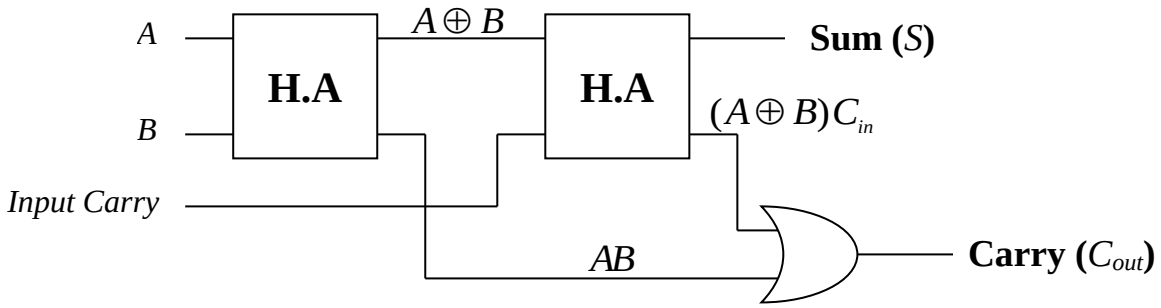
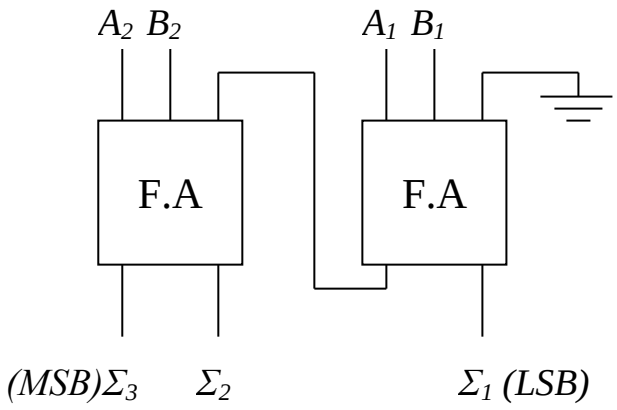
***Example:*** Arrange two half-adders to form full-adder.




## Parallel Binary Adders:

Adders that are available in integrated circuits form are parallel binary adders. As you saw in the previous section, a single full-adder is capable of adding two 1-bit numbers and an input carry. To add binary numbers with more than one bit, additional full-adders must be employed. To implement the addition of binary numbers, a full-adder is required for each bit in the numbers. So, for 2-bit numbers, two adders are needed; for 4-bit numbers, four adders are used; and so on.

$$\begin{array}{r} A_2A_1 \\ +\ B_2B_1 \\ \hline \Sigma_3\ \Sigma_2\ \Sigma_1 \end{array}$$



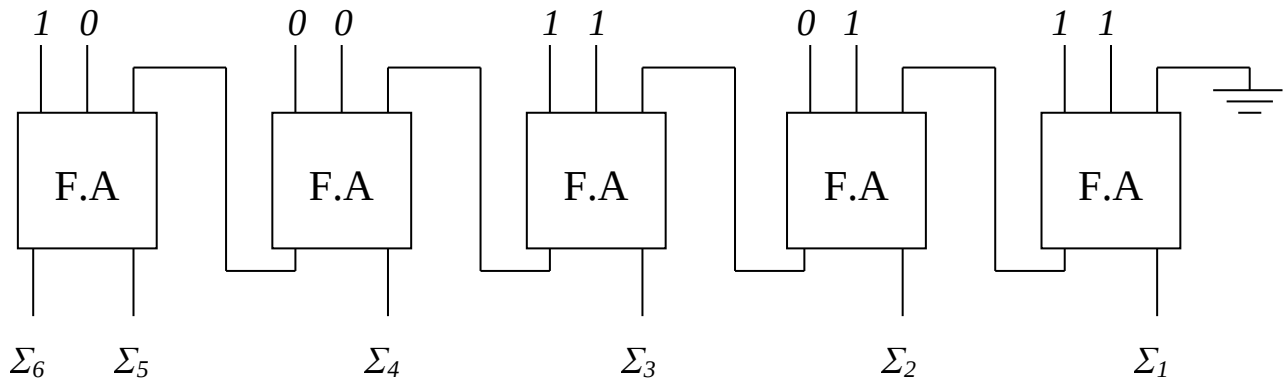
## Review Questions:

21. Show how a H.A can be fabricated using only NOR gates.
22. Design a 4-bit parallel adder.
23. Draw a circuit for 2-bit parallel adder.
24. Draw a block diagram for 8-bit parallel adder.

25. Determine an alternative method for implementing the full-adder.

*Hint: write the expressions of the circuit and simplify using Karnaugh Map. Then implement using AND-OR gates.*

26. For the parallel adder in figure below, determine the complete sum by analysis of the logical operation of the circuit. Verify your results by long hand addition of the two input number.



## Half- and Full- Subtractors:

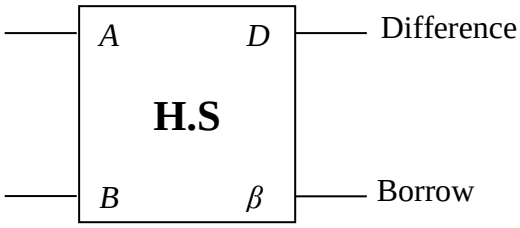
Instead of using complements to subtract, circuits can subtract binary numbers directly.

### 5- *The Half-Subtractor (H.S):*

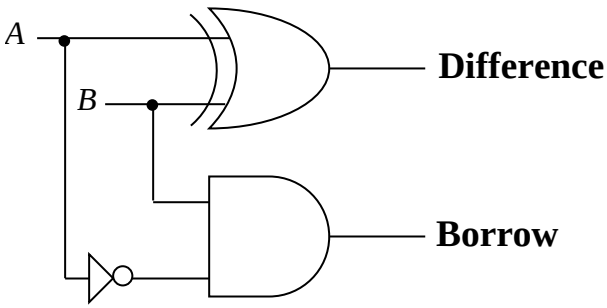
Recall the basic rules for binary subtraction as stated in the previous lectures:

$0-0=0$ *with a borrow of* 0
$0-1=1$ *with a borrow of* 1
$1-0=1$ *with a borrow of* 0
$1-1=0$ *with a borrow of* 0

These operations are performed by a logic circuit called a half-subtractor. The half-subtractor accepts two binary digits on its inputs and produces two binary digits on its outputs, a **Difference** bit and a **Borrow** bit.




