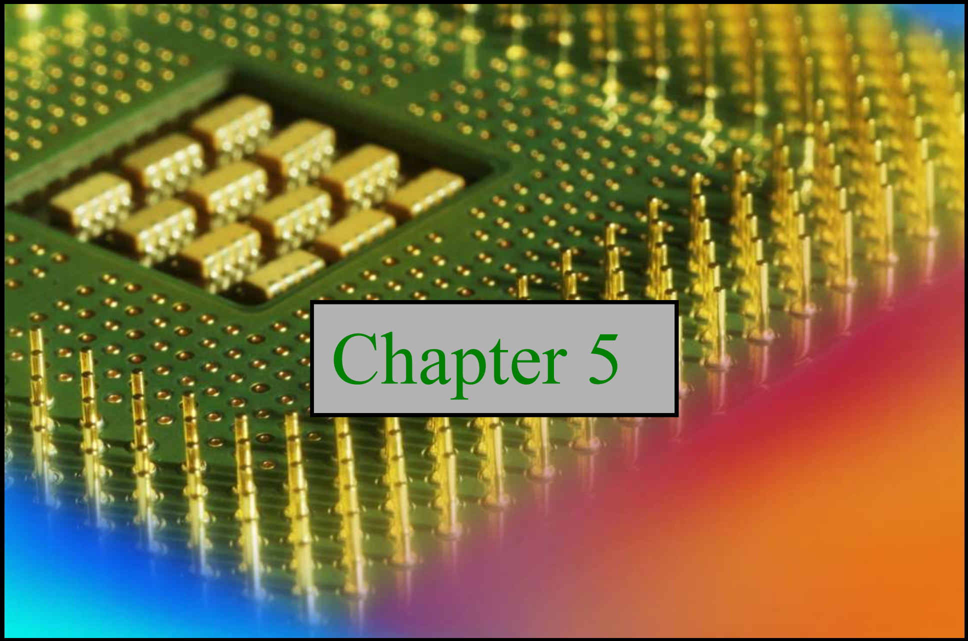
Digital Fundamentals

Tenth Edition

Floyd

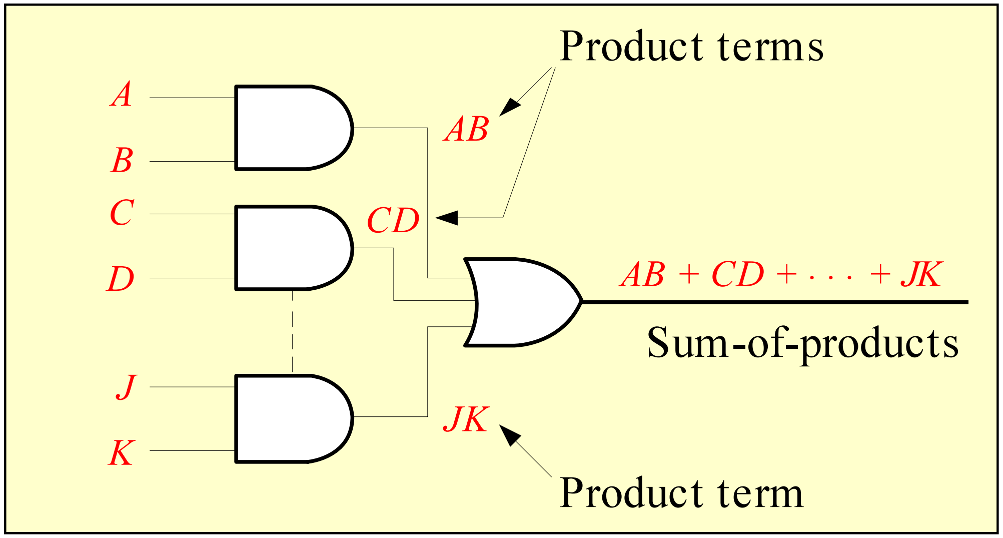


Chapter 5

Summary

Combinational Logic Circuits

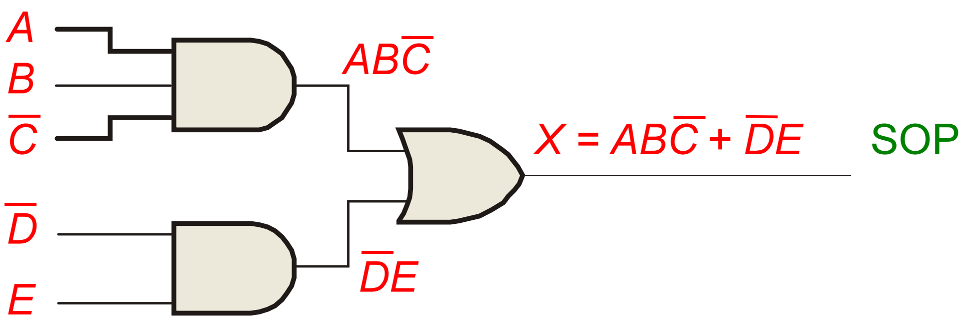
In Sum-of-Products (SOP) form, basic combinational circuits can be directly implemented with AND-OR combinations if the necessary complement terms are available.



Summary

AND-OR Logic

An example of an SOP implementation is shown. The SOP expression is an AND-OR combination of the input variables and the appropriate complements.

