

Chapter 4

Combinational Logic Analysis and Design

Content

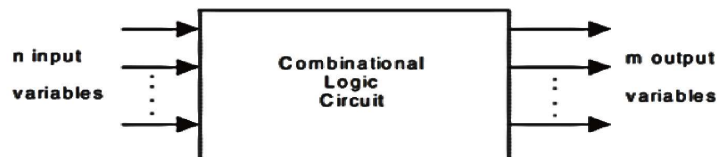
- Introduction
- Design Procedure
- Universal Logic Gates
- Adders and Subtractors
- Comparators
- Encoders and Decoders
- Multiplexers and De-multiplexers

Digital System

- The digital system consists of two types of circuits, namely
 - Combinational Logic Circuits and
 - Sequential Logic Circuits
- **A combinational Logic Circuit** consists of **logic gates**, where outputs at any instant are determined only by the present combination of inputs.
- **A Sequential Logic Circuits** contain **logic gates** as well as **memory cells**. And Their outputs depend on the **present inputs** and also on the **states of memory** elements.

Combinational Logic Circuit

- A combinational circuit consists of input variables, logic gates, and output variables.
- The logic gates accept signals from inputs and output signals are generated according to the logic circuits employed in it
- Figure below shows a block diagram of a combinational logic circuit.
- There are n number of input variables coming from an electric source and m number of output signals go to an external destination.



Design Procedure Combinational Logic Circuit

- Any combinational circuit can be designed by the following steps of design procedure.
 1. The problem is stated.
 2. Identify the input variables and output functions.
 3. The input and output variables are assigned letter symbols.
 4. The truth table is prepared that completely defines the relationship between the input variables and output functions.
 5. The simplified Boolean expression is obtained by any method of minimization—algebraic method, Karnaugh map method, or tabulation method.
 6. A logic diagram is realized from the simplified expression using logic gates.
- The output Boolean functions from the truth table are simplified by any available method, such as algebraic manipulation, the K-map method, or the tabulation procedure.

Design Constraints

- However, in any particular application, certain restrictions, limitations, and criteria will serve as a guide in the process of choosing a particular algebraic expression.
- A practical design method would have to consider such constraints as
 1. Minimum number of gates,
 2. Minimum number of inputs to a gate,
 3. Minimum propagation time of the signal through the circuit,
 4. Minimum number of interconnections, and
 5. Limitations of the driving capabilities of each gate.

Design Constraints cont.

- Since all these criteria **cannot be satisfied** simultaneously, and since the importance of each constraint is dictated by the **particular application**, it is difficult to make a general statement as to what constitutes an acceptable simplification.
- In most cases, the simplification begins by satisfying **an elementary objective**, such as producing a simplified Boolean function in **a standard form**, and from that proceeds to meet any other performance criteria.

Example 1: Design a logic circuit that has three inputs, A, B, and C, and whose output will be HIGH only when a majority of the inputs are HIGH.

Example 2: Design a logic circuit that performs arithmetic operation for adding two bits.

Universal Logic Gates

- Digital circuits are frequently constructed with NAND or NOR gates
- NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all IC digital logic families
- Rules and procedures have been developed for the conversion from Boolean functions given in terms of AND, OR, and NOT into equivalent NAND and NOR logic diagrams.

NAND Implimentation

- The NAND gate is said to be a universal gate because any digital system can be implemented with it.
- To facilitate the conversion to NAND logic, it is convenient to use the two alternate graphic symbols shown in Figure below.

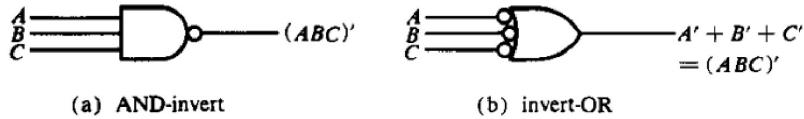
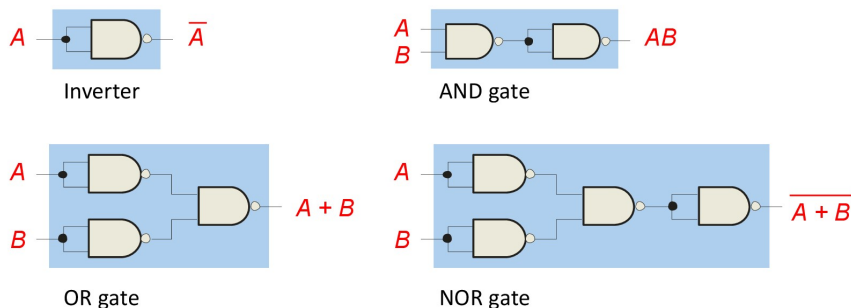


