UNIT-II Combinational Logic

Code Converter

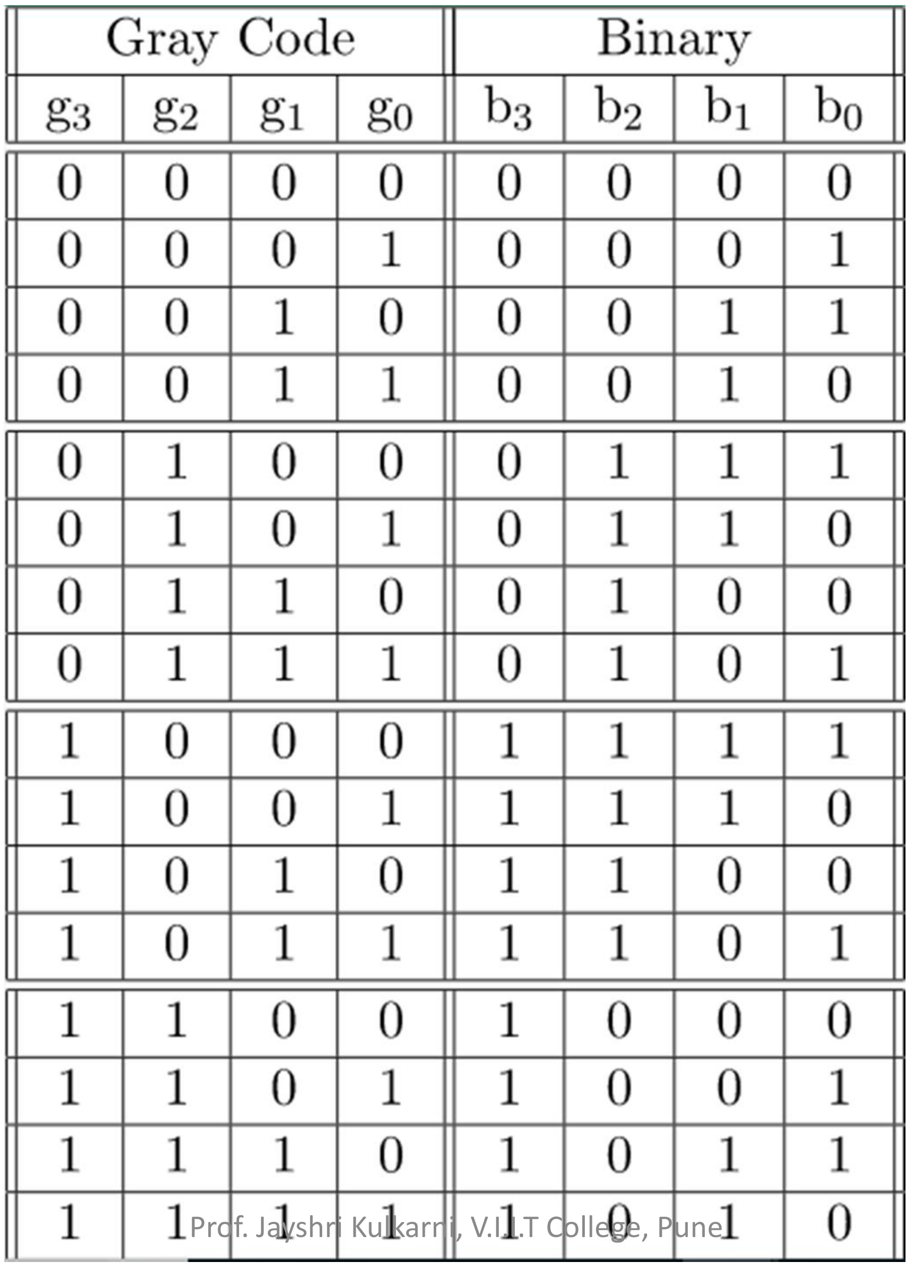
- **Gray Code**

- 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.
- For example, the states of a system may change from 3(011) to 4(100) as- 011 — 001 — 101 — 100.
- Therefore there is a high chance of a wrong state being read while the system changes from the initial state to the final state.
- This could have serious consequences for the machine using the information.
- The Gray code eliminates this problem since only one bit changes its value during any transition between two numbers.

Gray to Binary Code Converter

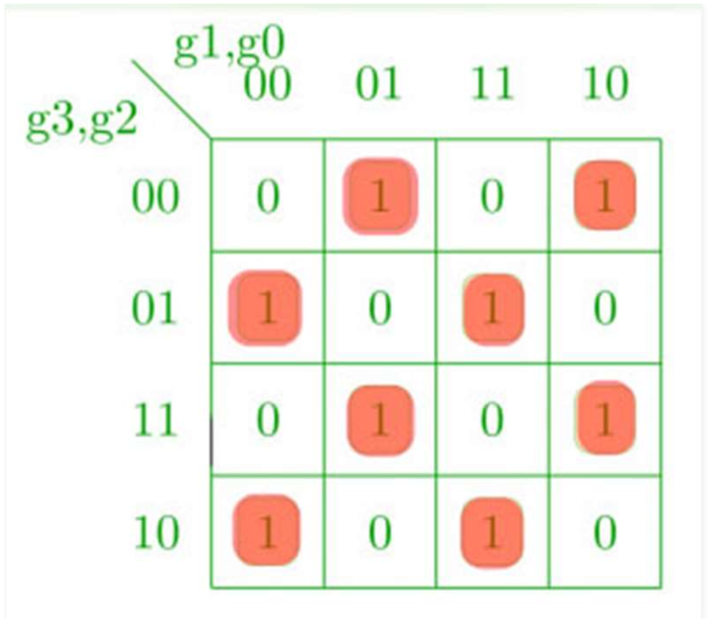
- Let b_0, b_1, b_2, and b_3 be the bits representing the binary numbers, where b_0 is the LSB and b_3 is the MSB, and
- Let g_0, g_1, g_2, and g_3 be the bits representing the gray code of the binary numbers, where g_0 is the LSB and g_3 is the MSB.
- The truth table is as given below in table 2.2
- K-Map is used to get the Binary bit from the grey code

