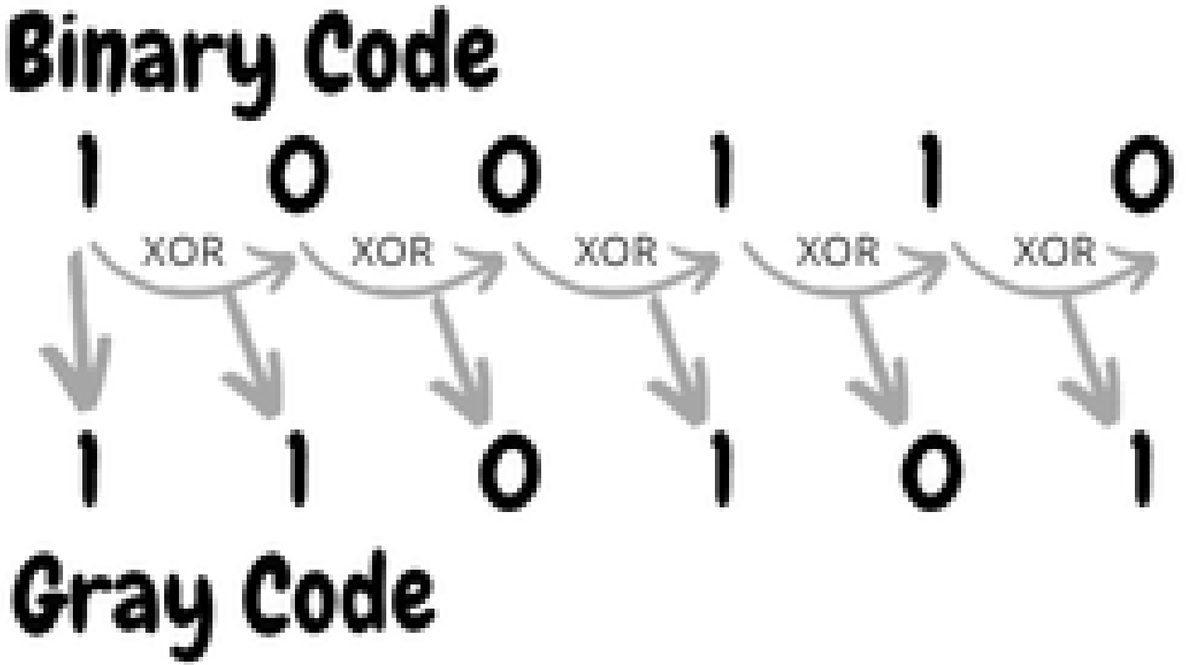
Gray code?

- The reflected binary code or Gray code is an ordering of the binary numeral system such that two successive values differ in only one bit (binary digit).
- Gray codes are very useful in the normal sequence of binary numbers generated by the hardware that may cause an error or ambiguity during the transition from one number to the next.
- So, the Gray code can eliminate this problem easily since only one bit changes its value during any transition between two numbers.
- Gray code is not weighted that means it does not depends on positional value of digit.
- This cyclic variable code that means every transition from one value to the next value involves only one bit change.
-

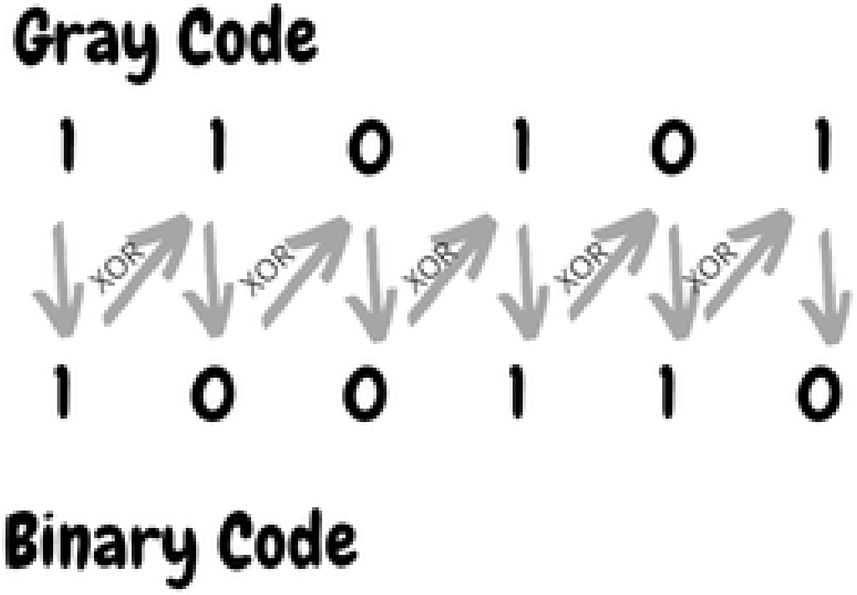
Binary to Gray conversion :

The Most Significant Bit (MSB) of the gray code is always equal to the MSB of the given binary code.
Other bits of the output gray code can be obtained by XORing binary code bit at that index and previous index.



