## • Using Adders For Subtractor

With A few a little thinks we can use as adder to also do subtraction .there is a mathematical technique that helps us use an adder to do binary subtraction.

**Exp: Decimal subtraction Binary Subtraction**

**10 1010**

**1010**

**- 6 - 0110** → **1's complement and add**

**+1001**

---

**4 100** $\boxed{1}$

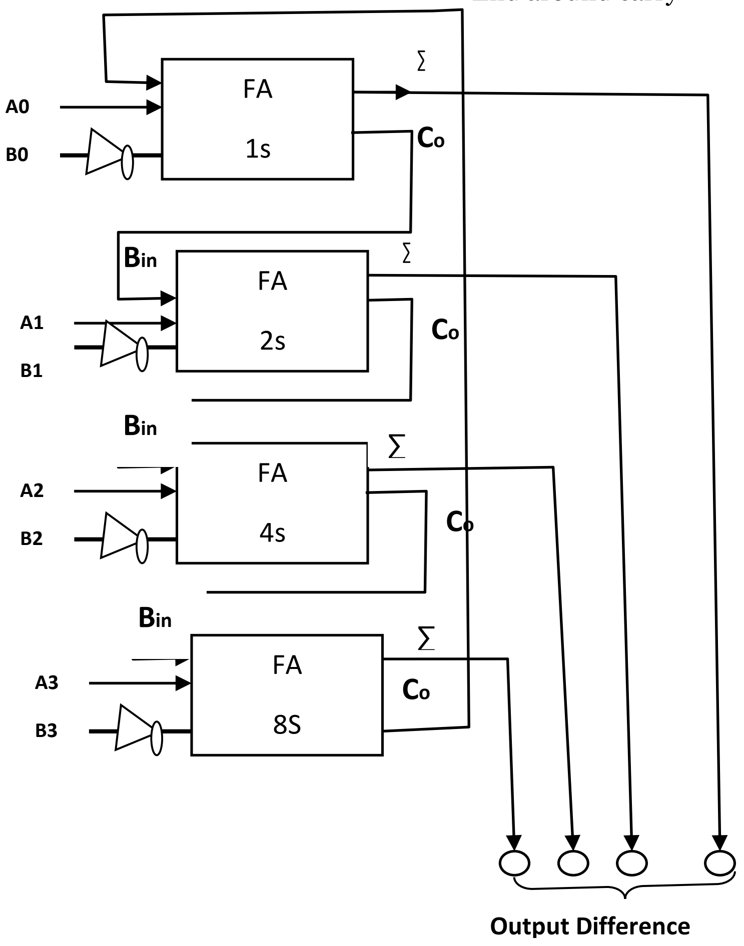
**0011**

End around carry +

**1**

---

**100**



In this special technique the steps are first to first wire the 1's complement of the number being subtracted (change all 1's to 0's and all 0'sto 1's ) and then add .

Now let us us adders to do binary subtraction in this example, the temporary answer to this addition is shown as (10011).next , the last carry on the left I carried around to the 1's place. This is called an end-around carry. When the end around carry is added to the rest of the number, the result is the difference between the original binary numbers , (1010) and (0110). The an swer to this problem is 0100.

## 1- Comparator

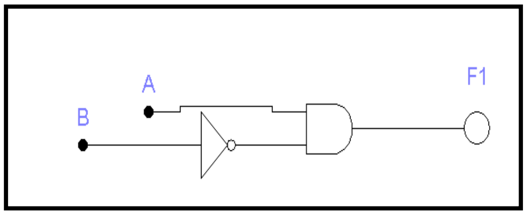
The comparison of two numbers is an operations that determines if one number is greater then , less than , or equal to other number.

**A Magnitude comparator** : is a combinational circuit that compares two numbers A and B , and determines there relative . The outcome of the comparison is specified by three binary variables that indicate whether A > B , A=B or A <B .

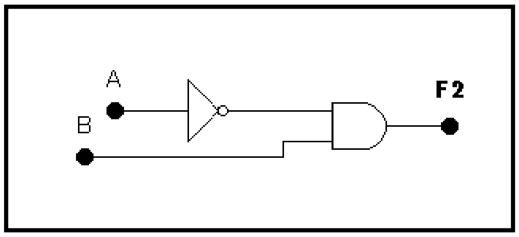
| A | B | A>B | A<B | A=B | A>=B | A<=B | A<>B |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |

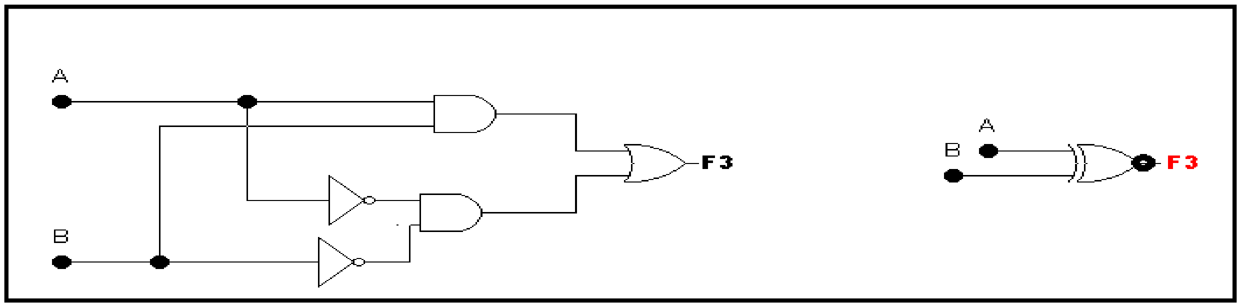
## ♦by us logic circuit

$1-\boldsymbol{A} > \boldsymbol{B} \Leftrightarrow \boldsymbol{F}1=\boldsymbol{A}\overline{\boldsymbol{B}}$



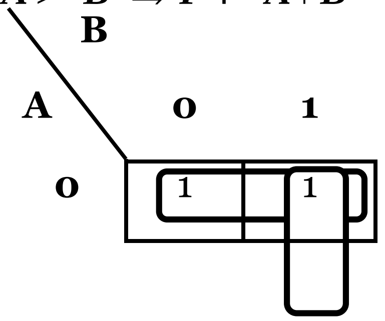
$2 - A < \mathbf{B} \Rightarrow F2 = \overline{A}B$



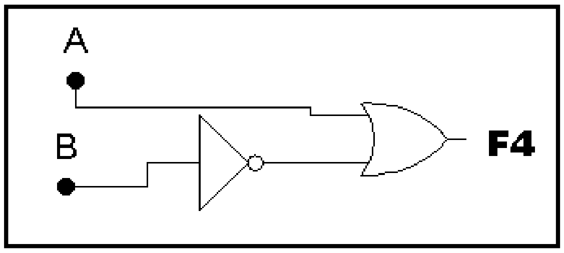


⟺

$4 - A >= B \Rightarrow F4 = A + \overline{B}$

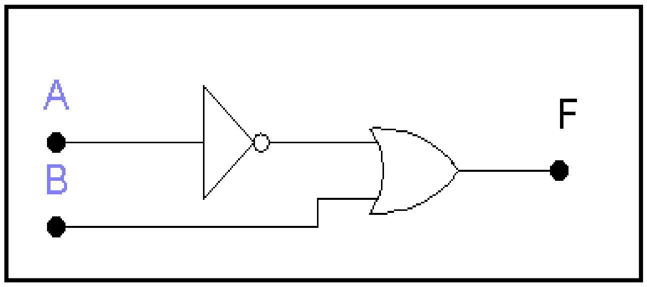


| | | |
|---|---|---|
| **1** | | 1 |



$5-\ A <= B \Rightarrow F5 = \overline{A} + B$

| A \ B | **0** | **1** |
|---|---|---|
| **0** | 1 | |
| **1** | 1 | 1 |



This is a comparison of two binary variables each has one bit only .

**Note : Now what about comparing two numbers with n-bits ?**

$$Let\ \ A = a_{n-1}\, a_{n-2} \ldots\ldots\ldots a_1 a_0$$
$$Let\ \ B = b_{n-1}\, b_{n-2} \ldots\ldots\ldots b_1 b_0$$

where $a_0 , b_0$ is **LSB** while $a_{n-1} , b_{n-2}$ is **MSB** of two number

## Equality Relation

We say that if the number A equal to B we are implement an XNOR gate to do this state , but with one bit number, then now about numbers with n- bits ?

