Unit III

Session I
Combinational circuits

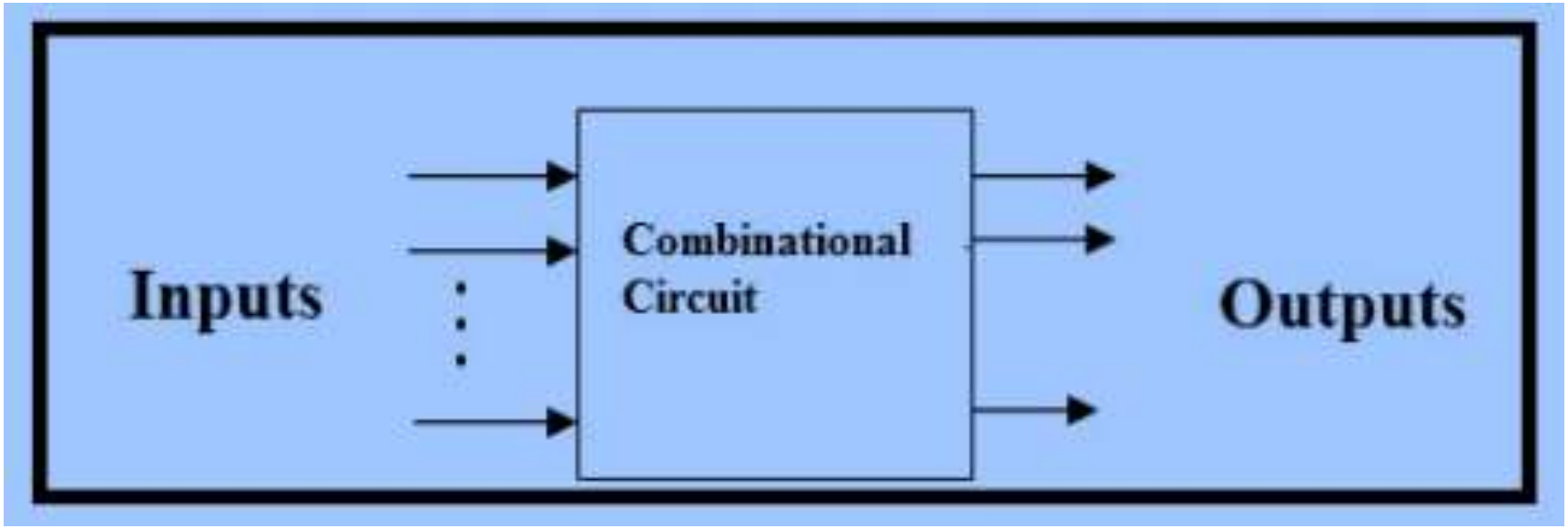
Combinational Circuits

- The output of combinational circuit at any instant of time, depends only on the levels present at input terminals.
- The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit.
- A combinational circuit can have an n number of inputs and m number of outputs

Combinational Logic Circuit

- **:**A combinational logic circuit consists of logic gates whose output is determined by the combination of current inputs.
- It consists of input variables, logic gate and output variables.
- No feedback is required.
- No memory is required.
- Examples of Combinational Circuits: Adders, Code converters,Multiplexer, Decoder etc.

Block diagram

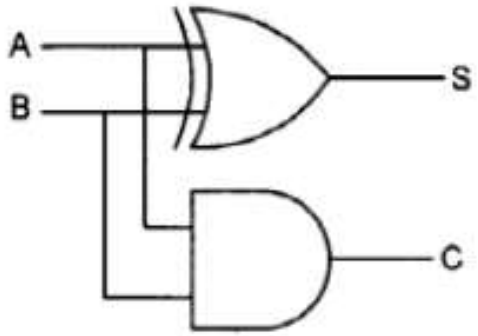


Half Adder

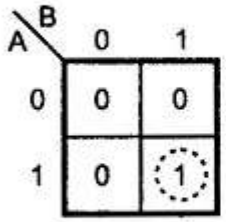
Half adder is a combinational logic circuit with two inputs and two outputs. The half adder circuit is designed to add two single bit binary number A and B. It is the basic building block for addition of two **single** bit numbers. This circuit has two outputs **carry** and **sum**.

