- *Selected areas covered in this chapter*:
  - Converting logic expressions to sum-of-products expressions.
  - Boolean algebra and the Karnaugh map as tools to simplify and design logic circuits.
  - Operation of exclusive-OR & exclusive-NOR circuits.
  - Designing simple logic circuits without a truth table.
  - Basic characteristics of TTL and CMOS digital ICs.
  - Basic troubleshooting rules of digital systems.

Combinational logic circuits:

- at any time, the logic level at the output depends on the combination of logic levels present at the inputs
- has no *memory characteristic*

4-1 Sum-of-Products Form

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

1. $ABC + \bar{A}B\bar{C}$
2. $AB + \bar{A}B\bar{C} + \bar{C}\bar{D} + D$
3. $\bar{A}B + C\bar{D} + EF + GK + H\bar{L}$

a **minterm** is a logical expression of n variables that employs only the **NOT** operator and the **AND** operator

minterms are often numbered by a binary encoding of the variables (1--> normal form, 0 --> complemented form)

$$m2 = \bar{A}B\bar{C}$$

In canonical form: logic expression is described in sum of minterms

$$x = f(A, B, C) = m1 + m3 + m6 + m7 = \sum(1,3,6,7)$$

4-1 Sum-of-Products Form

The **product-of-sums (POS)** form consists of two or more **OR** terms (sums) **AND**ed together.

1. $(A + \overline{B} + C)(A + C)$
2. $(A + \overline{B})(\overline{C} + D)F$
3. $(A + C)(B + \overline{D})(\overline{B} + C)(A + \overline{D} + \overline{E})$

a **Maxterm** is a logical expression of n variables that employs only the **NOT** operator and the **OR** operator

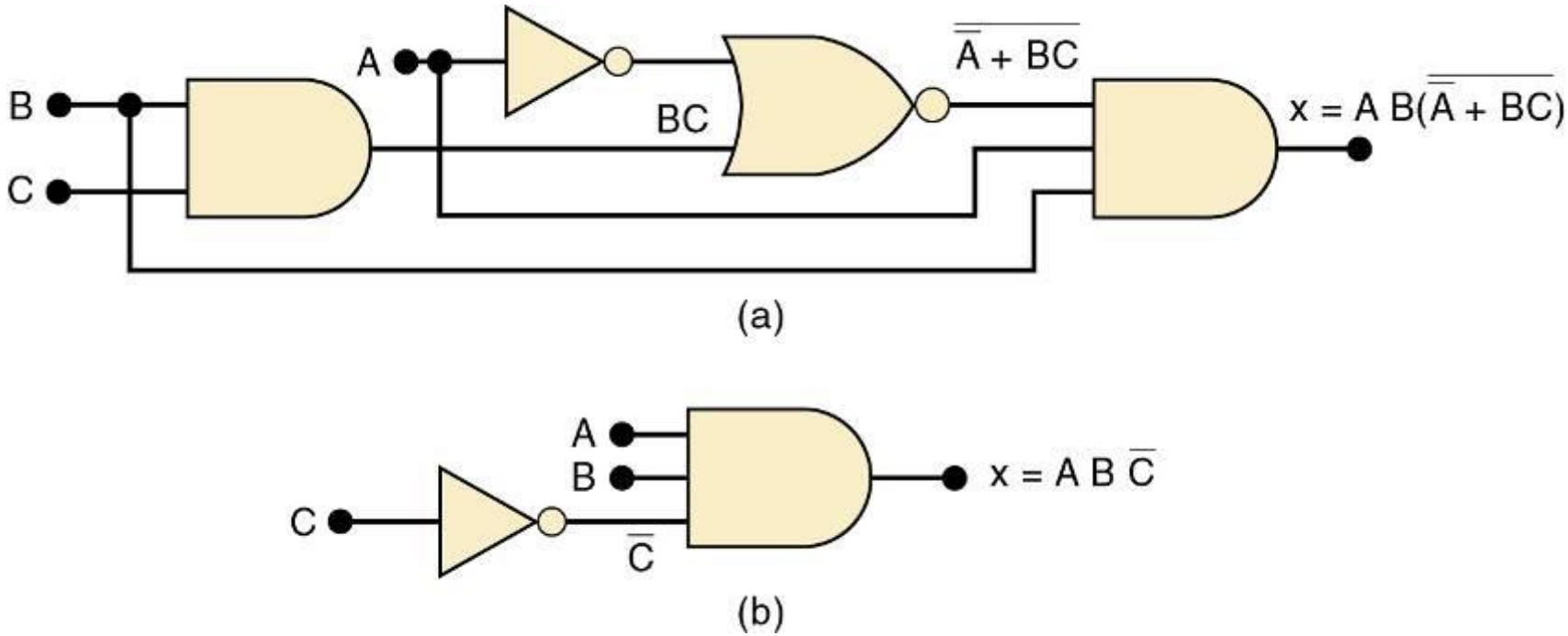
Each maxterm is assigned an index based on the opposite conventional binary encoding used for minterms (0 --> normal form; 1 --> complemented form).

$$M2 = A + \bar{B} + C$$

In canonical form: logic expression is described in product of maxterms

$$x = f(A, B, C) = M1M3M4M6 = \prod(1,3,4,6)$$

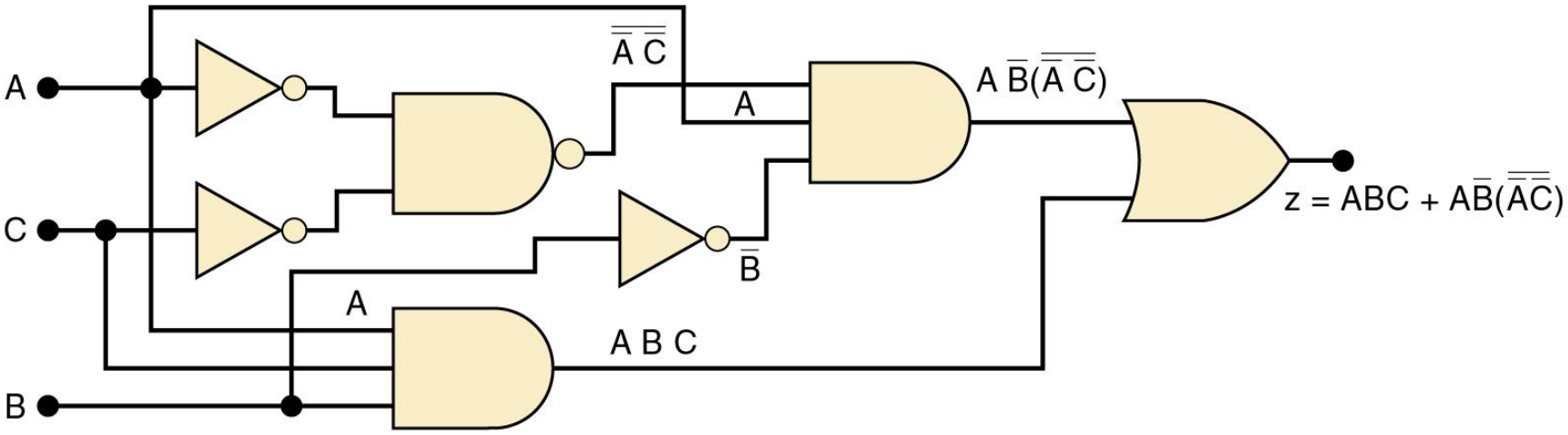
- The circuits shown provide the same output
  - Circuit (b) is clearly less complex.



Logic circuits can be simplified using Boolean algebra and Karnaugh mapping.

4-3 Algebraic Simplification

- Place the expression in SOP form by applying DeMorgan's theorems and multiplying terms.
- Check the SOP form for common factors.
- Factoring where possible should eliminate one or more terms.



The first step is to determine the expression for the output: $z = ABC + A\overline{B} \cdot (\overline{\overline{A}\,\overline{C}})$

4-3 Algebraic Simplification

Once the expression is determined, break down large inverter signs by DeMorgan's theorems & multiply out all terms.

$$Z = ABC + A\bar{B}\left(\overline{\bar{A}\bar{C}}\right)$$

$$\begin{aligned} z &= ABC + A\bar{B}(\bar{\bar{A}} + \bar{\bar{C}}) && \text{[theorem (17)]} \\ &= ABC + A\bar{B}(A + C) && \text{[cancel double inversions]} \\ &= ABC + A\bar{B}A + A\bar{B}C && \text{[multiply out]} \\ &= ABC + A\bar{B} + A\bar{B}C && [A \cdot A = A] \end{aligned}$$

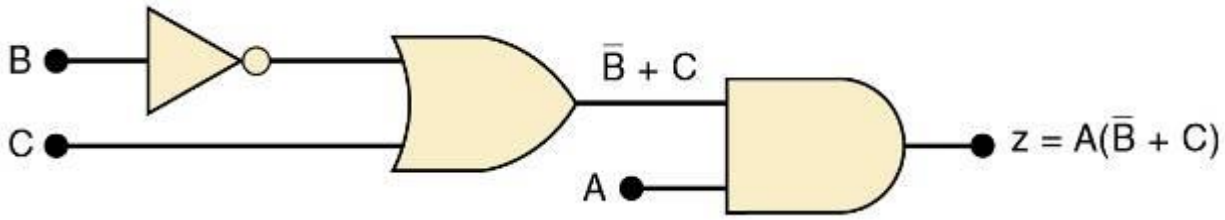
Factoring—the first & third terms above have ***AC*** in common, which can be factored out:

$$z = AC(B + \bar{B}) + A\bar{B}$$

Since ***B*** + ***B*** = 1, then…

$$\begin{aligned} z &= AC(1) + A\bar{B} \\ &= AC + A\bar{B} \end{aligned}$$

Factor out ***A***, which results in…

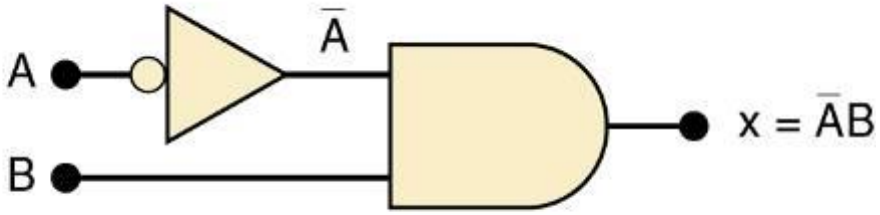


$$\boldsymbol{z = A(C + \bar{B})}$$

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

Circuit that produces a 1 output only for the A = 0, B = 1 condition.