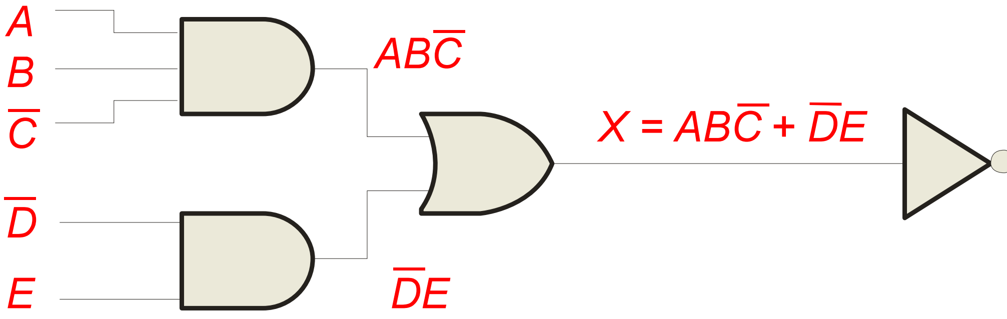


Summary

AND-OR-Invert Logic

When the output of a SOP form is inverted, the circuit is called an AND-OR-Invert circuit. The AOI configuration lends itself to product-of-sums (POS) implementation.

An example of an AOI implementation is shown. The output expression can be changed to a POS expression by applying DeMorgan's theorem twice.



$X = \overline{AB\overline{C} + \overline{D}E}$ AOI

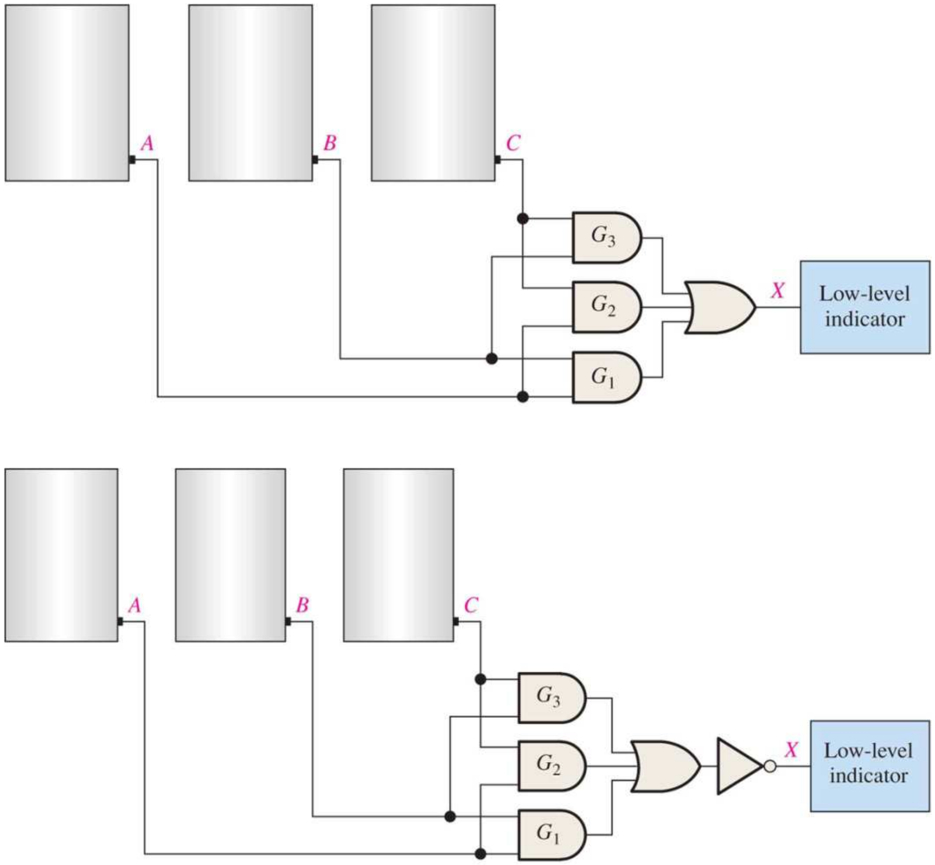
$X = (\overline{AB\overline{C}})(\overline{\overline{D}E})$ DeMorgan

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

Summary

Example Logic

AO or AOI can be used for positive- or negative-logic control circuits



Summary

Exclusive-OR Logic

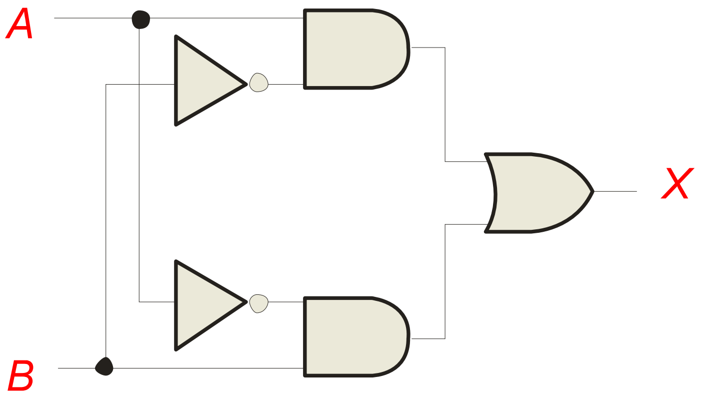
The truth table for an exclusive-OR gate is

Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

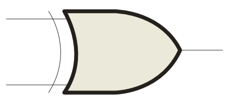
Notice that the output is HIGH whenever *A* and *B* disagree.

The Boolean expression is $X = \overline{A}B + A\overline{B}$

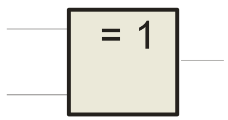
The circuit can be drawn as



Symbols:



Distinctive shape

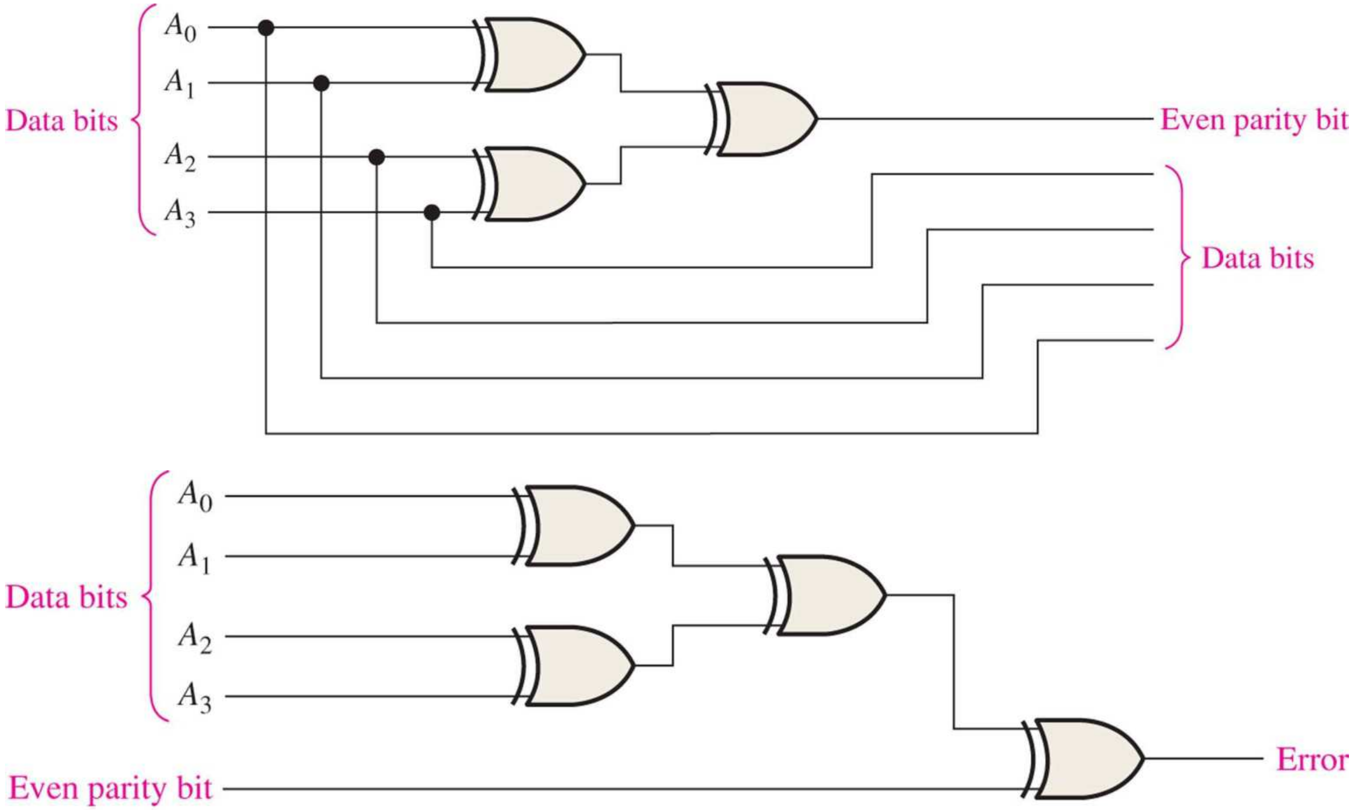


Rectangular outline

Summary

Example

Even-parity generator/checker



Summary

Exclusive-NOR Logic

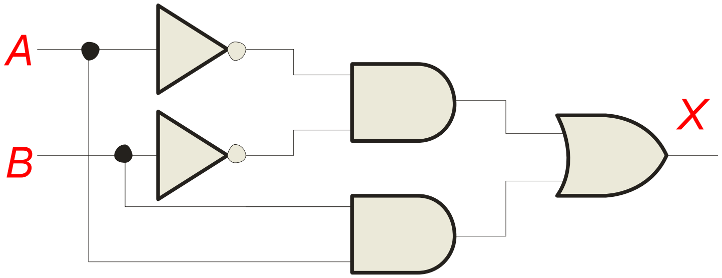
The truth table for an exclusive-NOR gate is

Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	1

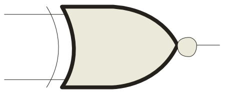
Notice that the output is HIGH whenever *A* and *B* agree.

The Boolean expression is $X = \overline{A}\overline{B} + AB$

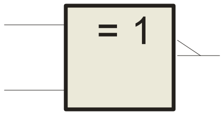
The circuit can be drawn as



Symbols:



Distinctive shape

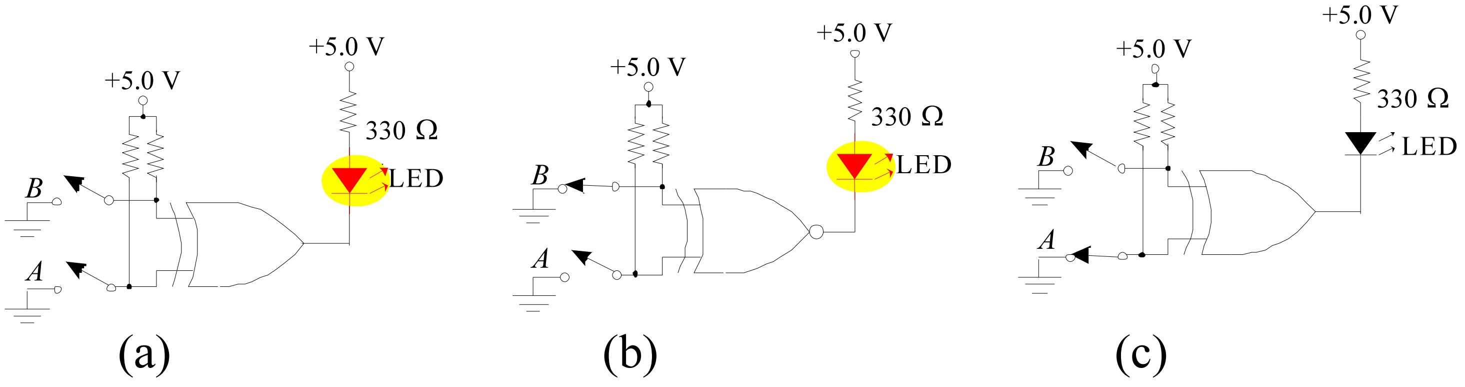


Rectangular outline

Summary

Example

For each circuit, determine if the LED should be on or off.



