Digital Logic Design 1

Pham Quoc Cuong
HCMC University of Technology
http://www.cse.hcmut.edu.vn/~cuongpham

Introduction

- Basic logic gate functions will be combined in *combinational* logic circuits.
- Simplification of logic circuits will be done using Boolean algebra and a mapping technique.
- Troubleshooting of combinational circuits will be introduced.
- PLD structures will be explained.

Sum-of-Products & Product-of-sums Forms

- A Sum-of-products (SOP) expression will appear as two or more AND terms ORed together.

$$ABC + \overline{A}B\overline{C}$$

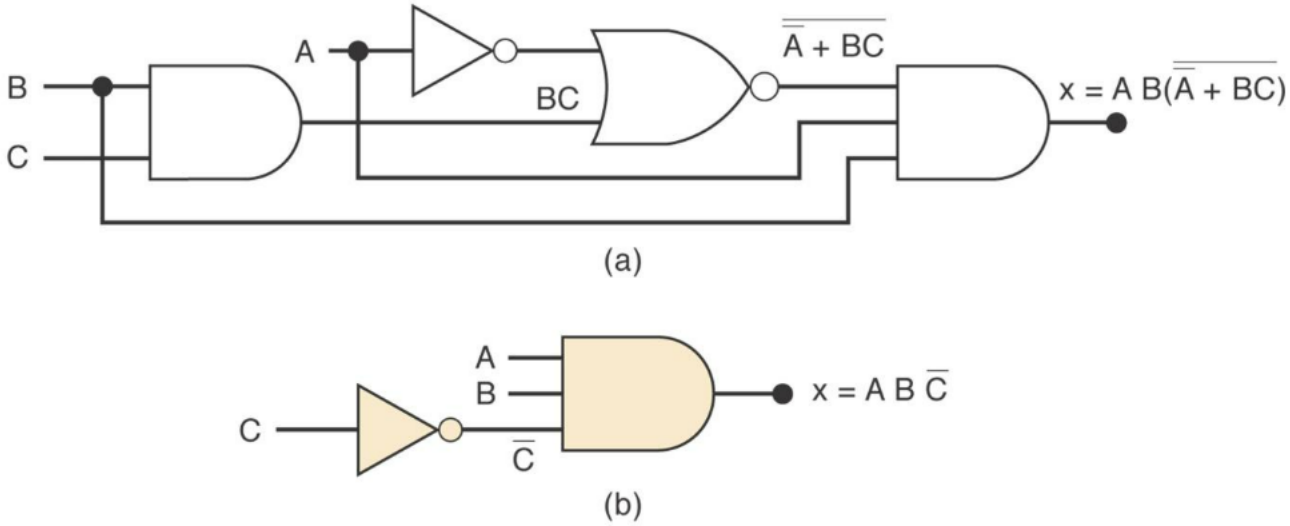
$$AB + \overline{A}B\overline{C} + \overline{C}\overline{D} + D$$

- A Product-of-sums(POS) expression is sometimes used in logic design.

$$(A + B + C)(\overline{A} + B + \overline{C})$$

Simplifying Logic Circuits

- The circuits below both provide the same output, but the lower one is clearly less complex.



- We will study simplifying logic circuits using Boolean algebra and Karnaugh mapping

Algebraic Simplification

- Place the expression in SOP form by applying DeMorgan's theorems and multiplying terms.
- Check the SOP form for common factors and perform factoring where possible.
- Note that this process may involve some trial and error to obtain the simplest result.

Designing Combinational Logic Circuits

- To solve any logic design problem:
  - Interpret the problem and set up its truth table.
  - Write the AND (product) term for each case where the output equals 1.
  - Combine the terms in SOP form.
  - Simplify the output expression if possible.
  - Implement the circuit for the final, simplified expression.

Example of Logic Design

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

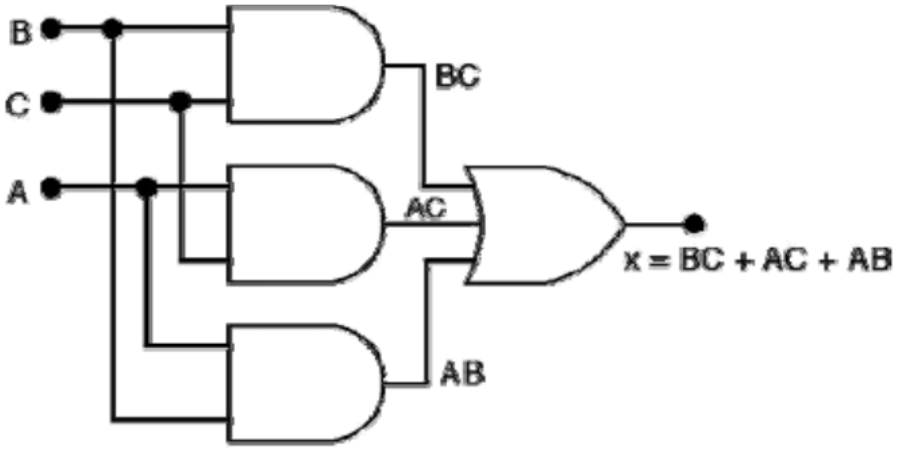
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

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

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

X = BC (A'+ A) + AC(B'+ B) + AB(C' + C)

X = BC + AC + AB



Example of Logic Design

4-1. Simplify the following expressions using Boolean algebra.

(a) $x = ABC + \overline{A}C$

(b) $y = (Q + R)(\overline{Q} + \overline{R})$

(c) $w = ABC + A\overline{B}C + \overline{A}$

(d) $q = \overline{RST}\,(\overline{R + S + T})$

(e) $x = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}BC + ABC + A\overline{B}\,\overline{C} + A\overline{B}C$

(f) $z = (B + \overline{C})(\overline{B} + C) + \overline{\overline{A} + B + \overline{C}}$

(g) $y = \overline{(C + D)} + \overline{A}C\overline{D} + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}CD + AC\overline{D}$

(h) $x = AB\,(\overline{\overline{C}D}) + \overline{A}BD + \overline{B}\,\overline{C}\,\overline{D}$

Karnaugh Map Method

- A graphical method of simplifying logic equations or truth tables. Also called a K map.
- Theoretically can be used for any number of input variables, but practically limited to 5 or 6 variables.

Karnaugh Map Method

- The truth table values are placed in the K map.
- Adjacent K map square differ in only one variable both horizontally and vertically.
- The pattern from top to bottom and left to right must be in the form $\overline{A}\,\overline{B}, \overline{A}B, AB, A\overline{B}$
- A SOP expression can be obtained by ORing all squares that contain a 1.

Karnaugh Map Method

- Looping adjacent groups of 2, 4, or 8 1s will result in further simplification.
- When the largest possible groups have been looped, only the common terms are placed in the final expression.
- Looping may also be wrapped between top, bottom, and sides.

Karnaugh Map for 2, 3 variables

- Looping adjacent groups of 2, 4, or 8 1s will result in further simplification.

A	B	X
0	0	1 → $\bar{A}\bar{B}$
0	1	0
1	0	0
1	1	1 → AB

$$x = \bar{A}\bar{B} + AB$$

	$\bar{B}$	B
$\bar{A}$	1	0
A	0	1

A	B	C	X
0	0	0	1 → $\bar{A}\bar{B}\bar{C}$
0	0	1	1 → $\bar{A}\bar{B}C$
0	1	0	1 → $\bar{A}B\bar{C}$
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1 → $AB\bar{C}$
1	1	1	0

$$X = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + AB\bar{C}$$

	$\bar{C}$	C
$\bar{A}\bar{B}$	1	1
$\bar{A}B$	1	0
AB	1	0
$A\bar{B}$	0	0

Karnaugh Map for 4 variables

- Looping adjacent groups of 2, 4, or 8 1s will result in further simplification.

A	B	C	D		X	
0	0	0	0		0	
0	0	0	1		1	$\rightarrow \bar{A}\bar{B}\bar{C}D$
0	0	1	0		0	
0	0	1	1		0	
0	1	0	0		0	
0	1	0	1		1	$\rightarrow \bar{A}B\bar{C}D$
0	1	1	0		0	
0	1	1	1		0	
1	0	0	0		0	
1	0	0	1		0	
1	0	1	0		0	
1	0	1	1		0	
1	1	0	0		0	
1	1	0	1		1	$\rightarrow AB\bar{C}D$
1	1	1	0		0	
1	1	1	1		1	$\rightarrow ABCD$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	0	0
$\bar{A}B$	0	1	0	0
AB	0	1	1	0
$A\bar{B}$	0	0	0	0

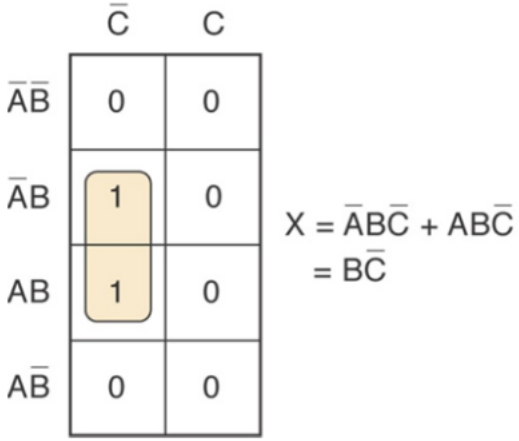
$$\left\{ \begin{array}{l} X = \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D \\ \quad + AB\bar{C}D + ABCD \end{array} \right\}$$