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



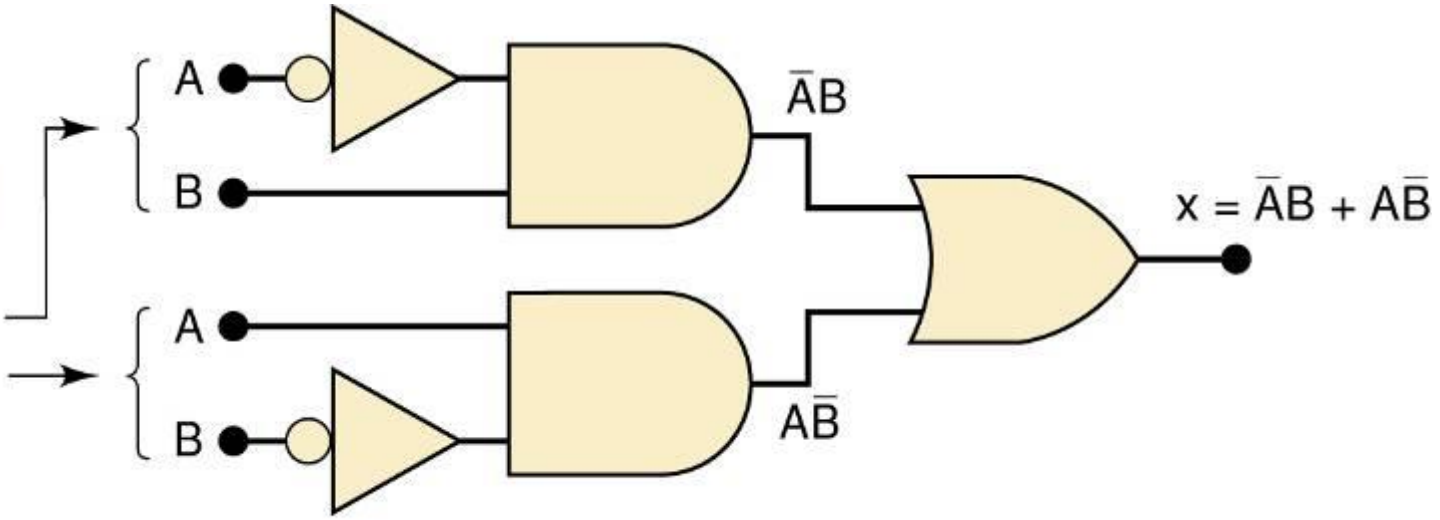
- To solve any logic design problem (2nd method):
  - Interpret the problem and set up its truth table.
  - Write the **OR** term for each case where output = 0.
  - Combine the terms in POS form.
  - Simplify the output expression if possible.
  - Implement the circuit for the final, simplified expression.

4-4 Designing Combinational Logic Circuits

Each set of input conditions that is to produce 1 output is implemented by a separate **AND** gate.

The **AND** outputs are **OR**ed to produce the final output.

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



4-4 Designing Combinational Logic Circuits

Design a logic circuit with three inputs, A, B, and C. Output to be HIGH only when a majority inputs are HIGH.

Truth table.

A	B	C	x	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\rightarrow \bar{A}BC$
1	0	0	0	
1	0	1	1	$\rightarrow A\bar{B}C$
1	1	0	1	$\rightarrow AB\bar{C}$
1	1	1	1	$\rightarrow ABC$

AND terms for each case where output is 1.

SOP expression for the output:

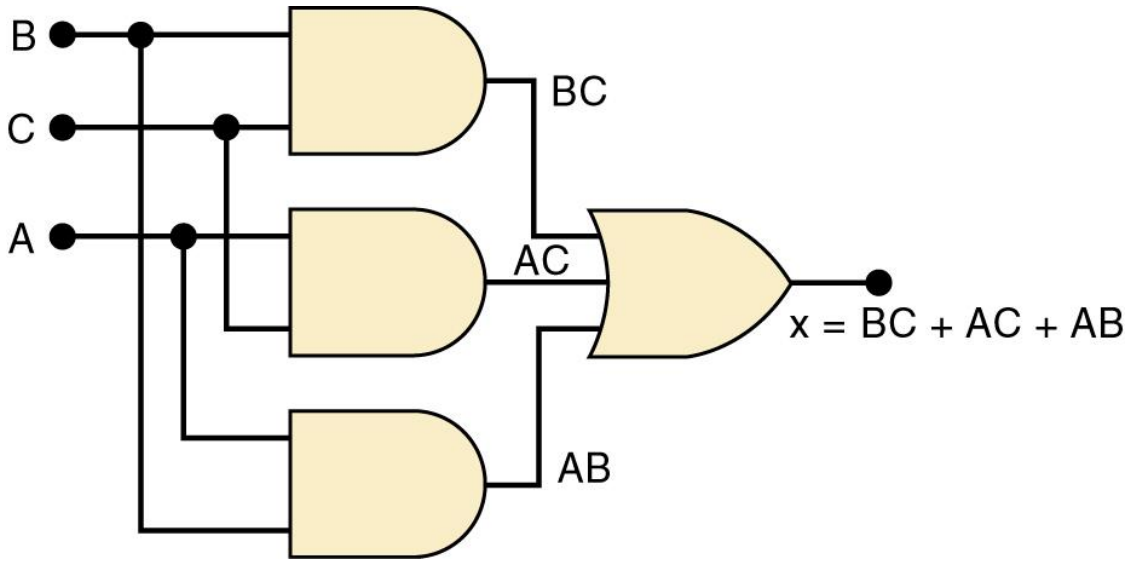
$$x = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

$$x = f(A,B,C) = m3 + m5 + m6 + m7 = \sum(3,5,6,7)$$

Design a logic circuit with three inputs, A, B, and C. Output to be HIGH only when a majority inputs are HIGH.

Implementing the circuit after factoring:

$$x = BC + AC + AB$$

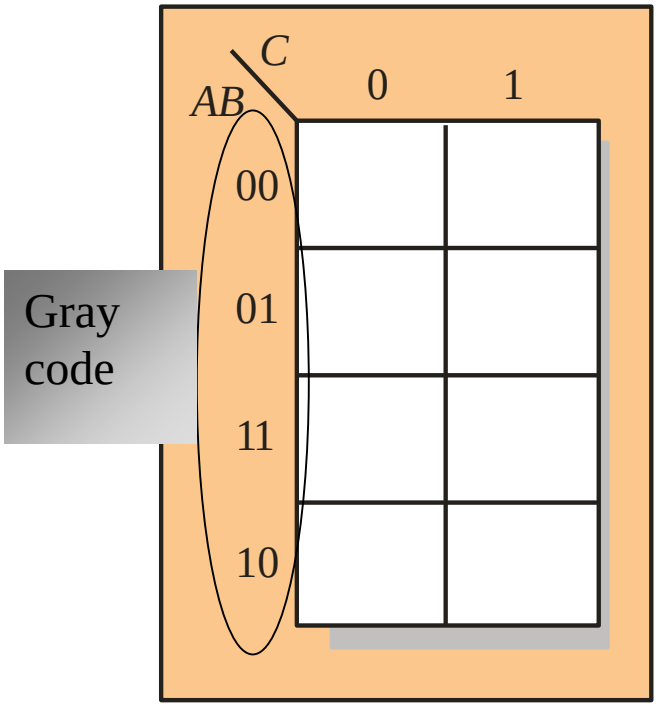


Since the expression is in SOP form, the circuit is a group of **AND** gates, working into a single **OR** gate,

4-5 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—practically limited to 5 or 6 variables.

Cells are usually labeled using 0's and 1's to represent the variable and its complement.



The numbers are entered in gray code, to force adjacent cells to be different by only one variable.

Ones are read as the true variable and zeros are read as the complemented variable.

- Alternatively, cells can be labeled with the variable letters. This makes it simple to read, but it takes more time preparing the map.

The truth table values are placed in the K map.
Shown here is a two-variable map.

A	B	X
0	0	$1 \rightarrow \overline{A}\,\overline{B}$
0	1	0
1	0	0
1	1	$1 \rightarrow AB$

$$x = \overline{A}\,\overline{B} + AB$$

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

Four-variable K-Map.

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	$\rightarrow ABCD$

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

	$\bar{C}\bar{D}$	$\bar{C}D$	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

Adjacent K map square differ in only one variable both horizontally and vertically.

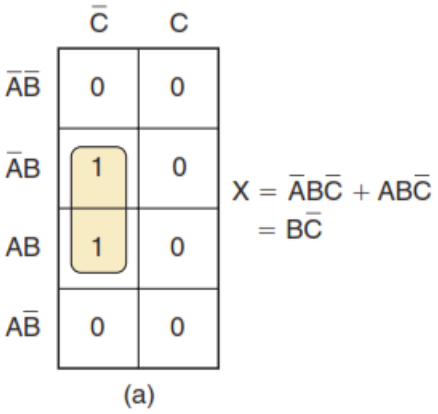
A SOP expression can be obtained by **OR**ing all squares that contain a 1.

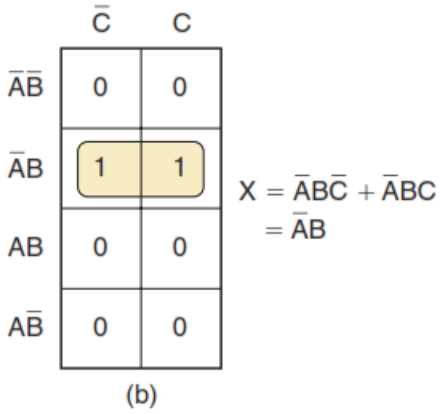
4-5 Karnaugh Map Method

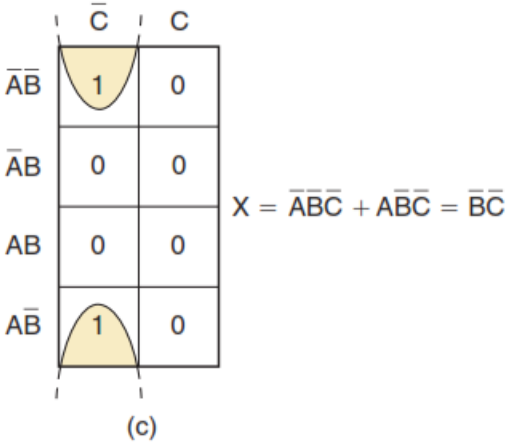
Looping 1s in adjacent groups of 2, 4, or 8 will result in further simplification.

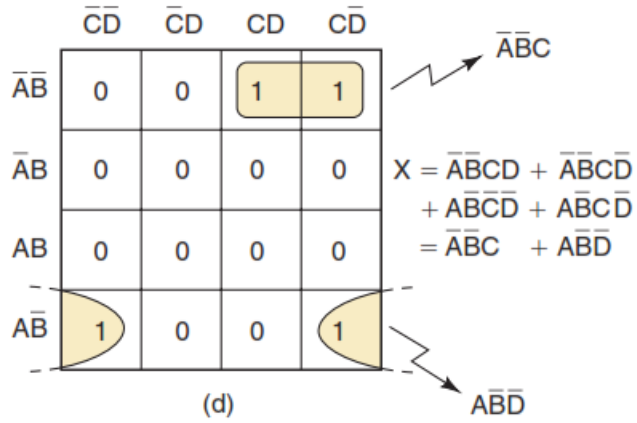
Looping groups of 2 (Pairs)

Looping a pair of adjacent 1s in a K map eliminates the variable that appears in complemented and uncomplemented form.





