Chapter 4: Combinational circuits

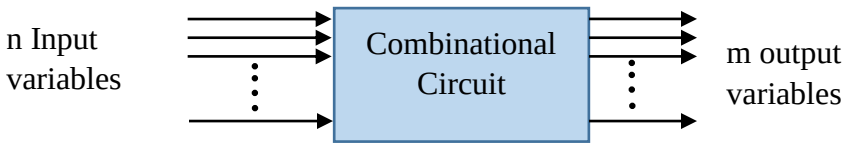
Digital circuits can be classified into two types:

→ Combinational Logic circuits and (Chapter 4)

→ Sequential Logic circuits (Chapter 5)

Combinational Logic Circuit

- Combination Logic Circuits are made up from basic gates (AND, OR, NOT) or universal gates (NAND, NOR) gates that are "combined" or connected together to produce more complicated switching circuits. These logic gates are the building blocks of combinational logic circuits. An example of a combinational circuit is a decoder, which converts the binary code data present at its input into a number of different output lines, one at a time producing an equivalent decimal code at its output.
- In these circuits "the outputs at any instant of time depends on the inputs present at that instant only."
- For examples of combinational digital circuits are Half adder, Full adder, Half subtractor, Full subtractor, Code converter, Decoder, Multiplexer, Demultiplexer, Encoder, ROM, etc.



The "n" binary input variable come from an external source, the "m" binary output variable go to an external destination and in between there is an interconnection of logic gates.

A combinational circuit can be described by a truth table showing the binary relationship between the "n" input and the "m" output variables.

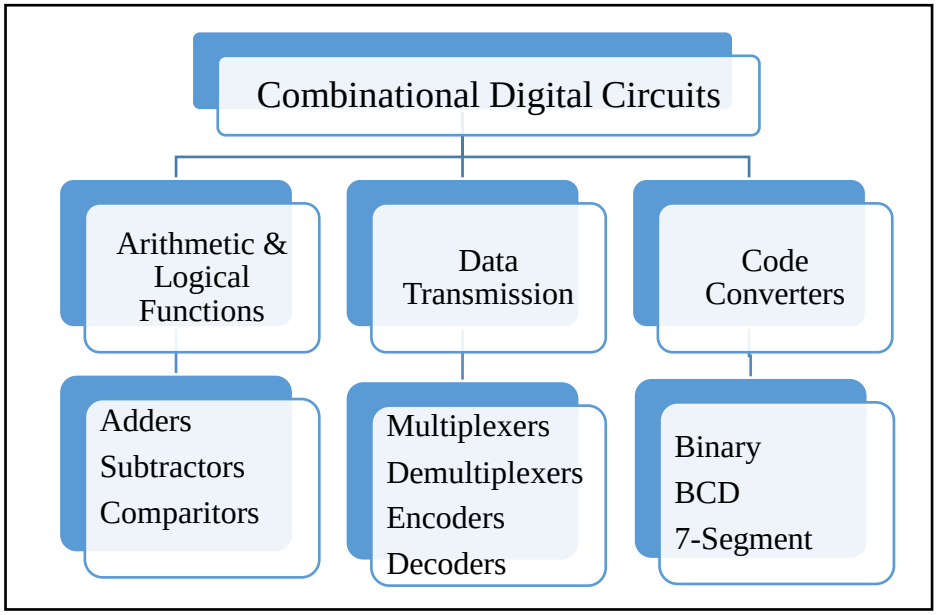


Fig 1.1: Combinational Digital Circuit.

4.1 Half-adder

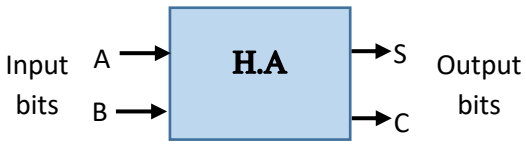
The most basic digital arithmetic circuit is the addition of two binary digits. A combinational circuit that performs the arithmetic addition of two bit is called a half-adder. The operations are performed by a logic circuit called a half-adder. The half-adder accepts two binary digits on its inputs and produces two binary digits on its outputs, a sum bit and a carry bit. A half-adder is represented by the logic symbol in Fig. (1.2).

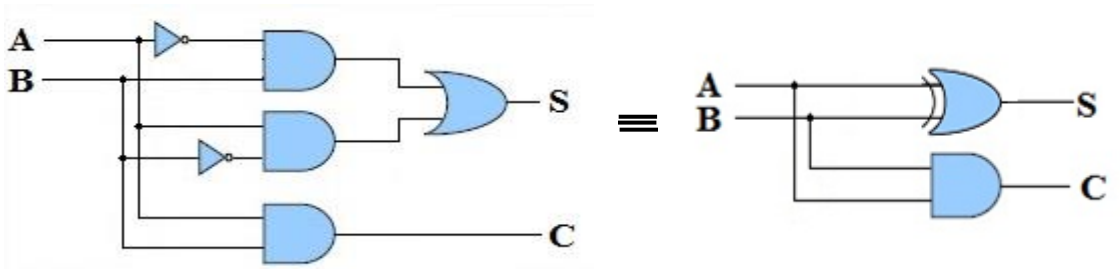
Table 1: Half-adder truth table.

Input		Output	
A	B	Carry (C)	Sum (S)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$\text{Sum} = \bar{A}B + A\bar{B} = A \oplus B$

$\text{Carry} = AB$





(A)

$S = \bar{A}B + A\bar{B}$

$C = AB$

(B)

$S = A \oplus B$

$C = AB$

Fig 1.2: Half-adder circuit.

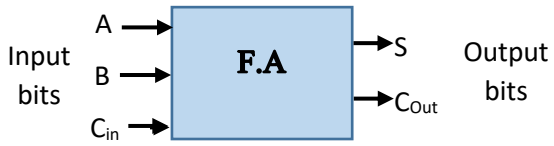
4.2 Full Adder

The second category of adder is the full-adder. The full-adder accepts two input bits and an input carry and generates a sum output and an output carry.

The basic difference between a full-adder and a half-adder is that the full-adder accepts an input carry. A logic symbol for a full-adder is shown in Fig. (1.3), and the truth table in Table 2 shows the operation of a full-adder.

Table 2: Full-adder truth table.

Input			Output	
A	B	C_{in}	Carry (C)	Sum (S)
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$$S = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$= C(\bar{A}\bar{B} + AB) + \bar{C}(\bar{A}B + A\bar{B})$$