Minimization Technique

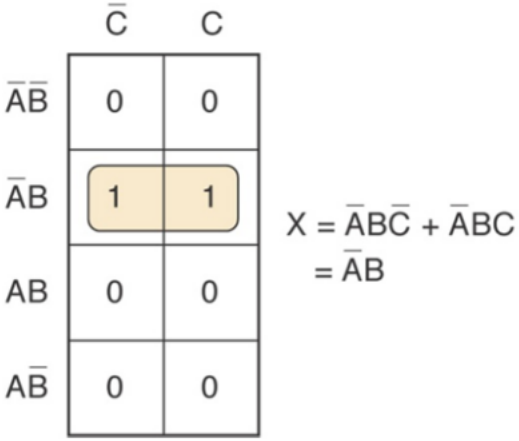
- Minimization is done by spotting patterns of 1's and 0's
- Simple theorems are then used to simplify the Boolean description of the patterns
- Pairs of adjacent 1's
  - remember that adjacent squares differ by only one variable
  - hence the combination of 2 adjacent squares has the form
  - $P\,(A + A')$
  - this can be simplified (from before) to just P

A\BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

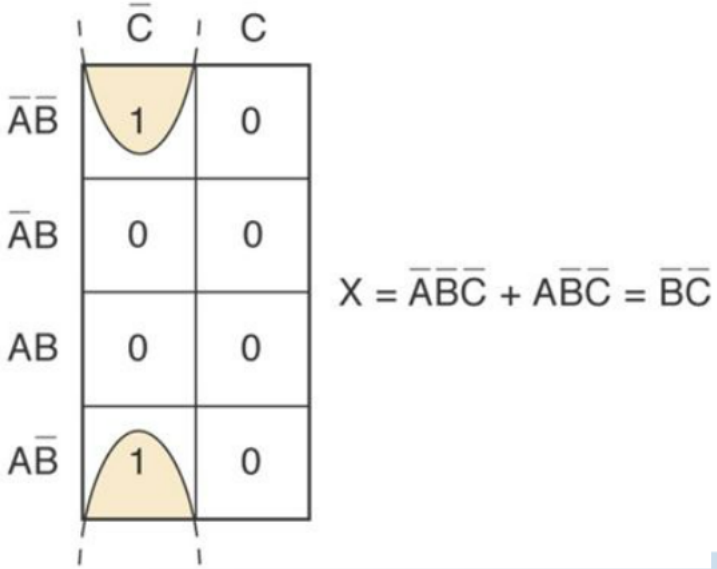
Example of pairs of adjacent of 1s



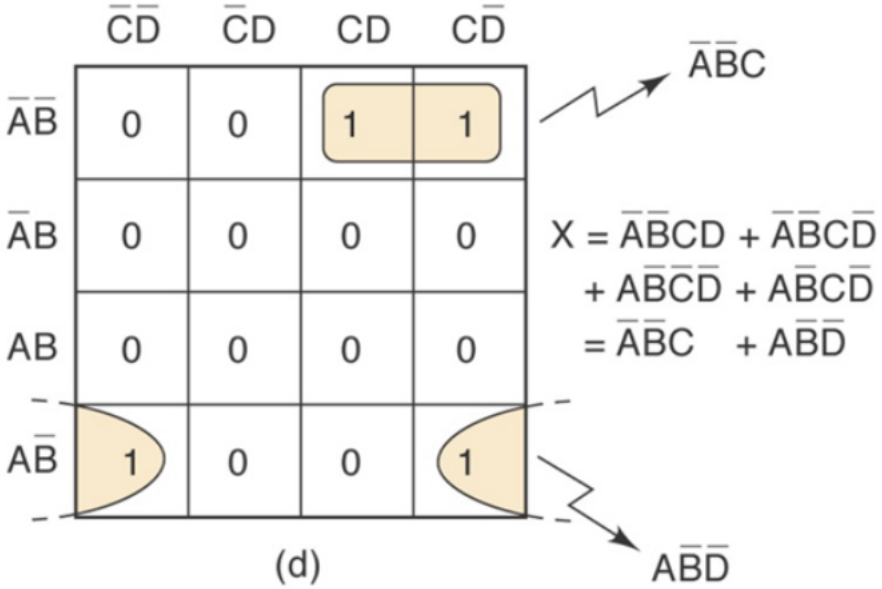
$X = \bar{A}B\bar{C} + AB\bar{C}$
$= B\bar{C}$



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



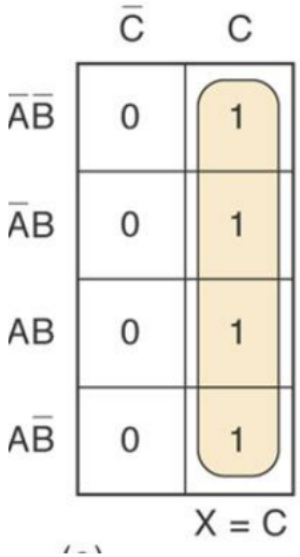
$X = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} = \bar{B}\bar{C}$

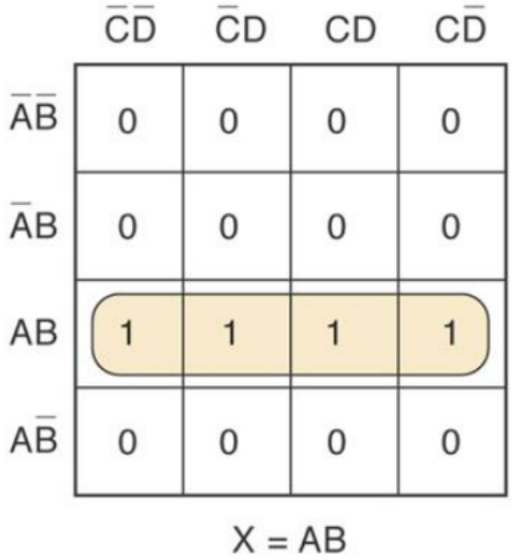


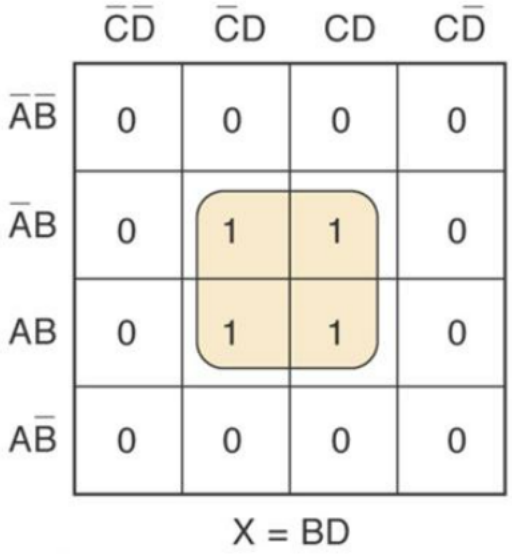
$X = \bar{A}\bar{B}CD + \bar{A}\bar{B}C\bar{D}$
$+ \bar{A}B\bar{C}D + \bar{A}BC\bar{D}$
$= \bar{A}\bar{B}C + \bar{A}B\bar{D}$

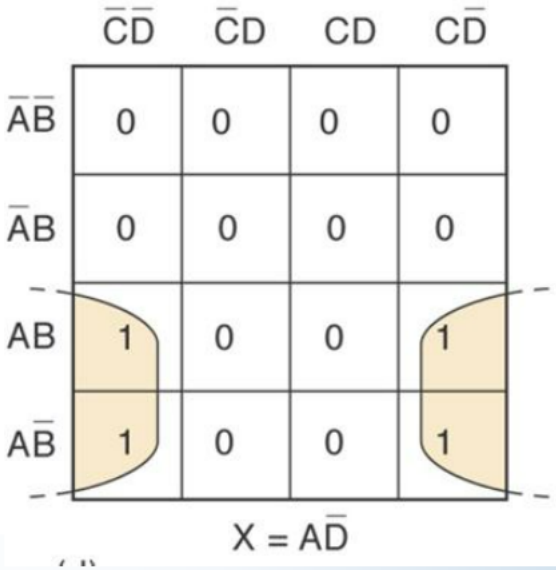
(d)

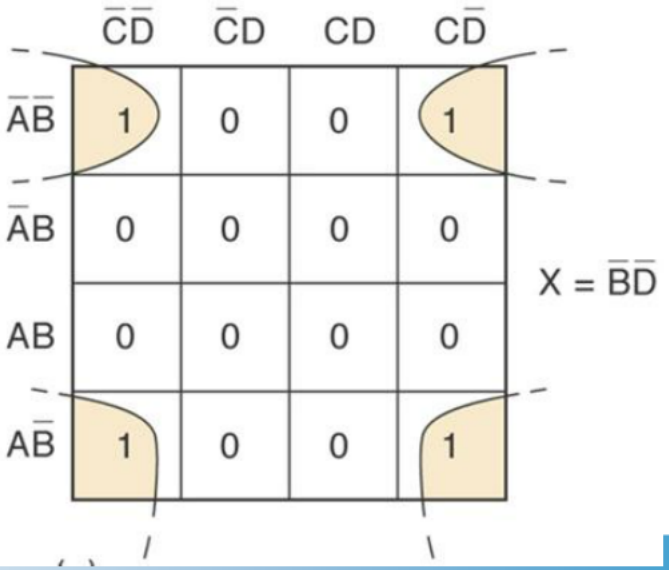
Example of grouping of fours 1s (quads)