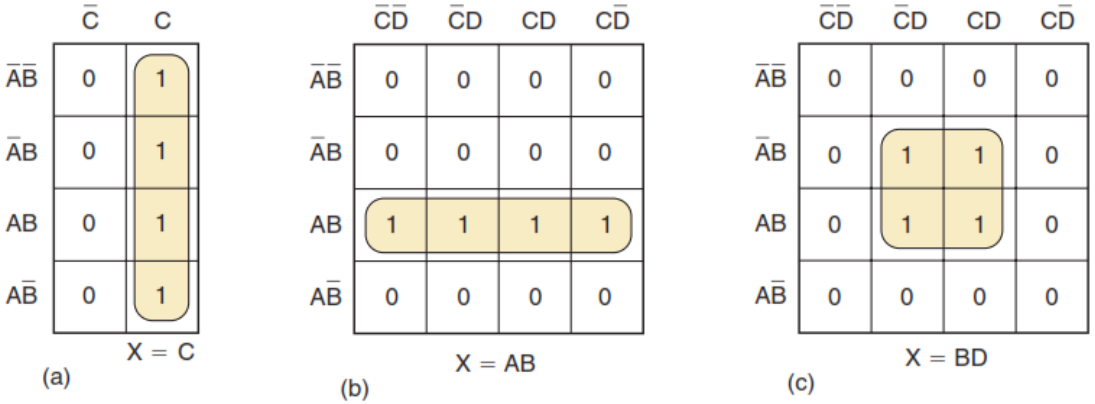




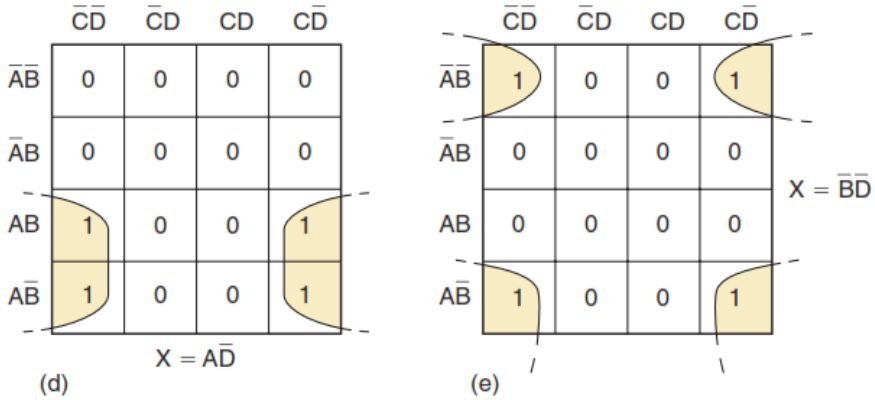
Groups of 4 (Quads)

The top and bottom rows are considered to be adjacent to each other, as are the leftmost and rightmost columns

When a quad is looped, the resultant term will contain only the variables that do not change form for all the squares in the quad

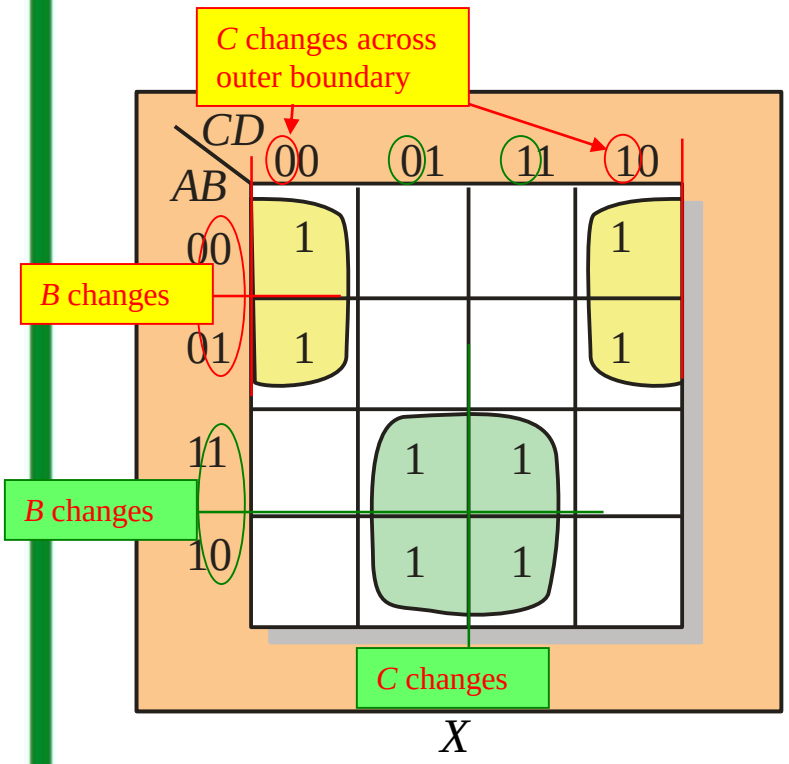


Looping a quad of adjacent 1s eliminates the two variables that appear in both complemented and uncomplemented form.



Groups of 4 (Quads)

Group the 1's on the map and read the minimum logic.



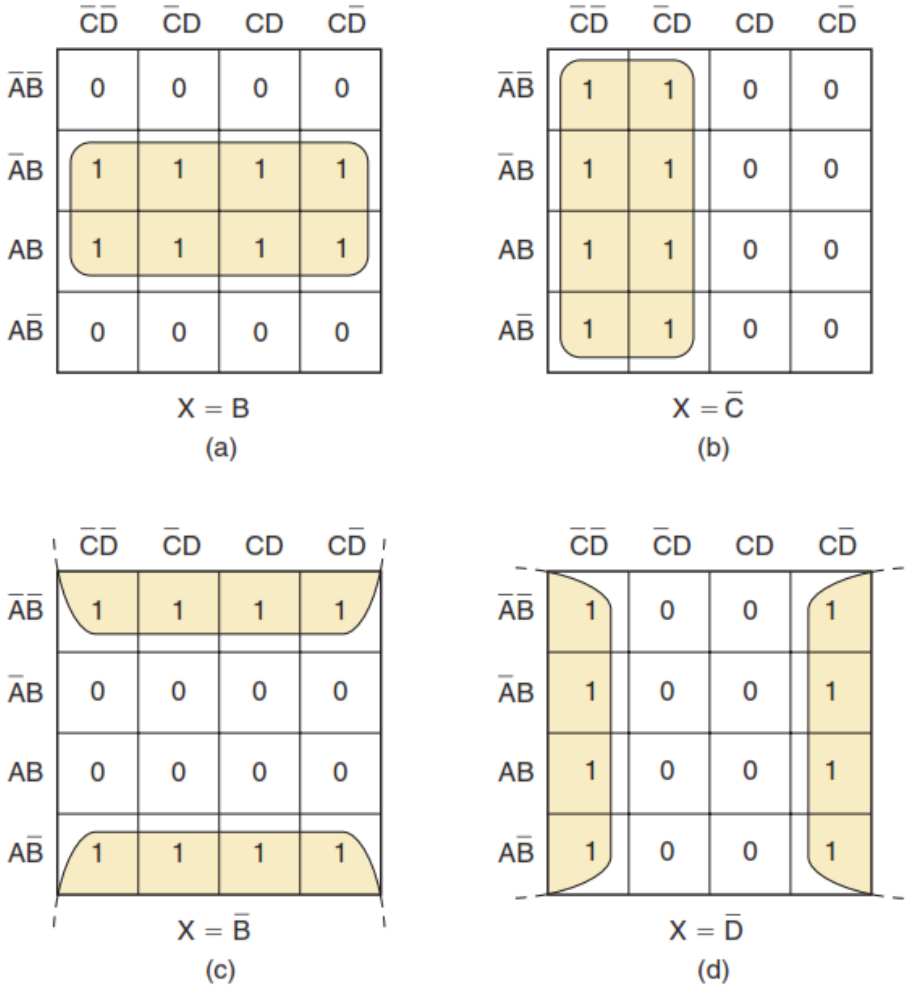
Solution

1. Group the 1's into two separate groups as indicated.
2. Read each group by eliminating any variable that changes across a boundary.
3. The upper (yellow) group is read as $A\overline{D}$.
4. The lower (green) group is read as AD.

$$X = A\overline{D} + AD$$

Groups of 8 (Octets)

Looping an octet of adjacent 1s eliminates the three variables that appear in both complemented and uncomplemented form.



- 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.

When a variable appears in both complemented and uncomplemented form within a loop, that variable is eliminated from the expression.

Variables that are the same for all squares of the loop must appear in the final expression.

4-5 Karnaugh Map Method

- Complete K map simplification process:

Step 1 Construct the K map and place 1s in those squares corresponding to the 1s in the truth table. Place 0s in the other squares.

Step 2 Examine the map for adjacent 1s and loop those 1s that are *not* adjacent to any other 1s. These are called *isolated* 1s.

Step 3 Next, look for those 1s that are adjacent to only one other 1. Loop *any* pair containing such a 1.

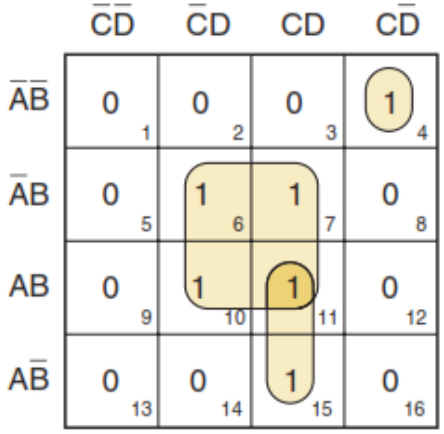
Step 4 Loop any octet even if it contains some 1s that have already been looped.

Step 5 Loop any quad that contains one or more 1s that have not already been looped, *making sure to use the minimum number of loops.*

Step 6 Loop any pairs necessary to include any 1s that have not yet been looped, *making sure to use the minimum number of loops.*

Step 7 Form the OR sum of all the terms generated by each loop.

4-5 Karnaugh Map Method



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

(a)

Solution

Step 2 Square 4 is the only square containing a 1 that is not adjacent to any other 1. It is looped and is referred to as loop 4.

Step 3 Square 15 is adjacent *only* to square 11. This pair is looped and referred to as loop 11, 15.

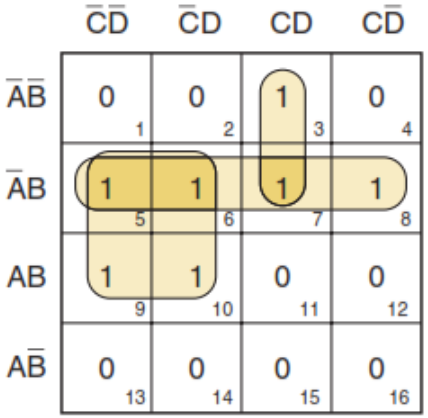
Step 4 There are no octets.

