Chapter 05: Combinational Logic

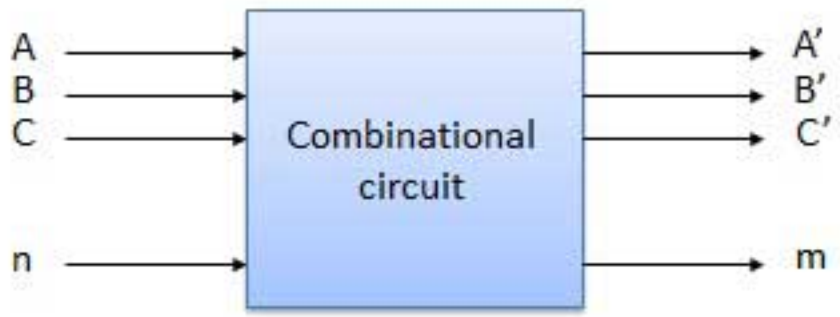
Hari K.C.

Department of software Engineering

Gandaki College of Engineering and Science

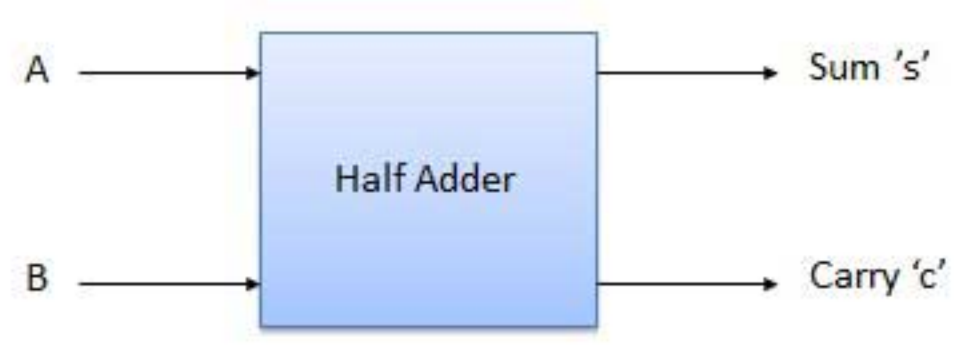
Combinational Logic circuits

- Combinational circuit is a circuit in which we combine the different gates in the circuit, for example encoder, decoder, multiplexer and demultiplexer.
- Some of the characteristics of combinational circuits are following −
- 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.

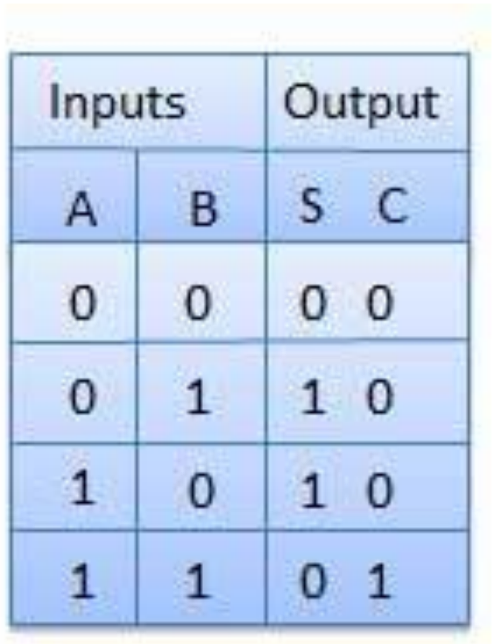


Half Adder

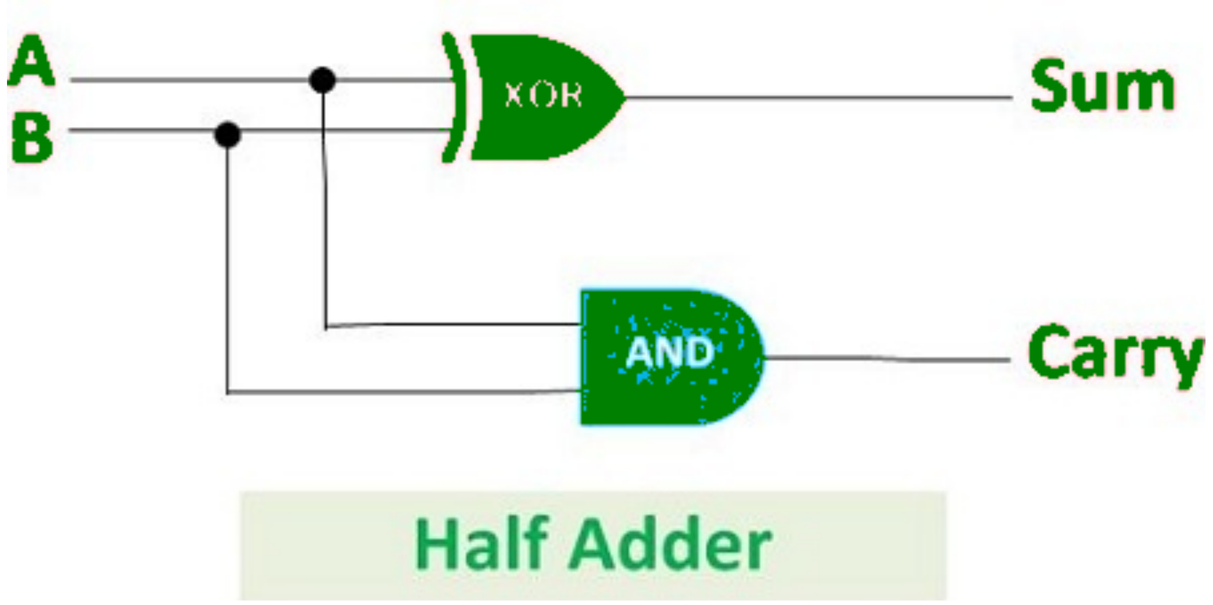
- Half adder is a combinational logic circuit with two inputs and two outputs.
- The half adder circuit is designed to add two single bit binary number A and B. It is the basic building block for addition of two **single** bit numbers.
- This circuit has two outputs **carry** and **sum**.



Truth table and circuit of Half adder



Inputs		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

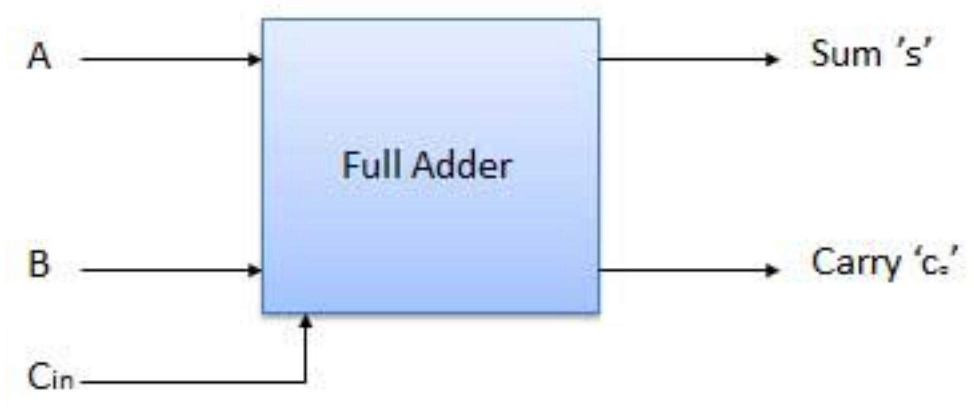


Sum = A XOR B

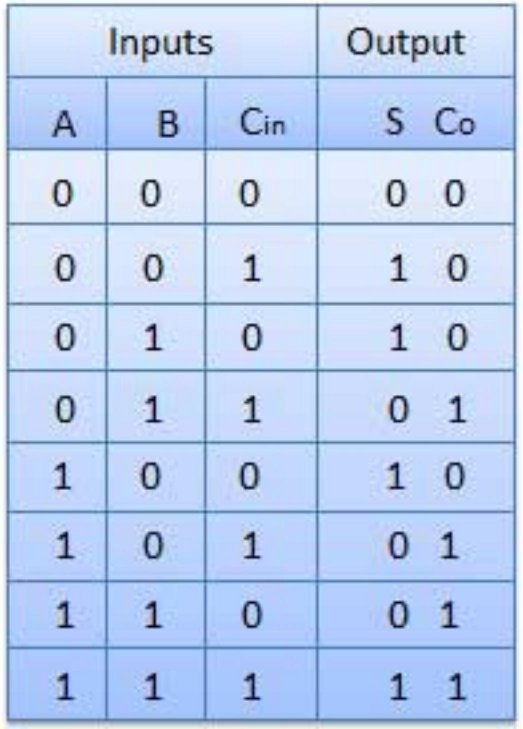
Carry = A AND B

Full Adder

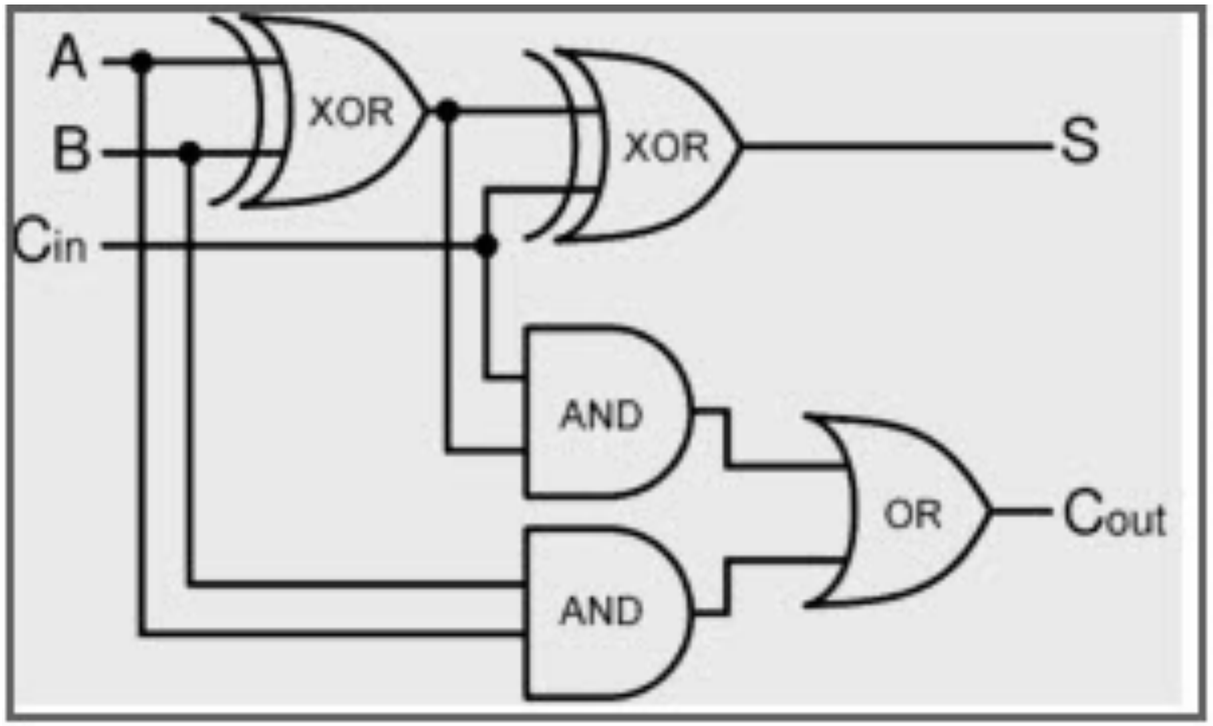
- Full adder is developed to overcome the drawback of Half Adder circuit.
- It can add two one-bit numbers A and B, and carry c.
- The full adder is a three input and two output combinational circuit.