$$= C(\overline{A \oplus B}) + \bar{C}(A \oplus B)$$

$$S = A \oplus B \oplus C$$

$$C_{out} = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

$$= C(\bar{A}B + A\bar{B}) + AB(\bar{C} + C)$$

$$C_{out} = AB + C(A \oplus B)$$

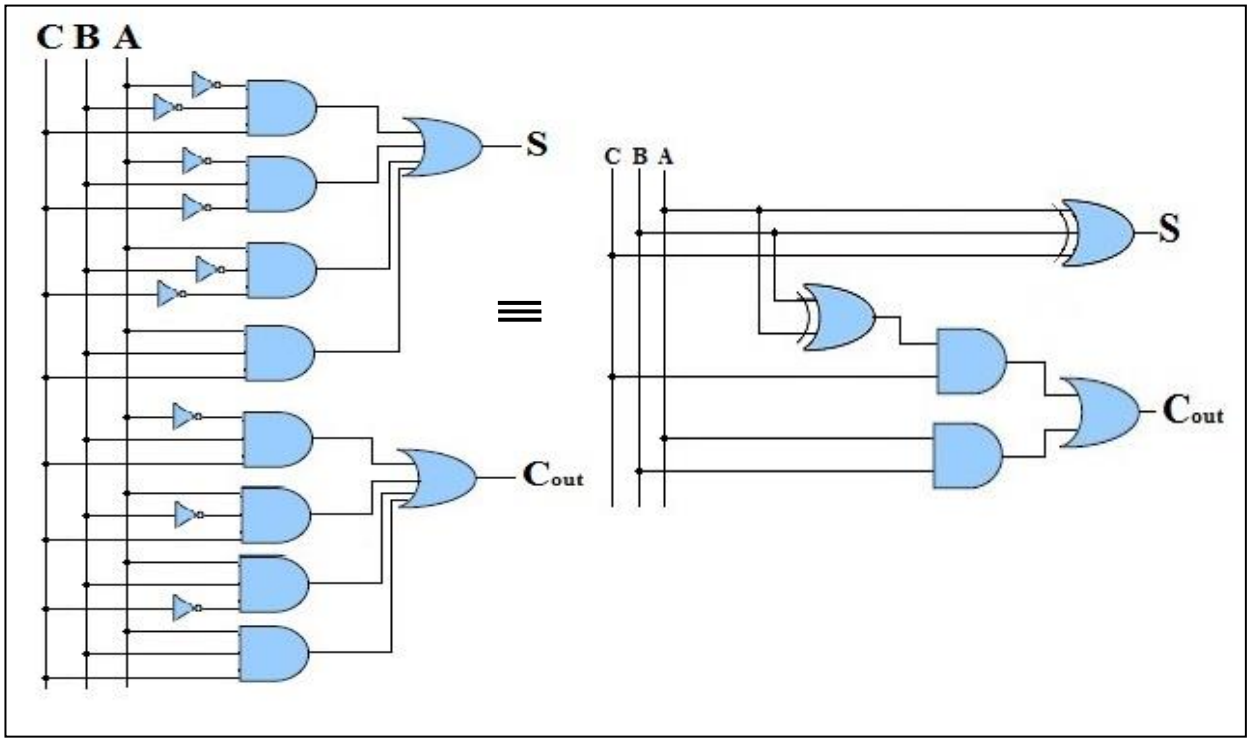
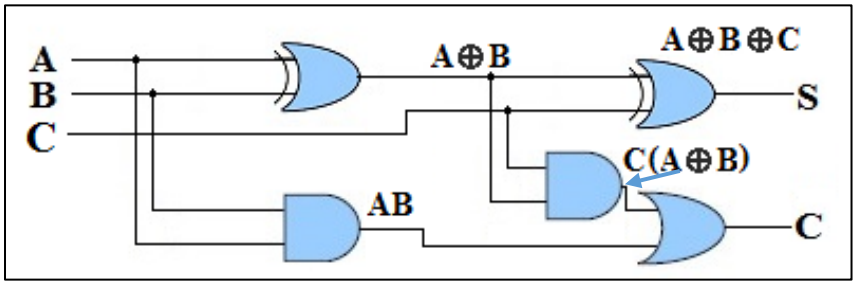


Fig 1.3: Full-adder circuit.

Example: Implement the full adder with half adder gates.

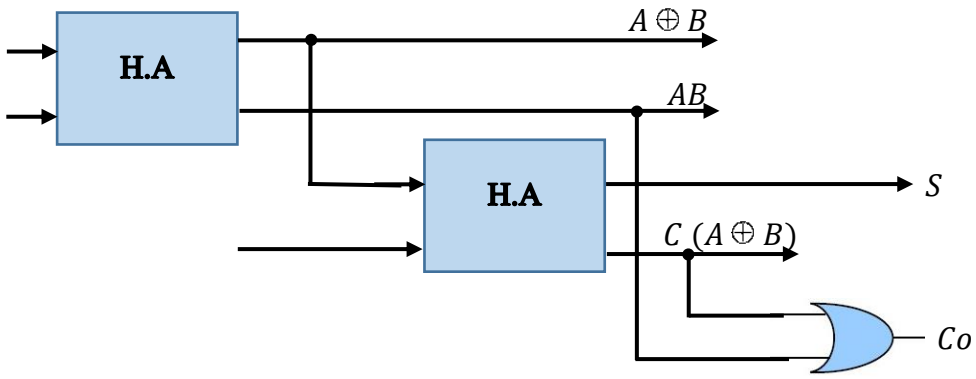
Solution:



Example: Implement the full adder circuit block diagram using half adder gates.

Solution: $S = A \oplus B \oplus C$

$C_{out} = AB + C(A \oplus B)$

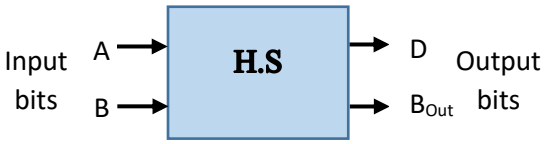


4.3 Half-Subtractors

Combinational circuit that subtract two bits and produces their difference. The truth table for the input and output relationships of a half-subtractor con be derived as follows:

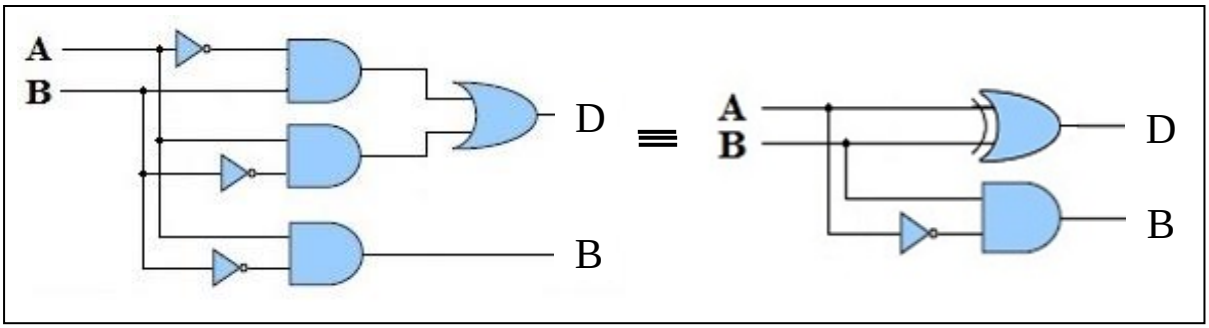
Table 3: Half-Subtractor truth table.

Input		Output	
A	B	Borrow (B)	Difference (D)
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0



$Difference = \bar{A}B + A\bar{B} = A \oplus B$

$Borrow = \bar{A}B$

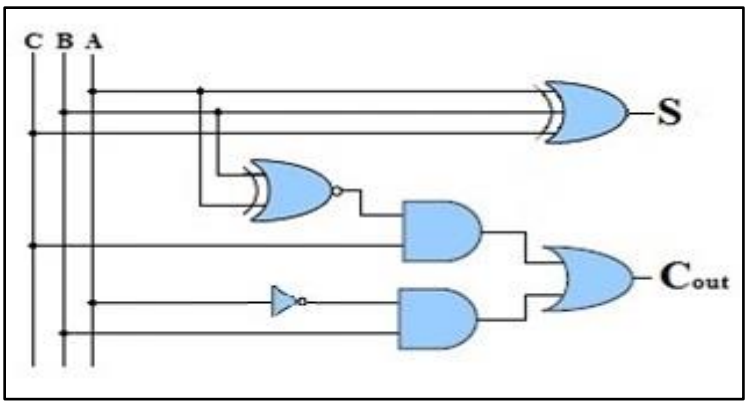


4.4 Full-Subtractors

Combinational circuit that performs a subtraction between two bits, taking into account that a 1 may have been borrowed by a lower significant stage. It has three input and two output. The truth table for the circuit is as follows:

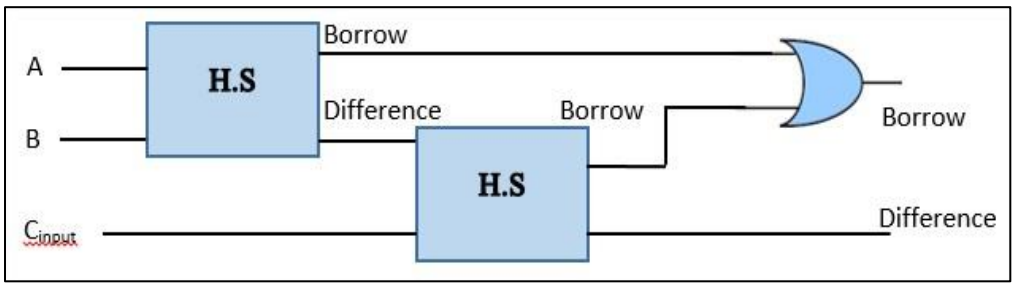
Input			Output	
A	B	C	Borrow (B)	Difference (D)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$D = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$
$$= \bar{C}(\bar{A}B + A\bar{B}) + C(\bar{A}\bar{B} + AB)$$
$$= \bar{C}(A \oplus B) + C(\overline{A \oplus B})$$
$$D = A \oplus B \oplus C$$
$$B = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + ABC$$
$$= C(\bar{A}\bar{B} + AB) + \bar{A}B(\bar{C} + C)$$
$$B = \bar{A}B + C(\overline{A \oplus B})$$



