Chapter 3

Logic Circuits

Describing logic circuits

- Boolean expression
- Truth table
- Logic diagram
- Timing diagram

Objectives

- Three basic logic operations: AND, OR, NOT
- Operation of logic circuits and construction of truth tables
- Timing diagrams for the various logic-circuit gates
- Boolean expression for the logic circuits
- Implement logic circuits using AND, OR, NOT
- Simplify logic expressions
  - DeMorgan's theorem
- Using NAND or NOR to implement a circuit
- Alternate gate symbols vs. standard logic-gate symbols
- Active high & Active low

Boolean Constants and Variables

- 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
- 0/1 and Low/High are used most of the time
- Three Logic operations: AND, OR, NOT

Logic 0	Logic 1
False	True
Off	On
LOW	HIGH
No	Yes
Open switch	Closed switch

Logic functions

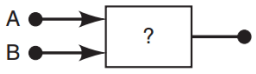
- The function $x = f(A_1, A_2, \ldots, A_N)$ is called a N-variable logic function if $A_1, A_2, \ldots, A_N$ are logic variables and $x \in X$, with $X=\{0,1\}$.
  - The right side: logic expression.
  - The left side: logic variable
- Logic expression (Boolean expression) is an algebraic expression with the operands are logic variables and operators are basic logic operations

Logic functions and logic circuits

- Logic functions → Logic circuits
- Logic circuits → Logic functions
- The inputs of logic circuits ↔ Logic variables
- The outputs of logic circuits ↔ The value of function
- Logic Gates
  - A logic circuit corresponding to a basic logic operation of Boolean algebra is called the logic gate
  - Logic gate is constructed from diodes, transistors, and resistors whose output is the result of a basic logic operation (OR, AND, NOT) performed on the inputs

Truth Table

- How a logic circuit's output depends on the logic levels present at the inputs
  - List all the outputs of logic circuit based on the a combination of input values



Inputs / Output

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

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

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

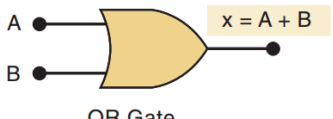
7

OR Operation with OR gates

- Truth Table
  - x is a logic 1 for every combination of input levels where one or more inputs are 1
  - Boolean expression x = A + B
    - x equals A OR B
    - To describe this circuit in the English language we could say that x is true (1) WHEN A is true (1) OR B is true (1)
- OR Gate
  - is a circuit that has two or more inputs and whose output is equal to the OR combination of the inputs
  - its output is HIGH (logic 1) if either input A or B or both are at a logic 1 l

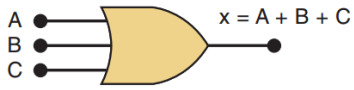
OR

A	B	x = A + B
0	0	0
0	1	1
1	0	1
1	1	1



OR Gate

Symbol and Truth Table for a 3-input OR Gate

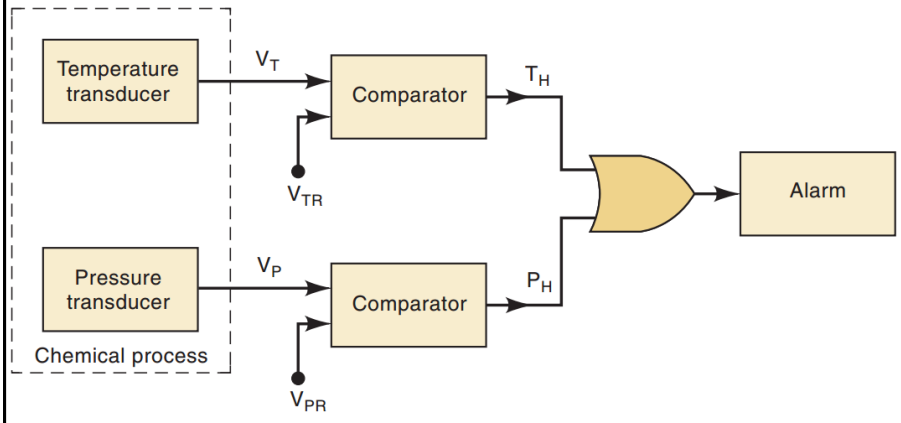


A	B	C	x = A + B + C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

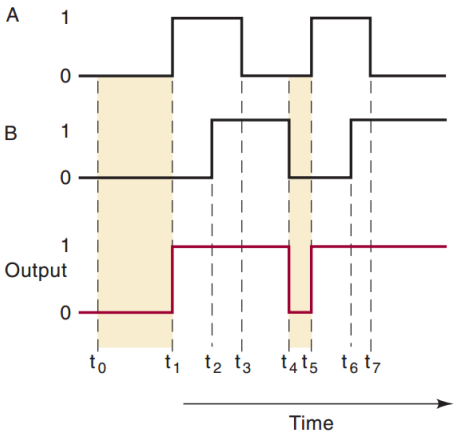
Summary of OR Operation

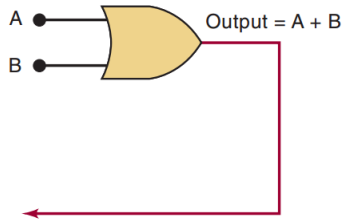
- The OR operation produces a result (output) of 1 whenever any input is a 1. Otherwise the output is 0
- An OR gate is a logic circuit that performs an OR operation on the circuit's input
- The expression x=A+B is read as “x equals A OR B”

Example of the Use of an OR Gate in an Alarm System



OR Gate Timing Diagram



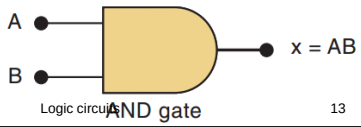


AND Operation with AND Gates

- Truth Table
  - x is a logic 1 only when both A and B are at the logic 1 level. For any case where one of the inputs is 0, the output is 0.
  - Boolean expression x = A.B
    - x equals A AND B
- AND gate
  - is a digital circuit performing the AND operation
  - An AND gate output will be 1 only for the case when all inputs are 1; for all other cases the output will be 0