Truth table and circuit of Full adder



Inputs			Output	
A	B	C_{in}	S	C_o
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



SUM = (A XOR B) XOR Cin = (A ⊕ B) ⊕ Cin

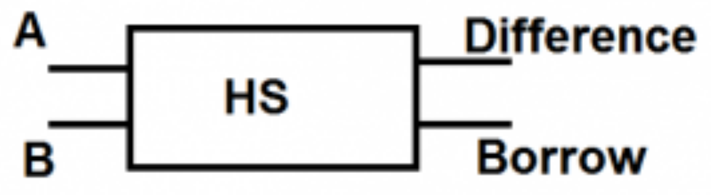
CARRY-OUT = A AND B OR Cin(A XOR B) = A.B + Cin(A ⊕ B)

Difference between Half adder and full adder :

S.NO.	HALF ADDER	FULL ADDER
1	Half Adder is combinational logic circuit which adds two 1-bit digits. The half adder produces a sum of the two inputs.	Full adder is combinational logical circuit that performs an addition operation on three one-bit binary numbers. The full adder produces a sum of the three inputs and carry value.
2	Previous carry is not used.	Previous carry is used.
3	In Half adder there are two input bits (A, B).	In full adder there are three input bits (A, B, C-in).
4	Logical Expression for half adder is : $S=a \oplus b$; $C=a*b$.	Logical Expression for Full adder is : $S=a \oplus b \oplus Cin$; $Cout=(a*b)+(Cin*(a \oplus b))$.
5	It consists of one EX-OR gate and one AND gate.	It consists of two EX-OR, two AND gate and one OR gate.
6	It is used in Calculators, computers, digital measuring devices etc.	It is used in Multiple bit addition, digital processors etc.

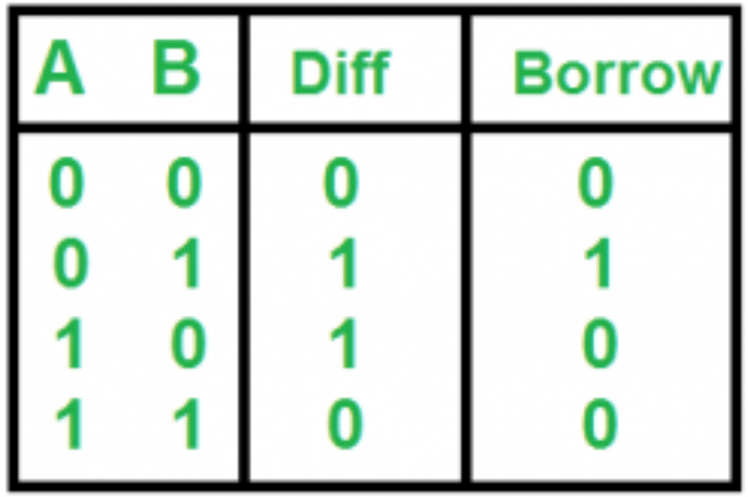
Half subtractor

- Half subtractor is the most essential combinational logic circuit which is used in digital electronics.
- Basically, this is an electronic device or in other terms, we can say it as a logic circuit.
- This circuit is used to perform two binary digits subtraction.

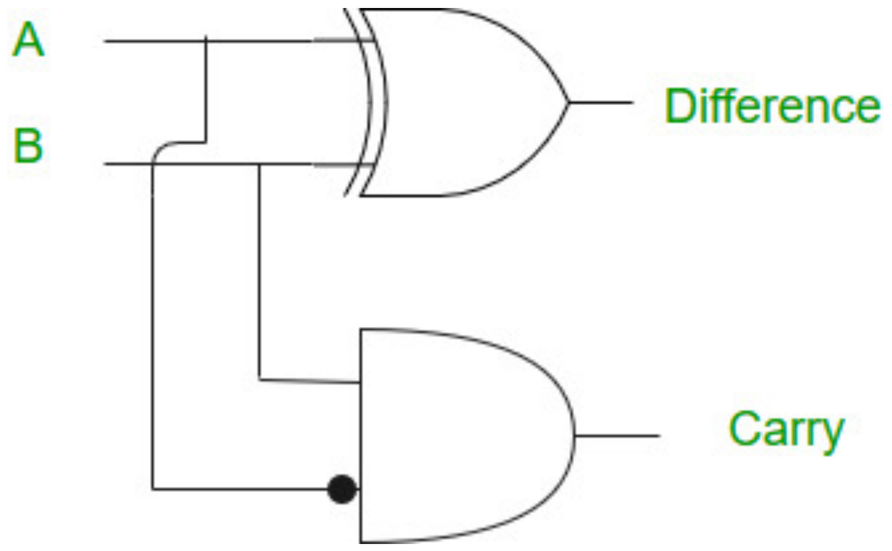


Half Subtractor

Truth Table of Half subtractor



A	B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



Logical Expression

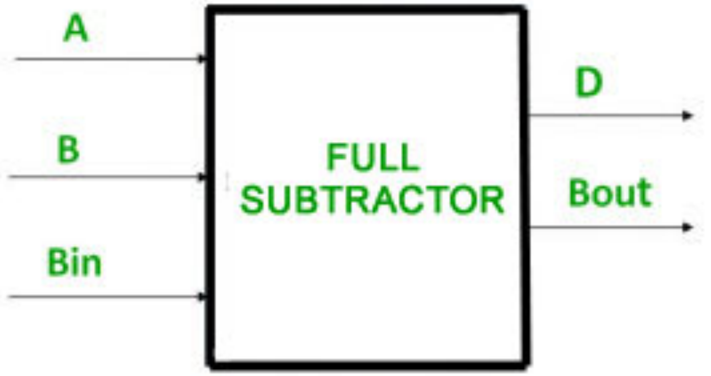
Difference = A XOR B

Borrow =

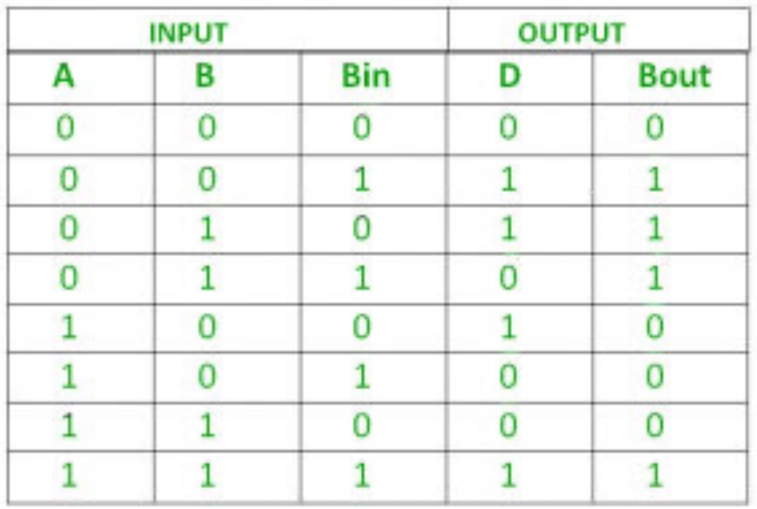
$\overline{A}B$

Full Subtractor

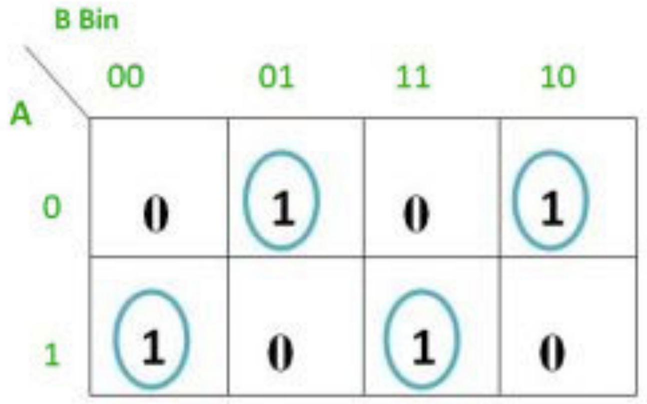
- A full subtractor is a **combinational circuit** that performs subtraction of two bits, one is minuend and other is subtrahend, taking into account borrow of the previous adjacent lower minuend bit.
- This circuit **has three inputs and two outputs**.
- The three inputs A, B and Bin, denote the minuend, subtrahend, and previous borrow, respectively.
- The two outputs, D and Bout represent the difference and output borrow, respectively.



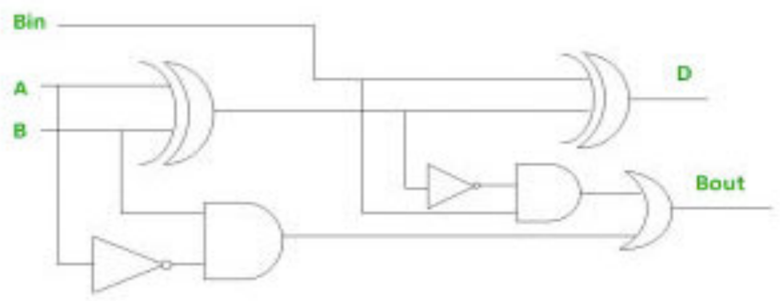
Truth Table and circuit of Full subtractor

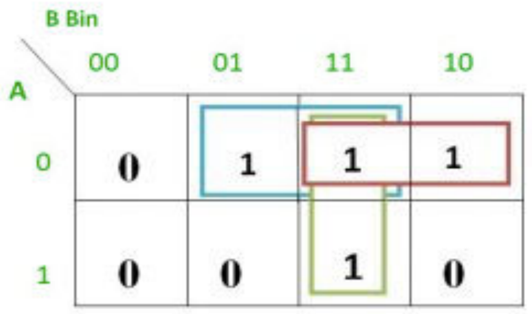


INPUT			OUTPUT	
A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



D=A’B’Bin + AB’Bin’ + A’BBin’ + ABBin





Bout=A’Bin + A’B + BBin

Logical expression for difference –

```
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
```

Logical expression for borrow –

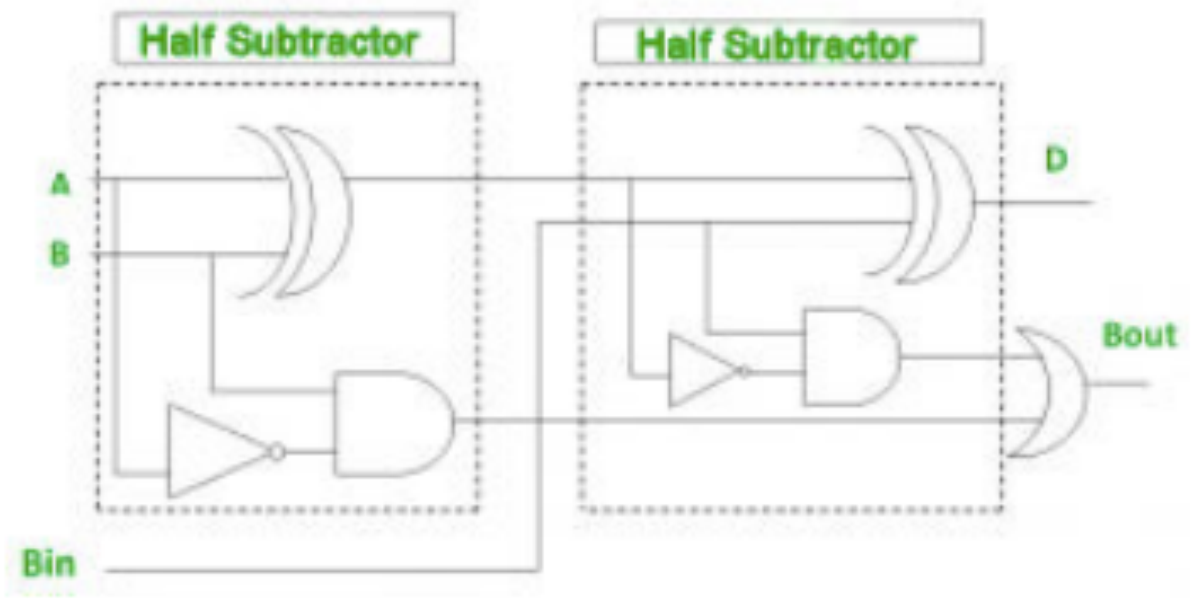
```
Bout = A’B’Bin + A’BBin’ + A’BBin + ABBin
      = A’B’Bin +A’BBin’ + A’BBin + A’BBin + A’BBin + ABBin
      = A’Bin(B + B’) + A’B(Bin + Bin’) + BBin(A + A’)
      = A’Bin + A’B + BBin

OR

Bout = A’B’Bin + A’BBin’ + A’BBin + ABBin
      = Bin(AB + A’B’) + A’B(Bin + Bin’)
      = Bin( A XNOR B) + A’B
      = Bin (A XOR B)’ + A’B
```