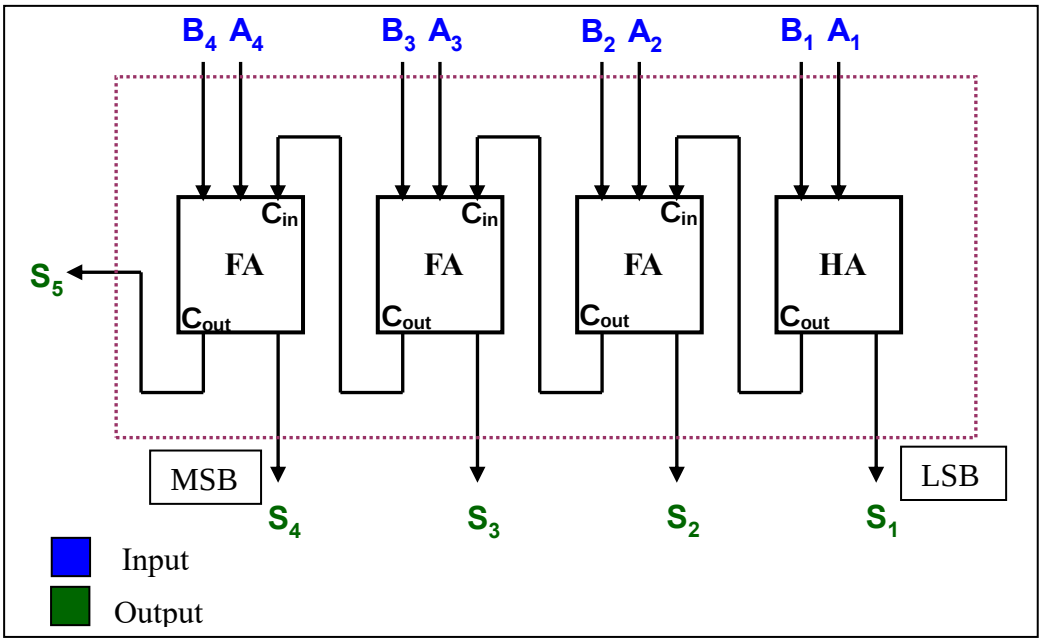
Example: Design a full subtractor circuit block diagram using half subtractor.

Solution:



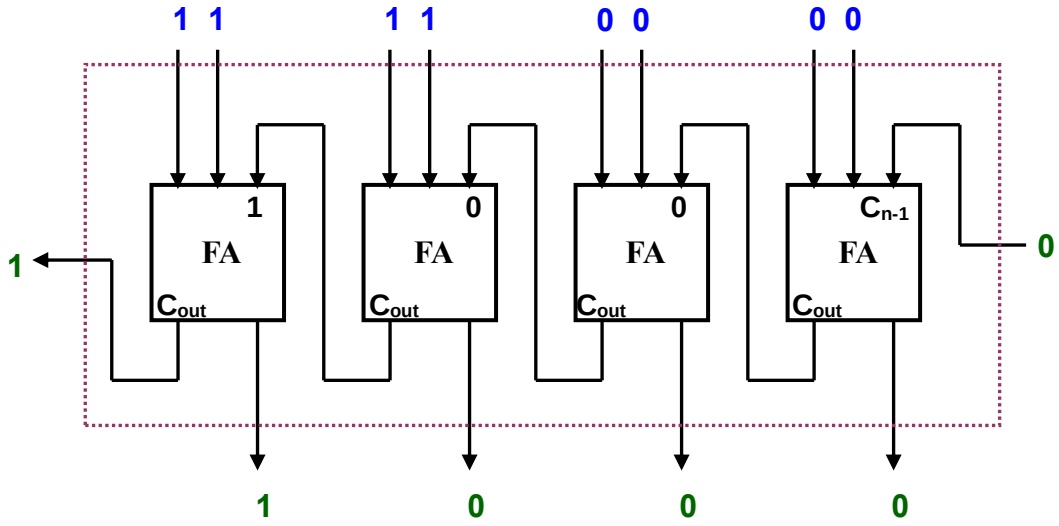
4.5 Parallel adder:

Called Parallel Adder because inputs are presented simultaneously (in parallel). Also, called Ripple-Carry Adder. We can connect multi-adders to add two binary number. The number of adders used depend on the number of bits in each number. For two number of two bits, 2 adders are needed, and for 3 bits number, three adders are needed. The figure below shows the binary adders for two number with 4 bits each. The first column requires only a half adder. For any column above the first, there may be a carry form the preceding column, therefore, we must use a full adder for each column above the first.



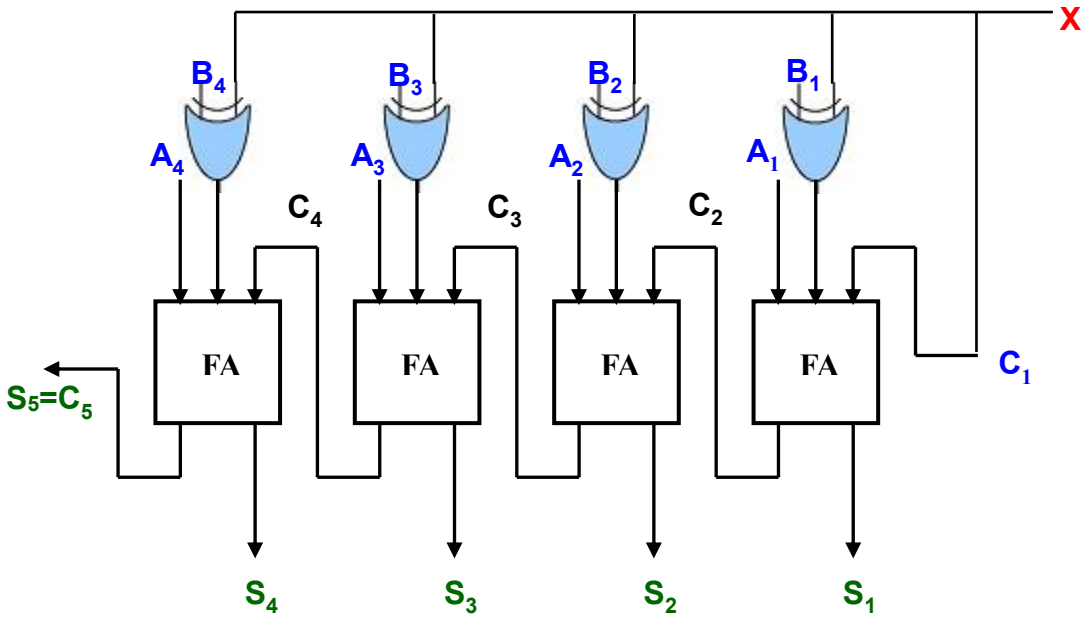
Example: Use the 4-bit parallel adder to find the sum and output carry for the addition of the following two 4-bit numbers 1100 and 1100 if the input carry (C_{n-1}) is 0.

Solution:



Example: Design an adder/subtractor circuit using full adders and gats.

Solution:



When:

X=0: adding operation and output result = $S_5S_4S_3S_2S_1$

X=1: subtracting operation and output result = $S_3S_2S_1S_0$ whole S_5 or C_5 will be neglected due to 2's complement operation.

4.6 Parity generator/checker

Parity is a very useful tool in information processing in digital computers to indicate any presence of error in bit information. External noise and loss of signal strength cause loss of data bit information while transporting data from one device to other device, located inside the computer or externally. To indicate any occurrence of error, an extra bit is included with the message according to the total number of 1s in a set of data, which is called parity. The two methods used are:

1- Even Parity

It means attaching an extra bit to a group of bits to produce an even number of (1).

2- Odd Parity

It means attaching an extra bit to a group of bits to produce an odd number of (1).