**Note:** the two numbers are equal if all pairs of significant digit are equal:

$$a_{n-1} = b_{n-1} \;,\; \ldots\ldots\ldots\ldots\ldots\ldots , \; a_0 = b_0$$

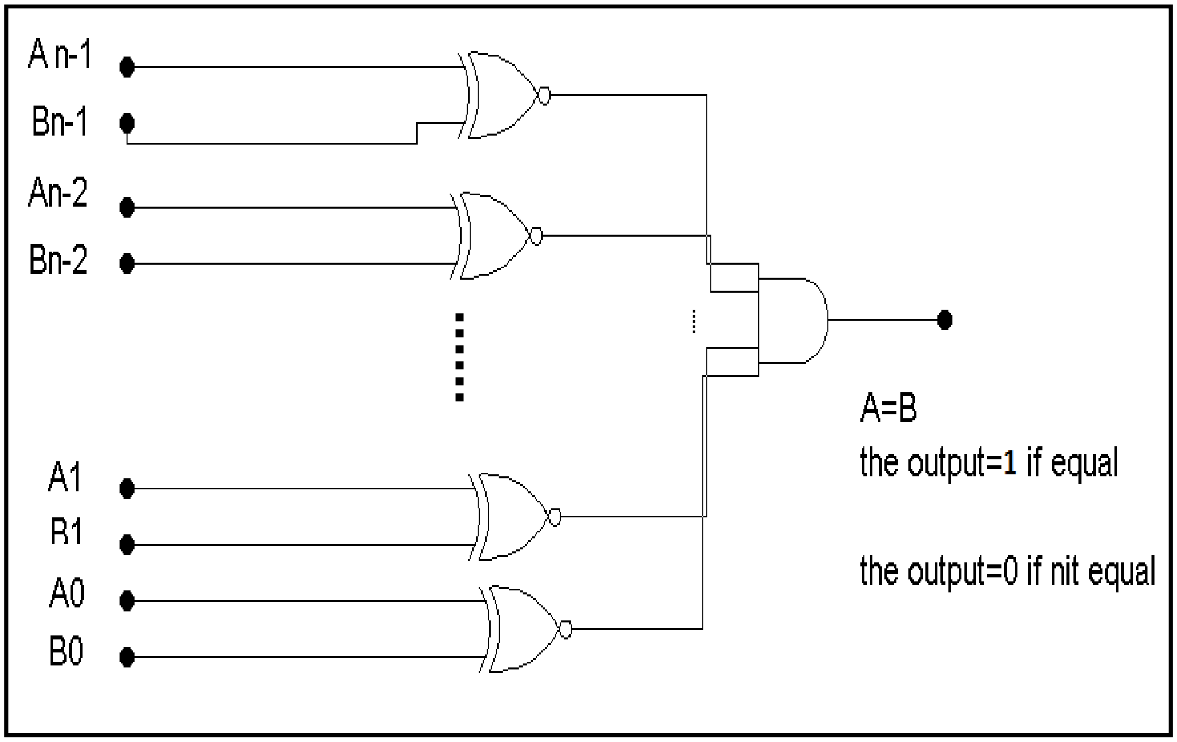
$$\xrightarrow{\hspace{2.5cm}}$$

That means for the equality condition being true (equal 1) if all equality relation of each pair must equal to 1, this dictates **AND** gate to combine the

outputs to gather to get the final output 1.

## Therefore Equality Relation:-

$$\boxed{A = B \;\; if \;\; (\overline{a_{n-1} \oplus b_{n-1}}).(\overline{a_{n-2} \oplus b_{n-2}}) \;\ldots\ldots\ldots\ldots\ldots\ldots, (\overline{a \;\; \oplus b_0}) = 1}$$



## ♦ Greater than relation

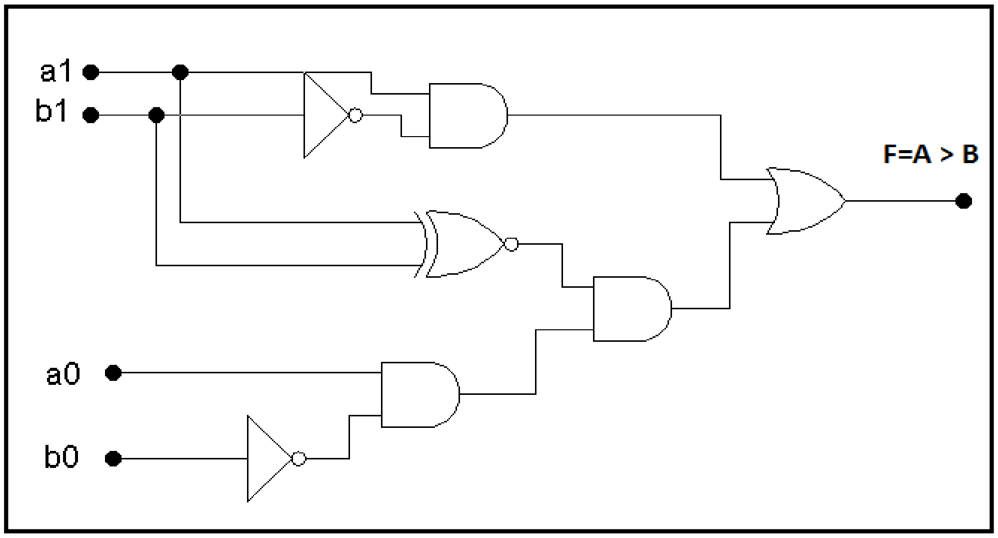
If the corresponding digit of  is (1) and that of B is (0) , we conclude that A >B.