Table 2.2 Truth table for Gray to Binary Code Converter



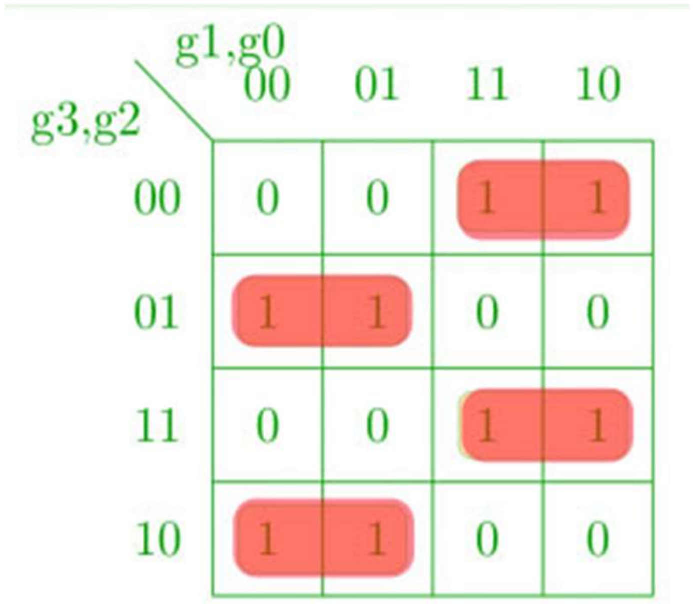
Gray Code				Binary			
g_3	g_2	g_1	g_0	b_3	b_2	b_1	b_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1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

K-Map for b_0



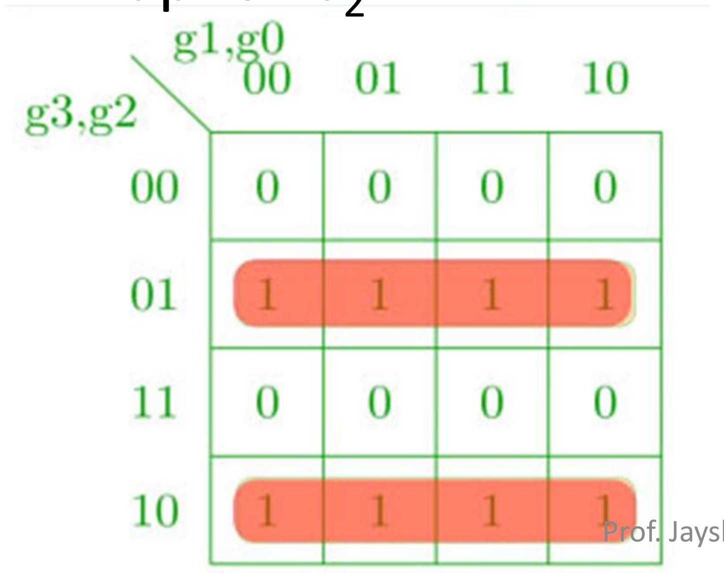
g3,g2 \ g1,g0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

K-Map for b_1



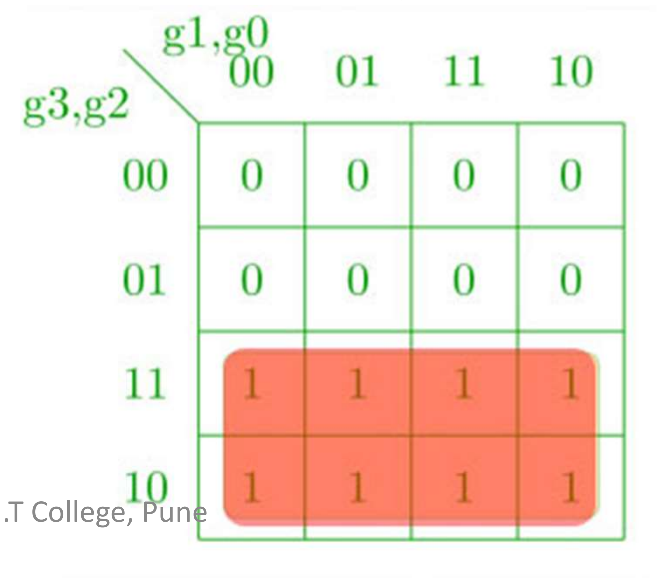
g3,g2 \ g1,g0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

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

- From the above k-maps, we get the following Boolean expressions:

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

$$\begin{aligned}
b_1 &= g_3'g_2'g_1 + g_3'g_2g_1' + g_3g_2g_1 + g_3g_2'g_1' \\
&= g_3'(g_2'g_1 + g_2g_1') + g_3(g_2g_1 + g_2'g_1') \\
&= g_3'(g_2 \oplus g_1) + g_3(g_2 \odot g_1) \\
&= g_3 \oplus g_2 \oplus g_1
\end{aligned}$$

$$\begin{aligned} b_2 &= g_3' g_2 + g_3 g_2' \\ &= g_3 \oplus g_2 \end{aligned}$$

$$b_3 = g_3$$

- Figure 2.21 shows the logical implementation of Gray code to binary code conversion