The table list below shows the parity bit for BCD code number for both even and odd parity:

BCD	Even Parity	Odd Parity
0 0 0 0	0	1
0 0 0 1	1	0
0 0 1 0	1	0
0 0 1 1	0	1
0 1 0 0	1	0
0 1 0 1	0	1
0 1 1 0	0	1
0 1 1 1	1	0
1 0 0 0	1	0
1 0 0 1	0	1

Notes:

- The parity bit can be attached to the code group at either the beginning or the ends depending on the system design.
- The odd parity is always the complement of even parity bit.
- The parity bit generator operation is made at transmission side and the parity checking operation is made at receiving side.

Parity logic:

In order to check for generation the parity in a given code word, the basic principle can used is the sum. The sum of an even numbers of 1's is always zero, and the sum of odd numbers of 1's is always one. Therefore, in order to determine if the given word is even or odd parity all of the bits in that code word are summed.

Example: Design and even/odd parity generator for 3 bit data.

Solution:

A Truth table of even parity and odd parity is show as below:

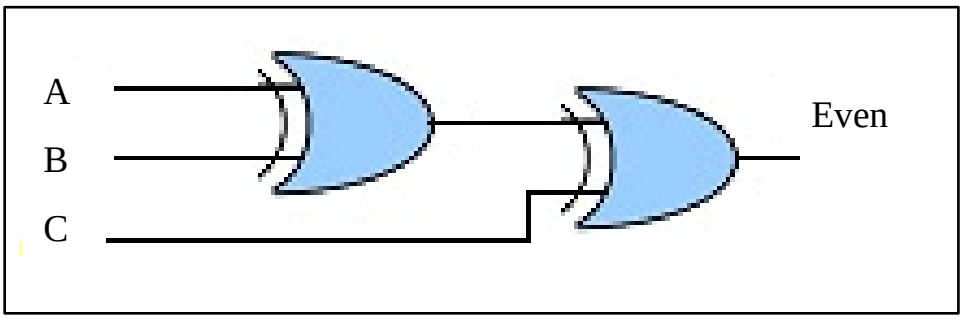
Input			Output	
C	B	A	Even Parity *Pe*	Odd Parity *Po*
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

$$Pe = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$= \bar{A}(\bar{B}C + B\bar{C}) + A(\bar{B}\bar{C} + BC)$$

$$= \bar{A}(B \oplus C) + A\overline{(B \oplus C)}$$

$$Pe = A \oplus B \oplus C$$

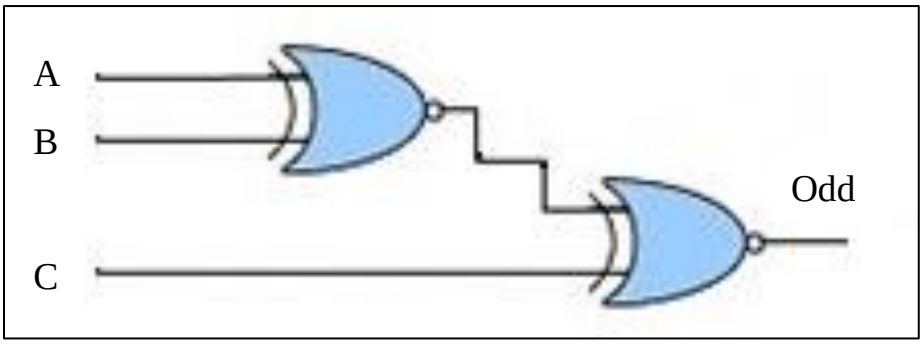


$$Po = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$= \bar{A}(\bar{B}\bar{C} + BC) + A(\bar{B}C + B\bar{C})$$

$$= \bar{A}\,\overline{(B \oplus C)} + A(B \oplus C)$$

$$Po = \overline{A \oplus B \oplus C}$$



Example: Suppose you receive a binary bit word "0101" and you know you are using an odd parity. Is the binary word error?

Solution:

- The answer is yes.
- There are 2 1-bit, which is an even number
- We are using an odd parity
- So there must have an error.

Example: Assume we are using even parity with 7-bit.The letter in 7-bit is encoded as 0110101. How will the letter be transmitted? If we are using odd parity, how will the letter be transmitted?

Solution:

- Because there are four 1s (an even number), parity is set to zero.
- This would be transmitted as: 01101010.
- If we are using an odd parity:
- The letter will be transmitted as 01101011.

Example: Suppose you are using an odd parity. What should the binary word "1010" look like after you add the parity bit?

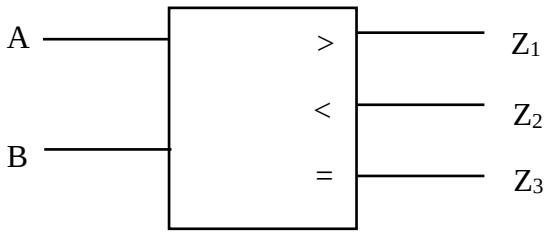
Solution:

- There is an even number of 1-bits.
- So we need to add another 1-bit
- Our new word will look like "10101".

4.7 Comparator

A magnitude comparator is a combinational circuit that compares two numbers (A and B) and determine their relative magnitudes. The output of the comparison are specified by 3 binary variables that indicate whether (A=B) or (A>B) or (A<B).

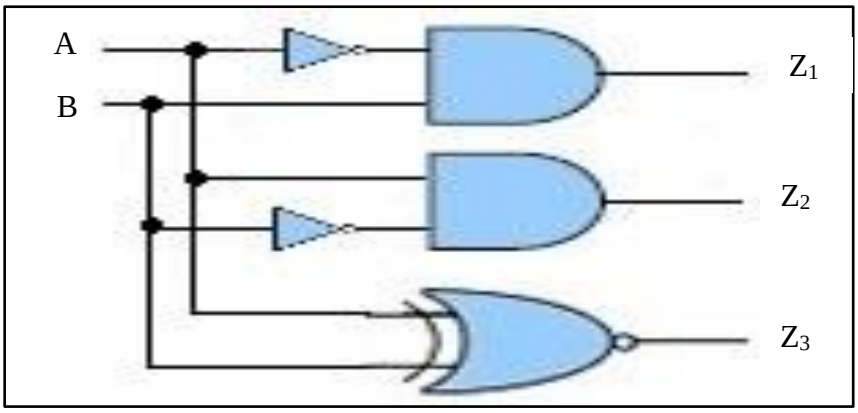
To compare two numbers having n-bits, we have 2^n entries in the truth table. The block diagram for comparator is shown below:



For two binary numbers with one-bit comparator the truth table is:

Input		Output		
A	B	(A>B), Z_1	(A<B), Z_2	(A=B), Z_3
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

$Z1 = A\bar{B}, \qquad Z2 = \bar{A}B, \qquad Z3 = \bar{A}\bar{B} + AB = \overline{A \oplus B}$



Example: Compare two binary numbers having 2-bits each.

Solution: $A=A_1A_0$, $B=B_1B_0$

Input				Output		
A		B				
A_1	A_0	B_1	B_0	(A>B), Z_1	(A<B), Z_2	(A=B), Z_3
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	1
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	1	0	1	0
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	0	1

$$(A > B) = \overline{A_1}A_0\overline{B_1}\,\overline{B_0} + A_1\,\overline{A_0}\,\overline{B_1}\,\overline{B_0} + A_1\,\overline{A_0}\,\overline{B_1}\,B_0 + A_1A_0\overline{B_1}\,\overline{B_0} + A_1A_0\overline{B_1}\,B_0 + A_1A_0B_1B_0$$

By using K-Map we get

$$A > B := A1\overline{B1} + A0\,\overline{B0}\,\overline{A1}\,\overline{B1} + A1B1\,A0\,\overline{B0}$$

$$= A1\overline{B1} + A0\overline{B0}(\overline{A1}\,\overline{B1} + A1B1)$$

$$= A1\overline{B1} + A0\overline{B0}\,X1$$

$$(A < B) = \overline{A1}\,\overline{A0}\,\overline{B1}B0 + \overline{A1}\,\overline{A0}\,B1\overline{B0} + \overline{A1}\,\overline{A0}B1B0 + \overline{A1}A0B1\overline{B0} + \overline{A1}A0B1B0 + A1\overline{A0}B1B0$$

By using K-Map we get

$$A < B: = \overline{A1}B1 + \overline{A0}B0\overline{A1}\,\overline{B1} + \overline{A0}B0A1B1$$