Gray to binary conversion :

The Most Significant Bit (MSB) of the binary code is always equal to the MSB of the given gray code.
Other bits of the output binary code can be obtained by checking the gray code bit at that index. If the current gray code bit is 0, then copy the previous binary code bit, else copy the invert of the previous binary code bit.

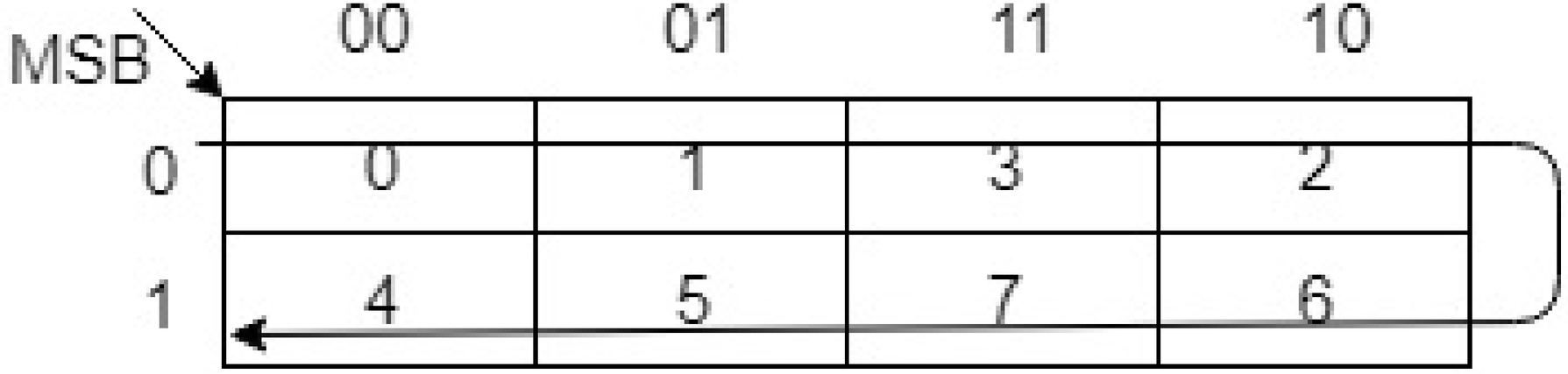


Conversion of Gray to Binary Code

Gray codes are used in rotary and optical encoders, Karnaugh maps, and error detection. The hamming distance of two neighbours Gray codes is always 1 and also first Gray code and last Gray code also has Hamming distance is always 1, so it is also called Cyclic codes. You can convert a Gray code to Binary number using two methods.