AND

A	B	x = A • B
0	0	0
0	1	0
1	0	0
1	1	1

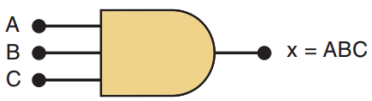


Logic circuits AND gate

13

Truth Table and Symbol for a 3-input AND Gate

A	B	C	x = ABC
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

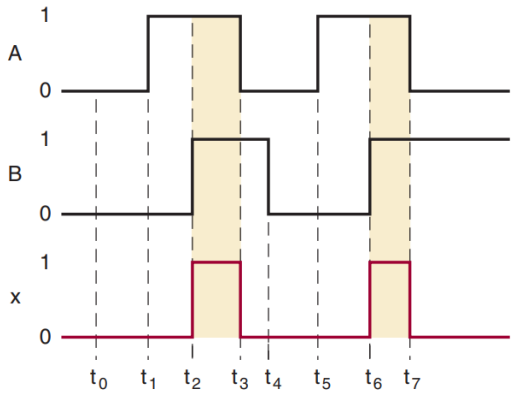


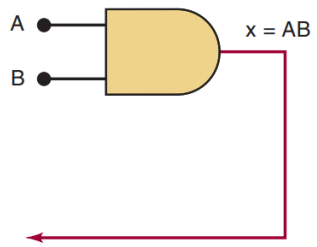
Summary of the AND Operation

- The AND operation is performed the same as ordinary multiplication of 1s and 0s
- An AND gate is a logic circuit that performs the AND operation on the circuit's inputs
- An AND gate output will be 1 only for the case when all inputs are 1; for all other cases the output will be 0
- The expression x=AB is read as "x equals A AND B"

AND Gate Timing Diagram

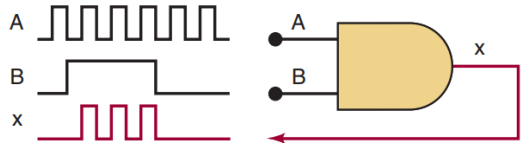
- Determine the output x from the AND gate for the given input waveforms.





AND Gate Timing Diagram

- Determine the output waveform for the AND gate

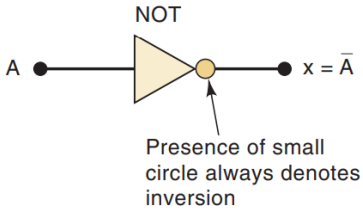


NOT Operation with NOT Gate

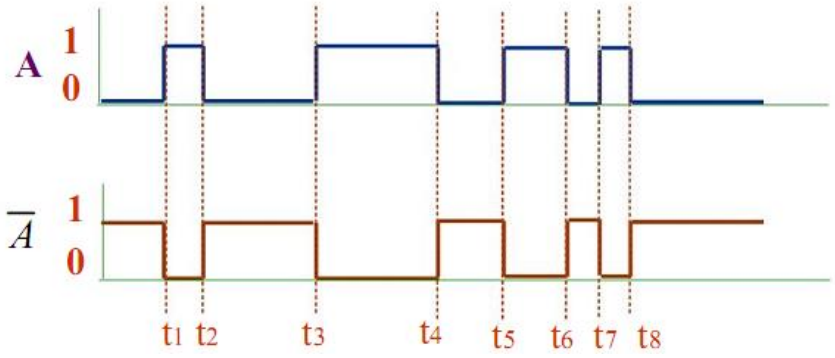
- NOT operation
  - One variable function.
  - Boolean expression: $x = \bar{A}$ (inverse of A, complement of A, or NOT A)
  - Truth table
- NOT gate
  - is a digital circuit, Inverter, performing the NOT operation
  - It inverts (complements) the input signal so that whenever the input=0, output=1, and vice versa

NOT

A	$x = \bar{A}$
0	1
1	0



INVERTER Diagram

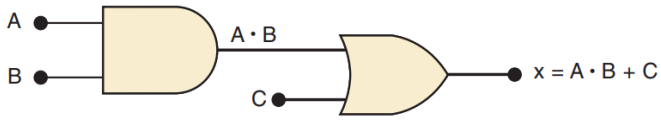


Summary of Boolean Operations

- OR (resembles binary addition, except for one operation)
  - 0+0=0 0+1=1 1+0=1 1+1=1
- AND (resembles binary multiplication)
  - 0·0=0 0·1=0 1·0=0 1·1=1
- NOT (invert)
  - $\overline{1}=0$ $\overline{0}=1$

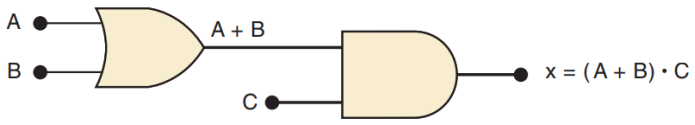
Describing Logic Circuits Algebraically

- Any logic circuit, no matter how complex, can be completely described using the three basic Boolean operations: OR, AND, NOT
  - the basic building blocks of digital systems
- Example: logic circuit with its Boolean expression



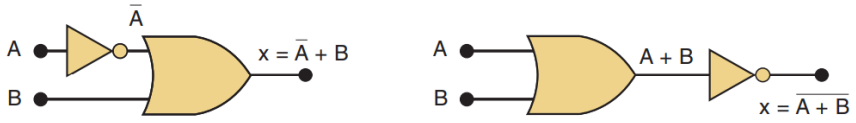
Parentheses

- Often needed to establish precedence; sometimes used optionally for clarity
- How to interpret A · B+C?
  - Is it A · B ORed with C ? Is it A ANDed with B+C ?
- Order of precedence for Boolean algebra: AND before OR. Parentheses make the expression clearer, but they are not needed for some cases
- Note that parentheses are needed here