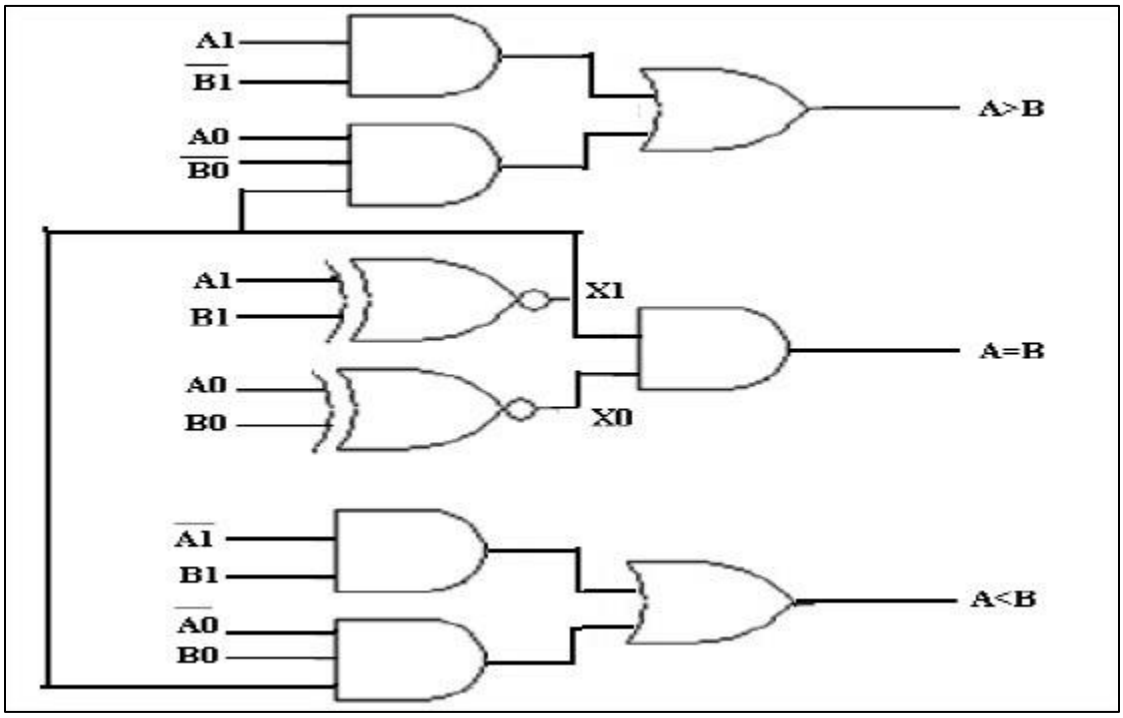
$$= \overline{A1}B1 + \overline{A0}B0(\overline{A1}\,\overline{B1} + A1B1)$$

$$= \overline{A1}B1 + \overline{A0}B0\,X1$$

$$(A = B) = \overline{A1}\,\overline{A0}\,\overline{B1}\,\overline{B0} + \overline{A1}A0\overline{B1}B0 + A1\overline{A0}B1\overline{B0} + A1A0B1B0$$

$$= (\overline{A1}\,\overline{B1} + A1B1)\,(\overline{A0}\,\overline{B0} + A0B0)$$

$$= X1X0$$



4.8 Decoder and Encoder

Decoder

A decoder is a combinational logic circuit that converts coded information such as binary, into a recognizable form, such as decimal. That decoders are called n-to-m line decoders where n is the input and m is the output.

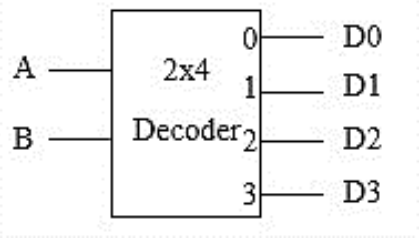
There are many type of decoders such as, binary decoder, BCD decoder and 7-segment decoder…etc.

Binary decoder

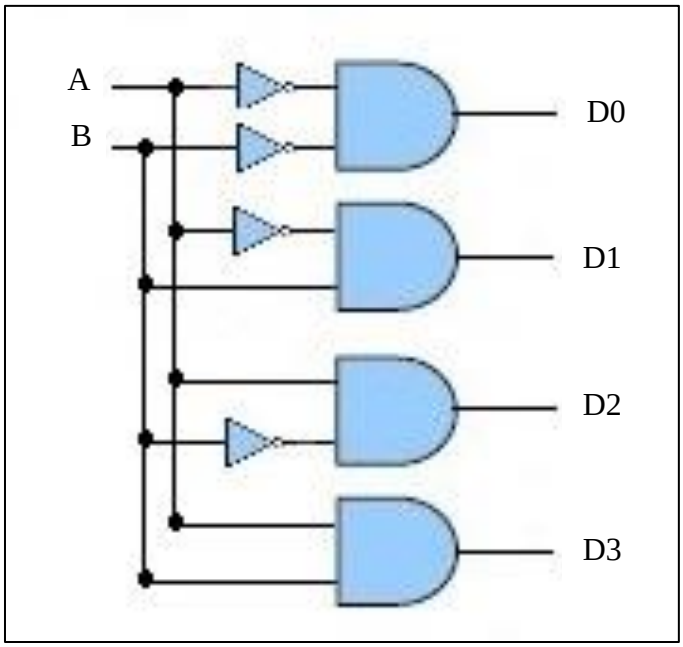
It converts binary coded information into a decimal recognizable form. The output lines of this decoder is $\mathbf{2^n}$ where "n" is the number of bits of binary input such as 2x4 decoder, 3x8 decoder, 4x16 decoder.

A 2x4 line decoder circuit is shown below with its truth table:

Input		Output			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

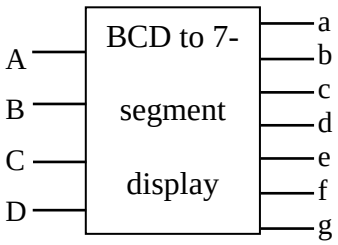


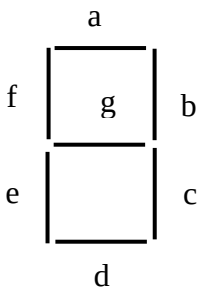
$D0 = \bar{A}\bar{B}, \quad D1 = \bar{A}B, \quad D2 = A\bar{B} \quad and \quad D3 = AB$

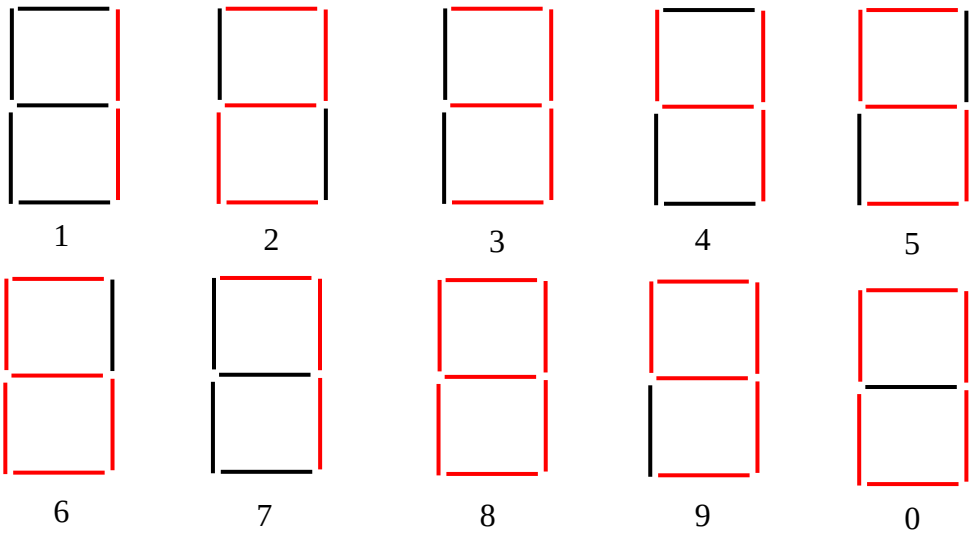


BCD to 7-segment decoder

For this type of decoder, the input is BCD digit and the output is a decimal digit display as shown: -







	Input				Output						
	A	B	C	D	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

Note: If the 7-segment work *active high* it mean 1 is on and 0 is off, if work *active low* it means 0 is on and 1 is off.

