# Code Converter

**Code converters:**

The availability of a large variety of codes for the same discrete elements of information results in the use of different codes by different digital systems. It is sometimes necessary to use the output of one system as the input to another. A conversion circuit must be

inserted between the two systems if each uses different codes for the same information. Thus a code converter is a logic circuit whose inputs are bit patterns representing numbers (or character) in one cod and whose outputs are the corresponding representation in a different code. Code converters are usually multiple output circuits.

To convert from binary code A to binary code B, the input lines must supply the bit combination of elements as specified by code A and the output lines must generate the corresponding bit combination of code B. A combinational circuit performs this transformation by means of logic gates.

For example, a binary –to-gray code converter has four binary input lines $B_4$, $B_3$,$B_2$,$B_1$ and four gray code output lines $G_4$,$G_3$,$G_2$,$G_1$. When the input is 0010, for instance, the output should be 0011 and so forth. To design a code converter, we use a code table treating it as a truth table to express each output as a Boolean algebraic function of all the inputs.

In this example, of binary –to-gray code conversion, we can treat the binary to the gray code table as four truth tables to derive expressions for $G_4$, G3, G2, and G1. Each of these four expressions would, in general, contain all the four input variables $B_4$, B3,$B_2$,and B1. Thus,this code converter is actually equivalent to four logic circuits, one for each of the truth tables.

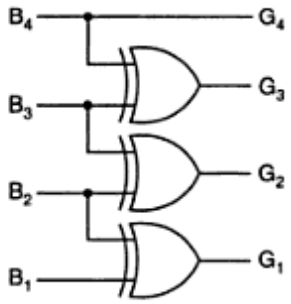
The logic expression derived for the code converter can be simplified using the usual techniques, including ‗don't cares' if present. Even if the input is an unweighted code, the same cell numbering method which we used earlier can be used, but the cell numbers --must correspond to the input combinations as if they were an 8-4-2-1 weighted code. s

## Design of a 4-bit binary to gray code converter:

$$G_4 = \Sigma m(8, 9, 10, 11, 12, 13, 14, 15) \qquad G_4 = B_4$$
$$G_3 = \Sigma m(4, 5, 6, 7, 8, 9, 10, 11) \qquad G_3 = \bar{B}_4B_3 + B_4\bar{B}_3 = B_4 \oplus B_3$$
$$G_2 = \Sigma m(2, 3, 4, 5, 10, 11, 12, 13) \qquad G_2 = \bar{B}_3B_2 + B_3\bar{B}_2 = B_3 \oplus B_2$$
$$G_1 = \Sigma m(1, 2, 5, 6, 9, 10, 13, 14) \qquad G_1 = \bar{B}_2B_1 + B_2\bar{B}_1 = B_2 \oplus B_1$$

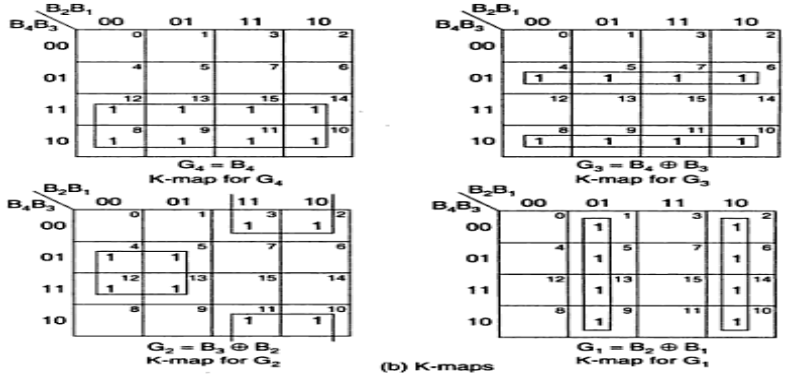
| 4-bit binary | | | | 4-bit Gray | | | |
|---|---|---|---|---|---|---|---|
| $B_4$ | $B_3$ | $B_2$ | $B_1$ | $G_4$ | $G_3$ | $G_2$ | $G_1$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1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

4-bit binary-to-Gray code converter




(b) K-maps

4-bit binary-to-Gray code converter.