Design Half Adder

Inputs		Outputs	
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

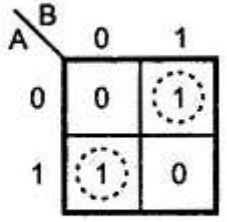


For Carry

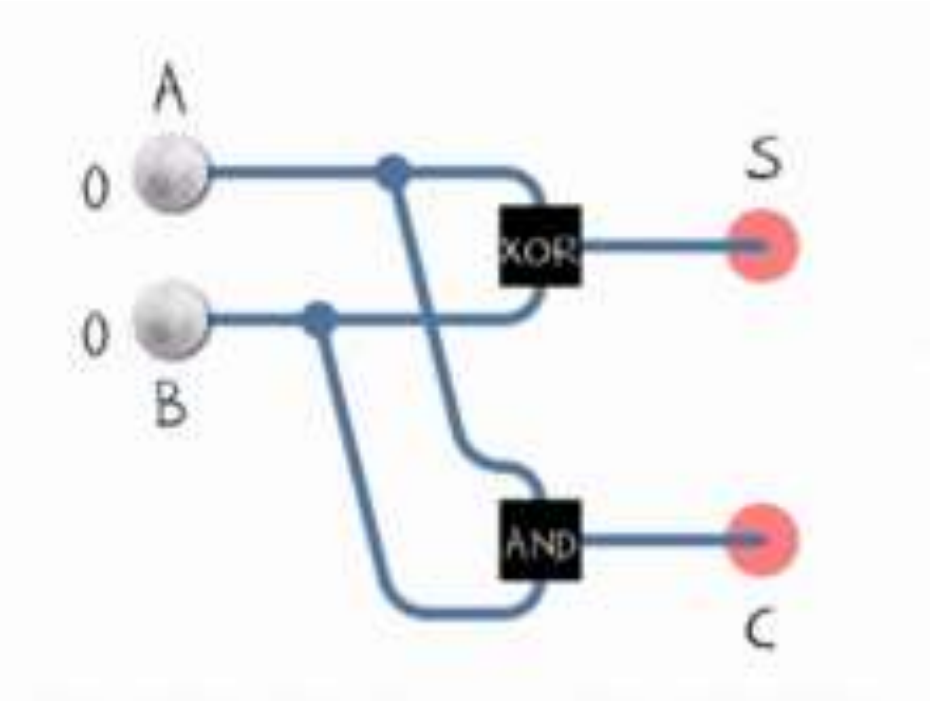


Carry = AB

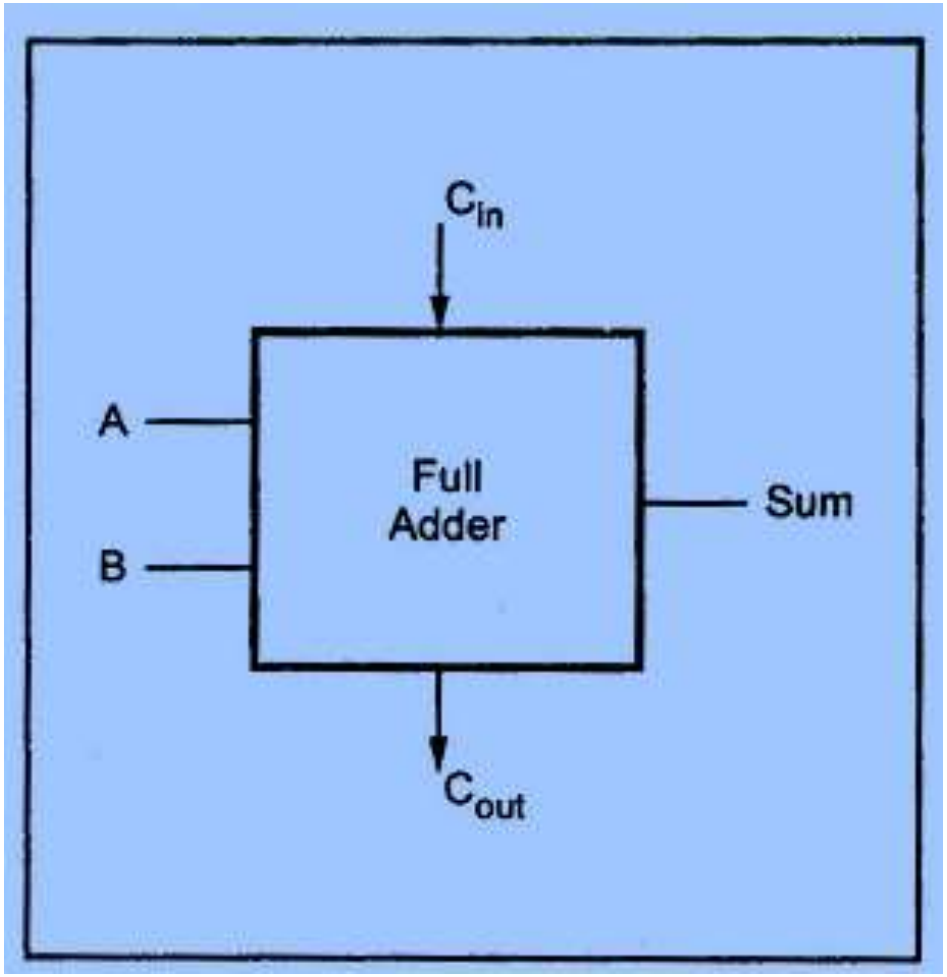
For Sum



Sum = $A\overline{B} + \overline{A}B$
= $A \oplus B$


A
0
0
B
XOR
AND
S
C

Full Adder

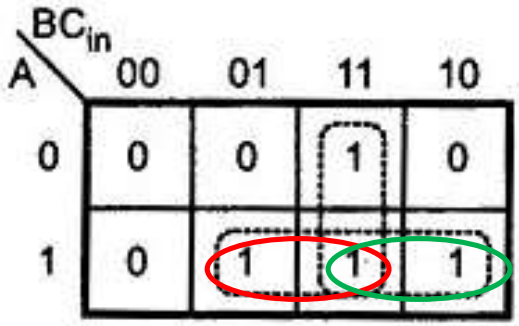


Truth Table

Inputs			Outputs	
A	B	C_{in}	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

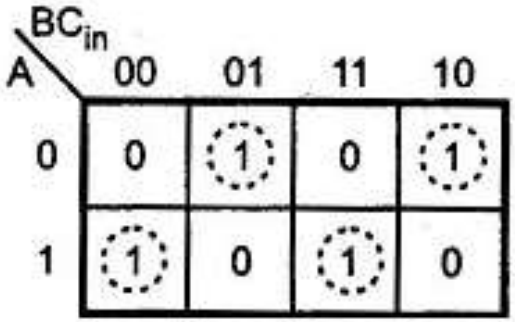
K map

For Carry (C_{out})



$C_{out} = AB + A\,C_{in} + B\,C_{in}$

For Sum



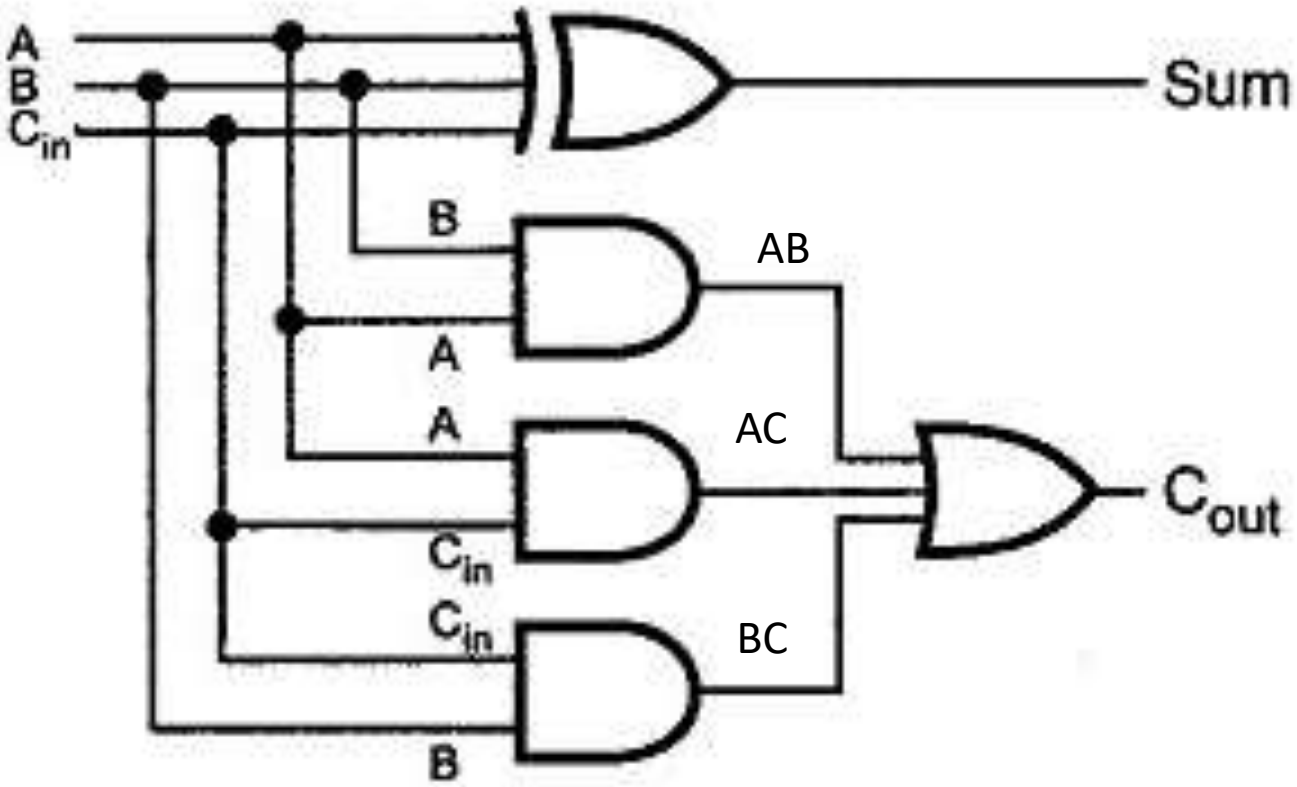
$Sum = \overline{A}\,\overline{B}C_{in} + \overline{A}B\overline{C}_{in} + A\overline{B}\,\overline{C}_{in} + ABC_{in}$