Step 5 Squares 6, 7, 10, and 11 form a quad. This quad is looped (loop 6, 7, 10, 11). Note that square 11 is used again, even though it was part of loop 11, 15.

Step 6 All 1s have already been looped.

Step 7 Each loop generates a term in the expression for X. Loop 4 is simply $\overline{A}\,\overline{B}C\overline{D}$. Loop 11, 15 is ACD (the B variable is eliminated). Loop 6, 7, 10, 11 is BD (A and C are eliminated).

4-5 Karnaugh Map Method



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

(b)

Step 2 There are no isolated 1s.

Step 3 The 1 in square 3 is adjacent *only* to the 1 in square 7. Looping this pair (loop 3, 7) produces the term $\bar{A}CD$.

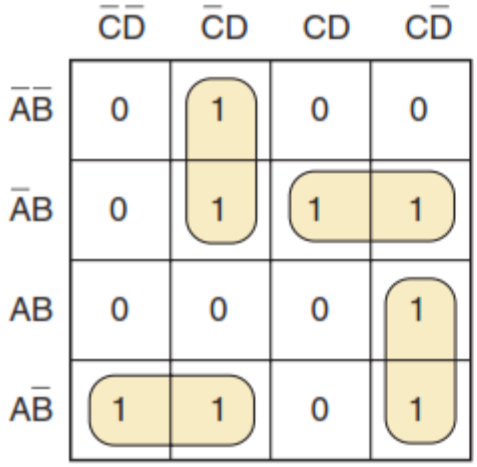
Step 4 There are no octets.

Step 5 There are two quads. Squares 5, 6, 7, and 8 form one quad. Looping this quad produces the term $\bar{A}B$. The second quad is made up of squares 5, 6, 9, and 10. This quad is looped because it contains two squares that have not been looped previously. Looping this quad produces $B\bar{C}$.

Step 6 All 1s have already been looped.

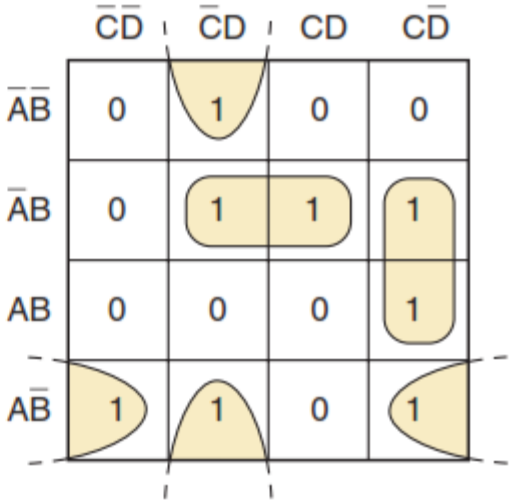
Step 7 The terms generated by the three loops are ORed together to obtain the expression for X.

4-5 Karnaugh Map Method



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

(a)



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

(b)

Is one better than the other?

4-5 Karnaugh Map Method

Filling a K Map from an Output expression

When the desired output is presented as a Boolean expression instead of a truth table, the K map can be filled by using the following steps:

1. Get the expression into SOP form if it is not already in that form.
2. For each product term in the SOP expression, place a 1 in each K-map square whose label contains the same combination of input variables. Place a 0 in all other squares.

Example

Use a K map to simplify $y = \overline{C}(\overline{A}\,\overline{B}\,\overline{D} + D) + A\overline{B}C + \overline{D}$.

	$\overline{C}\,\overline{D}$	$\overline{C}D$	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

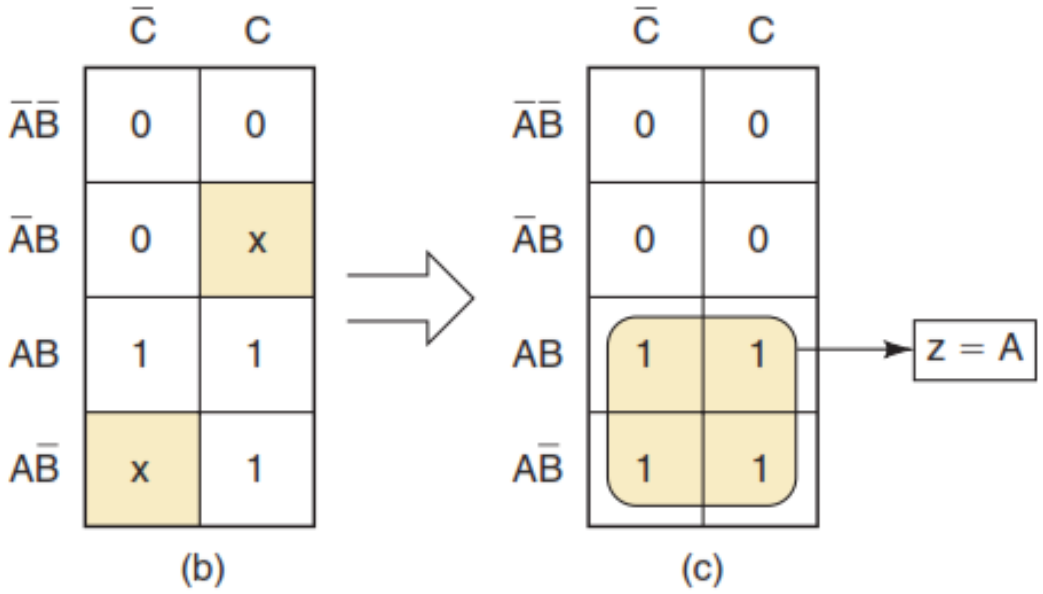
$y = A\overline{B} + \overline{C} + \overline{D}$

Don't-Care Conditions

Some logic circuits can be designed so that there are certain input conditions for which there are no specified output levels, usually because these input conditions will never occur. In other words, there will be certain combinations of input levels where we "don't care" whether the output is 1 or 0

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

(a)



(b) (c)

Whenever don't-care conditions occur, we must decide which x to change to 0 and which to 1 to produce the best K-map looping

4-5 Karnaugh Map Method

Example 1: Design the logic circuit corresponding to the truth table

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

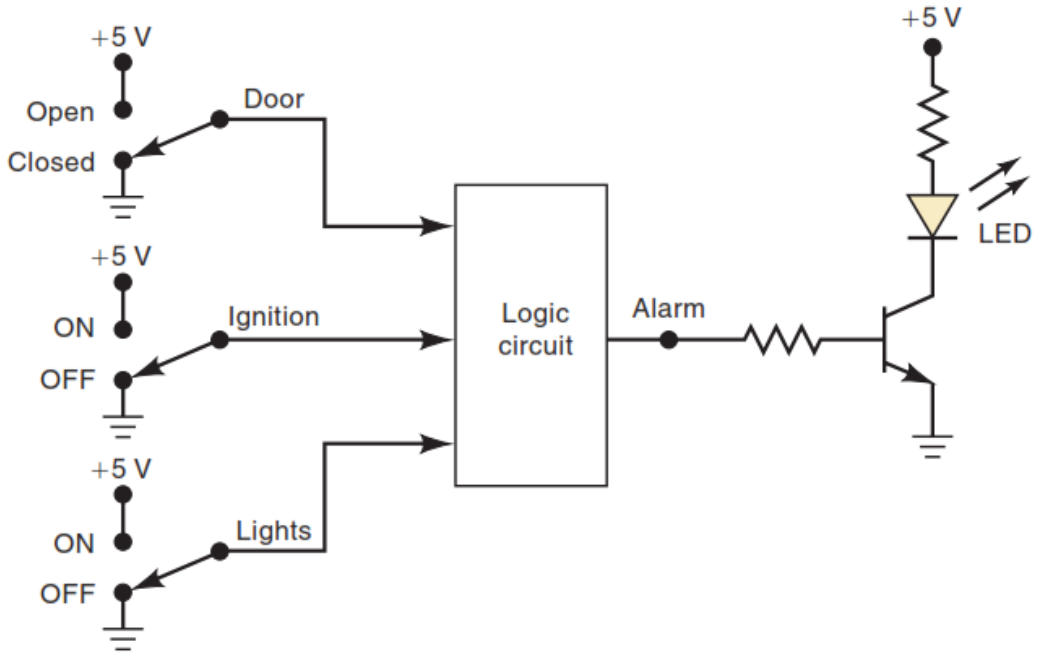
Implement the circuit using all NAND gates.

Implement the circuit using all NOR gates.

Example 2: Design a logic circuit that will produce a HIGH output whenever the binary number A3A2A1A0 is greater than 0010 and less than 1000

4-5 Karnaugh Map Method

Example 3:



An automobile alarm circuit used to detect certain undesirable conditions. The three switches are used to indicate the status of the door by the driver's seat, the ignition, and the headlights, respectively. Design the logic circuit with these three switches as inputs so that the alarm will be activated whenever either of the following conditions exists

- The headlights are on while the ignition is off.
- The door is open while the ignition is on.

4-5 Karnaugh Map Method

Example 4: Determine the minimum expression for each K map

	$\bar{C}\bar{D}$	$\bar{C}D$	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

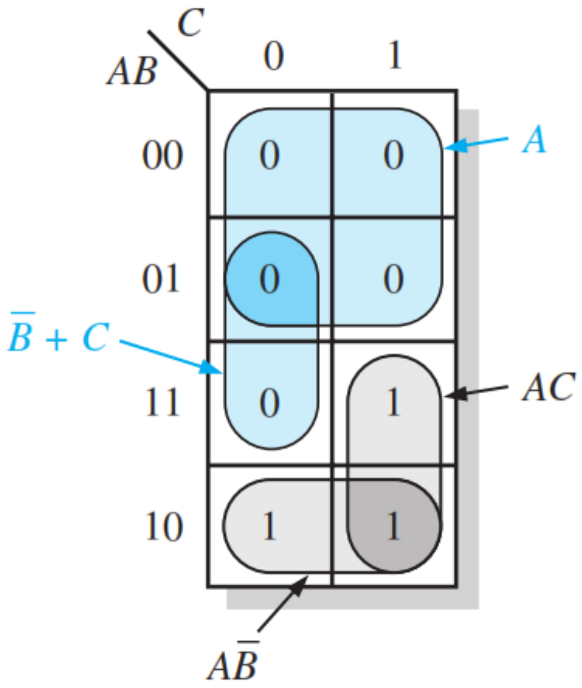
(b)

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

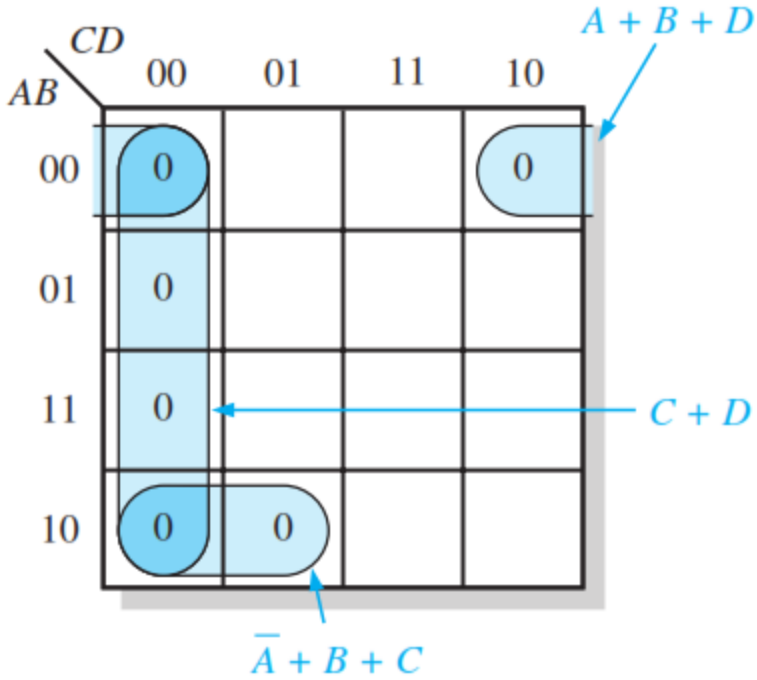
(c)

Karnaugh Map Simplification of POS Expressions

The process for minimizing a POS expression is basically the same as for an SOP expression except that you group 0s to produce minimum sum terms instead of grouping 1s to produce minimum product terms.

