
Digital Systems

Digital Systems

- Basic logic circuit concepts,
- Representation of Numerical Data in Binary Form.
- Combinational logic circuits,
- Synthesis of logic circuits.

Numerical Representation

In science, technology, business, and, in fact, most other fields of endeavor, we are constantly dealing with quantities.

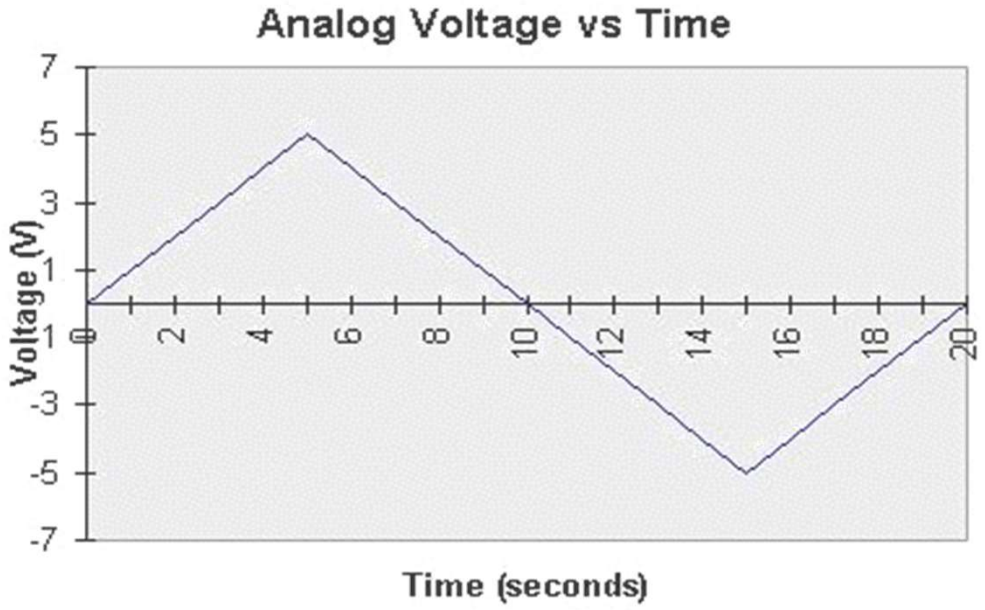
Quantities are measured, monitored, recorded, manipulated arithmetically, observed, or in some other way utilized in most physical systems.

It is important when dealing with various quantities that we be able to represent their values efficiently and accurately.

There are basically two ways of representing the numerical value of quantities: *analog* and *digital*.

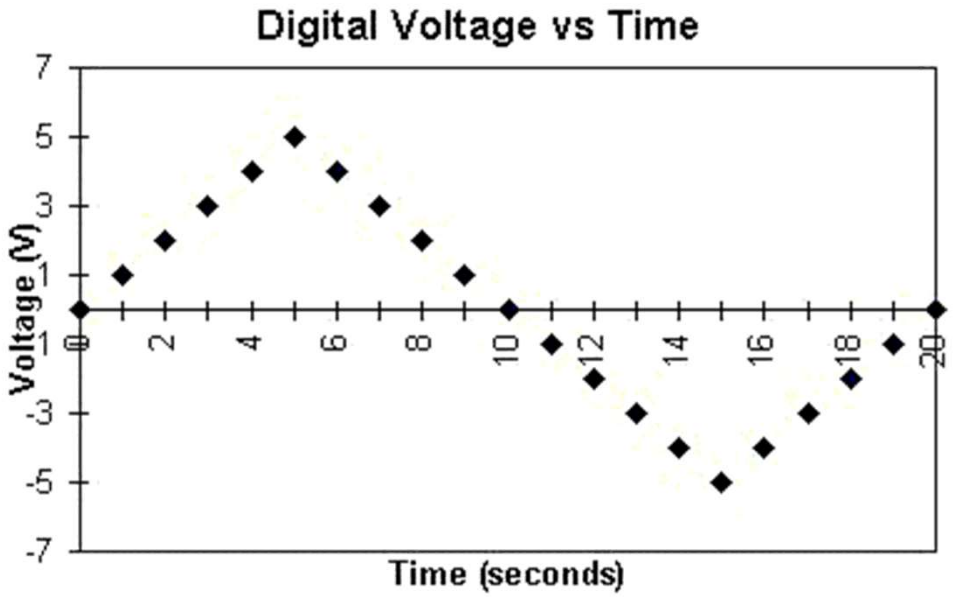
Analog Representation

In analog representation a quantity is represented by a voltage, current, or meter movement that is proportional to the value of that quantity. Analog quantities such as those cited above have an important characteristic: they can vary over a continuous range of values.



Digital Representation

In digital representation the quantities are represented not by proportional quantities but by symbols called **digits**. As we know, the time of day changes continuously, but the digital watch reading does not change continuously; rather, it changes in steps of one per minute (or per second). In other words, this digital representation of the time of day changes in discrete steps, as compared with the representation of time provided by an analog watch, where the dial reading changes continuously.



The major difference between analog and digital quantities,

Analog ➡ continuous

Digital ➡ discrete (step by step)

Advantages

- Easier to design. Exact values of voltage or current are not important, only the range (HIGH or LOW) in which they fall.
- Information storage is easy.
- Accuracy and precision are greater.
- Operation can be programmed. Analog systems can also be programmed, but the variety and complexity of the available operations is severely limited.
- Digital circuits are less affected by noise. As long as the noise is not large enough to prevent us from distinguishing a HIGH from a LOW.
- More digital circuitry can be fabricated on IC chips.

Limitations

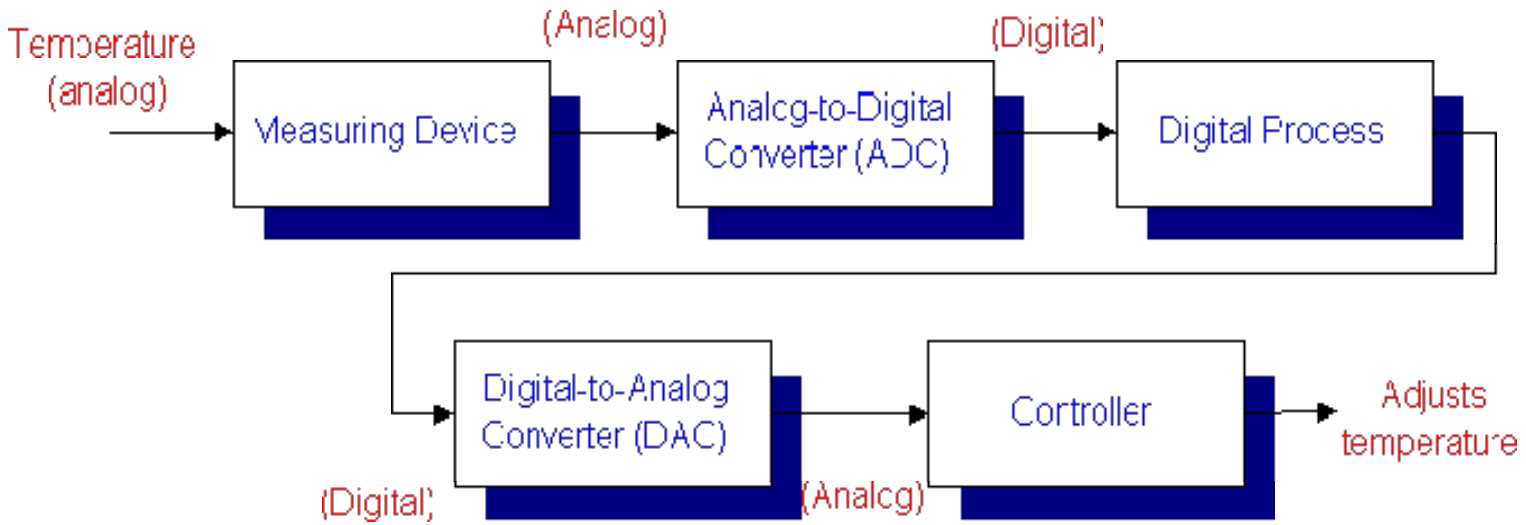
There is really only one major drawback when using digital techniques: **The real world is mainly analog.**

Most physical quantities are analog in nature, and it is these quantities that are often the inputs and outputs that are being monitored, operated on, and controlled by a system.

To take advantage of digital techniques when dealing with analog inputs and outputs, three steps must be followed:

1. Convert the real-world analog inputs to digital form. (ADC)
2. Process (operate on) the digital information.
3. Convert the digital outputs back to real-world analog form. (DAC)

The following diagram shows a temperature control system that requires analog/digital conversions in order to allow the use of digital processing techniques.



Digital Number System

Many number systems are in use in digital technology. The most common are the **decimal, binary, octal,** and **hexadecimal** systems.

Decimal System

Decimal System The decimal system is composed of 10 numerals or symbols. These 10 symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9; using these symbols as digits of a number, we can express any quantity. The decimal system, also called the base-10 system because it has 10 digits.

10^3	10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}
=1000	**=100**	**=10**	**=1**	**.**	**=0.1**	**=0.01**	**=0.001**
Most Significant Digit				**Decimal point**			**Least Significant Digit**

Binary System

In the binary system, there are only two symbols or possible digit values, 0 and 1. This base-2 system can be used to represent any quantity that can be represented in decimal or other number system.