Sum
Cin(y')+Cin'(y)=Cin xor Y=Cin xor A xor B

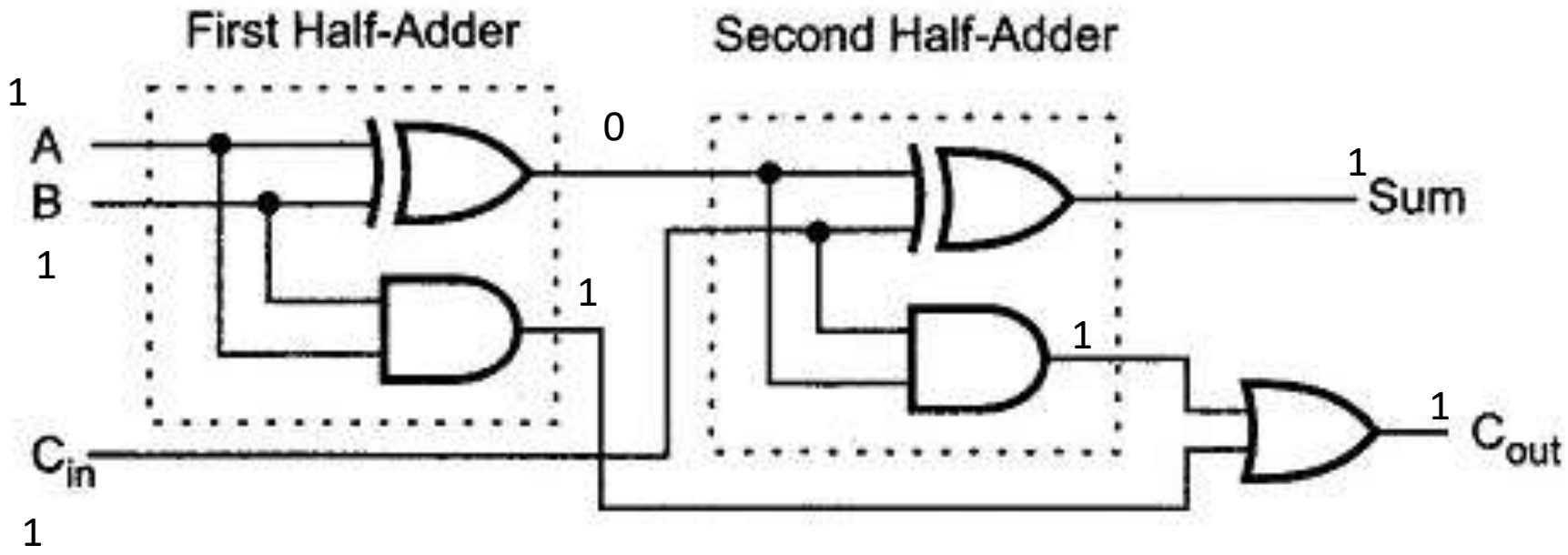
The **Boolean Expression** for sum can be further simplified as follows :

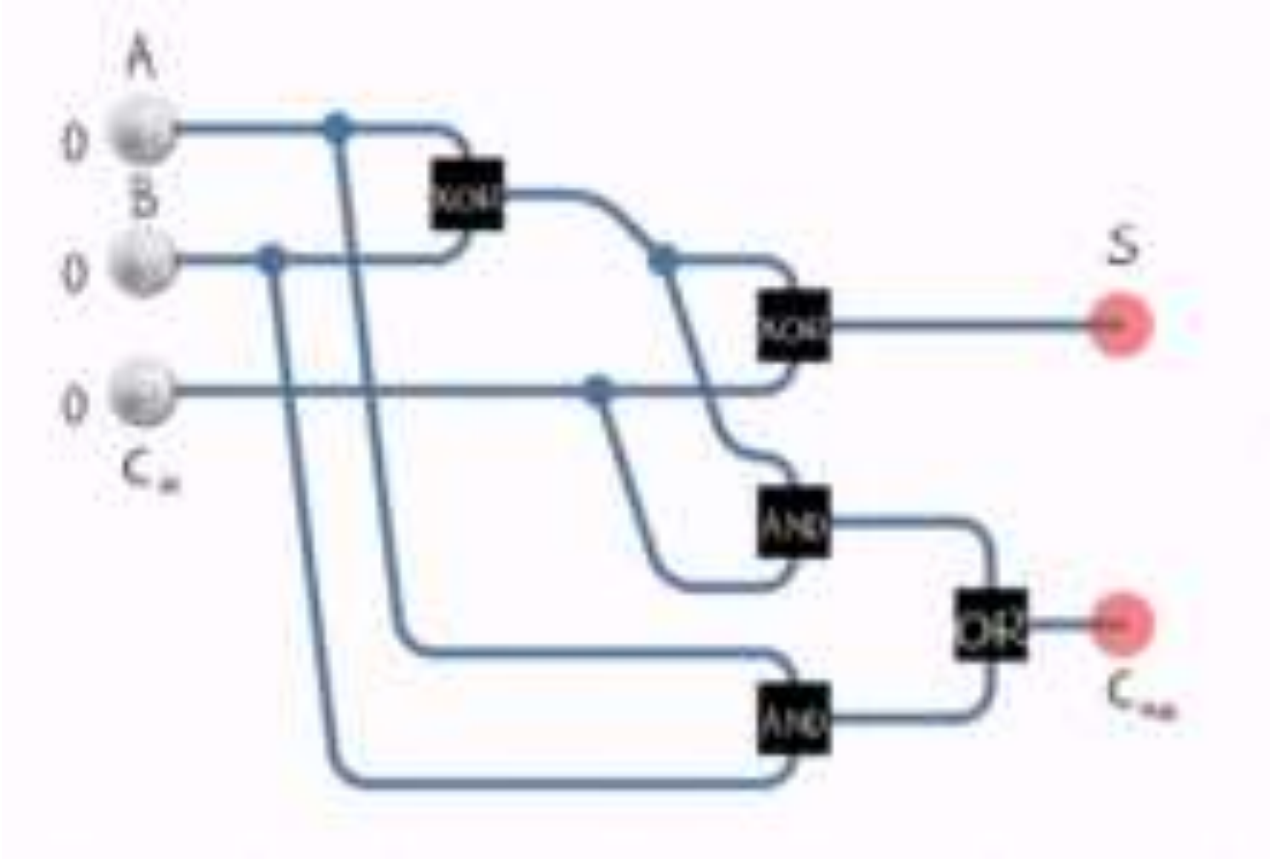
$$\begin{aligned}
\text{Sum} &= \overline{A}\ \overline{B}\ C_{in} + \overline{A}\ B\ \overline{C}_{in} + A\ \overline{B}\ \overline{C}_{in} + A\ B\ C_{in} \\
&= C_{in}\ (\overline{A}\ \overline{B} + AB) + \overline{C}_{in}\ (\overline{A}\ B + A\ \overline{B}) \\
&= C_{in}\ (A \odot B) + \overline{C}_{in}\ (A \oplus B) \\
&= C_{in}\ (\overline{A \oplus B}) + \overline{C}_{in}\ (A \oplus B) \\
&= C_{in} \oplus (A \oplus B)
\end{aligned}$$