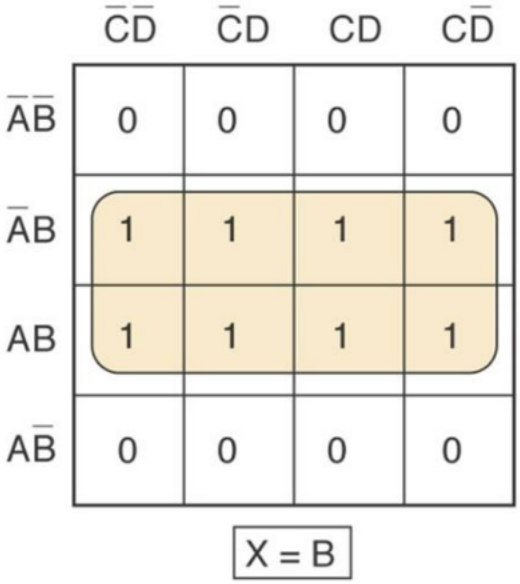


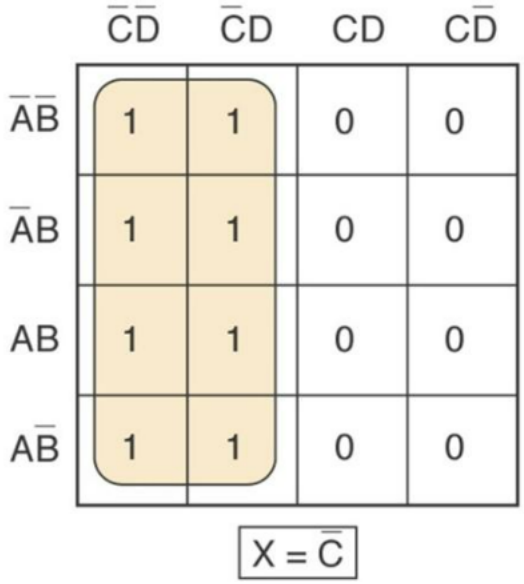


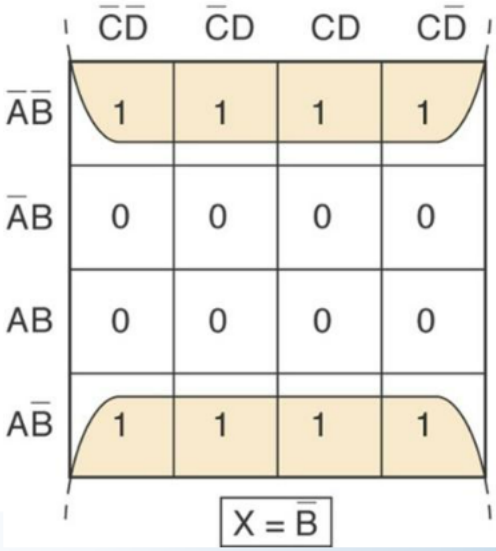


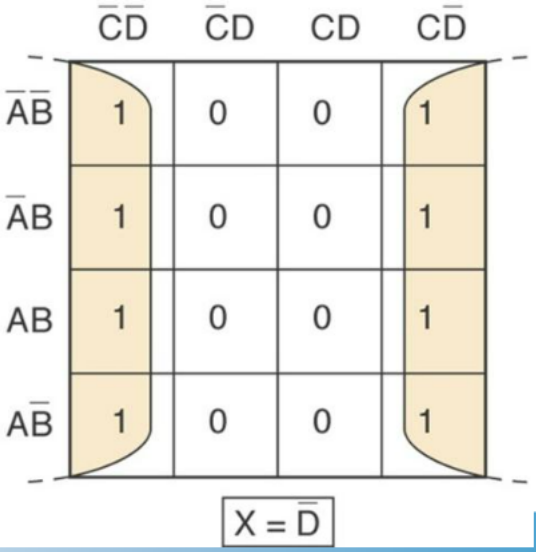


Example of grouping of eight 1s (octals)





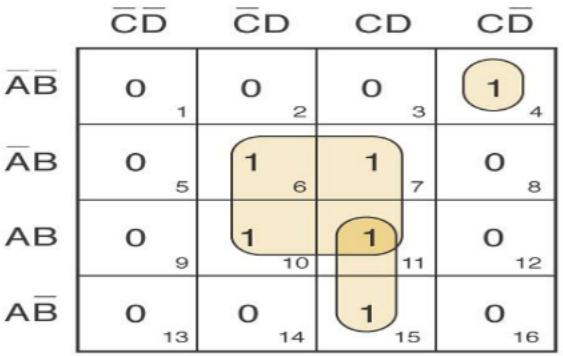




Complete Simplification Process

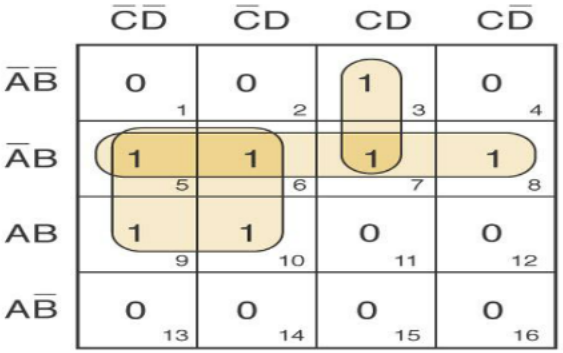
- Complete K map simplification process:
  - Construct the K map, place 1s as indicated in the truth table.
  - Loop 1s that are not adjacent to any other 1s.
  - Loop 1s that are in pairs
  - Loop 1s in octets even if they have already been looped.
  - Loop quads that have one or more 1s not already looped.
  - Loop any pairs necessary to include 1st not already looped.
  - Form the OR sum of terms generated by each loop.

Example



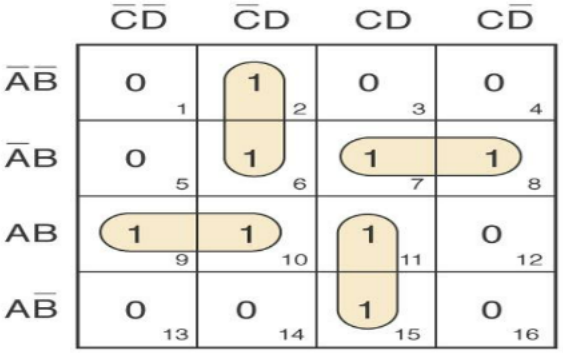
$$X = \underbrace{\bar{A}\bar{B}C\bar{D}}_{\text{loop 4}} + \underbrace{ACD}_{\text{loop 11, 15}} + \underbrace{BD}_{\text{loop 6, 7, 10, 11}}$$

(a)



$$X = \underbrace{\bar{A}B}_{\text{loop 5, 6, 7, 8}} + \underbrace{B\bar{C}}_{\text{loop 5, 6, 9, 10}} + \underbrace{\bar{A}CD}_{\text{loop 3,7}}$$

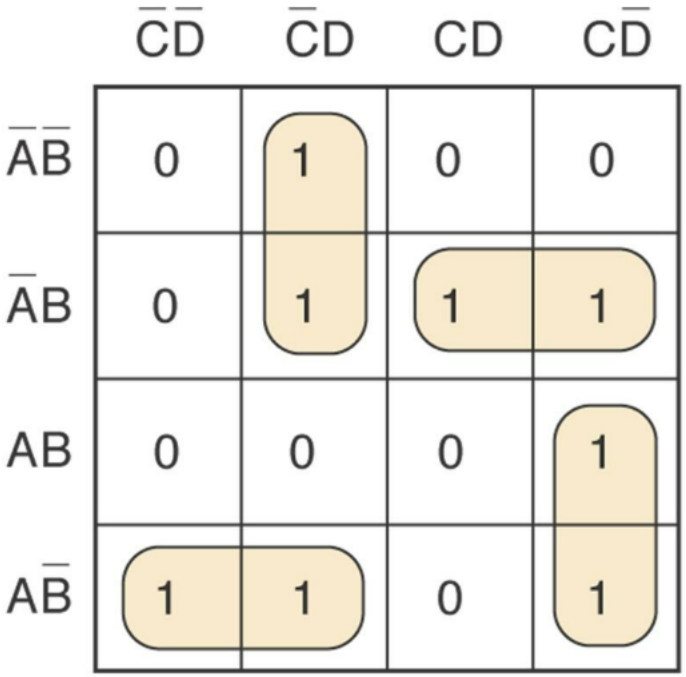
(b)



$$X = \underbrace{AB\bar{C}}_{9, 10} + \underbrace{\bar{A}\bar{C}D}_{2, 6} + \underbrace{\bar{A}BC}_{7, 8} + \underbrace{ACD}_{11, 15}$$

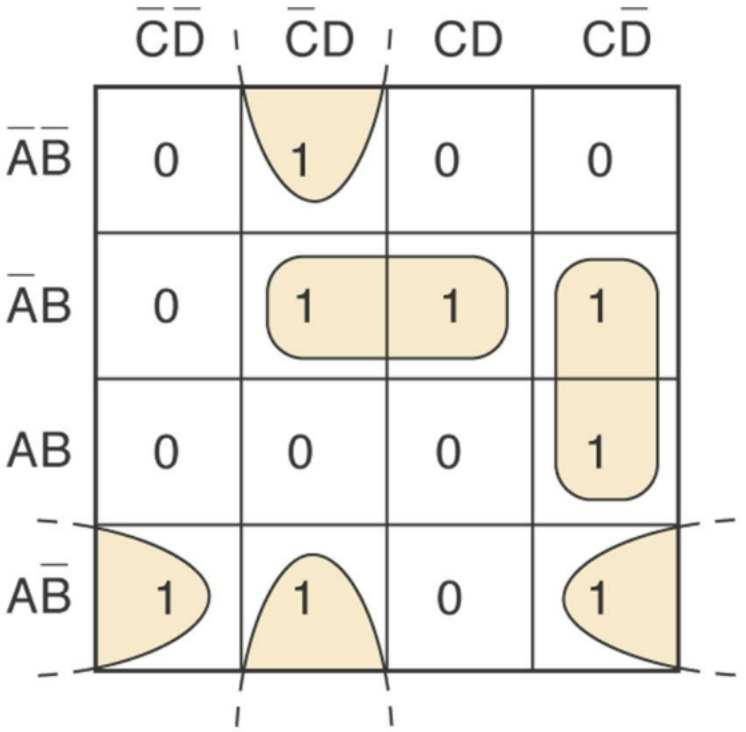
(c)

Example



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

(a)



$$X = \bar{A}BD + BC\bar{D} + \bar{B}\bar{C}D + A\bar{B}\bar{D}$$

(b)

Example

- Use a K map to simplify:

$Y = C'(A'B'D' + D) + AB'C + D'$

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	1	1	0	1
$\overline{A}B$	1	1	0	1
AB	1	1	0	1
$A\overline{B}$	1	1	1	1

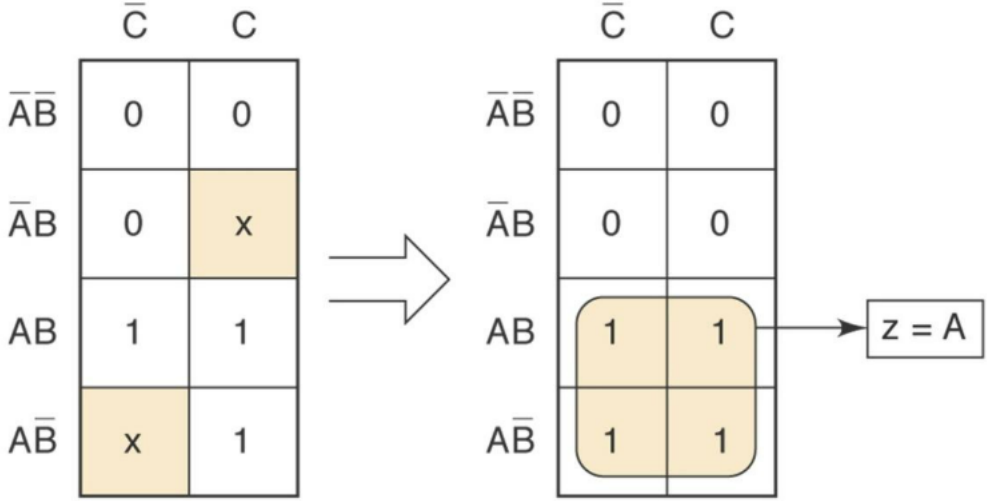
$y = A\overline{B} + \overline{C} + \overline{D}$

Don’t Care Conditions

- In certain cases some of the minterms may never occur or it may not matter what happens if they do
  - In such cases we fill in the Karnaugh map with X
    - meaning don't care
  - When minimizing an X is like a "joker"
    - X can be 0 or 1 - whatever helps best with the minimization
- “Don’t care” conditions should be changed to either 0 or 1 to produce K-map looping that yields the simplest expression.

A	B	C	z	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	x	"don't care"
1	0	0	x	
1	0	1	1	
1	1	0	1	
1	1	1	1	

(a)



Terminology: Minterms

- A minterm is a special product of literals, in which each input variable appears exactly once.
- A function with n variables has 2^n minterms (since each variable can appear complemented or not)
- A three-variable function, such as f(x,y,z), has $2^3 = 8$ minterms:

  x'y'z' x'y'z x'yz' x'yz
  xy'z' xy'z xyz' xyz

- Each minterm is true for exactly one combination of inputs:

Minterm	Is true when…	Shorthand
x'y'z'	x=0, y=0, z=0	m_0
x'y'z	x=0, y=0, z=1	m_1
x'yz'	x=0, y=1, z=0	m_2
x'yz	x=0, y=1, z=1	m_3
xy'z'	x=1, y=0, z=0	m_4
xy'z	x=1, y=0, z=1	m_5
xyz'	x=1, y=1, z=0	m_6
xyz	x=1, y=1, z=1	m_7

Terminology: Sum of minterms form

- Every function can be written as a sum of minterms, which is a special kind of sum of products form
- The sum of minterms form for any function is *unique*
- If you have a truth table for a function, you can write a sum of minterms expression just by picking out the rows of the table where the function output is 1.

x	y	z	f(x,y,z)	f'(x,y,z)
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

$f = x'y'z' + x'y'z + x'yz' + x'yz + xyz'$
$= m_0 + m_1 + m_2 + m_3 + m_6$
$= \Sigma m(0,1,2,3,6)$

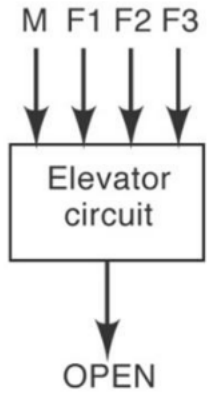
$f' = xy'z' + xy'z +$
xyz
$= m_4 + m_5 + m_7$
$= \Sigma m(4,5,7)$

f' contains all the minterms not in f

Example

- Let's design a logic circuit that controls an elevator door in a three-story building.
  - The circuit has four inputs.
  - M is a logic signal that indicates when the elevator is moving (M= 1) or stopped (M = 0).
  - F1,F2, and F3 are floor indicator signals that are normally LOW, and they go HIGH only when the elevator is positioned at the level of that particular floor.
  - For example, when the elevator is lined up level with the second floor, F2 = 1 and F1 = F3 = 0. The circuit output is the OPEN signal, which is normally LOW and will go HIGH when the elevator door is to be opened.

Example



M	F1	F2	F3	OPEN
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	X
0	1	0	0	1
0	1	0	1	X
0	1	1	0	X
0	1	1	1	X
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	X
1	1	0	0	0
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

	$\overline{F2}\,\overline{F3}$	$\overline{F2}F3$	$F2F3$	$F2\overline{F3}$
$\overline{M}\,\overline{F1}$	0	1	X	1
$\overline{M}\,F1$	1	X	X	X
$M\,F1$	0	X	X	X
$M\,\overline{F1}$	0	0	X	0

	$\overline{F2}\,\overline{F3}$	$\overline{F2}F3$	$F2F3$	$F2\overline{F3}$
$\overline{M}\,\overline{F1}$	0	1	1	1
$\overline{M}\,F1$	1	1	1	1
$M\,F1$	0	0	0	0
$M\,\overline{F1}$	0	0	0	0

$$OPEN = \overline{M}\,(F1 + F2 + F3)$$

Example

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1	1	1
$\bar{A}B$	1	1	0	0
AB	0	0	0	1
$A\bar{B}$	0	0	1	1

(a)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	1	1
$\bar{A}B$	1	0	0	1
AB	0	0	0	0
$A\bar{B}$	1	0	1	1

(b)

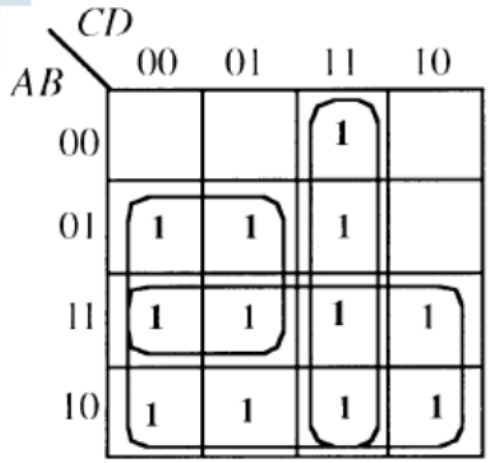
	$\bar{C}$	C
$\bar{A}\bar{B}$	1	1
$\bar{A}B$	0	0
AB	1	0
$A\bar{B}$	1	X

(c)

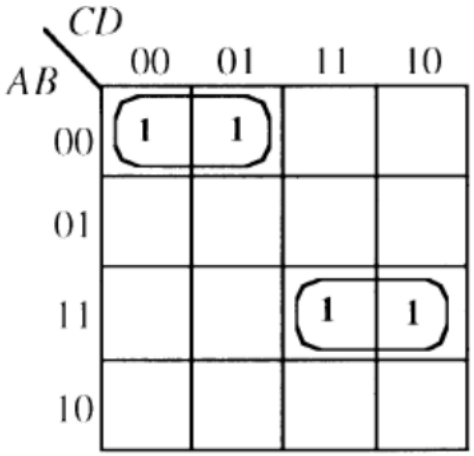
Example

- Use a Karnaugh map to reduce each expression to a minimum SOP form:
- a) X = A+ B'C + CD
- b) X = A' B' C D + A' B' C' D + A B C D + A B C D'
- c) X = A' B(C' D' + C' D) + AB(C' D' + C'D) + A B' C' D
- d) X = (A' B' + A B')(CD + C D')
- e) X = A' B' + A B' + C' D' + C D'

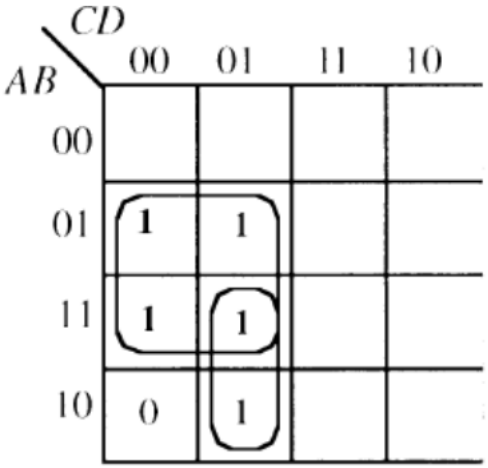
Example



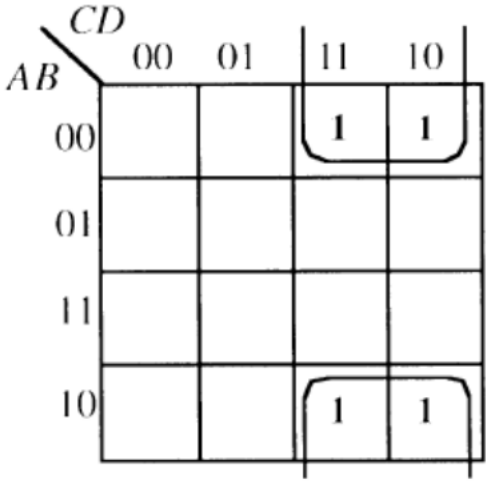
(a) $X = A + B\overline{C} + CD$



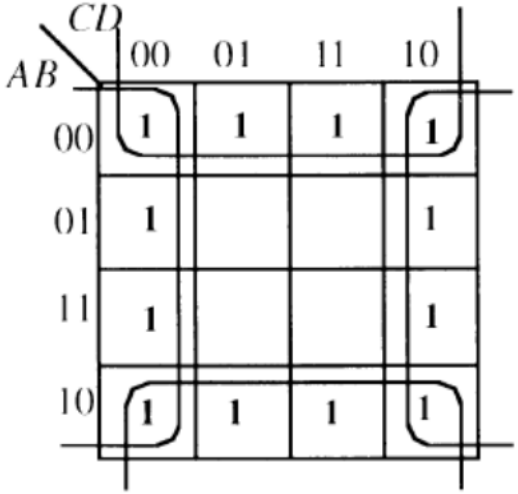
(b) $X = \overline{A}\overline{B}\overline{C} + ABC$