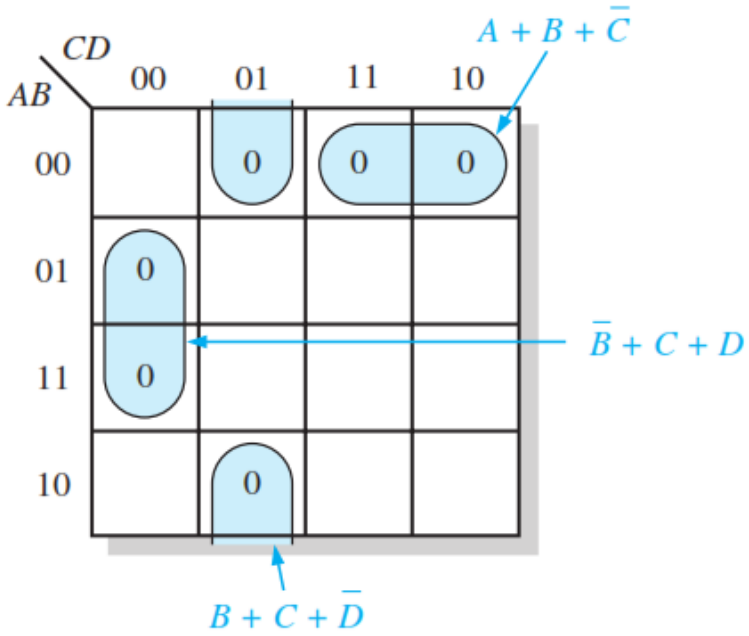


$A(\overline{B} + C)$

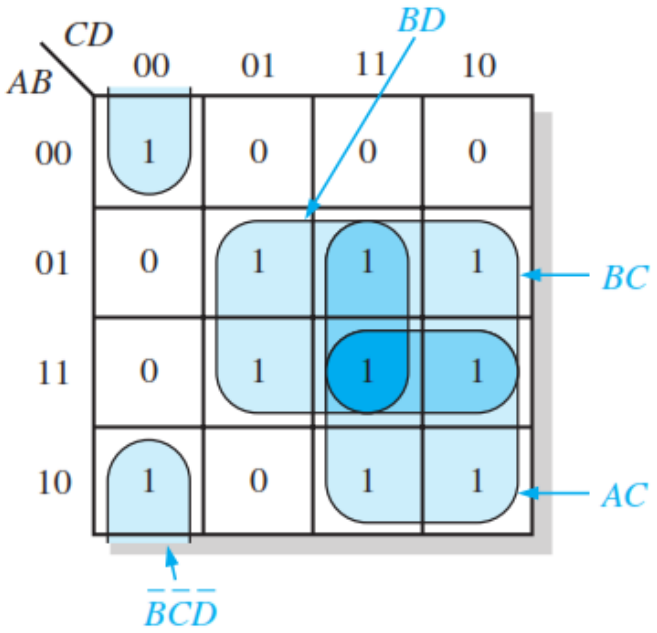


$(C + D)(A + B + D)(\overline{A} + B + C)$

4-5 Karnaugh Map Method



(a) Minimum POS: $(A + B + C)(\bar{B} + \bar{C} + D)(B + C + \bar{D})$

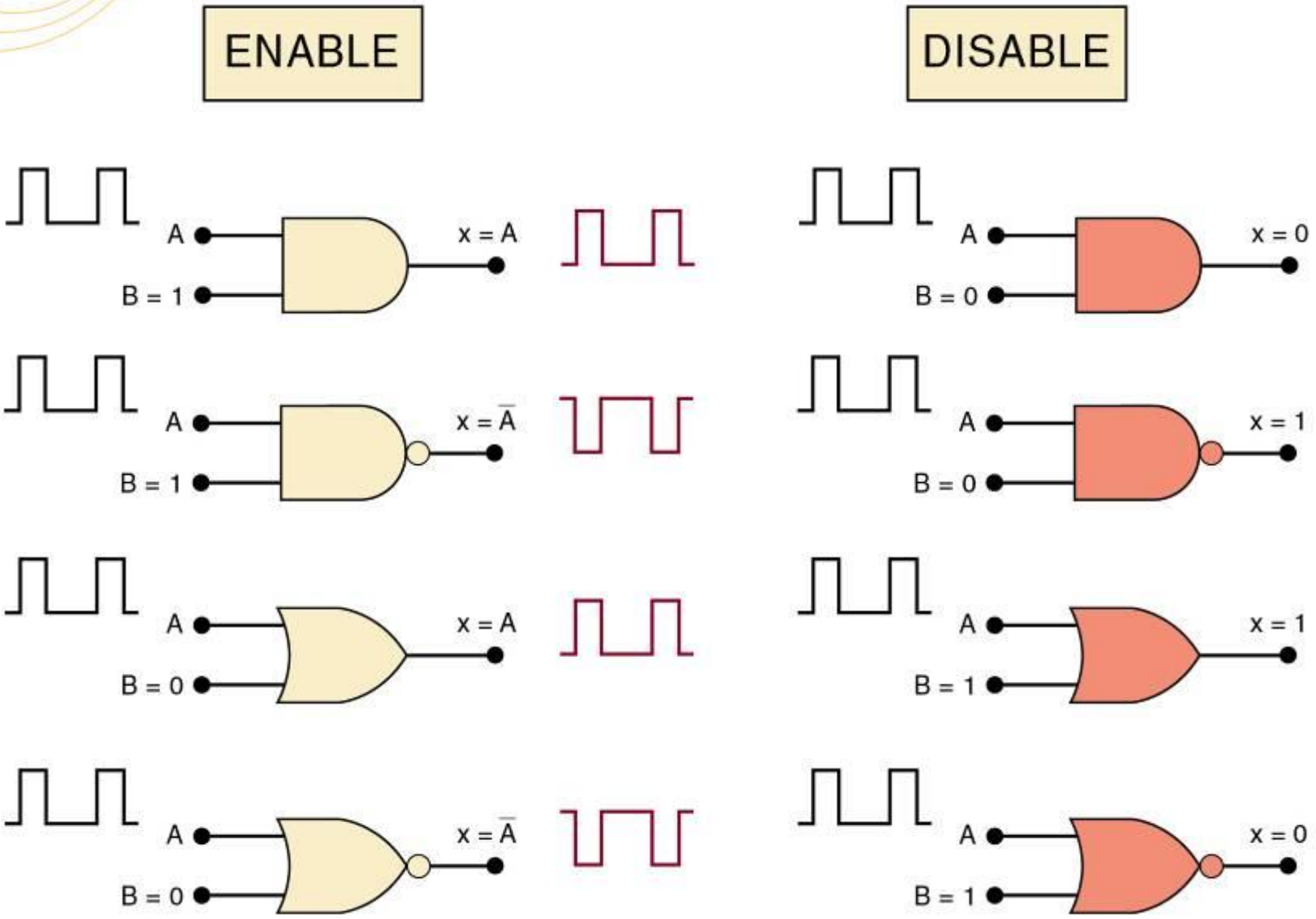


(c) Minimum SOP: $AC + BC + BD + \bar{B}\bar{C}\bar{D}$

4-6 Enable/Disable Circuits

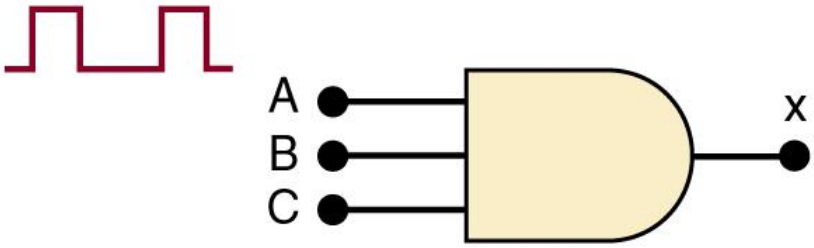
- Situations requiring enable/disable circuits occur frequently in digital circuit design.
  - A circuit is *enabled* when it *allows* the passage of an input signal to the output.
  - A circuit is *disabled* when it *prevents* the passage of an input signal to the output.

4-6 Enable/Disable Circuits

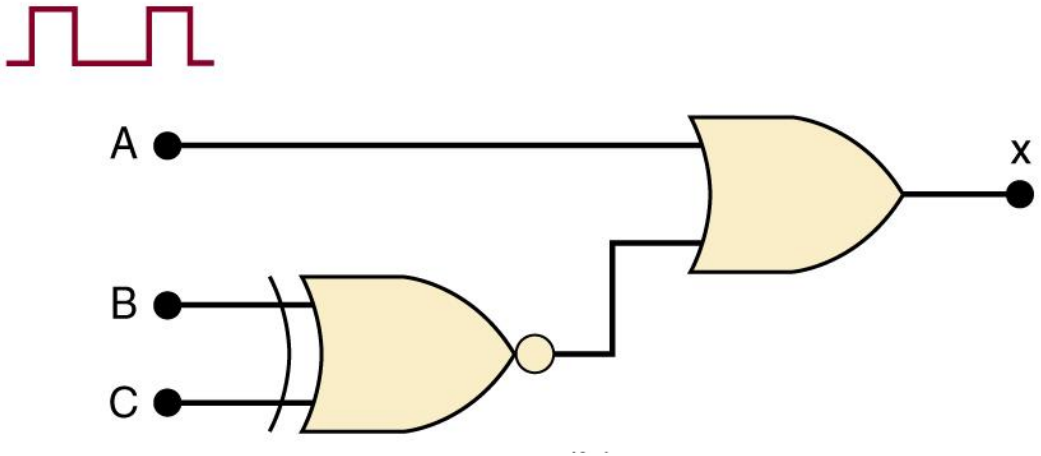


4-6 Enable/Disable Circuits

A logic circuit that will allow a signal to pass to output only when control inputs *B* and *C* are both HIGH. Otherwise, output will stay LOW.



