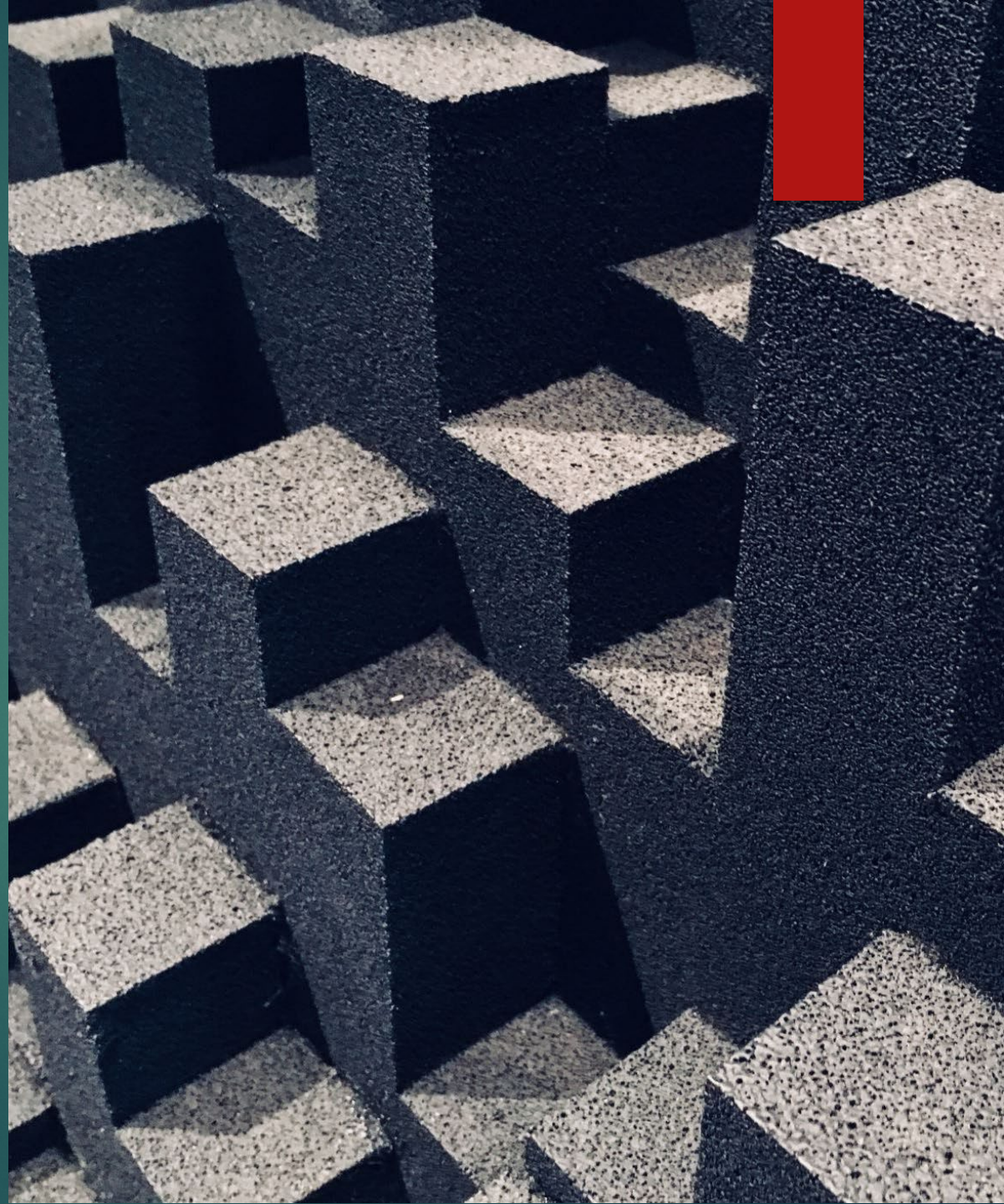


Digital Logic Circuits

Black Boxes and Gates



Black Boxes and Gates

- ▶ Combinations of signal bits (1's and 0's) can be transformed into other combinations of signal bits (1's and 0's) by means of various circuits.
- ▶ Because a variety of different technologies are used in circuit construction, computer engineers and digital system designers find it useful to think of certain basic circuits as black boxes.

