operations can be performed on it. The binary results of arithmetic operations must be converted to BCD for transmission to output devices. Therefore, conversions are often accomplished by using the major components of the computer system itself rather than special converter circuits. Conversion tables may be stored in the ROM. In some systems, conversions are accomplished by the computer itself, through execution of a specially designed program. This is called software conversion, as opposed to the hardware conversion performed by logic circuits.

### 4.16.1 Design of a 4-bit Binary-to-Gray Code Converter

The input to the 4-bit binary-to-Gray code converter circuit is a 4-bit binary and the output is a 4-bit Gray code. There are 16 possible combinations of 4-bit binary input and all of them are valid. Hence no don't cares. The 4-bit binary and the corresponding Gray code are shown in the conversion table (Figure 4.32a). From the conversion table, we observe that the expressions for the outputs $G_4$, $G_3$, $G_2$, and $G_1$ are as follows:

$$G_4 = \Sigma m(8, 9, 10, 11, 12, 13, 14, 15)$$
$$G_3 = \Sigma m(4, 5, 6, 7, 8, 9, 10, 11)$$
$$G_2 = \Sigma m(2, 3, 4, 5, 10, 11, 12, 13)$$
$$G_1 = \Sigma m(1, 2, 5, 6, 9, 10, 13, 14)$$

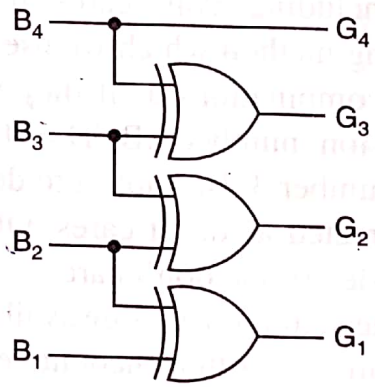
The K-maps for $G_4$, $G_3$, $G_2$, and $G_1$ and their minimization are shown in Figure 4.32b. The minimal expressions for the outputs obtained from the K-map are:

$$G_4 = B_4$$
$$G_3 = \overline{B}_4 B_3 + B_4 \overline{B}_3 = B_4 \oplus B_3$$
$$G_2 = \overline{B}_3 B_2 + B_3 \overline{B}_2 = B_3 \oplus B_2$$
$$G_1 = \overline{B}_2 B_1 + B_2 \overline{B}_1 = B_2 \oplus B_1$$

So, the conversion can be achieved by using three X-OR gates as shown in the logic diagram in Figure 4.32c.

| 4-bit binary | | | | 4-bit Gray | | | |
|---|---|---|---|---|---|---|---|
| $B_4$ | $B_3$ | $B_2$ | $B_1$ | $G_4$ | $G_3$ | $G_2$ | $G_1$ |
| 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 |