FIGURE. Two graphic symbols for a NAND gate

- The **AND-invert** graphic symbol consists of an AND graphic symbol followed by a small circle.
- The **invert-OR** graphic symbol consists of an OR graphic symbol that is preceded by small circles in all the inputs.
- **Either** symbol can be used to represent a NAND gate.

NAND Gates

NAND gates are sometimes called **universal** gates because they can be used to produce the other basic Boolean functions.



Cont....

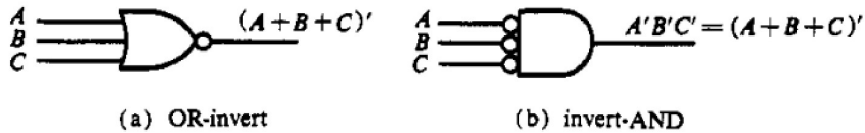
- To obtain a multilevel NAND diagram from a Boolean expression, proceed as follows:
 - a) From the given Boolean expression, **draw the logic diagram** with AND, OR, and inverter gates. Assume that **both** the normal and complement inputs are available.
 - b) Convert all **AND** gates to **NAND gates** with **AND-invert** graphic symbols.
 - c) Convert all **OR** gates to **NAND gates** with **invert-OR** graphic symbols.
 - d) Check all small circles in the diagram. For every small circle that is not compensated by another small circle along the same line, insert an inverter (one-input NAND gate) or complement the input variable.
- In general, the number of **NAND** gates required to implement a Boolean expression is **equal to** the number of **AND-OR** gates except for an occasional inverter.

Cont...

- Example: implement using only NAND gates.
 - a) $F = A + (B' + C)(D' + BE')$
 - b) $F = (CD + E)(A + B')$
 - c) $F = BC' + A(B + CD)$
 - d) $F = (AB' + CD')E + BC(A + B)$
 - e) $F = w(x + y + z) + xy;$

NOR Implimentation

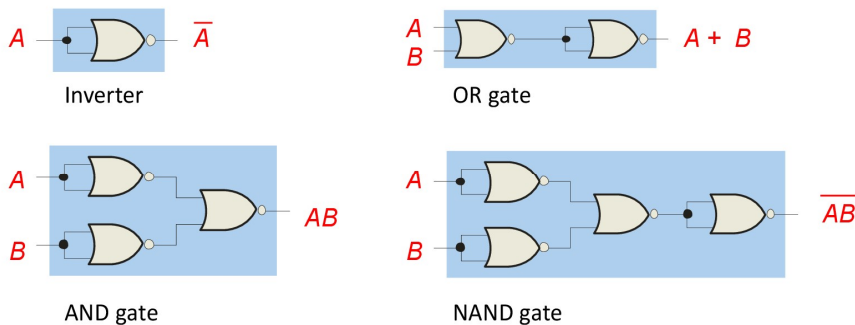
- The two graphic symbols for the NOR gate are shown in Fig. below.



- The OR-invert symbol defines the NOR operation as an OR followed by a complement.
- The invert-AND symbol complements each input and then performs an AND operation.
- The two symbols designate **the same NOR** operation and are logically identical because of DeMorgan's theorem.

NOR Gates

NOR gates are also **universal** gates and can form all of the basic gates.

