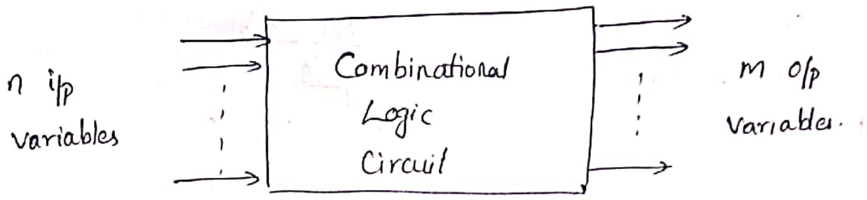
UNIT-4

Combinational Logic Circuits – I

Introduction :

When logic gates are connected together to produce a specified output for certain specified combinations of input variables, with no storage involved, the resulting circuit is called combinational logic. In combinational logic, the output variables are at all times dependent on the combination of i/p variables.

A combinational circuit consists of input variables, logic gates, and output variables.



Design Procedure :

The design procedure of the combinational circuit involves the following steps :

1. Define the problem
2. Mark the no. of inputs and no. of outputs
3. Obtain the truth table.
4. Write the output boolean function interms of input.
5. Simplified function for each o/p is obtained.
6. Draw the logic diagram.

Adders :

Digital computers perform various arithmetic operations. The most basic operation is the addition of two binary digits.

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 Carry 1.

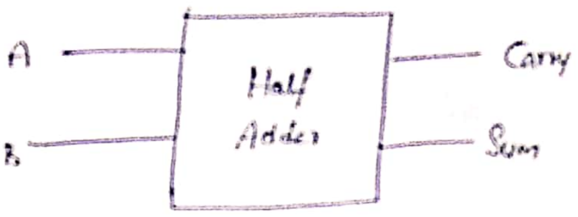
The ckt which performs addition of 2 bits is known as Half-adder. The circuit performs addition of 3 bits (two significant & a previous carry) is known as full-adder.

Half - Adder

The half adder operation needs two binary inputs a

and addend bits, and two binary outputs : Sum and Carry.

Inputs		outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



For Sum

	$\bar{B}$	B
$\bar{A}$	0	1
A	1	0

$Sum = A\bar{B} + \bar{A}B$

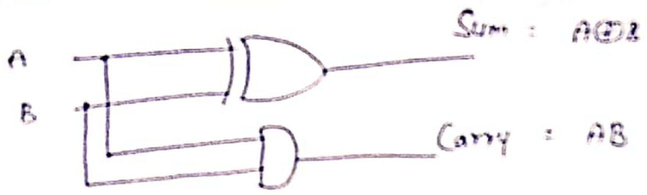
$= A \odot B$

For Carry

	$\bar{B}$	B
$\bar{A}$	0	0
A	0	1

$Carry = AB$

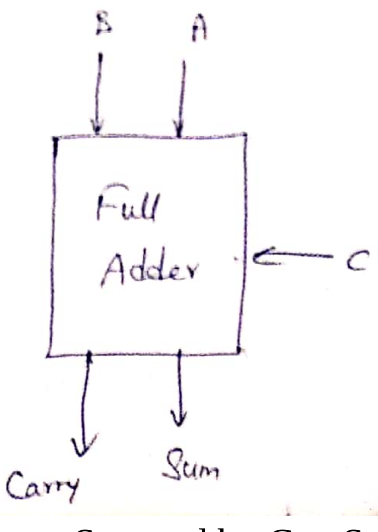
Logic diagram :



Full - Adder :

A full adder is a Combinational Circuit that forms the arithmetic Sum of three input bits. It Consists of three inputs and two outputs.

Inputs			Outputs	
A	B	C	Sum	Carry
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