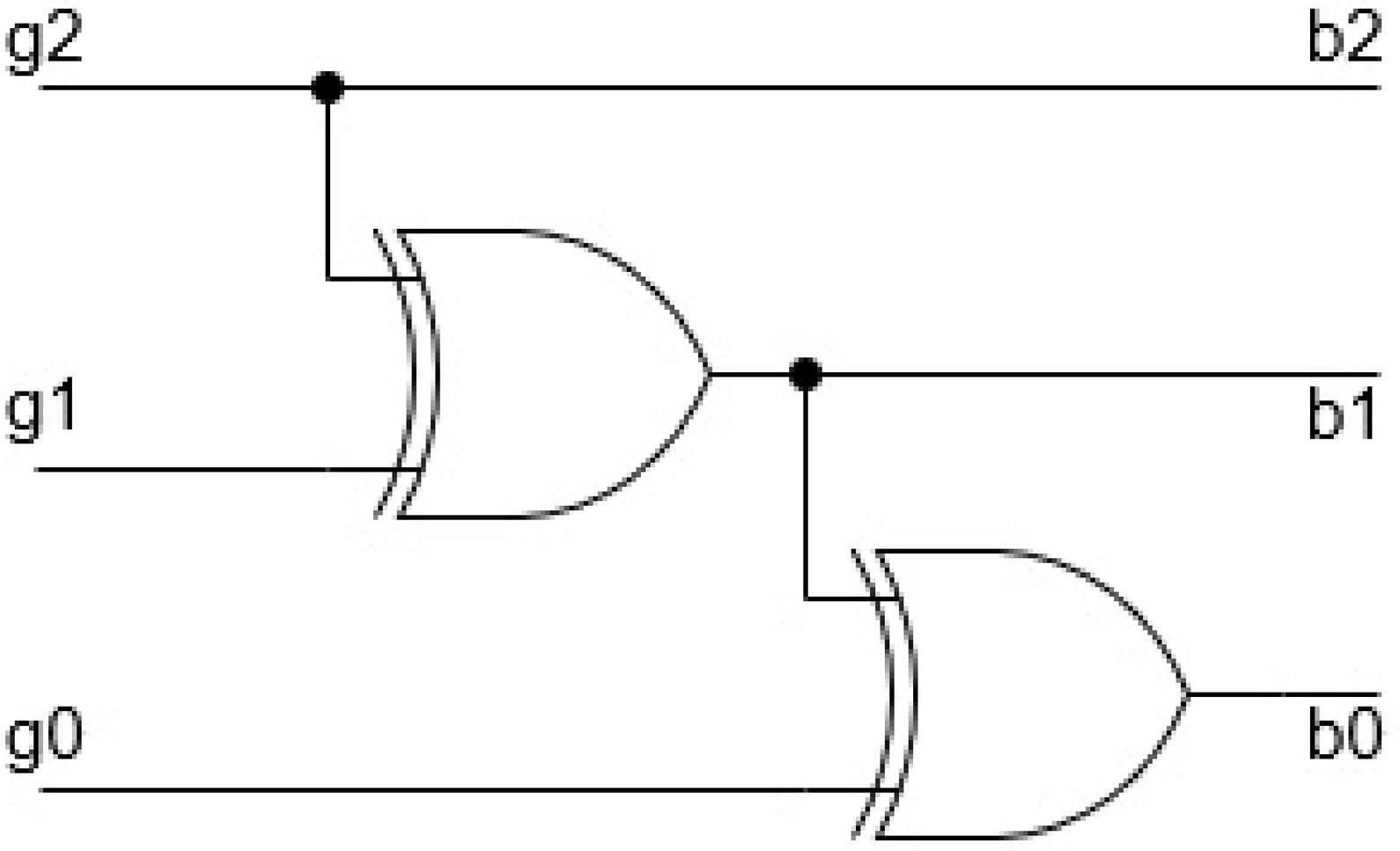
Using Karnaugh (K) - map –

You can construct Gray codes using other methods but they may not be performed in parallel like given above method. For example, 3 bit Gray codes can be contracted using K-map which is given as following below:

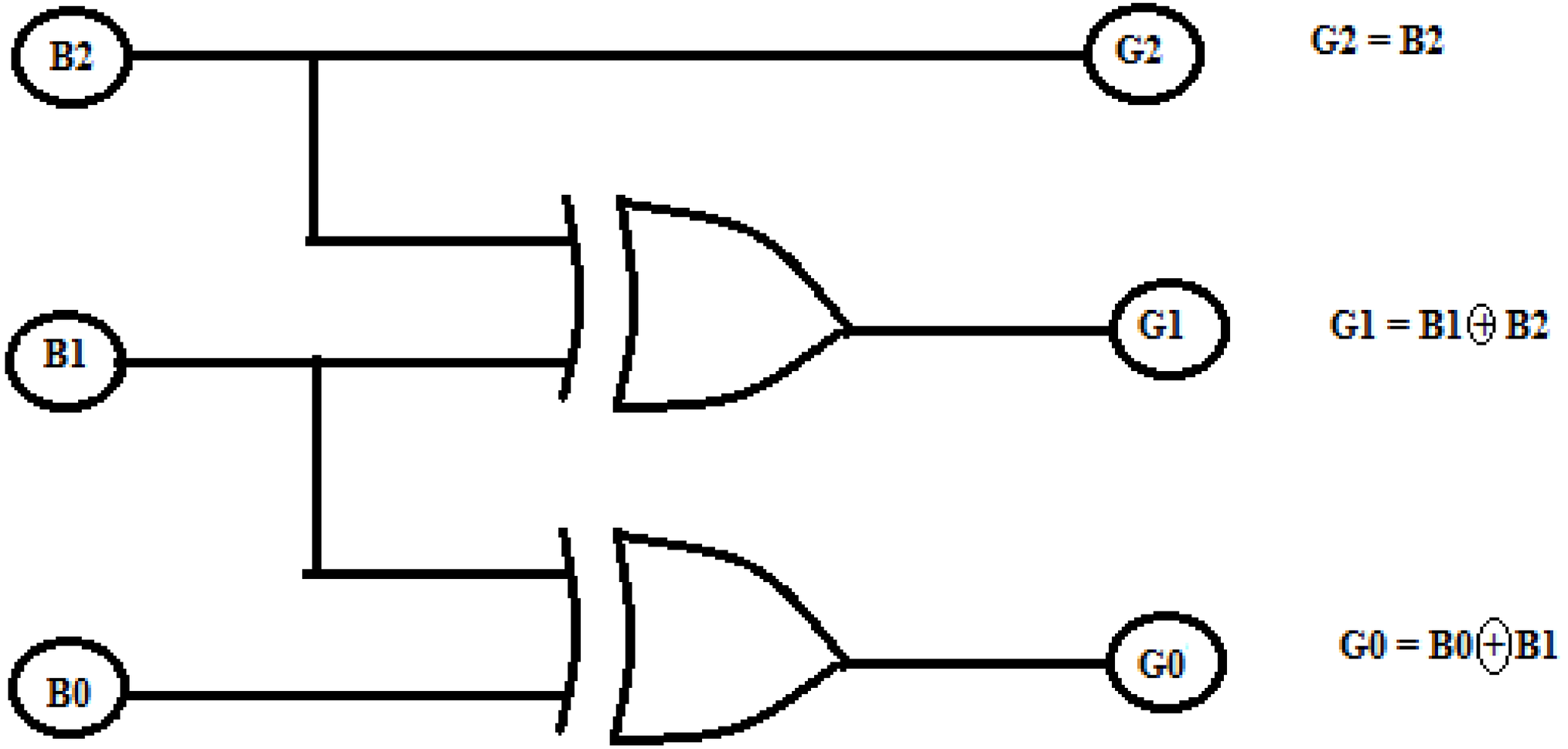


Decimal	binary	grey
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111

solve boolean expression using k-map, you will get b2=g2, b1=g1⊕g2, and b0=g0⊕g1⊕g2.


g2
b2
g1
b1
g0
b0

or



Using Exclusive-Or (⊕) operation –

This is very simple method to get Binary number from Gray code. These are following steps for n-bit binary numbers –

The Most Significant Bit (MSB) of the binary code is always equal to the MSB of the given binary number.
Other bits of the output binary code can be obtained by checking gray code bit at that index. If current gray code bit is 0, then copy previous binary code bit, else copy invert of previous binary code bit.

BCD or Binary Coded Decimal

- Binary Coded Decimal, or BCD, is another process for converting decimal numbers into their binary equivalents.
- It is a form of binary encoding where each digit in a decimal number is represented in the form of bits.
- This encoding can be done in either 4-bit or 8-bit (usually 4-bit is preferred).
- It is a fast and efficient system that converts the decimal numbers into binary numbers as compared to the existing binary system.
- These are generally used in digital displays where is the manipulation of data is quite a task.
- Thus BCD plays an important role here because the manipulation is done treating each digit as a separate single sub-circuit.

The BCD equivalent of a decimal number is written by replacing each decimal digit in the integer and fractional parts with its four bit binary equivalent.
the BCD code is more precisely known as 8421 BCD code , with 8,4,2 and 1 representing the weights of different bits in the four-bit groups, Starting from MSB and proceeding towards LSB.
This feature makes it a weighted code , which means that each bit in the four bit group representing a given decimal digit has an assigned weight.

REASON TO USE BCD :

Many decimal values, have an infinite place-value representation in binary but have a finite place-value in binary-coded decimal. For example, 0.2 in binary is .001100... and in BCD is 0.0010. It avoids fractional errors and is also used in huge financial calculations.

DECIMAL NUMBER	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

For example:

1. Convert (123)10 in BCD

From the truth table above,
1 -> 0001
2 -> 0010
3 -> 0011
thus, BCD becomes -> 0001 0010 0011

Convert (324)10 in BCD

(324)10 -> 0011 0010 0100 (BCD)

Again from the truth table above,
3 -> 0011
2 -> 0010
4 -> 0100
thus, BCD becomes -> 0011 0010 0100