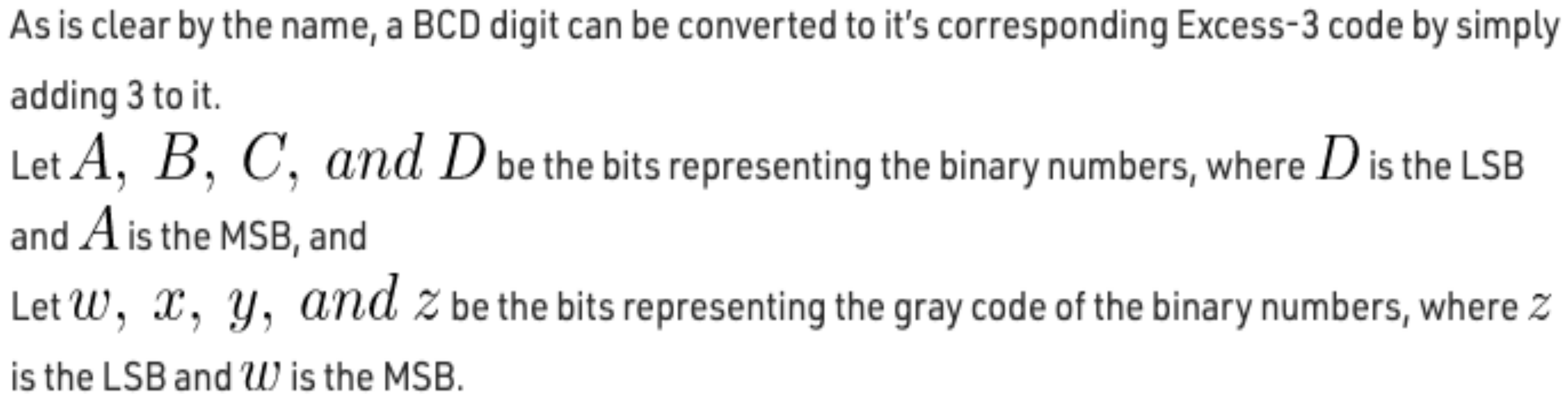
Implementation of Full Subtractor using Half Subtractors –

2 Half Subtractors and an OR gate is required to implement a Full Subtractor.



CODE CONVERSION CIRCUIT

- **Converting BCD(8421) to Excess-3**

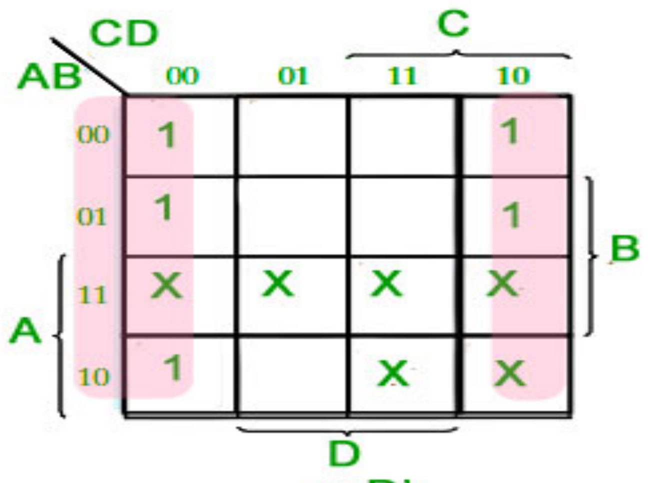


As is clear by the name, a BCD digit can be converted to it's corresponding Excess-3 code by simply adding 3 to it.
Let $A,\ B,\ C,\ and\ D$ be the bits representing the binary numbers, where D is the LSB and A is the MSB, and
Let $w,\ x,\ y,\ and\ z$ be the bits representing the gray code of the binary numbers, where z is the LSB and w is the MSB.

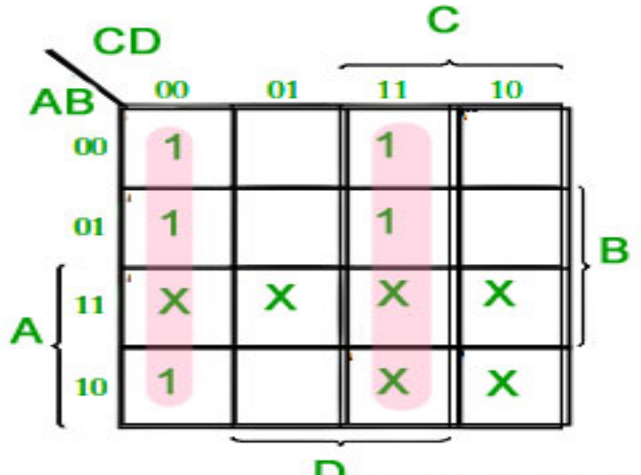
The truth table for the conversion is given below. The X's mark don't care conditions.

BCD(8421)				Excess-3			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

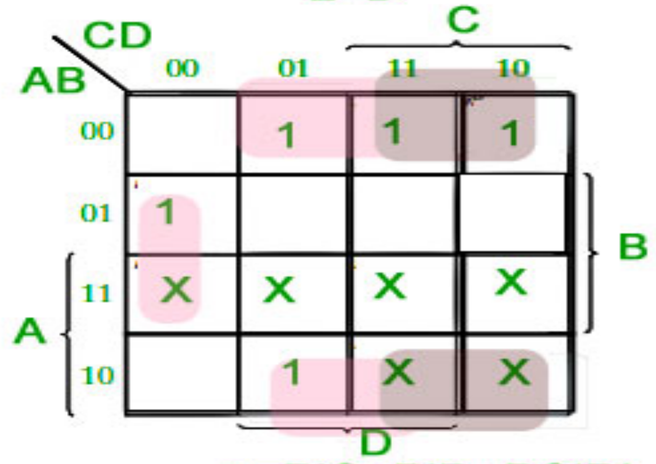
- To find the corresponding digital circuit, we will use the K-Map technique for each of the Excess-3 code bits as output with all of the bits of the BCD number as input.



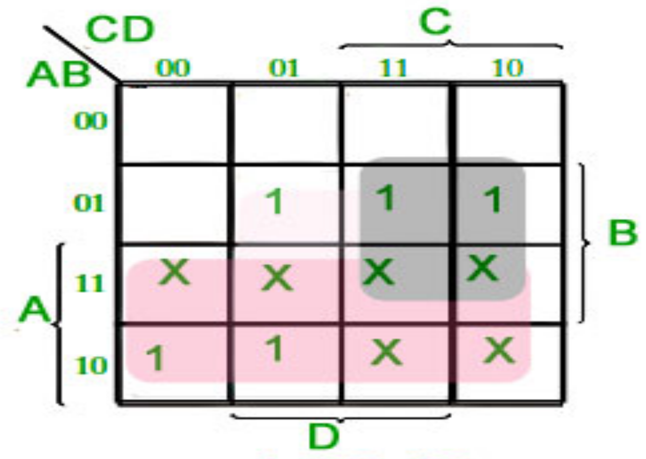
z=D'



y=CD+C'D'



x=B'C+B'D+BC'D'



w=A+BC+BD

Corresponding minimized Boolean expressions for Excess-3 code bits –

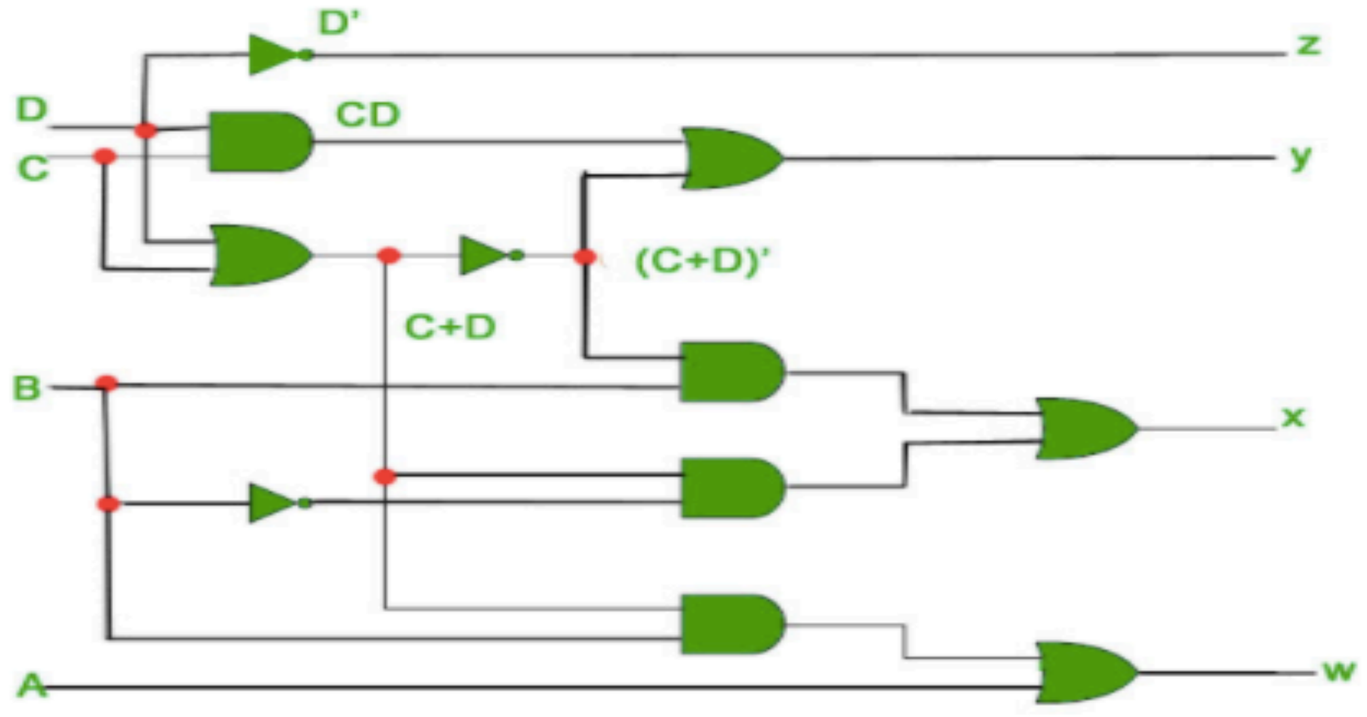
$$w = A + BC + BD$$

$$x = B'C + B'D + BC'D'$$

$$y = CD + C'D'$$

$$z = D'$$

The corresponding digital circuit-



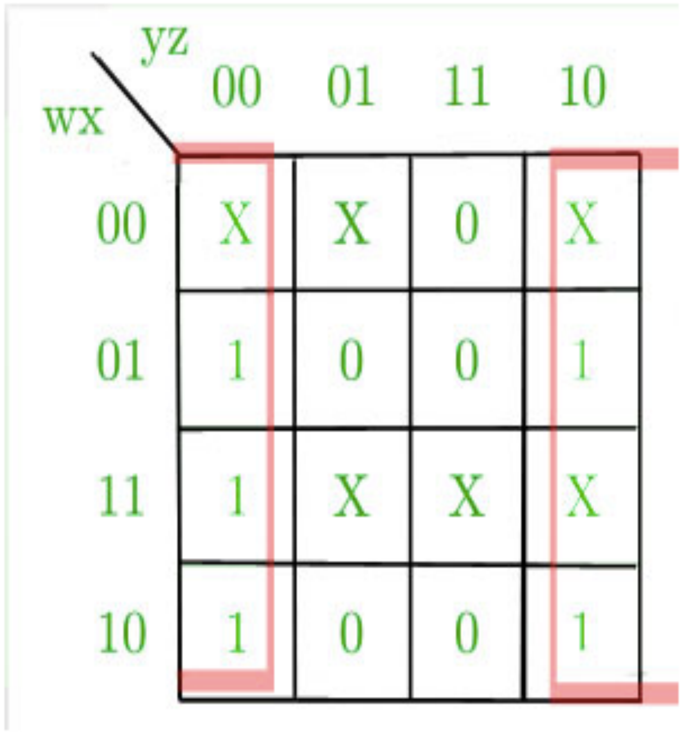
Converting Excess-3 to BCD(8421)

Excess-3 code can be converted back to BCD in the same manner.
Let $A,\ B,\ C,\ and\ D$ be the bits representing the binary numbers, where D is the LSB and A is the MSB, and
Let $w,\ x,\ y,\ and\ z$ be the bits representing the gray code of the binary numbers, where z is the LSB and w is the MSB.