For Carry

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$			1	
A		1	1	1

For Sum

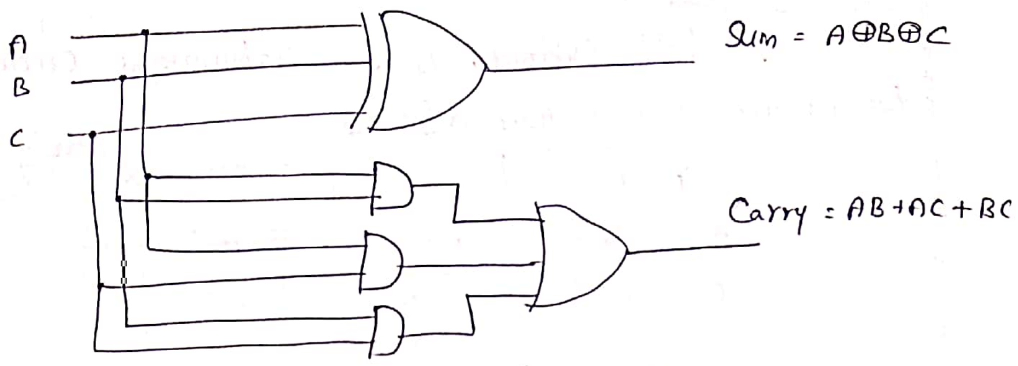
	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$		1		1
A	1		1	

Carry $= AB + AC + BC$

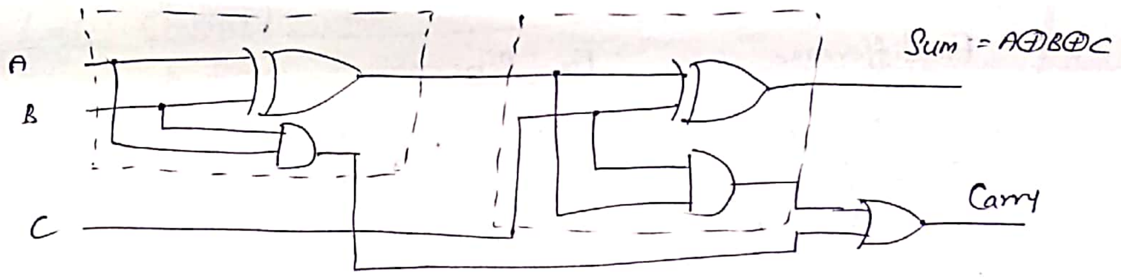
Sum $= \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

$= A \oplus B \oplus C$

Logic diagram :



Implementation of full adder using half adders :



In above circuit carry value is

$$
\begin{aligned}
\text{Carry} &= AB + (A \oplus B)C \\
&= AB + C(A\bar{B} + \bar{A}B) \\
&= AB + A\bar{B}C + \bar{A}BC \\
&= AB(1+C) + A\bar{B}C + \bar{A}BC \\
&= AB + ABC + A\bar{B}C + \bar{A}BC \\
&= AB + AC(B+\bar{B}) + \bar{A}BC \\
&= AB + AC + \bar{A}BC \\
&= AB(1+C) + AC + \bar{A}BC \\
&= AB + ABC + AC + \bar{A}BC \\
&= AB + AC + ABC + \bar{A}BC \\
&= AB + AC + BC(A+\bar{A}) \\
&= AB + AC + BC
\end{aligned}
$$

Subtractor :

$0 - 0 = 0$

$0 - 1 = 1$ with 1 borrow

$1 - 0 = 1$

$1 - 1 = 0$

In all operations, each Subtrahend bit is Subtracted from the minuend bit. In case of second operation the minuend bit is smaller than the Subtrahend bit, hence '1' is borrowed.

Half Subtractor :

A half Subtractor is a combinational circuit that subtracts two bits and produces their difference.

Inputs		Outputs	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

For difference

	$\bar{B}$	B
$\bar{A}$	0	1
A	1	0

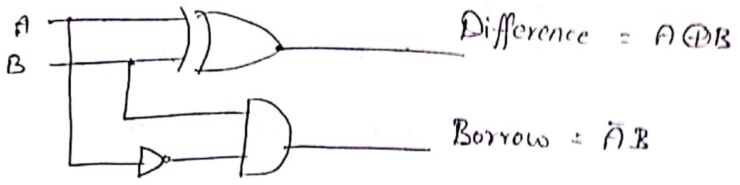
Difference = $A \oplus B$

For Borrow

	$\bar{B}$	B
$\bar{A}$	0	1
A	0	0

Brrow = $\bar{A}B$

Logic Circuit :



Full Subtractor :

A full Subtractor is a combinational circuit that performs a subtraction between two bits, taking into account borrow of the lower significant stage. This circuit has three inputs and two outputs.