| *A* | *B* | *Borrow* ( $\beta$ ) | *Difference (D)* |
|---|---|---|---|
| **0** | **0** | **0** | **0** |
| **0** | **1** | **1** | **1** |
| **1** | **0** | **0** | **1** |
| **1** | **1** | **0** | **0** |

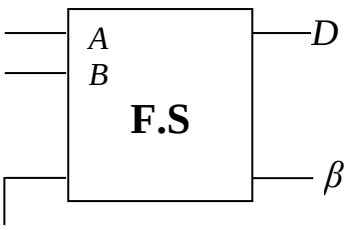



$$Difference = D = A \oplus B$$
$$Borrow = \beta = \overline{A}B$$

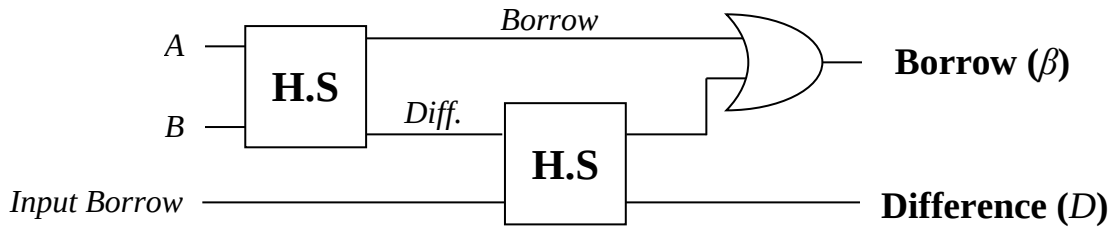
### 6- *The Full- Subtractor (F.S):*

The half-subtractor handles only two bits at a time and can be used for the least significant column of a subtraction problem. To take care of a higher-order column, we need a full-subtractor.

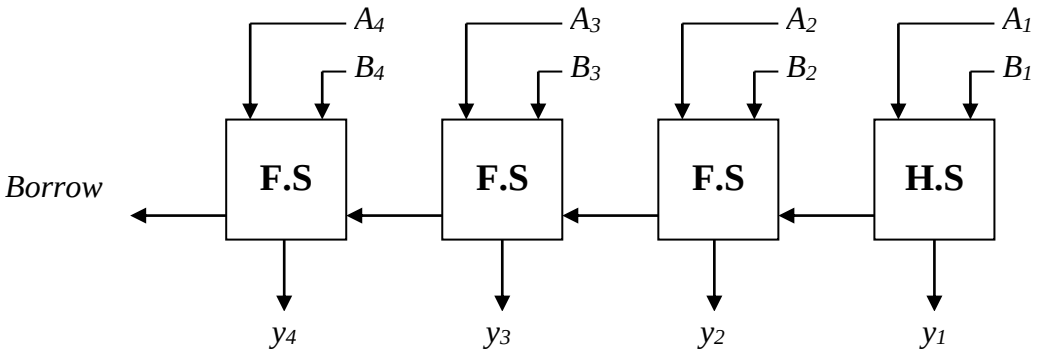
The full-subtractor uses two half-subtractors And an OR gate as shown in figure below:




Input Borrow

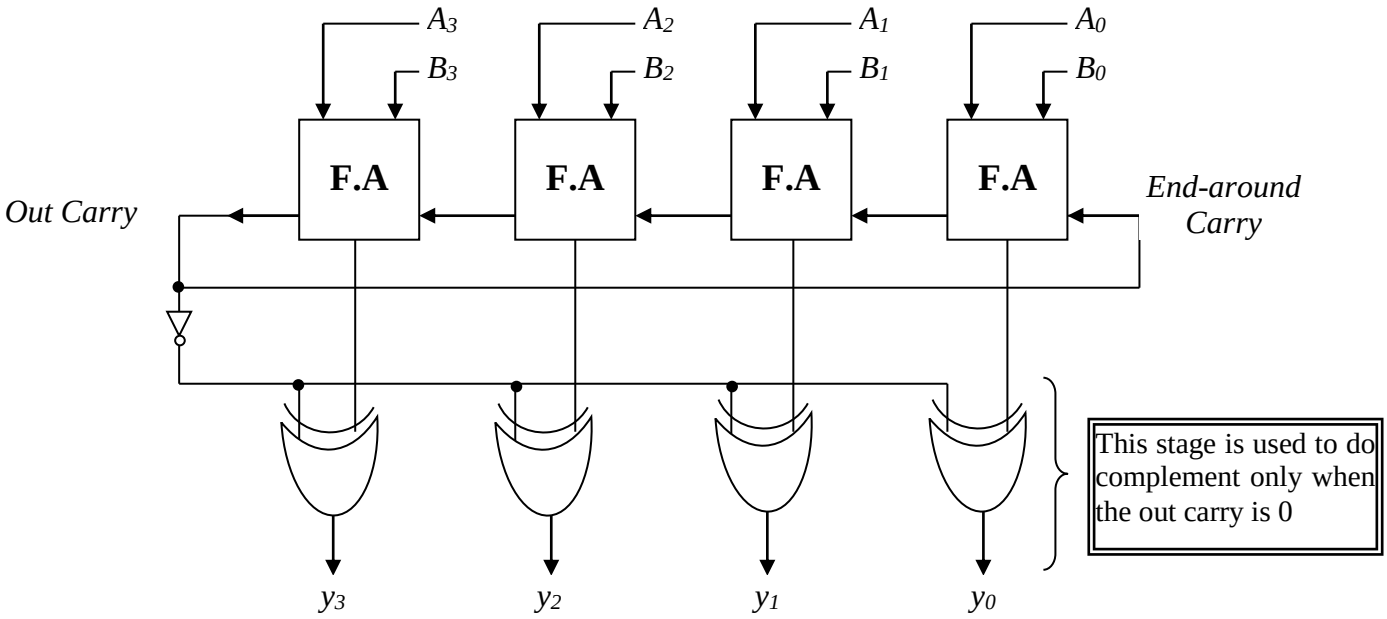



Half- and Full- subtractors are analogous to half- and full- adders; by cascading half- and full- subtractors as shown in figure below, we have a system that directly subtract $B_4B_3B_2B_1$ *From* $A_4A_3A_2A_1$.