(c) $X = B\overline{C} + A\overline{C}D$



(d) $X = \overline{B}C$



(e) $X = \overline{B} + \overline{D}$

FIGURE 4-14

K Map Method Summary

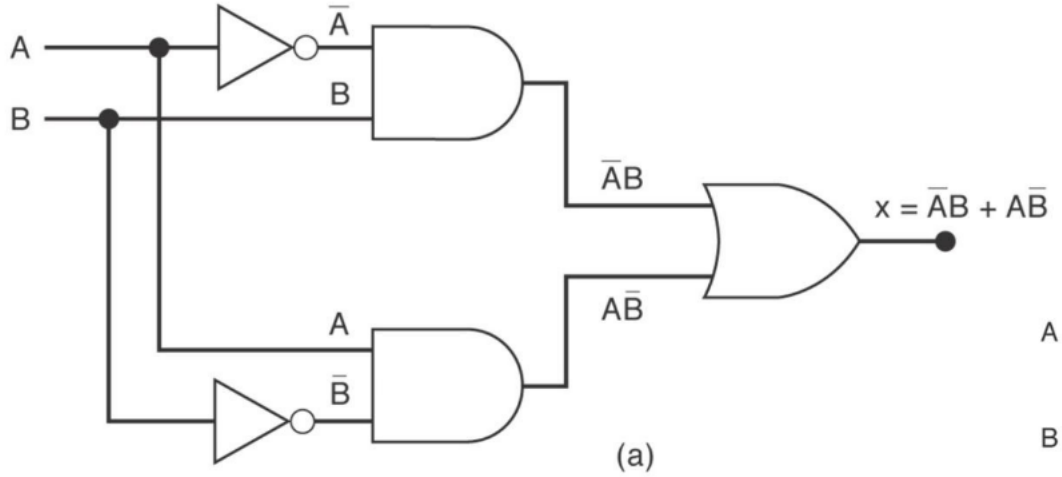
- Compared to the algebraic method, the K-map process is a more orderly process requiring fewer steps and always producing a minimum expression.
- The minimum expression in generally is NOT unique.
- For the circuits with large numbers of inputs (larger than four), other more complex techniques are used.

Summary

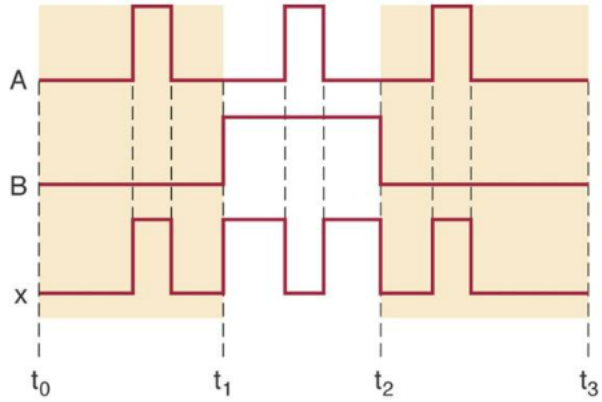
- SOP and POS –useful forms of Boolean equations
- Design of a comb. Logic circuit
  - (1) construct its truth table, (2) convert it to a SOP, (3) simplify using Boolean algebra or K mapping, (4) implement
- K map: a graphical method for representing a circuit's truth table and generating a simplified expression
- "Don't cares" entries in K map can take on values of 1 or 0. Therefore can be exploited to help simplification

Exclusive-OR

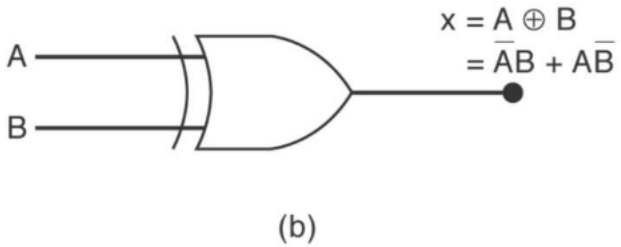
- The exclusive OR (XOR) produces a HIGH output whenever the two inputs are at opposite levels.

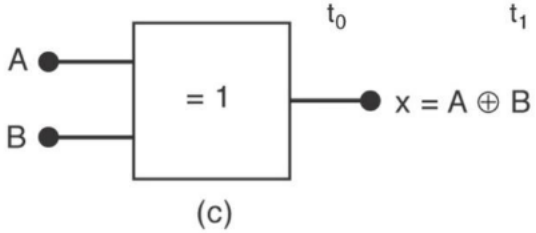


A	B	x
0	0	0
0	1	1
1	0	1
1	1	0



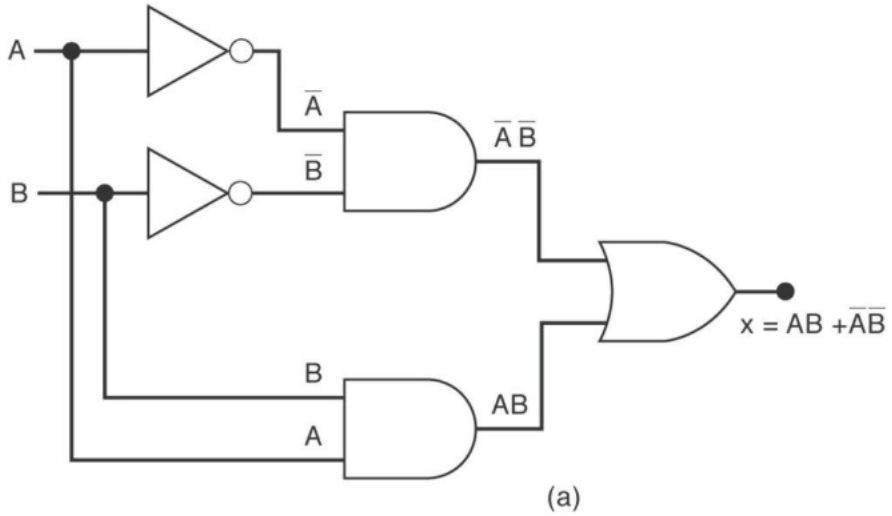
XOR gate symbols





Exclusive-NOR

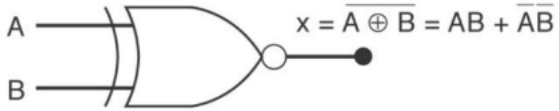
- The exclusive NOR (XNOR) produces a HIGH output whenever the two inputs are at the same level.
- XOR and XNOR outputs are opposite.



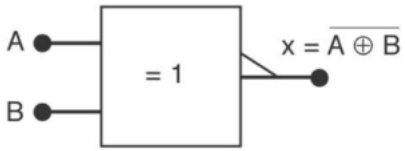
(a)

A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

XNOR gate symbols



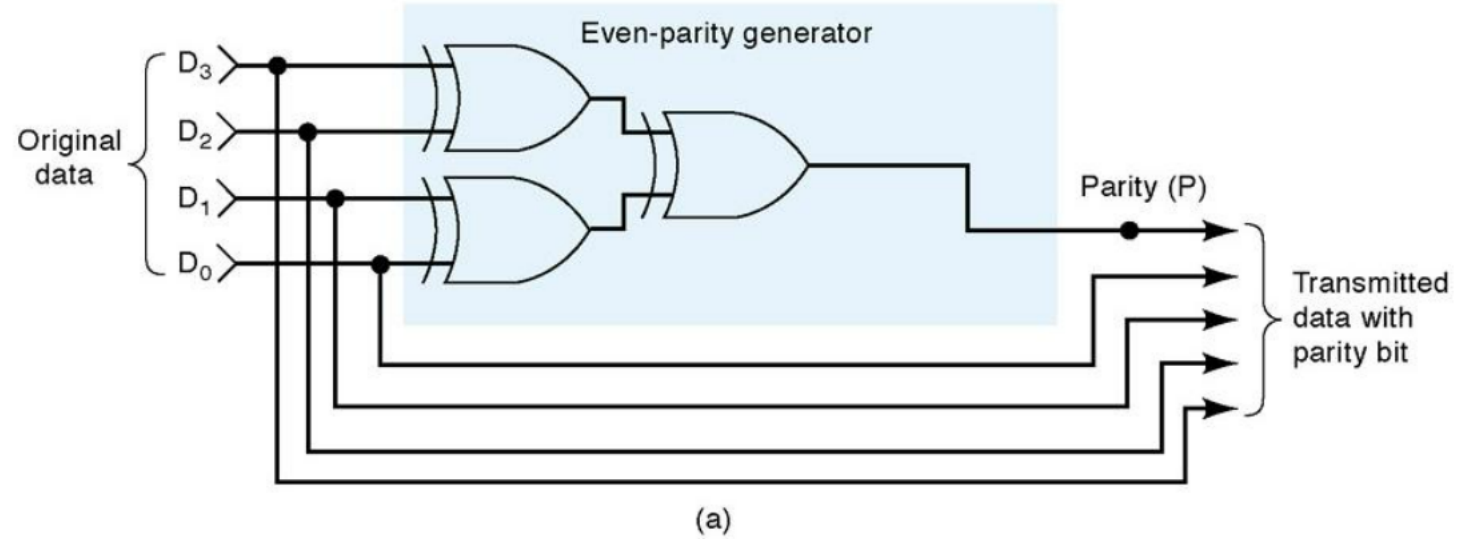
(b)