Steps of BCD Addition

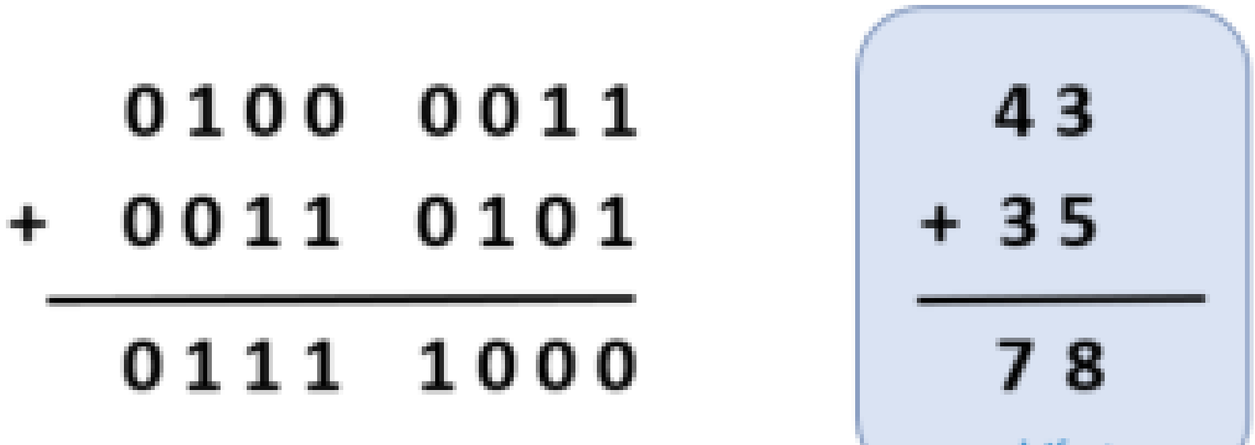
Step 1: Add the two BCD numbers using the rules for binary addition.
Step 2: If a 4-bit sum is equal to or less than 9, it is a valid BCD number.
Step 3: If a 4-bit sum is greater than 9 or if a carry-out of the 4-bit group is generated, it is an invalid result. Add 6 (0110) to the 4-bit sum in order to skip the six invalid BCD code words and return the code to 8421. If a carry results when 6 is added, simply add the carry to the next 4-bit group.

Example 1: Find the sum of the BCD numbers 01000011 + 00110101

Solution:

Decimal number of the given BCD numbers are as below:

(01000011)BCD = 43)BASE 10 and (00110101)BCD = 35)BASE10



- **In the above example, all the 4-bit BCD additions generate valid BCD numbers, which means less than 9. So, the final correct result is 7810 = 01111000BCD.**

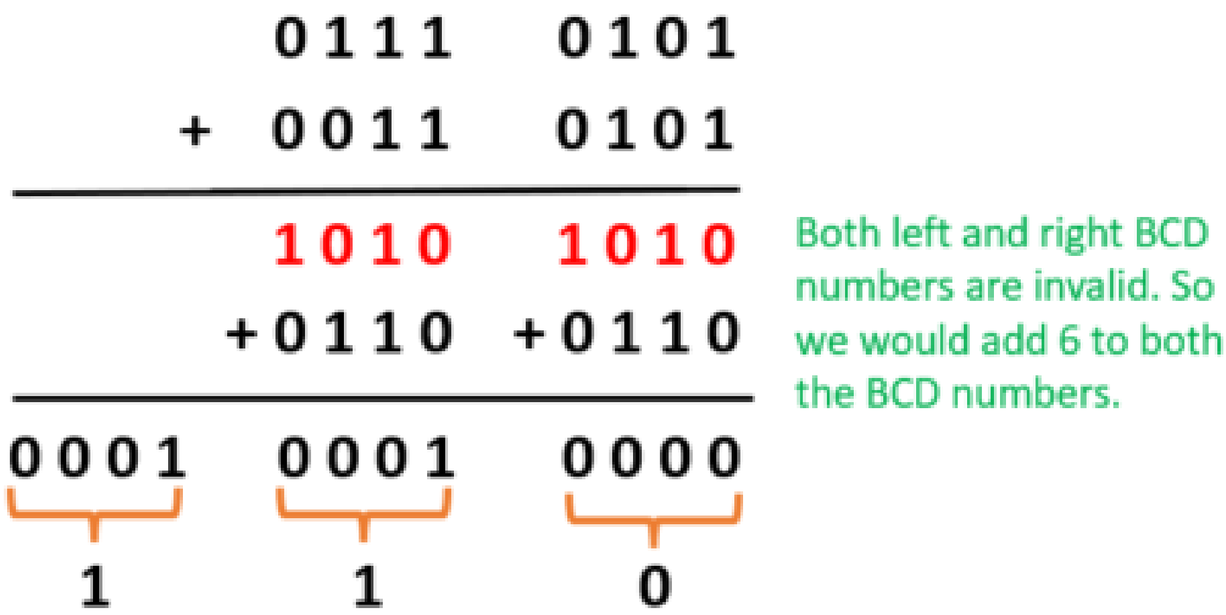
Let’s take an example where the addition generates an invalid BCD number.