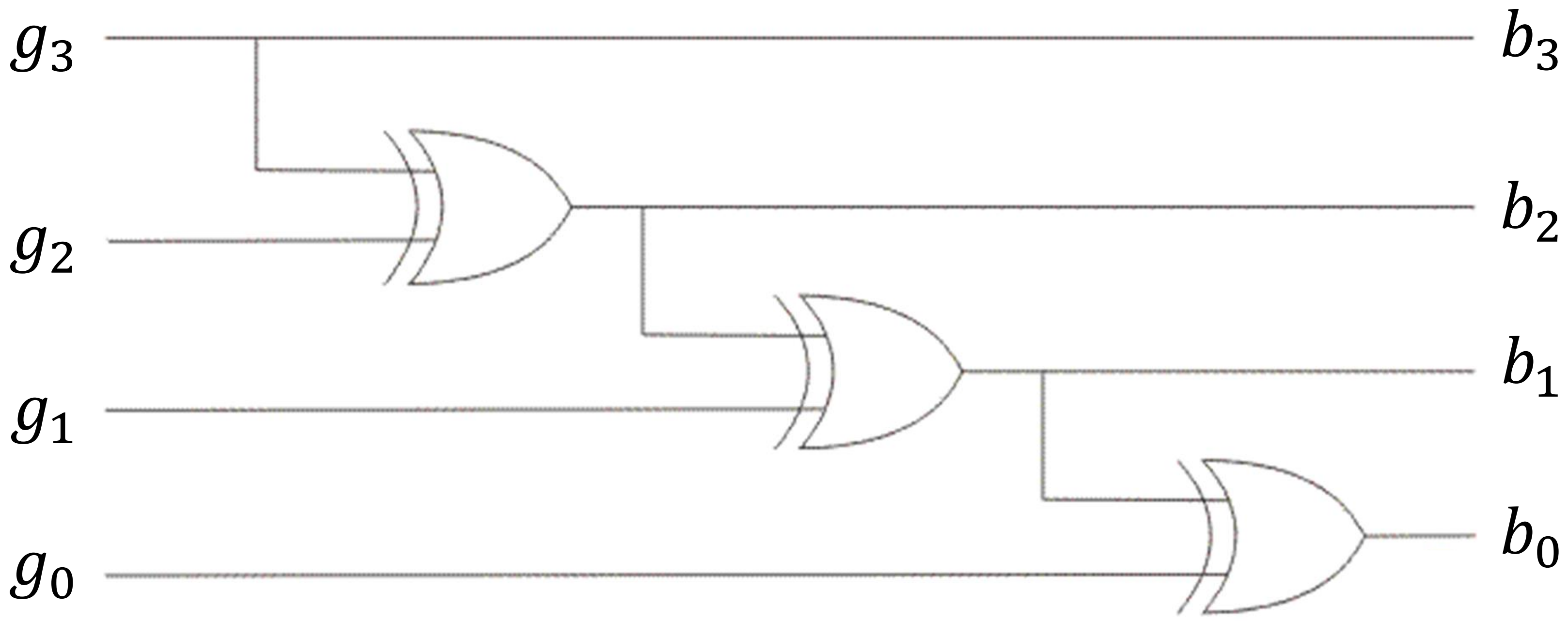


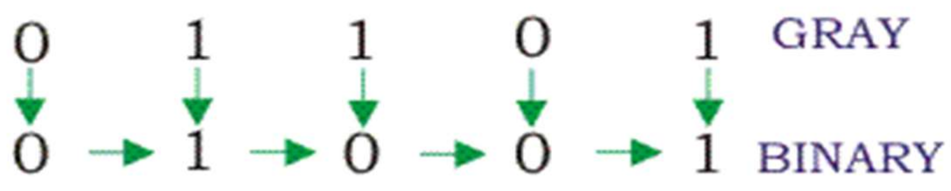
Figure 2.21 logical implementation of Gray code to binary code conversion

How to Convert Gray code to Binary

Gray code to binary conversion is a very simple and easy process. Following steps needs to be followed for this type of conversions.

- The MSB of the binary number will be equal to the MSB of the given gray code.
- Now if the second gray bit is 0, then the second binary bit will be the same as the previous or the first bit. If the gray bit is 1 the second binary bit will alter. If it was 1 it will be 0 and if it was 0 it will be 1.
- This step is continued for all the bits to do **Gray code to binary conversion**.

Eg: Convert Gray (01101) to Binary



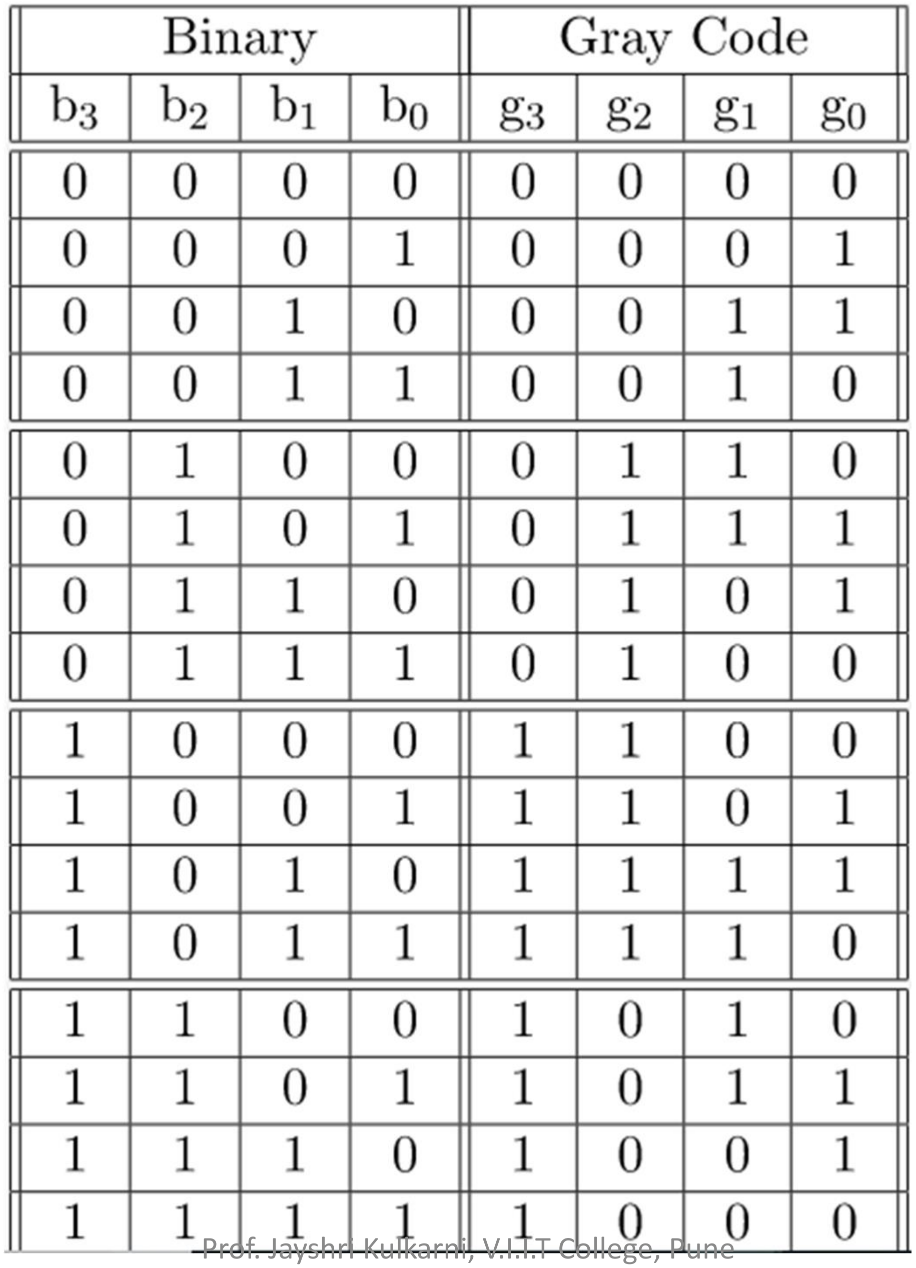
$$\begin{tikzcd} 0 \arrow[d] & 1 \arrow[d] & 1 \arrow[d] & 0 \arrow[d] & 1 \arrow[d] \\ 0 \arrow[r] & 1 \arrow[r] & 0 \arrow[r] & 0 \arrow[r] & 1 \end{tikzcd}$$

- The MSB of the binary will be 0 as the MSB of gray is 0.
- Now move to the next gray bit. As it is 1 the previous binary bit will alter i.e it will be 1, thus the second binary bit will be 1.
- Next look at the third bit of the gray code. It is again 1 thus the previous bit i.e the second binary bit will again alter and the third bit of the binary number will be 0.
- Now, the 4th bit of the given gray is 0 so the previous binary bit will be unchanged, i.e 4th binary bit will be 0.
- Now again the 5th grey bit is 1 thus the previous binary bit will alter, it will be 1 from 0.
- Therefore the equivalent binary number in case of gray code to the binary conversion will be (01001).