***Then How can we implement a comparator circuit to do this comparison:***

**1- if N= 2**

$$Let \quad A = a_1 a_0$$
$$Let \quad B = b_1 b_0$$
$$to\ prof\ that \quad A > B \ if \quad a_1 a_0 > b_1 b_0$$
$$= (a_1 > b_1) + (a_1 = b_1).(a_0 > b_0)$$
$$and \quad from \quad truth table:$$
$$(a_1 > b_1) = a_1 \overline{b_1}$$
$$(a_1 = b_1) = (\overline{a_1 \oplus b_1})$$
$$(a_0 > b_0) = a_0 \overline{b_0}$$

$$then \quad F(A > B) = a_1 \overline{b_1} + (\overline{a_1 \oplus b_1}).a_0 \overline{b_0}$$



**2- if N= 3**

$Let\ \ A = a_2 a_1 a_0$

$Let\ \ B = b_2 b_1 b_0$

*then*

$$F(A > B) = (a_2 > b_2) + (a_2 = b_2) . (a_1 > b_1) + (a_2 = b_2) . (a_1 = b_1) . (a_0 > b_0)$$

$$\because (a_2 > b_2) = a_2 \overline{b_2}$$

$$(a_2 = b_2) = (\overline{a_2 \oplus b_2})$$

$$(a_1 > b_1) = a_1 \overline{b_1}$$

$$(a_1 = b_1) = (\overline{a_1 \oplus b_1})$$

$$(a_0 > b_0) = a_0 \overline{b_0}$$

$$\therefore A > \mathrm{B} = (a_2 \overline{b_2}) + (\overline{a_2 \oplus b_2}) . (a_1 \overline{b_1}) + (\overline{a_2 \oplus b_2}) . (\overline{a_1 \oplus b_1}) . (a_0 \overline{b_0})$$

**Note : in the same way of comparison we can find**

**- If N=2**

$$Then\ \ A < B = \overline{a_1}\, b_1 + (\overline{a_1 \oplus b_1}) . \overline{a_0}\, b_0$$

**- If N= 3**

$$\text{Then}\ \ A < B = (\overline{a_2}\, b_2) + (\overline{a_2 \oplus b_2}) . (\overline{a_1}\, b_1) + (\overline{a_2 \oplus b_2}) . (\overline{a_1 \oplus b_1}) . (\overline{a_0}\, b_0)$$

## 2- Decoders

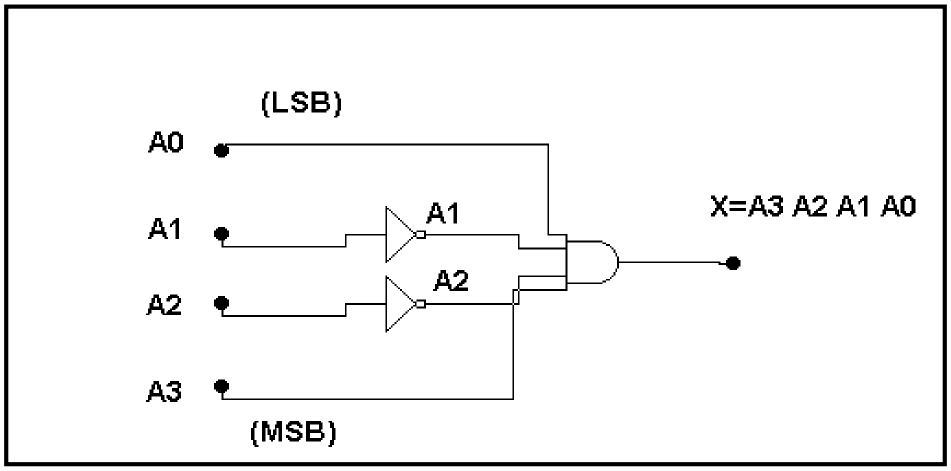
A decoder is a digital circuit that detects the presence of a specified combination of bits (code)on its inputs and indicates the presence of that code by a specified output level .

***In its general form :*** a decoder has **n** input lines to handle n bits and from one to **$2^n$** output lines to indicate the presence of one or more n-bit combinations.

## ♦ The Basic Binary Decoder

Suppose you need to determine when a binary 1001 occurs on the inputs of a digital circuit. An AND gate can be used as the basic decoding element because it produces a HIGH or (1) output only when all of its inputs are HIGH (1).

Therefore , you must make sure that all of the inputs to the AND gate are HIGH when the binary number 1001 occurs; this can be done by inverting the two middle nits (the 0s), as shown in the next figure .



you should verify that the output is (0) except when A0=1 ,A1=0, A2=0, AND A3=1 are applied to the inputs. A0 is the LSB and A3 is the MSB.

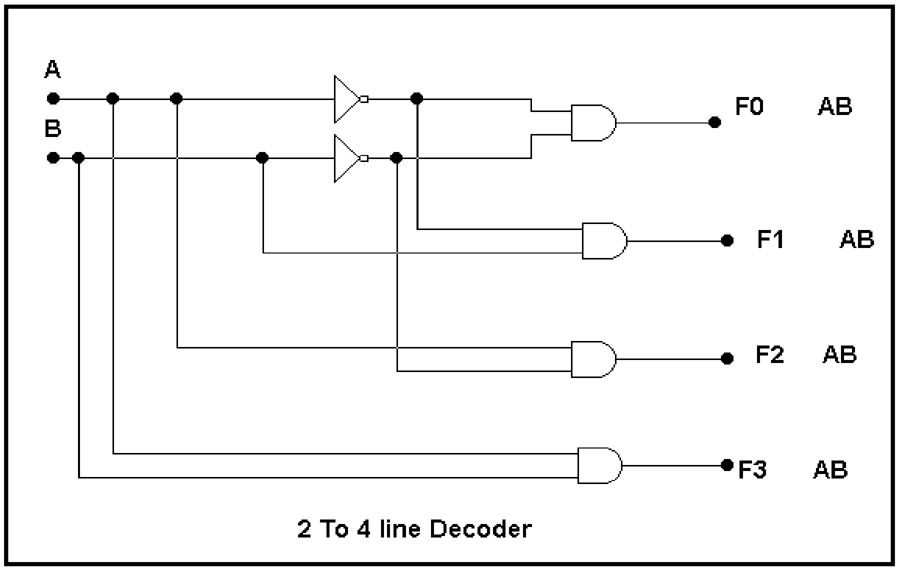
**Note:** If a NAND gate is used in place of the AND gate in pervious figure , a LOW output will indicate the presence of the proper binary code, which is 1001 in this case.