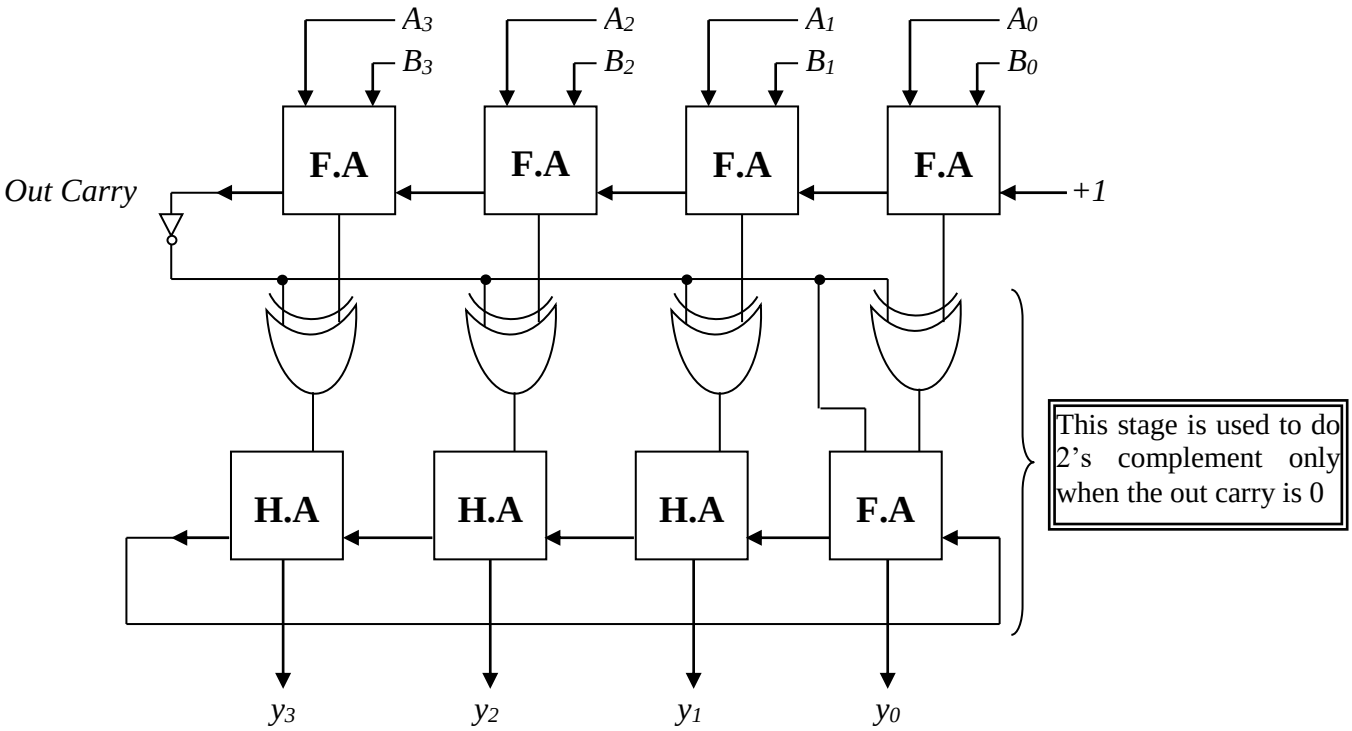



## 1's Complement Subtractors:

The following figure shows a circuit that subtracts $B_3B_2B_1B_0$ *from* $A_3A_2A_1A_0$.The first four inverters complement each *B* bit to get $\overline{B}_3\,\overline{B}_2\,\overline{B}_1\,\overline{B}_0$, the 1's complement of $B_3B_2B_1B_0$. The full-adders add *A* and *B* and the end-around carry.

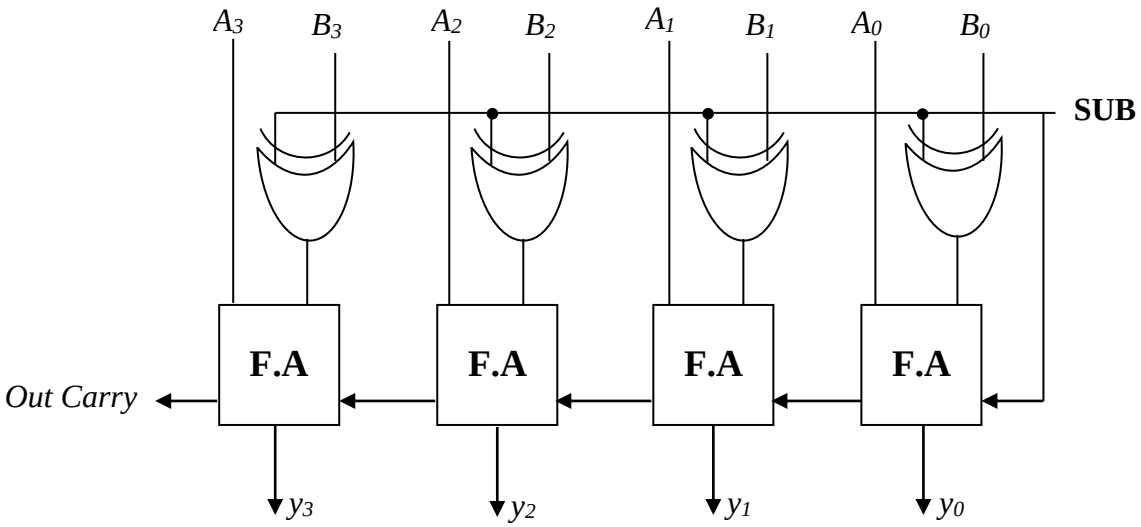


## 2's Complement Subtractors:



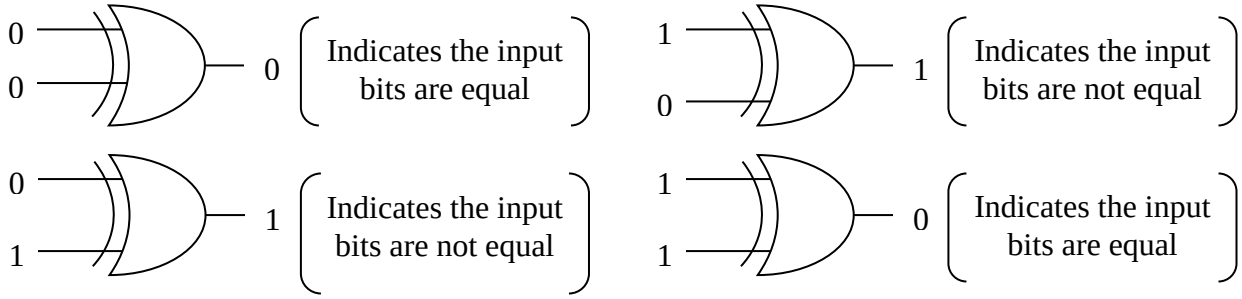

## 2's Complement Adders/ Subtractors:

The following figure shows an adder/subtractor based on the 2's complement. When SUB is **Low**, the *B* bits pass through the controlled inverter to the full-adder. Therefore; the full adder produces the sum of *A*&*B*. When SUB is **High**, the Bits are inverted before reaching the full-adder. Also, a High SUB adds a 1 to the first full-adders. This addition of 1 forms the 2's complement of *B*. therefore; the output of the full-adders is the difference of *A*&*B*.