(a) (b) (c)

Solution

Circuit (a): XOR, inputs agree, output is LOW, LED is ON.

Circuit (b): XNOR, inputs disagree, output is LOW, LED is ON.

Circuit (c): XOR, inputs disagree, output is HIGH, LED is OFF.

Summary

Implementing Combinational Logic

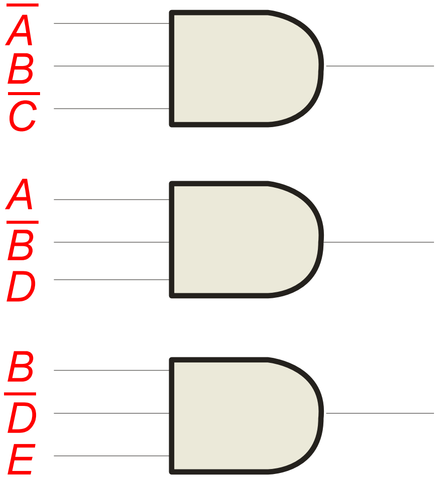
Implementing a SOP expression is done by first forming the AND terms; then the terms are ORed together.

Example

Show the circuit that will implement the Boolean expression $X = \overline{A}B\overline{C} + A\overline{B}D + B\overline{D}E$. (Assume that the variables and their complements are available.)

Solution

Start by forming the terms using three 3-input AND gates. Then combine the three terms using a 3-input OR gate.

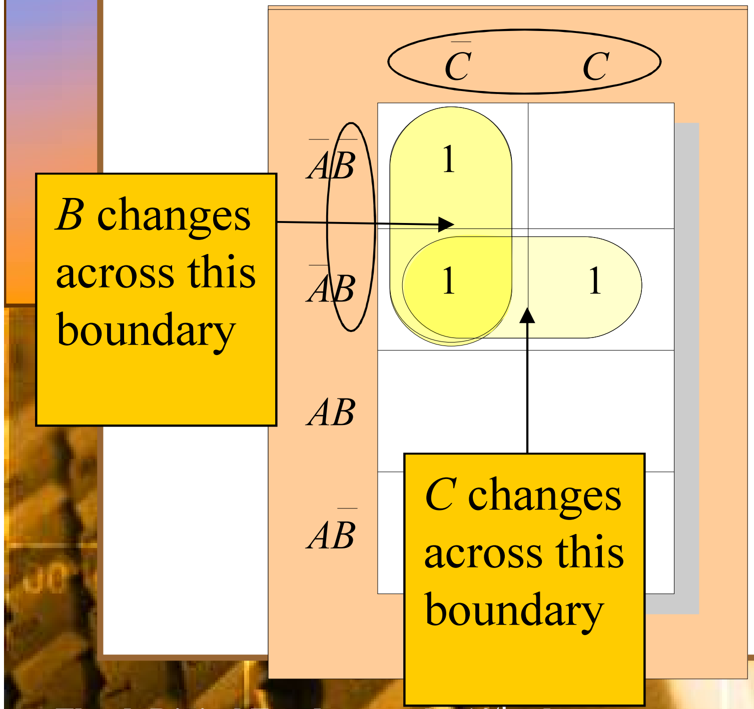


$X = \overline{A}B\overline{C} + A\overline{B}D + B\overline{D}E$

Karnaugh Map Implementation

For basic combinational logic circuits, the Karnaugh map can be read and the circuit drawn as a minimum SOP.

Example A Karnaugh map is drawn from a truth table. Read the minimum SOP expression and draw the circuit.



	$\overline{C}$	C
$\overline{A}\,\overline{B}$	1	
$\overline{A}B$	1	1
AB		
$A\overline{B}$		

B changes across this boundary

C changes across this boundary

Solution

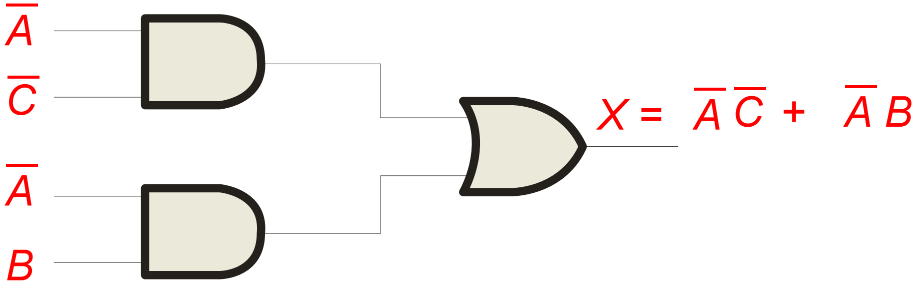
1. Group the 1's into two overlapping groups as indicated.
2. Read each group by eliminating any variable that changes across a boundary.
3. The vertical group is read $\overline{A}\,\overline{C}$.
4. The horizontal group is read $\overline{A}B$.