**Note :**Then the decoders presented here are called **n to m line decoders,** where m <= $\mathbf{2^n}$ . the purpose is to generate the $\mathbf{2^n}$ ( or fewer) minterms of n input variable. The name decoder is also used in conjunction with other code converters such as **BCD-to-Seven segment decoder**.

## 2 to 4 line decoder

We can design 2 to 4 line decoder by using the following truth table:

| Inputs | | Outputs | | | |
|---|---|---|---|---|---|
| A | B | F0 | F1 | F2 | F3 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |



## 3 to 8 line decoder

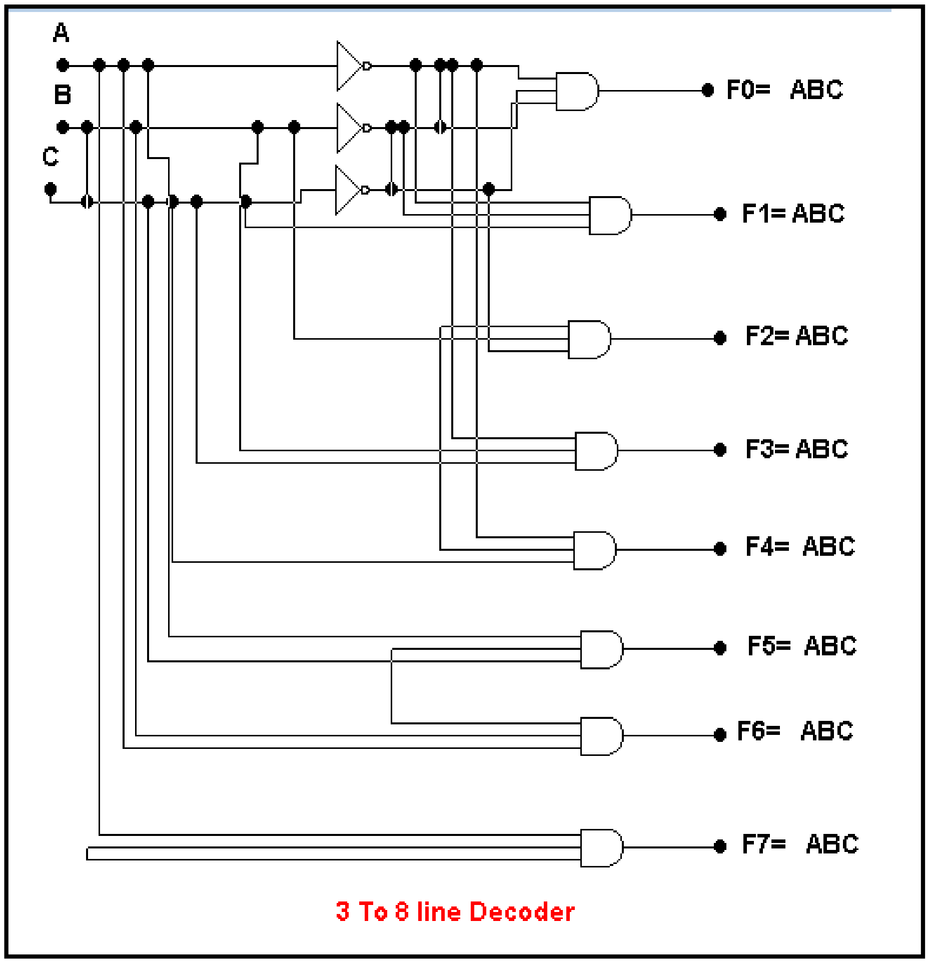
We can design 3 to 8 line decoder by using the following truth table:

| A | B | C | F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 |  | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

The three inputs are decoded into eight outputs each representing one of the mineterms of the three input variables.

The  three inputs are decoded into eight outputs each representing one of the mineterms of the three input variables.

The three inverters provides the complement of the inputs, and each of the eight AND gates generates one of the mineterms. A particular application of this decoder is Binary –To -Octal conversion.