(a) Conversion table




(c) Logic diagram

**Figure 4.32** 4-bit binary-to-Gray code converter *(Contd.)*

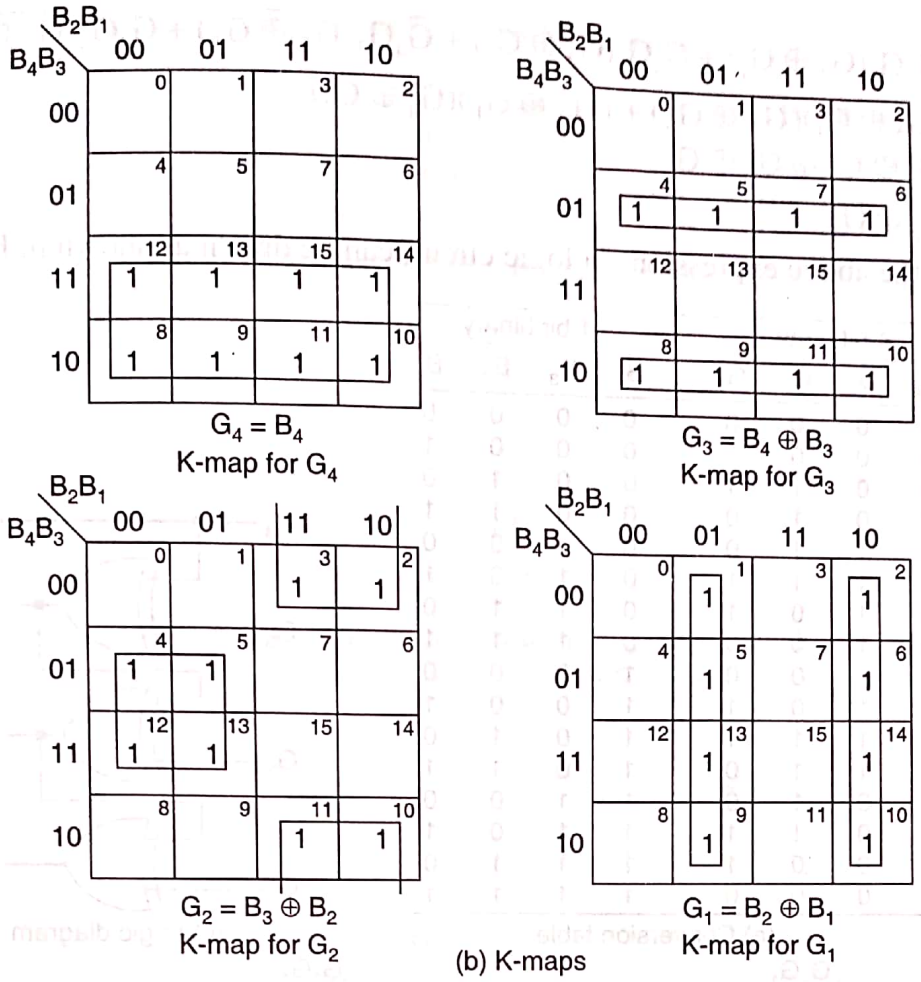



**Figure 4.32** 4-bit binary-to-Gray code converter.

### 4.16.2 Design of a 4-bit Gray-to-Binary Code Converter

The input to the 4-bit Gray-to-binary code converter circuit is a 4-bit Gray code and the output is a 4-bit binary. There are 16 possible combinations of 4-bit Gray input and all of them are valid. Hence no don't cares. The 4-bit input Gray code and the corresponding output binary numbers are shown in the conversion table of Figure 4.33a. From the conversion table we observe that the expressions for the outputs $B_4$, $B_3$, $B_2$ and $B_1$ are:

$$B_4 = \Sigma\, m(12, 13, 15, 14, 10, 11, 9, 8) = \Sigma\, m(8, 9, 10, 11, 12, 13, 14, 15)$$
$$B_3 = \Sigma\, m(6, 7, 5, 4, 10, 11, 9, 8) = \Sigma\, m(4, 5, 6, 7, 8, 9, 10, 11)$$
$$B_2 = \Sigma\, m(3, 2, 5, 4, 15, 14, 9, 8) = \Sigma\, m(2, 3, 4, 5, 8, 9, 14, 15)$$
$$B_1 = \Sigma\, m(1, 2, 7, 4, 13, 14, 11, 8) = \Sigma\, m(1, 2, 4, 7, 8, 11, 13, 14)$$

Drawing the K-maps for $B_4$, $B_3$, $B_2$ and $B_1$ in terms of $G_4$, $G_3$, $G_2$, and $G_1$ as shown in Figure 4.33b and simplifying them, the minimal expressions for the outputs are as follows:

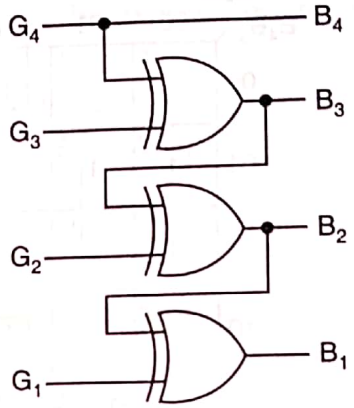
$$B_4 = G_4$$
$$B_3 = \bar{G}_4 G_3 + G_4 \bar{G}_3 = G_4 \oplus G_3$$
$$B_2 = \bar{G}_4 G_3 \bar{G}_2 + \bar{G}_4 \bar{G}_3 G_2 + G_4 \bar{G}_3 \bar{G}_2 + G_4 G_3 G_2$$
$$= \bar{G}_4 (G_3 \oplus G_2) + G_4 (\overline{G_3 \oplus G_2}) = G_4 \oplus G_3 \oplus G_2 = B_3 \oplus G_2$$
$$B_1 = \bar{G}_4 \bar{G}_3 \bar{G}_2 G_1 + \bar{G}_4 \bar{G}_3 G_2 \bar{G}_1 + \bar{G}_4 G_3 G_2 G_1 + \bar{G}_4 G_3 \bar{G}_2 \bar{G}_1 + G_4 G_3 \bar{G}_2 G_1$$
$$+ G_4 G_3 G_2 \bar{G}_1 + G_4 \bar{G}_3 G_2 G_1 + G_4 \bar{G}_3 \bar{G}_2 \bar{G}_1$$

$$= \bar{G}_4\bar{G}_3(G_2 \oplus G_1) + G_4G_3(G_2 \oplus G_1) + \bar{G}_4G_3(\overline{G_2 \oplus G_1}) + G_4\bar{G}_3(\overline{G_2 \oplus G_1})$$
$$= (G_2 \oplus G_1)(\overline{G_4 \oplus G_3}) + (\overline{G_2 \oplus G_1})(G_4 \oplus G_3)$$
$$= G_4 \oplus G_3 \oplus G_2 \oplus G_1$$
$$= B_2 \oplus G_1$$

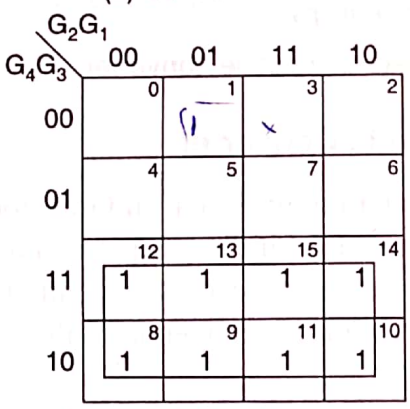
Based on the above expressions, a logic circuit can be drawn as shown in Figure 4.33c.