2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}
=8	**=4**	**=2**	**=1**	**.**	**=1/2**	**=1/4**	**=1/8**
Most Significant Bit				**Binary point**			**Least Significant Bit**

Binary Counting

The Binary counting sequence is shown in the table:

$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	Decimal Equivalent
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Binary-To-Decimal Conversion

Any binary number can be converted to its decimal equivalent simply by summing together the weights of the various positions in the binary number which contain a 1.

$$\mathbf{1\ 1\ 0\ 1\ 1_{2}(binary)}$$

$$2^4+2^3+0+2^1+2^0 = 16+8+0+2+1$$

$$= 27_{10}\ (\text{decimal})$$

Decimal-To-Binary Conversion

25_{10} to binary

25/ 2	= 12+ remainder of 1	1 (Least Significant Bit)
12/ 2	= 6 + remainder of 0	0
6 / 2	= 3 + remainder of 0	0
3 / 2	= 1 + remainder of 1	1
1 / 2	= 0 + remainder of 1	1 (Most Significant Bit)
		(Stop when quotient equal to zero)
Result	25_{10} =1 1 0 0 1_2	

The conversion of a decimal *fraction* to binary is accomplished by a method similar to that used for integers. However, multiplication is used instead of division, and integers are accumulated instead of remainders. Again, the method is best explained by example.

	Integer		Fraction	Coefficient
$0.6875 \times 2 =$	1	+	0.3750	$a_{-1} = 1$
$0.3750 \times 2 =$	0	+	0.7500	$a_{-2} = 0$
$0.7500 \times 2 =$	1	+	0.5000	$a_{-3} = 1$
$0.5000 \times 2 =$	1	+	0.0000	$a_{-4} = 1$

Answer: $(0.6875)_{10} = (0.a_{-1}a_{-2}a_{-3}a_{-4})_2 = (0.1011)_2$

Stop when the desired degree of precision has been reached

OCTAL AND HEXADECIMAL NUMBERS

The conversion from and to binary, octal, and hexadecimal plays an important part in digital computers. Since $2^3 = 8$ and $2^4 = 16$, each octal digit corresponds to three binary digits and each hexadecimal digit corresponds to four binary digits. The conversion from binary to octal is easily accomplished by partitioning the binary number into groups of three digits each, starting from the binary point and proceeding to the left and to the right. The corresponding octal digit is then assigned to each group. The following example illustrates the procedure:

$$\underset{2}{\underbrace{10}}\ \underset{6}{\underbrace{110}}\ \underset{1}{\underbrace{001}}\ \underset{5}{\underbrace{101}}\ \underset{3}{\underbrace{011}}\ .\ \underset{7}{\underbrace{111}}\ \underset{4}{\underbrace{100}}\ \underset{0}{\underbrace{000}}\ \underset{6}{\underbrace{110}})_2 = (26153.7460)_8$$

Conversion from binary to hexadecimal is similar, except that the binary number is divided into groups of four digits:

$$\underset{2}{\underbrace{10}}\ \underset{C}{\underbrace{1100}}\ \underset{6}{\underbrace{0110}}\ \underset{B}{\underbrace{1011}}\ .\ \underset{F}{\underbrace{1111}}\ \underset{2}{\underbrace{0010}})_2 = (2C6B.F2)_{16}$$

Boolean Variables & Truth Tables

Boolean algebra differs in a major way from ordinary algebra in that Boolean constants and variables are allowed to have only two possible values, **0 or 1**.

Boolean 0 and 1 do not represent actual numbers but instead represent the state of a voltage variable, or what is called its logic level.

Some common representation of 0 and 1 is shown in the following table.

Logic 0	**Logic 1**
False	True
Off	On
Low	High
No	Yes
Open Switch	Close Switch

In Boolean algebra, there are three basic logic operations: OR, AND and NOT.

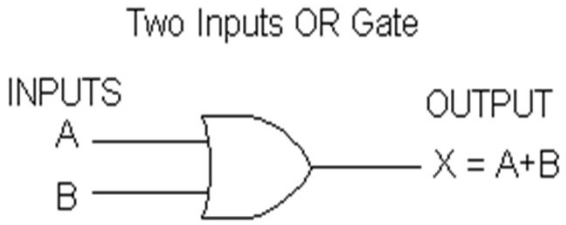
These logic gates are digital circuits constructed from diodes, transistors, and resistors connected in such a way that the circuit output is the result of a basic logic operation (OR, AND, NOT) performed on the inputs.

Truth Table

A truth table is describing how a logic circuit's output depends on the logic levels present at the circuit's inputs.

A	B	X = A+B
0	0	C
0	1	1
1	0	1
1	1	1

OR Operation



A	B	X = A+B
0	0	0
0	1	1
1	0	1
1	1	1

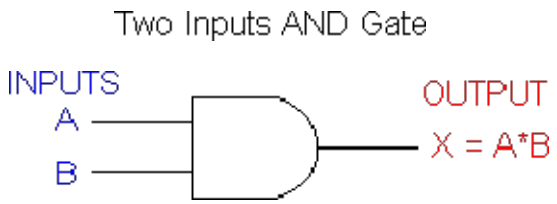
The expression X = A + B reads as "X equals A **OR** B".

The + sign stands for the OR operation, not for ordinary addition.

The OR operation produces a result of 1 when **any** of the input variable is 1.

The OR operation produces a result of 0 only when **all** the input variables are 0

AND Operation



A	B	X = A*B
0	0	0
0	1	0
1	0	0
1	1	1

The expression X = A * B reads as "X equals A **AND** B".

The multiplication sign stands for the AND operation, same for ordinary multiplication of 1s and 0s.

The AND operation produces a result of 1 occurs only for the single case when **all** of the input variables are 1.

The output is 0 for any case where one or more inputs are 0

NOT Operation

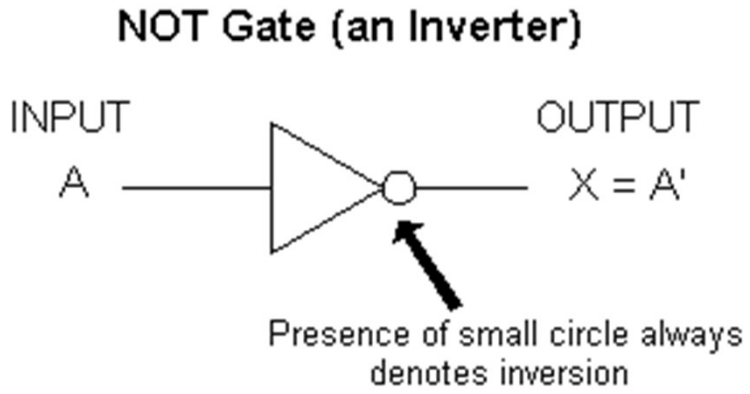
The NOT operation is unlike the OR and AND operations in that it can be performed on a **single** input variable. For example, if the variable A is subjected to the NOT operation, the result x can be expressed as x = A‘ or Ā

where the prime (') represents the NOT operation. This expression is read as:

x equals NOT A

x equals the inverse of A

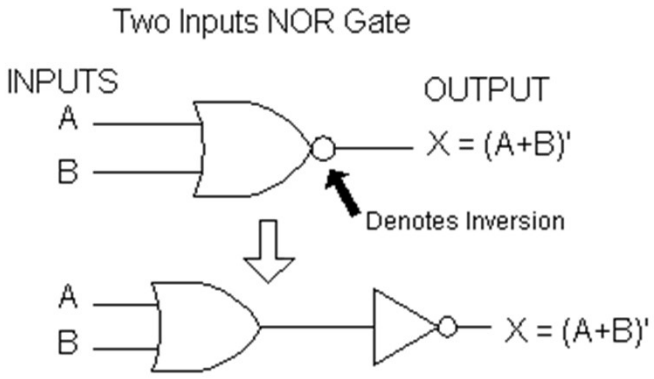
x equals the complement of A



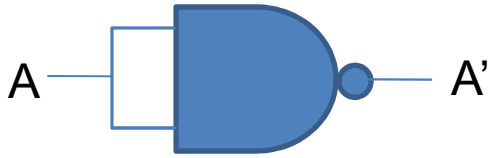
A	X = A’
0	1
1	0

Universal Building Blocks

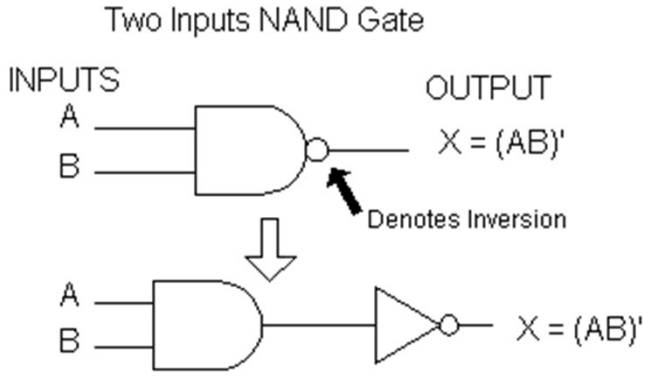
NOR Operation



INPUTS		OR	NOR
A	B	X = A+B	X= (A+B)`
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0