The truth table for the conversion is given below. The X's mark don't care conditions.

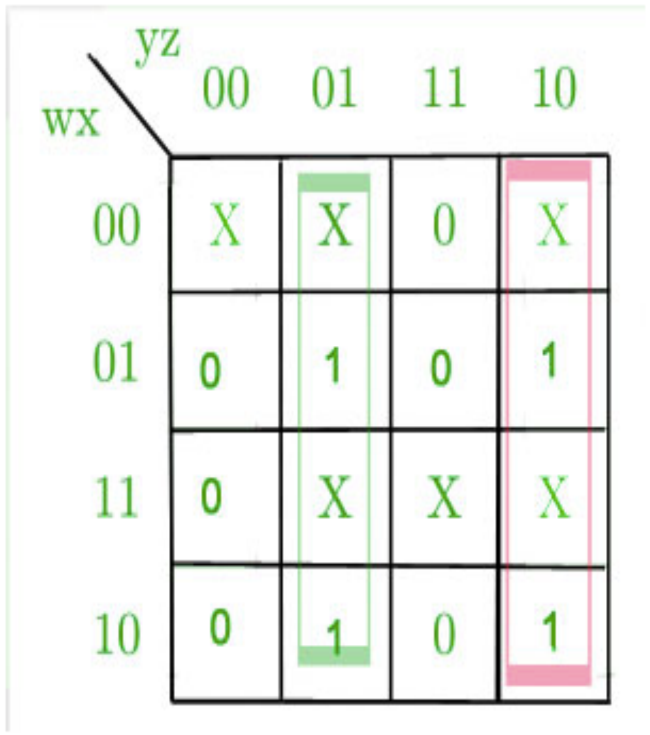
Excess-3				BCD			
w	x	y	z	A	B	C	D
0	0	0	0	X	X	X	X
0	0	0	1	X	X	X	X
0	0	1	0	X	X	X	X
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

K-Map for D-



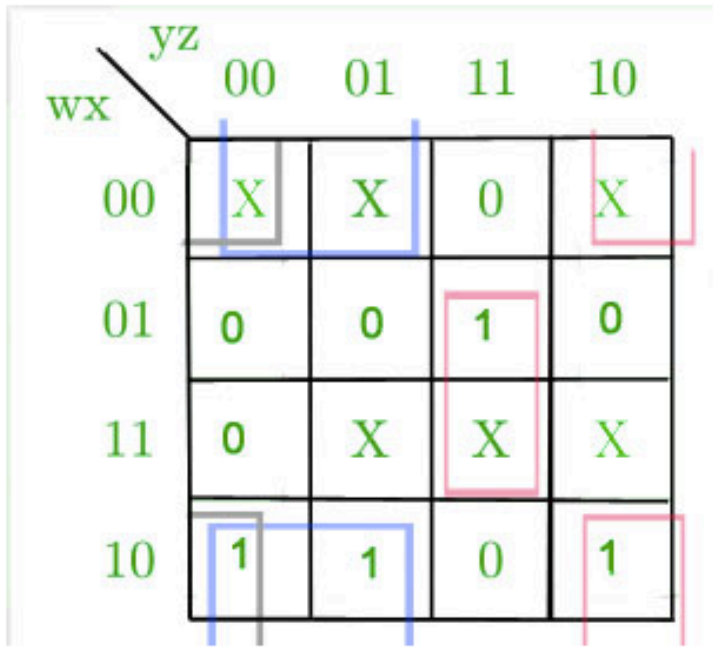
wx \ yz	00	01	11	10
00	X	X	0	X
01	1	0	0	1
11	1	X	X	X
10	1	0	0	1

K-Map for C-



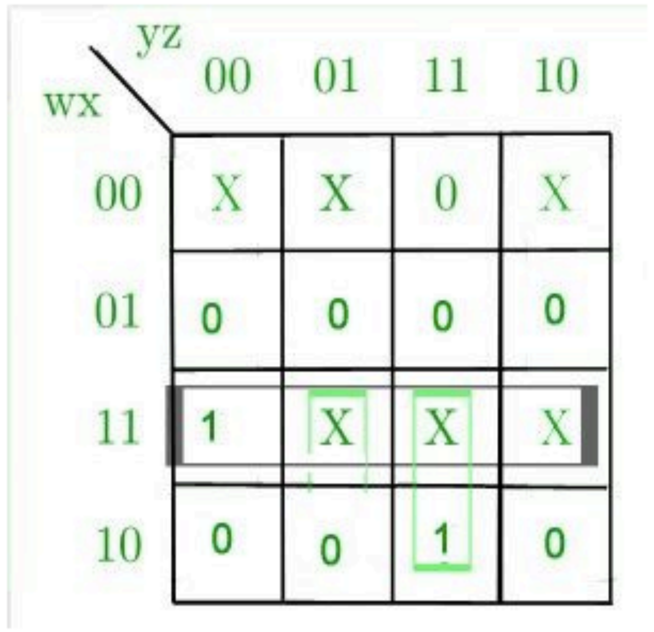
wx \ yz	00	01	11	10
00	X	X	0	X
01	0	1	0	1
11	0	X	X	X
10	0	1	0	1

K-Map for B-



wx yz	00	01	11	10
00	X	X	0	X
01	0	0	1	0
11	0	X	X	X
10	1	1	0	1

K-Map for A-



wx yz	00	01	11	10
00	X	X	0	X
01	0	0	0	0
11	1	X	X	X
10	0	0	1	0

Corresponding minimized boolean expressions for Excess-3 code bits –

$$A = wx + wyz$$

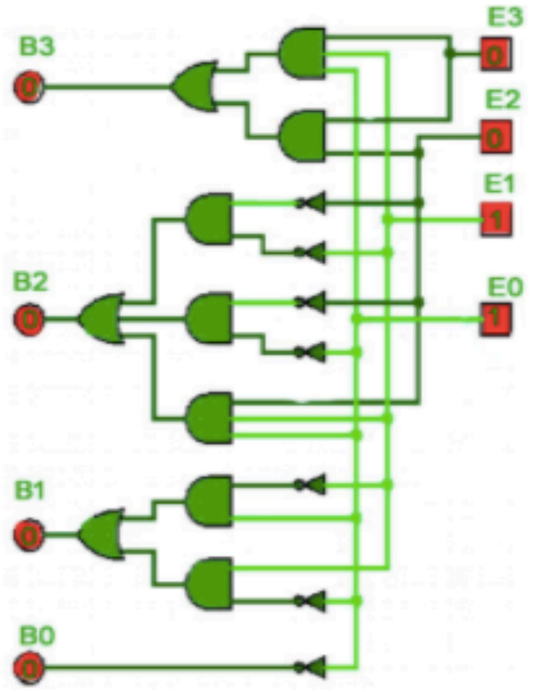
$$B = x'y' + x'z' + xyz$$

$$C = y'z + yz'$$

$$D = z'$$

The corresponding digital circuit –

Here $E_3,\ E_2,\ E_1,\ and\ E_0$ correspond to $w,\ x,\ y,\ and\ z$ and $B_3,\ B_2,\ B_1,\ and\ B_0$ correspond to $A,\ B,\ C,\ and\ D$.



Binary to Gray Code Conversion circuit

Gray Code system is a binary number system in which every successive pair of numbers differs in only one bit. It is used in applications in which the normal sequence of binary numbers generated by the hardware may produce an error or ambiguity during the transition from one number to the next.

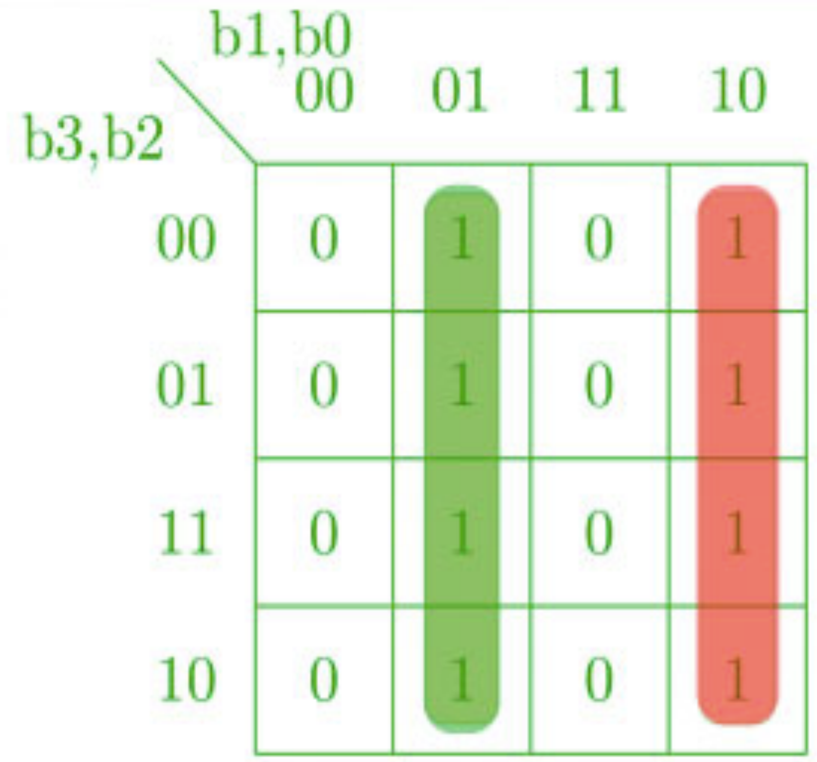
Let $b_0,\ b_1,\ b_2\ ,\ and\ b_3$ be the bits representing the binary numbers, where b_0 is the LSB and b_3 is the MSB, and
Let $g_0,\ g_1,\ g_2\ ,\ and\ g_3$ be the bits representing the gray code of the binary numbers, where g_0 is the LSB and g_3 is the MSB.

The truth table for the conversion is-

Binary				Gray Code			
b_3	b_2	b_1	b_0	g_3	g_2	g_1	g_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

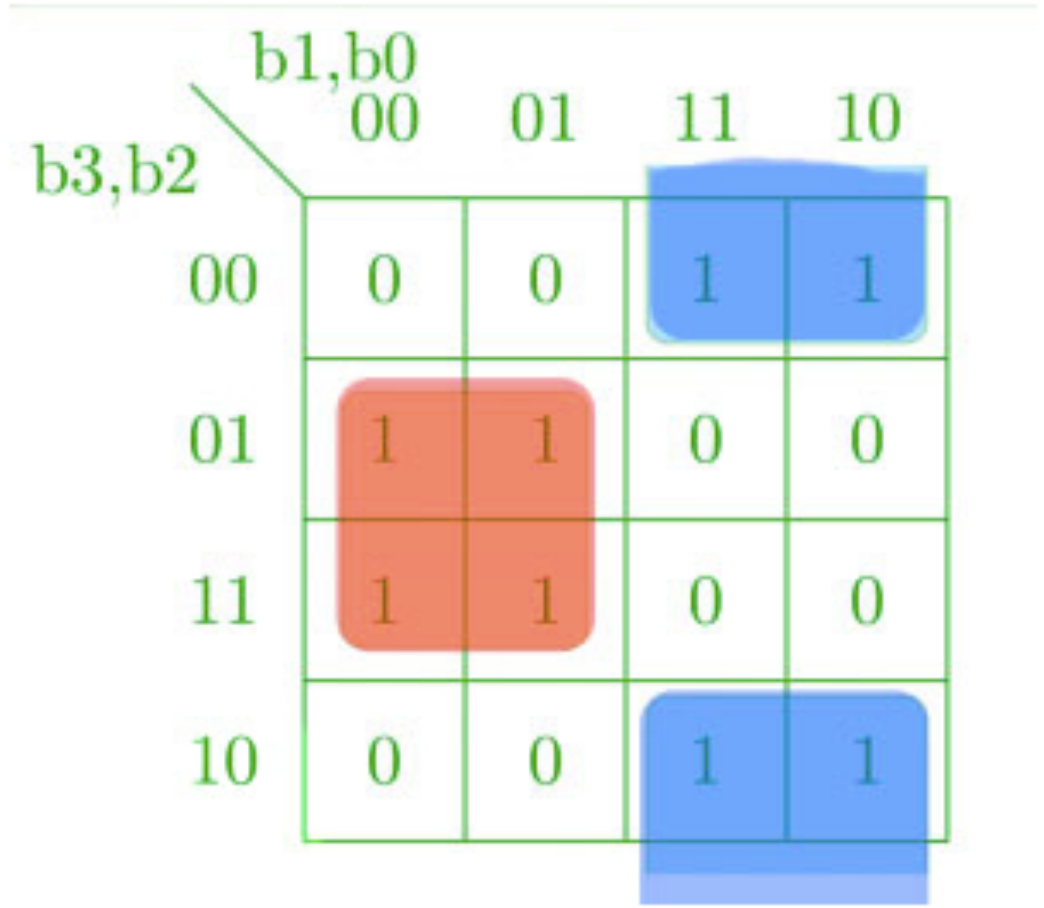
To find the corresponding digital circuit, we will use the K-Map technique for each of the gray code bits as output with all of the binary bits as input.