Inputs			Outputs	
A	B	C	Difference	Borrow
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

For difference

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	1	0	1
A	1	0	1	0

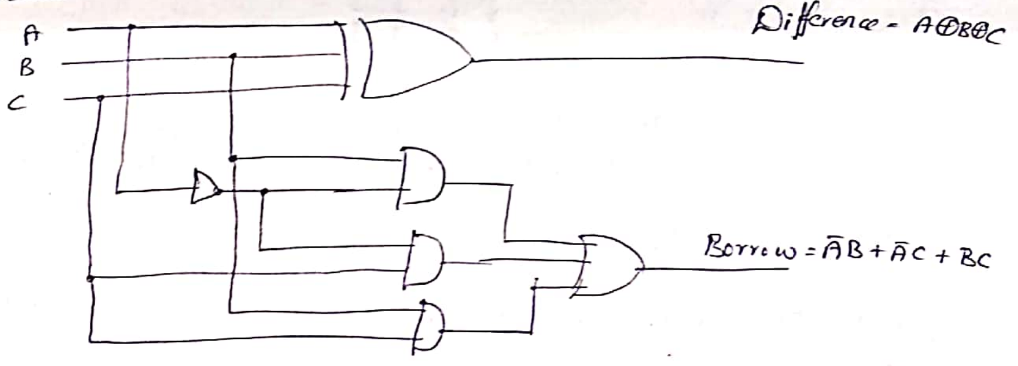
For Borrow

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	1	1	1
A	0	0	1	0

$$\text{Difference} = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$
$$= A \oplus B \oplus C$$

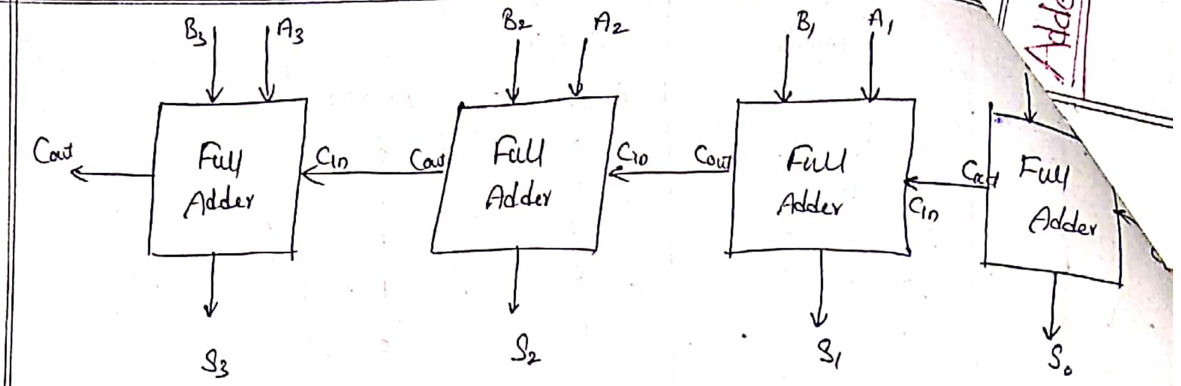
$$\text{Borrow} = \bar{A}B + \bar{A}C + BC$$

Logic Circuit:



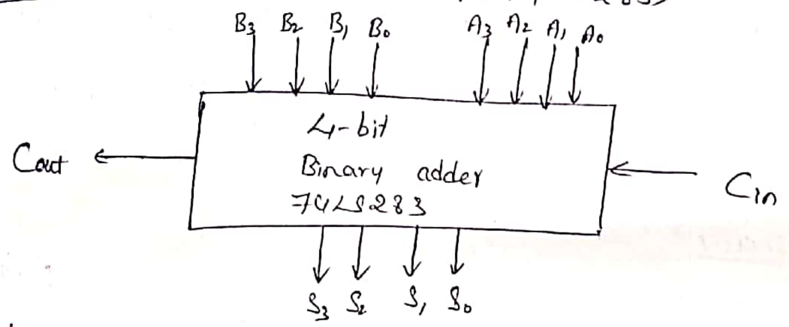
4-bit Binary Adder:

In order to add binary numbers with more than one bit, additional full-adders must be employed. A 4-bit, parallel adder can be constructed using 4 full adders Circuits Connected in parallel. Here the full adders are connected in cascade i.e., O/p carry of each adder is connected to the carry i/p of the next higher-order.



It should be noted that either a half adder can be used for the least significant position & the carry input of a full-adder is made '0' because there is no carry into the least significant bit position.