| 4-bit Gray | | | | 4-bit binary | | | |
|---|---|---|---|---|---|---|---|
| $G_4$ | $G_3$ | $G_2$ | $G_1$ | $B_4$ | $B_3$ | $B_2$ | $B_1$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

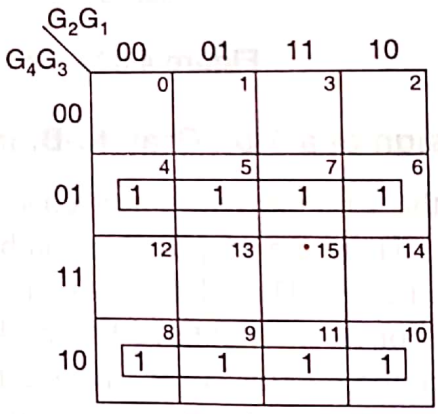
(a) Conversion table



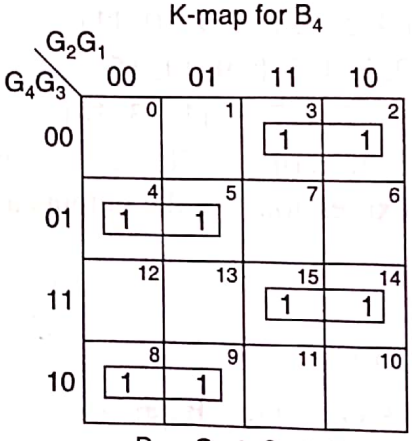

(c) Logic diagram



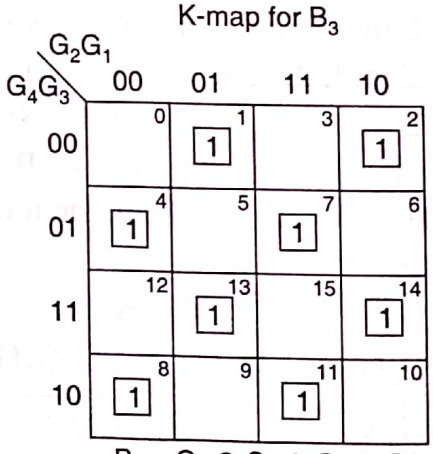

$B_4 = G_4$
K-map for $B_4$




$B_3 = G_4 \oplus G_3$
K-map for $B_3$




$B_2 = G_4 \oplus G_3 \oplus G_2$
K-map for $B_2$




$B_1 = G_4 \oplus G_3 \oplus G_2 \oplus G_1$
K-map for $B_1$

(b) K-maps

**Figure 4.33** 4-bit Gray-to-binary code converter.

### 4.16.3 Design of a 4-bit Binary-to-BCD Code Converter

The input is a 4-bit binary. There are 16 possible combinations of 4-bit binary inputs (representing 0–15) and all are valid. Hence there are no don't cares. Since the input is of 4 bits (i.e. a maximum of 2 decimal digits), the output has to be an 8-bit one; but since the first three bits will all be a 0 for all combinations of inputs, the output can be treated as a 5-bit one. The conversion is shown in the conversion table in Figure 4.34a. From the conversion table, we observe that the expressions for BCD outputs are as follows:

$$A = \Sigma\, m(10, 11, 12, 13, 14, 15)$$
$$B = \Sigma\, m(8, 9)$$
$$C = \Sigma\, m(4, 5, 6, 7, 14, 15)$$
$$D = \Sigma\, m(2, 3, 6, 7, 12, 13)$$
$$E = \Sigma\, m(1, 3, 5, 7, 9, 11, 13, 15)$$

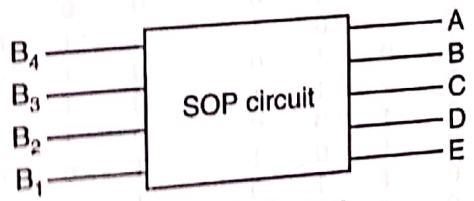
Drawing the K-maps for the outputs and minimizing them as shown in Figure 4.34c the minimal expressions for the BCD outputs A, B, C, D, and E in terms of the 4-bit binary inputs $B_4$, $B_3$, $B_2$, and $B_1$ are as follows:

$$A = B_4B_3 + B_4B_2$$
$$B = B_4\bar{B}_3\bar{B}_2$$
$$C = \bar{B}_4B_3 + B_3B_2$$
$$D = B_4B_3\bar{B}_2 + \bar{B}_4B_2$$
$$E = B_1$$

A logic diagram can be drawn based on the above minimal expressions.