Design of a 4-bit gray to Binary code converter:

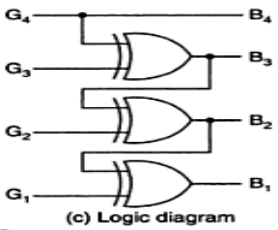
$$B_4 = \Sigma\, m(12, 13, 15, 14, 10, 11, 9, 8) = \Sigma\, m(8, 9, 10, 11, 12, 13, 14, 15)$$
$$B_3 = \Sigma\, m(6, 7, 5, 4, 10, 11, 9, 8) = \Sigma\, m(4, 5, 6, 7, 8, 9, 10, 11)$$
$$B_2 = \Sigma\, m(3, 2, 5, 4, 15, 14, 9, 8) = \Sigma\, m(2, 3, 4, 5, 8, 9, 14, 15)$$
$$B_1 = \Sigma\, m(1, 2, 7, 4, 13,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

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

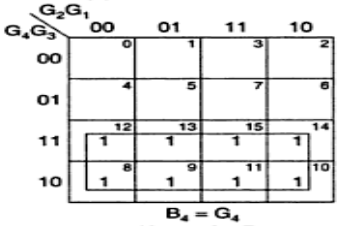
$$= \bar{G}_4 \bar{G}_3 (G_2 \oplus G_1) + G_4 G_3 (G_2 \oplus G_1) + \bar{G}_4 G_3 (\overline{G_2 \oplus G_1}) + G_4 \bar{G}_3 (\overline{G_2 \oplus G_1})$$
$$= (G_2 \oplus G_1)(\overline{G_4 \oplus G_3}) + (\overline{G_2 \oplus G_1})(G_4 \oplus G_3)$$
$$= G_4 \oplus G_3 \oplus G_2 \oplus G_1$$

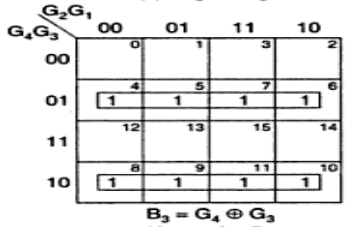
| 4-bit Gray | | | | 4-bit binary | | | |
|---|---|---|---|---|---|---|---|
| $G_4$ | $G_3$ | $G_2$ | $G_1$ | $B_4$ | $B_3$ | $B_2$ | $B_1$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1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

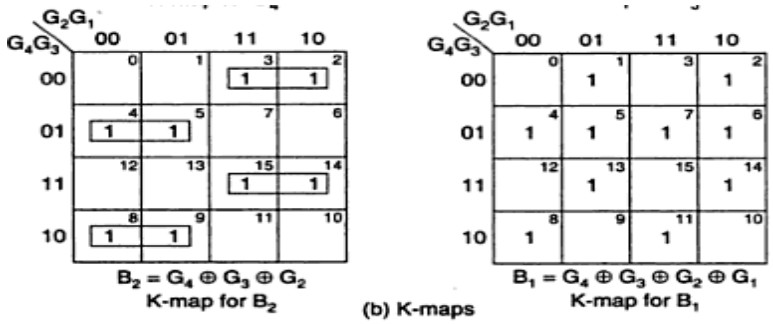
(a) Conversion table




(c) Logic diagram






(b) K-maps

4-bit Gray-to-binary code converter.

**Design of a 4-bit BCD to XS-3 code converter:**

| 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

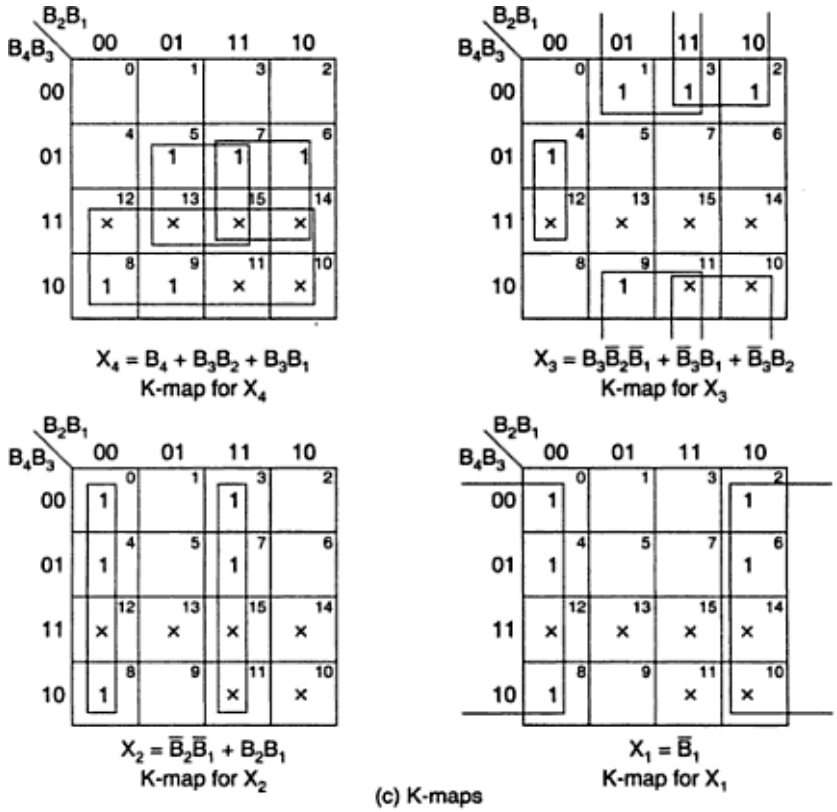
$X_4 = \Sigma\, m(5, 6, 7, 8, 9) + d(10, 11, 12, 13, 14, 15)$
$X_3 = \Sigma\, m(1, 2, 3, 4, 9) + d(10, 11, 12, 13, 14, 15)$
$X_2 = \Sigma\, m(0, 3, 4, 7, 8) + d(10, 11, 12, 13, 14, 15)$
$X_1 = \Sigma\, m(0, 2, 4, 6, 8) + d(10, 11, 12, 13, 14, 15)$