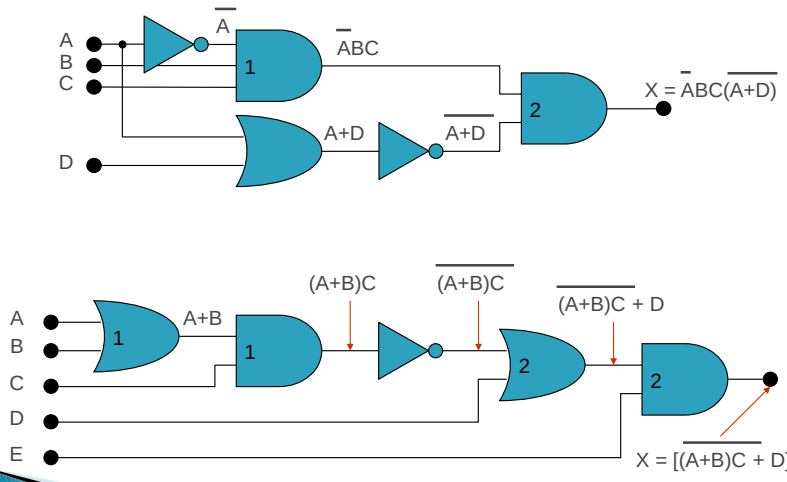


Circuits Contain INVERTERs

- Whenever an INVERTER is present in a logic-circuit diagram, its output expression is simply equal to the input expression with a bar over it



Examples

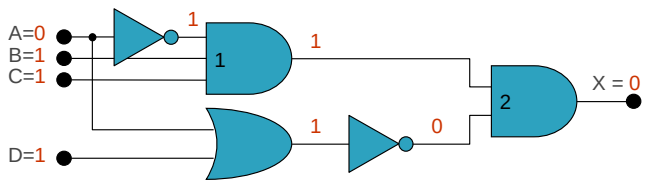


Evaluating logic circuit outputs

- Rule
  - First, perform all inversions of single terms
  - Perform all operations with parentheses
  - Perform an AND operation before an OR operation unless parentheses indicate otherwise
  - If an expression has a bar over it, perform the operations inside the expression first and then invert the result
- Example, $x = \bar{A}BC(\overline{A + D})$
  - A=0, B=1, C=1, D=1

$$
\begin{aligned}
x &= \bar{A}BC(\overline{A + D}) \\
&= \bar{0} \cdot 1 \cdot 1 \cdot (\overline{0 + 1}) \\
&= 1 \cdot 1 \cdot 1 \cdot (\overline{0 + 1}) \\
&= 1 \cdot 1 \cdot 1 \cdot (\bar{1}) \\
&= 1 \cdot 1 \cdot 1 \cdot 0 \\
&= 0
\end{aligned}
$$

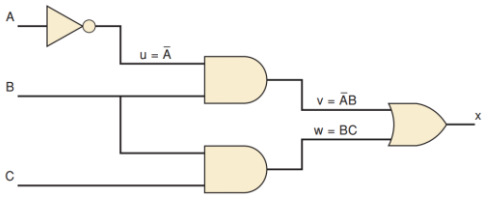
Determining Output Level from a Diagram



Determine the output for the condition where all inputs are LOW?

Analysis Using a Table

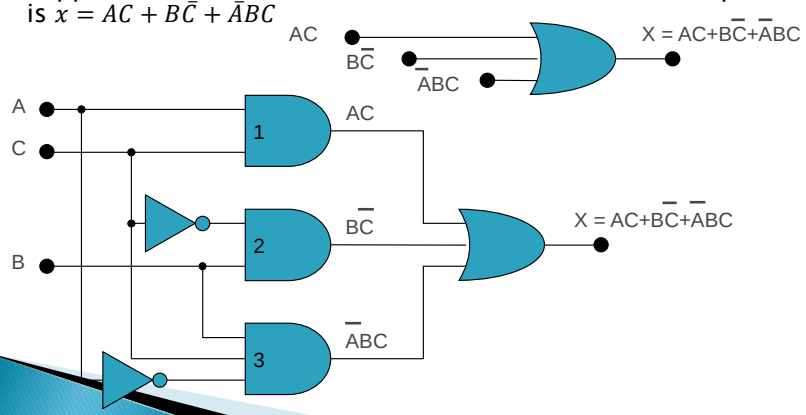
- Whenever you have a combinational logic circuit and you want to know how it works, the best way to analyze it is to use a truth table.
- The advantages of this method are:
  - It allows you to analyze one gate or logic combination at a time.
  - It allows you to easily double-check your work.
  - When you are done, you have a table that is of tremendous benefit in troubleshooting the logic circuit.



A	B	C	u= Ā	v= ĀB	w= BC	x= v+w
0	0	0	1			
0	0	1	1			
0	1	0	1			
0	1	1	1			
1	0	0	0			
1	0	1	0			
1	1	0	0			
1	1	1	0			

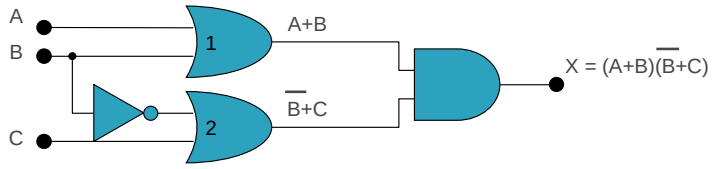
Implementing Circuits from Boolean Expressions

- When the operation of a circuit is defined by a Boolean expression, we can draw a logic circuit diagram directly from that expression
- Suppose that we wanted to construct a circuit whose output is $x = AC + B\bar{C} + \bar{A}BC$



Example

- Draw the circuit diagram to implement the expression $x = (A + B)(\overline{B} + C)$

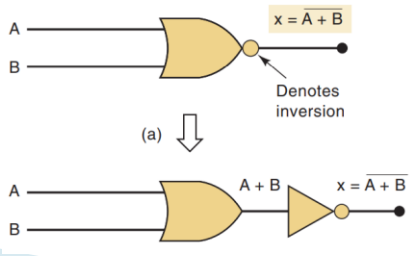


NOR operation and gate

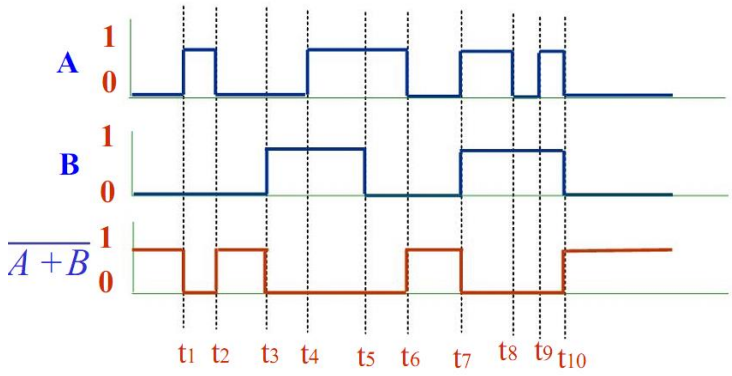
- 2-input NOR operation
  - Boolean expression $x = \overline{A + B}$
  - Symbol: $\overline{A + B}$ (A NOR B)
  - Truth table