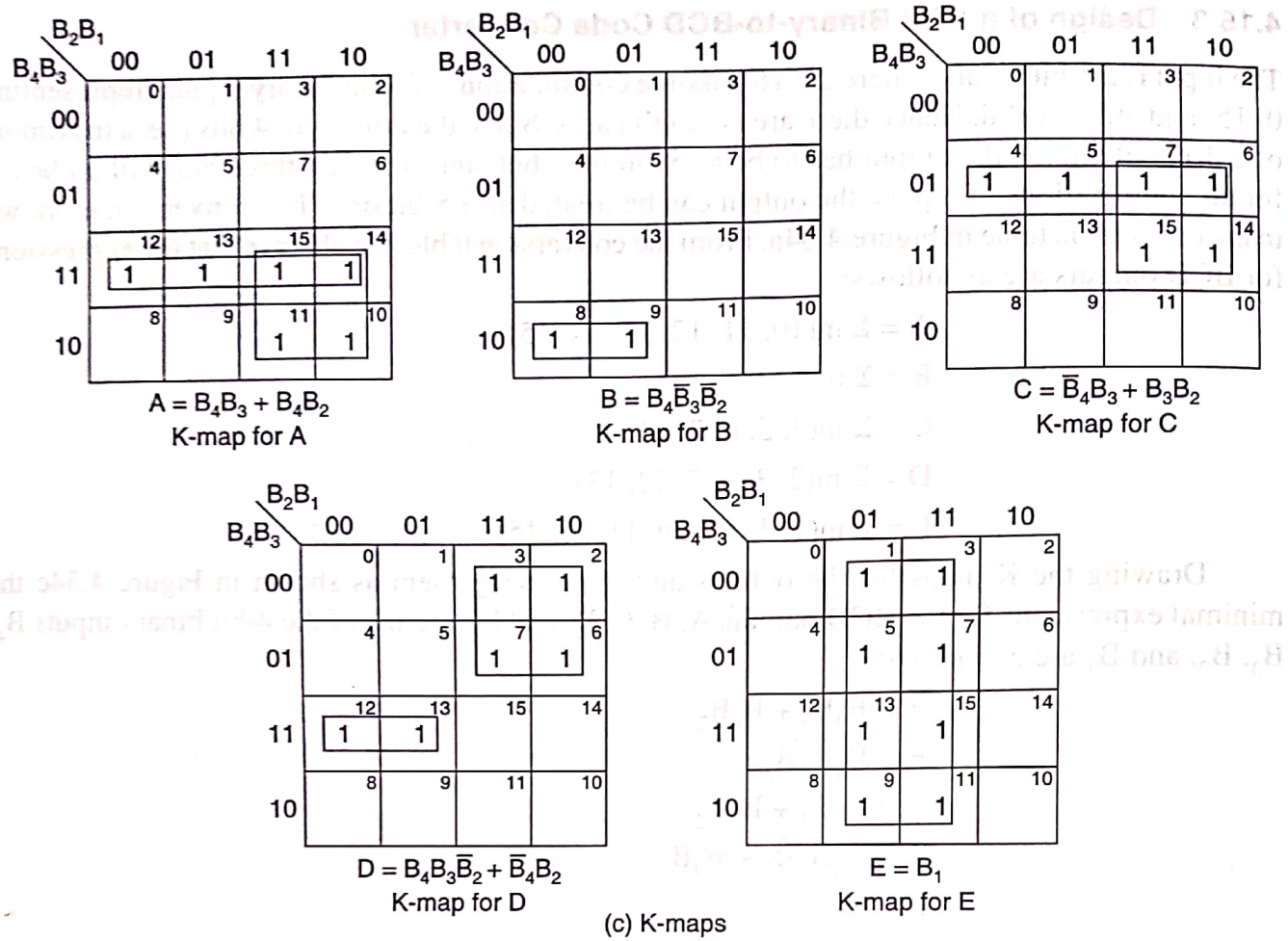
| Decimal | 4-bit binary | | | | BCD output | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $B_4$ | $B_3$ | $B_2$ | $B_1$ | A | B | C | D | E |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 12 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 13 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 14 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 15 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

(a) Conversion table




(b) Block diagram

**Figure 4.34** 4-bit binary-to-BCD code converter *(Contd.)*




(c) K-maps

**Figure 4.34** 4-bit binary-to-BCD code converter.

### 4.16.4 Design of a 4-bit BCD-to-XS-3 Code Converter

BCD means 8421 BCD. The 4-bit input BCD code ($B_4\ B_3\ B_2\ B_1$) and the corresponding output XS-3 code($X_4\ X_3\ X_2\ X_1$) numbers are shown in the conversion table in Figure 4.35a. The input combinations 1010, 1011, 1100, 1101, 1110, and 1111 are invalid in BCD. So they are treated as don't cares.

| 8421 code | | | | XS-3 code | | | |
|---|---|---|---|---|---|---|---|
| $B_4$ | $B_3$ | $B_2$ | $B_1$ | $X_4$ | $X_3$ | $X_2$ | $X_1$ |
| 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 |