The circuit is on the next slide:

Summary

Solution *continued…*

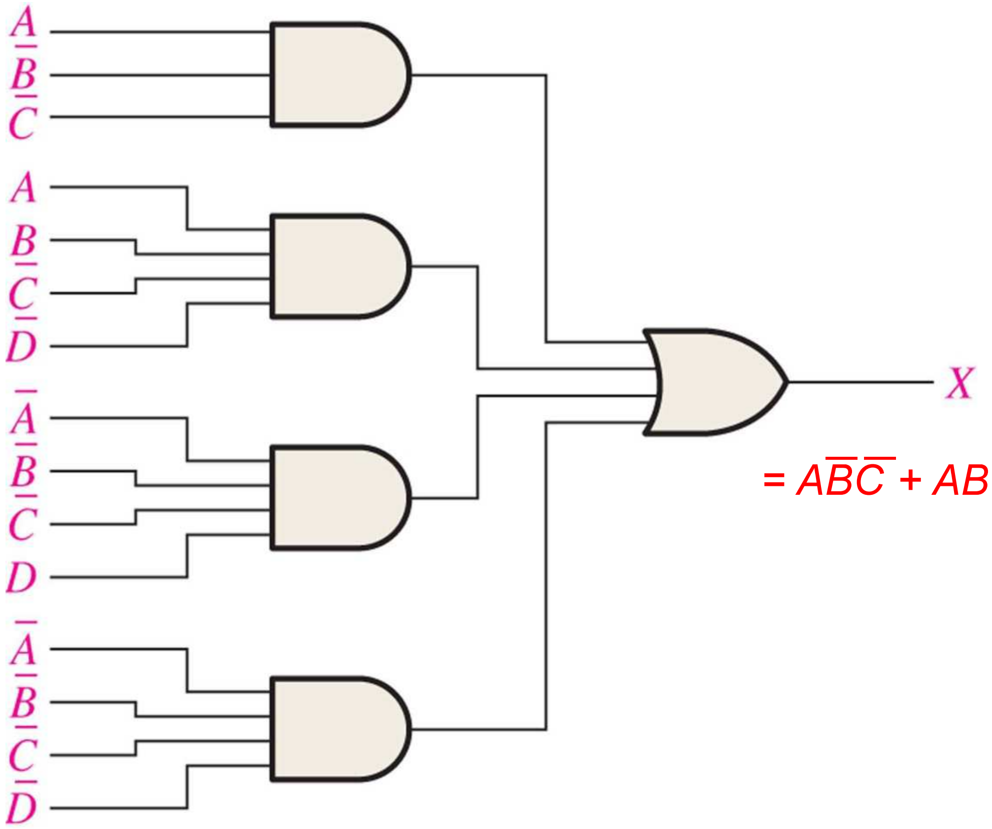
Circuit:



The result is shown as a sum of products.

Summary

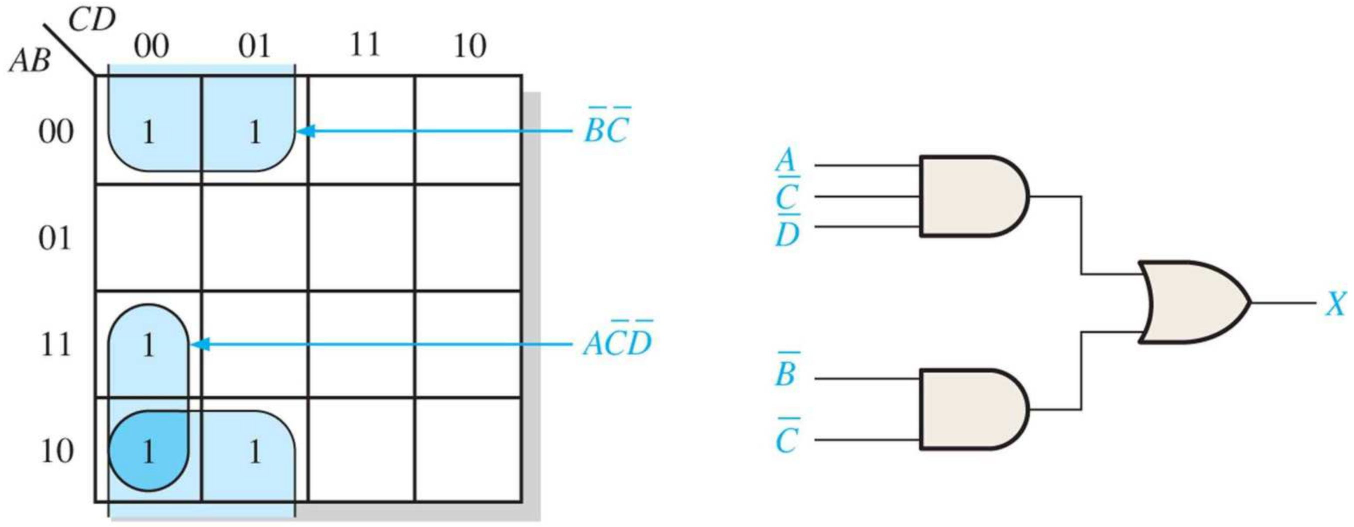
Example



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

Summary

Solution



AB \ CD	00	01	11	10
00	1	1		
01				
11	1			
10	1	1		

(a)

(b)

It is a simple matter to implement this form using only NAND gates as shown in the text and following example.

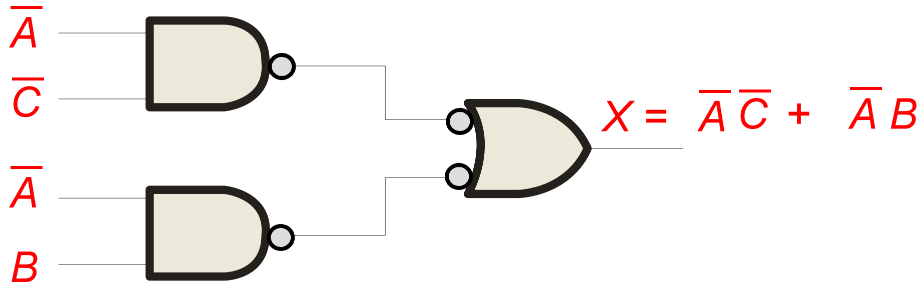
Summary

NAND Logic

Example Convert the circuit in the previous example to one that uses only NAND gates.