		OR	NOR
A	B	A + B	$\overline{A + B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

- NOR gate
  - is a digital circuit performing the NOR operation
  - the output goes LOW when any input is HIGH

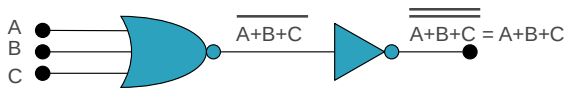


NOR gate Timing Diagram



Example

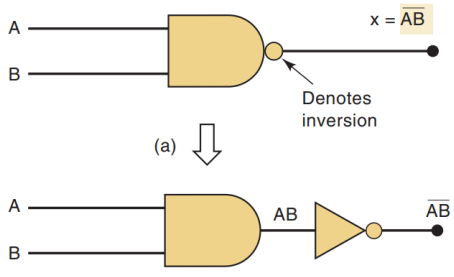
- Determine the Boolean expression for a 3-input NOR gate followed by an INVERTER



NAND operation and gate

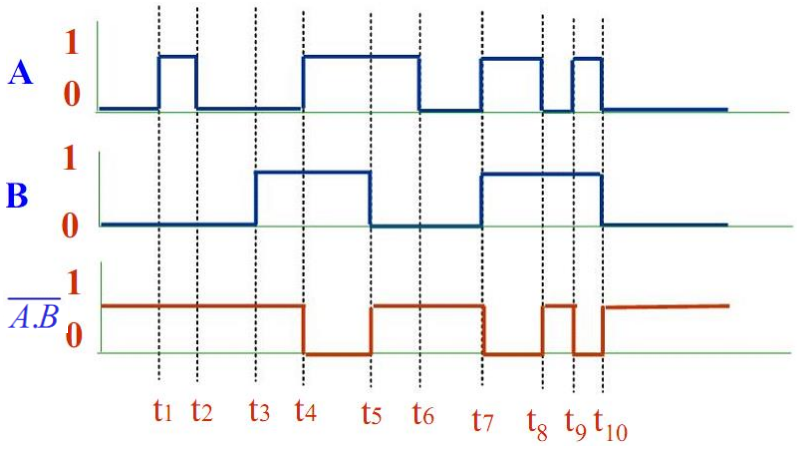
- NAND operation
  - Boolean expression x = $\overline{A.B}$
  - Symbol: $\overline{A.B}$ (A NAND B)
  - Truth table
- NAND gate
  - is a digital circuit performing the NAND operation
  - the output goes LOW only when all inputs are HIGH

		AND	NAND
A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0



33

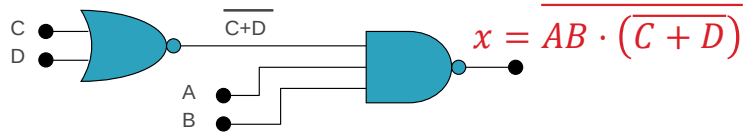
NAND gate Timing Diagram



Logic circuits 34

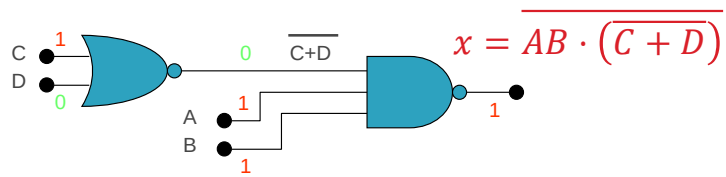
Example

- Implement the logic circuit that has the expression $x = \overline{AB \cdot \left(\overline{C + D}\right)}$ using only NOR and NAND gates



Example

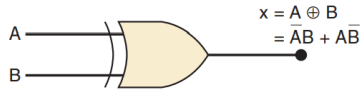
- Determine the output level in last example for A=B=C=1 and D=0

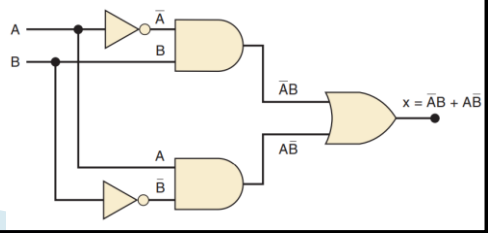


Exclusive-OR circuit

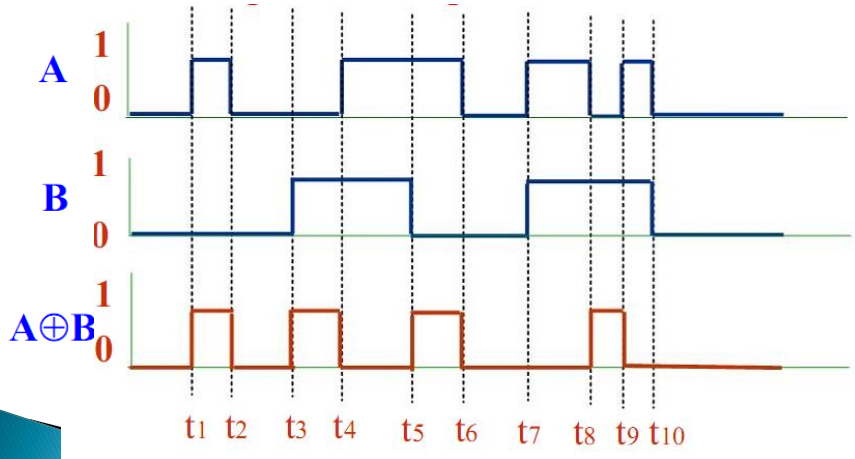
- Exclusive-OR
  - Boolean expression: $x = A.\bar{B} + \bar{A}.B$
  - Symbol: $A \oplus B$ (A XOR B)
- XOR gate
  - is a digital circuit performing the XOR operation
  - The circuit produces a HIGH output whenever the two inputs are at opposite levels.