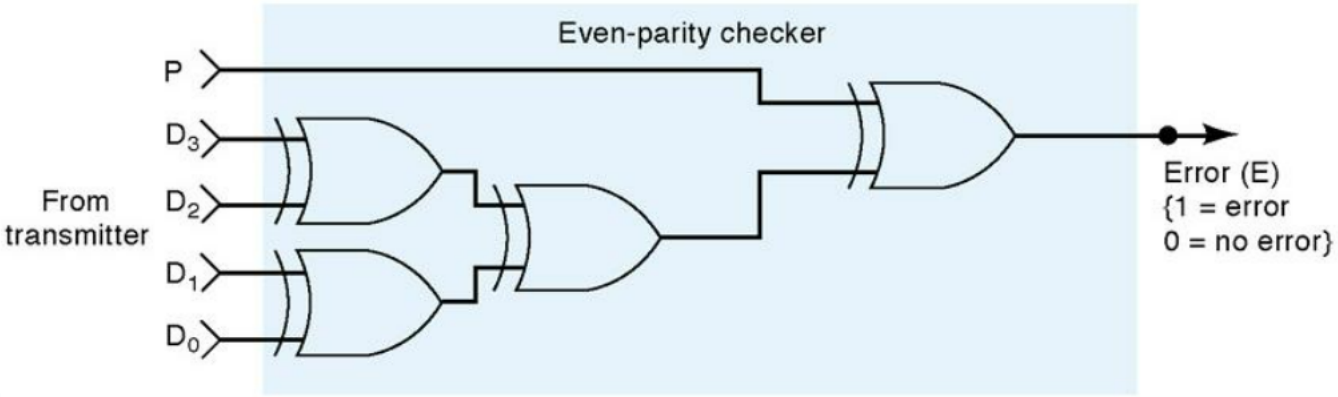
(c)

Parity Generator and Checker

- XOR and XNOR gates are useful in circuits for parity generation and checking.



(a)



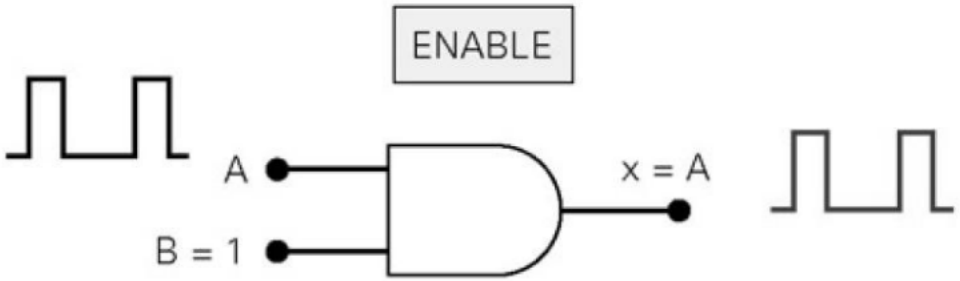
(b)

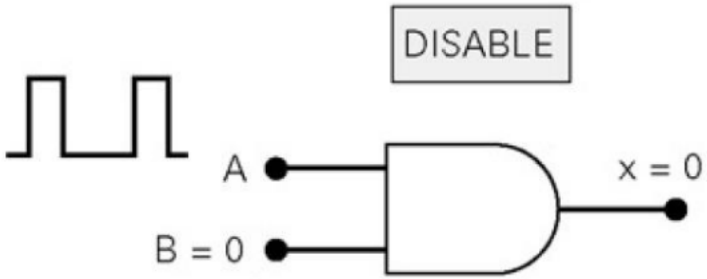
Enable/Disable Circuits

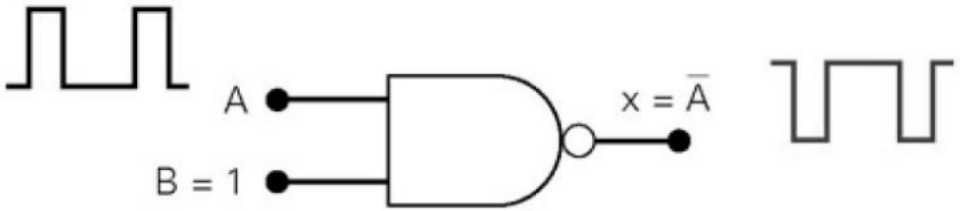
- A circuit is enabled when it allows the passage of an input signal to the output.
- A circuit is disabled when it prevents the passage of an input signal to the output.
- Situations requiring enable/disable circuits occur frequently in digital circuit design.

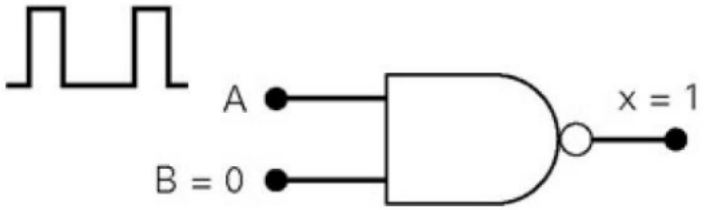
Enable/Disable Circuits

- AND gate function act as enable/disable circuits



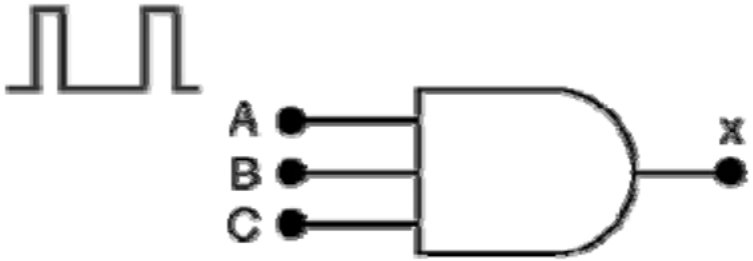




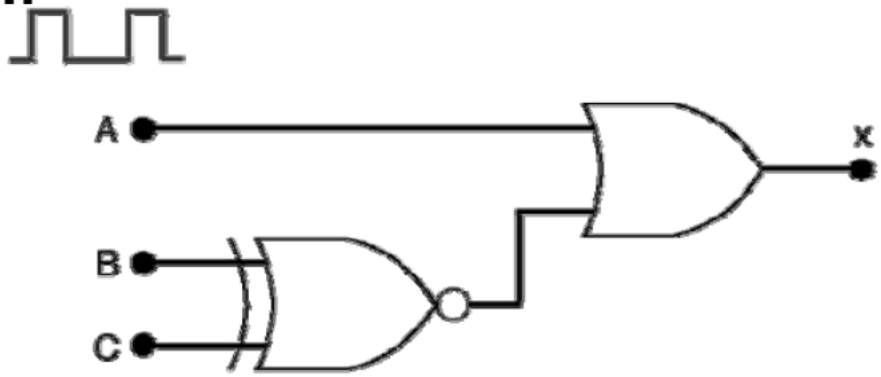


Enable/Disable Circuits

- **Design a logic circuit that will allow a signal to pass to the output only when control inputs B and C are both HIGH; otherwise, the output will stay LOW.**

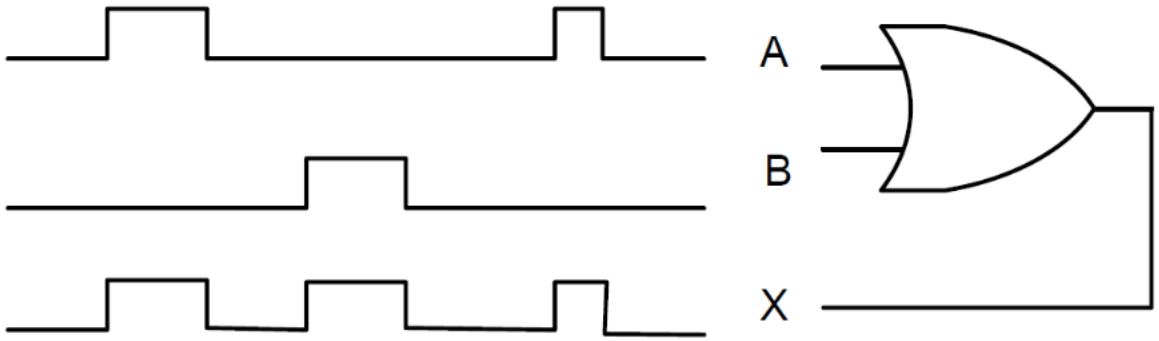


- **Design a logic circuit that will allow a signal to pass to the output only when one, but not both, of the control inputs are HIGH; otherwise, the output will stay HIGH.**