K-map for g_0-



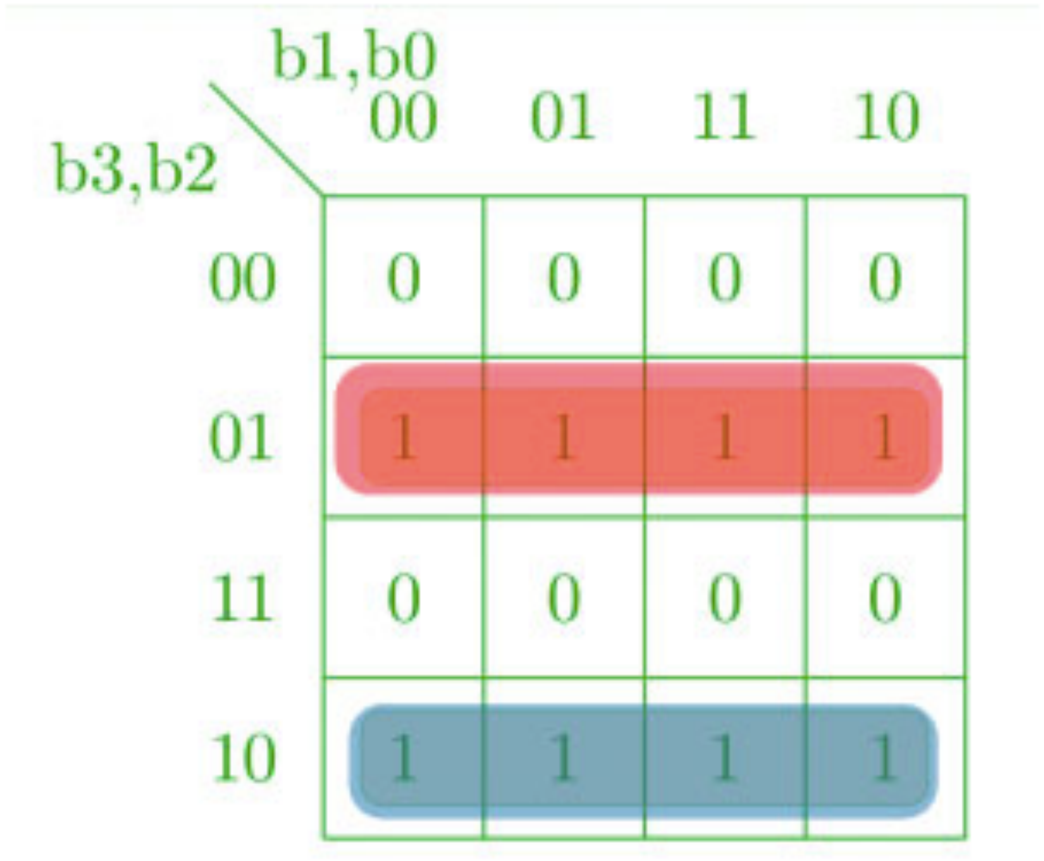
b3,b2 \ b1,b0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

K-map for g_1-



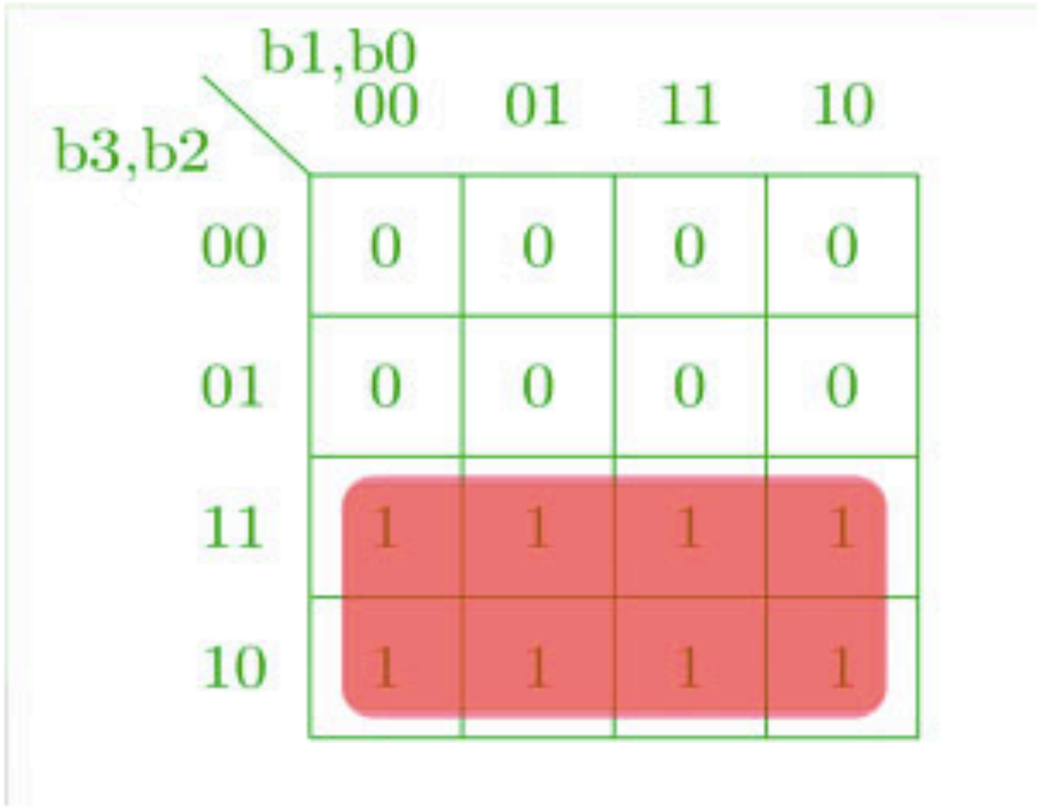
b3,b2 \ b1,b0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

K-map for g_2-



b3,b2 \ b1,b0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

K-map for g_3-



b3,b2 \ b1,b0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

Corresponding minimized boolean expressions for gray code bits –

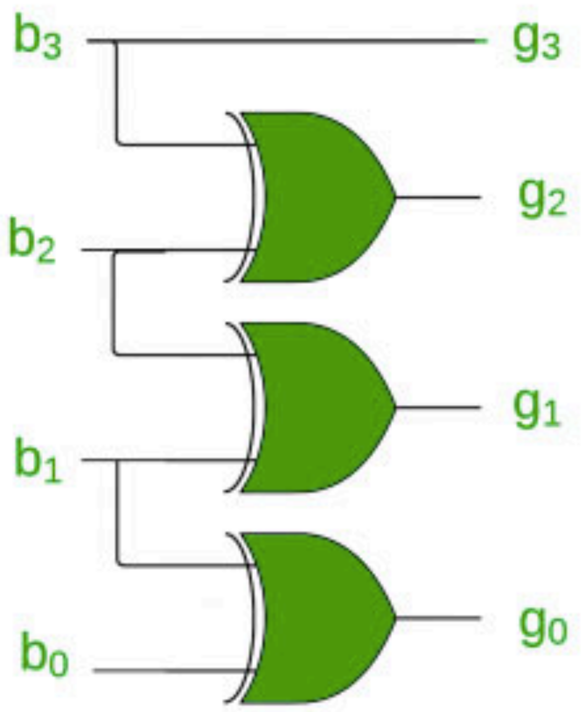
$$g_0 = b_0b_1' + b_1b_0' = b_0 \oplus b_1$$

$$g_1 = b_2b_1' + b_1b_2' = b_1 \oplus b_2$$

$$g_2 = b_2b_3' + b_3b_2' = b_2 \oplus b_3$$

$$g_3 = b_3$$

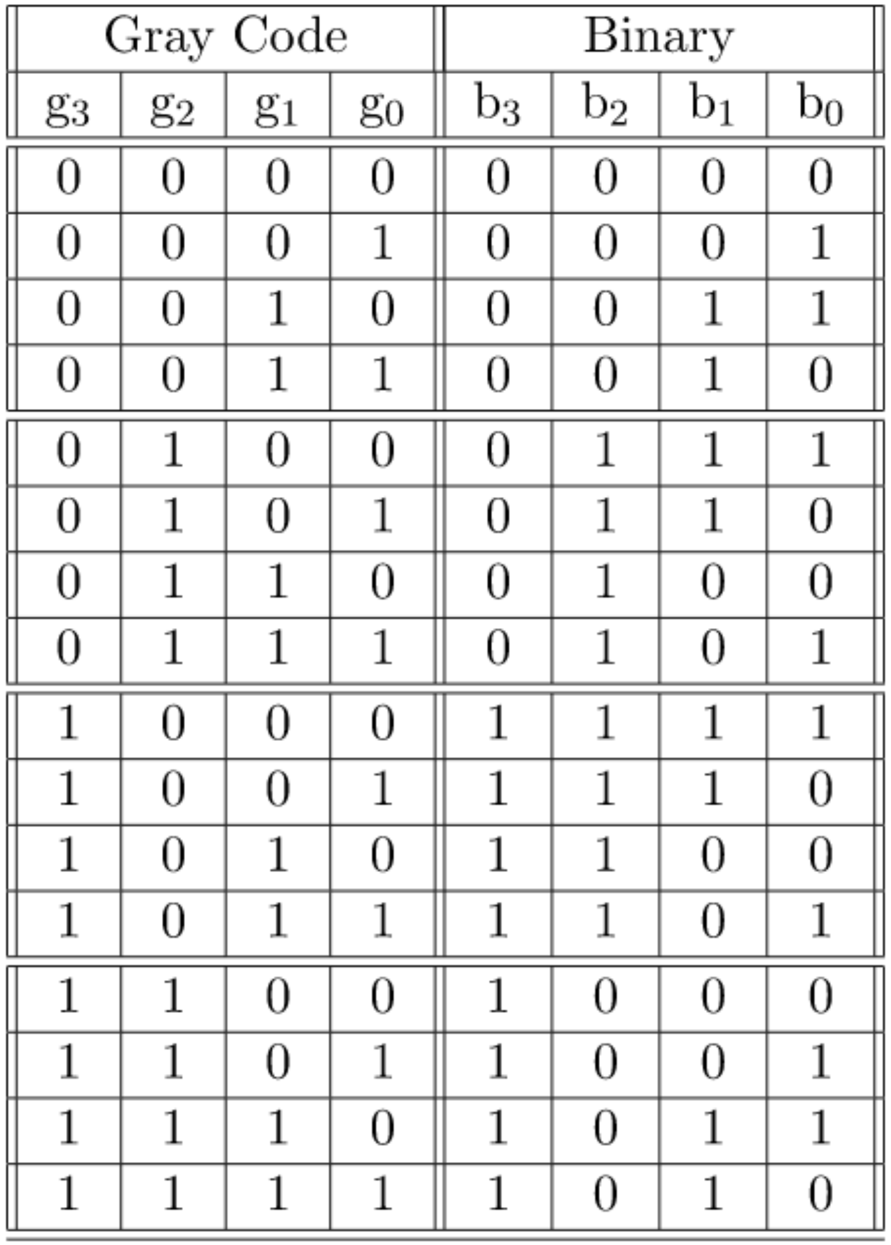
The corresponding digital circuit –



Gray code to Binary conversion circuit

Converting gray code back to binary can be done in a similar manner.
Let $b_0,\ b_1,\ b_2\ ,\ and\ b_3$ be the bits representing the binary numbers, where b_0 is the LSB and b_3 is the MSB, and
Let $g_0,\ g_1,\ g_2\ ,\ and\ g_3$ be the bits representing the gray code of the binary numbers, where g_0 is the LSB and g_3 is the MSB.