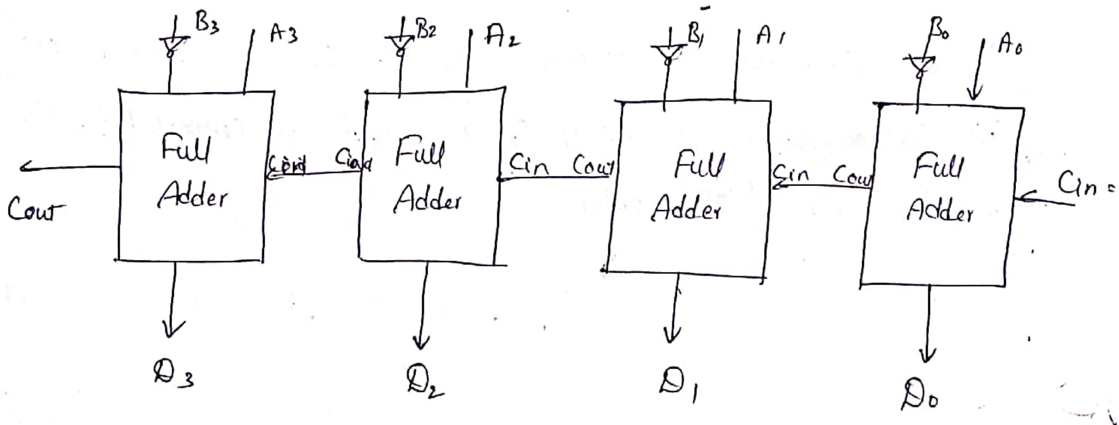
Logic Symbol : (IC 74LS83 / 74LS283)



4-bit binary Subtractor :

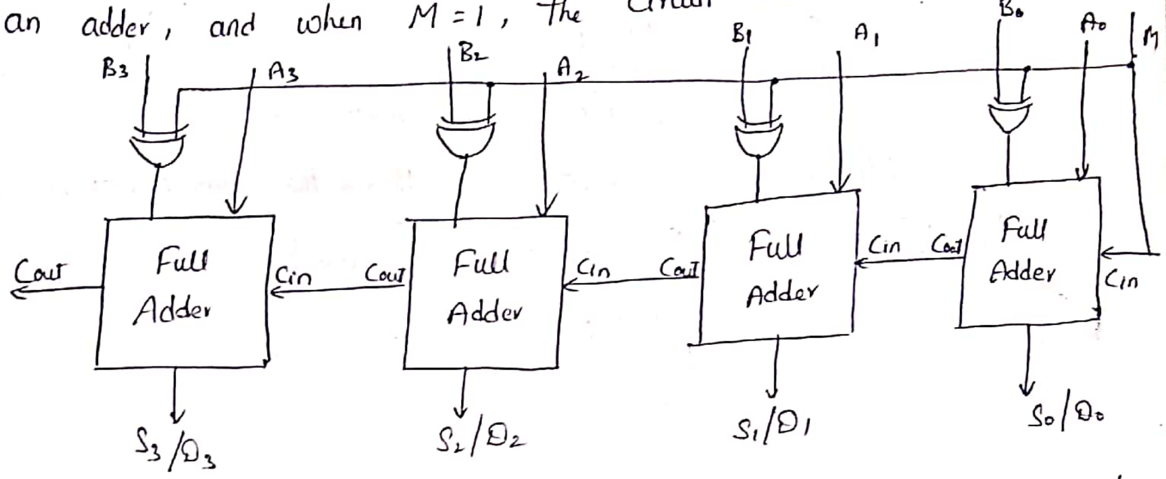
The subtraction of binary numbers can be done most conveniently by means of complements. The subtraction A-B can be done by taking the 2's complement of B and adding it to A. The 2's complement can be obtained by taking 1's complement and adding one to the least significant bit.

The 1's complement can be implemented with inverters and 1 can be added to the sum through the i/p carry.



Adder - Subtractor Circuit :

The addition and subtraction operations can be combined into one circuit with one common binary adder. This is done by including an exclusive - OR gate with each full adder. The mode i/p M controls the operation of the circuit. When M=0, the circuit is an adder, and when M = 1, the circuit becomes a subtractor.