(a) Conversion table

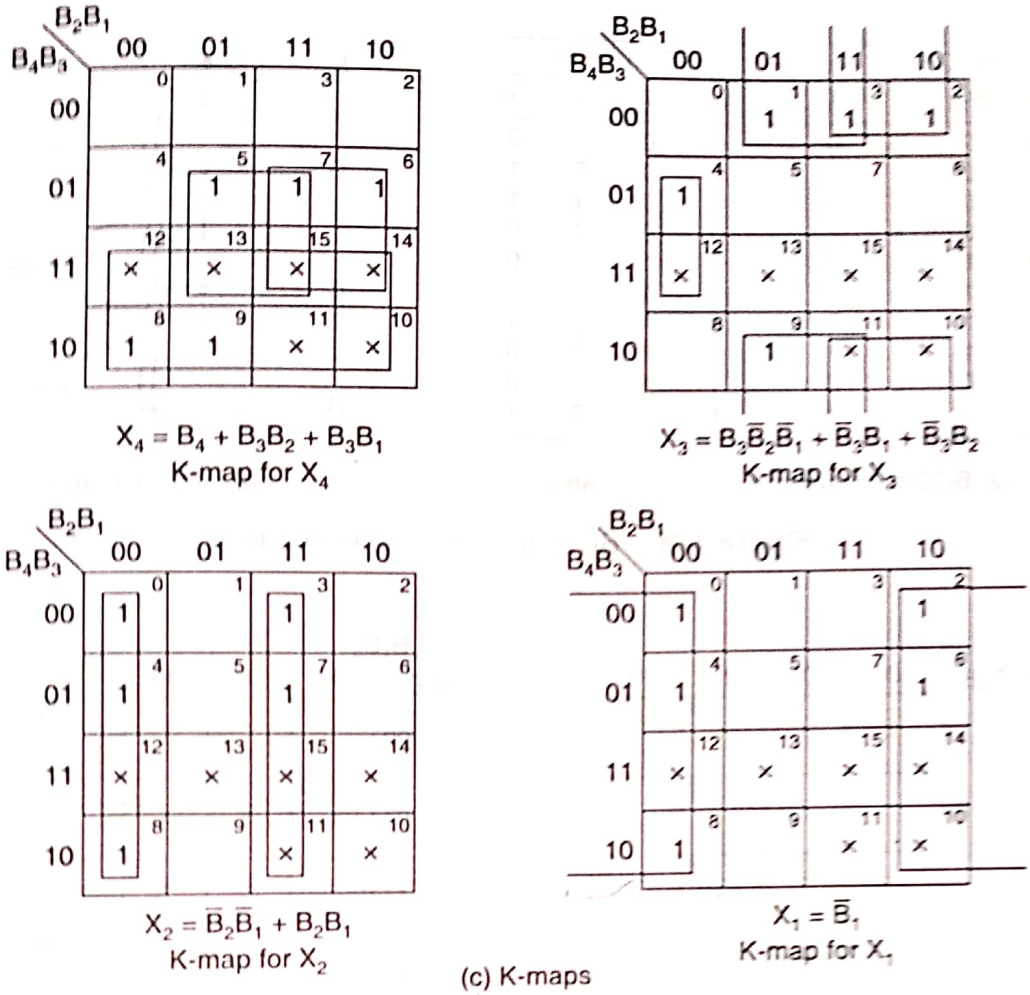
$X_4 = \Sigma\, m(5, 6, 7, 8, 9) + d(10, 11, 12, 13, 14, 15)$
$X_3 = \Sigma\, m(1, 2, 3, 4, 9) + d(10, 11, 12, 13, 14, 15)$
$X_2 = \Sigma\, m(0, 3, 4, 7, 8) + d(10, 11, 12, 13, 14, 15)$
$X_1 = \Sigma\, m(0, 2, 4, 6, 8) + d(10, 11, 12, 13, 14, 15)$

The minimal expressions are

$X_4 = B_4 + B_3B_2 + B_3B_1$
$X_3 = B_3\overline{B}_2\overline{B}_1 + \overline{B}_3B_1 + \overline{B}_3B_2$
$X_2 = \overline{B}_2\overline{B}_1 + B_2B_1$
$X_1 = \overline{B}_1$

(b) Minimal expressions

**Figure 4.35** 4-bit BCD-to-XS-3 code converter *(Contd.)*




(c) K-maps

**Figure 4.35** 4-bit BCD-to-XS-3 code converter.

The expressions for the outputs $X_4$, $X_3$, $X_2$ and $X_1$ are shown in Figure 4.35b. Drawing K-maps for the outputs $X_4$, $X_3$, $X_2$ and $X_1$ in terms of the inputs $B_4$, $B_3$, $B_2$, and $B_1$ and simplifying them, as shown in Figure 4.35c the minimal expressions for $X_4$, $X_3$, $X_2$, and $X_1$ are as shown in Figure 4.35b. A logic diagram can be drawn based on those minimal expressions.