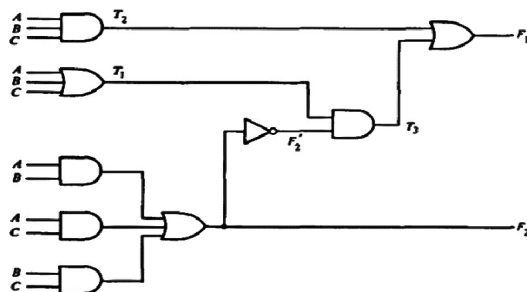


NOR Implimentation

- The procedure for implementing a Boolean function with NOR gates is similar to the procedure outlined in the previous section for NAND gates.
 - Draw the **AND-OR** logic diagram from the given algebraic expression Assume that both the normal and complement inputs are available.
 - Convert all **OR gates to NOR** gates with OR-invert graphic symbols.
 - Convert all **AND gates to NOR** gates with invert-AND graphic symbols.
 - Any small circle that is not compensated by another small circle along the same line needs an inverter or the complementation of the input variable.

Cont...

- Example: implement the following Boolean function using only NOR gates.
 - $F = (AB + E)(C + D)$
 - $F = A + (B' + C)(D' + BE')$
 - $F = (CD + E)(A + B')$
 - $F = [(C + D)B' + A](B + C')$
 - $F = (AB' + CD')E + BC(A + B)$
 - $F = w(x + y + z) + xy;$
- Convert the logic diagram the following to a multiple-level NAND and NOR circuit



Half Adder

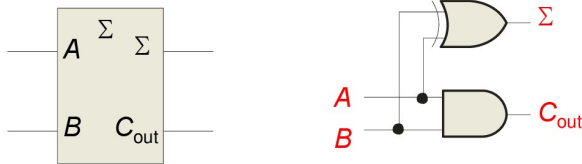
- Basic rules of binary addition are performed by a **half adder**, which has **two binary inputs** (A and B) and **two binary outputs** (Carry out and Sum).
- The inputs and outputs can be summarized on a truth table.
- The Σ (sum) represents **the least significant bit** of the sum.
- The C_{out} represents **the most significant bit** of the sum.
- The Boolean functions for the two output from the truth table:

$$\Sigma = A'B + AB' = A \oplus B$$

$$C_{out} = AB$$

The logic symbol and equivalent circuit are:

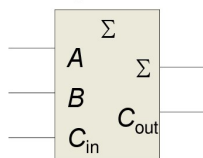
Inputs		Outputs	
A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Full Adder

- Full Adder** has three binary inputs (A , B , and Carry in) and two binary outputs (Carry out and Sum)

Symbol



The truth table summarizes the operation

Inputs			Outputs	
A	B	C_{in}	C_{out}	Σ
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

- The Boolean functions for the two output from the truth table:

$$\text{Sum} = A'B'C + A'BC' + AB'C' + ABC$$

$$C_{out} = A'BC + AB'C + ABC' + ABC$$

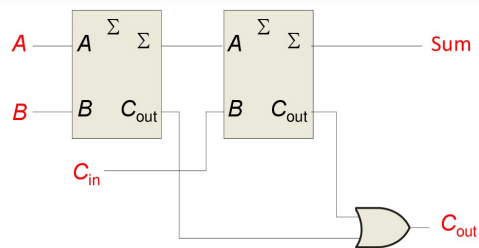
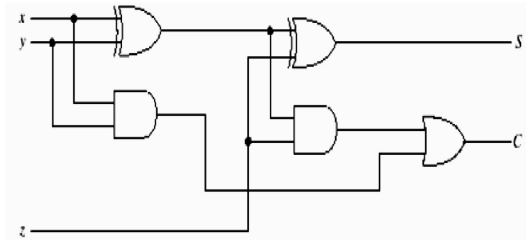
- Using X-OR,

$$\text{Sum} = A \oplus B \oplus C$$

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

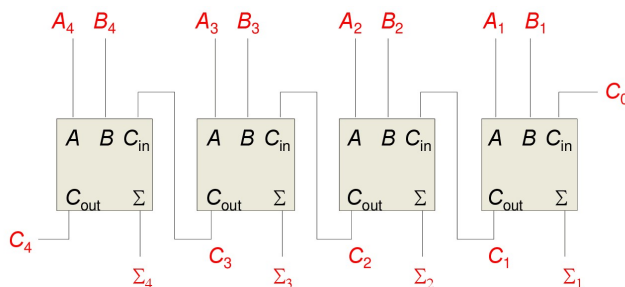
Full Adder

A full-adder can be constructed from two half adders as shown:



Parallel Adder

- Full adders are combined into parallel adders that can add binary numbers with multiple bits.
- It consists of full-adders connected in a chain, with the output carry from each full-adder connected to the input carry of the next full-adder in the chain.
- An n -bit parallel adder requires n full-adders.
- A 4-bit adder is shown.