Each exclusive - OR gate receives i/p M and one of the inputs of B. When M=0, we have B⊕0 = B. The full-adders receives the value of B, the input carry is 0, and the circuit performs A plus B operation. When M = 1, we have B⊕1 = B̄ & The i/p carry is 1. The B inputs are all complemented & a 1 is added through the input carry. The circuit performs the operation A plus the 2's complement of B i.e., A - B

BCD Adder :

A BCD adder is a circuit that adds two BCD digits and produces a sum digit also in BCD. BCD numbers use 10 digits, 0 to 9 which are represented in the binary form 0000 to 1001.

526 number can be represented as

5 2 6
0101 0010 0110

Sum equals 9 & less with carry 0

```
  6        0110
 +3        0011
 --        ----
  9        1001   ← BCD for 9
```

The addition is carried out as in normal binary addition and the sum is 1001, which is BCD code for 9.

Sum greater than 9 with carry 0

```
  6        0110
 +8        1000
 --        ----
 14        1110   ← Invalid BCD number
```

Whenever sum exceeds '9' then the sum has to be corrected by the addition of six (0110) in the invalid BCD.

```
            1110
          1 0110
       ---------
       0001 0100
         1    4      ← BCD for 14.
```

After addition of 6 carry is produced into the second decimal position.

Sum equals 9 & less with carry 1 :

```
  8        1000
 +9        1001
 --        ----
 17       10001   ← Invalid BCD number
            110
      ---------
      0001 0111   ← BCD for 17.
        1    7
```

Thus to implement BCD adder we require

→ 4-bit binary adder for initial addition

→ Logic circuit to detect sum greater than 9 and

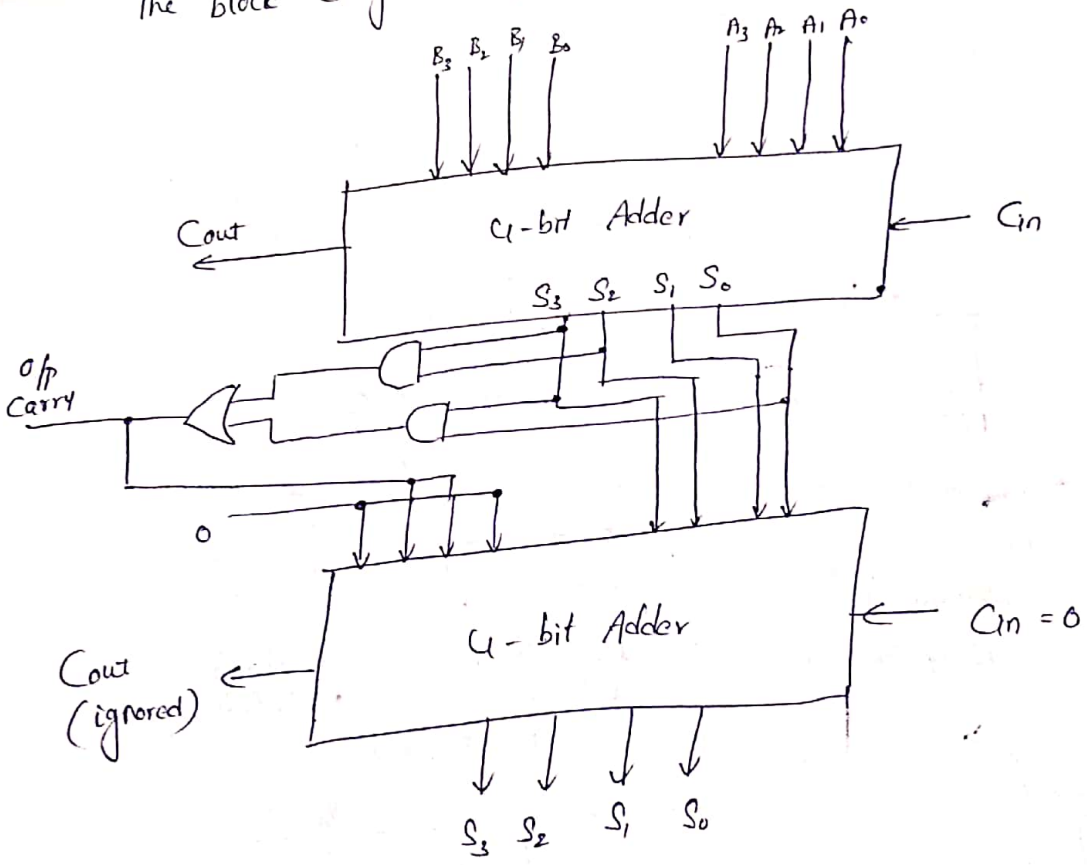
→ One more 4-bit adder to add 0110_2 in the sum if sum is greater than 9 & carry is 1.