Example 2: Find the sum of the BCD numbers 01110101 + 00110101

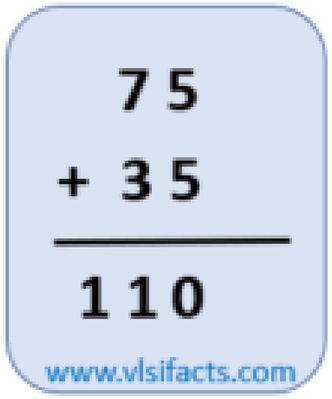
Solution:

The decimal number of the given BCD numbers are as below:

01110101BCD = 75 and 00110101BCD = 35



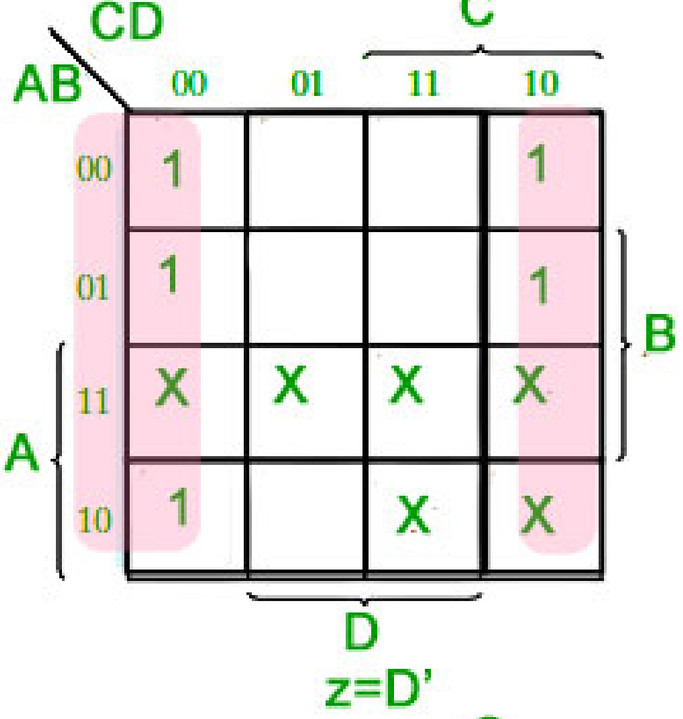
Both left and right BCD numbers are invalid. So we would add 6 to both the BCD numbers.



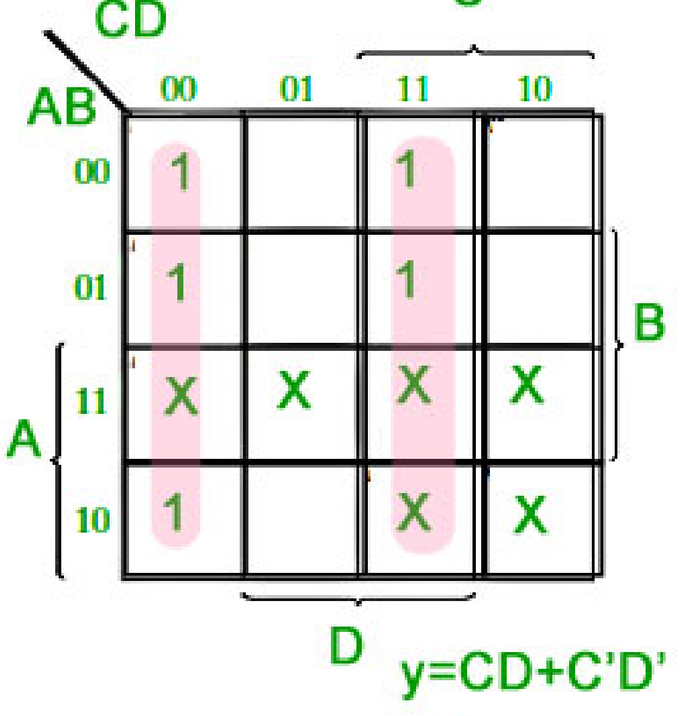
BCD to Excess-3 conversion

The Excess-3 code can be calculated by adding 3, i.e., 0011 to each four-digit BCD code.