The output carry (C_4) is not ready until it propagates through all of the full adders. This is called **ripple carry**, delaying the addition process.

Parallel Adder

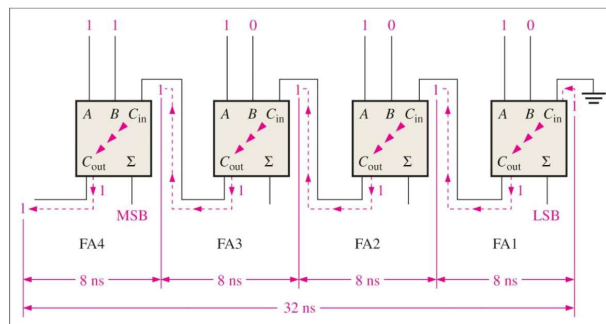
- Example:

- What are the sum outputs when 111 and 101 are added by the 3-bit parallel adder?
- What are the sum outputs when 1110 and 1011 are added by the 4-bit parallel adder?

Ripple Carry Adder

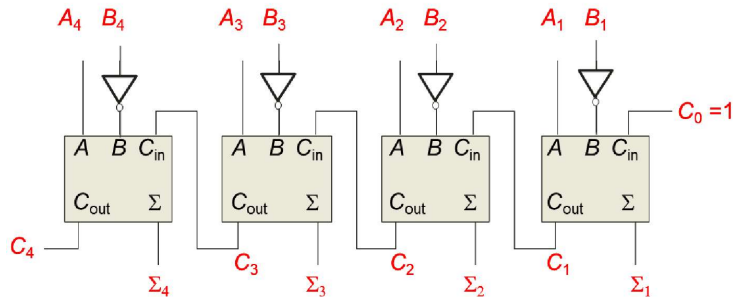
The parallel adder introduced earlier is a **ripple carry adder**. Because the carry from each adder is applied to the next, time must be provided for the carry bits to “ripple” through the circuit.

If each adder has an 8 ns delay,
it takes 32 ns for the carry to work its
way through the adder (as shown).



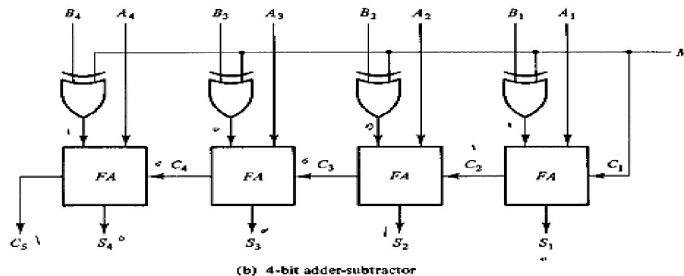
Subtractor

- The subtraction of binary numbers can be done most conveniently by means of 2's complements.
- Remember that the subtraction $A - B$ can be done by taking the 2's complement of B and adding it to A .
- The 2's complement can be obtained by taking the 1's complement and adding one to the least significant pair of bits.
- A 4-bit subtractor circuit is shown



Parallel Adder and Subtractor

- The addition and subtraction operations can be combined into one circuit with one common binary adder.
- This is done by including an exclusive-OR gate with each full adder.
- A 4-bit adder and subtractor is shown.



- The mode input M controls the operation.
- When $M = 0$, the circuit is an adder, and when $M = 1$, the circuit becomes a subtractor.
- Each exclusive-OR gate receives input M and one of the inputs of B .
- When $M = 0$, we have $B \oplus 0 = B$
- When $M = 1$, we have $B \oplus 1 = B'$ and $C_1 = 1$.

Comparator

- The function of a comparator is to compare the magnitudes of two binary numbers to determine the relationship between them.
- The comparison of two numbers is an operation that determines if one number is **greater than**, **less than**, or **equal to** the other number.
- A magnitude comparator is a combinational circuit that compares two numbers, A and B, and determines their relative magnitudes.
- The outcome of the comparison is specified by three binary variables that indicate whether $A > B$, $A = B$, or $A < B$.