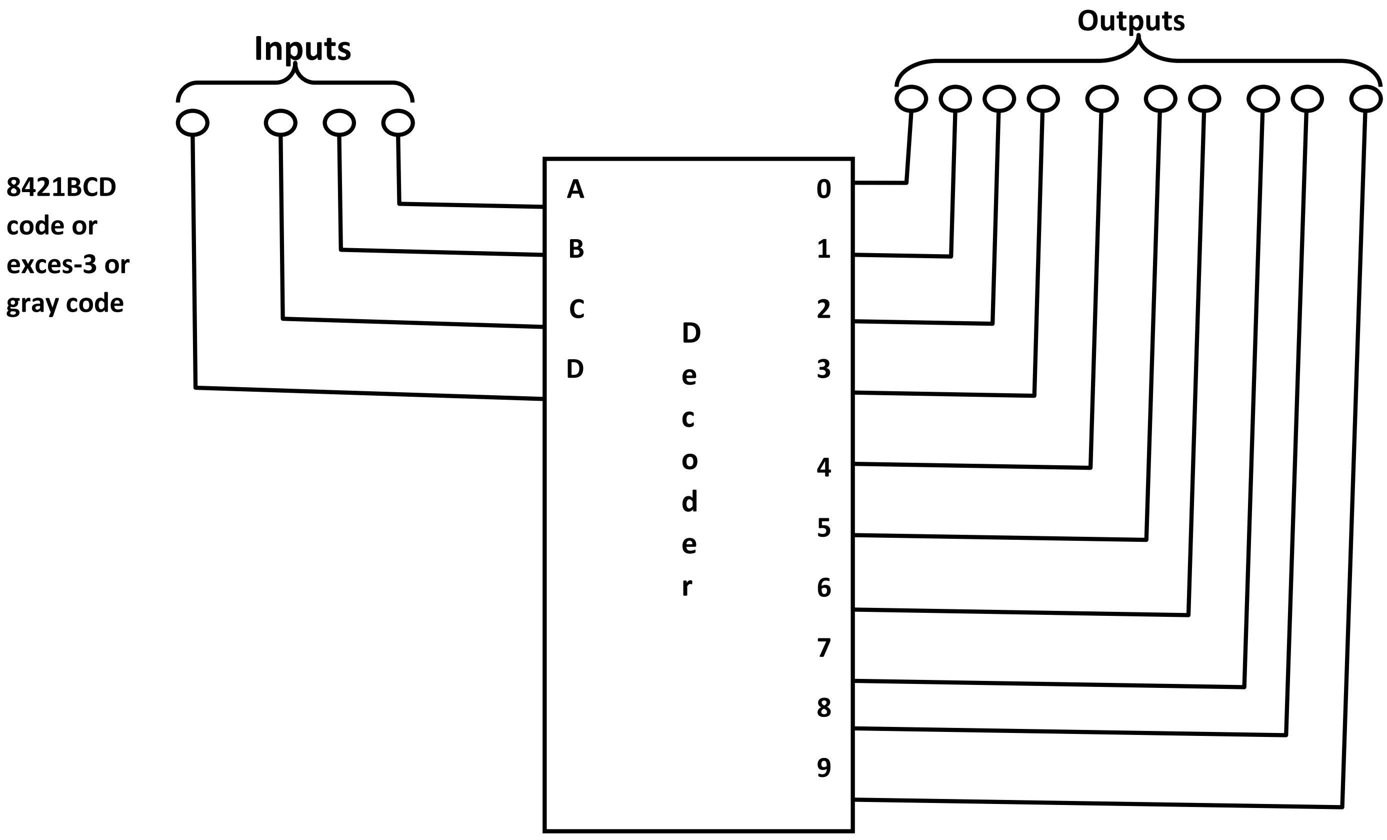
The input variables represent a binary number, and the outputs represent the eight digits in the octal number system.

However , a (3 –to - 8) line decoder can be used for decoding any **3-bit code** to provide **eight** output, one for each element of the code.

The operation of the decoder may be clarified by the truth table. For each possible input combination there are seven outputs that are equal to **(0)** and only one that is equal to **(1)** .

The output whose value is equal to **(1)** represents the minterms equivalent of binary number a variable in the input lines.