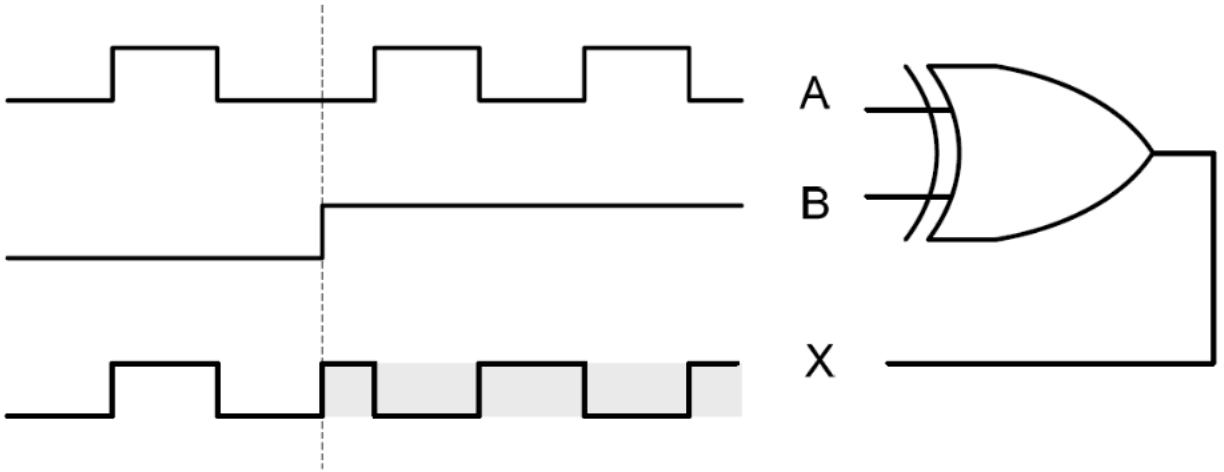


Merging & Inversion Circuits

- **OR gate performs signal merging function**



- **XOR gate performs selectable inversion function**

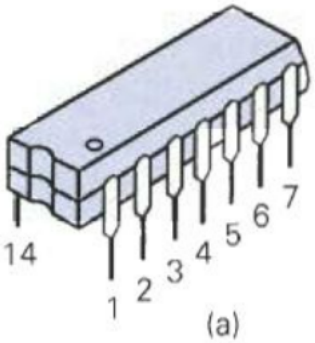


Basic Characteristics of Digital ICs

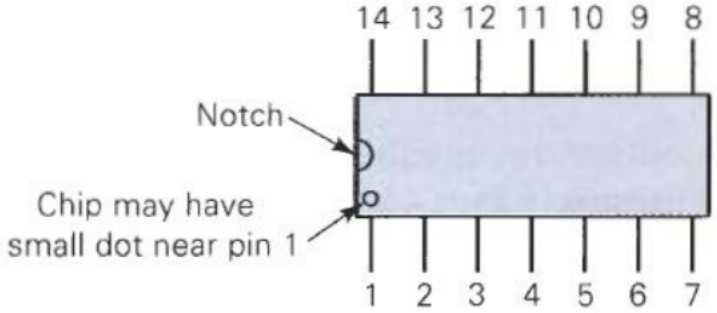
- IC “chips” consist of resistors, diodes, and transistors fabricated on a piece of semiconductor material called a substrate.
- Digital ICs may be categorized according to the number of logic gates on the substrate:
  - SSI – less than 12
  - MSI – 12 to 99
  - LSI – 100 to 9999
  - VLSI – 10,000 to 99,999
  - ULSI – 100,000 to 999,999
  - GSI – 1,000,000 or more

Basic Characteristics of Digital ICs

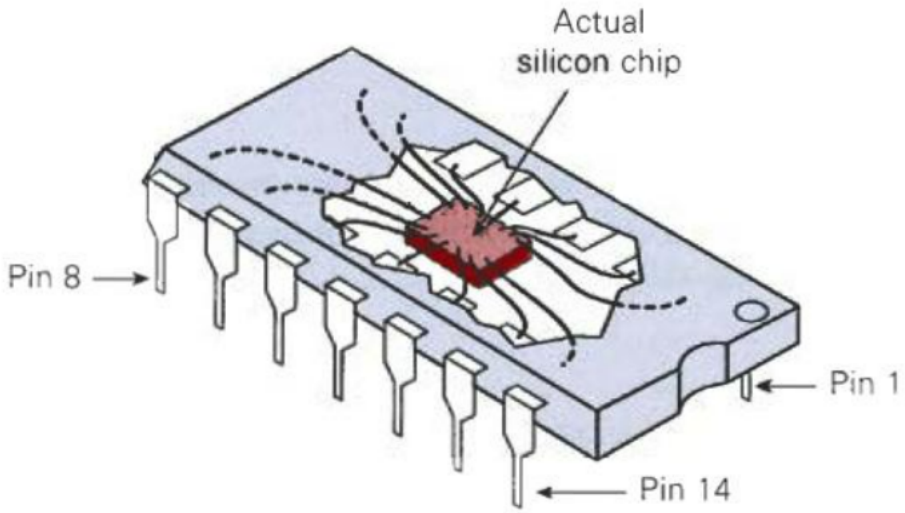
- The first package we will examine is the dual in line package (DIP).



(a)

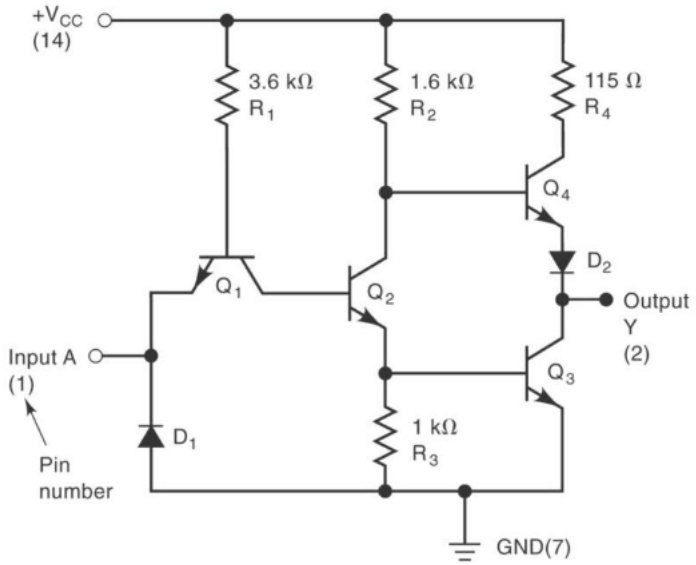


(b)

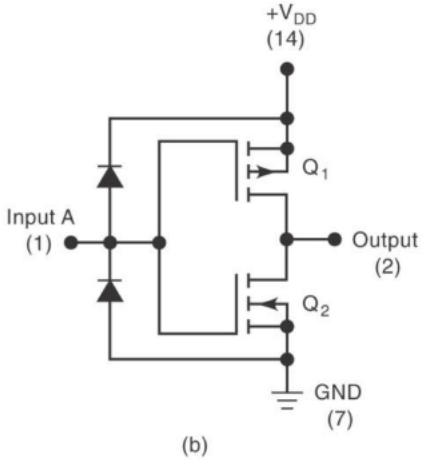


Basic Characteristics of Digital ICs

- ICs are also categorized by the type of components used in their circuits.
  - Bipolar ICs use NPN and PNP transistors
  - Unipolar ICs use FET transistors.
- The transistor-transistor logic (TTL) and the complementary metal-oxide semiconductor (CMOS) families will both be examined.



(a)



(b)

Basic Characteristics of Digital ICs

- The TTL family consists of subfamilies as listed in the table.

TTL Series	Prefix	Example IC
Standard TTL	74	7404 (hex INVERTER)
Schottky TTL	74S	74S04 (hex INVERTER)
Low-power Schottky TTL	74LS	74LS04 (hex INVERTER)
Advanced Schottky TTL	74AS	74AS04 (hex INVERTER)
Advanced low-power Schottky TTL	74ALS	74ALS04 (hex INVERTER)

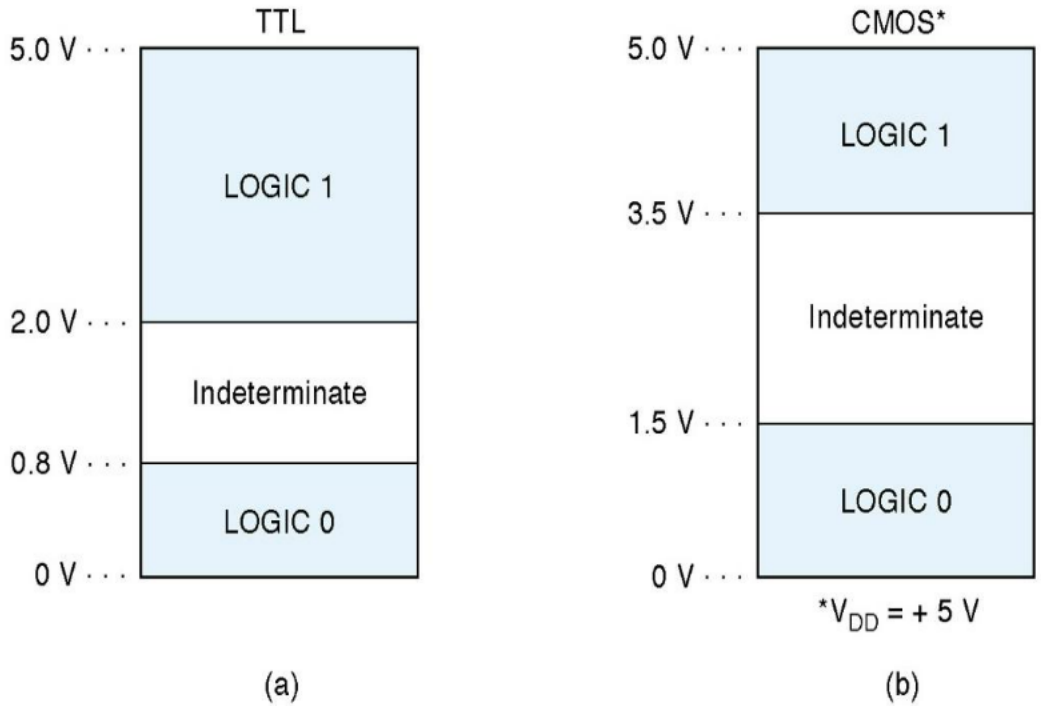
Basic Characteristics of Digital ICs

- The CMOS family consists of several series, some of which are shown in the table.