Comparator

- Algorithm
 - $A = A_3A_2A_1A_0$; $B = B_3B_2B_1B_0$
 - $A=B$ if $A_3=B_3$, $A_2=B_2$, $A_1=B_1$ and $A_0=B_0$
 - equality: $x_i = A_iB_i + A_i'B_i'$
 - $(A=B) = x_3x_2x_1x_0$
- If the corresponding digit of A is 1 and that of B is 0, we conclude that $A > B$.
- If the corresponding digit of A is 0 and that of B is 1, we have that $A < B$.
 - $(A>B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$
 - $(A<B) = A_3'B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1 + x_3x_2x_1A_0'B_0$
- Implementation
 - $x_i = (A_iB_i' + A_i'B_i)'$

Comparator

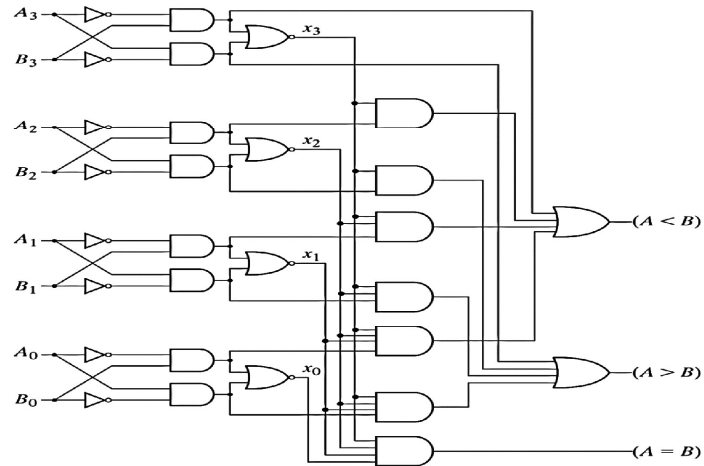
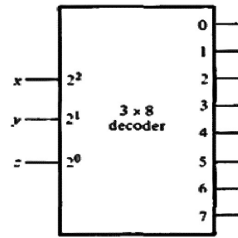


Fig. 4-17 4-Bit Magnitude Comparator

Decoder

- A **decoder** is a combinational circuit that converts **binary information from n input lines** to a **maximum of 2^n unique output lines**.
- If the n-bit decoded information has unused or don't-care combinations, the decoder output will have fewer than 2^n output.
- The decoders presented here are called n-to-m-line decoders, where $m \leq 2^n$.
- Their purpose is to generate the 2^n (or fewer) minterms of n input variables.
- The name decoder is also used in conjunction with some code converters such as a BCD-to-seven segment decoder.
- Only one output can be active (high) at any times.

3x8 Decoder



b) Block diagram

Truth Table of a 3-to-8-Line Decoder

Inputs			Outputs							
x	y	z	D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇
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	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

a) Truth table

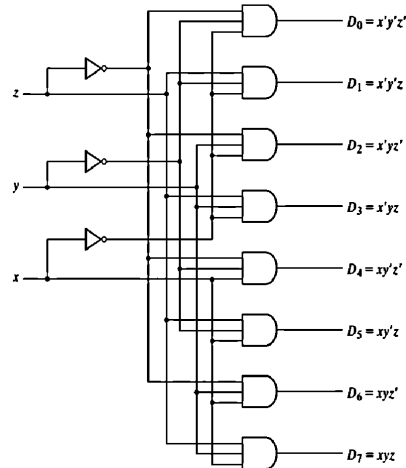


Fig. 4-18 3-to-8-Line Decoder

Decoder With Enable Input

- Normally every commercially available decoder ICs have a special input other than normal working input variables called ENABLE.
- The use of this ENABLE input is that when **activated** the complete IC comes to the **working condition for its normal functioning**.
- If ENABLE input is **deactivated** the IC goes to **sleep mode**, the normal functioning is **suspended**, and all the outputs become logic 0 irrespective of normal input variables conditions.
- Its function is to build **higher decoder from lower** decoders.