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





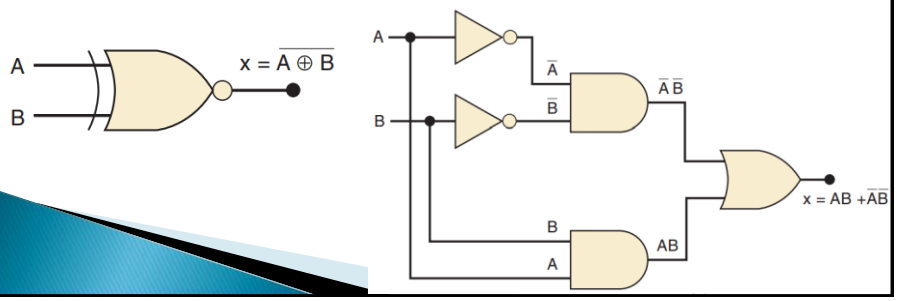
XOR Timing Diagram



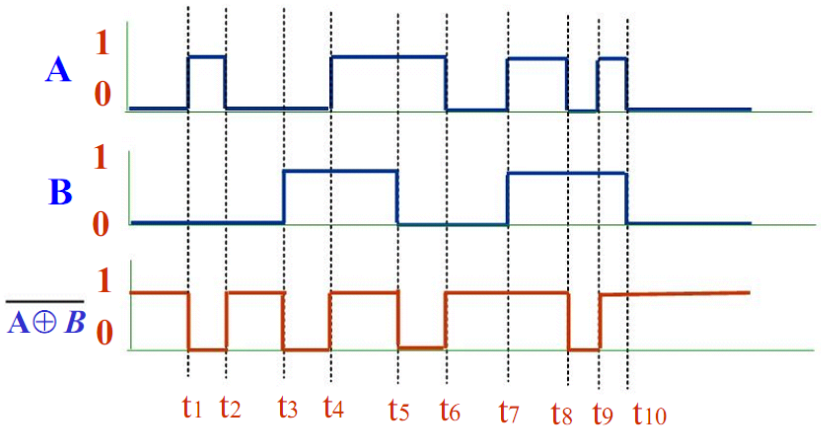
Exclusive-NOR circuit

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

- Exclusive-NOR
  - Boolean expression: $x = \overline{A.\bar{B} + \bar{A}.B}$
  - Symbol: $\overline{A \oplus B}$ (A XNOR B)
- XNOR gate
  - is a digital circuit performing the exclusive-NOR operation
  - The XNOR produces a HIGH output whenever the two inputs are at the same level



XNOR Timing Diagram

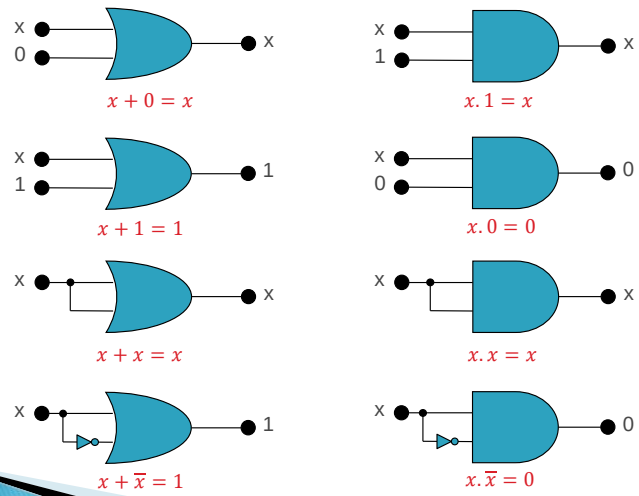


Boolean Theorems (Single variable)

- Let X: boolean variable, 0,1: constants

1. X + 0 = X -- Zero Axiom
2. X • 1 = X -- Unit Axiom
3. X + 1 = 1 -- Unit Property
4. X • 0 = 0 -- Zero Property
5. X + X = X -- Idepotence
6. X • X = X -- Idepotence
7. X + X' = 1 -- Complement
8. X • X' = 0 -- Complement
9. (X')' = X -- Involution

Boolean Theorems (single-variable)



Boolean Theorems (multivariable)

- Commutativity
1. $x + y = y + x$
2. $xy = yx$
- Associativity
1. $x + (y + z) = (x + y) + z$
2. $(xy)z = x(yz)$
- Distributivity
1. $x(y + z) = xy + xz$
2. $x + yz = (x + y)(x + z)$
- Absorption (covering)
1. $x + xy = x$
2. $x + \bar{x}y = x + y$
3. $\bar{x} + xy = \bar{x} + y$

Example

- Simplify the expression $y = A\overline{B}D + A\overline{BD}$
  - $y = A\overline{B}(D + \overline{D}) = A\overline{B}(1) = A\overline{B}$
- Simplify $z = (\bar{A} + B)(A + B)$
  - $z = \bar{A}A + \bar{A}B + AB + B.B$
    $= \bar{A}B + AB + B$
    $= B(\bar{A} + A + 1)$
    $= B$
- Simplify $x = ACD + \overline{A}BCD$
  - $x = CD(A + \overline{A}B) = CD(A + B) = ACD + BCD$

DeMorgan's Theorems

$\overline{(x+y)} = \overline{x} \cdot \overline{y}$ $\quad$ $\overline{(x \cdot y)} = \overline{x} + \overline{y}$

- Simplify the expression $z = \overline{(\overline{A} + C) \cdot (B + \overline{D})}$ to one having only single variables inverted
  - $z = \overline{(\overline{A} + C)} + \overline{(B + \overline{D})}$
    $= \bar{\bar{A}}.\bar{C} + \bar{B}.\bar{\bar{D}}$
    $= A\bar{C} + \bar{B}D$