Binary to Gray Code Converter

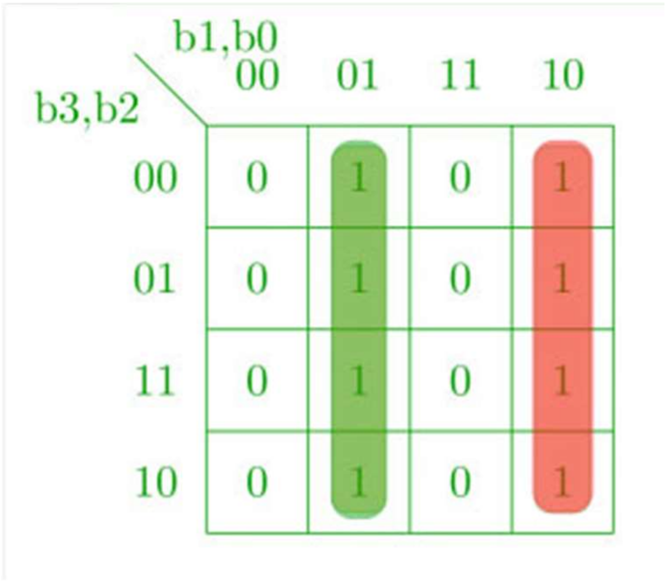
- Let b_0, b_1, b_2, and b_3 be the bits representing the binary numbers, where b_0 is the LSB and b_3 is the MSB, and
- Let g_0, g_1, g_2, and g_3 be the bits representing the gray code of the binary numbers, where g_0 is the LSB and g_3 is the MSB.
- The truth table is as given below in table 2.3
- K-Map is used to get the Binary bit from the grey code

Table 2.3 Truth table for Binary to gray Code Converter



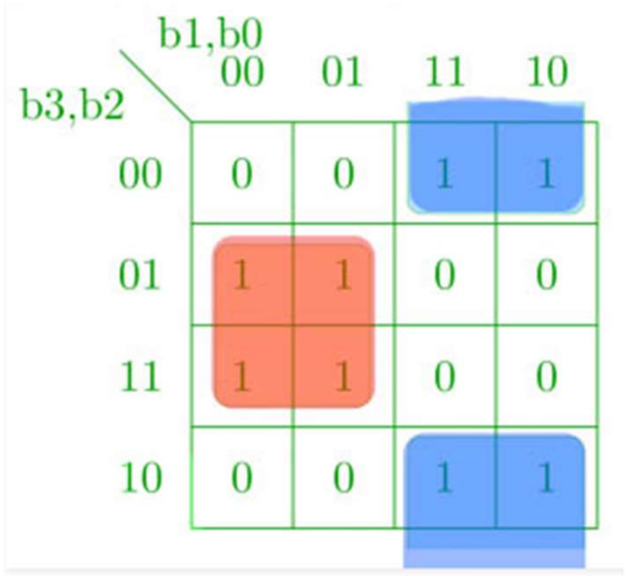
Binary				Gray Code			
b_3	b_2	b_1	b_0	g_3	g_2	g_1	g_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1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

K-Map for g_0



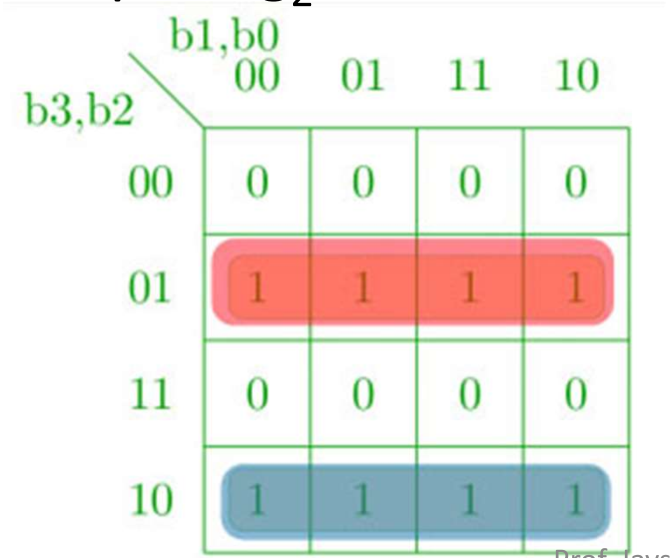
b3,b2 \ b1,b0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

K-Map for g_1



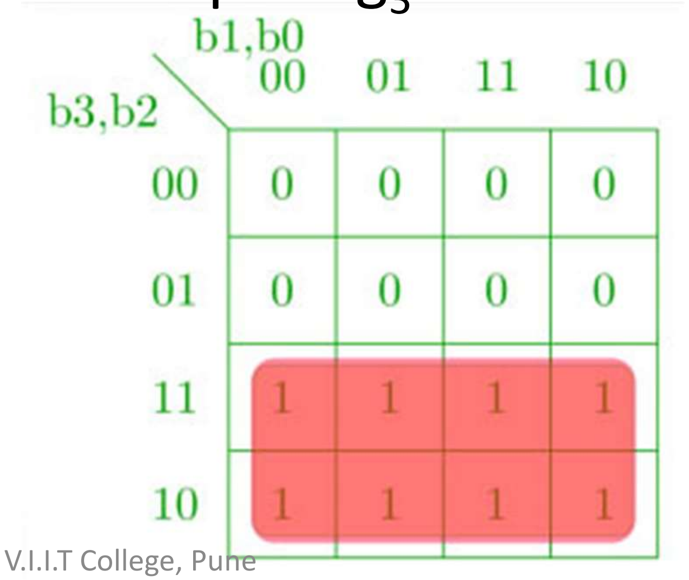
b3,b2 \ b1,b0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

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

- From the above k-maps, we get the following Boolean expressions:

$$g_0 = b_0 b_1' + b_1 b_0' = b_0 \oplus b_1$$
$$g_1 = b_2 b_1' + b_1 b_2' = b_1 \oplus b_2$$
$$g_2 = b_2 b_3' + b_3 b_2' = b_2 \oplus b_3$$
$$g_3 = b_3$$