Full adder Circuit Diagram

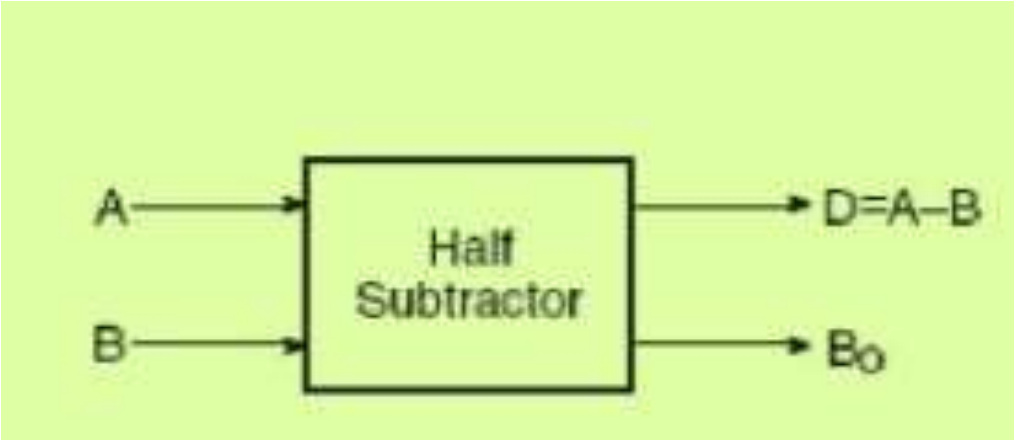


Full adder using 2 half adders




A
0
B
0
0
S
AND
OR

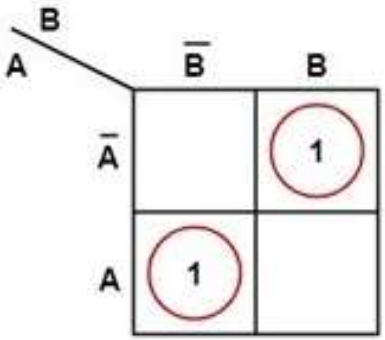
Half Subtractor



A	B	D	B_O
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

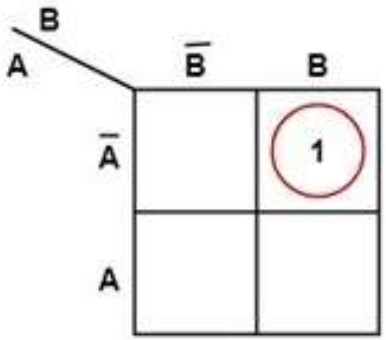
K map

For D:



$D = A \oplus B$

For b:



$b = \bar{A}\,B$

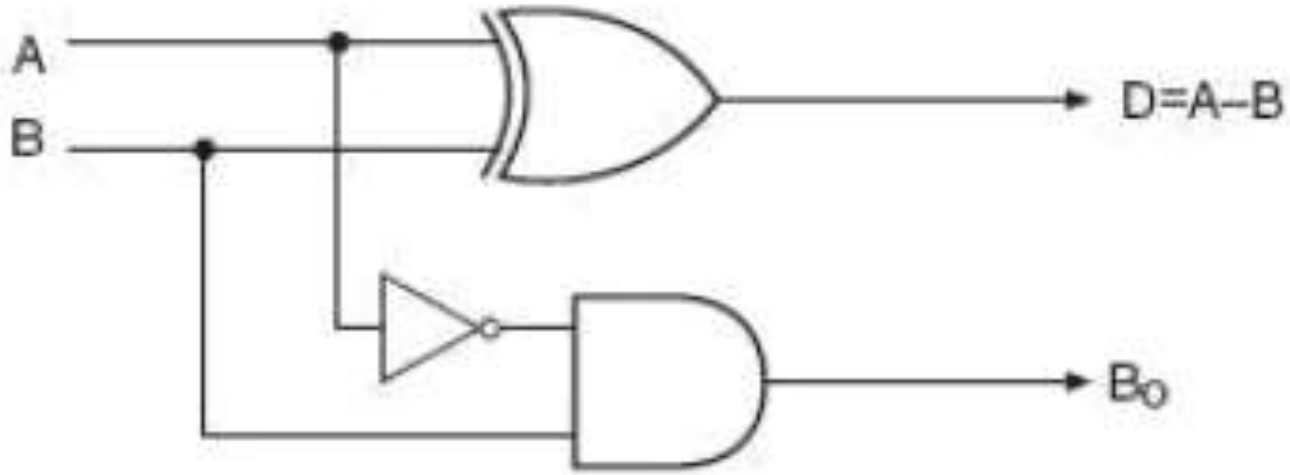
$$D = \overline{A}.B + A.\overline{B}$$

$$B_0 = \overline{A}.B$$

Half Subtractor

$$D = \overline{A}.B + A.\overline{B}$$

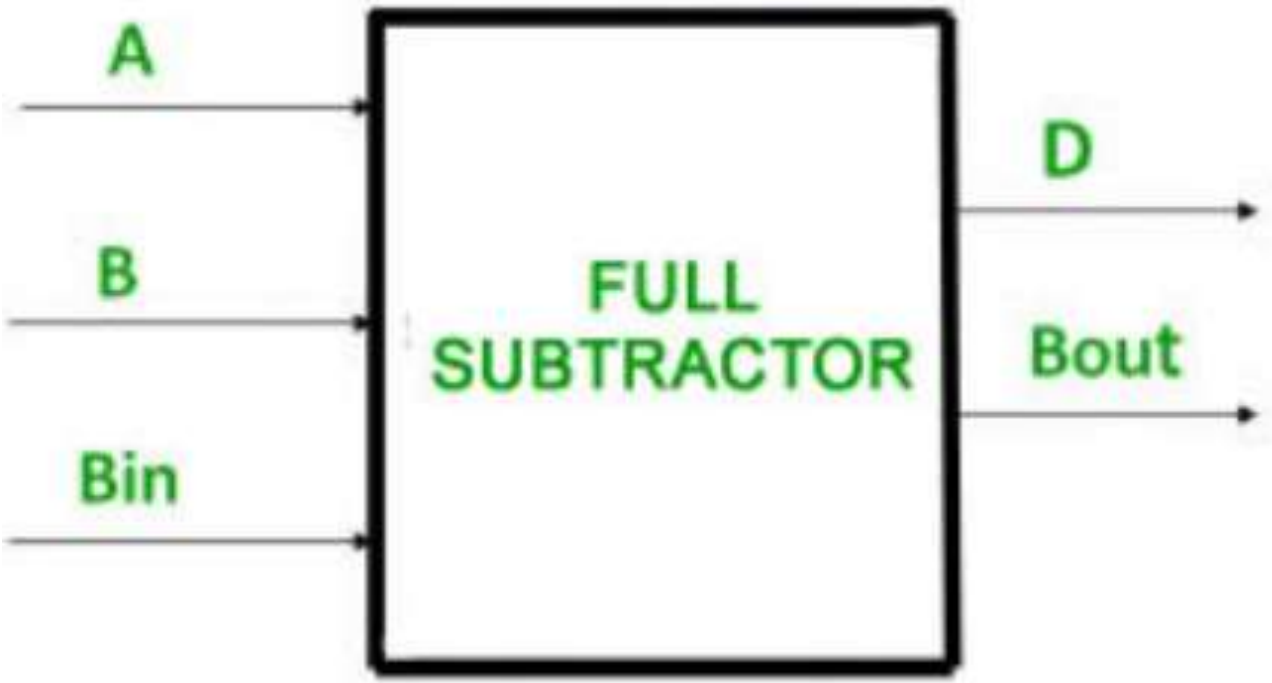
$$B_o = \overline{A}.B$$



Full Subtractor

- The Half Subtractor is used to subtract only two numbers. To overcome this problem, a full subtractor was designed. The full subtractor is used to subtract three 1-bit numbers A, B, and C, which are minuend, subtrahend, and borrow, respectively. The full subtractor has three input states and two output states i.e., difference and borrow.

