (elprocus.com)