- Figure 2.22 shows the logical implementation of binary to gray code conversion

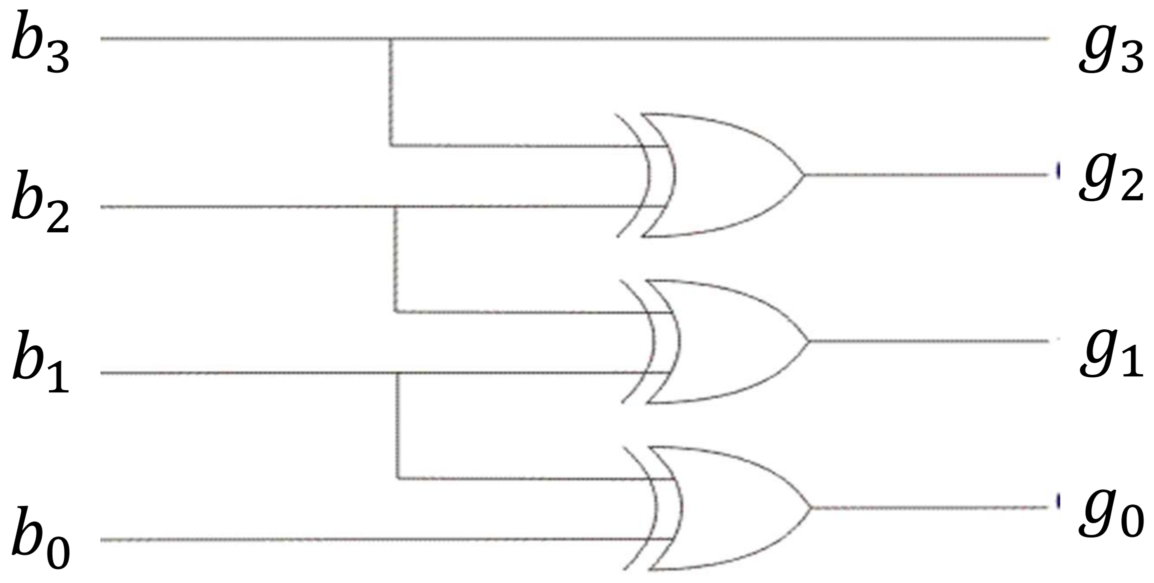
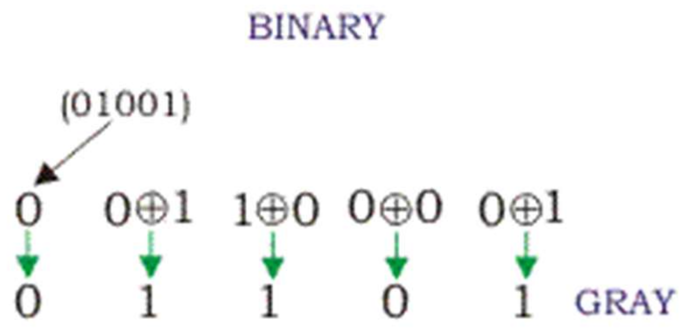


Figure 2.22 logical implementation of Binary to Gray code conversion

How to Convert Binary to Gray Code

- The MSB (Most Significant Bit) of the gray code will be exactly equal to the first bit of the given binary number.
- The second bit of the code will be exclusive-or (XOR) of the first and second bit of the given binary number, i.e if both the bits are same the result will be 0 and if they are different the result will be 1.
- The third bit of gray code will be equal to the exclusive-or (XOR) of the second and third bit of the given binary number. Thus the binary to gray code conversion goes on.

- Eg: Convert Binary number (01001) to gray code

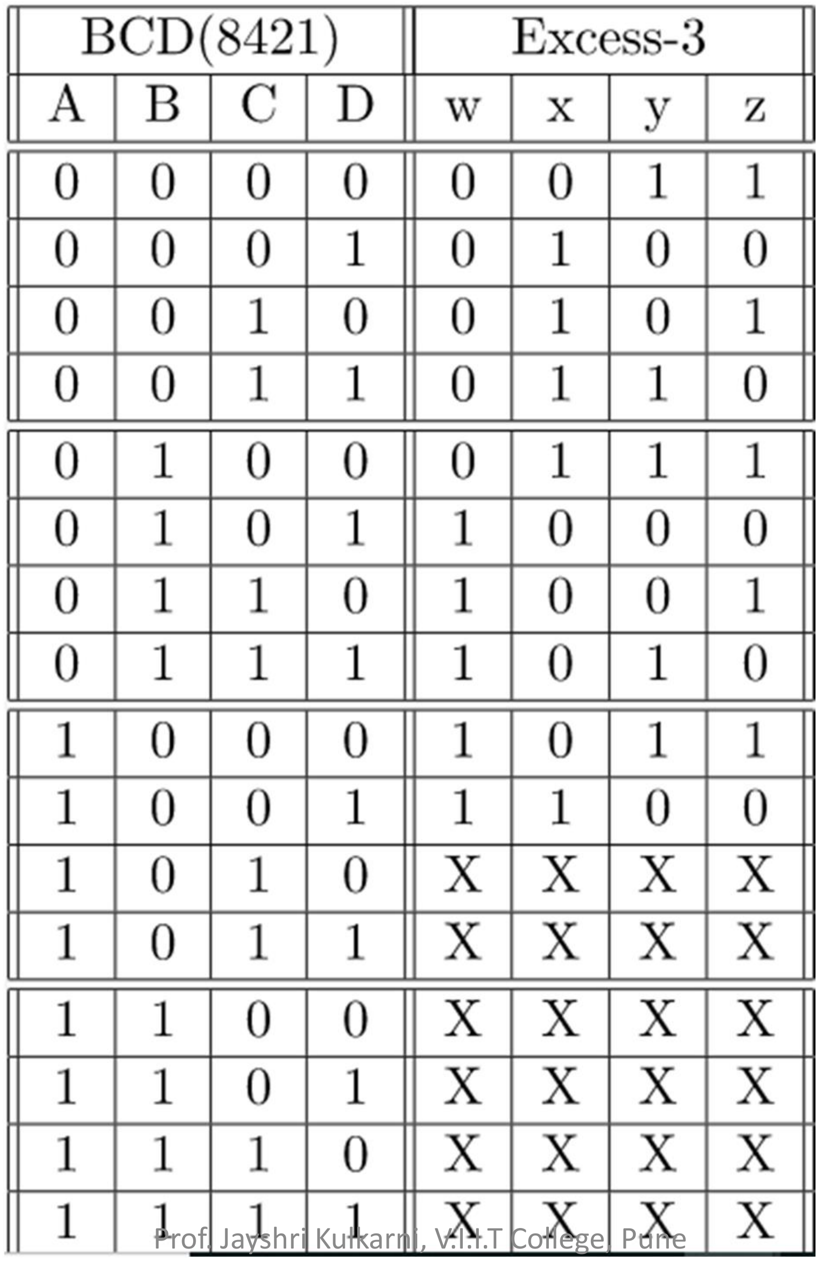


- The MSB is kept the same. As the MSB of the binary is 0, the MSB of the gray code will be 0 as well (first gray bit)
- Next, take the XOR of the first and the second binary bit. The first bit is 0, and the second bit is 1. The bits are different so the resultant gray bit will be 1 (second gray bit)
- Next, take the XOR of the second and third binary bit. The second bit is 1, and the third bit is 0. These bits are again different so the resultant gray bit will be 1 (third gray bit)
- Next, take the XOR of third and fourth binary bit. The third bit is 0, and the fourth bit is 0. As these are the same, the resultant gray bit will be 0 (fourth gray bit)
- Lastly, take the XOR of the fourth and fifth binary bit. The fourth bit is 0, and the fifth bit is 1. These bits are different so the resultant gray bit will be 1 (fifth gray bit)
- Hence the result of binary to gray code conversion of 01001 is complete, and the equivalent gray code is 01101.