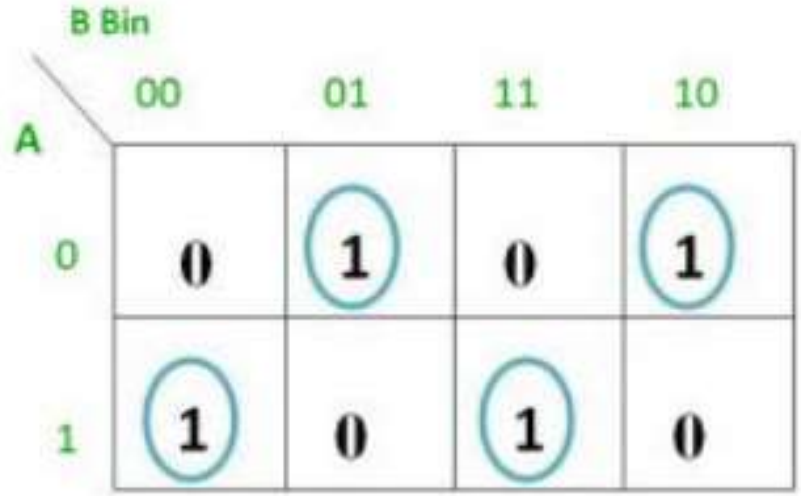
Full Subtractor



Truth Table

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

K map for difference



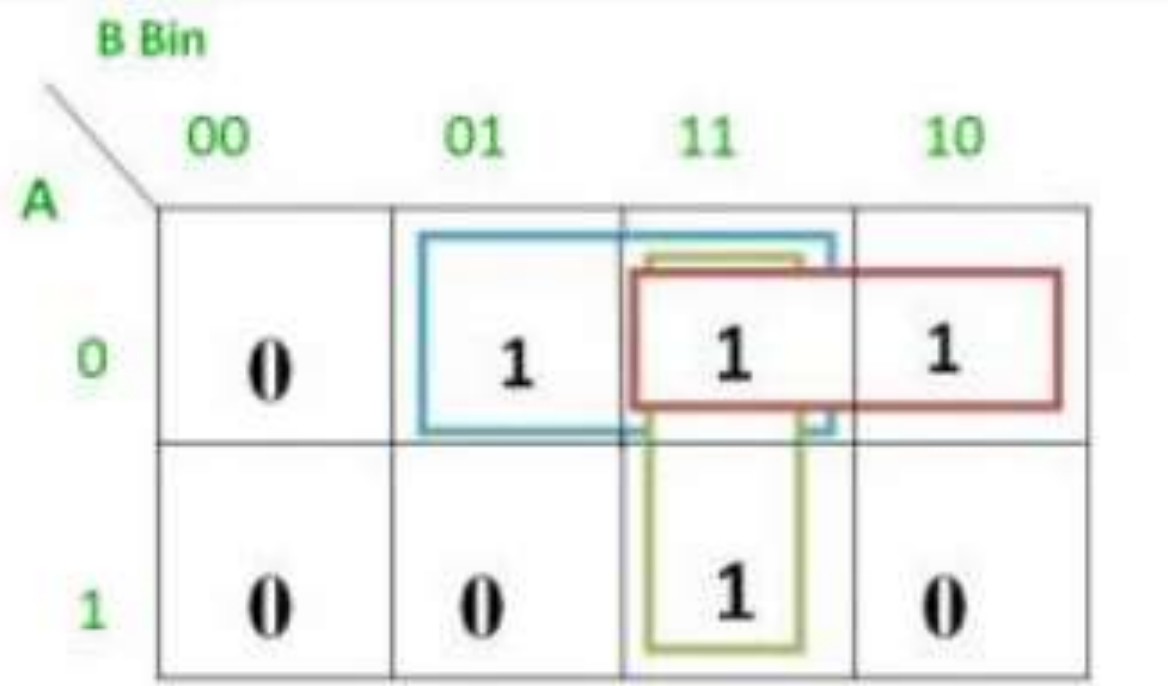
D=A'B'Bin + AB'Bin' + A'BBin' + ABBin

Bin Y'+Bin' Y= Bin XOR Y= Bin XOR A XOR B

$$
\begin{aligned}
D &= A'B'Bin + A'BBin' + AB'Bin' + ABBin \\
&= Bin(A'B' + AB) + Bin'(AB' + A'B) \\
&= Bin(\ A\ XNOR\ B) + Bin'(A\ XOR\ B) \\
&= Bin\ (A\ XOR\ B)' + Bin'(A\ XOR\ B) \\
&= Bin\ XOR\ (A\ XOR\ B) \\
&= (A\ XOR\ B)\ XOR\ Bin
\end{aligned}
$$