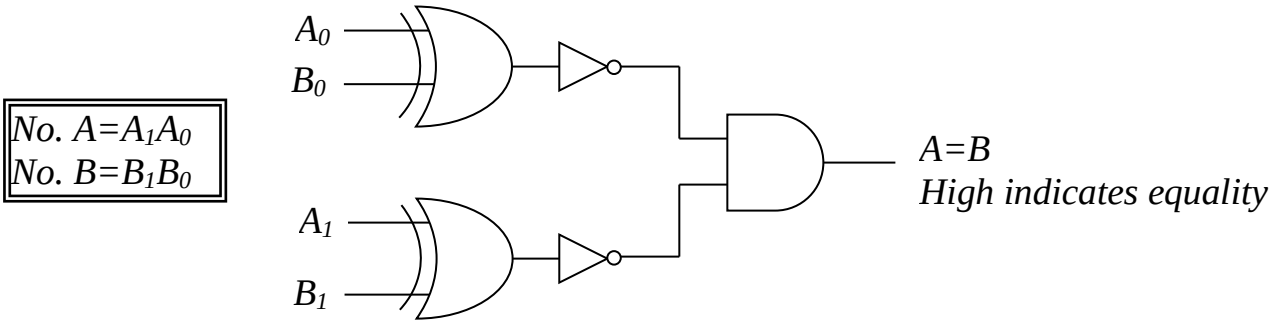


## Comparators:

The basic function of a comparator is to compare the magnitudes of two quantities to determine the relationship of those quantities. In its simplest form, a comparator circuit determines whether two numbers are equal. The exclusive-OR gate can be used as a basic comparator because its output is a 1 if two input bits are not equal and a 0 if the input bits are equal.

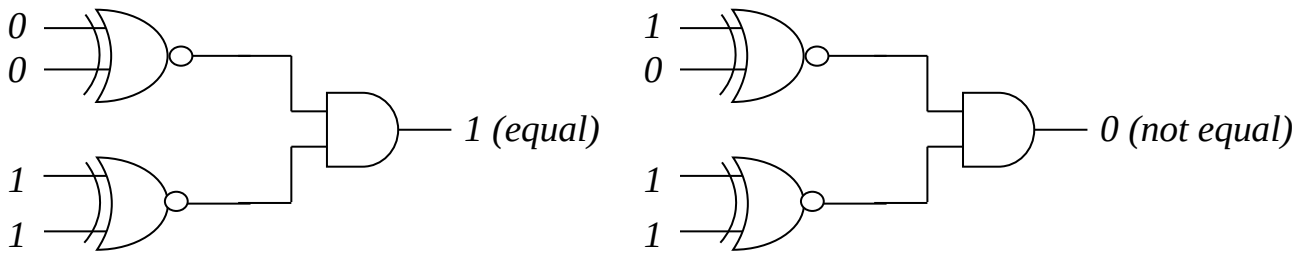



In order to compare binary numbers containing two bits each, an additional exclusive-OR gate is necessary.

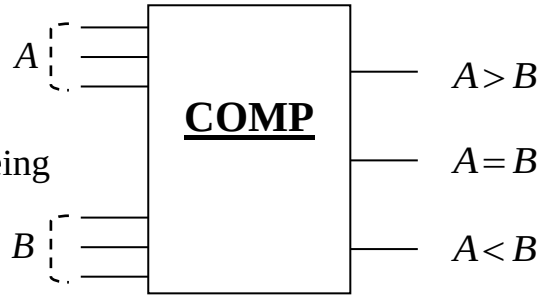



***Example:*** Apply each of the following sets of binary numbers to the comparator inputs and determine the output by following the logic levels through the circuits.