BCD(8421) to Excess-3 Converter

- 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 BCD numbers, where D is the LSB and A is the MSB, and
- Let w, x, y, and z be the bits representing Excess-3 numbers, where z is the LSB and w is the MSB.
- The truth table for the conversion is given below in table 2.4. The X’s mark don’t care conditions.

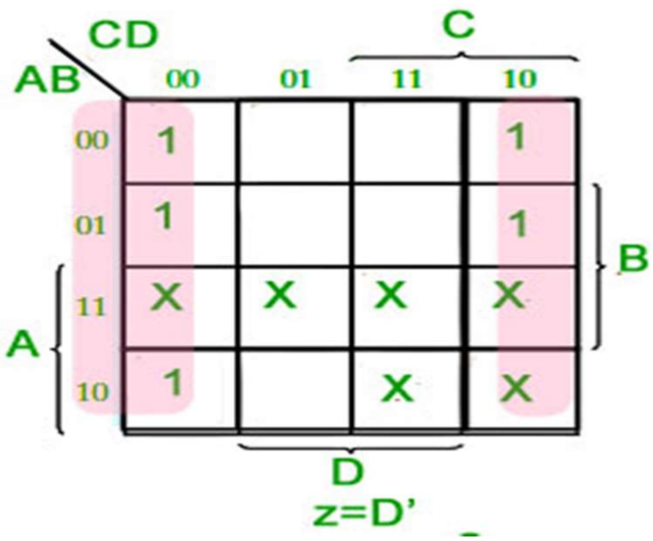
Table 2.4 BCD to Excess-3 Conversion



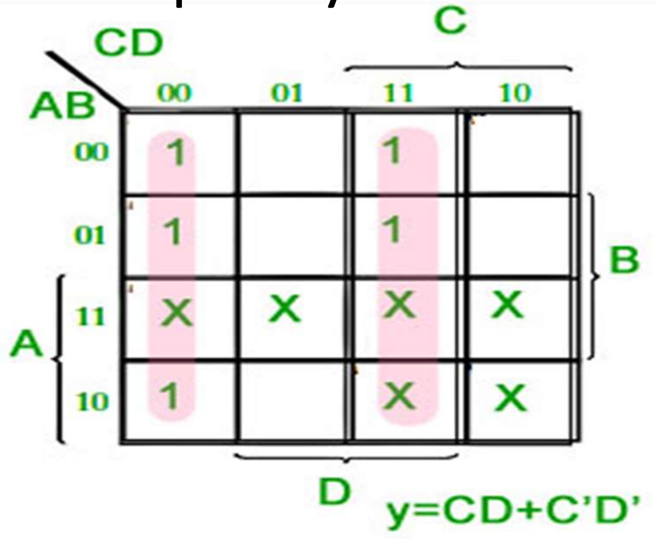
BCD(8421)				Excess-3			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

- The K-Map technique is used for each of the Excess-3 code bits as output with all of the bits of the BCD number as input

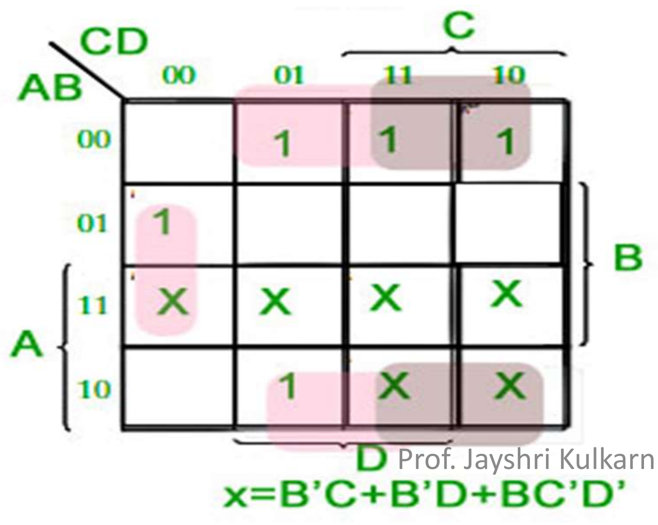
K-Map for z:



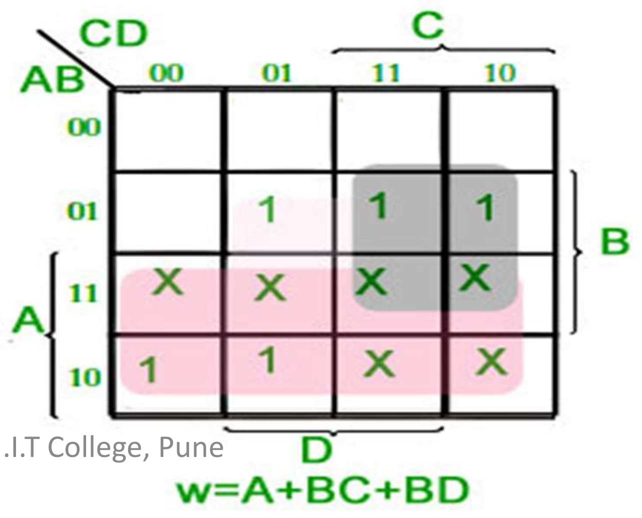
K-Map for y:



K-Map for x:



K-Map for w:



- Corresponding minimized Boolean expressions for Excess-3 code bits.

$$w = A + BC + BD$$
$$x = B'C + B'D + BC'D'$$
$$y = CD + C'D'$$
$$z = D'$$

- Figure 2.23 shows the logical implementation of BCD to Excess-3 code conversion