$$D = A \oplus B \oplus B_{in}$$

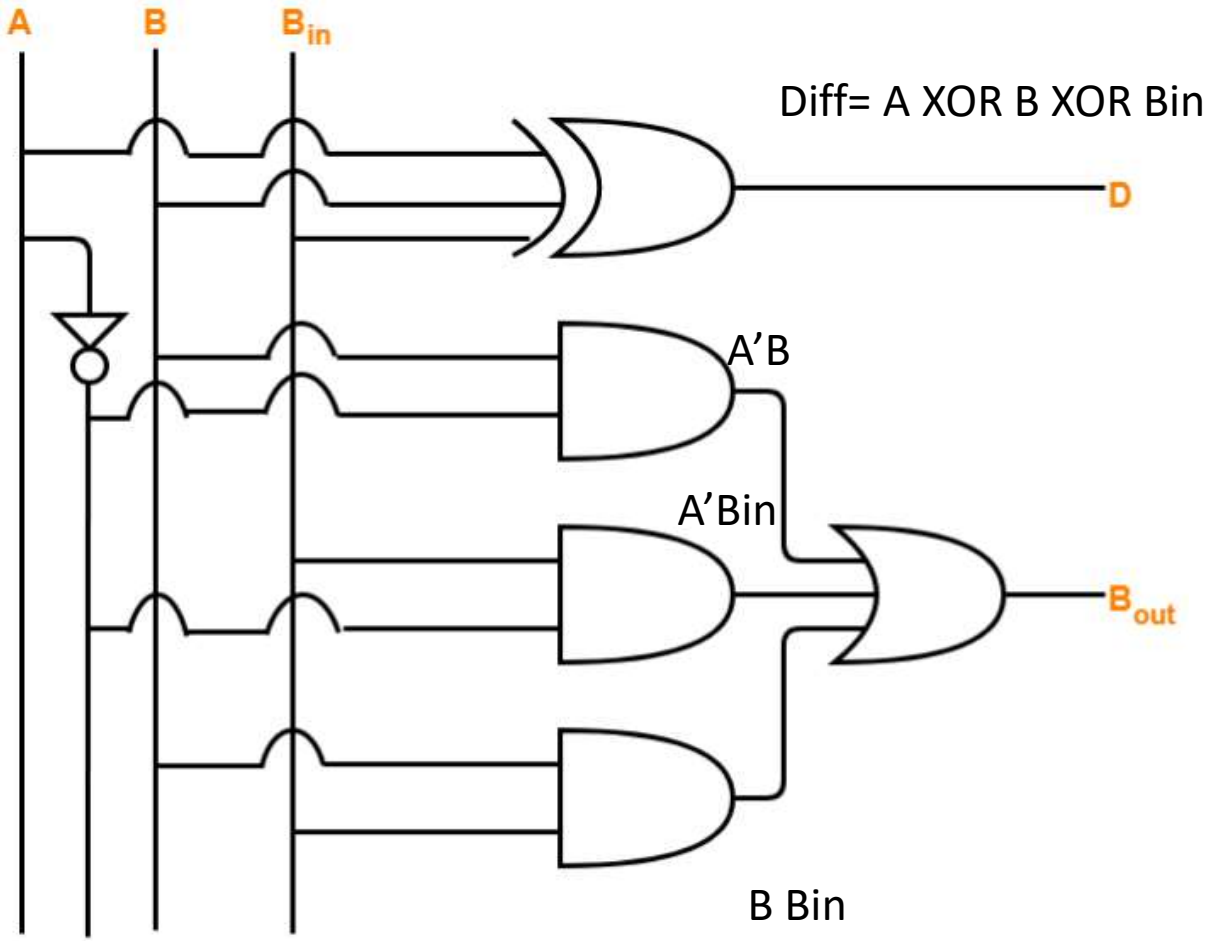
K map for borrow



$$B_{out} = A'Bin + A'B + BBin$$

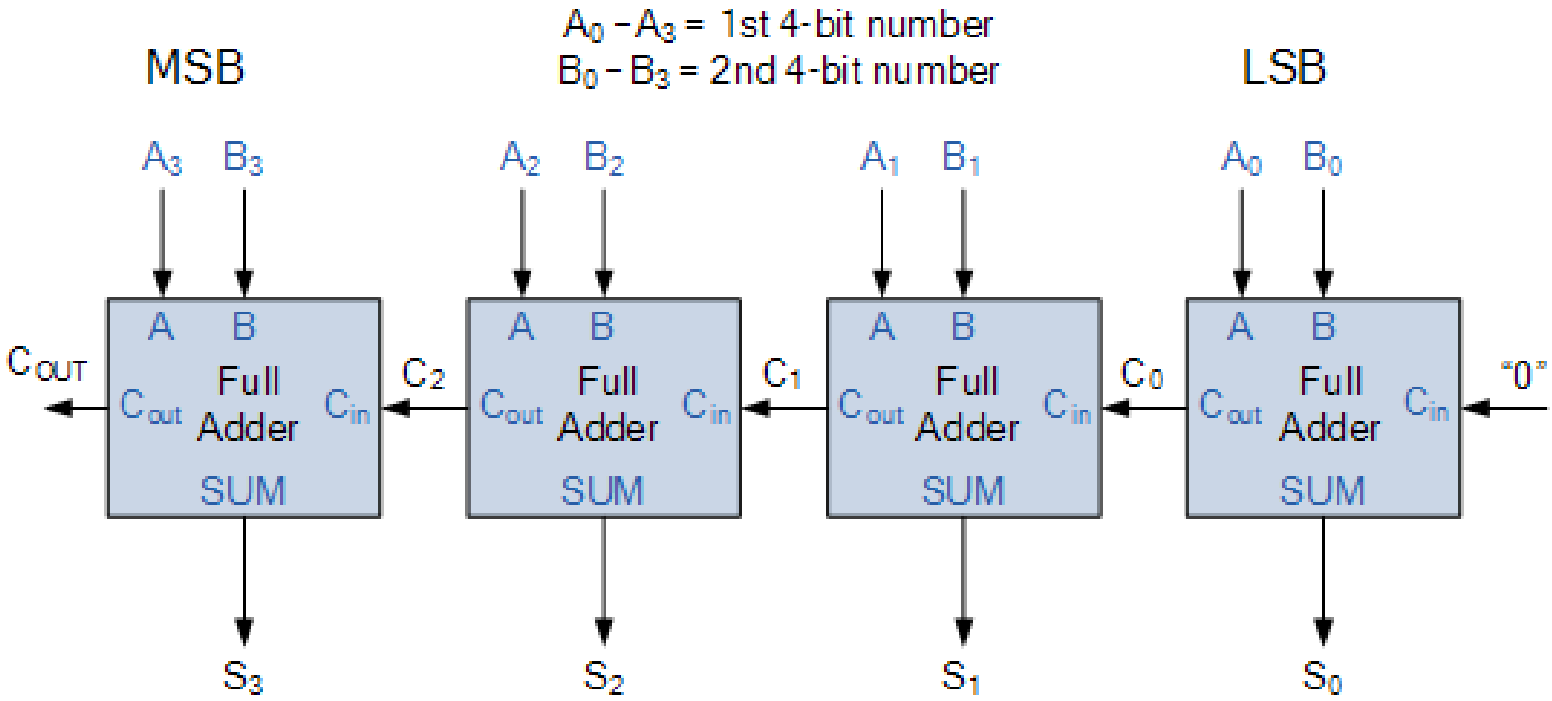
$$B_{out} = \overline{A}\,B + (\overline{A} + B)\,B_{in}$$