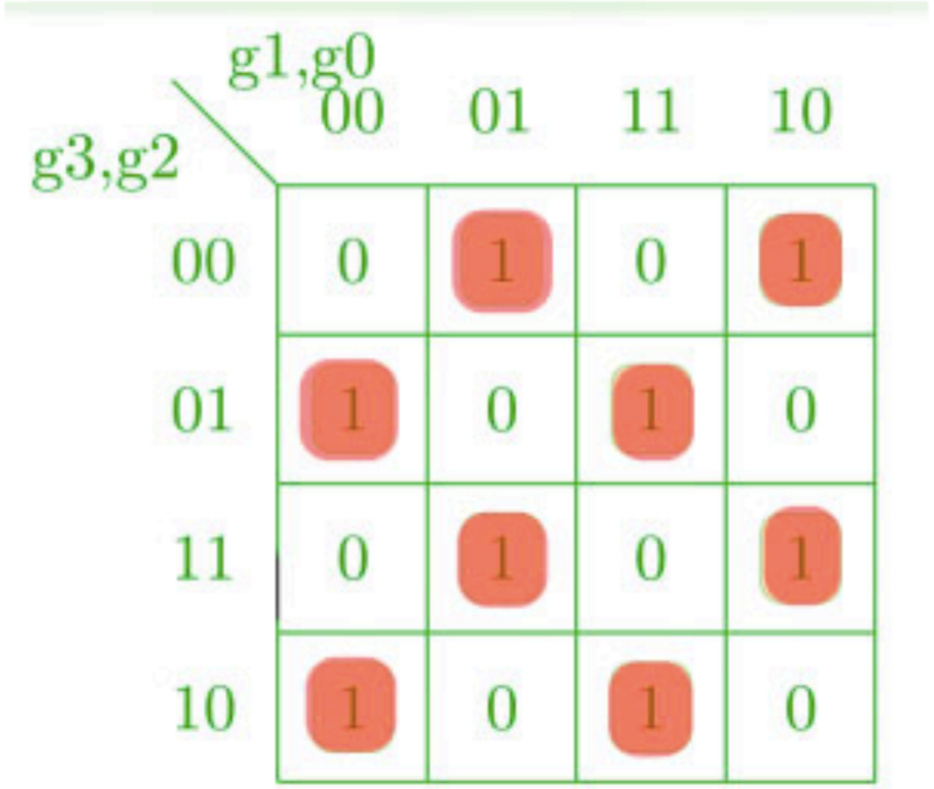
Truth table-



Gray Code				Binary			
g_3	g_2	g_1	g_0	b_3	b_2	b_1	b_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

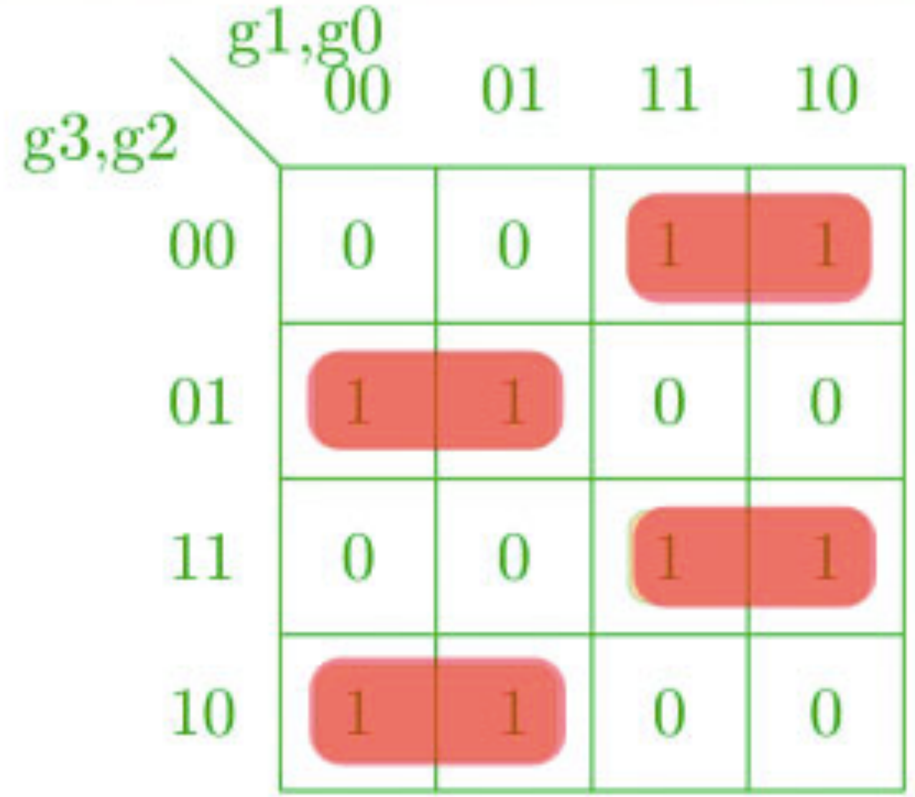
Using K-map to get back the binary bits from the gray code –

K-map for b_0-



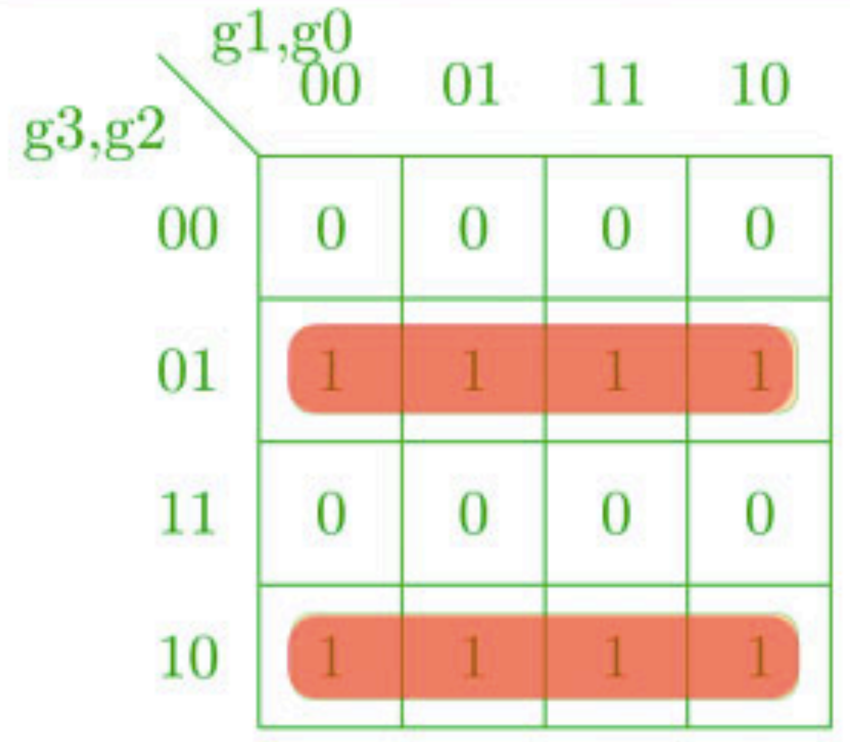
g3,g2 \ g1,g0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

K-map for b_1-



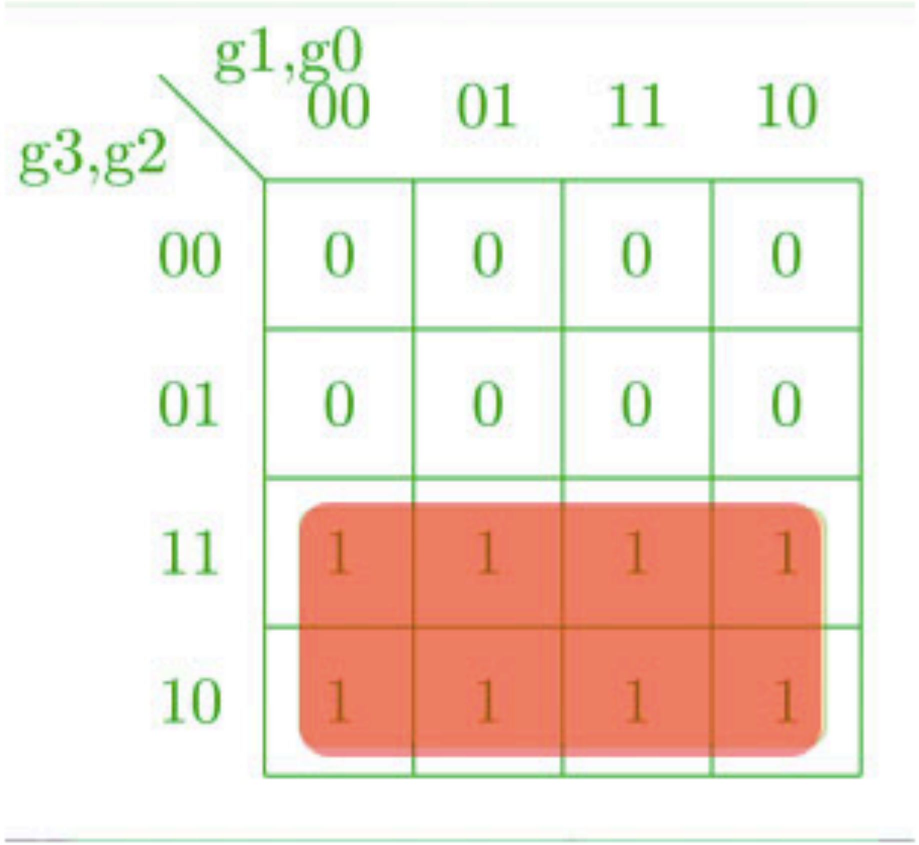
g3,g2 \ g1,g0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

K-map for b_2-



g3,g2 \ g1,g0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

K-map for b_3-

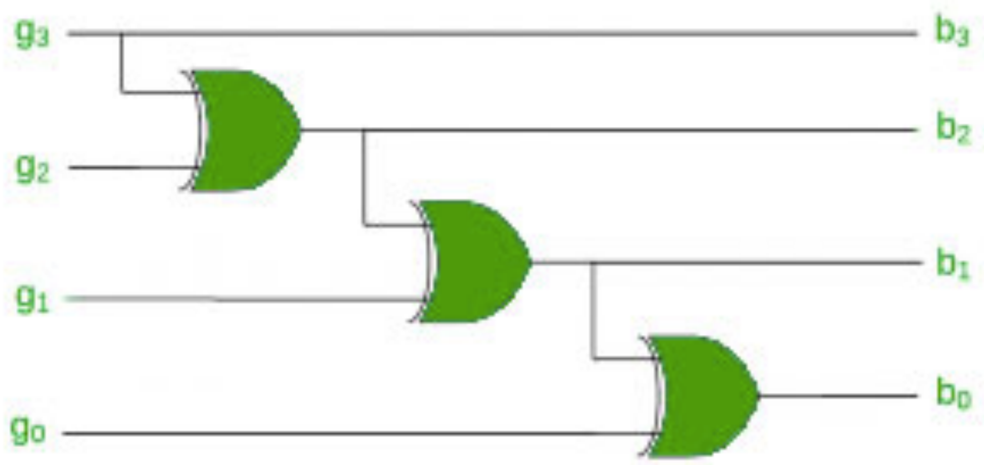


g3,g2 \ g1,g0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

Corresponding Boolean expressions –

$$
\begin{aligned}
b_0 &= g_3'g_2'g_1'g_0 + g_3'g_2'g_1g_0' + g_3'g_2g_1'g_0' + g_3'g_2g_1g_0 + g_3g_2'g_1'g_0' + g_3g \\
&\quad + g_3g_2g_1'g_0 + g_3g_2g_1g_0' \\
&= g_3'g_2'(g_1'g_0 + g_1g_0') + g_3'g_2(g_1'g_0' + g_1g_0) + g_3g_2'(g_1'g_0' + g_1g_0) \\
&\quad + g_3g_2(g_1'g_0 + g_1g_0') \\
&= g_3'g_2'(g_0 \oplus g_1) + g_3'g_2(g_0 \odot g_1) + g_3g_2'(g_0 \odot g_1) + g_3g_2(g_0 \oplus g_1 \\
&= (g_0 \oplus g_1)(g_2 \odot g_3) + (g_0 \odot g_1)(g_2 \oplus g_3) \\
&= g_3 \oplus g_2 \oplus g_1 \oplus g_0 \\
b_1 &= g_3'g_2'g_1 + g_3'g_2g_1' + g_3g_2g_1 + g_3g_2'g_1' \\
&= g_3'(g_2'g_1 + g_2g_1') + g_3(g_2g_1 + g_2'g_1') \\
&= g_3'(g_2 \oplus g_1) + g_3(g_2 \odot g_1) \\
&= g_3 \oplus g_2 \oplus g_1 \\
b_2 &= g_3'g_2 + g_3g_2' \\
&= g_3 \oplus g_2 \\
b_3 &= g_3
\end{aligned}
$$