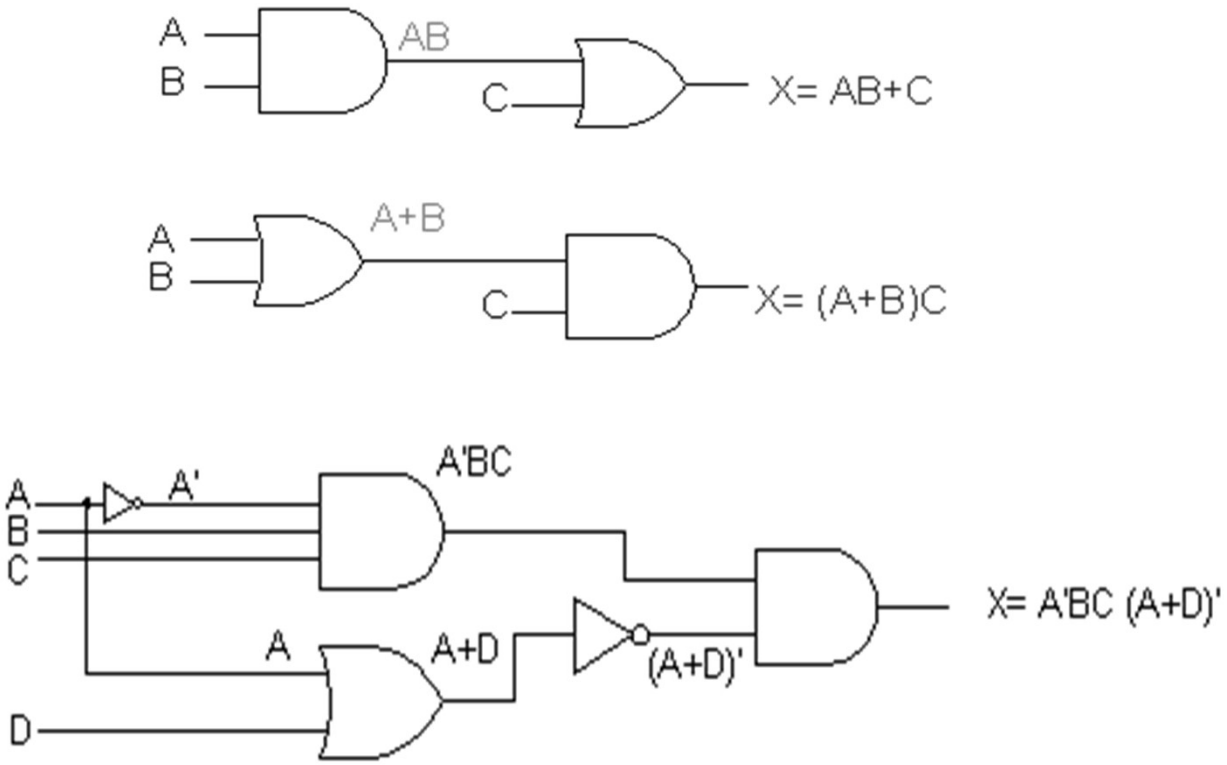
NAND Operation

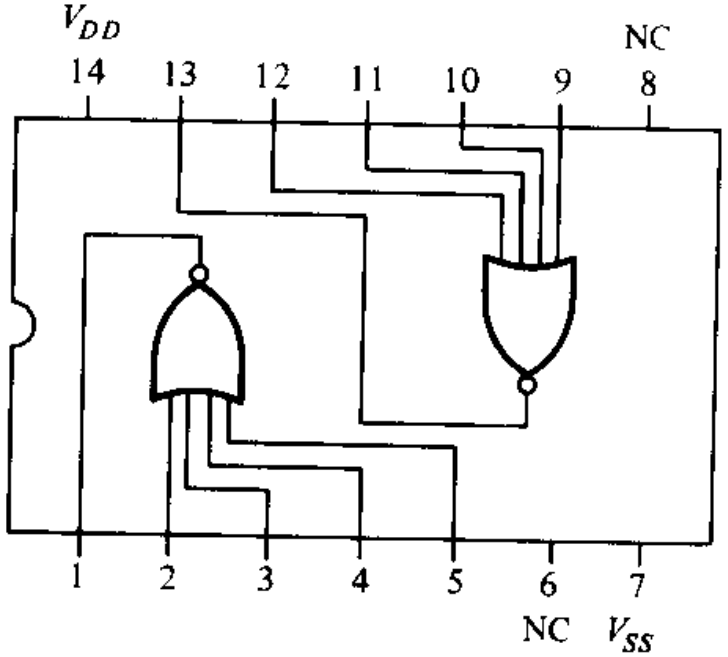


INPUTS		AND	NAND
A	B	X = AB	X= (AB)`
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

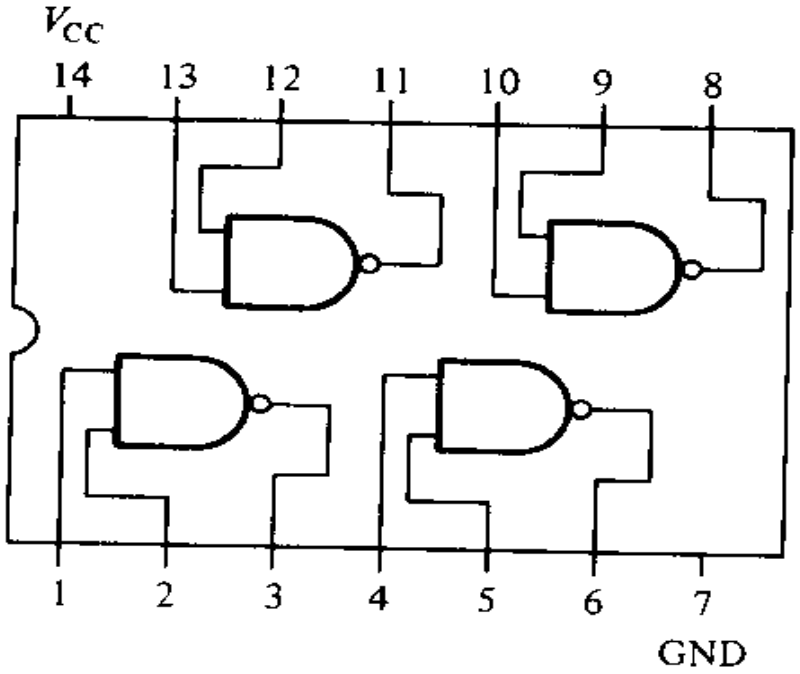
Describing Logic Circuits Algebraically

Any logic circuit, no matter how complex, may be completely described using the Boolean operations, because the OR gate, AND gate, and NOT circuit are the basic building blocks of digital systems.





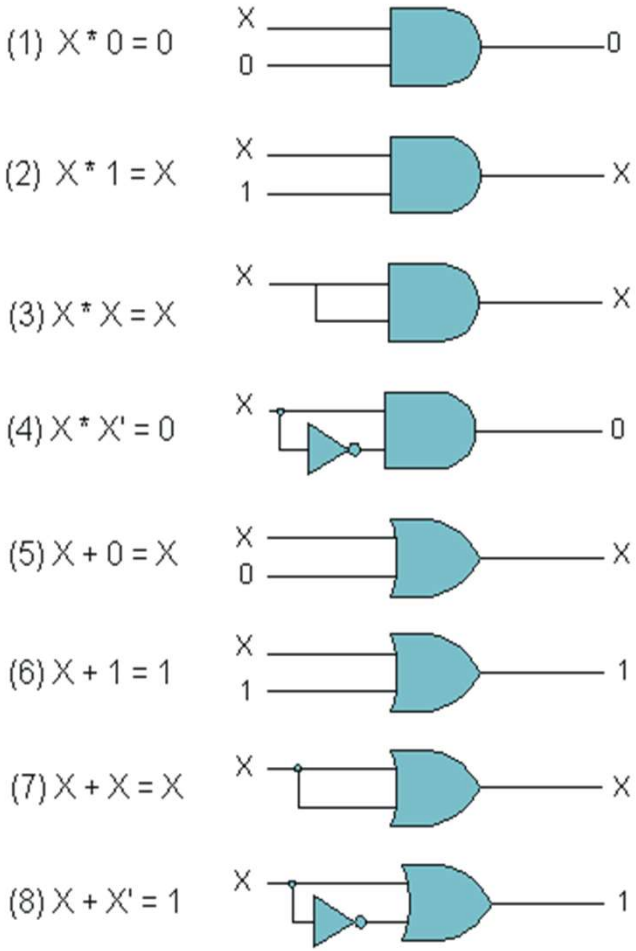
4002—Dual 4-input NOR gates.



7400—Quadruple 2-input NAND gates

Boolean Theorems

Investigating the various Boolean theorems (rules) can help us to simplify logic expressions and logic circuits.



(9) x + y = y + x *(commutative law)*

(10) x * y = y * x *(commutative law)*

(11) x+ (y+z) = (x+y) +z = x+y+z *(associative law)*

(12) x (yz) = (xy) z = xyz *(associative law)*

(13a) x (y+z) = xy + xz

(13b) (w+x)(y+z) = wy + xy + wz + xz

(14) x + xy = x [x(1+y)=x*1=x]

(15) x + x'y = x + y

DeMorgan's Theorem

DeMorgan's theorems are **extremely** useful in simplifying expressions in which a product or sum of variables is inverted. The two theorems are:

(16) $\overline{(x+y)} = \overline{x} * \overline{y}$

(17) $\overline{(x*y)} = \overline{x} + \overline{y}$

"Break the Line, Change the Sign."

Theorem (16) says that when the OR sum of two variables is inverted, this is the same as inverting each variable individually and then ANDing these inverted variables.

Theorem (17) says that when the AND product of two variables is inverted, this is the same as inverting each variable individually and then ORing them.