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

4-bit BCD-to-XS-3 code converter



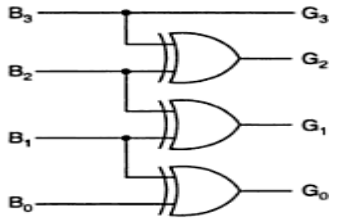

(c) K-maps

4-bit BCD-to-XS-3 code converter.

**Design of a BCD to gray code converter:**

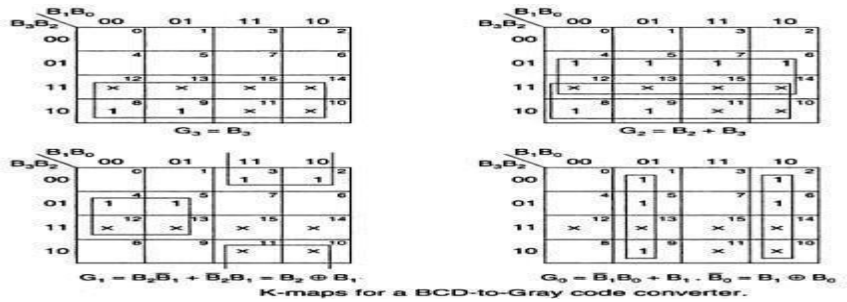
| BCD code | | | | Gray code | | | |
|---|---|---|---|---|---|---|---|
| $B_3$ | $B_2$ | $B_1$ | $B_0$ | $G_3$ | $G_2$ | $G_1$ | $G_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 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

BCD-to-Gray code converter.

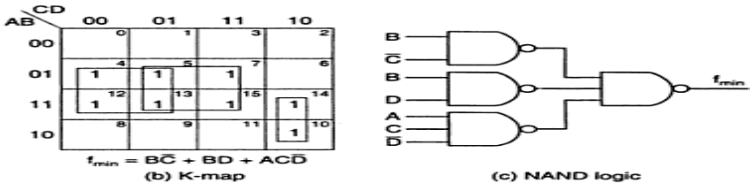



K-maps for a BCD-to-Gray code converter.

**Design of a SOP circuit to Detect the Decimal numbers 5 through 12 in a 4-bit gray code Input:**

| 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



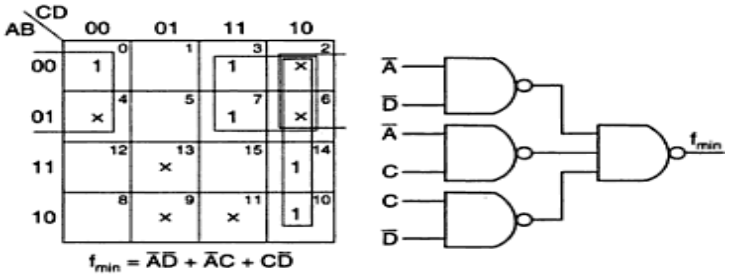

(b) K-map (c) NAND logic

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

**Design of a SOP circuit to detect the decimal numbers 0,2,4,6,8 in a 4-bit 5211 BCD code input:**

| 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




(b) K-map (c) Logic diagram

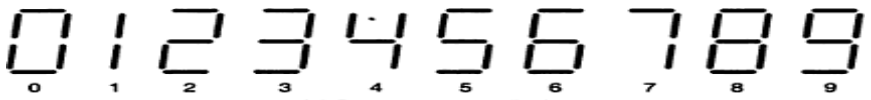
Truth table, K-map and logic diagram for the SOP circuit.