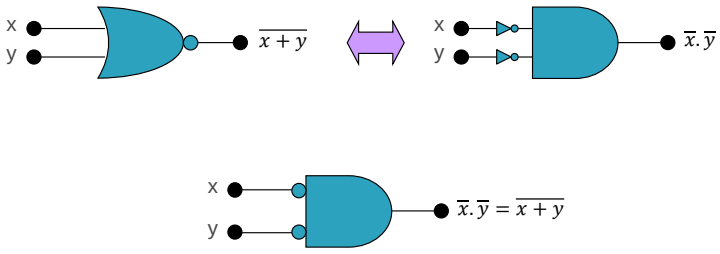
DeMorgan's Theorem

- The complement of the sum is the product of the complements
  - $\overline{(x+y)} = \overline{x} \cdot \overline{y}$
- The complement of the product is the sum of the complements
  - $\overline{(x \cdot y)} = \overline{x} + \overline{y}$
- In general,
  - $\overline{(x+y+z)} = \overline{x} \cdot \overline{y} \cdot \overline{z}$
  - $\overline{(x \cdot y \cdot z)} = \overline{x} + \overline{y} + \overline{z}$

$$x = \overline{\overline{AB} \cdot \overline{CD} \cdot \overline{EF}}$$
$$= \overline{\overline{AB}} + \overline{\overline{CD}} + \overline{\overline{EF}}$$
$$= AB + CD + EF$$

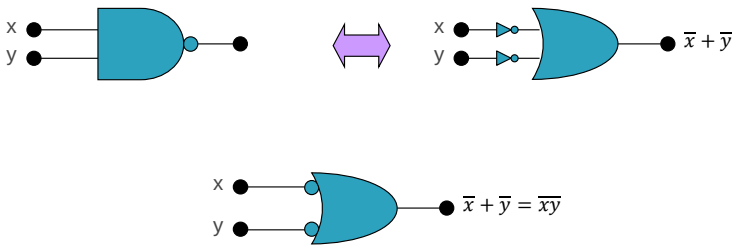
Implications of DeMorgan's Theorems (I)

$$\overline{(x+y)} = \overline{x} \cdot \overline{y}$$



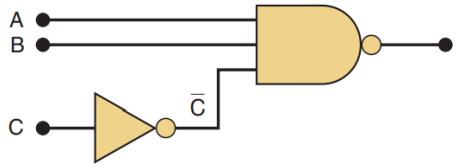
Implications of DeMorgan's Theorems (II)

$$\overline{(x \cdot y)} = \overline{x} + \overline{y}$$



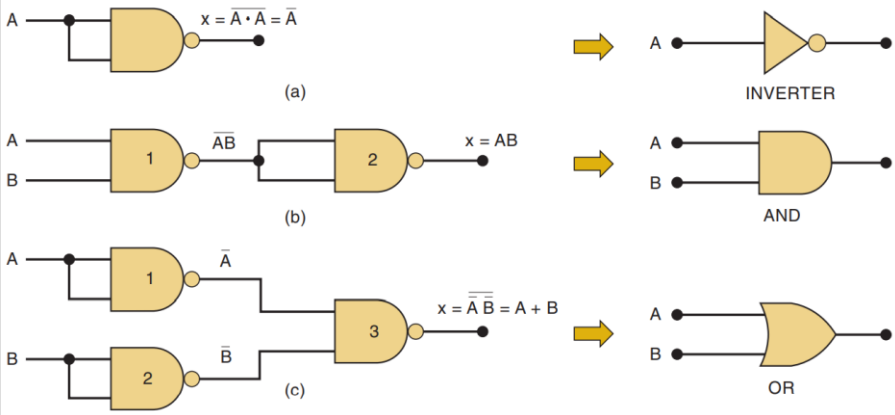
Example

- Determine the output expression for the below circuit and simplify it using DeMorgan's Theorem

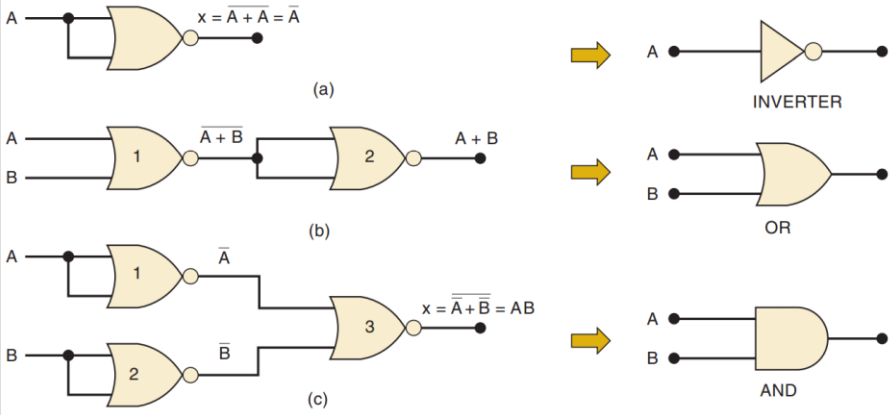


$$z = \overline{AB\overline{C}}$$
$$= \overline{A} + \overline{B} + \overline{\overline{C}}$$
$$= \overline{A} + \overline{B} + C$$

Universality of NAND gates



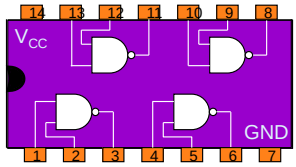
Universality of NOR gates



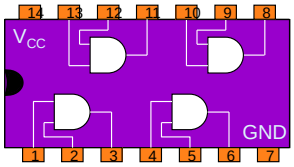
Example

- A conveyor belt will shut down whenever specific conditions occur. These conditions are monitored and reflect by the states of four logic signals as follows:
  - signal A will be HIGH whenever the conveyor belt speed is too fast;
  - signal B will be HIGH whenever the collection bin at the end of the belt is full;
  - signal C will be HIGH when the belt tension is too high;
  - signal D will be HIGH when the manual override is off.
- A logic circuit is needed to generate a signal x that will go HIGH whenever conditions A and B exist simultaneously or whenever conditions C and D exist simultaneously. Clearly, the logic expression for x will be x=AB+CD. The circuit is to be implemented with a minimum number of ICs.