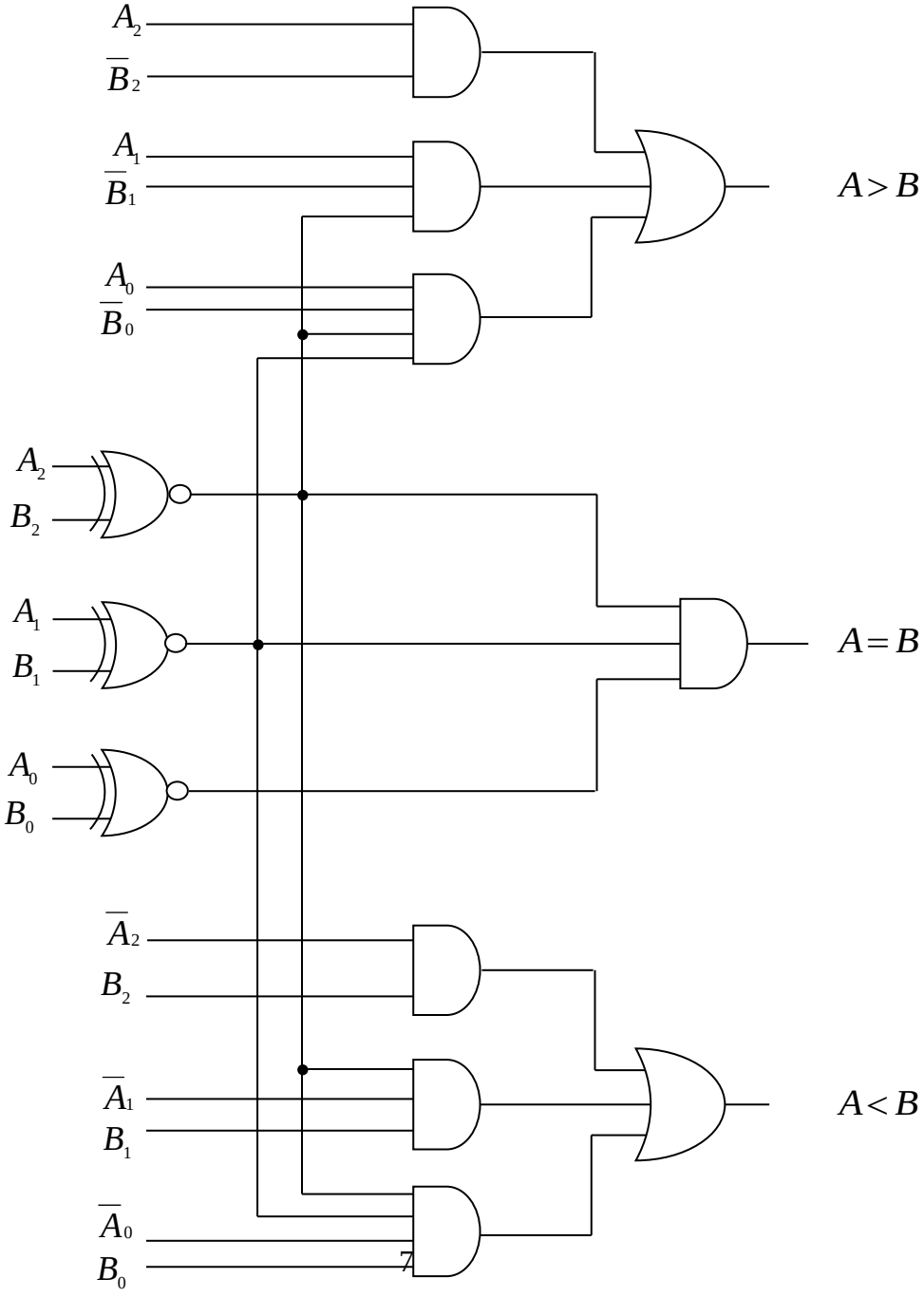
(a) 10 and 10 (b) 11 and 10



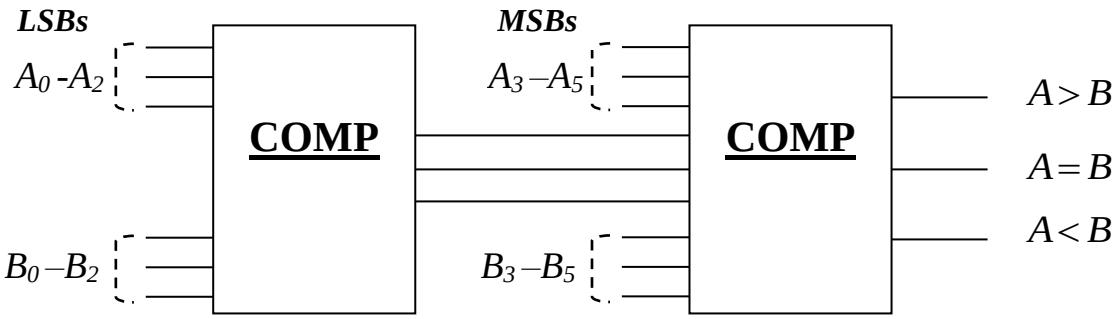
## Integrated Circuit Comparators:




In addition to the equality output, many integrated circuit comparators provide additional outputs that indicate which of the two numbers being compared is larger. That is, there is an output that indicates when number $A$ is greater than number $B$ ($A>B$) and an output that indicates when number $A$ is less than $B$ ($A<B$), as shown in the logic symbol for a 3-bits comparator in above figure:

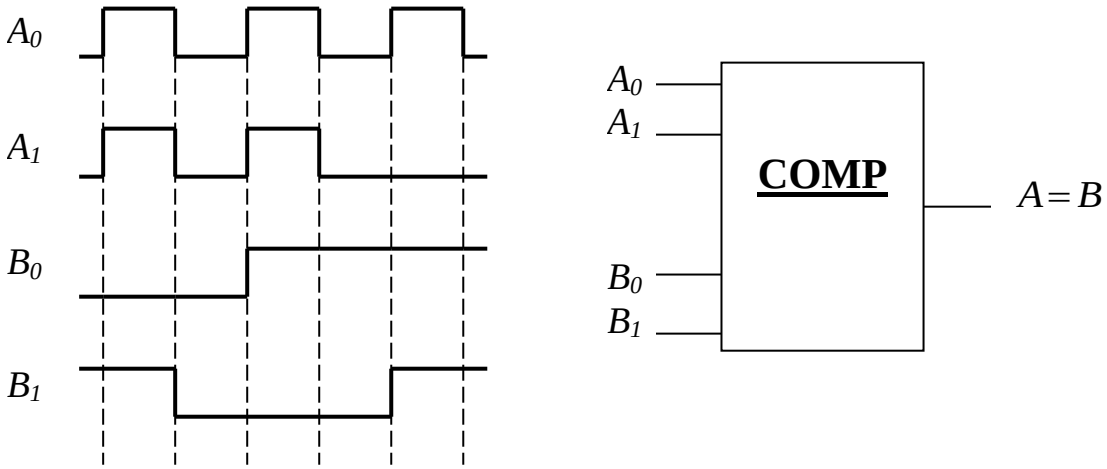


***Example:*** Use 3-bit integrated comparators to compare the magnitude of two 6-bit binary numbers. Show the comparators with proper connections.




## *Review Questions:*

27. Design a full-subtractor circuit.
28. Design a subtractor to subtract two numbers of 4-bit using F.A.
29. Draw circuit for equality comparison of two 4-bit numbers.
30. Draw a general circuit for a 4-bit comparator. Analyze the circuit for $A$=1010 and $B$=1100.
31. The waveforms in figure below are applied to the comparator as shown. Determine the output ($A$=$B$) waveform.