Corresponding digital circuit –



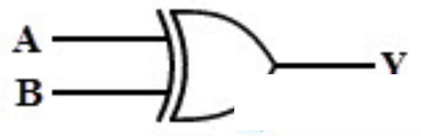
Parity Bit generation and checker circuit

- The parity generating technique is one of the most widely used error detection techniques for the data transmission.
- In digital systems, when binary data is transmitted and processed , data may be subjected to noise so that such noise can alter 0s (of data bits) to 1s and 1s to 0s.
- Hence, **parity bit** is added to the word containing data in order to make number of 1s either even or odd.
- Thus it is used to detect errors , during the transmission of binary data .
- The message containing the data bits along with parity bit is transmitted from transmitter node to receiver node.
- At the receiving end, the number of 1s in the message is counted and if it doesn’t match with the transmitted one, then it means there is an error in the data.

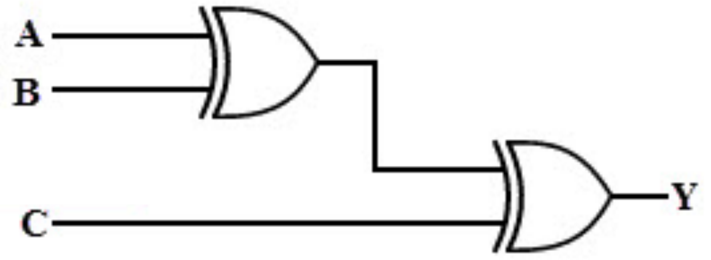
Parity generator and checker

- A parity generator is a combinational logic circuit that generates the parity bit in the transmitter.
- On the other hand, a circuit that checks the parity in the receiver is called parity checker.
- A combined circuit or devices of parity generators and parity checkers are commonly used in digital systems to detect the single bit errors in the transmitted data word.
- The sum of the data bits and parity bits can be even or odd . In even parity, the added parity bit will make the total number of 1s an even amount whereas in odd parity the added parity bit will make the total number of 1s odd amount.

- The basic principle involved in the implementation of parity circuits is that sum of odd number of 1s is always 1 and sum of even number of 1s is always zero.
- Such error detecting and correction can be implemented by using Ex-OR gates (since Ex-OR gate produce zero output when there are even number of inputs).
- To produce two bits sum, one Ex-OR gate is sufficient whereas for adding three bits two Ex-OR gates are required as shown in below figure.



Two Bits



Three Bits

Parity Generator

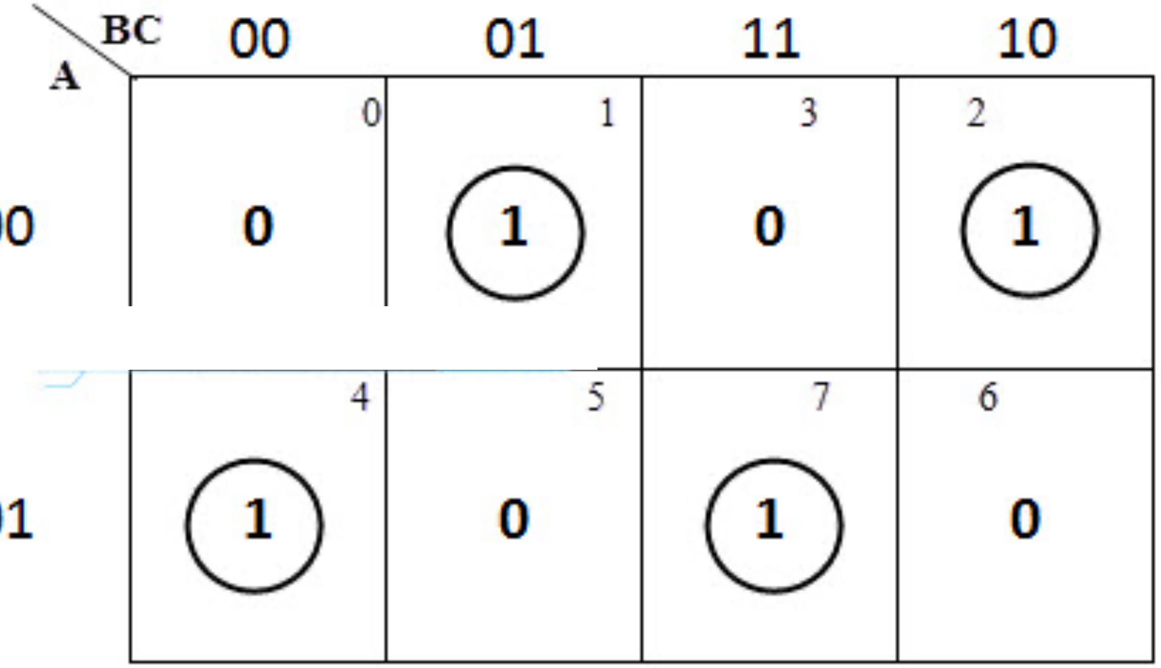
- It is combinational circuit that accepts an n-1 bit stream data and generates the additional bit that is to be transmitted with the bit stream.
- This additional or extra bit is termed as a parity bit.
- In **even parity** bit scheme, the parity bit is ‘**0**’ if there are **even number of 1s** in the data stream and the parity bit is ‘**1**’ if there are **odd number of 1s** in the data stream.
- In **odd parity** bit scheme, the parity bit is ‘**1**’ if there are **even number of 1**s in the data stream and the parity bit is ‘**0**’ if there are **odd number of 1s** in the data stream. Let us discuss both even and odd parity generators.

Even Parity Generator

- Let us assume that a 3-bit message is to be transmitted with an even parity bit.
- Let the three inputs A, B and C are applied to the circuits and output bit is the parity bit P.
- The total number of 1s must be even, to generate the even parity bit P.
- The figure below shows the truth table of even parity generator in which 1 is placed as parity bit in order to make all 1s as even when the number of 1s in the truth table is odd.

3-bit message			Even parity bit generator (P)
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

- The K-map simplification for 3-bit message even parity generator is



A \ BC	00	01	11	10
00	^{0} 0	^{1} 1	^{3} 0	^{2} 1
01	^{4} 1	^{5} 0	^{7} 1	^{6} 0

- From the above truth table, the simplified expression of the parity bit can be written as:

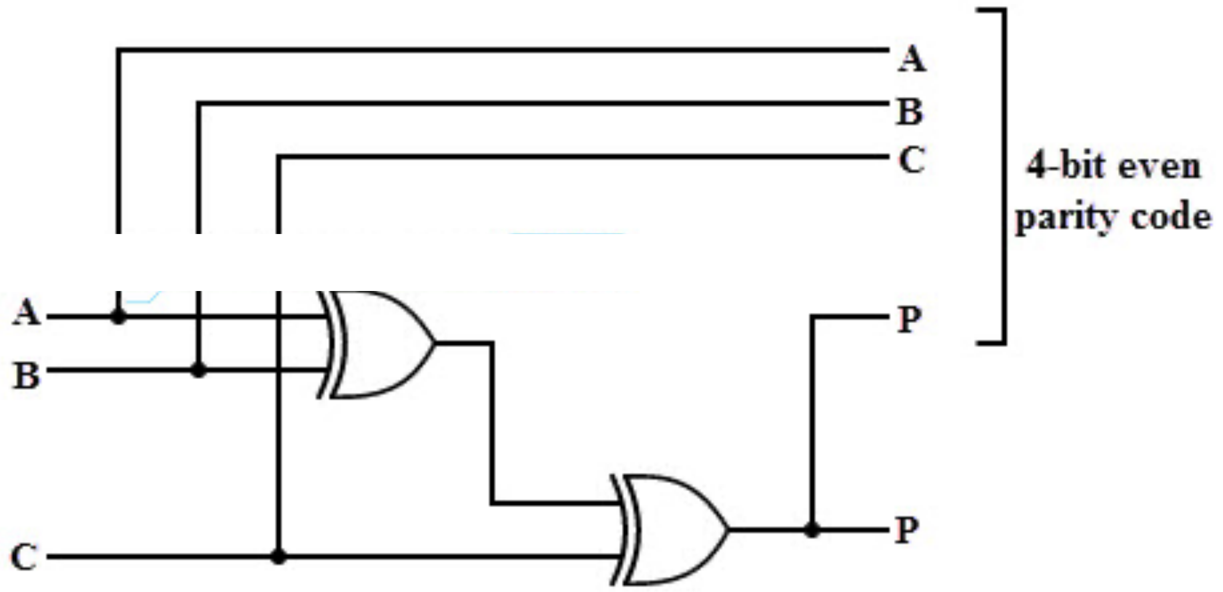
$$P = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,\overline{C} + A\,B\,C$$

$$= \overline{A}\,(\overline{B}\,C + B\,\overline{C}) + A\,(\overline{B}\,\overline{C} + B\,C)$$

$$= \overline{A}\,(B \oplus C) + A\,(\overline{B \oplus C})$$

$$P = A \oplus B \oplus C$$

- The above expression can be implemented by using two Ex-OR gates.
- The logic diagram of even parity generator with two Ex – OR gates is shown below.
- The three bit message along with the parity generated by this circuit which is transmitted to the receiving end where parity checker circuit checks whether any error is present or not.
- To generate the even parity bit for a 4-bit data, three Ex-OR gates are required to add the 4-bits and their sum will be the parity bit.

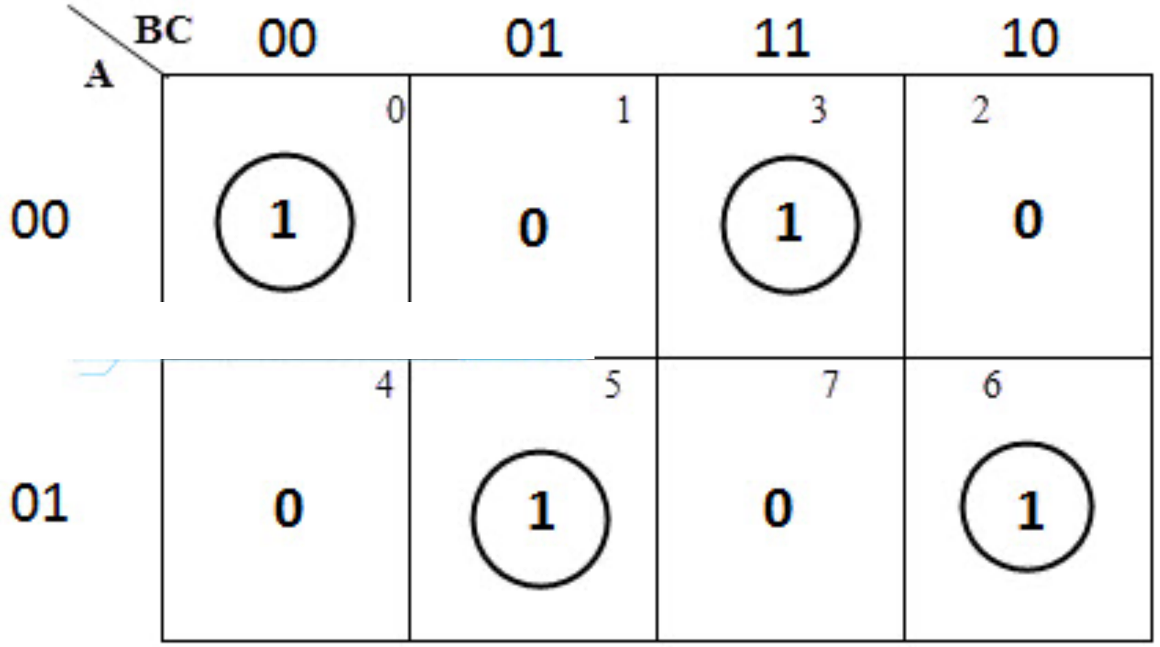


Odd Parity Generator

- Let us consider that the 3-bit data is to be transmitted with an odd parity bit.
- The three inputs are A, B and C and P is the output parity bit. The total number of bits must be odd in order to generate the odd parity bit.
- In the given truth table below, 1 is placed in the parity bit in order to make the total number of bits odd when the total number of 1s in the truth table is even.