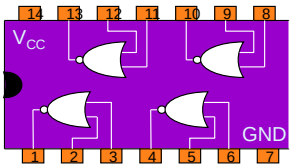
Example



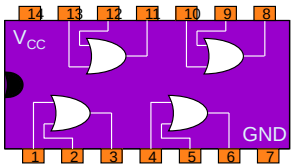
74LS00



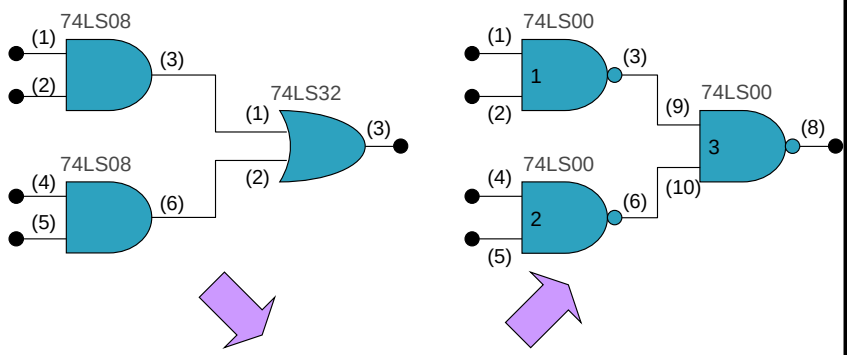
74LS08



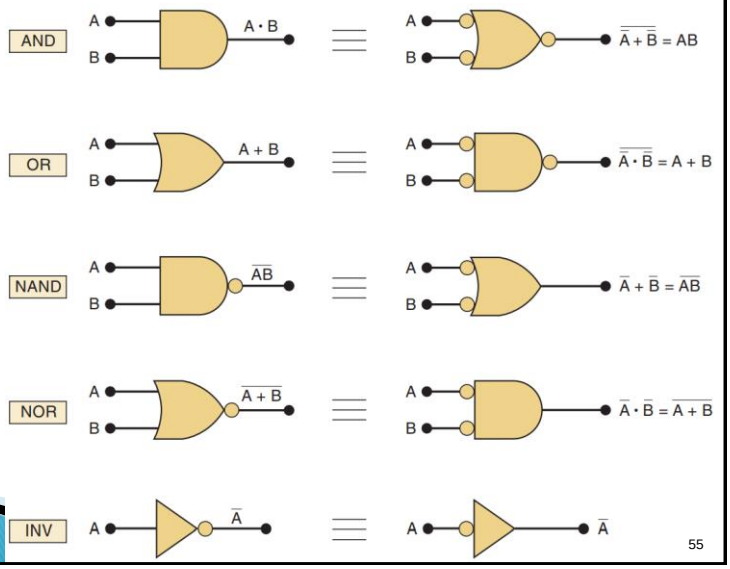
74LS02



74LS32



Alternate Logic-Gate Representations



55

How to obtain the alternative symbol from standard ones

- Invert each input and output of the standard symbol. This is done by adding bubbles (small circles) on input and output lines that do not have bubbles and by removing bubbles that are already there
- Change the operation symbol from AND to OR, or from OR to AND. (In the special case of the INVERTER, the operation symbol is not changed)

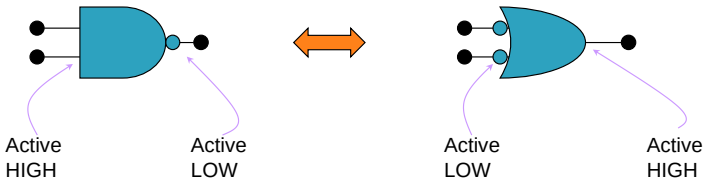
Several Points to be stressed

- The equivalences can be extended to gates with any number of inputs
- None of the standard symbols have bubbles on their inputs, and all the alternate symbols do
- The standard and alternate symbols for each gate represent the same physical circuit; there is no difference in the circuits represented by the two symbols
- NAND and NOR gates are inverting gates, and so both the standard and the alternate symbols for each will have a bubble on either the input or the output, AND and OR gates are non-inverting gates, and so the alternate symbols for each will have bubbles on both inputs and output

Logic-Symbol Interpretation

- Active high/low
  - When an input or output line on a logic circuit symbol has no bubble on it, that line is said to be **active-high**, otherwise it is **active-low**
  - The presence or absence of a bubble, then, determines the active-HIGH/active-LOW status of a circuit's inputs and output, and is used to interpret the circuit operation

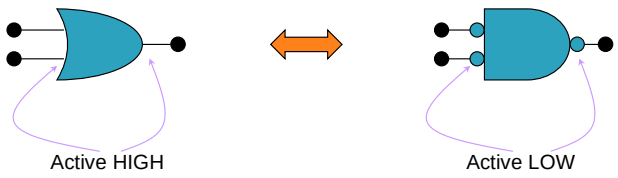
Interpretation of the two NAND Gate Symbols



Output goes LOW only when **all** inputs are HIGH

Output is HIGH when **any** input is LOW

Example: Interpretation of the two OR Gate Symbols



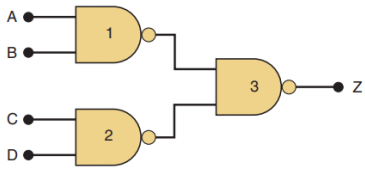
Output goes HIGH when any input is HIGH

Output goes LOW only when all inputs are LOW

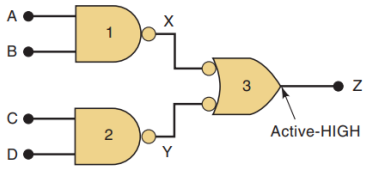
Important points concerning the logic-gate representations

1. To obtain the alternate symbol for a logic gate, take the standard symbol and change its operation symbol (OR to AND, or AND to OR), and change the bubbles on both inputs and output (i.e., delete bubbles that are present, and add bubbles where there are none).
2. To interpret the logic-gate operation, first note which logic state, 0 or 1, is the active state for the inputs and which is the active state for the output. Then realize that the output's active state is produced by having all of the inputs in their active state (if an AND symbol is used) or by having any of the inputs in its active state (if an OR symbol is used).