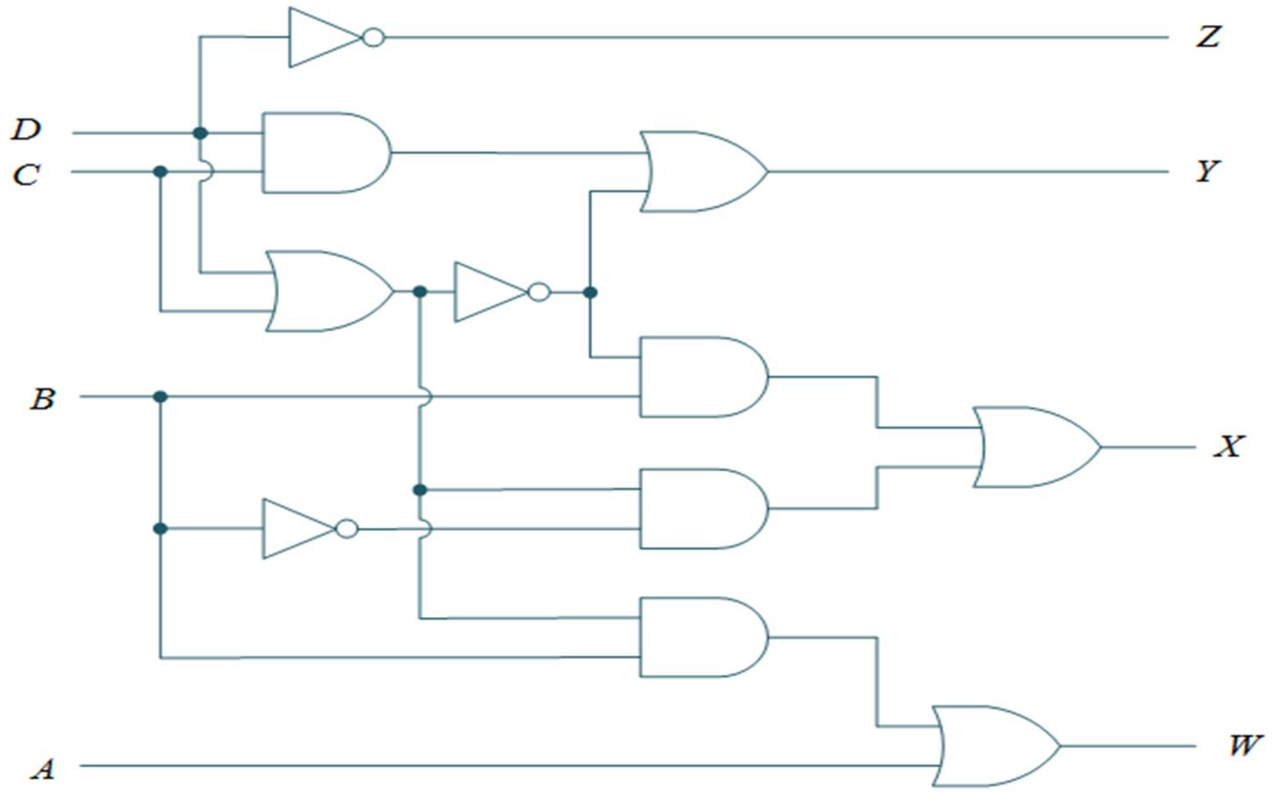
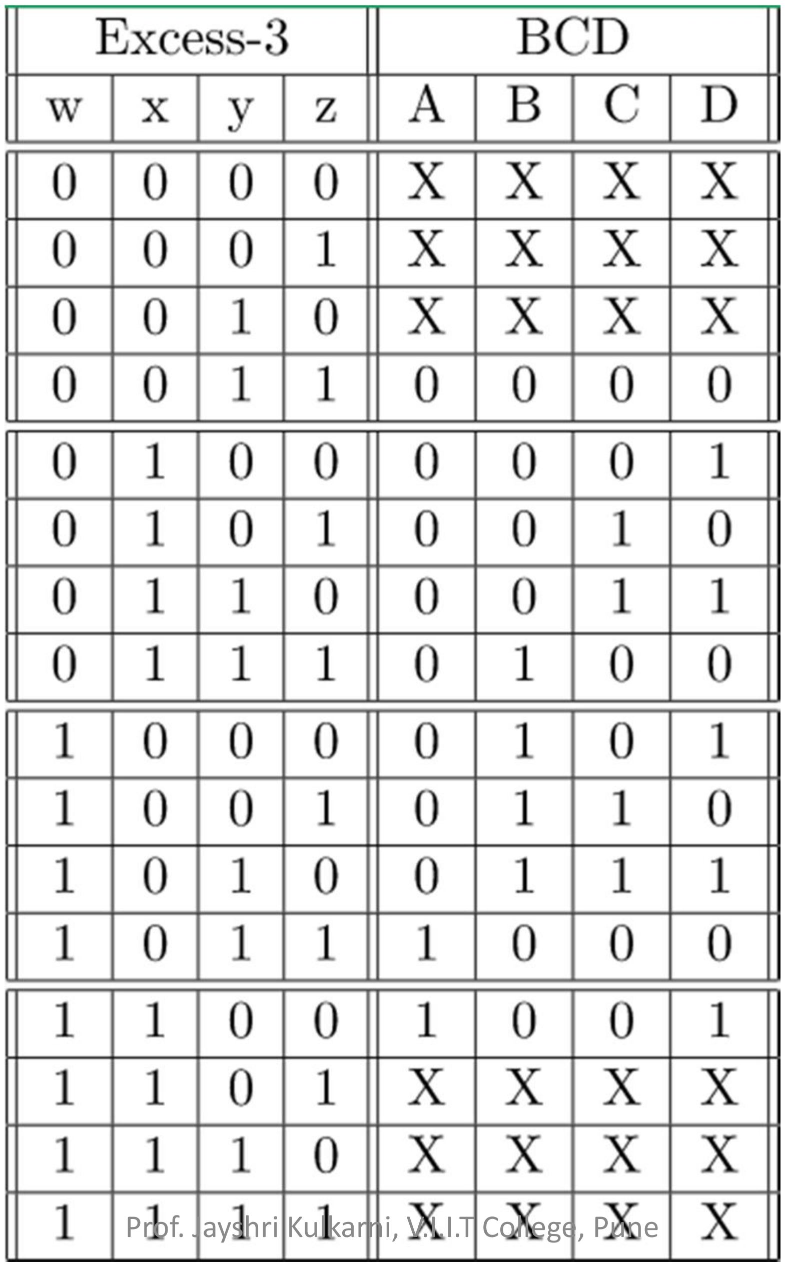


Figure 2.23 logical implementation of BCD to Excess-3 code conversion

Excess-3 to BCD Converter

- Excess-3 to BCD can be converted in same manner.
- Let A, B, C, and D be the bits representing the BCD numbers, where D is the LSB and A is the MSB, and
- Let w, x, y, and z be the bits representing Excess-3 numbers, where z is the LSB and w is the MSB.
- The truth table for the conversion is given below in table 2.5. The X's mark don't care conditions.