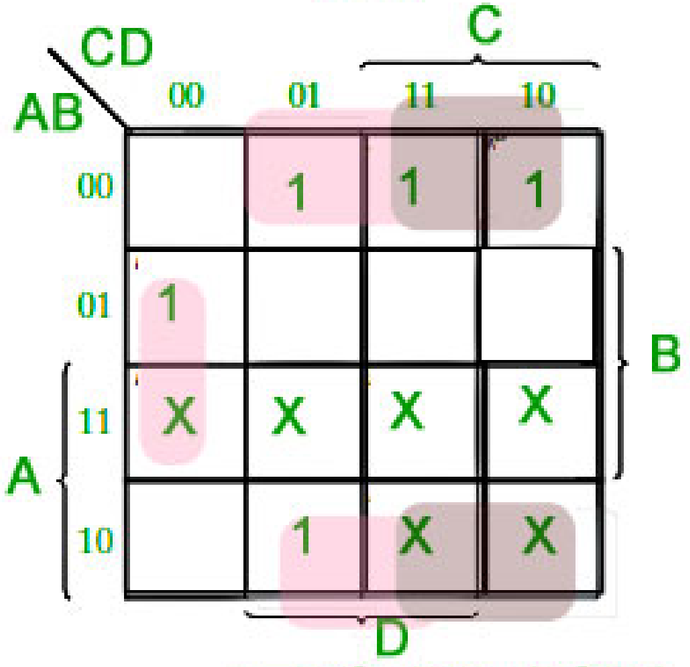
Decimal Number	BCD Code				Excess-3 Code			
	A	B	C	D	W	x	y	z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
10	1	0	1	0	X	X	X	X
11	1	0	1	1	X	X	X	X
12	1	1	0	0	X	X	X	X
13	1	1	0	1	X	X	X	X
14	1	1	1	0	X	X	X	X
15	1	1	1	1	X	X	X	X



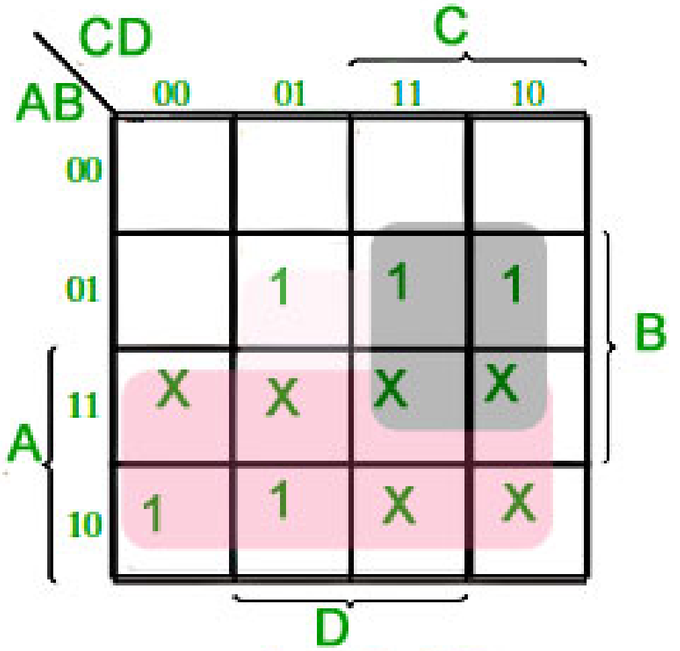
z=D'



y=CD+C'D'