E	A	B	D ₀	D ₁	D ₂	D ₃
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

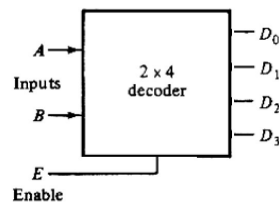
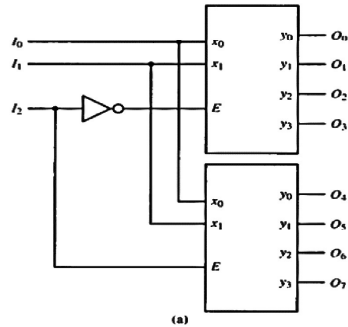


Figure: 2-to-4 decoder with enable input E.

Decoder

- Example:- Construct a 3-to-8 line decoder with the use of a 2-to-4 line decoder.



- Example:- Construct a 4-to-16 line decoder with the use of a 2-to-4 line decoder.
- Example:- Construct a 5-to-32 line decoder with the use of a 3-to-8 line decoder.

Encoder

- An encoder is a digital circuit that performs the **inverse operation** of a decoder.
- An encoder has 2^n (or fewer) input lines and n output lines.
- The output lines generate the binary code corresponding to the input value.
- It is assumed that only one input has a value of 1 at any given time; otherwise the circuit has no meaning.

8x3 Encoder

- The encoder can be implemented with OR gates whose inputs are determined directly from the truth table.
- These conditions can be expressed by the following output Boolean functions:

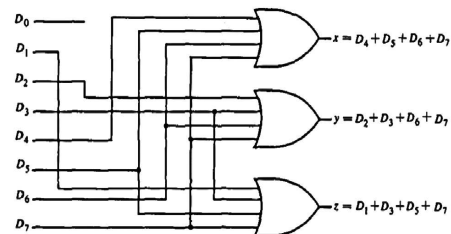
$$z = D_1 + D_3 + D_5 + D_7$$

$$y = D_2 + D_3 + D_6 + D_7$$

$$x = D_4 + D_5 + D_6 + D_7$$

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

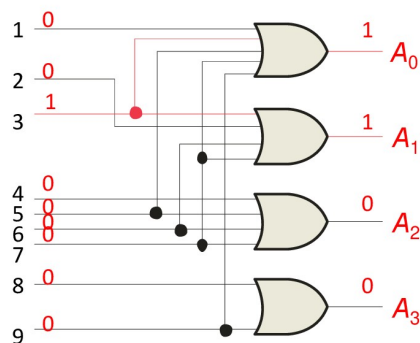
Truth table of 8x3 encoder



Encoder

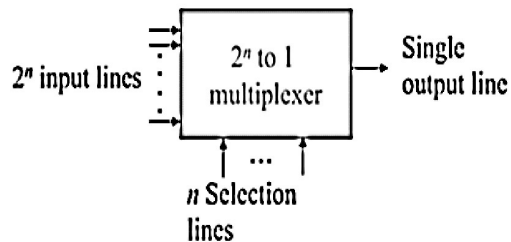
Example Show how the decimal-to-BCD encoder converts the decimal number 3 into a BCD 0011.

Solution The top two OR gates have ones as indicated with the red lines. Thus the output is 0111.



Multiplexer

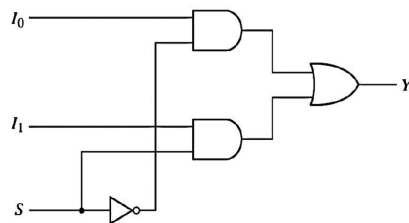
- Is also called **data selectors**.
- A digital multiplexer is a combinational circuit that selects binary information from one of the 2^n input channels and transmits to a single output line.
- The selection of the particular input channel is controlled by a set of select inputs.
- It select binary information from one of many input lines and direct it to a single output line
- 2^n input lines needs n selection lines and one output line