### 4.16.5 Design of a BCD-to-Gray Code Converter

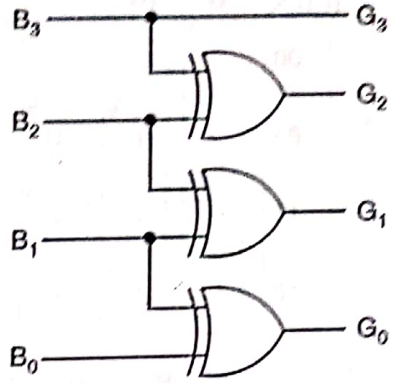
The BCD to Gray code conversion table is shown in Figure 4.36a. For a 4-bit BCD code minterms 10, 11, 12, 13, 14, and 15 are don't cares. So the expressions for the Gray code outputs in terms of BCD inputs are as follows:

$$G_3 = \Sigma\, m(8, 9) + d\,(10, 11, 12, 13, 14, 15)$$
$$G_2 = \Sigma\, m(4, 5, 6, 7, 8, 9) + d\,(10, 11, 12, 13, 14, 15)$$
$$G_1 = \Sigma\, m(2, 3, 4, 5) + d\,(10, 11, 12, 13, 14, 15)$$
$$G_0 = \Sigma\, m(1, 2, 5, 6, 9) + d\,(10, 11, 12, 13, 14, 15)$$

The K-maps for $G_3$, $G_2$, $G_1$, and $G_0$, their minimization, and the minimal expressions obtained from them are shown in Figure 4.37. The logic diagram of the BCD to Gray code converter based on those minimal expressions is shown in Figure 4.36b.

| BCD code | | | | 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 |

(a) BCD-to-Gray code conversion table




(b) Logic diagram

**Figure 4.36** BCD-to-Gray code converter.

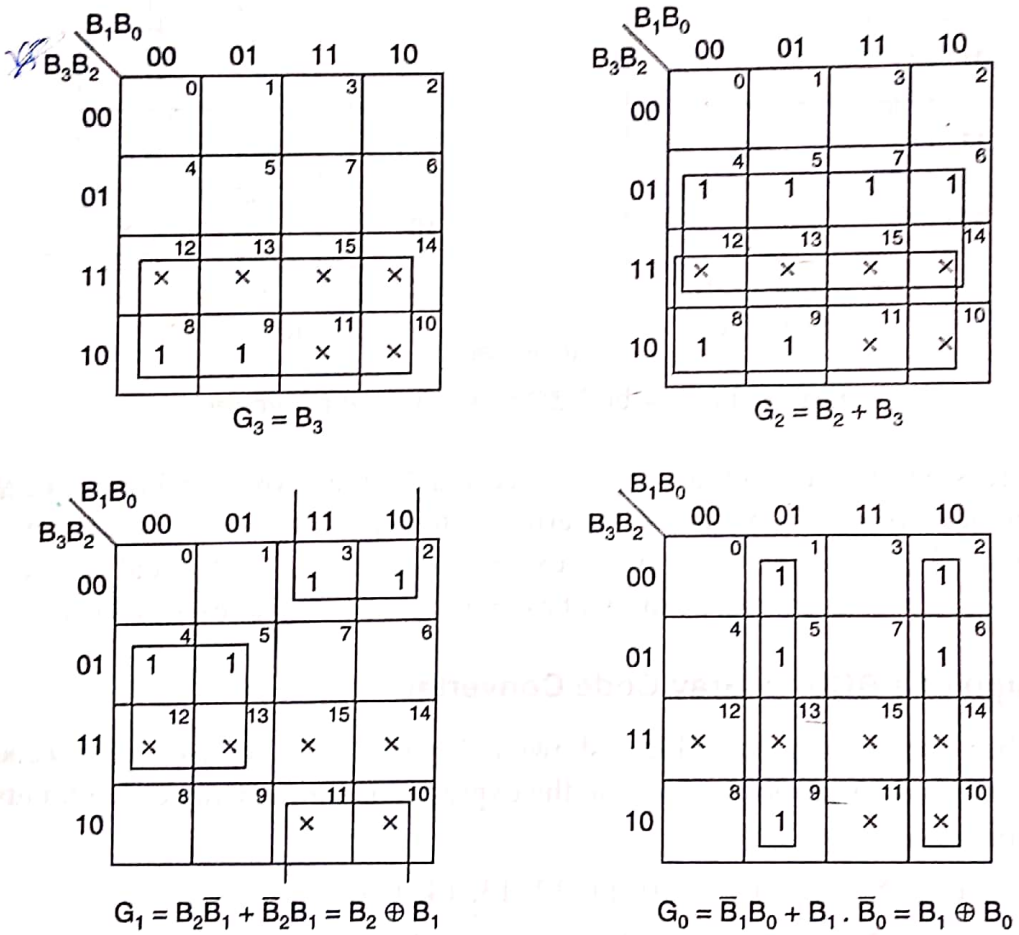



**Figure 4.37** K-maps for a BCD-to-Gray code converter.

### 4.16.6 Design of an SOP Circuit to Detect the Decimal Numbers 5 through 12 in a 4-bit Gray Code Input

The input to the SOP circuit is a 4-bit Gray code. Let the input Gray code be ABCD. There are 16 possible combinations of 4-bit Gray code. All of them are valid and hence there are no don't cares.