Example: Implement a full-adder circuit with a decoder and two OR gates.

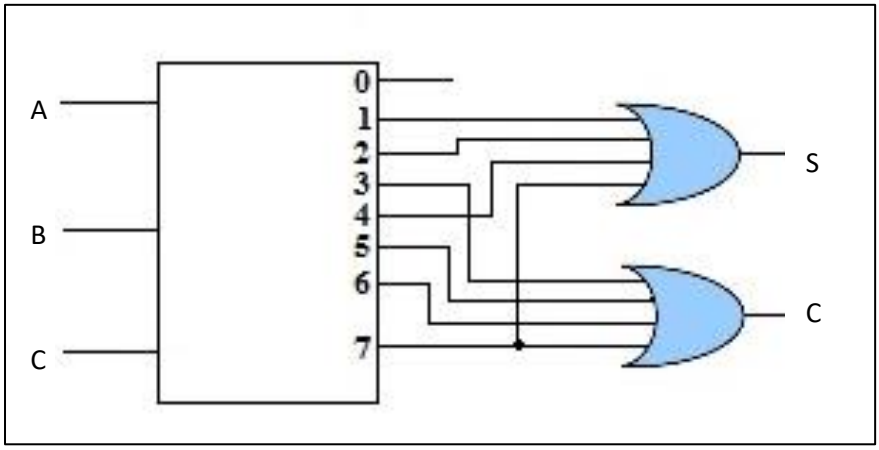
Solution: Form the truth table of full-adder below we obtain the function for this combinational circuit:

Table 2: Full-adder truth table.

Input			Output	
A	B	C_{in}	Carry (C)	Sum (S)
0	0	0	0	0
0	0	1	0	**1**
0	1	0	0	**1**
0	1	1	**1**	0
1	0	0	0	**1**
1	0	1	**1**	0
1	1	0	**1**	0
1	1	1	**1**	**1**

S (A, B, C) =∑1, 2, 4, 7

C (A, B, C) =∑3, 5, 6, 7



Example: Implement the following Boolean function with a decoder

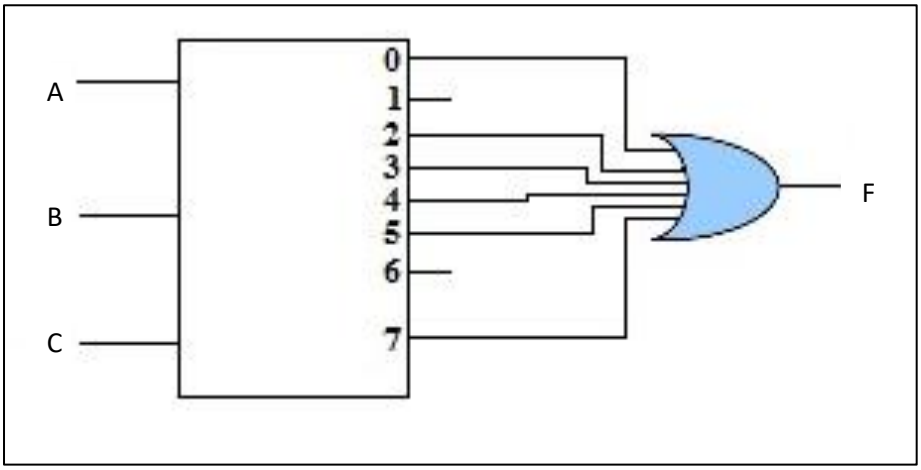
$$F = \bar{A}B + AC + BC + \bar{B}\bar{C}$$

Solution:

$$F = \bar{A}B(C + \bar{C}) + AC(B + \bar{B}) + BC(A + \bar{A}) + \bar{B}\bar{C}(A + \bar{A})$$

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

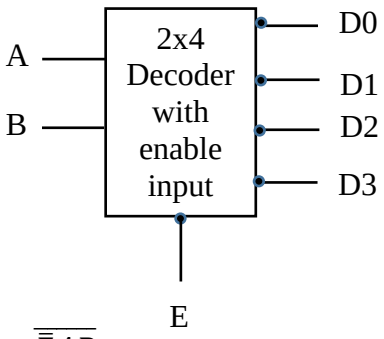
$$= \bar{A}BC + \bar{A}B\bar{C} + A\bar{B}C + ABC + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} = \sum 0,2,3,4,5,7$$



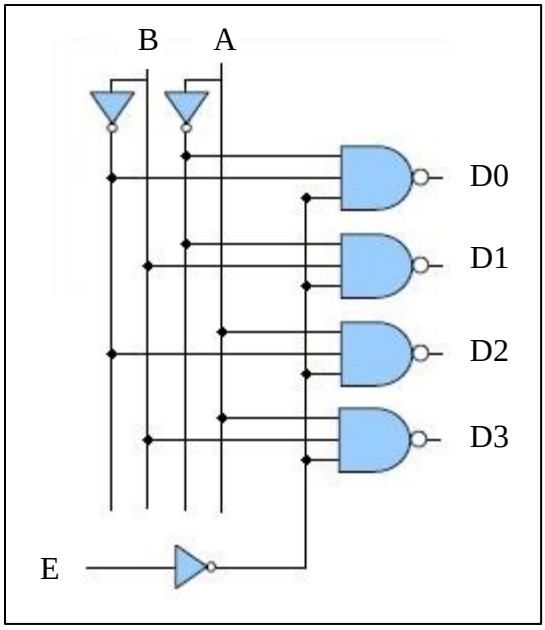
Decoder with enable input:

This type of decoder can be used to construct one or more of decoder circuit. A 2x4 decoder with an enable input constructed with NAND gat is shown below. The circuit operates with complemented outputs and a complement input. In general, a decoder may operate with complemented or un-complemented output. The enable i/p may be activated with "1" or with a "0" signal. Some decoder have 2 or more enable input that must satisfy a given logic conditions in order to enable the circuit.

Input			Output			
E	A	B	D_1	D_2	D_3	D_4
1	x	x	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

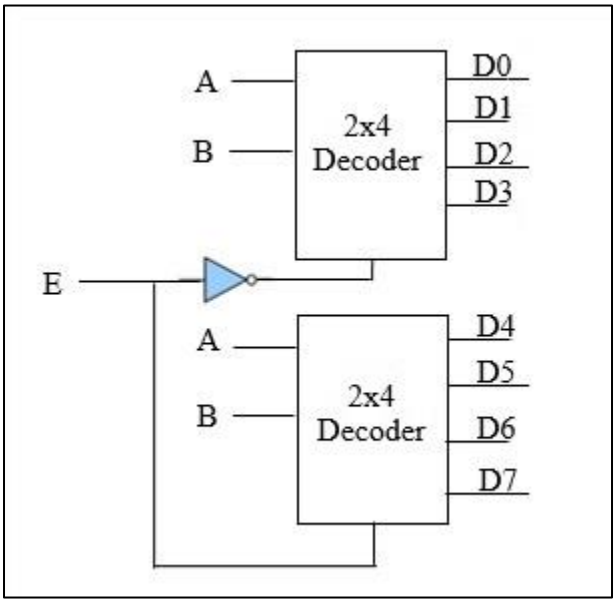


$D0 = \overline{\bar{E}\bar{A}\bar{B}}$, $D1 = \overline{\bar{E}\bar{A}B}$, $D2 = \overline{\bar{E}A\bar{B}}$, $D3 = \overline{\bar{E}AB}$



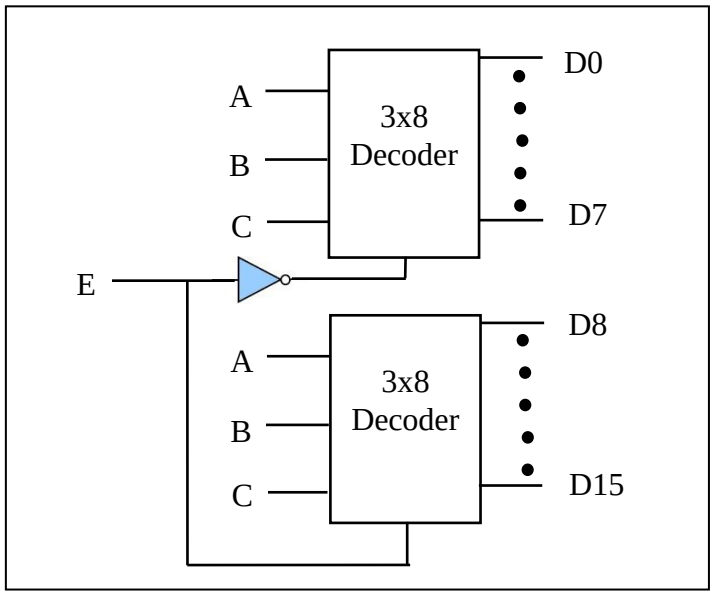
Example: Design a 3x8 decoder using two 2x4 decoder with enable input.