## BCD Adder:

The addition of BCD number is summarized in the following steps:

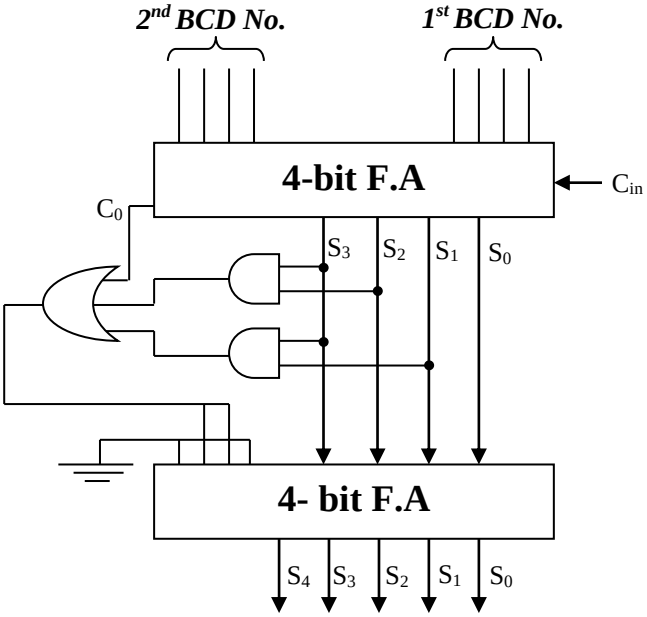
1. Add the BCD code groups for each decimal digit position, use ordinary binary addition.
2. For those positions where the sum is 9 or less, the sum is in proper BCD form and no correction is needed.
3. When the sum of two digits is greater than 9, a correction of 0110 (6) should be added to that sum to produce the proper BCD results. This will produce a carry to be added to the next decimal position.

A **BCD** adder cct. must be able to do the following:-

1. Add two 4-bit BCD code groups, using straight binary addition.
2. Determine if the sum of this addition is greater than 1001 (9), if it is, add 0110 (6) to this sum and generate a carry to the next decimal position.

| | |
|---|---|
| 1000 | 8 |
| + 0101 | 5 |
| 1101 | Binary equivalent of 13 |
| + 0110 | Add 6 to return to BCD |
| 0001 0011 | BCD equivalent of 13 |

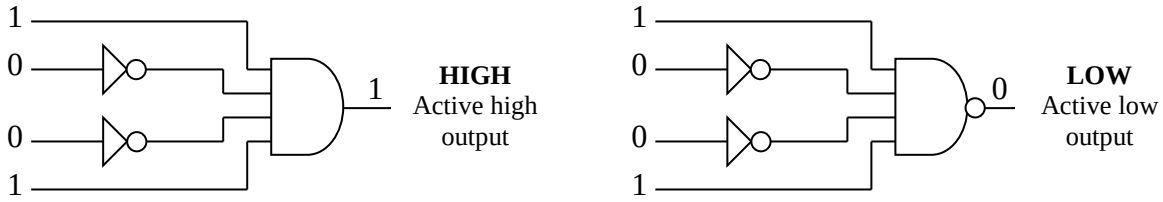
| | |
|---|---|
| 1001 | 9 |
| + 0011 | 3 |
| 1100 | Binary equivalent of 12 |
| + 0110 | Add 6 to return to BCD |
| 0001 0010 | BCD equivalent of 12 |



## Decoders:

The basic function of a decoder is to detect the presence of a specified combination of bits (code) on its inputs and to indicate the presence of that code by a specified output level. In its general form, a decoder has *n* lines to handle *n* bits and from one to $2^n$ output lines to indicate the presence of one or more *n*-bit combination.

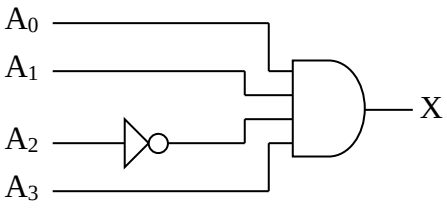
***Example:*** A decoder for binary number 1001.




***Example:*** Determine the logic required to decode the binary number 1011 by producing a **HIGH** level on the output.

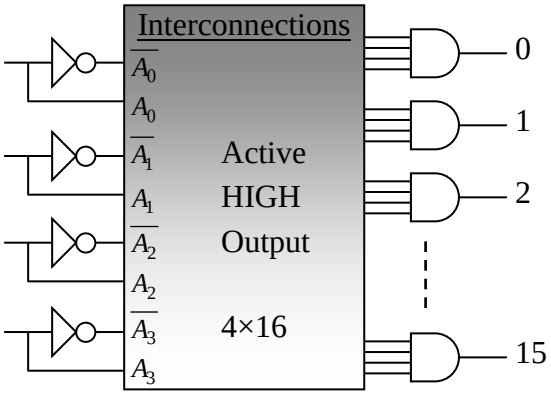
The decoding function can be formed by complementing only the variables that appear as 0 in the binary number, as follows:

$$X = A_3 \overline{A}_2 A_1 A_0 \qquad (1011)$$




## Four bit binary Decoders:

In order to decode all possible combinations of four bits, sixteen decoding gates are required ($2^4$=16). This type of decoder is commonly called a 4-line-to-16-line decoder because there are four inputs and sixteen outputs or a 1-of-16 decoder because for any given code on the inputs, one of the sixteen outputs is activated.



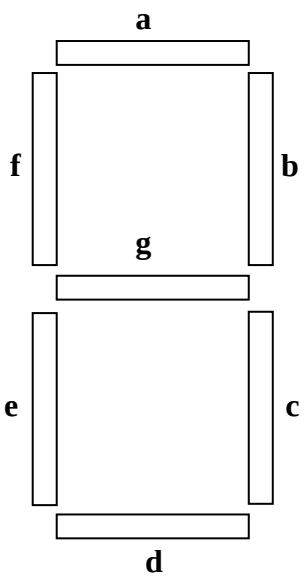

**Pebble is added for each AND gate to produce active low output**

| Decimal | Binary Input | | | | Output | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $A_3$ | $A_2$ | $A_1$ | $A_0$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| **0** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **4** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **5** | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **6** | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **7** | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **8** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **9** | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **10** | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **11** | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **12** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **13** | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **14** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| **15** | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

## BCD to Seven Segment Decoder:

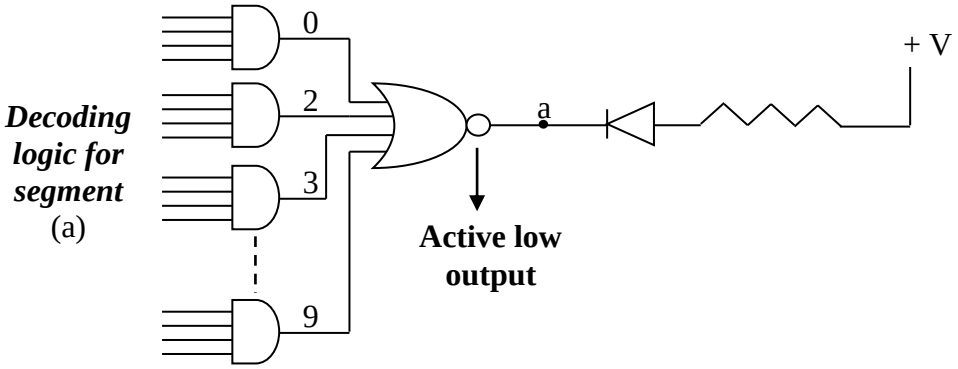
This decoder accepts BCD code on its inputs and provides outputs to energize 7-segment display device.