The truth table of the SOP circuit is shown in Figure 4.38a. Looking at the truth table of the SOP circuit, we observe that the output is 1 for the input combinations corresponding to minterms 7, 5, 4, 12, 13, 15, 14, and 10 (i.e. corresponding to the Gray code of decimal numbers 5, 6, 7, 8, 9, 10, 11 and 12). So the expression for the output is

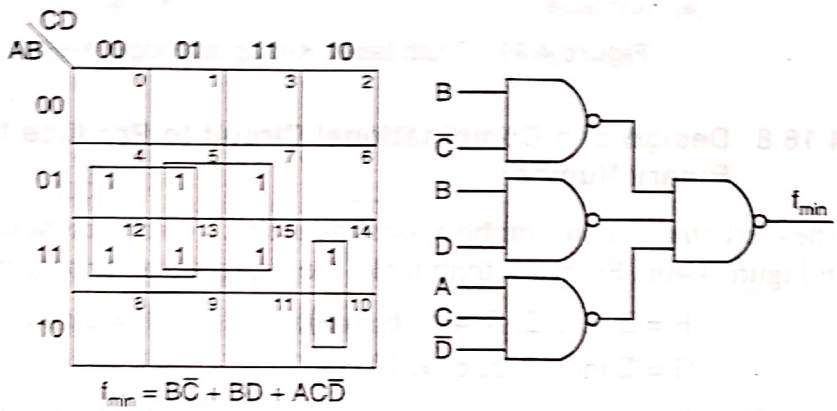
$$f = \Sigma\, m(7, 5, 4, 12, 13, 15, 14, 10) = \Sigma\, m(4, 5, 7, 10, 12, 13, 14, 15)$$

The K-map for f, its minimization, the minimal expression obtained from it and its realization in NAND logic are shown in Figures 4.38b and c respectively.

$$f_{min} = B\overline{C} + BD + AC\overline{D} = \overline{\overline{B\overline{C}} \cdot \overline{BD} \cdot \overline{AC\overline{D}}}$$

| Decimal number | 4-bit Gray code A | B | C | D | Output f |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 1 | 0 |
| 3 | 0 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 1 | 1 | 1 |
| 6 | 0 | 1 | 0 | 1 | 1 |
| 7 | 0 | 1 | 0 | 0 | 1 |
| 8 | 1 | 1 | 0 | 0 | 1 |
| 9 | 1 | 1 | 0 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 | 0 | 1 |
| 12 | 1 | 0 | 1 | 0 | 1 |
| 13 | 1 | 0 | 1 | 1 | 0 |
| 14 | 1 | 0 | 0 | 1 | 0 |
| 15 | 1 | 0 | 0 | 0 | 0 |

(a) Truth table




(b) K-map (c) NAND logic

**Figure 4.38** Truth table, K-map and logic diagram for the SOP circuit.

### 4.16.7 Design of an SOP Circuit to Detect the Decimal Numbers 0, 2, 4, 6, and 8 in a 4-bit 5211 BCD Code Input

The input to the SOP circuit is a 5211 BCD code. Let it be ABCD. It is a 4-bit input. Therefore, there are 16 possible combinations of inputs, out of which only the 10 combinations shown in the truth table of Figure 4.39a are used to code the decimal digits in 5211 code. The remaining 6 combinations 0010, 0100, 0110, 1001, 1011, and 1101 are invalid. So, the corresponding outputs are don't cares (i.e. minterms 2, 4, 6, 9, 11, and 13 are don't cares). Looking at the truth table of the SOP circuit shown in Figure 4.39a we observe that the output is 1 for the input combinations corresponding to minterms 0, 3, 7, 10, 14 (i.e. corresponding to 5211 code of decimal numbers 0, 2, 4, 6, and 8).

So the problem may be stated as

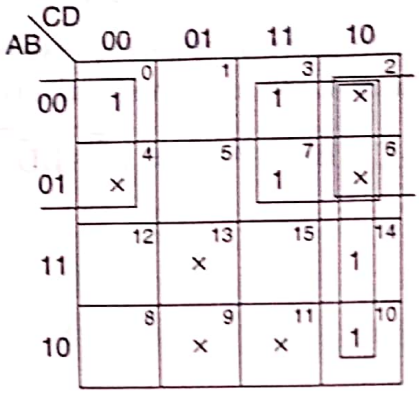
$$F = \Sigma\, m(0, 3, 7, 10, 14) + d(2, 4, 6, 9, 11, 13)$$

The K-map, its minimization, the minimal expression obtained from it and the realization of the minimal expression in NAND logic are shown in Figure 4.39.

$$f_{min} = \bar{A}\bar{D} + \bar{A}C + C\bar{D} = \overline{\overline{\bar{A}\bar{D}} \cdot \overline{\bar{A}C} \cdot \overline{C\bar{D}}}$$