Example:

$$X = [(A'+C) * (B+D')]'$$

$$= (A'+C)' + (B+D')' \textit{ [by theorem (17)]}$$

A’’=A

$$= (A''*C') + (B‘*D'') \textit{ [by theorem (16)]}$$

$$= AC' + B'D$$

For practice, prove that

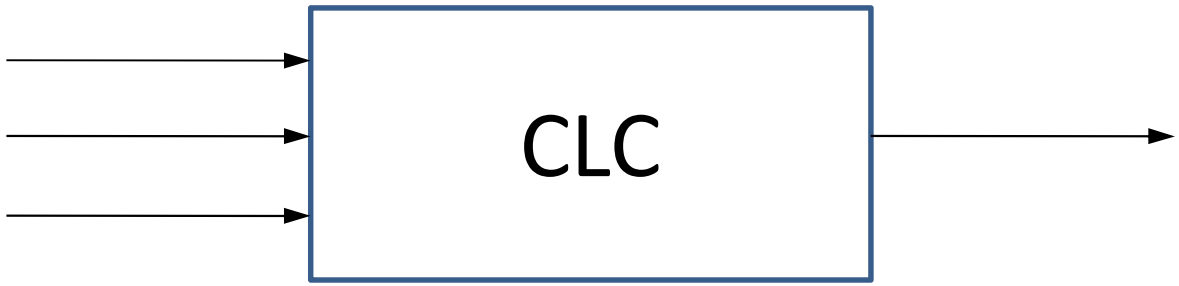
(a) [(AB)’+A’+AB]’ = 0

(b) AB+(AC)’+AB’C(AB+C) = 1

(c) [(AB’+ABC)’+A(B+AB’)]’ = 0

Combinational Logic Circuits

Is the logic circuit where the output always depends on the inputs irrespective of the previous state with out the feed back and memory.



Min Term:

ANDed product of literals in which each variable appears exactly once, in true or complemented form.

Canonical Forms

One of the most powerful theorems within Boolean algebra states that any Boolean function can be expressed as the Sum Of Products (SOP) of all the variables with in the system .

$$A+B = A(1)+B(1)$$
$$= A(B+B')+B(A+A')$$
$$= AB+AB'+AB+A'B \quad \text{Min Terms}$$

A	B	C	Minterms
0	0	0	$\overline{A}\,\overline{B}\,\overline{C} = m_0$
0	0	1	$\overline{A}\,\overline{B}\,C = m_1$
0	1	0	$\overline{A}\,B\,\overline{C} = m_2$
0	1	1	$\overline{A}\,B\,C = m_3$
1	0	0	$A\,\overline{B}\,\overline{C} = m_4$
1	0	1	$A\,\overline{B}\,C = m_5$
1	1	0	$A\,B\,\overline{C} = m_6$
1	1	1	$A\,B\,C = m_7$

F in canonical form:

$F(A,B,C) = \Sigma m(3,4,5,6,7)$

$= m3 + m4 + m5 + m6 + m7$

$= A' B C+A B' C'+A B' C+A B C'+A B C$

Product of Sums (POS)

ORed sum of literals in which each variable appears exactly once in either true or complemented form, but not both.

A	B	C	Maxterms
0	0	0	$A+B+\overline{C}=M_0$
0	0	1	$A+\overline{B}+C=M_1$
0	1	0	$A+\overline{B}+\overline{C}=M_2$
0	1	1	$\overline{A}+B+C=M_3$
1	0	0	$\overline{A}+B+\overline{C}=M_4$
1	0	1	$\overline{A}+\overline{B}+C=M_5$
1	1	0	$\overline{A}+\overline{B}+\overline{C}=M_6$
1	1	1	$\overline{A}+\overline{B}+\overline{C}=M_7$

$F(A,B,C) = \Pi\, M(0,1,2)$

$= (A + B + C)\,(A + B + C')\,(A + B' + C)$

Maxterms

Finding the Minterms

A. Write down all the terms

B. Put X's where letters must be provided to convert the term to a min term.

C. Use all combinations of the X's in each term to generate minterms, where an X is a 0, write barred letter; where it is 1, write an unbarred letter.

D. Drop out redundant terms.

Example: *Find the minterms for A+BC*

=A+BC

=AXX +XBC (A00,A01,A10,A11+0BC,1BC)

AXX= AB'C', AB'C, ABC', ABC

XBC=A'BC, ABC

= AB'C'+ AB'C+ ABC'+ ABC+A'BC+ ABC

= AB'C'+ AB'C+ ABC'+ ABC+A'BC

Minterm Designation

Example: find the minterm designation of AB'C'D'

Copy the term	AB'C'D'
Substitute 1 for non barred And 0 for barred letters	1 0 0 0
Express as decimal subscript of m	m_8

Therefore, AB'C'D'= m_8

So the answer of the last Example can be written as,

AB'C'+ AB'C+ ABC'+ ABC+A'BC = Σ m(4,5,6,7,3)

Minimization Technique

Logic Minimization: reduce complexity of the gate level implementation

- reduce number of literals (gate inputs)
- reduce number of gates
- reduce number of levels of gates

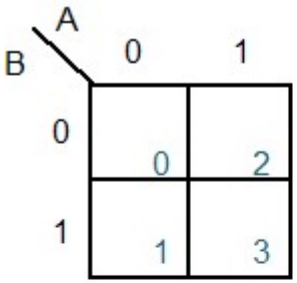
❑KARNAUGH MAP (K-Map)

❑TABULATION METHOD
(Quine-McCluskey Method)

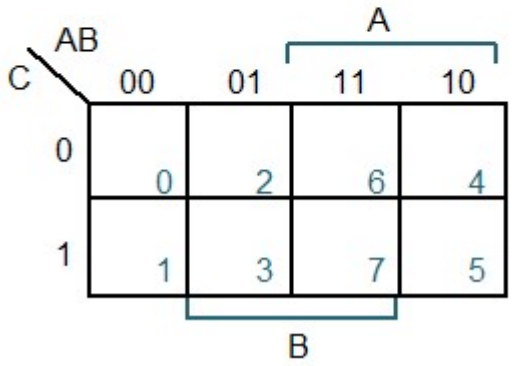
Karnaugh Map Method

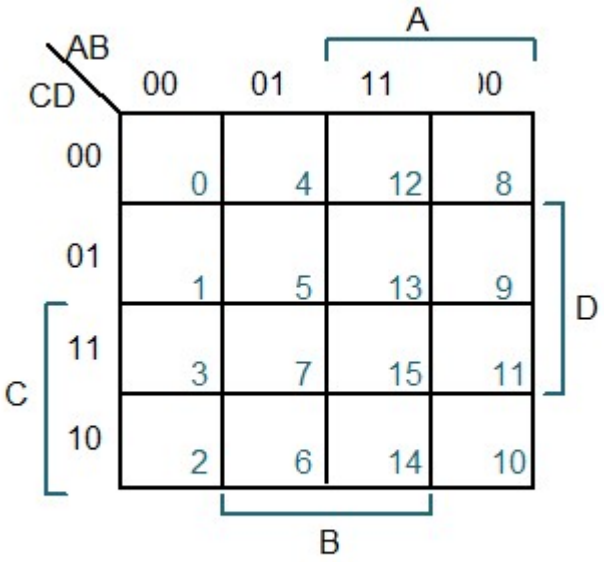
K-map is a method of representing the truth table that helps visualize adjacencies in up to 6 dimensions