Full Subtractor



4 bit parallel adder

A 4-bit Ripple Carry Adder



A = 1 1 0 1

B= 1 0 0 1

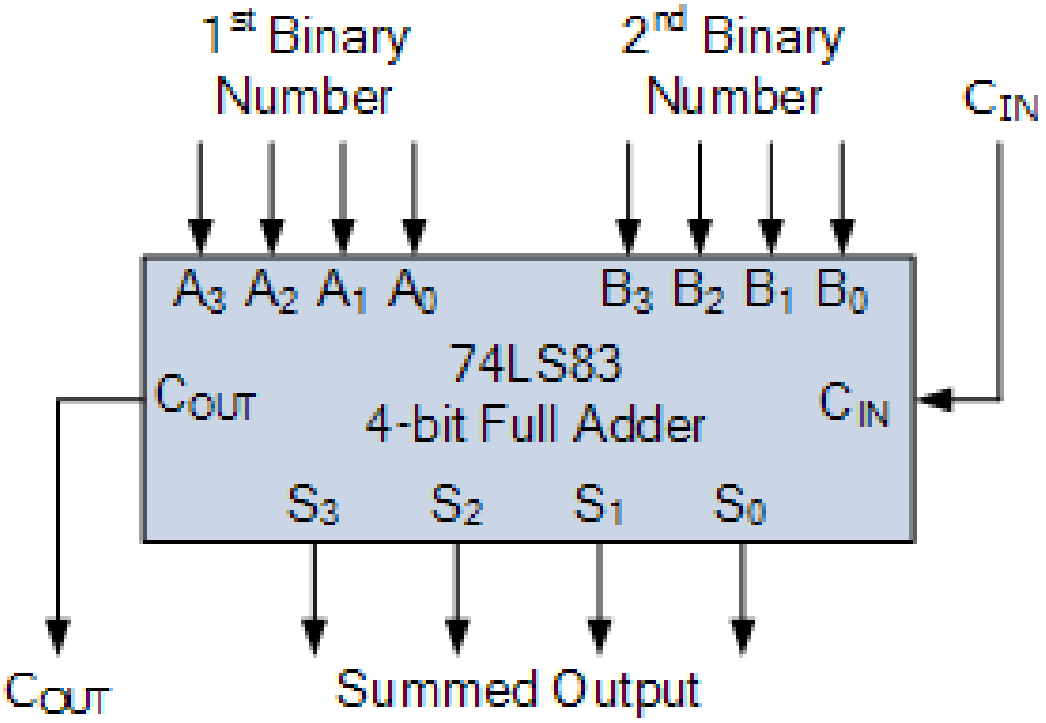
Sum= 1 0110

S3=0 Cout=1

S2 = 1 C2=0

S1= 1 C1=0

S0 =0 C0=1


1st Binary Number
2nd Binary Number
CIN
A3 A2 A1 A0
B3 B2 B1 B0
74LS83
4-bit Full Adder
COUT
CIN
S3 S2 S1 S0
COUT
Summed Output

BCD Numbers

BCD or **Binary Coded Decimal** is that number system or code which has the binary numbers or digits to represent a decimal number.
A decimal number contains 10 digits (0-9). Now the equivalent binary numbers can be found out of these 10 decimal numbers. In case of **BCD** the binary number formed by four binary digits, will be the equivalent code for the given decimal digits. In **BCD** we can use the binary number from 0000-1001 only, which are the decimal equivalent from 0-9 respectively. Suppose if a number have single decimal digit then it's equivalent **Binary Coded Decimal** will be the respective four binary digits of that decimal number and if the number contains two decimal digits then it's equivalent **BCD** will be the respective eight binary of the given decimal number. Four for the first decimal digit and next four for the second decimal digit.

Let, $(12)_{10}$ be the decimal number whose equivalent **Binary coded decimal** will be 0001 0010. Four bits from L.S.B is binary equivalent of 2 and next four is the binary equivalent of 1.
Table given below shows the binary and **BCD** codes for the decimal numbers 0 to 15.
From the table below, we can conclude that after 9 the decimal equivalent binary number is of four bit but in case of BCD it is an eight bit number. This is the main difference between Binary number and binary coded decimal. For 0 to 9 decimal numbers both binary and BCD is equal but when decimal number is more than one bit BCD differs from binary.

Decimal number	Binary number	Binary Coded Decimal(BCD)
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001

Decimal number	Binary number	Binary Coded Decimal(BCD)
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010
13	1101	0001 0011
14	1110	0001 0100
15	1111	0001 0101

Example 1

But the result of addition here is less than 9, which is valid for BCD numbers.

$$\begin{array}{r} 0001 \\ +\ 0101 \\ \hline 0110 \end{array}$$

Example 2

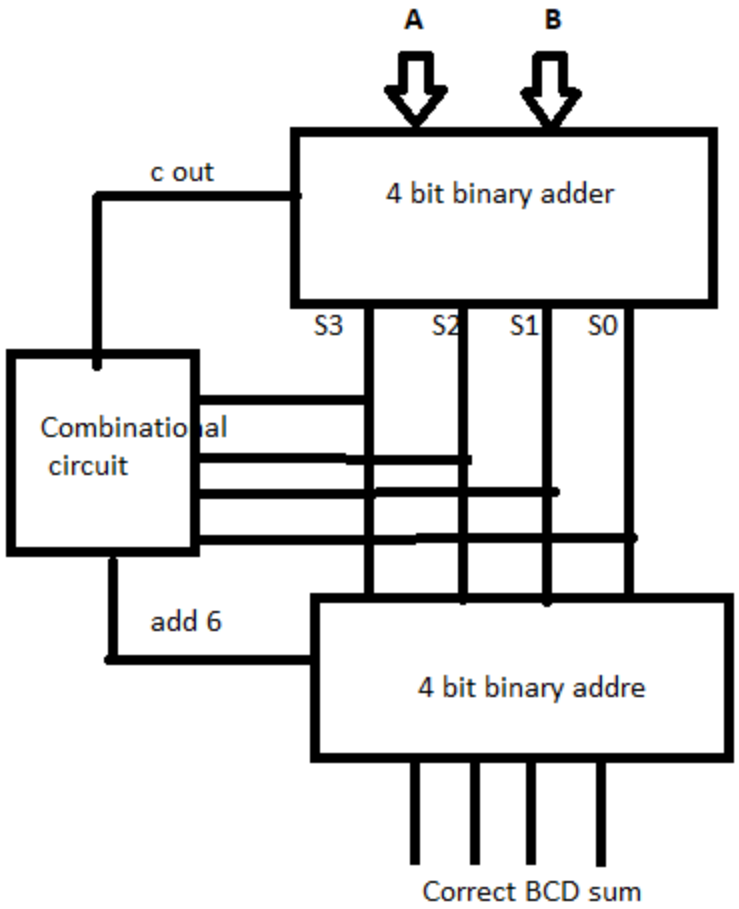
If the four bit result of addition is greater than 9 and if a carry bit is present in the result then it is invalid and we have to add 6 whose binary equivalent is $(0110)_2$ to the result of addition. Then the resultant that we would get will be a valid binary coded number. In case 2 the result was $(1111)_2$, which is greater than 9 so we have to add 6 or $(0110)_2$ to it.

$$\begin{array}{r} 1010 \\ +\ 0101 \\ \hline 1111 \end{array}$$

$$\begin{array}{r} 1111 \\ +\ 0110 \\ \hline 0001\ 0101 \\ 1 \quad\ 5 \end{array}$$