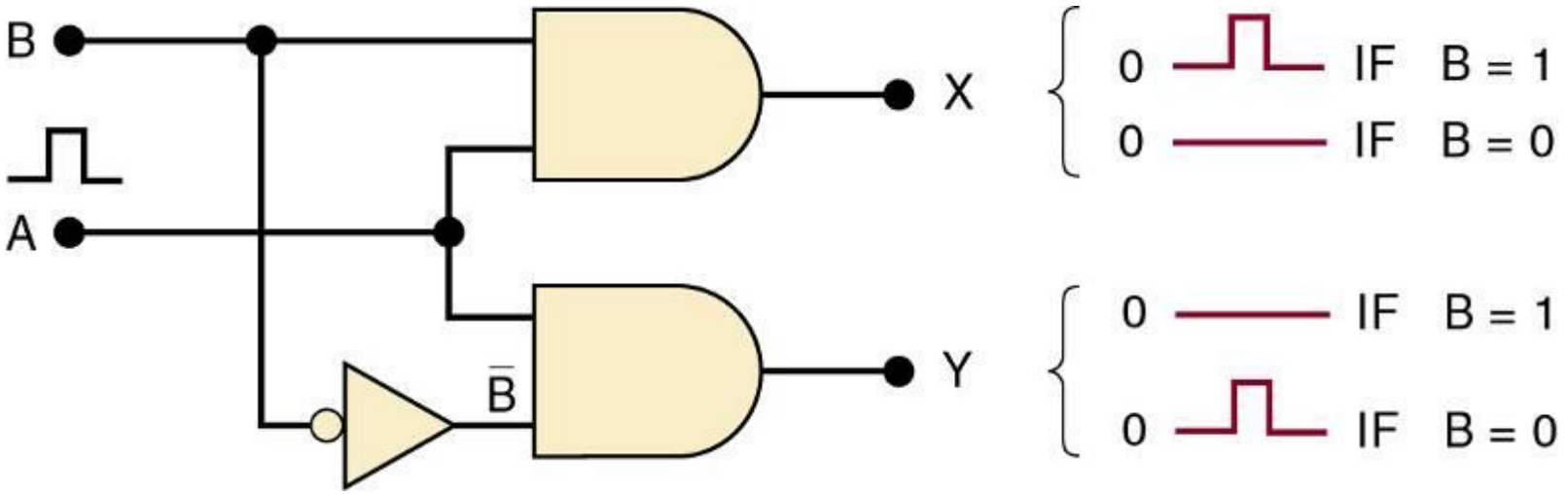
A logic circuit that will allow a signal to pass to output only when one, but *not* both control inputs are HIGH. Otherwise, output will stay HIGH.



A logic circuit with input signal *A*, control input *B*, and outputs *X* and *Y*, which operates as:

When *B* = 1, output *X* will follow input *A*, and output *Y* will be 0.

When *B* = 0, output *X* will be 0, and output *Y* will follow input *A*.



4-7 Basic Characteristics of Digital ICs

- IC “chips” consist of resistors, diodes & transistors fabricated on a piece of semiconductor material called a **substrate**.

Digital ICs are often categorized by complexity, according to the number of logic gates on the substrate.

Complexity	Gates per Chip
Small-scale integration (SSI)	Fewer than 12
Medium-scale integration (MSI)	12 to 99
Large-scale integration (LSI)	100 to 9999
Very large-scale integration (VLSI)	10,000 to 99,999
Ultra large-scale integration (ULSI)	100,000 to 999,999
Giga-scale integration (GSI)	1,000,000 or more

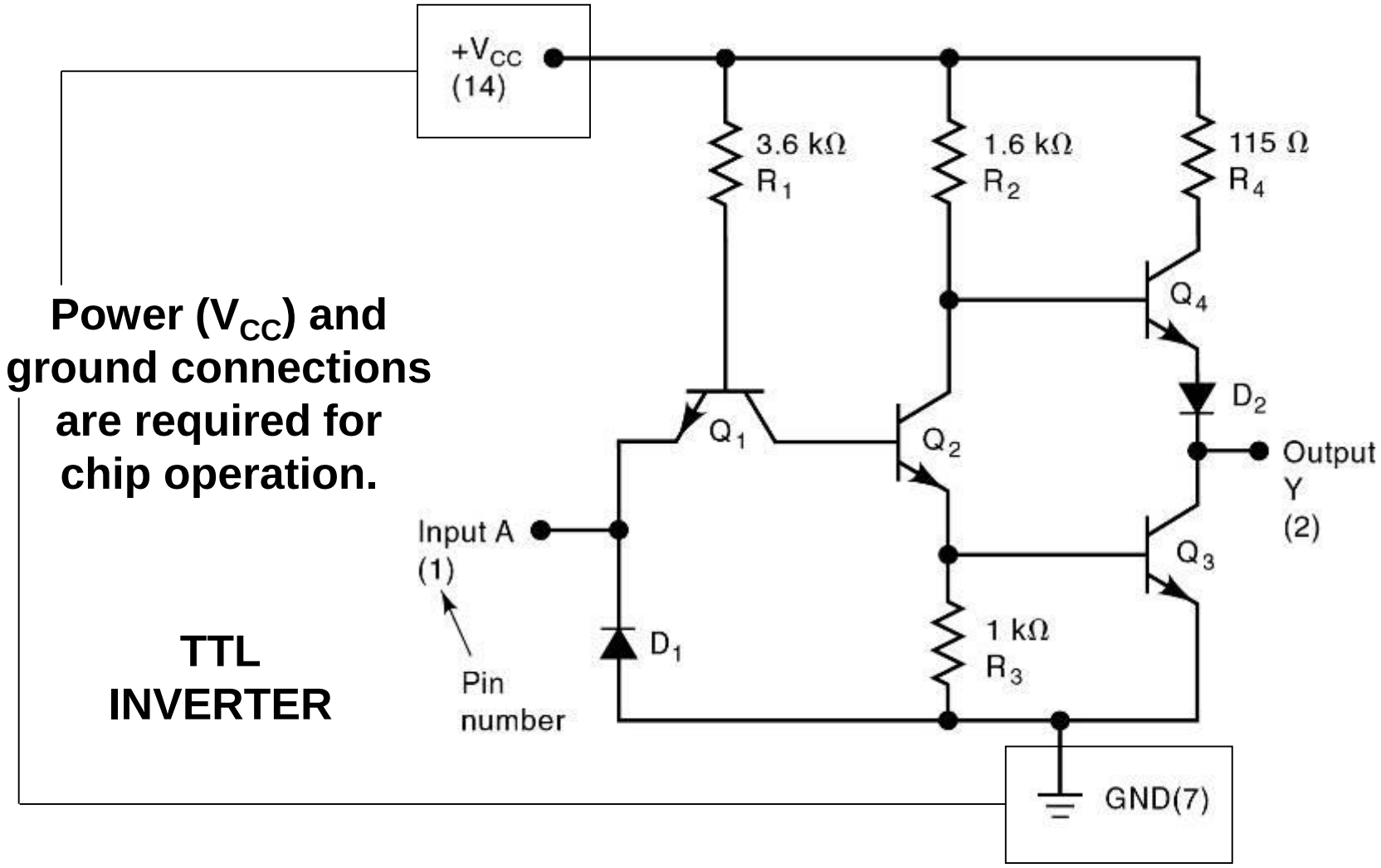
The transistor-transistor logic (TTL) family consists of subfamilies shown here:

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)

Differences between the TTL devices is limited to electrical characteristics such as power dissipation & switching speed. Pin layout and logic operations are the same.

4-7 Basic Characteristics of Digital ICs

V_{CC} for TTL devices is normally +5 V.



Power (V_{CC}) and ground connections are required for chip operation.

TTL INVERTER

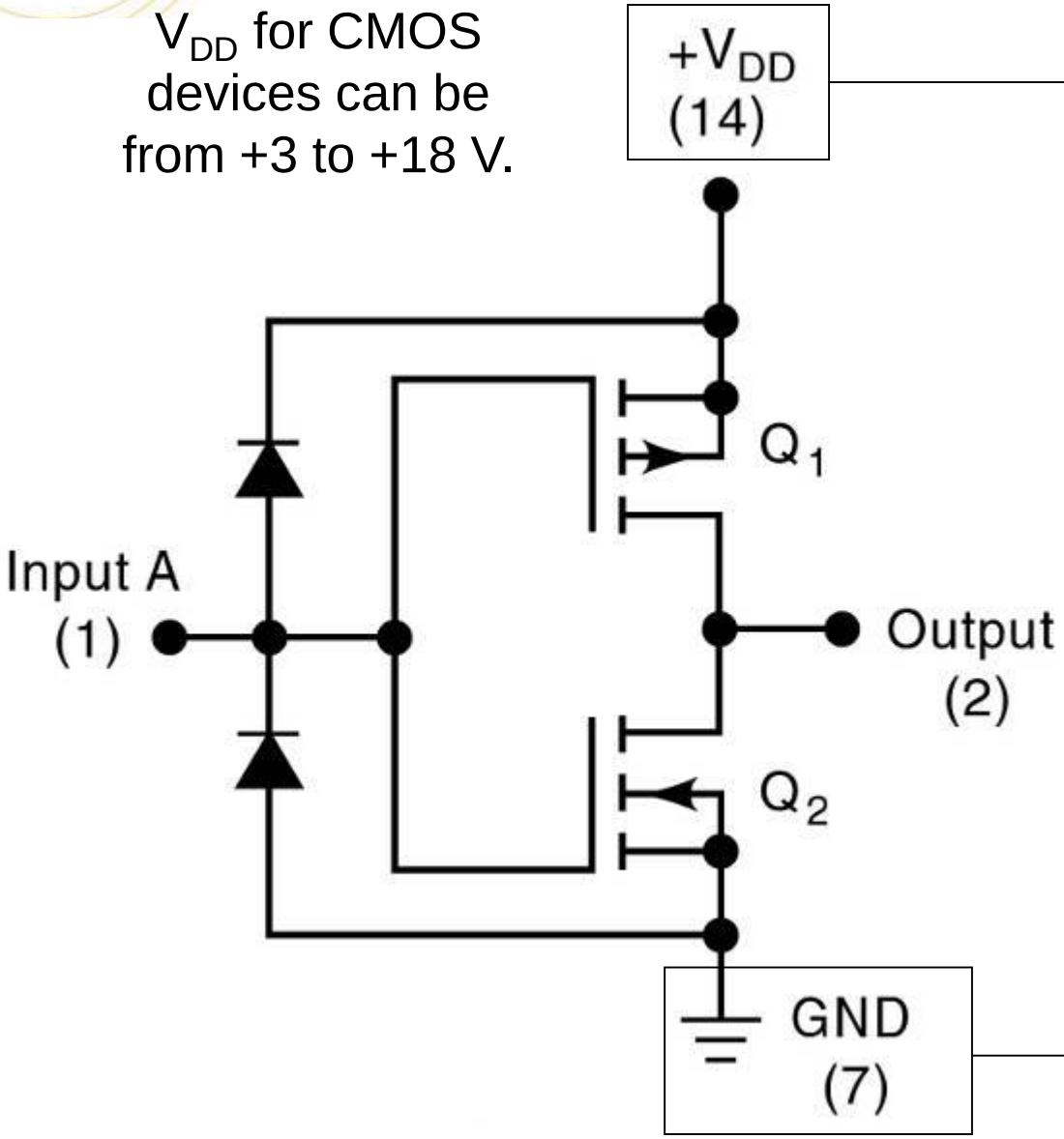
The Complimentary Metal-Oxide Semiconductor (CMOS) family consists of several series

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)

CMOS devices perform the same function as, but are not necessarily pin for pin compatible with TTL devices.