Solution

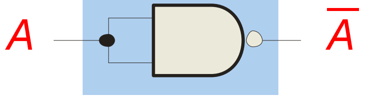
Recall from Boolean algebra that double inversion cancels. By adding inverting bubbles to above circuit, it is easily converted to NAND gates:



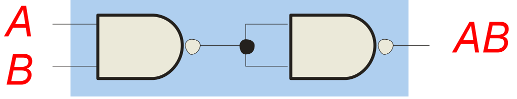
Summary

Universal Gates

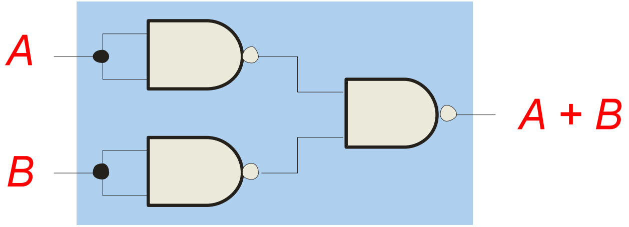
NAND gates are sometimes called **universal** gates because they can be used to produce the other basic Boolean functions.



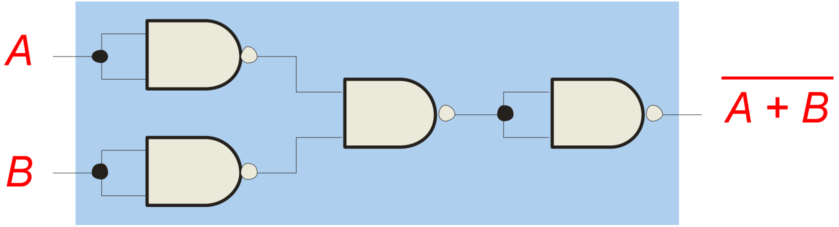
Inverter



AND gate



OR gate

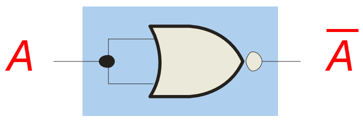


NOR gate

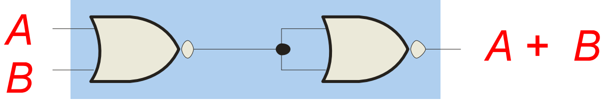
Summary

Universal Gates

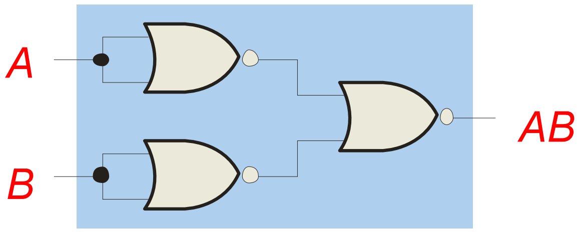
NOR gates are also **universal** gates and can form all of the basic gates.