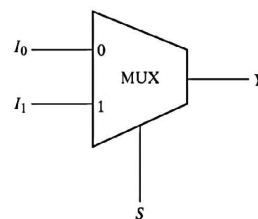
2-to-1-line multiplexer

S	Y
0	I_0
1	I_1

$$Y = \bar{S}I_0 + SI_1$$



(a) Logic diagram

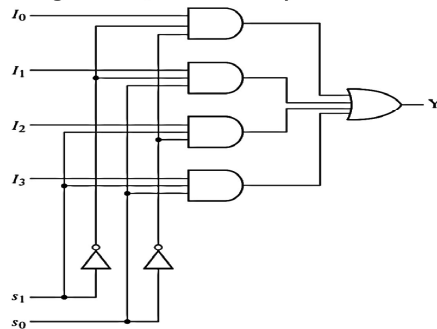


(b) Block diagram

Fig. 4-24 2-to-1-Line Multiplexer

4x1 multiplexer

- Example: a logic circuit, for 4x1 multiplexer



(a) Logic diagram

s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) Function table

$$Y = \overline{s_1} \overline{s_0} I_0 + \overline{s_1} s_0 I_1 + s_1 \overline{s_0} I_2 + s_1 s_0 I_3$$

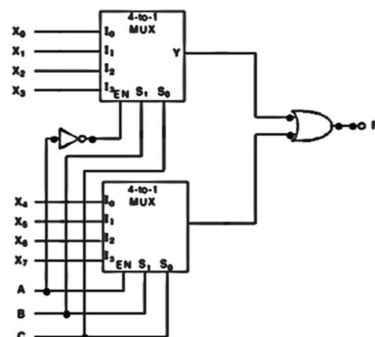
Fig. 4-25 4-to-1-Line Multiplexer

Multiplexer with enable inputs

- As in decoders, multiplexer ICs may have an enable input to control the operation of the unit.
- When the enable input is in a given binary state, the outputs are disabled, and when it is in the other state (the enable state), the circuit functions as a normal multiplexer.
- The enable input (sometimes called strobe) can be used to expand two or more multiplexer ICs to a digital multiplexer with a larger number of inputs.

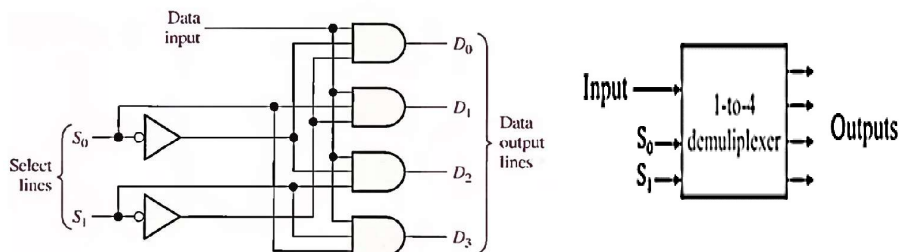
Example:- Construct a 8-to-1 multiplexer with the use of a 4-to-1 line multiplexer and external gate.

Example:- Construct a 16-to-1 multiplexer with the use of only a 4-to-1 line multiplexer.



De-Multiplexer

- A de-multiplexer is a circuit that receives information from a single line and directs it to one of 2^n possible output lines.
- The selection of a specific output is controlled by the bit combination of n elected lines.
- The term “demultiplex” means **one into many**.



Parity Generator and Checker

- Whenever information is transmitted from one device (the transmitter) to another device (the receiver), there is a possibility that errors can occur such that the receiver does not receive the identical information that was sent by the transmitter.
- One of the simplest and most widely used schemes for error detection is the parity method.
- **Parity Bit**:- A parity bit is an extra bit that is attached to a code group that is being transferred from one location to another.
- The parity bit is made either 0 or 1, depending on the number of 1s that are contained in the code group. Two different methods are used.
 - The even-parity method, and
 - The odd-parity method.

Cont...

- In the even-parity method, the value of the parity bit is chosen so that the total number of 1s in the code group (including the parity bit) is an even number.

1 1 0 0 0 0 1 1
 ↑ added parity bit*

- In the odd-parity method, the value of the parity bit is chosen so that the total number of 1s in the code group (including the parity bit) is an odd number.

0 1 0 0 0 0 1 1
 ↑ added parity bit*

cont.

- **Parity generator**:- is a combinational logic system to generate the parity bit at the transmitting side.
- **Parity Checker**:- The message bits with the parity bit are transmitted to their destination, where they are applied to a parity checker circuit.
- The circuit that checks the parity at the receiver side is called the parity checker.
- The parity checker circuit produces a check bit and is very similar to the parity generator circuit.
- If the check bit is 1, then it is assumed that the received data is incorrect. The check bit will be 0 if the received data is correct.

Parity Generator and Checker