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



w=A+BC+BD

So,

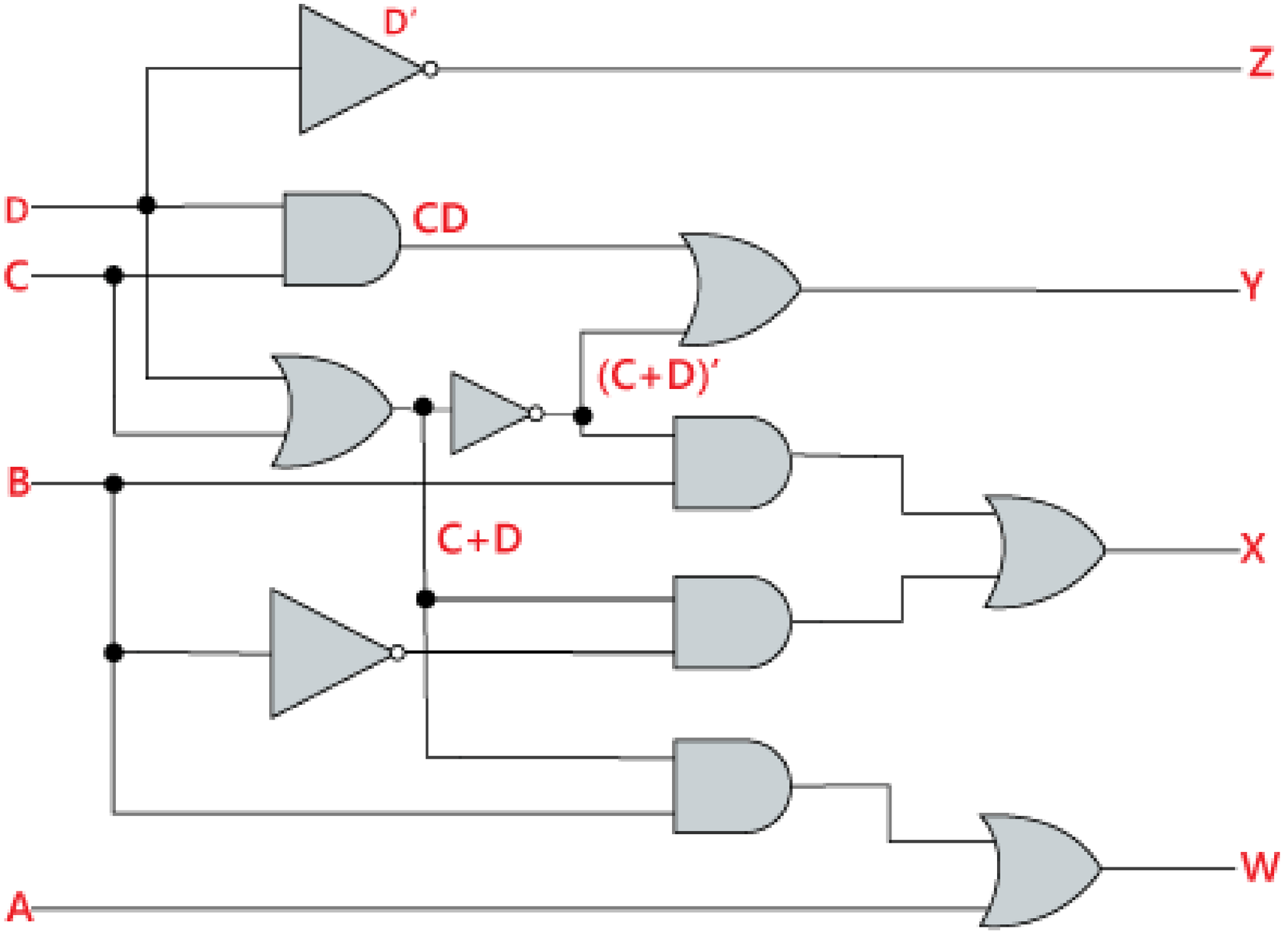
$$w=A+BC+BD$$

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

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

$$z=D'$$

https://electricalworkbook.com/bcd-to-excess-3-code-converter-circuit/


D'
Z
D
CD
C
Y
(C+D)'
B
C+D
X
W
A

Example: (100001011001)BCD

To find the Excess-3 code of the given Excess-3 code, first, we will make the group of 4 bits from right to left. Then, we will add 0011 in each group of 4 bits in order to get the excess-3 code.

00001011001

Making Group pf 4 bit

1000 0101 1001

Add 001 in each 4-bit Group

1000	0101	1001
+0011	+0011	+0011
1011	1000	1100

So. $(100001011001)_{BCD} = (101110001100)_{Excess\text{-}3}$

Excess-3 to BCD conversion

The BCD code can be calculated by subtracting 3, i.e., 0011 from each four-digit Excess-3 code

Decimal Number	Excess-3 Code	BCD Code
	A B C D	W x y z
0	0 0 0 0	X X X X
1	0 0 0 1	X X X X
2	0 0 1 0	X X X X
3	0 0 1 1	0 0 0 0
4	0 1 0 0	0 0 0 1
5	0 1 0 1	0 0 1 0
6	0 1 1 0	0 0 1 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	0 1 0 1
9	1 0 0 1	0 1 1 0
10	1 0 1 0	0 1 1 1
11	1 0 1 1	1 0 0 0
12	1 1 0 0	1 0 0 1
13	1 1 0 1	X X X X
14	1 1 1 0	X X X X
15	1 1 1 1	X X X X