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



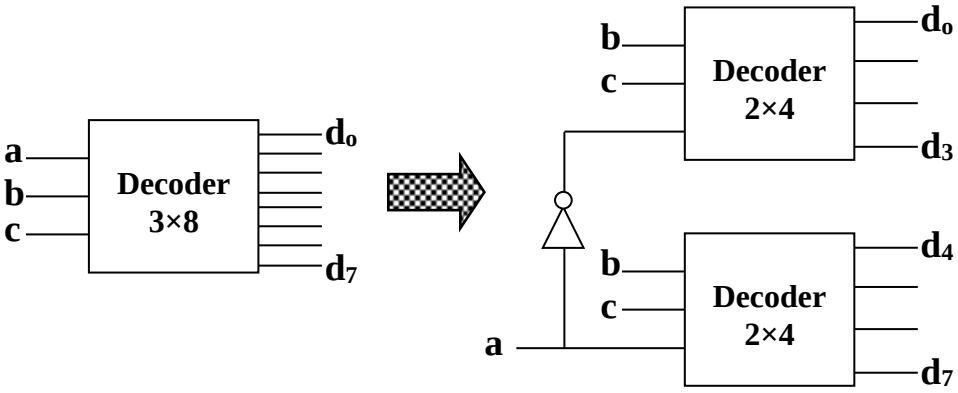
**a** appears in 0,2,3,5,7,8,9, the Boolean expression for LED a is:

$$a = \overline{D}\overline{C}\overline{B}\overline{A} + \overline{D}\overline{C}\overline{B}A + \overline{D}\overline{C}B\overline{A} + \overline{D}C\overline{B}\overline{A} + \overline{D}CB\overline{A} + D\overline{C}\overline{B}\overline{A} + D\overline{C}\overline{B}A$$

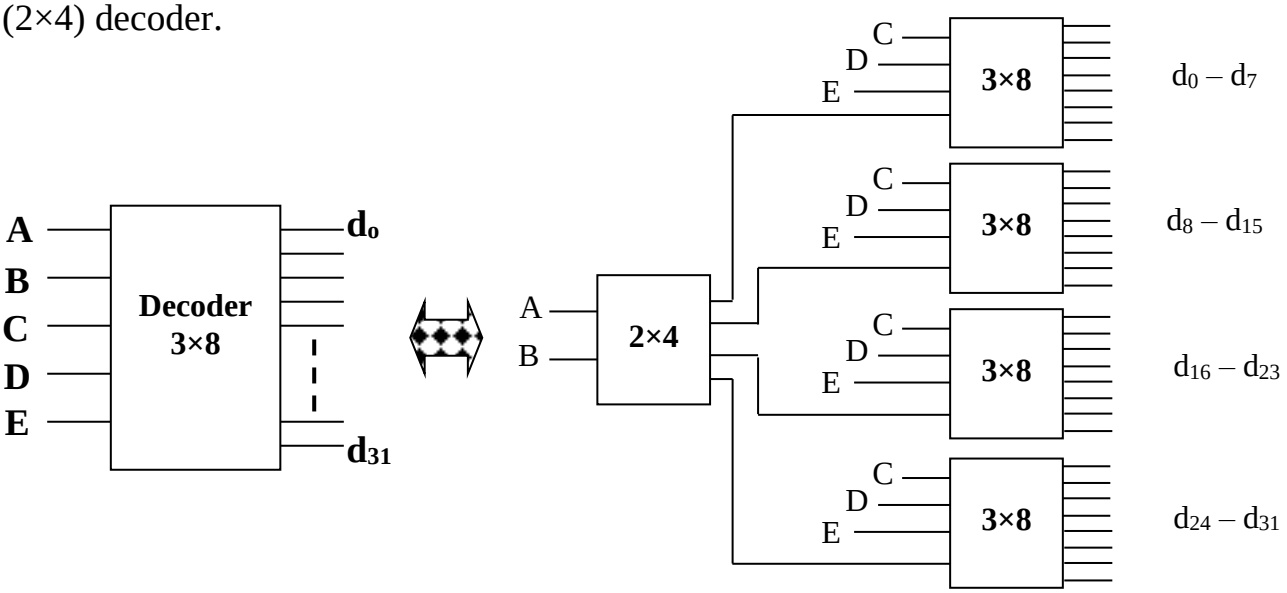



***Example:*** Design (3×8) decoder using (2×4) decoders and NOT gate .

| a | b | c | o/p |
|---|---|---|---|
| 0 | 0 | 0 | $d_0$ |
| 0 | 0 | 1 | $d_1$ |
| 0 | 1 | 0 | $d_2$ |
| 0 | 1 | 1 | $d_3$ |
| 1 | 0 | 0 | $d_4$ |
| 1 | 0 | 1 | $d_5$ |
| 1 | 1 | 0 | $d_6$ |
| 1 | 1 | 1 | $d_7$ |



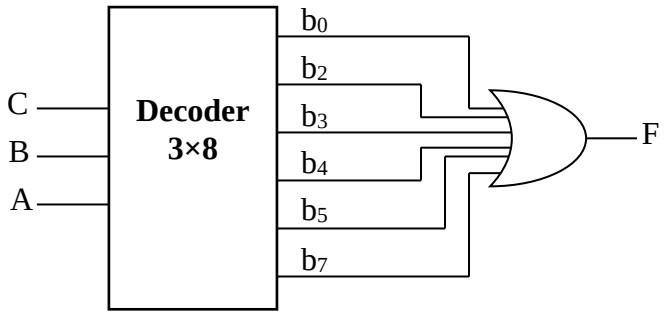

***Example:*** Design a 5-input to 32 decoder using 4 of (3×8) decoder and one of (2×4) decoder.



***Example:*** Design the following Boolean function with a (3×8) decoder and OR gate: $F = \bar{A}B + AC + BC + B\bar{C}$

$$F = \bar{A}BC + \bar{A}B\bar{C} + ABC + A\bar{B}C + ABC + \bar{A}BC + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C}$$
$$= \bar{A}BC + \bar{A}B\bar{C} + ABC + A\bar{B}C + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C}$$