3-bit message			Odd parity bit generator (P)
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

- The truth table of the odd parity generator can be simplified by using K-map as:



A \ BC	00	01	11	10
00	^{0} 1	^{1} 0	^{3} 1	^{2} 0
01	^{4} 0	^{5} 1	^{7} 0	^{6} 1

- The output parity bit expression for this generator circuit is obtained as:

$P = A \oplus B \odot C$

The above Boolean expression can be implemented by using one Ex-OR gate and one Ex-NOR gate in order to design a 3-bit odd parity generator.

- The logic circuit of this generator is shown in below figure , in which . two inputs are applied at one Ex-OR gate, and this Ex-OR output and third input is applied to the Ex-NOR gate , to produce the odd parity bit. It is also possible to design this circuit by using two Ex-OR gates and one NOT gate.

- see final circuit in video

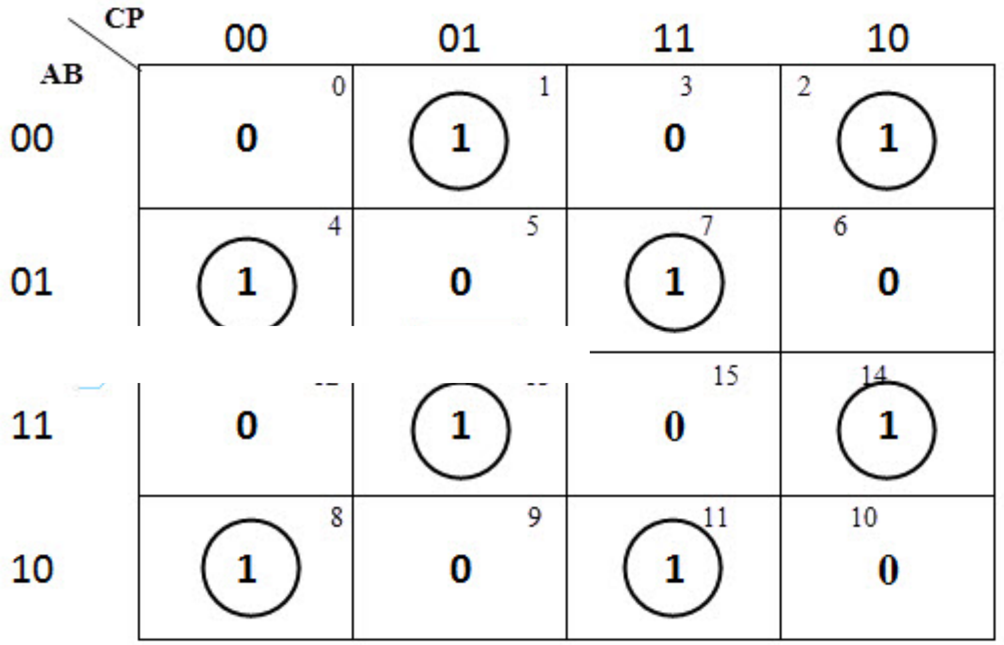
Parity check

- It is a logic circuit that checks for possible errors in the transmission. This circuit can be an even parity checker or odd parity checker depending on the type of parity generated at the transmission end. When this circuit is used as even parity checker, the number of input bits must always be even.
- When a parity error occurs, the ‘sum even’ output goes low and ‘sum odd’ output goes high. If this logic circuit is used as an odd parity checker, the number of input bits should be odd, but if an error occurs the ‘sum odd’ output goes low and ‘sum even’ output goes high.

Even Parity Checker

- Consider that three input message along with even parity bit is generated at the transmitting end. These 4 bits are applied as input to the parity checker circuit which checks the possibility of error on the data. Since the data is transmitted with even parity, four bits received at circuit must have an even number of 1s.
- If any error occurs, the received message consists of odd number of 1s. The output of the parity checker is denoted by PEC (parity error check).
- The below table shows the truth table for the even parity checker in which PEC = 1 if the error occurs, i.e., the four bits received have odd number of 1s and PEC = 0 if no error occurs, i.e., if the 4-bit message has even number of 1s.

4-bit received message				Parity error check C_p
A	B	C	P	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0



AB \ CP	00	01	11	10
00	0 (0)	1 (1)	0 (3)	1 (2)
01	1 (4)	0 (5)	1 (7)	0 (6)
11	0	1	0 (15)	1 (14)
10	1 (8)	0 (9)	1 (11)	0 (10)

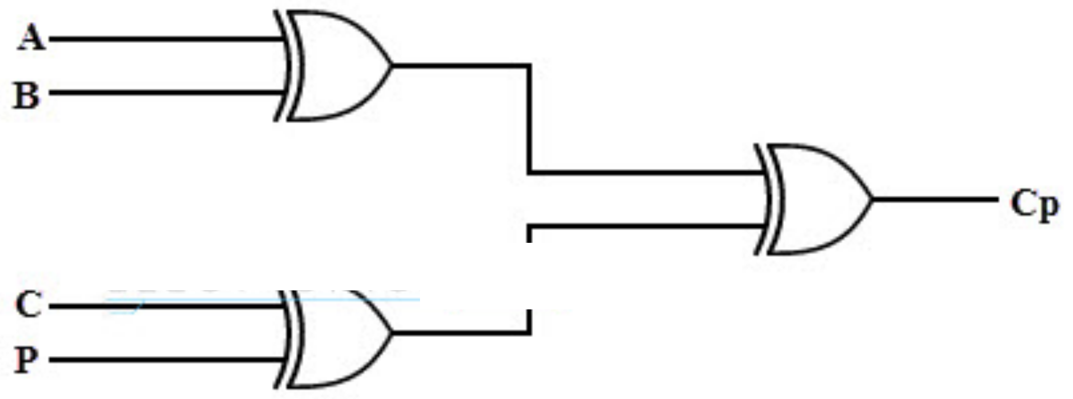
$$PEC = \bar{A}\,\bar{B}\,(\bar{C}\,D + C\,\bar{D}) + \bar{A}\,B\,(\bar{C}\,\bar{D} + C\,D) + A\,B\,(\bar{C}\,D + C\,\bar{D}) + A\,\bar{B}\,(\bar{C}\,\bar{D} + C\,D)$$

$$= \bar{A}\,\bar{B}\,(C \oplus D) + \bar{A}\,B\,(\overline{C \oplus D}) + A\,B\,(C \oplus D) + A\,\bar{B}\,(\overline{C \oplus D})$$

$$= (\bar{A}\,\bar{B} + A\,B)\,(C \oplus D) + (\bar{A}\,B + A\,\bar{B})\,(\overline{C \oplus D})$$

$$= (A \oplus B) \oplus (C \oplus D)$$

- The above logic expression for the even parity checker can be implemented by using three Ex-OR gates as shown in figure. If the received message consists of five bits, then one more Ex-OR gate is required for the even parity checking.

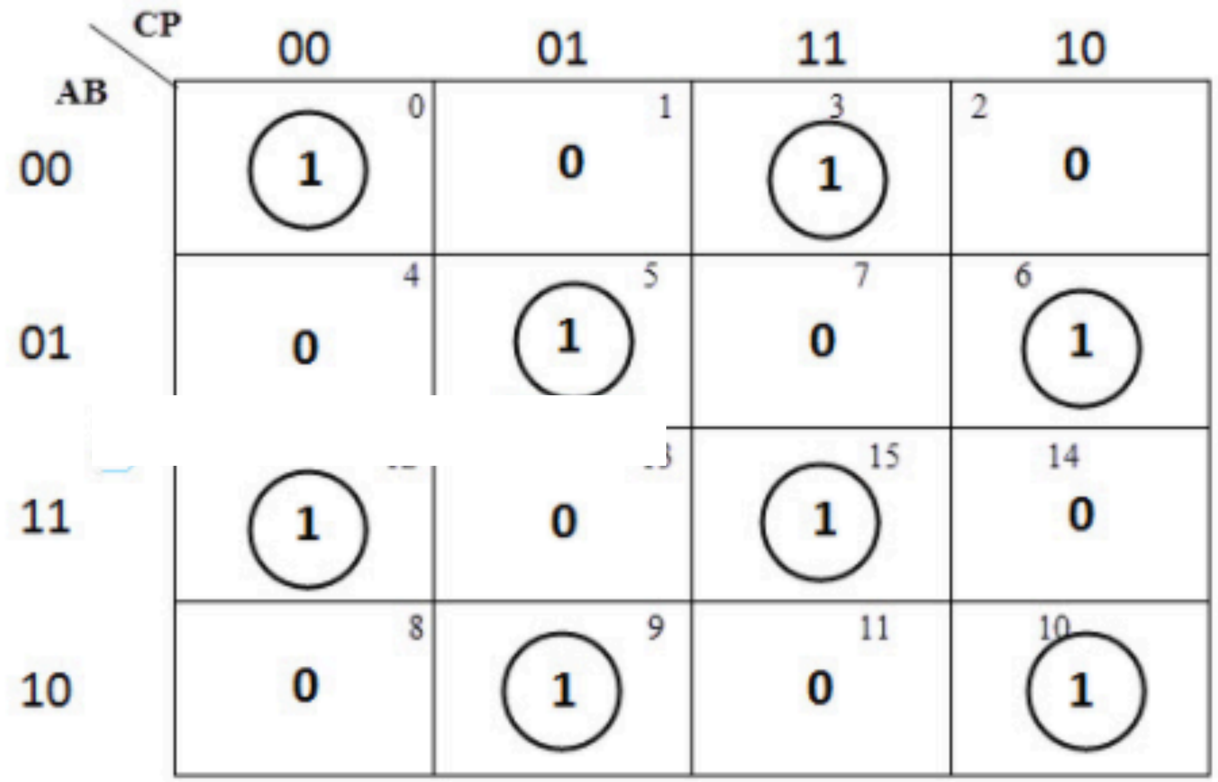


Odd Parity Checker

- Consider that a three bit message along with odd parity bit is transmitted at the transmitting end. Odd parity checker circuit receives these 4 bits and checks whether any error are present in the data.
- If the total number of 1s in the data is odd, then it indicates no error, whereas if the total number of 1s is even then it indicates the error since the data is transmitted with odd parity at transmitting end.
- The below figure shows the truth table for odd parity generator where **PEC =1** if the 4-bit message received consists of **even number of 1s** (hence the error occurred) and **PEC= 0** if the message contains **odd number of 1s** (that means no error).

4-bit received message				Parity error check C_p
A	B	C	P	
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

The expression for the PEC in the above truth table can be simplified by K-map as shown below.



AB \ CP	00	01	11	10
00	$\textcircled{1}$ ^{0}	0 ^{1}	$\textcircled{1}$ ^{3}	^{2} 0
01	0 ^{4}	$\textcircled{1}$ ^{5}	0 ^{7}	^{6} $\textcircled{1}$
11	$\textcircled{1}$	0	$\textcircled{1}$ ^{15}	^{14} 0
10	0 ^{8}	$\textcircled{1}$ ^{9}	0 ^{11}	^{10} $\textcircled{1}$

After simplification, the final expression for the PEC is obtained as

PEC = (A Ex-NOR B) Ex-NOR (C Ex-NOR D)

The expression for the odd parity checker can be designed by using three Ex-NOR gates as shown below.