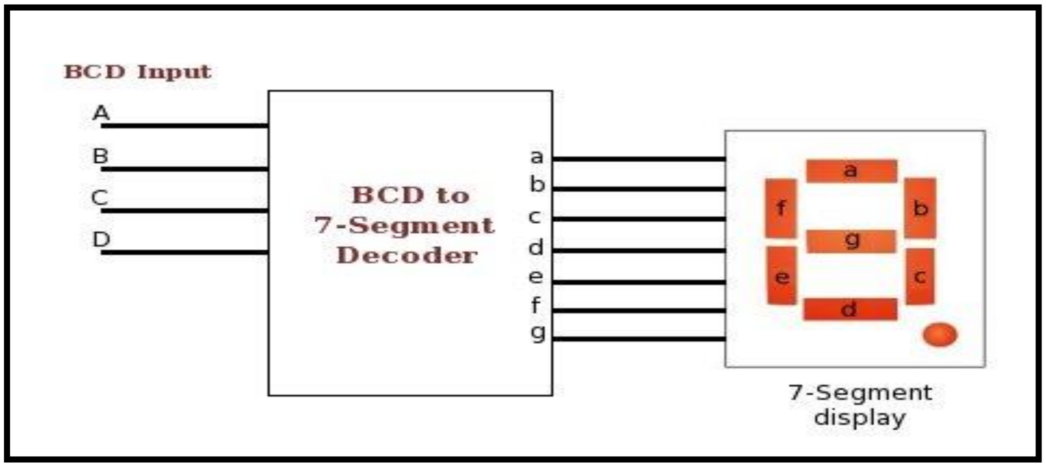
**Notes :**Decoders as we say came in several varieties in typical decoder , the inputs may be: 8421 BCD , axcess-3 , or gray code .



**A typical decoder block diagram**

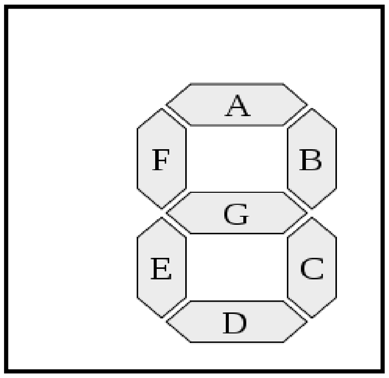
## 💧 BCD-to-Seven segment decoder

The decoders are translating the (8421 BCD) code to ( a seven segement display)

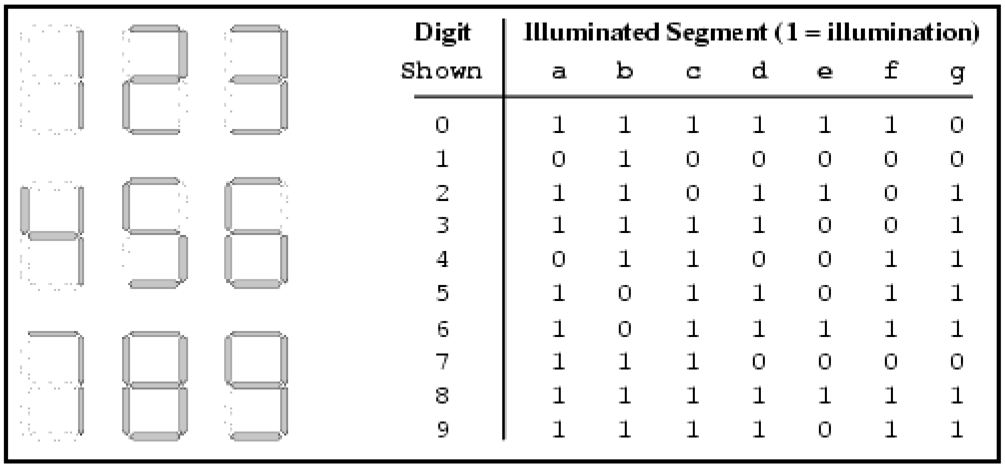


Suppose you wanted the decimal (4) to light on the display board, the BCD umber (0100) at the input of the BCE-to-Seven-segment decoder.

The Decoder activates outputs(a ,c ,d , f and g)  to light segments.



**Segment identification**



| Digit Shown | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 3 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 4 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 5 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

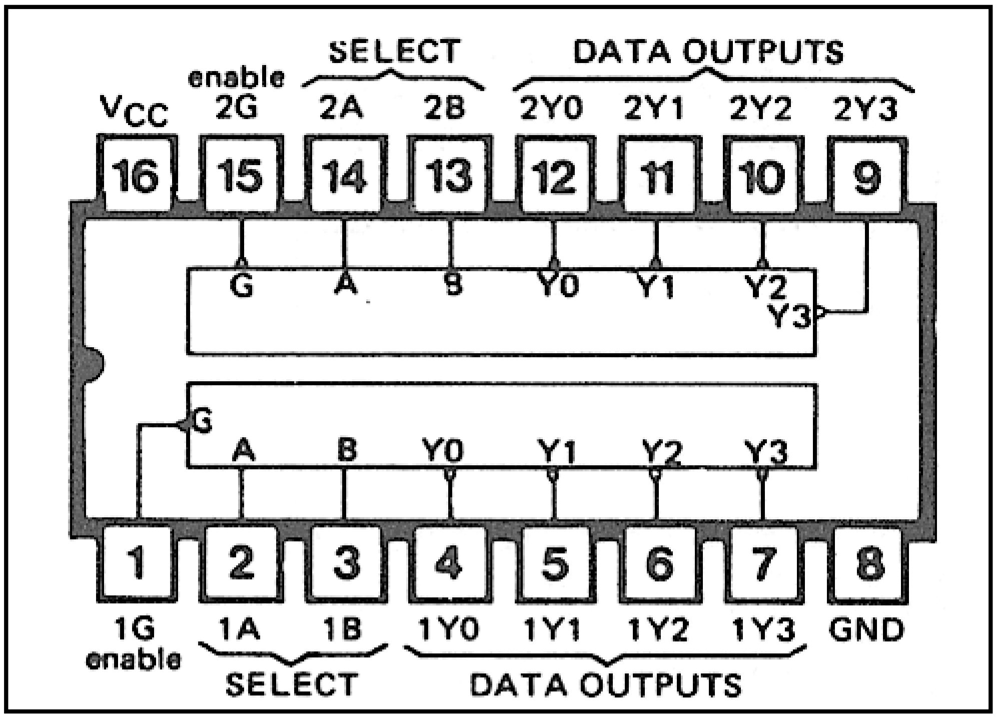
Illuminated Segment (1 = illumination)