Inputs				Outputs
S_3	S_2	S_1	S_0	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

$S_3S_2 \backslash S_1S_0$	$\bar{S_1}\bar{S_0}$	$\bar{S_1}S_0$	S_1S_0	$S_1\bar{S_0}$
$\bar{S_3}\bar{S_2}$				
$\bar{S_3}S_2$				
S_3S_2	1	1	1	1
$S_3\bar{S_2}$			1	1

$$Y = S_3S_2 + S_3S_1$$

The block diagram of BCD adder is shown below



Look

Excess-3 Adder Circuit :

To perform Excess-3 addition we have to

→ Add two Excess-3 numbers

→ If

Carry = 1 ⟶ add 3 to the sum of two digits

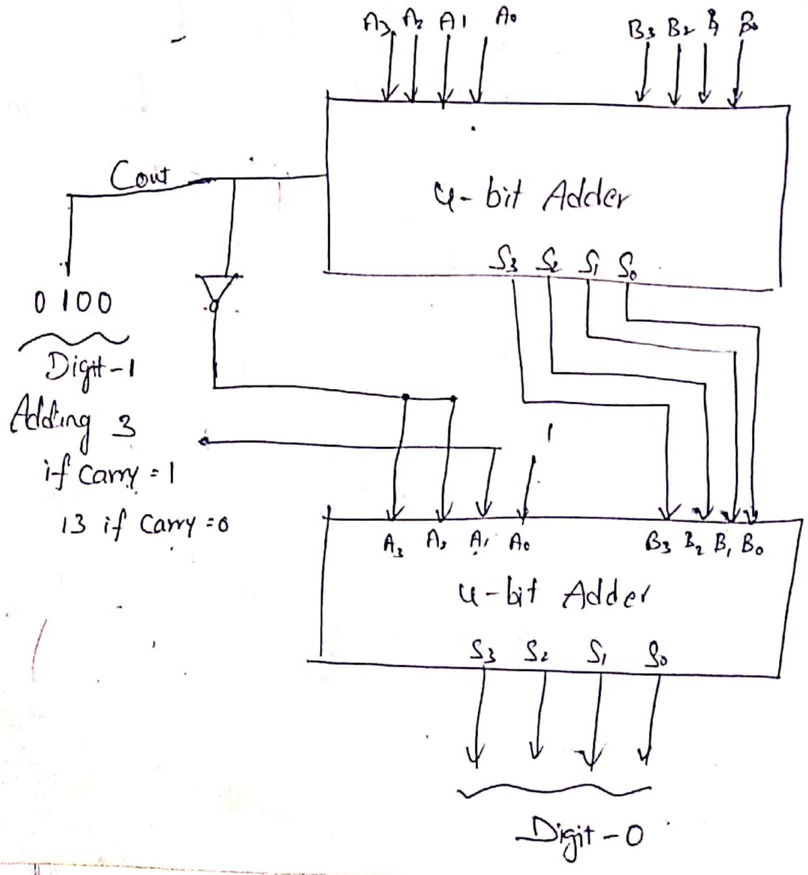
= 0 → Subtract 3 i.e., add 1101 (13 in decimal)

Ex:

```
            1011  → Excess-3 for 8
            1001  → Excess-3 for 6
                                    14
   0001     0100
   0011     0011
   0100     0111
     1        4        → Excess-3 for 14.
  Digit-1  Digit-0
```

Ex:

```
            0100  → Excess-3 for 1
            0101  → Excess-3 for 2
            1001
            1101
ignore → 1  0110. → Excess-3 for 3.
```

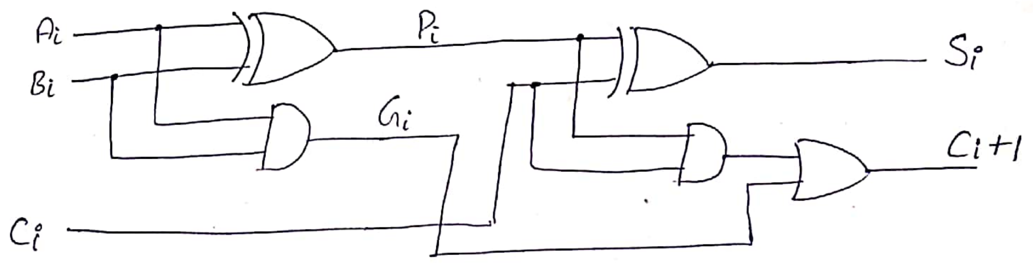


Look ahead Adder Circuit :

In parallel adder the carry output of each full-adder stage is connected to the carry input of the next higher-order stage. Therefore, the sum and carry outputs of any stage cannot be produced until the input carry occurs; this leads to a time delay in the addition process. This delay is known as carry propagation delay.