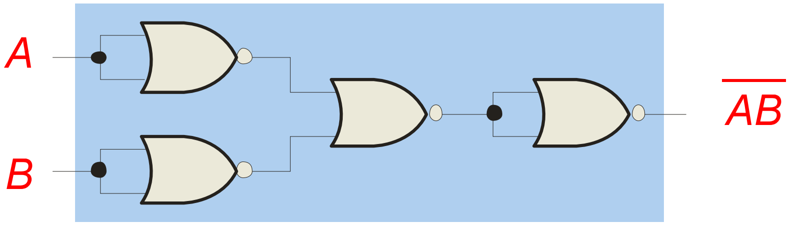
Inverter



OR gate



AND gate

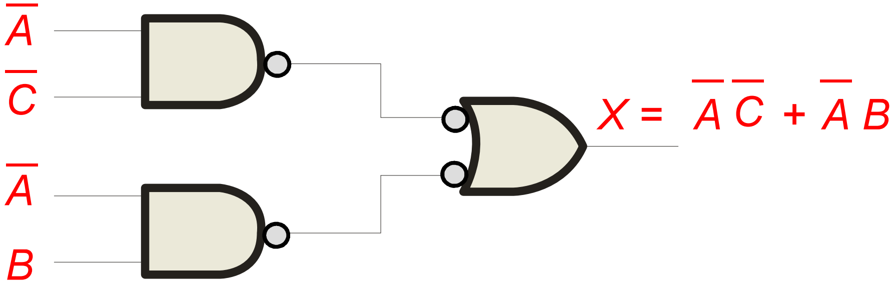


NAND gate

Summary

NAND Logic

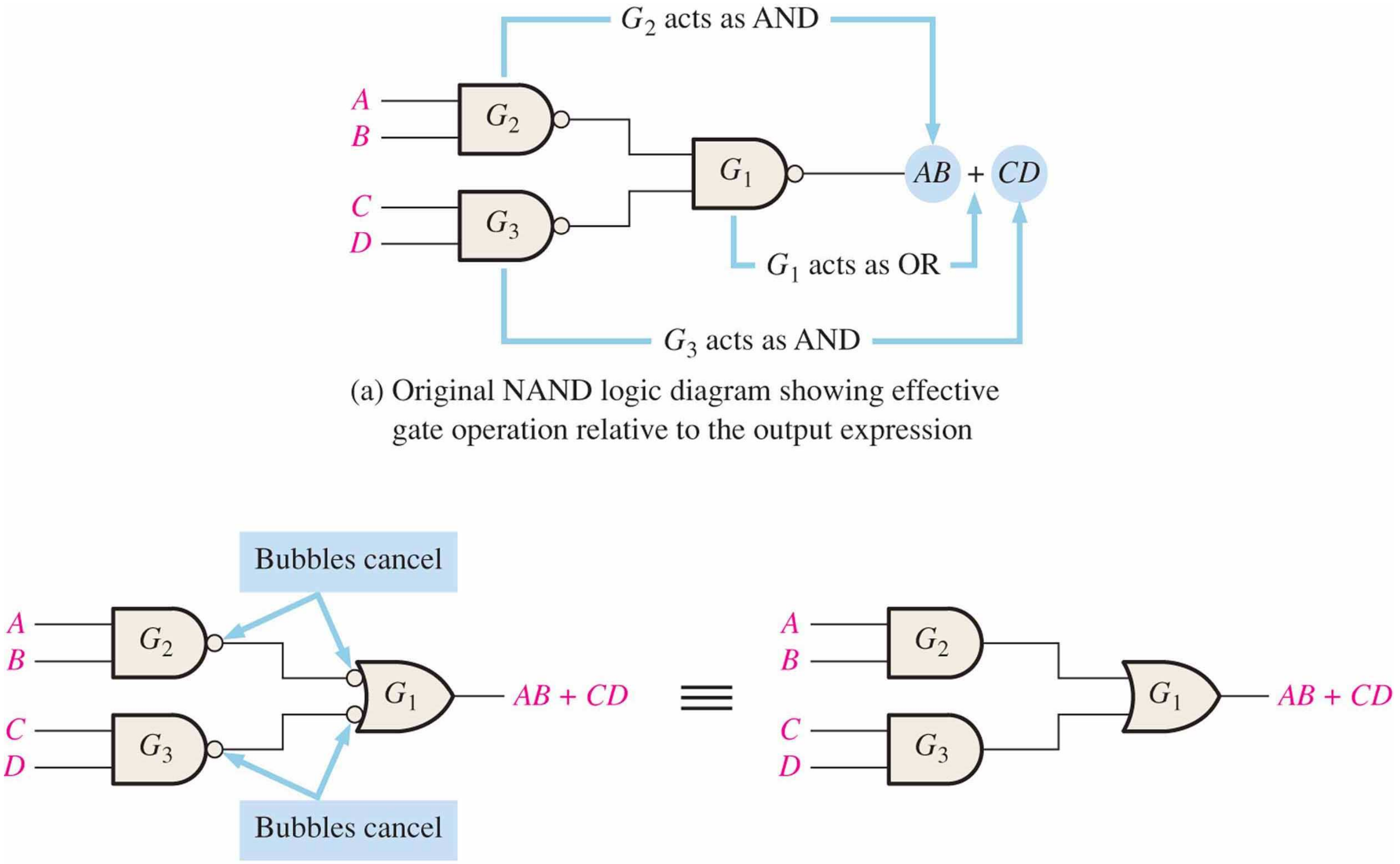
Recall from DeMorgan's theorem that $\overline{AB} = \overline{A} + \overline{B}$. By using equivalent symbols, it is simpler to read the logic of SOP forms. The earlier example shows the idea:



The logic is easy to read if you (mentally) cancel the two connected bubbles on a line.

Summary

NAND Logic



(a) Original NAND logic diagram showing effective gate operation relative to the output expression

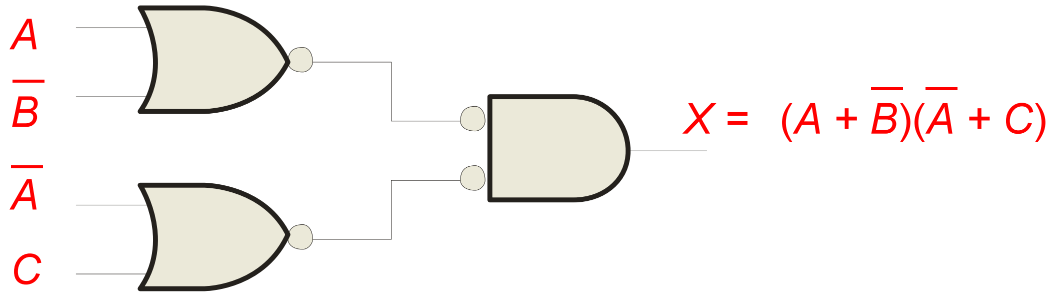
(b) Equivalent NAND/Negative-OR logic diagram

(c) AND-OR equivalent

Summary

NOR Logic

Alternatively, DeMorgan's theorem can be written as $\overline{A + B} = \overline{A}\,\overline{B}$. By using equivalent symbols, it is simpler to read the logic of POS forms. For example,

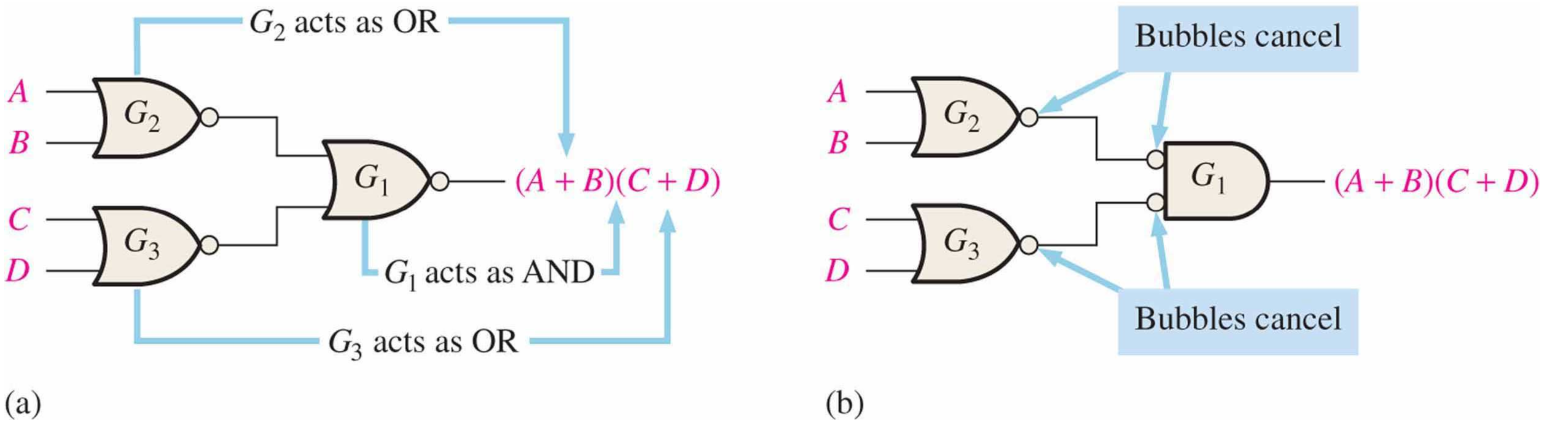


$X = (A + \overline{B})(\overline{A} + C)$

Again, the logic is easy to read if you cancel the two connected bubbles on a line.