2-variable K-map

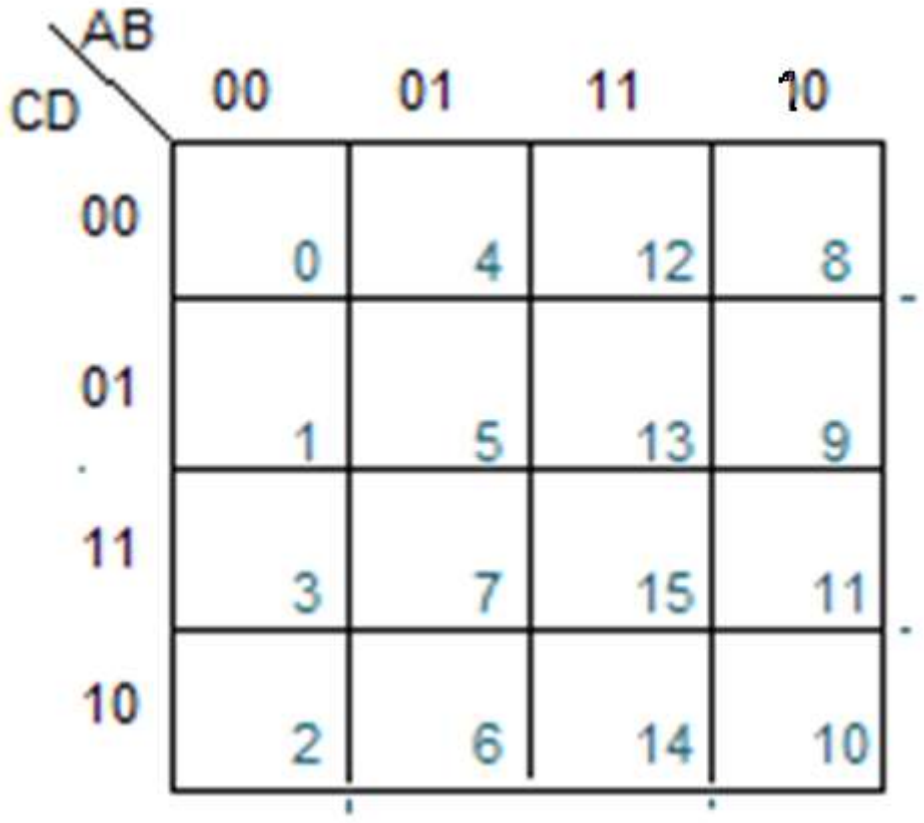


3-variable K-map



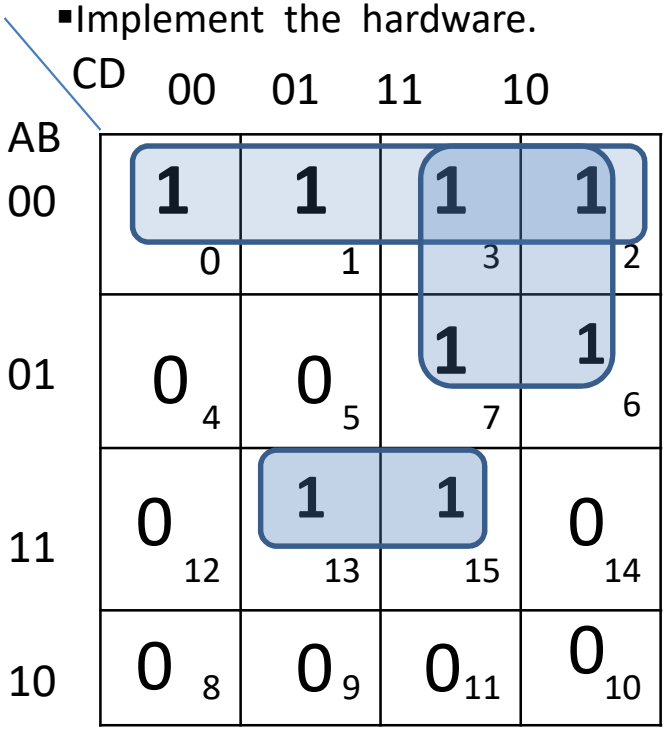


4-variable K-map


AB
CD
00
01
11
10
00
01
11
10
0
4
12
8
1
5
13
9
3
7
15
11
2
6
14
10

Example: Reduce the expression F = Σm(0,1,2,3,6,7,13,15) by mapping and implement in **NAND** logic.

- Enter **1** for given minterms in corresponding location and **0** for others.
- Group the maximum number of 1s in the order of 2^n.
- If a set of **1**s said to be group they must be the **gray** numbers able to form a cycle.
- Ensure that all the 1s must come under at least once in a group.
- Simplify the resultant minterms using basic laws and OR the Result.
- Implement the hardware.



Group I = A'B'C'D' A'B'C'D A'B'CD A'B'CD'
= A'B'

Group II = A'BCD A'BCD' A'B'CD A'B'CD'
= A'C

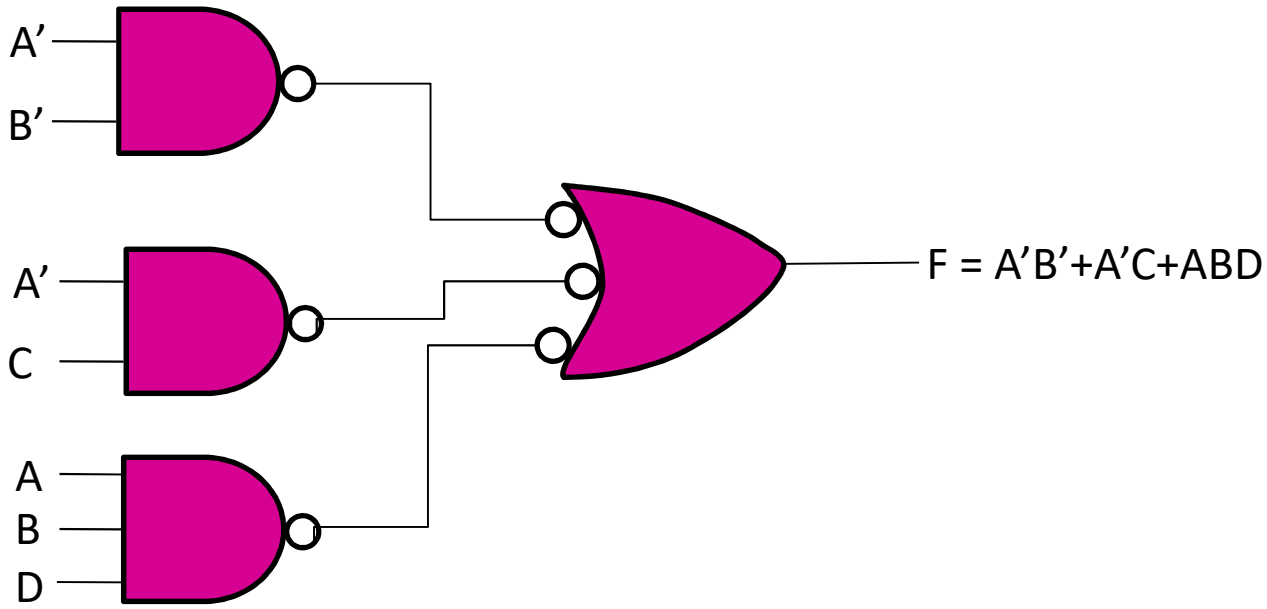
Group III = ABC'D ABCD
= ABD

F = A'B'+A'C+ABD

Examples

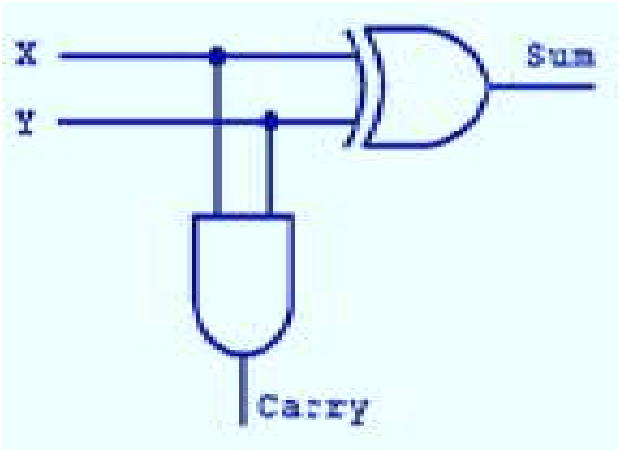
- Draw the simplified logical circuit for the given expression, and use Karnaugh map for reducing the logic. F = A'BC'D' + ABC'D' + A'BC'D + ABCD + A'BCD'+ A'B'CD'
- Applying Boolean algebra simplify: AB+A(B+C)+B(B+C) and draw its logic diagram.
- Using K-map simplify the expression given below and draw its logic diagram: AB'C+A'BC+A'B'C+A'B'C'+AB'C'
- Express the following function in sum of minterms, simplify the Boolean function using K – map and realize the output using NAND gate. F = A B C + B C D + A B C D + A B C
- A logic circuit has inputs A, B, C and D. The output of the circuit is given by E = ∑ m(1,3,4,5,7,10,12,13). Find the minimum sum of product for E.

NAND Implementation



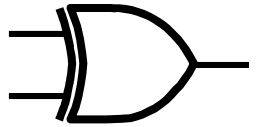
Half Adder

Inputs		Outputs	
X	Y	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



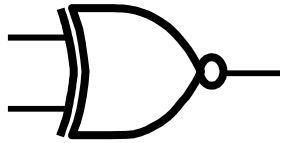
Sum = X'Y+XY'
= X⊕Y

Carry = XY



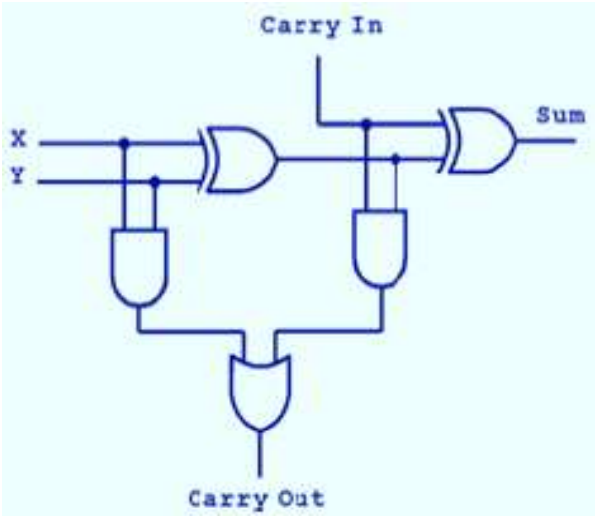
$$X \oplus Y = X\,\overline{Y} + \overline{X}\,Y \quad \text{XOR}$$

$$\overline{X \oplus Y} = X\,Y + \overline{X}\,\overline{Y} \quad \text{XNOR}$$



Full Adder

Inputs			Outputs	
X	Y	Carry In	Sum	Carry Out
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



SUM= $\bar{x}\bar{y}Cin + \bar{x}y\overline{Cin} + x\bar{y}\overline{Cin} + xyCin$