If each full-adder is considered to have a propagation delay of 30ns, then for 4-bit adder to perform addition it takes 120ns. If the adder were handling 16-bit numbers, the carry propagation delay could be 480 ns.

One method of speeding up this process by eliminating inter stage carry delay is called Look ahead carry addition. It uses two functions : Carry generate and Carry propagate.



$$P_i = A_i \oplus B_i$$

$$G_i = A_i \cdot B_i$$

The output sum and carry can be expressed as

$$S_i = P_i \oplus C_i$$

$$C_{i+1} = G_i + P_i C_i$$

G_i is called a Carry generate

P_i is called Carry propagate.

C_0 = input carry

$C_1 = G_0 + P_0 C_0$

$C_2 = G_1 + P_1 C_1$

$= G_1 + P_1 (G_0 + P_0 C_0)$

$= G_1 + P_1 G_0 + P_0 P_1 C_0$

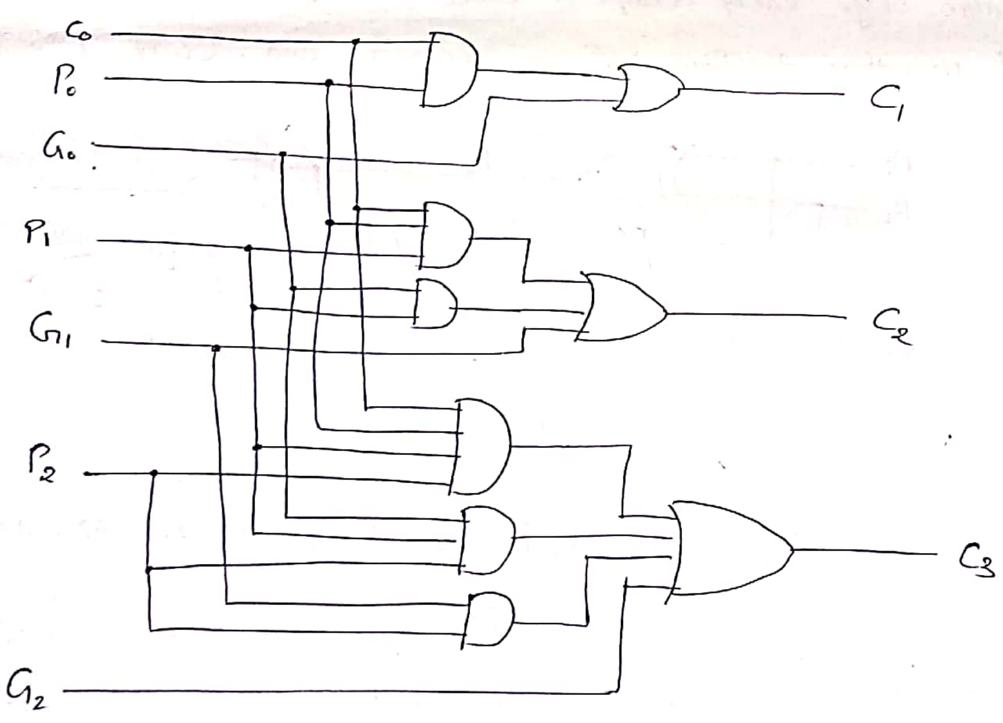
$C_3 = G_2 + P_2 C_2$

$= G_2 + P_2 (G_1 + P_1 G_0 + P_0 P_1 C_0)$

$= G_2 + P_2 G_1 + P_1 P_2 G_0 + P_0 P_1 P_2 C_0$

From the above boolean function it can be seen that C_3 does not have to wait for C_2 & C_1 to propagate; in fact C_3 is propagated at the same time as C_1 and C_2.