4-7 Basic Characteristics of Digital ICs

V_{DD} for CMOS devices can be from +3 to +18 V.

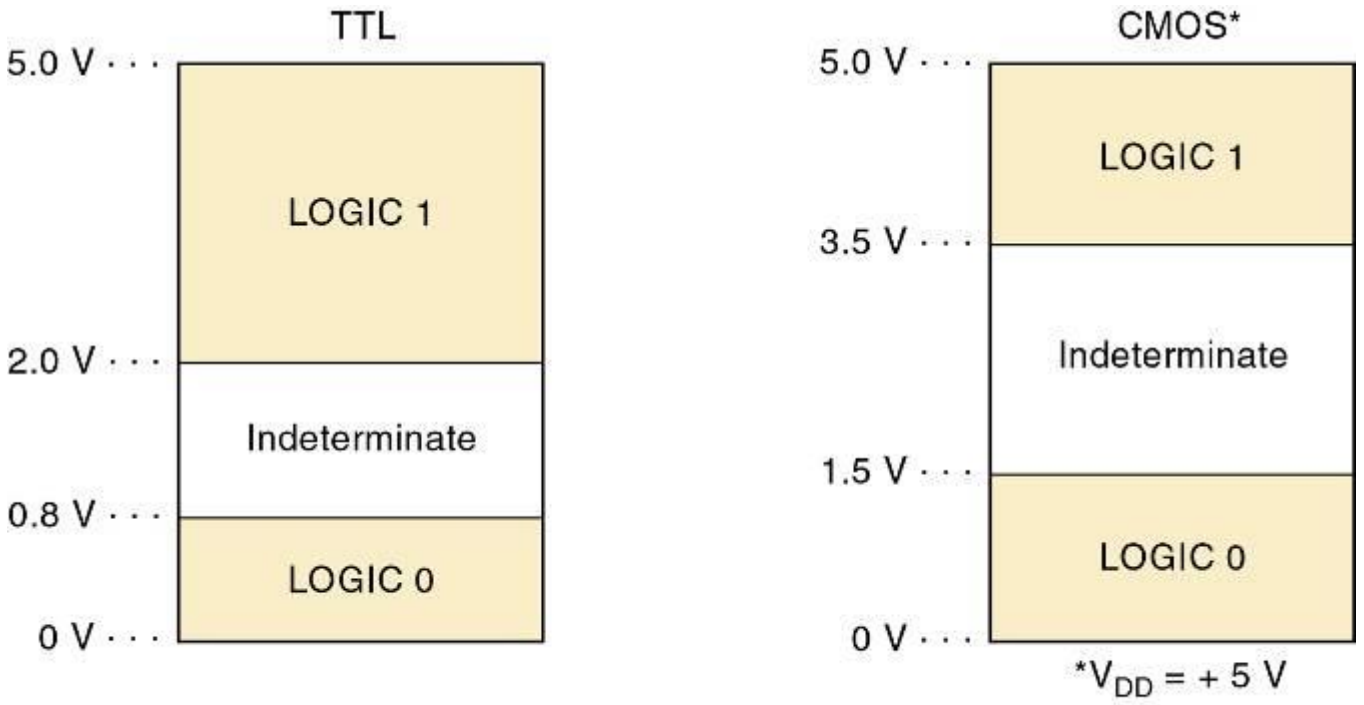


Power (V_{DD}) and ground connections are required for chip operation.

CMOS INVERTER

- Inputs not connected are said to be floating.
  - Floating TTL input acts like a logic 1.
    - Voltage measurement may appear *indeterminate,* but the device behaves 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.
  - The best practice is to tie all unused inputs.
    - Either high or low.

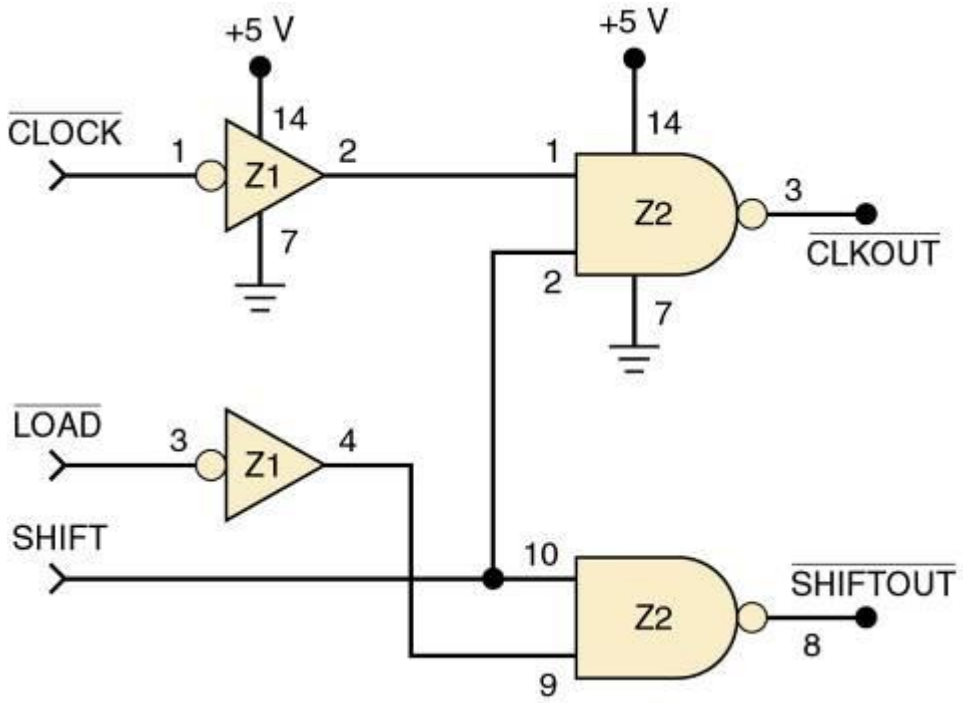
Voltages in the *indeterminate* range provide unpredictable results and should be avoided.



Logic levels for TTL and CMOS devices.

A connection diagram shows *all* electrical connections, pin numbers, IC numbers, component values, signal names, and power supply voltages.

IC	Type
Z1	74HC04 hex inverter
Z2	74HC00 quad nand



This circuit uses logic gates from two different ICs.

Each gate input & output pin number is indicated on the diagram, to easily reference any point in the circuit.

Power/ ground connections to each IC are shown.

4-8 Troubleshooting Digital Systems

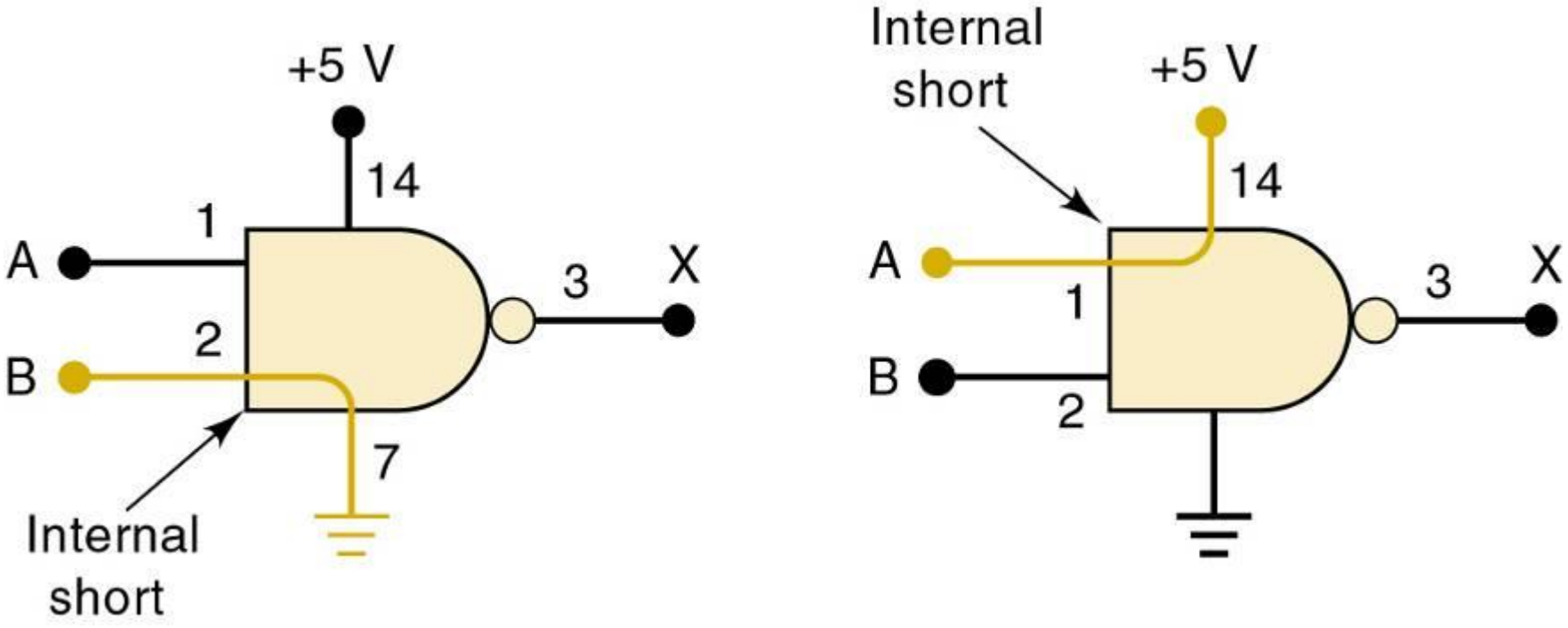
- Three basic steps in fixing a digital circuit or system that has a fault (failure):
  - **Fault detection**—determine operation to expected operation.
  - **Fault isolation**—test & measure to isolate the fault.
  - **Fault correction—**repair the fault.
- The basic troubleshooting tools are the logic probe, oscilloscope, and logic pulser.