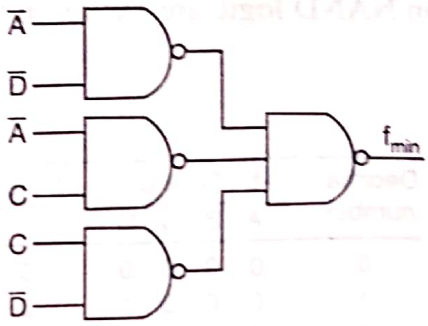
| Decimal number | 5211 code A | B | C | D | Output f |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 1 | 1 |
| 3 | 0 | 1 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 | 1 | 1 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 6 | 1 | 0 | 1 | 0 | 1 |
| 7 | 1 | 1 | 0 | 0 | 0 |
| 8 | 1 | 1 | 1 | 0 | 1 |
| 9 | 1 | 1 | 1 | 1 | 0 |

(a) Truth table




$f_{min} = \bar{A}\bar{D} + \bar{A}C + C\bar{D}$

(b) K-map




(c) Logic diagram

**Figure 4.39** Truth table, K-map and logic diagram for the SOP circuit.

### 4.16.8 Design of a Combinational Circuit to Produce the 2's Complement of a 4-bit Binary Number

The 4-bit binary input combinations and their 2's complement versions are shown in the truth table in Figure 4.40a. From the truth table, the expressions for outputs E, F, G, and H are:

$$E = \Sigma\, m(1, 2, 3, 4, 5, 6, 7, 8), \qquad F = \Sigma\, m(1, 2, 3, 4, 9, 10, 11, 12),$$

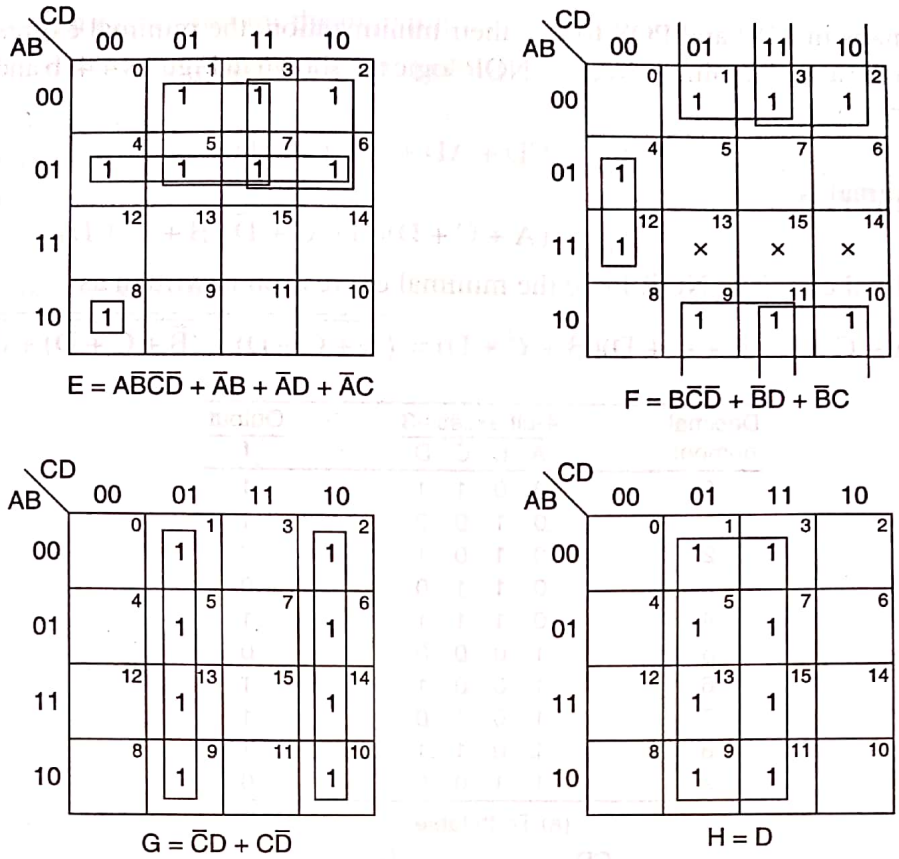
$$G = \Sigma\, m(1, 2, 5, 6, 9, 10, 13, 14), \qquad H = \Sigma\, m(1, 3, 5, 7, 9, 11, 13, 15)$$

The K-maps for the output expressions, their minimization and the minimal expressions obtained from them are shown in Figure 4.40b.

| Input A | B | C | D | Output E | F | G | H |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

(a) Conversion table

**Figure 4.40** Conversion table and K-maps for the circuit *(Contd.)*




(b) K-maps

**Figure 4.40** Conversion table and K-maps for the circuit.

After simplification the minimal expressions are

$E = A\bar{B}\bar{C}\bar{D} + \bar{A}B + \bar{A}D + \bar{A}C, \quad F = B\bar{C}\bar{D} + \bar{B}D + \bar{B}C, \quad G = \bar{C}D + C\bar{D}, \quad H = D$

A logic circuit can be drawn based on these minimal expressions.