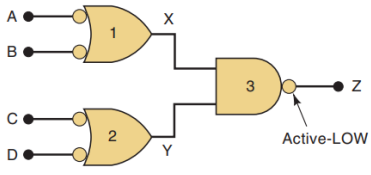
Which circuit diagram should be used?



Original circuit (all NAND gates)



Equivalent representation where output Z is active HIGH



Equivalent representation where output Z is active LOW

A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

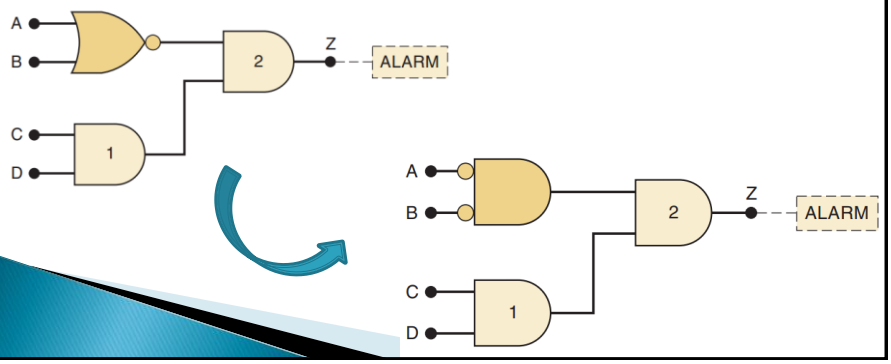
(d)

Which circuit diagram should be used?

- If the circuit is being used to cause some action (e.g., turn on an LED or activate another logic circuit) when output Z goes to the 1 state, then we say that Z is to be active-HIGH, and the circuit diagram "active-HIGH" should be used.
  - On the other hand, if the circuit is being used to cause some action when Z goes to the 0 state, then Z is to be active-LOW, and the diagram "active LOW" should be used
- Bubble placement
  - Whenever possible, choose gate symbols so that bubble outputs are connected to bubble inputs, and nonbubble outputs to nonbubble inputs

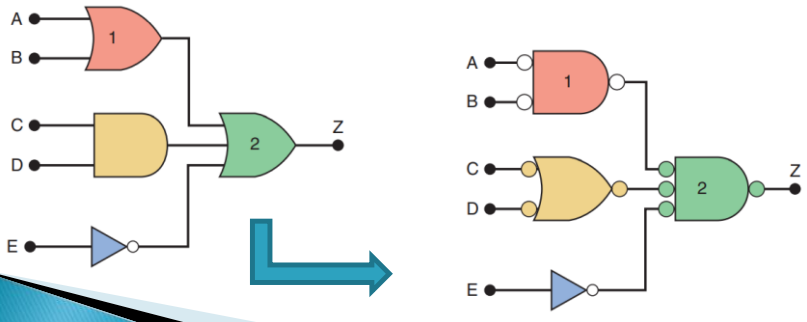
Example

- The logic circuit is being used to activate an alarm when its output Z goes HIGH. Modify the circuit diagram so that it represents the circuit operation more effectively



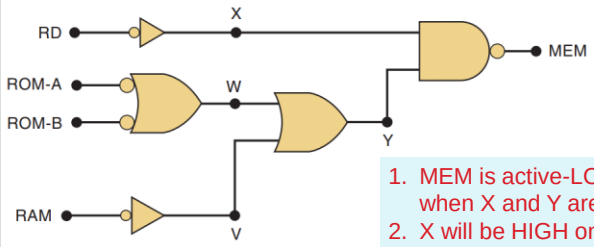
Example

- When the output of the logic circuit goes LOW, it activates another logic circuit. Modify the circuit diagram to represent the circuit operation more effective



Analyzing Circuits

- The logic circuit generates an output, MEM, that is used to activate the memory ICs in a particular microcomputer. Determine the input conditions necessary to activate MEM



1. MEM is active-LOW, and it will go LOW only when X and Y are HIGH.
2. X will be HIGH only when RD=0.
3. Y will be HIGH when either W or V is HIGH.
4. V will be HIGH when RAM=0.
5. W will be HIGH when either ROM-A or ROM-B is 0.
6. Putting this all together, MEM will go LOW only when RD=0 and at least one of the three inputs ROM-A, ROM-B, or RAM is LOW.

Asserted Level

- When a logic signal is in its active state, it can be said to be asserted (=activated)
- When a logic signal is not in its active state, it is said to be unasserted (=inactive)
- The overbar is simply a way to emphasize that these are active-LOW signals

How to represent the basic logic functions

- Many ways
  - Logical statements in our own language
  - Truth tables
  - Traditional graphic logic symbols
  - Boolean algebra expressions
  - Timing diagrams

Example

- The following English expression describes the way a logic circuit needs to operate in order to drive a seatbelt warning indicator in a car.

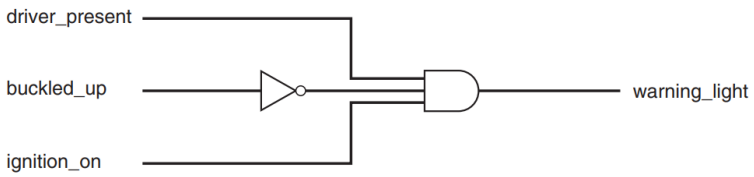
If the driver is present AND the driver is NOT buckled up AND the ignition switch is on, THEN turn on the warning light.

- Describe the circuit using Boolean algebra, schematic diagrams with logic symbols, truth tables, and timing diagrams.

Boolean expression

$warning\_light = driver\_present \bullet \overline{buckled\_up} \bullet ignition\_on$

Schematic diagram



Truth table

driver_present	buckled_up	ignition_on	warning_light
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

70

Summary

- Boolean Algebra: a mathematical tool used in the analysis and design of digital circuits
- OR, AND, NOT: basic Boolean operations
- OR: HIGH output when any input is HIGH
- AND: HIGH output only when all inputs are HIGH
- NOT: output is the opposite logic level as the input
- NOR: OR with its output connected to an INVERTER
- NAND: AND with its output connected to an INVERTER
- Boolean theorems and rules: to simplify the expression of a logic circuit and can lead to a simpler way of implementing the circuit
- NAND, NOR: can be used to implement any of the basic Boolean operations