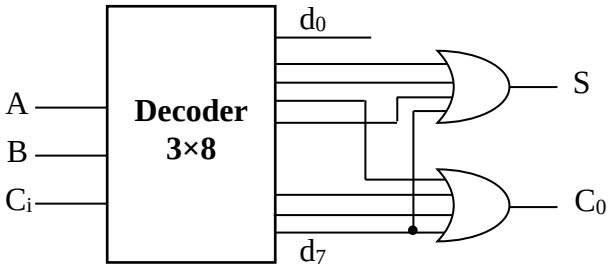
| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |




$$f(A,B,C) = \sum_m (0,2,3,4,5,7)$$
$$f(A,B,C) = \Pi_m (1,6)$$

***Example:*** Design a full adder cct. using a decoder of (3×8) and two OR gates.

| $C_i$ | A | B | S | $C_o$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |




$$S = d_1 + d_2 + d_4 + d_7$$
$$C_o = d_3 + d_5 + d_6 + d_7$$

**Encoder:**

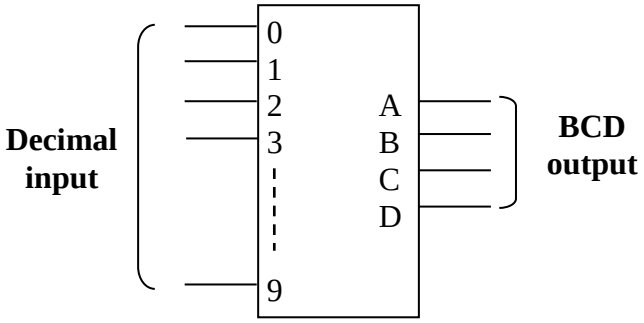
An encoder is a combinational logic circuit that essentially performs a "reverse" decoder function. An encoder accepts an active level on one of its inputs representing a digit such as a decimal or octal digit, and converts it to a coded output, such as BCD or binary. Encoders can also be desired to encode various

symbols and alphabetic characters. The process of converting from familiar symbols or numbers to a coded format is called encoding.

**Decimal to BCD Encoder:**

This type of encoder has ten inputs – one for each decimal digit – and four outputs corresponding to the BCD code as shown in figure below. This is a basic 10-line-to-4-line encoder.

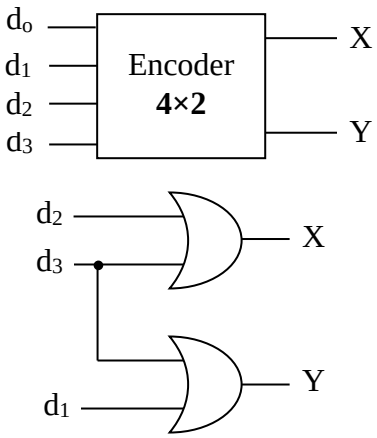
| Decimal | D | C | B | A |
|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 |
| **1** | 0 | 0 | 0 | 1 |
| **2** | 0 | 0 | 1 | 0 |
| **3** | 0 | 0 | 1 | 1 |
| **4** | 0 | 1 | 0 | 0 |
| **5** | 0 | 1 | 0 | 1 |
| **6** | 0 | 1 | 1 | 0 |
| **7** | 0 | 1 | 1 | 1 |
| **8** | 1 | 0 | 0 | 0 |
| **9** | 1 | 0 | 0 | 1 |




The expressions for output in term of decimal are:

$$D = 8 + 9$$
$$C = 4 + 5 + 6 + 7$$
$$B = 2 + 3 + 6 + 7$$
$$A = 1 + 3 + 5 + 7 + 9$$

***Example:*** (4×2) encoder: where 4= No. of input, 2= No. of output.





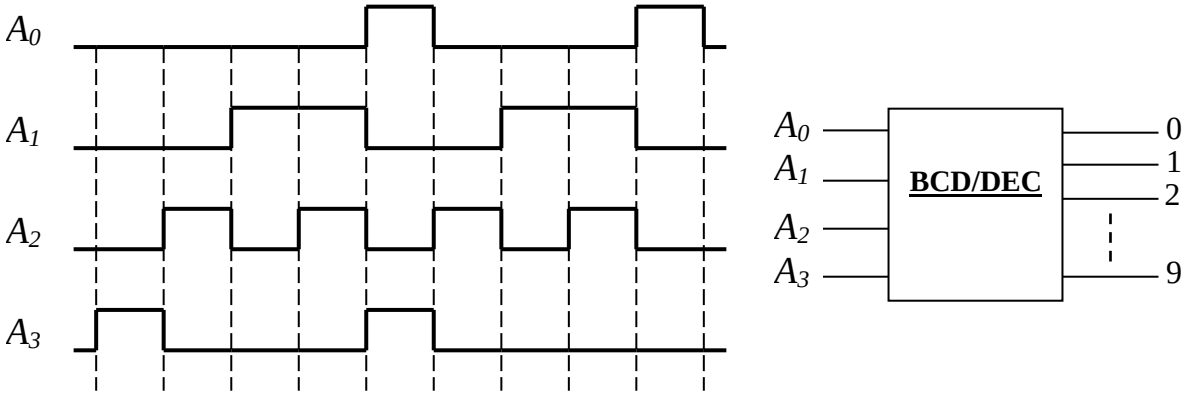


| $d_0$ | $d_1$ | $d_2$ | $d_3$ | X | Y |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |

$$X = d_2 + d_3$$
$$Y = d_1 + d_3$$

## ***Review Questions:***

32. Design a logic circuit to convert octal number to binary using (8×3) encoder.
33. BCD numbers are applied sequentially to the BCD-to-decimal decoder in figure below. Draw a timing diagram, showing each output in the proper relationship with others and with the inputs.




34. Convert decoder to H.A.
35. Convert decoder to H.S.
36. Convert decoder to comparator.
37. Convert decoder to: **AND, OR, NAND, NOR, NOT, EX-OR, EX-NOR** gates.