$\overline{Cin}(\bar{x}y + x\bar{y}) + Cin(\overline{\bar{x}\bar{y} + xy})$

$\overline{Cin}(x \oplus y) + Cin(\overline{x \oplus y})$

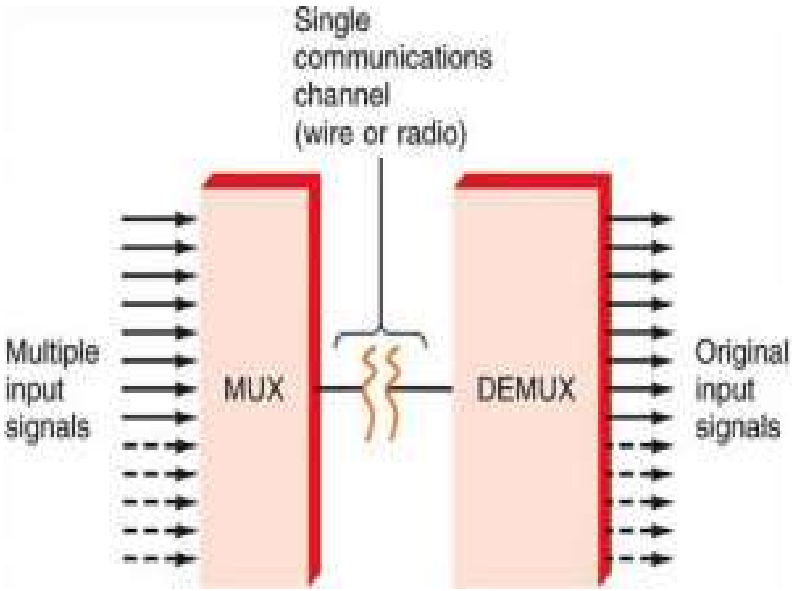
$(x \oplus y) \oplus Cin$

Carry Out= $x\bar{y}Cin + xyCin + \bar{x}yCin + xy\overline{Cin}$

$Cin(x\bar{y} + \bar{x}y) + xy\,(Cin + \overline{Cin})$

$Cin\,(x \oplus y) + xy$

Multiplexers / Data Selectors



- 2^n data inputs, n control inputs (called "select line"), 1 output
- Used to connect 2^n points to a single point
- Control signal pattern forms binary index of input connected to output

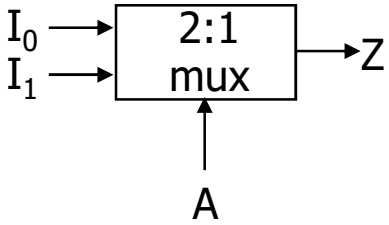
2:1 mux: $Z = A' I_0 + A I_1$

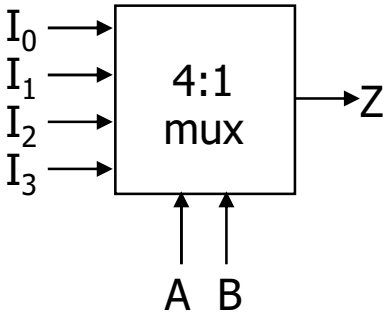
4:1 mux: $Z = A' B' I_0 + A' B I_1 + A B' I_2 + A B I_3$

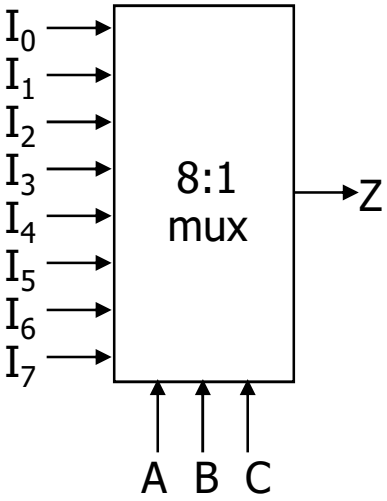
8:1 mux: $Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$

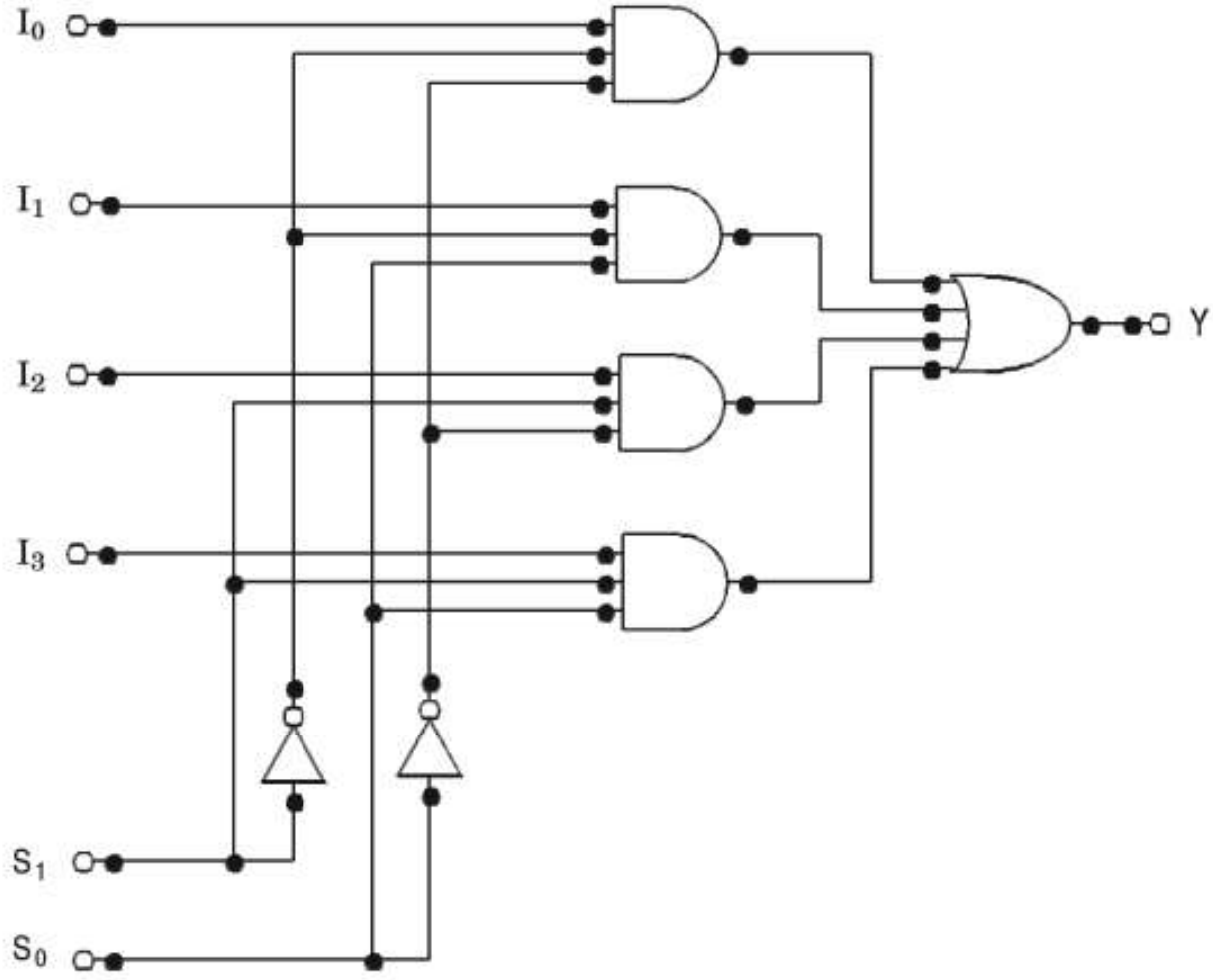
In general, $Z = \Sigma(m_k I_k)$

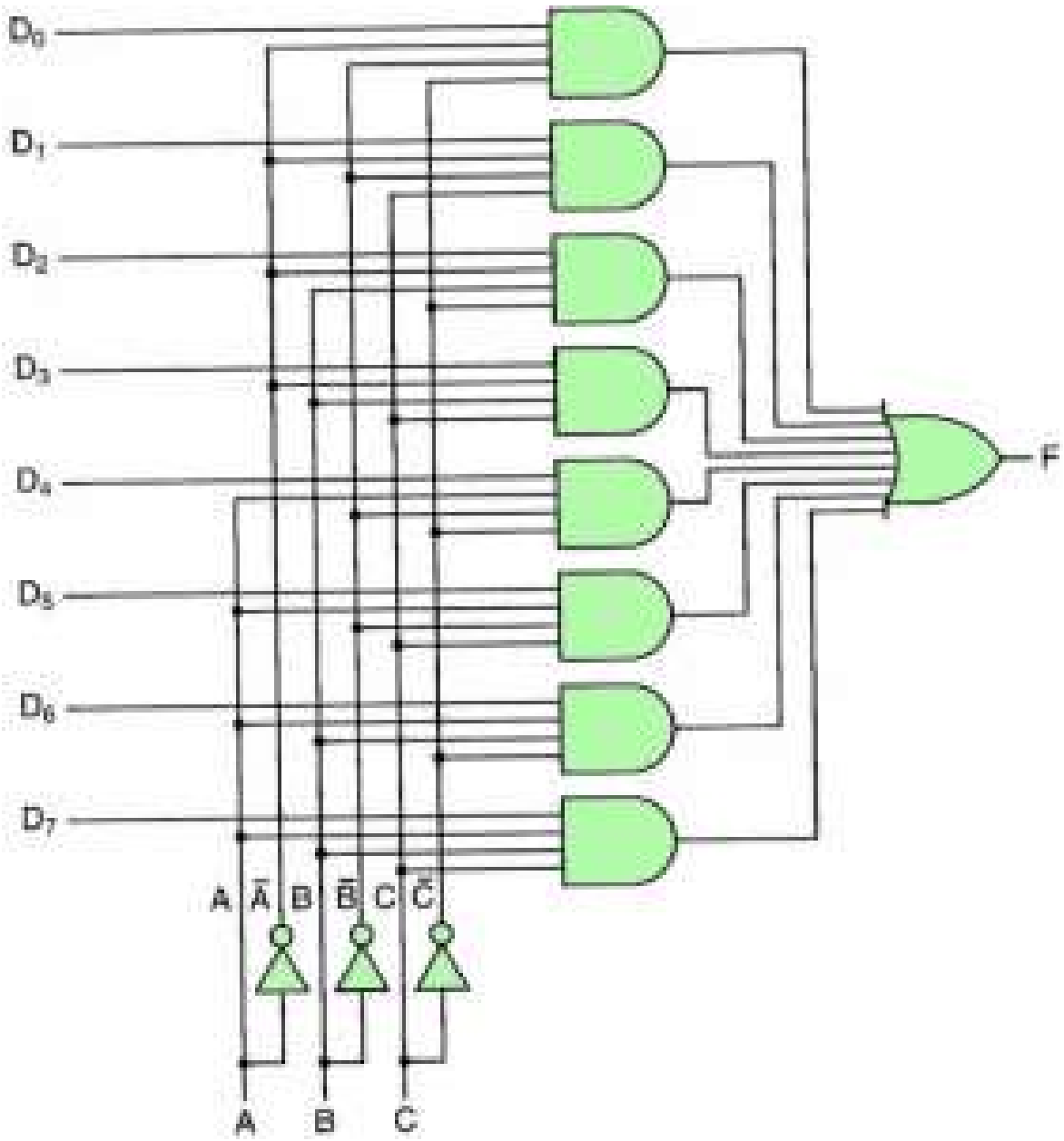
- 2-1 multiplexer (1select line)
- 4-1 multiplexer (2 select lines)
- 8-1 multiplexer (3 select lines)
- 16-1 multiplexer (4 select lines)