Solution:



Example: Design a 4x16 decoder using two 3x8 decoder with enable input.

Solution:



Example: Implement the following logic function using 2x4 decoder

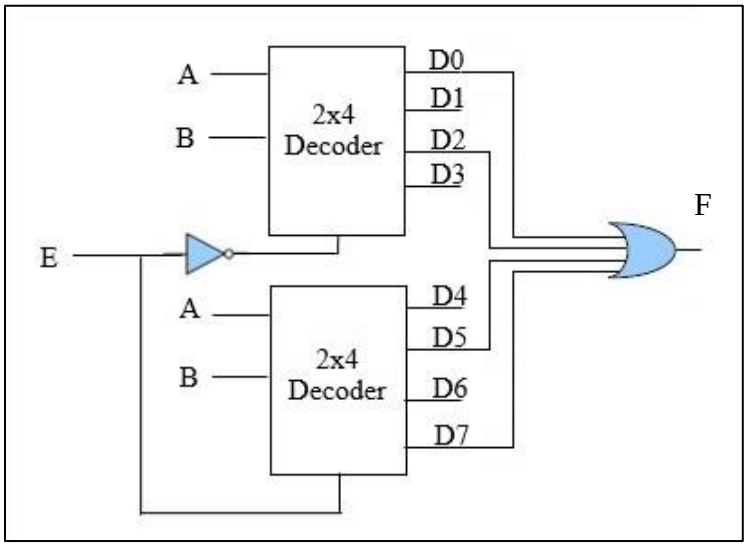
$$F = \sum (0, 2, 5, 7)$$

Solution:

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

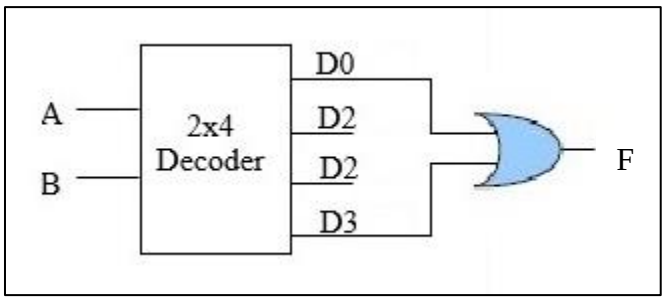
This function should be implement with 3x8 decoder. To implement it with a 2x4 decoder there two methods:

1- By using two 2x4 decoder with enable inputs.



2- By simplifying the Boolean function

$$F = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}C + ABC$$
$$F = \bar{A}\bar{C}(\bar{B} + B) + AC(\bar{B} + B)$$
$$F = \bar{A}\bar{C} + AC = \sum(0, 3)$$



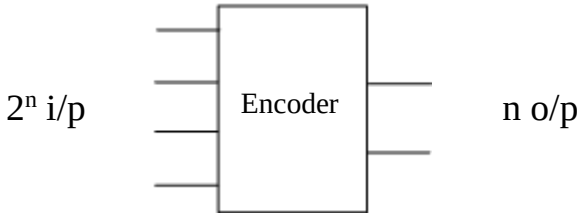
Homework: Design 4x16 decoder by using 2x4 decoder.

Homework: Design 3x8 decoder by using 1x2 decoder.

Encoder

An encoder is a digit circuit that performs the inverse operation of a decoder. An encoder has 2n (or less) input lines and n output lines.

It has eight inputs, and three outputs that generate the corresponding binary number.



An example of encoder is the octal to binary encoder whose truth table is shown below:

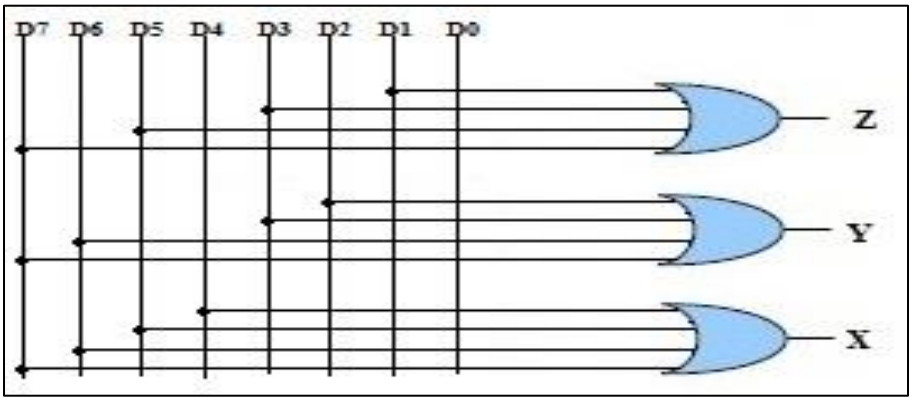
Input								Output		
D7	D6	D5	D4	D3	D2	D1	D0	A2	A1	A0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$A0 = D1 + D3 + D5 + D7$

$A1 = D2 + D3 + D6 + D7$

$A2 = D4 + D5 + D6 + D7$

(Implementation in three OR gates)



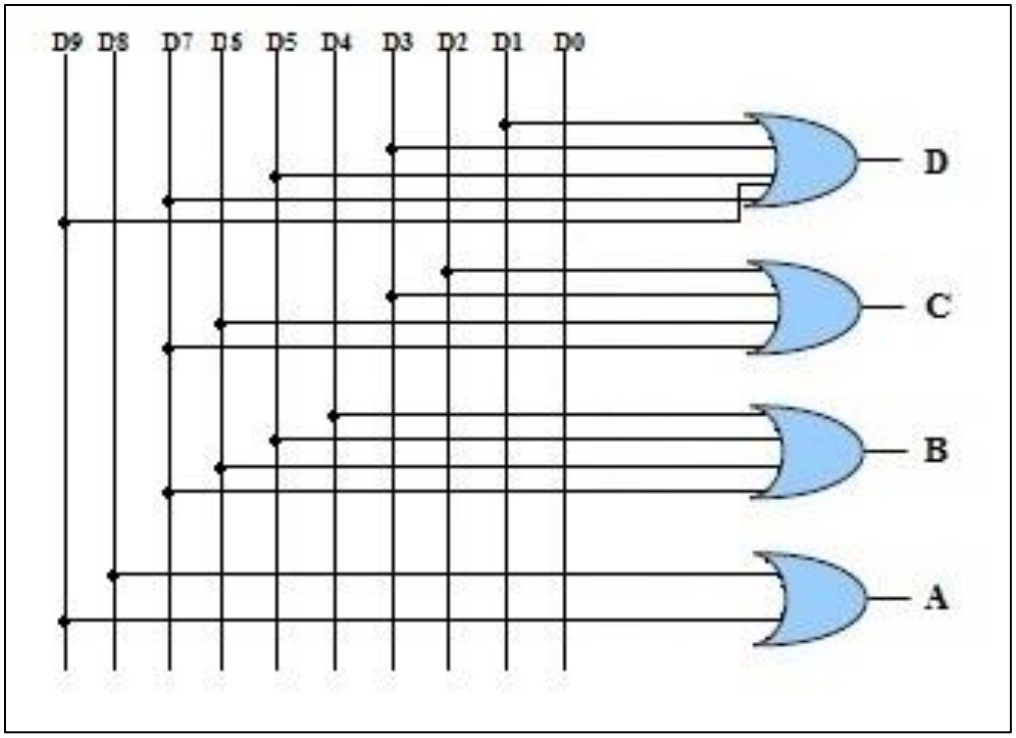
Example: Design encoder that convert decimal number to BCD.