Summary

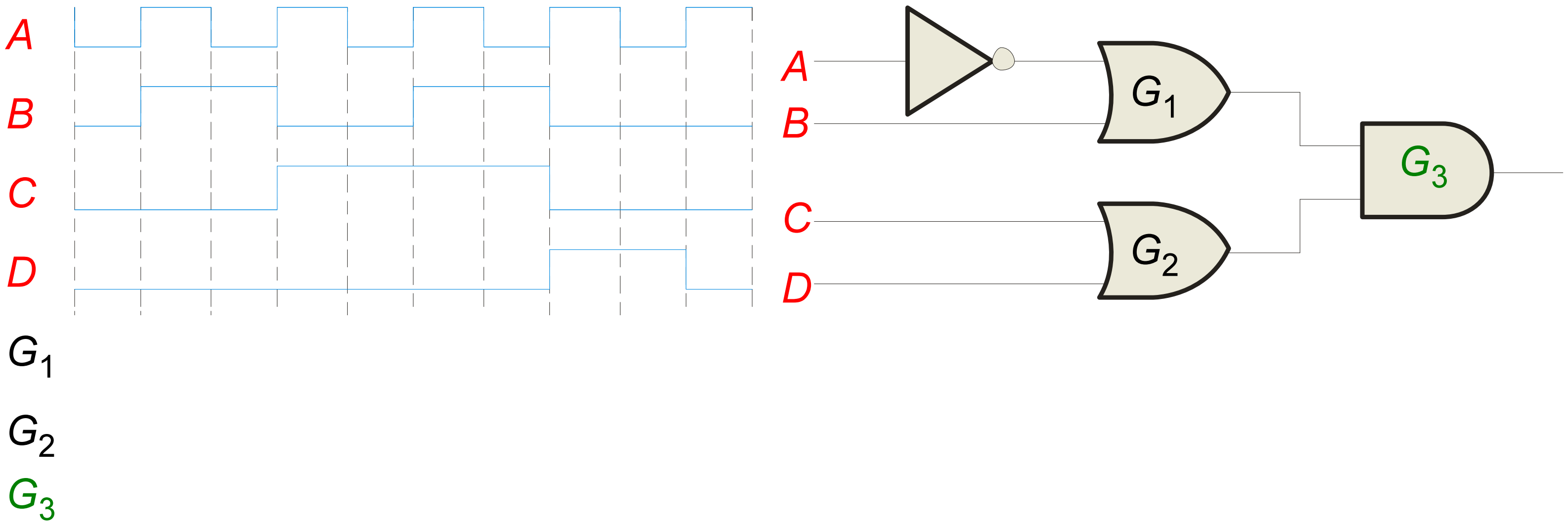
NOR Logic



Summary

Pulsed Waveforms

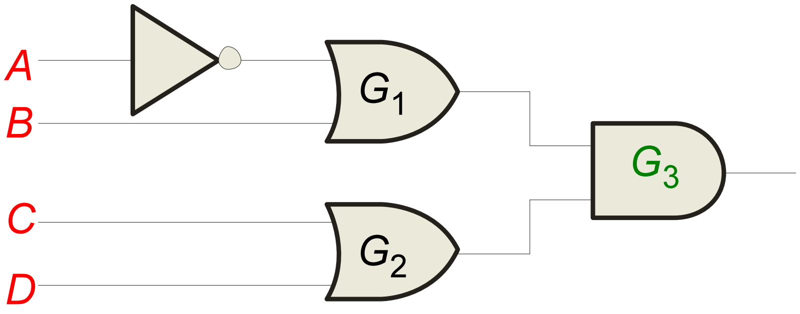
For combinational circuits with pulsed inputs, the output can be predicted by developing intermediate outputs and combining the result. For example, the circuit shown can be analyzed at the outputs of the OR gates:

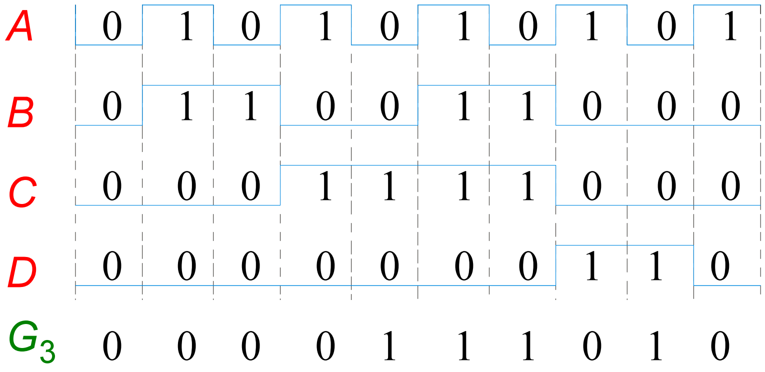


Summary

Pulsed Waveforms

Alternatively, you can develop the truth table for the circuit and enter 0's and 1's on the waveforms. Then read the output from the table.





A	0	1	0	1	0	1	0	1	0	1
B	0	1	1	0	0	1	1	0	0	0
C	0	0	0	1	1	1	1	0	0	0
D	0	0	0	0	0	0	0	1	1	0
G_3	0	0	0	0	1	1	1	0	1	0

Inputs				Output
A	*B*	*C*	*D*	*X*
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
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
1	1	1	0	1
1	1	1	1	1

Homework 8

- Chapter 5 (4, 12, 21, 32, 35)