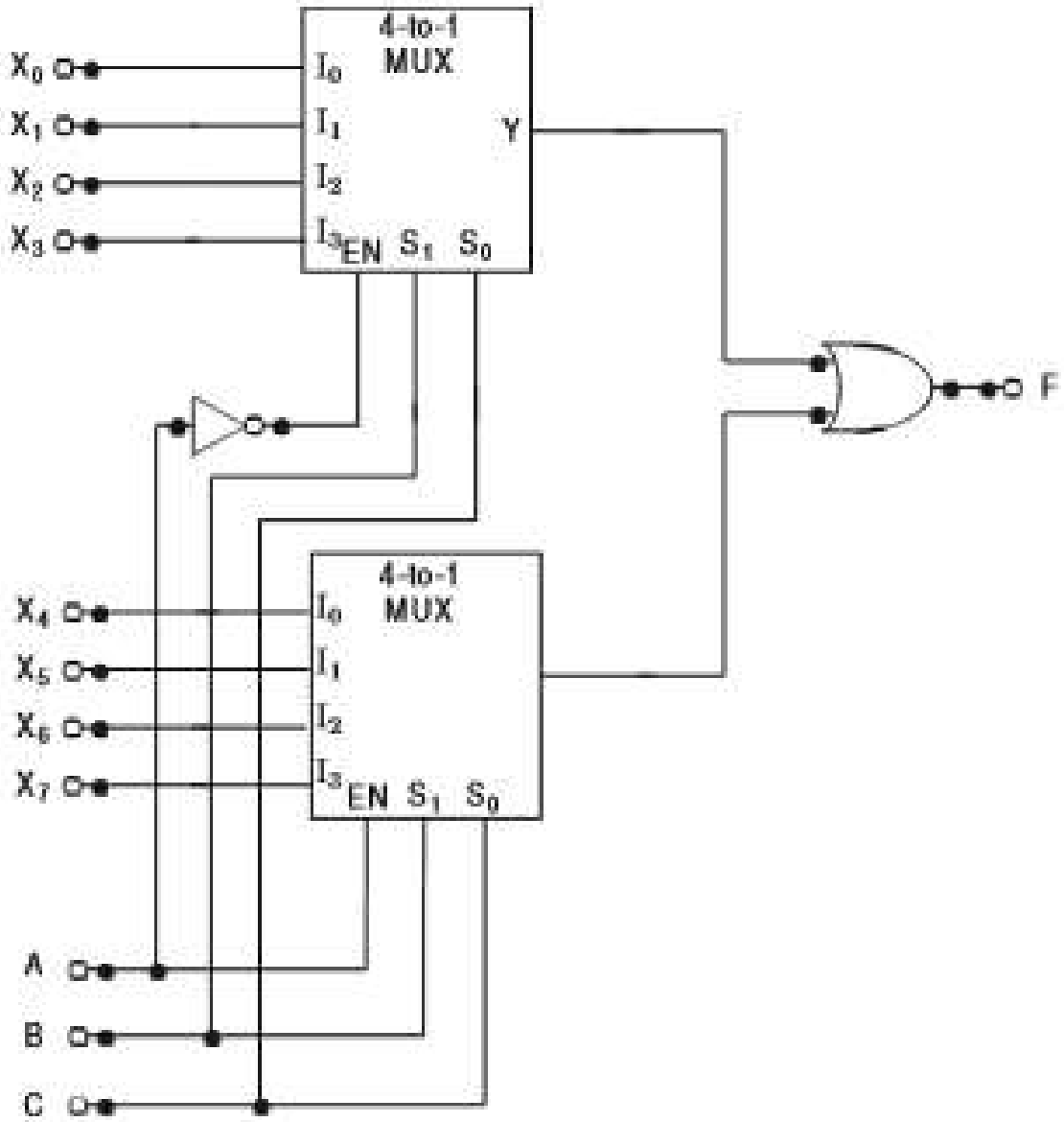


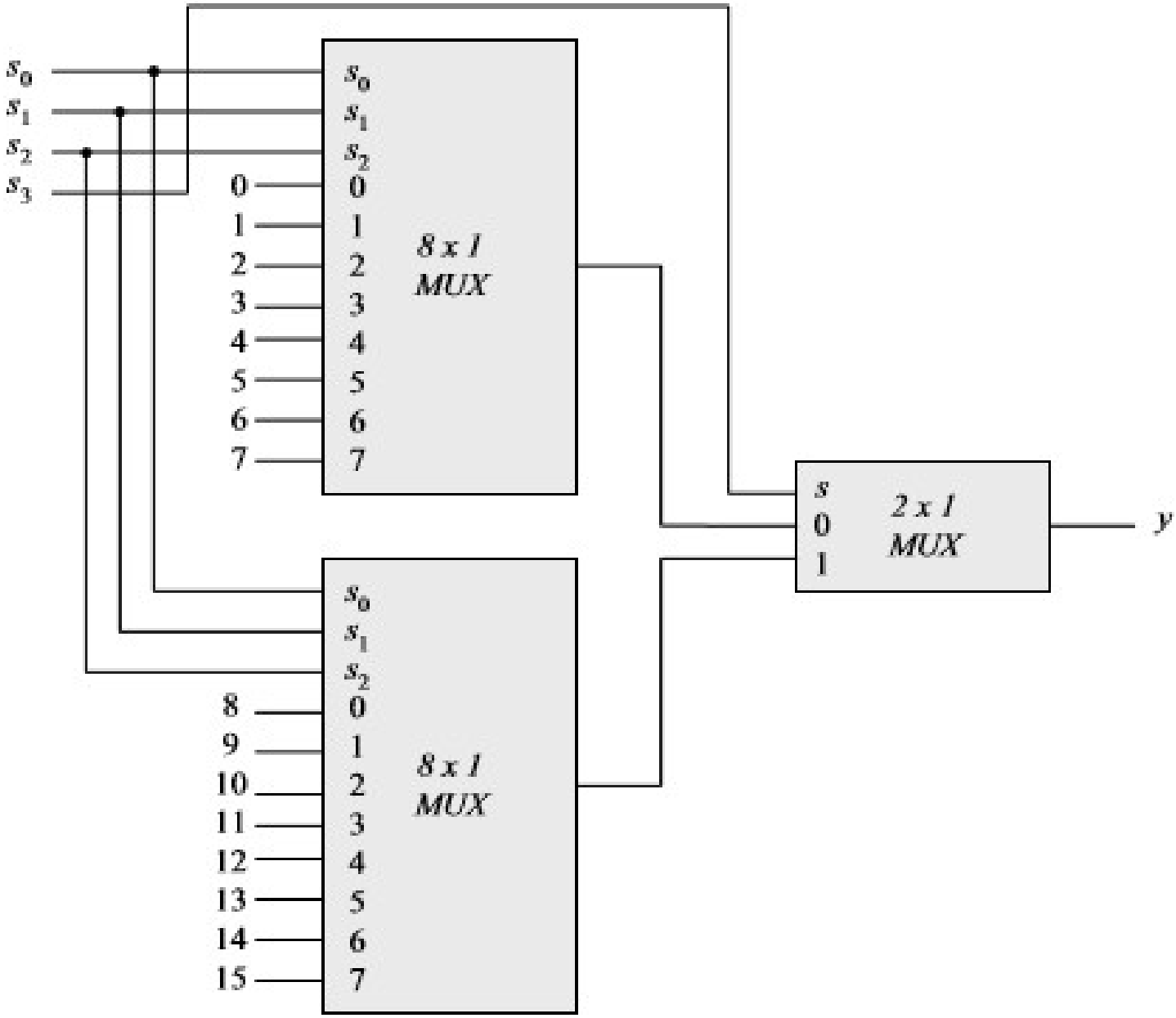



I0
I1
I2
I3
S1
S0
Y


D0
D1
D2
D3
D4
D5
D6
D7
F
A
Ā
B
B̄
C
C̄
A
B
C

Strobe method




s_0
s_1
s_2
s_3
s_0
s_1
s_2
0
1
2
3
4
5
6
7
0
1
2
3
4
5
6
7
8 x 1
MUX
s_0
s_1
s_2
8
9
10
11
12
13
14
15
0
1
2
3
4
5
6
7
8 x 1
MUX
s
0
1
2 x 1
MUX
y