CMOS Series	Prefix	Example IC
Metal-gate CMOS	40	4001 (quad NOR gates)
Metal-gate, pin-compatible with TTL	74C	74C02 (quad NOR gates)
Silicon-gate, pin-compatible with TTL, high-speed	74HC	74HC02 (quad NOR gates)
Silicon-gate, high-speed, pin-compatible and electrically compatible with TTL	74HCT	74HCT02 (quad NOR gates)
Advanced-performance CMOS, not pin-compatible or electrically compatible with TTL	74AC	74AC02 (quad NOR)
Advanced-performance CMOS, not pin-compatible with TTL, but electrically compatible with TTL	74ACT	74ACT02 (quad NOR)

Basic Characteristics of Digital ICs

- Power (referred to as V_{CC}) and ground connections are required for chip operation.
- V_{CC} for TTL devices is normally +5 V.
- V_{DD} for CMOS devices can be from +3 to +18 V.



Basic Characteristics of Digital ICs

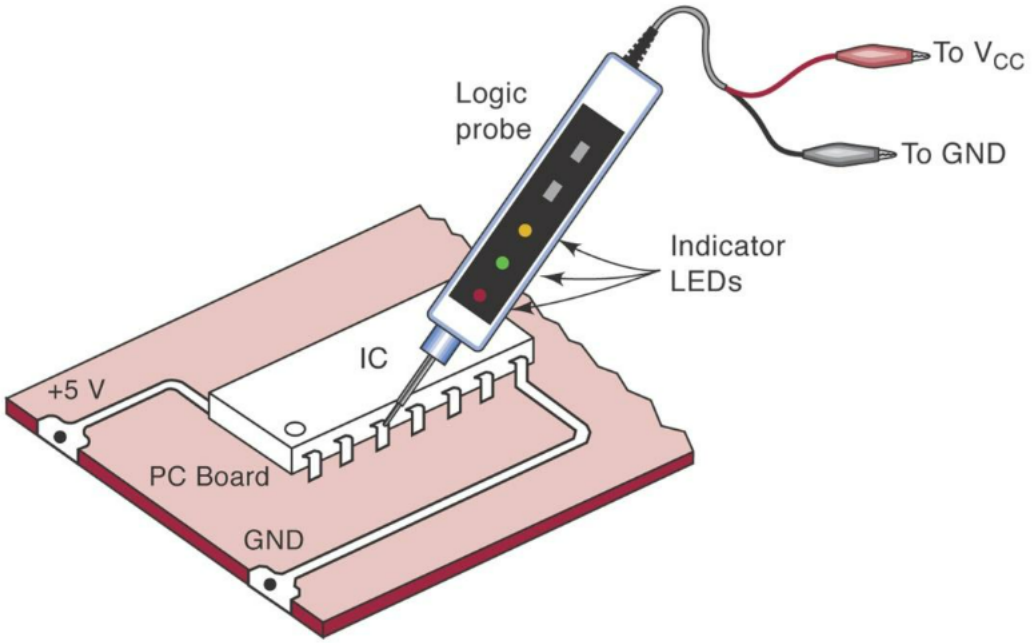
- Inputs that are not connected are said to be floating. The consequences of floating inputs differ for TTL and CMOS.
  - Floating TTL input acts like a logic 1. The voltage measurement may appear in the indeterminate range, but the device will behave as if there is a 1 on the floating input.
  - Floating CMOS inputs can cause overheating and damage to the device. Some ICs have protection circuits built in, but the best practice is to tie all unused inputs either high or low.

Troubleshooting Digital Systems

- 3 basic steps
  - Fault detection, determine operation to expected operation.
  - Fault isolation, test and measure to isolate the fault.
  - Fault correction, repair the fault.
- Good troubleshooting skills come through experience in actual hands-on troubleshooting.
- The basic troubleshooting tools used here will be: the logic probe, oscilloscope, and logic pulser.
- The most important tool is the technician's brain.

Troubleshooting Digital Systems

- The logic probe will indicate the presence or absence of a signal when touched to a pin as indicated below.



LEDs			
Red	Green	Yellow	Logic Condition
OFF	ON	OFF	LOW
ON	OFF	OFF	HIGH
OFF	OFF	OFF	INDETERMINATE*
X	X	FLASHING	PULSING

* Includes open or floating condition

Internal Digital IC Faults

- Most common internal failures:
  - Malfunction in the internal circuitry.
  - Inputs or outputs shorted to ground or V_{CC}
  - Inputs or outputs open-circuited
  - Short between two pins (other than ground or V_{CC})

Internal Digital IC Faults

- Malfunction in internal circuitry
  - Outputs do not respond properly to inputs. Outputs are unpredictable.
- Input internally shorted to ground or supply
  - The input will be stuck in LOW or HIGH state.
- Output internally shorted to ground or supply
  - Output will be stuck in LOW or HIGH state.
- Open-circuited input or output
  - Floating input in a TTL device will result in a HIGH output. Floating input in a CMOS device will result in erratic or possibly destructive output.
  - An open output will result in a floating indication.
- Short between two pins
  - The signal at those pins will always be identical.

External Faults

- Open signal lines – signal is prevented from moving between points. Some causes:
  - Broken wire
  - Poor connections (solder or wire-wrap)
  - Cut or crack on PC board trace
  - Bent or broken IC pins.
  - Faulty IC socket
- Detect visually and verify with an ohmmeter.

External Faults

- Shorted signal lines – the same signal will appear on two or more pins. V_{CC} or ground may also be shorted. Some causes:
  - Sloppy wiring
  - Solder bridges
  - Incomplete etching
- Detect visually and verify with an ohmmeter.

External Faults

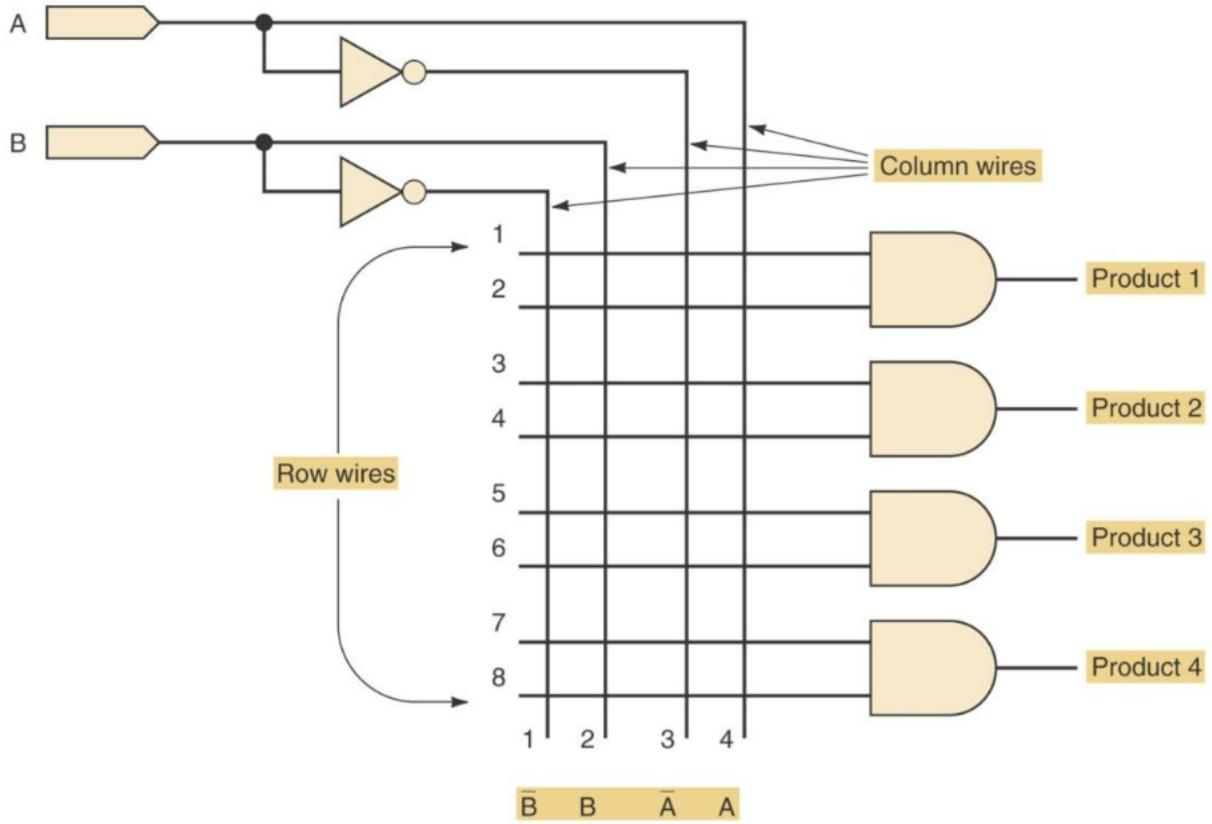
- Faulty power supply – ICs will not operate or will operate erratically.
  - May lose regulation due to an internal fault or because circuits are drawing too much current.
  - Always verify that power supplies are providing the specified range of voltages and are properly grounded.
  - Use an oscilloscope to verify that AC signals are not present.

External Faults

- Output loading – caused by connecting too many inputs to the output of an IC.
  - Causes output voltage to fall into the indeterminate range.
  - This is called *loading* the output.
  - Usually a result of poor design or bad connection.

Programmable Logic Devices

- PLDs allow the design process to be automated.
- Designers identify inputs, outputs, and logical relationships.
- PLDs are electronically configured to form the defined logic circuits.



Programmable Logic Devices

- PLD ICs can be programmed out of system or in system.
- Logic circuits can be described using schematic diagrams, logic equations, truth tables, and HDL.
- PLD development software can convert any of these descriptions into 1s and 0s and loaded into the PLD.

Programmable Logic Devices

- Hierarchical design – small logic circuits are defined and combined with other circuits to form a large section of a project. Large sections can be combined and connected for form a system.
- Top-down design requires the definition of sub sections that will make up the system, and definition of the individual circuits that will make up each sub section.
- Each level of the hierarchy can be designed and tested individually.

Programmable Logic Devices

- A system is built from the bottom up.
  - Each block is described by a design file.
  - The designed block is tested
  - After testing it is compiled using development software.
  - The compiled block is tested using a simulator for verify correct operation.
  - A PLD is programmed to verify correct operation.