**Design of a Combinational circuit to produce the 2's complement of a 4-bit binary number:**

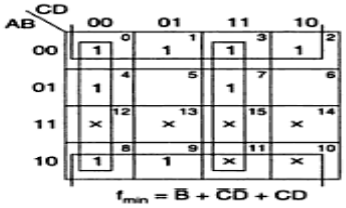
| Input | | | | Output | | | |
|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1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

Conversion table and K-maps for the circuit

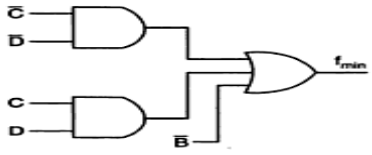



(a) Seven-segment display




$f_{min} = \bar{B} + \bar{C}\bar{D} + CD$

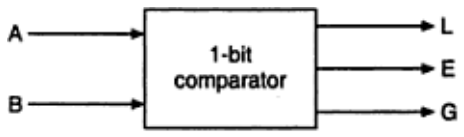
(b) K-map




(c) Logic diagram

**Comparators:**

$$\text{EQUALITY} = (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$$




Block diagram of a 1-bit comparator.

The logic for a 1-bit magnitude comparator: Let the 1-bit numbers be $A = A_0$ and $B = B_0$.
If $A_0 = 1$ and $B_0 = 0$, then $A > B$.
Therefore,

$$A > B: G = A_0\overline{B}_0$$

If $A_0 = 0$ and $B_0 = 1$, then $A < B$.
Therefore,

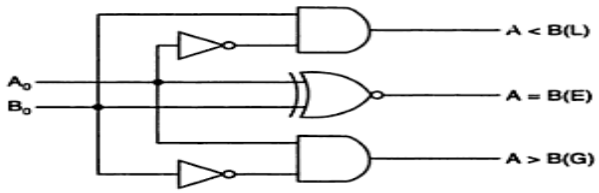
$$A < B: L = \overline{A}_0 B_0$$

If $A_0$ and $B_0$ coincide, i.e. $A_0 = B_0 = 0$ or if $A_0 = B_0 = 1$, then $A = B$.
Therefore,

$$A = B : E = A_0 \odot B_0$$

| $A_0$ | $B_0$ | L | E | G |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |

(a) Truth table




(b) Logic diagram

1-bit comparator.

## 1. Magnitude Comparator:

### 1- bit Magnitude Comparator:

The logic for a 2-bit magnitude comparator: Let the two 2-bit numbers be $A = A_1 A_0$ and $B = B_1 B_0$.

1. If $A_1 = 1$ and $B_1 = 0$, then $A > B$ or
2. If $A_1$ and $B_1$ coincide and $A_0 = 1$ and $B_0 = 0$, then $A > B$. So the logic expression for $A > B$ is

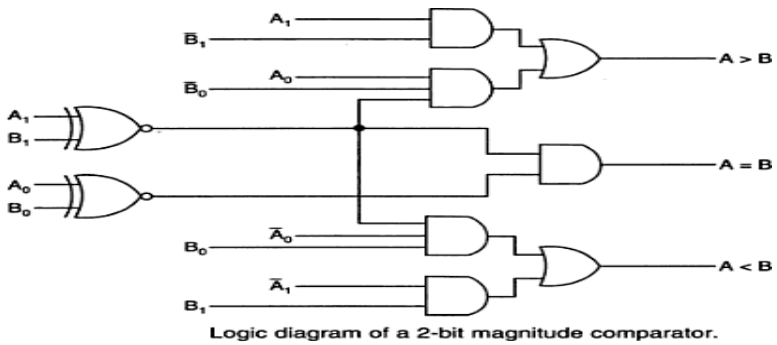
$$A > B : G = A_1\overline{B}_1 + (A_1 \odot B_1)A_0\overline{B}_0$$

1. If $A_1 = 0$ and $B_1 = 1$, then $A < B$ or
2. If $A_1$ and $B_1$ coincide and $A_0 = 0$ and $B_0 = 1$, then $A < B$. So the expression for $A < B$ is

$$A < B : L = \overline{A}_1 B_1 + (A_1 \odot B_1)\overline{A}_0 B_0$$

If $A_1$ and $B_1$ coincide and if $A_0$ and $B_0$ coincide then $A = B$. So the expression for $A = B$ is

$$A = B : E = (A_1 \odot B_1)(A_0 \odot B_0)$$




Logic diagram of a 2-bit magnitude comparator.