Decimal  numbers on typical seven-segment display

| Decimal | Inputs | | | | Outputs | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | D | C | B | A | a | b | c | d | E | f | g |
| **0** | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| **1** | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| **2** | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| **3** | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| **4** | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| **5** | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| **6** | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| **7** | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| **8** | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **9** | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

**Truth Table of Typical BCE-Seven-Segment Decoder**

## ♦ The 74139 decoder



## 2-line-to 4 Decoder IC. No(74139)

This type of decoders (which we used in laboratory) with an enable input to control the circuit operation, as we see in previous figure . the decoder is enabled when (enable )is equal to (0).

The circuit operates with complement outputs and complement input , when (E) is equal to (0) only one output can be equal to (0) at any given time all other outputs are equal to (1) .

The output whose value is equal to (0) represents the minterm selected by inputs (A) and (B).

The circuit is disabled when (E) is equal to (1). When the circuit is disabled ,none of the outputs are equal to (0) and none of the minterms.

## Example : Design Half-adder using (74139) (2-to-4) Decoder .

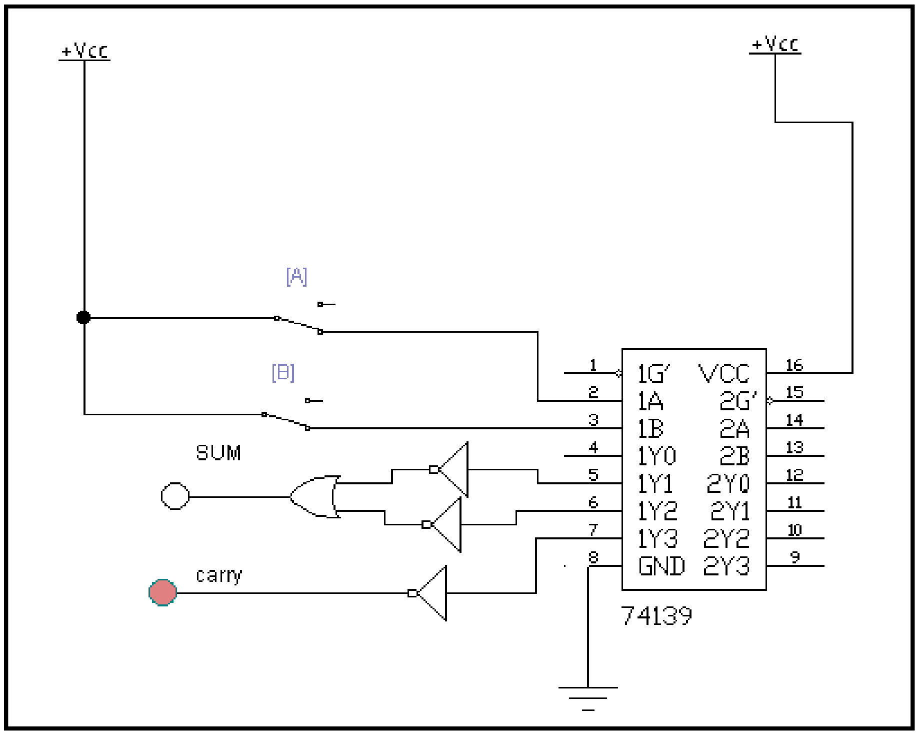
As indicated by the truth table , only two output can be equal to (1) at (S) output , while (C0) has only one output equal to (0).

| **Input** | | **Outputs** | |
|---|---|---|---|
| **A** | **B** | $\Sigma(s)$ | $C_o$ |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| **Binary digits to be added** | | **Sum** | **Carry out** |

|  | XOR | AND |
|---|---|---|

-Truth table of half-adder

The (74139) decoder provide a complement outputs as is shown in IC figure , therefore we must connect **not gate** for each active output.


+Vcc
+Vcc
[A]
[B]
SUM
carry
1G'
1A
1B
1Y0
1Y1
1Y2
1Y3
GND
VCC
2G'
2A
2B
2Y0
2Y1
2Y2
2Y3
74139