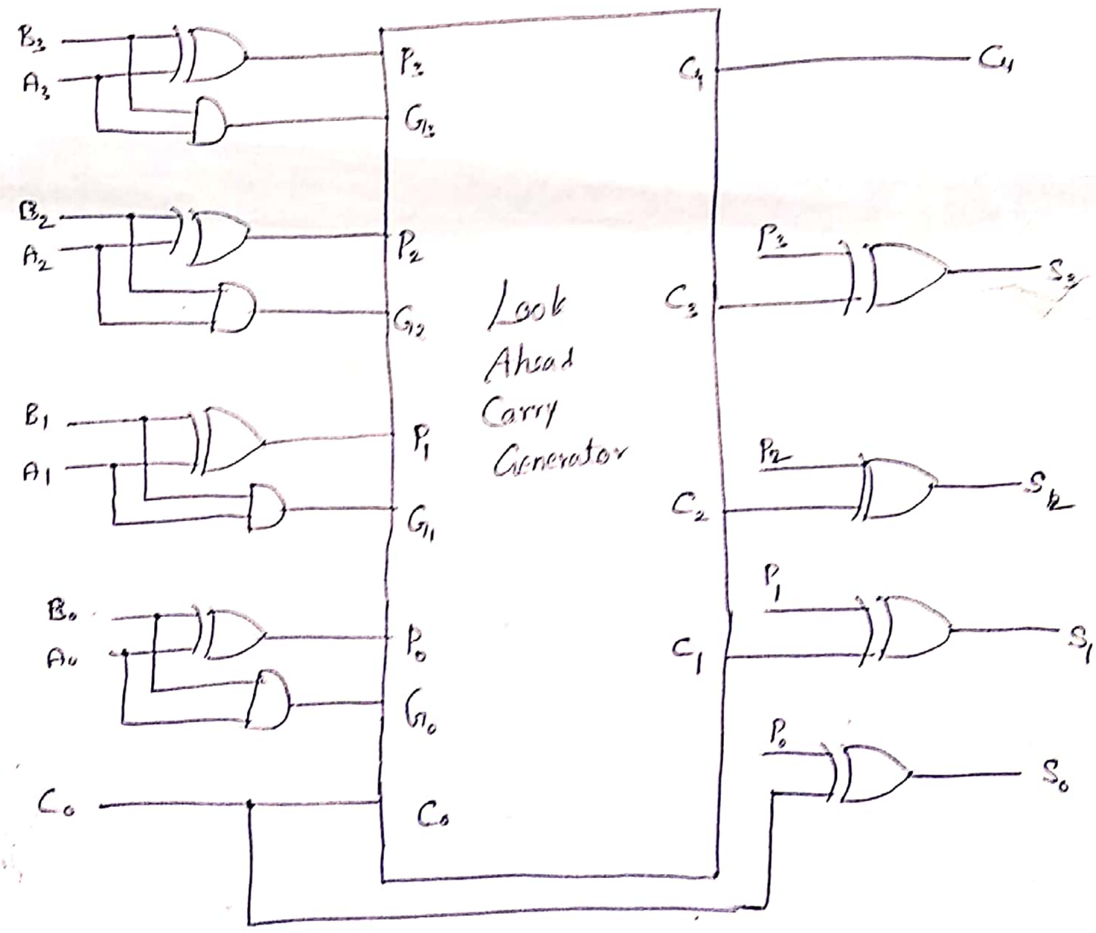
<u>Logic diagram of look-ahead carry generator:</u>



Using a look ahead carry generator we can easily contruct a 4-bit parallel adder with look ahead carry scheme.

Each sum output requires two exclusive - OR gates. The o/p of first exclusive - OR gate generates P_i, and the AND gate generates G_i. The carries are generated using look ahead carry generator and applied as inputs to the second exclusive - OR gate.

Other inputs to Ex-OR gate is P_i. Thus second Ex-OR gate generates sum outputs. Each o/p is generated after a delay of two levels of gate. Thus o/p's S_0 through S_4 have equal propagation delay times.



Degenerative forms :

It will be instructive from a theoretical point to find out how many two-level Combinations of gates are possible. We Consider four types of gates. AND, OR, NAND, and NOR.

If we assign one type of gate for the first level and one type for the second level, we find that there are 16 possible Combinations of two-level forms. Eight of these Combinations are said to be degenerative forms, because they degenerate to a single operation. This Can be seen from a Circuit with AND gates in the first level and an AND gate in the second level. The o/p of the Circuit is merely the AND function of all input variables.