Applications of Multiplexer:

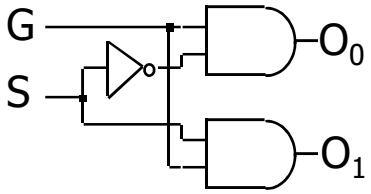
Multiplexer are used in various fields where multiple data need to be transmitted using a single line. Following are some of the applications of multiplexers –

- **Communication system** – Communication system is a set of system that enable communication like transmission system, relay and tributary station, and communication network. The efficiency of communication system can be increased considerably using multiplexer. Multiplexer allow the process of transmitting different type of data such as audio, video at the same time using a single transmission line.
- **Telephone network –** In telephone network, multiple audio signals are integrated on a single line for transmission with the help of multiplexers. In this way, multiple audio signals can be isolated and eventually, the desire audio signals reach the intended recipients.
- **Computer memory** – Multiplexers are used to implement huge amount of memory into the computer, at the same time reduces the number of copper lines required to connect the memory to other parts of the computer circuit.
- **Transmission from the computer system of a satellite** – Multiplexer can be used for the transmission of data signals from the computer system of a satellite or spacecraft to the ground system using the GPS (Global Positioning System) satellites.

Demultiplexers

- Single data input, n control inputs, 2^n outputs
- Control inputs (called "selects" (S)) represent binary index of output to which the input is connected
- Data input usually called "enable" (G)

1:2 Demultiplexers



1:2 Demultiplexers

$O_0 = G \bullet S'$

$O_1 = G \bullet S$

1:4 Demultiplexers

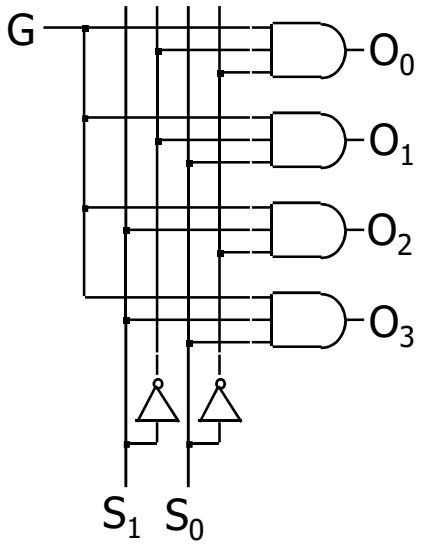
$O_0 = G \bullet S_1' \bullet S_0'$

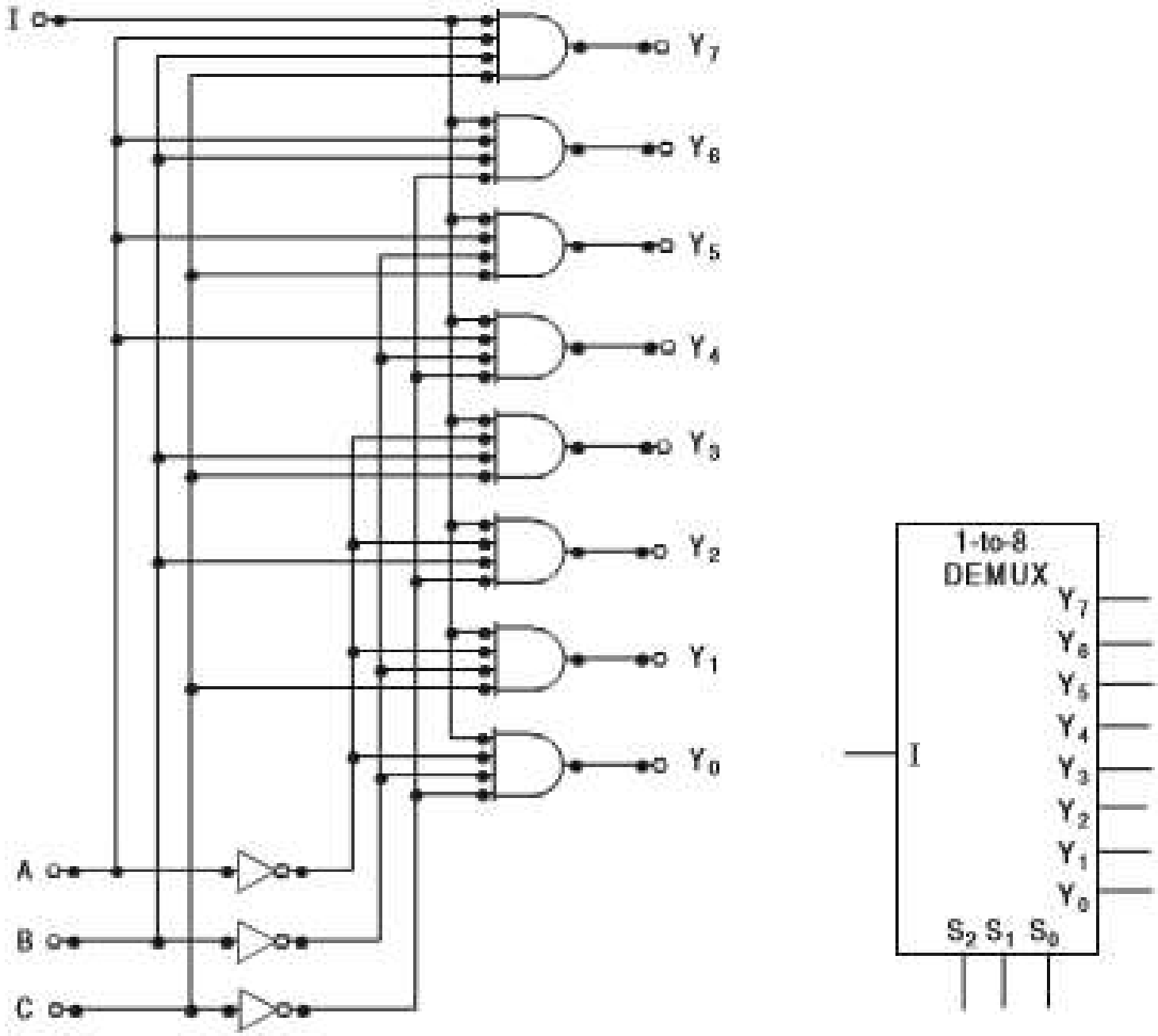
$O_1 = G \bullet S_1' \bullet S_0$

$O_2 = G \bullet S_1 \bullet S_0'$

$O_3 = G \bullet S_1 \bullet S_0$

1:4 Demultiplexers




I
Y7
Y6
Y5
Y4
Y3
Y2
Y1
Y0
A
B
C
1-to-8
DEMUX
I
Y7
Y6
Y5
Y4
Y3
Y2
Y1
Y0
S2 S1 S0

Selection Inputs			*Outputs*							
A	*B*	*C*	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	$Y_0 = I$	0	0	0	0	0	0	0
0	0	1	0	$Y_1 = I$	0	0	0	0	0	0
0	1	0	0	0	$Y_2 = I$	0	0	0	0	0
0	1	1	0	0	0	$Y_3 = I$	0	0	0	0
1	0	0	0	0	0	0	$Y_4 = I$	0	0	0
1	0	1	0	0	0	0	0	$Y_5 = I$	0	0
1	1	0	0	0	0	0	0	0	$Y_6 = I$	0
1	1	1	0	0	0	0	0	0	0	$Y_7 = I$

Applications of Demultiplexer:

Demultiplexer is used to connect a single source to multiple destinations. The main application area of demultiplexer is communication system where multiplexer are used. Most of the communication system are bidirectional i.e. they function in both ways (transmitting and receiving signals). Hence, for most of the applications, the multiplexer and demultiplexer work in sync. Demultiplexer are also used for reconstruction of parallel data and ALU circuits.

- **Communication System** – Communication system use multiplexer to carry multiple data like audio, video and other form of data using a single line for transmission. This process make the transmission easier. The demultiplexer receive the output signals of the multiplexer and converts them back to the original form of the data at the receiving end. The multiplexer and demultiplexer work together to carry out the process of transmission and reception of data in communication system.
- **ALU (Arithmetic Logic Unit)** – In an ALU circuit, the output of ALU can be stored in multiple registers or storage units with the help of demultiplexer. The output of ALU is fed as the data input to the demultiplexer. Each output of demultiplexer is connected to multiple register which can be stored in the registers.
- **Serial to parallel converter** – A serial to parallel converter is used for reconstructing parallel data from incoming serial data stream. In this technique, serial data from the incoming serial data stream is given as data input to the demultiplexer at the regular intervals. A counter is attach to the control input of the demultiplexer. This counter directs the data signal to the output of the demultiplexer where these data signals are stored. When all data signals have been stored, the output of the demultiplexer can be retrieved and read out in parallel.