4-9 Internal Digital IC Faults

- Most common internal failures:
  - Malfunction in the internal circuitry.
    - Outputs do not respond properly to inputs.
    - Outputs are unpredictable.
  - Inputs or outputs shorted to ground or V_{CC} .
    - The input will be stuck in LOW or HIGH state.
  - Inputs or outputs open-circuited .
    - An open output will result in a floating indication.
    - Floating input in a TTL will result in a HIGH output.
    - Floating input in a CMOS device will result in erratic or possibly destructive output.
  - Short between two pins (other than ground or V_{CC}).
    - The signal at those pins will always be identical.

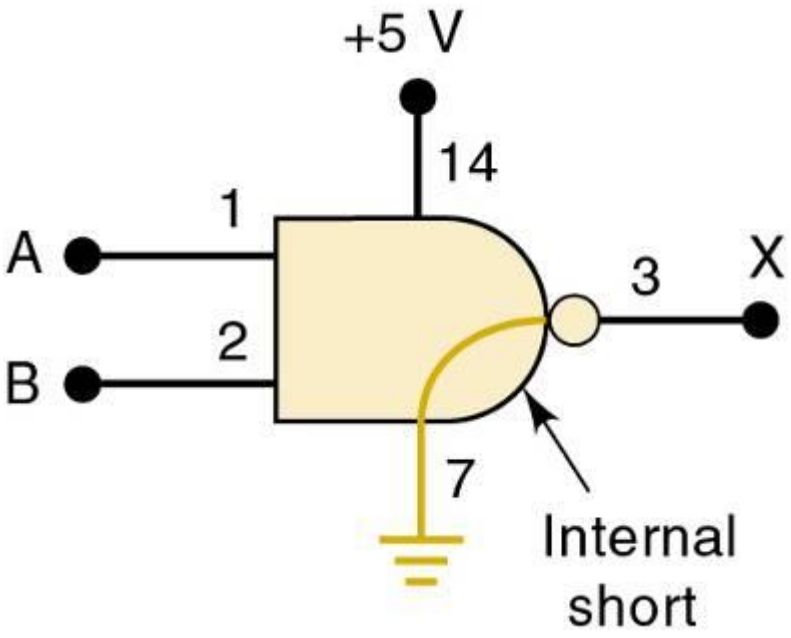
These two types of failures force the input signal at the shorted pin to stay in the same state.

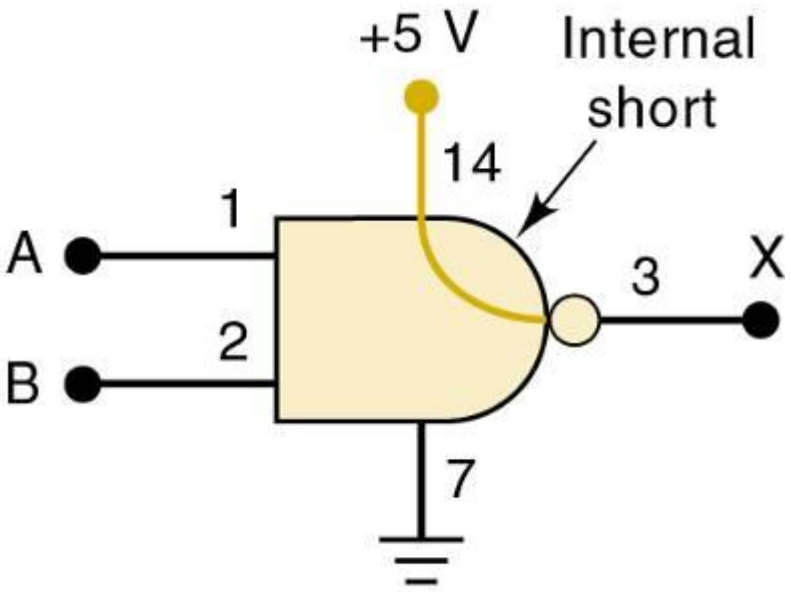
Left—IC input internally shorted to ground.
Right—IC input internally shorted to supply voltage.



These two types of failures do not affect signals at the IC inputs.

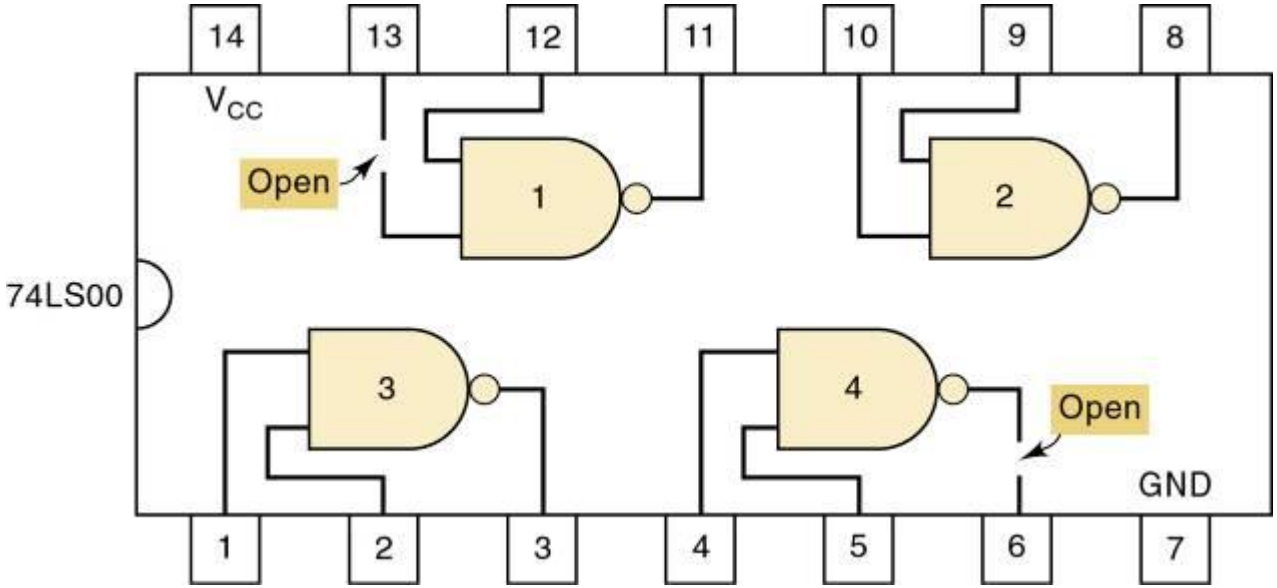
Left—IC output internally shorted to ground.
Right—IC output internally shorted to supply voltage.





4-9 Internal Digital IC Faults

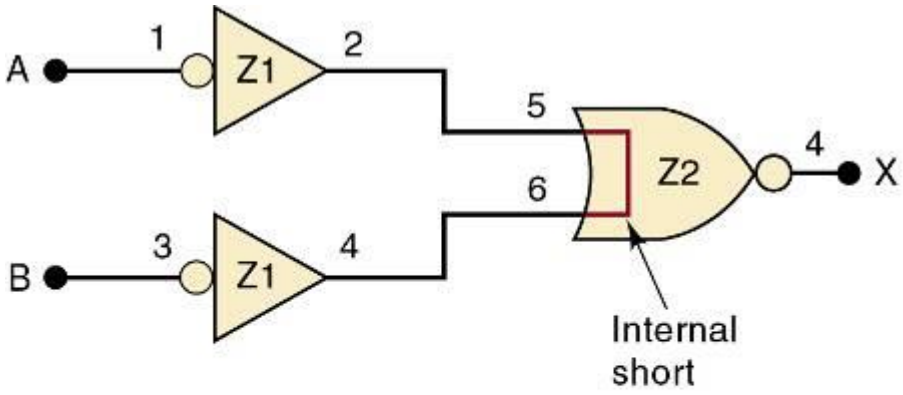
An IC with an internally open input will not respond to signals applied to that input pin.

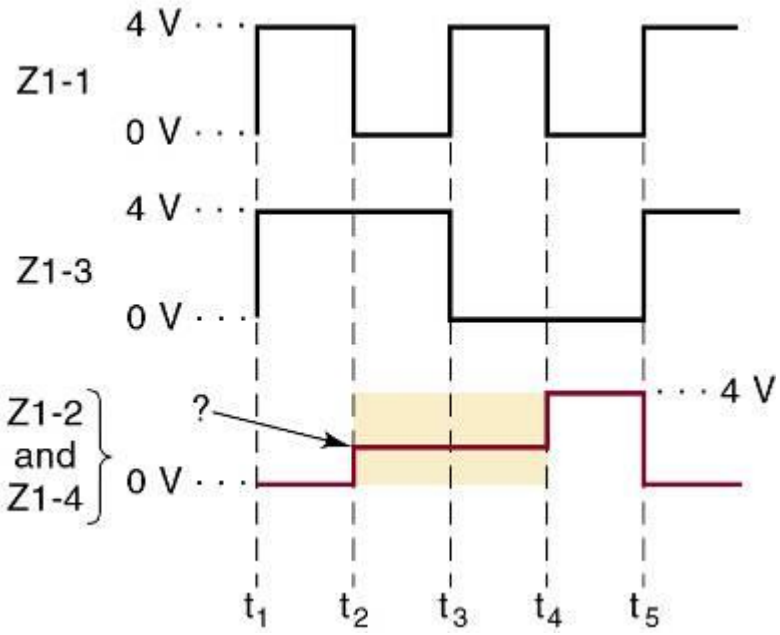


An internally open output will produce an unpredictable voltage at that output pin.

4-9 Internal Digital IC Faults

An internal short between two pins of an IC will force the logic signals at those pins always to be identical.





When two input pins are internally shorted, the signals driving these pins are forced to be identical, and usually a signal with three distinct levels results.

4-10 External Faults

- **Open signal lines**—signal prevented from moving between points—can be caused by:
  - Broken wire.
  - Poor connections (solder or wire-wrap).
  - Cut or crack on PC board trace.
  - Bent or broken IC pins.
  - Faulty IC socket.
- This type of fault can be detected visually and verified with an ohmmeter between the points in question.

4-10 External Faults

- **Shorted signal lines**—the same signal appears on two or more pins—and V_{CC} or ground may also be shorted, caused by:
  - Sloppy wiring.
  - Solder bridges.
  - Incomplete etching.
- This type of fault can be detected visually and verified with an ohmmeter between the points in question.

4-10 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.
- Verify that power supplies provide the specified range of voltages and are properly grounded.
  - Use an oscilloscope to verify that AC ripple is not present and verify that DC voltages stay regulated.
- Some ICs are more tolerant of power variations and may operate properly—others do not.
  - Check power and ground levels at each IC that appears to be operating incorrectly.

4-10 External Faults

- **Output loading**—caused by connecting too many inputs to the output of an IC, exceeding output current rating.
  - Output voltage falls into the indeterminate range.
    - Called *loading* the output signal.
  - Usually a result of poor design or bad connection.