Solution:

This type of encoder has tem inputs one for each decimal digit and four outputs corresponding to the BCD code

Input (Decimal)										Output (BCD)			
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	A	B	C	D
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

$$D = D1 + D3 + D5 + D7 + D9$$
$$C = D2 + D3 + D6 + D7$$
$$B = D4 + D5 + D6 + D7$$
$$A = D8 + D9$$

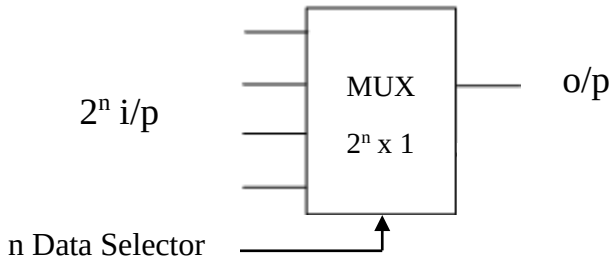

D9 D8 D7 D5 D5 D4 D3 D2 D1 D0
D
C
B
A

4.9 Multiplexers and Demultiplexers

Multiplexers

A multiplexer is a combinational circuit that receiver binary information form one of 2n input data lines and directs it to a single output line. Multiplexer is called Data Selector.

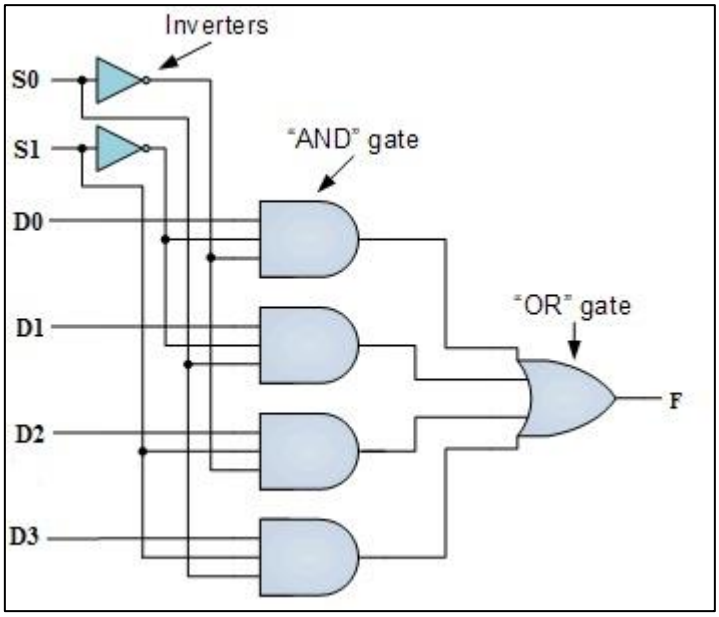
The selection of a particular input data line for the output is determined by a set of selection inputs. A 2n- to- 1 multiplexer has 2^n input data line and n input selection lines whose bit combinations determine which data are selected for the o/p..



Example: Design 4X1 multiplexer.

Solution: $2^n = 4$ ⟶ n=2 ⟶ No. of selection line =2

Selected		Data
S1	S0	D
0	0	D0
0	1	D1
1	0	D2
1	1	D3



$$F = \overline{S1}\,\overline{S0}D0 + \overline{S1}S0D1 + S1\overline{S0}D2 + S1S0D3$$

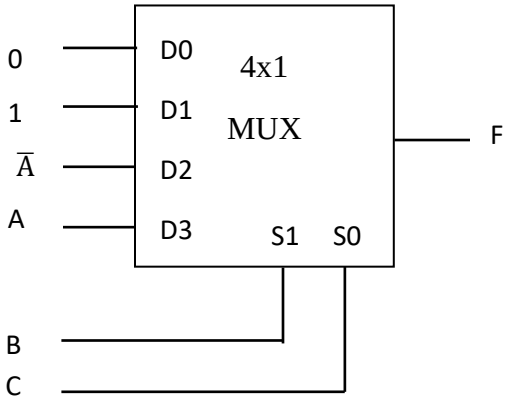
Example: Implement the following logic function using a multiplexer.

$$F(A,B,C) = \sum(1,3,5,6)$$

Solution:

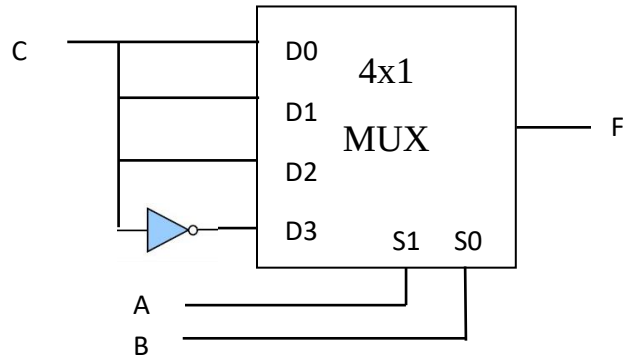
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

If we take the B C is the selector so A as input



	D0	*D1*	*D2*	*D3*
$\bar{A}$	0	**1**	2	**3**
A	4	**5**	**6**	7
	0	1	A	$\bar{A}$

If we take the A B is the selector so C as input

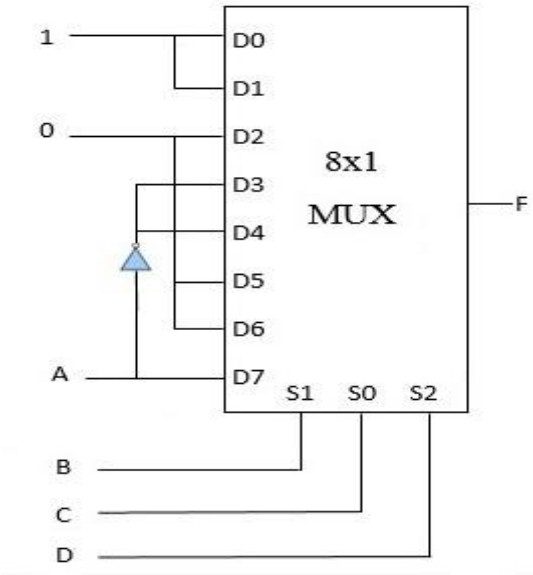


	D0	*D1*	*D2*	*D3*
$\bar{C}$	0	2	4	**6**
C	**1**	**3**	**5**	7
	C	C	C	$\bar{C}$

Example: Implement the following logic function using a multiplexer.

$$F(A,B,C,D) = \sum(0,1,3,4,8,9,15)$$

Solution:



	D0	*D1*	*D2*	*D3*	*D4*	*D5*	*D6*	*D7*
$\bar{A}$	**0**	**1**	2	**3**	**4**	5	6	7
A	**8**	**9**	10	11	12	13	14	**15**
	1	1	0	$\bar{A}$	$\bar{A}$	0	0	A

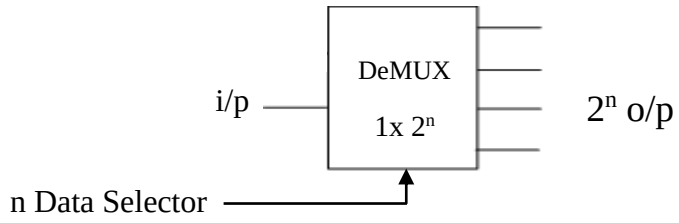
Homework: Design 4x1 multiplexer by using 2x1 multiplexer.

Homework: Design 8x1 multiplexer by using 2x1 multiplexer.

Homework: Design 8x1 multiplexer by using two 4x1 multiplexer and one 2x1 multiplexer.

Demultiplexers

A demultiplexer basically reverses the multiplexing function. It takes digital information from one line and distributes it to a given number of output lines. For this reason, the demultiplexer is also known as *a data distributor*.



No. of selected line = n

Example: Design 1 to 4 demultiplexer.

Solution:

A 1 to 4 lines demultiplexer circuit. The data input line goes to all of the AND gates. The two data select lines enable only one gate at a time, and the data appearing on the data input line will pass through the selected gate to the associated data output line.

No. of selected line to control 4 O/P = 2

Input		Output			
S1	S0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1