BCD ADDER

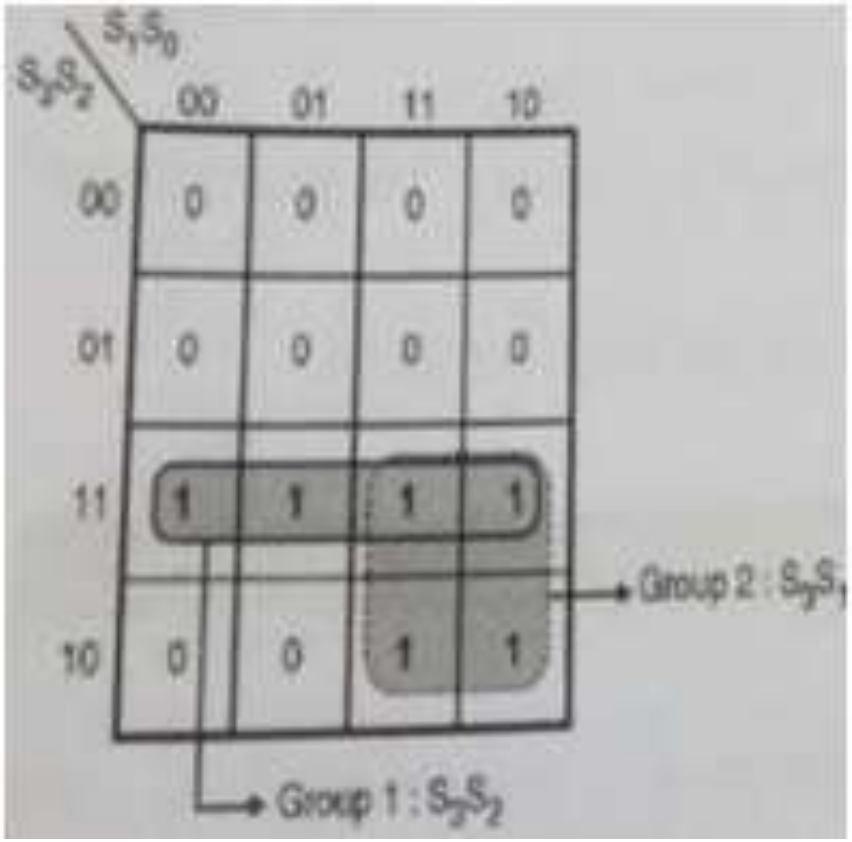
- A BCD adder adds two BCD digits and produces output as a BCD digit. A BCD or Binary Coded Decimal digit cannot be greater than 9.
- The two BCD digits are to be added using the rules of binary addition. If sum is less than or equal to 9 and carry is 0, then no correction is needed. The sum is correct and in true BCD form.
- But if sum is greater than 9 or carry =1, the result is wrong and correction must be done. The wrong result can be corrected adding six (0110) to it.
- For implementing a BCD adder using a binary adder circuit IC 7483, additional combinational circuit will be required, where the Sum output $S_3 - S_0$ is checked for invalid values from 10 to 15. The truth table and K-map for the same is as shown:


A
B
c out
4 bit binary adder
S3
S2
S1
S0
Combinational
circuit
add 6
4 bit binary addre
Correct BCD sum

TRUTH TABLE

I/P				O/P
S3	S2	S1	S0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

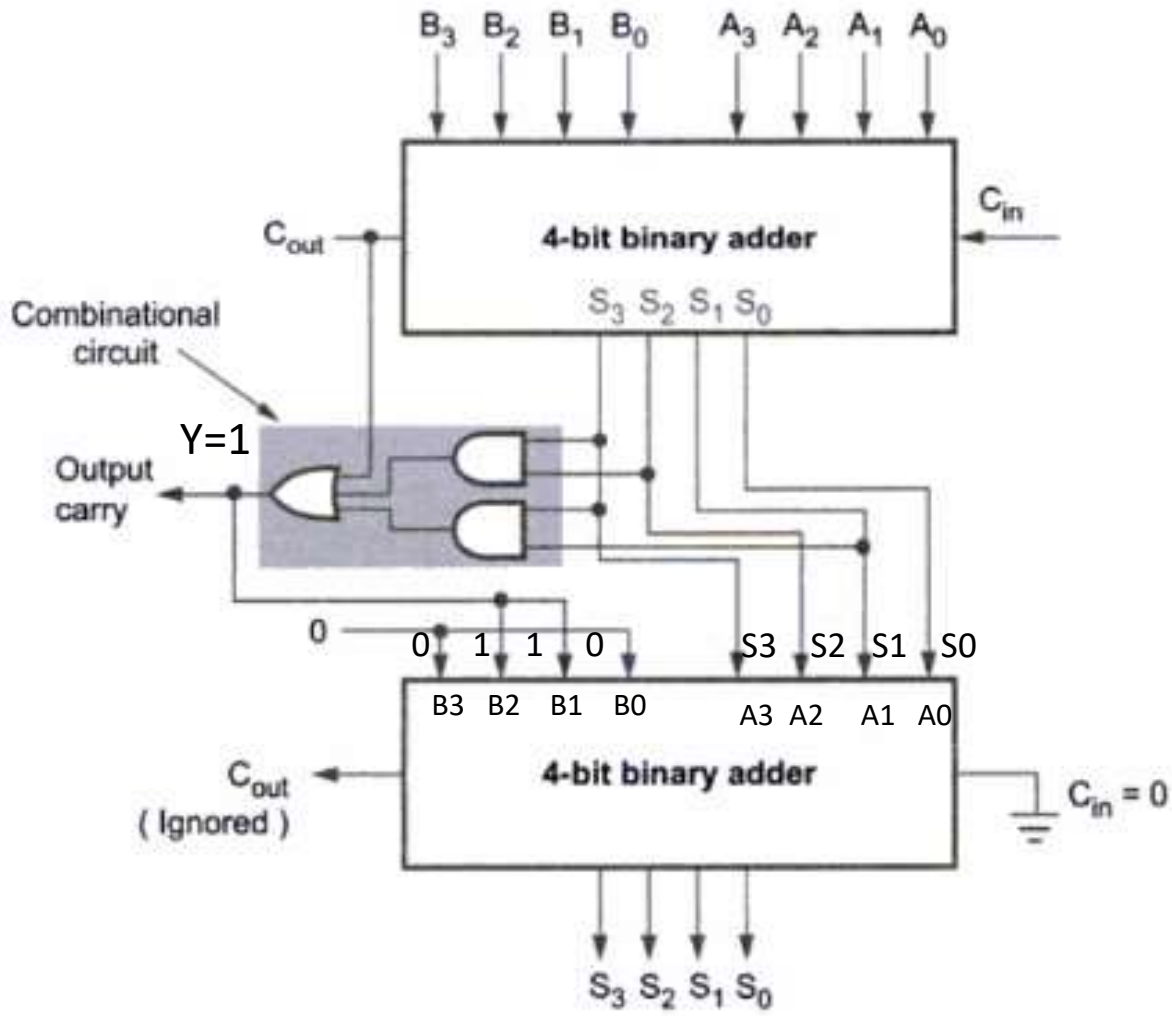
K map



The Boolean expression is, $Y = S_3S_2 + S_3S_1$

- The BCD adder is shown below. The output of the combinational circuit should be 1 if Cout of adder-1 is high. Therefore Y is ORed with Cout of adder 1.
- The output of combinational circuit is connected to B1B2 inputs of adder-2 and $B_3 = B_0 = 0$ as they are connected to ground permanently. This makes $B_3B_2B_1B_0 = 0110$ if Y' = 1.
- The sum outputs of adder-1 are applied to $A_3A_2A_1A_0$ of adder-2. The output of combinational circuit is to be used as final output carry and the carry output of adder-2 is to be ignored.

4 bit BCD ADDER



1. As shown in the Fig, the two BCD numbers, together with input carry, are first added in the top 4-bit binary adder to produce a binary sum.

2. When the output carry is equal to zero (i.e. when sum <= 9 and C_out = 0) nothing (zero) is added to the binary sum.

   When it is equal to one (i.e. when sum > 9 or C_out = 1), binary 0110 is added to the binary sum through the bottom 4-bit binary adder.

3. The output carry generated from the bottom binary adder can be ignored, since it supplies information already available at the output carry terminal.

Case1: sum <= 9 and carry=0

Case 2: sum > 9 and carry=0

Case 3: sum <= 9 but carry=1


thank
YOU