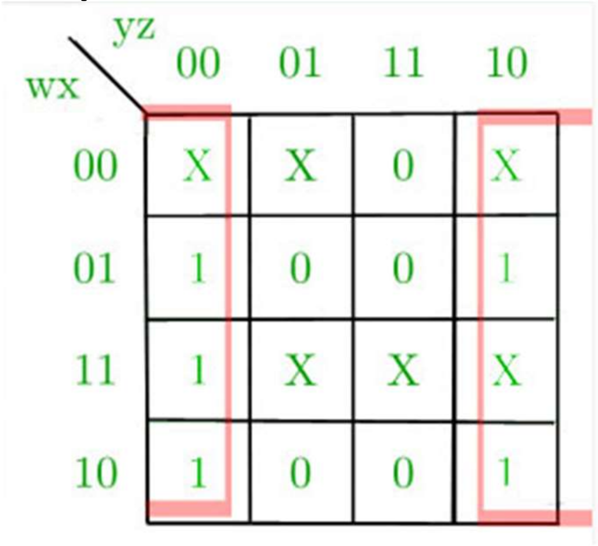
Table 2.5 Excess-3 to BCD Conversion



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

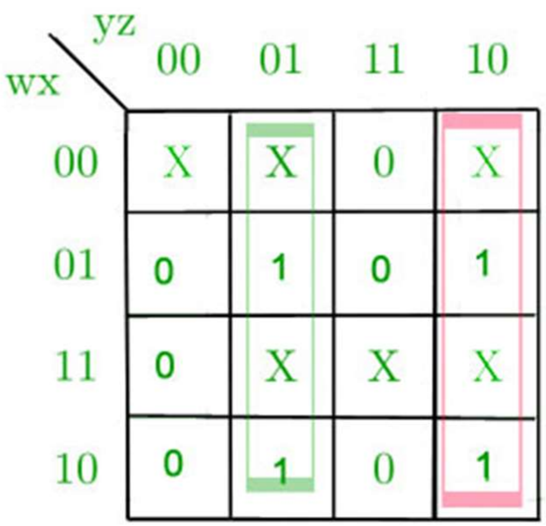
- The K-Map technique is used for each of the BCD code bits as output with all of the bits of the Excess-3 number as input

K-Map for D:



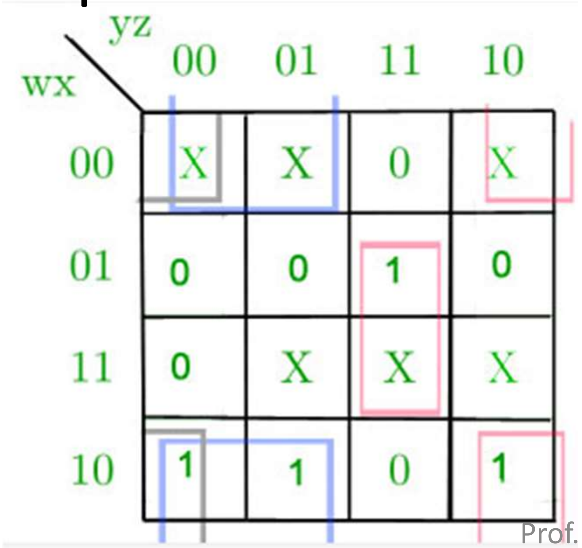
wx \ yz	00	01	11	10
00	X	X	0	X
01	1	0	0	1
11	1	X	X	X
10	1	0	0	1

K-Map for C:



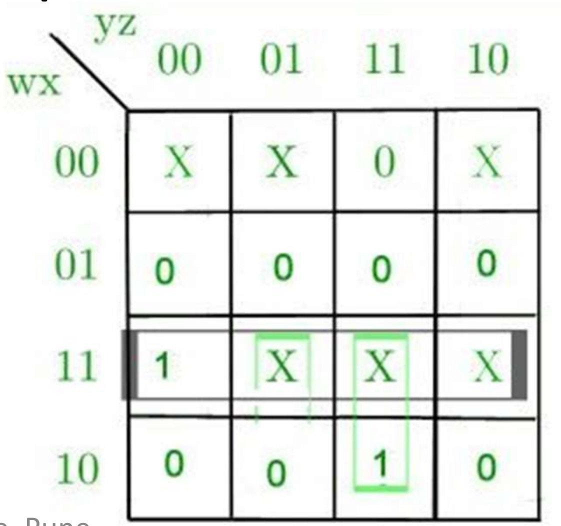
wx \ yz	00	01	11	10
00	X	X	0	X
01	0	1	0	1
11	0	X	X	X
10	0	1	0	1

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.

$$A = wx + wyz$$
$$B = x'y' + x'z' + xyz$$
$$C = y'z + yz'$$
$$D = z'$$

- Figure 2.24 shows the logical implementation of Excess-3 to BCD code conversion

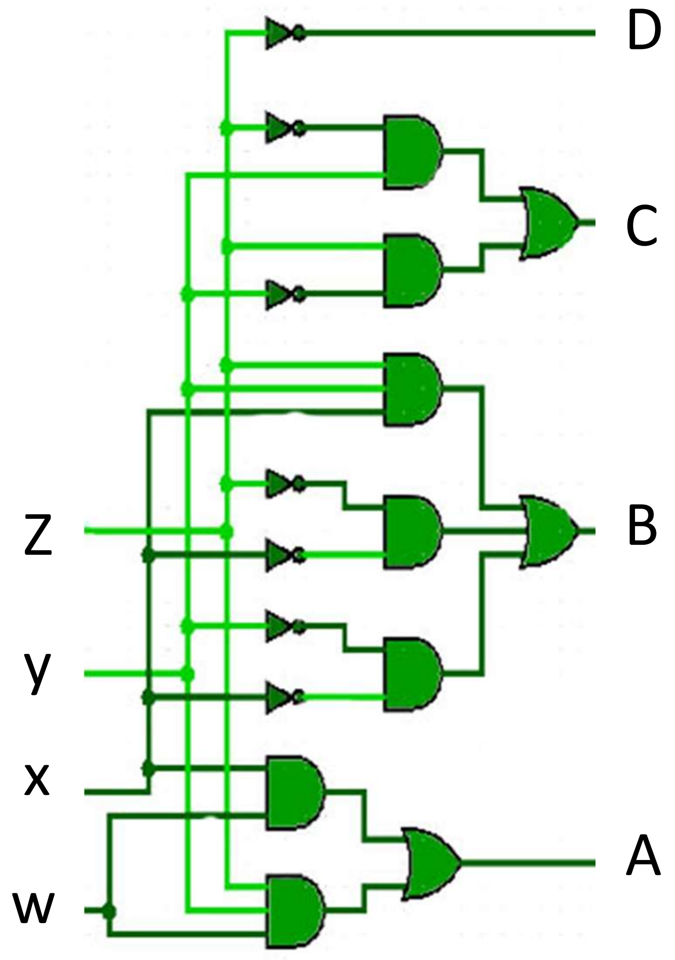


Figure 2.24 logical implementation of Excess-3 to BCD code conversion