Black Boxes and Gates (2/8)

- ▶ The inside of a black box contains the detailed implementation of the circuit and is often ignored while attention is focused on the relation between the input and the output signals.



Black Boxes and Gates (3/8)

The operation of a black box is completely specified by constructing an input/output table that lists all its possible input signals together with their corresponding output signals.



For example, the black box pictured above has three input signals. Since each of these signals can take the value 1 or 0, there are eight possible combinations of input signals.

Black Boxes and Gates (4/8)

One possible correspondence of input to output signals is as follows:

Input			Output
<i>P</i>	<i>Q</i>	<i>R</i>	<i>S</i>
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

Black Boxes and Gates (5/8)

The third row, for instance, indicates that for inputs $P = 1$,

$Q = 0$, and $R = 1$, the output S is 0.

A **NOT-gate** (or **inverter**) is a circuit with one input signal and one output signal. If the input signal is 1, the output signal is 0. Conversely, if the input signal is 0, then the output signal is 1.

Black Boxes and Gates (6/8)

An **AND-gate** is a circuit with two input signals and one output signal. If both input signals are 1, then the output signal is 1. Otherwise, the output signal is 0.

An **OR-gate** also has two input signals and one output signal. If both input signals are 0, then the output signal is 0. Otherwise, the output signal is 1.

Black Boxes and Gates

The actions of NOT-, AND-, and OR-gates are summarized in Figure 2.4.3, where P and Q represent input signals and R represents the output signal.




Type of Gate	Symbolic Representation	Action																		
NOT		<table><tr><th colspan="2">Input</th><th>Output</th></tr><tr><th>P</th><th></th><th>R</th></tr><tr><td>1</td><td></td><td>0</td></tr><tr><td>0</td><td></td><td>1</td></tr></table>	Input		Output	P		R	1		0	0		1						
Input		Output																		
P		R																		
1		0																		
0		1																		
AND		<table><tr><th colspan="2">Input</th><th>Output</th></tr><tr><th>P</th><th>Q</th><th>R</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>	Input		Output	P	Q	R	1	1	1	1	0	0	0	1	0	0	0	0
Input		Output																		
P	Q	R																		
1	1	1																		
1	0	0																		
0	1	0																		
0	0	0																		
OR		<table><tr><th colspan="2">Input</th><th>Output</th></tr><tr><th>P</th><th>Q</th><th>R</th></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>	Input		Output	P	Q	R	1	1	1	1	0	1	0	1	1	0	0	0
Input		Output																		
P	Q	R																		
1	1	1																		
1	0	1																		
0	1	1																		
0	0	0																		

Figure 2.4.3

Black Boxes and Gates

Rules for a Combinational Circuit

Never combine two input wires. 2.4.1

A single input wire can be split partway and used as input for two separate gates. 2.4.2

An output wire can be used as input. 2.4.3

No output of a gate can eventually feed back into that gate. 2.4.4

► Rule (2.4.4) is violated in more complex circuits, called **sequential circuits**, whose output at any given time depends both on the input at that time and also on previous inputs.



The Input/Output Table for a Circuit

The Input/Output Table for a Circuit (1/1)

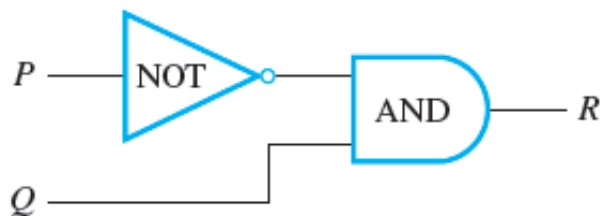
► If you are given a set of input signals for a circuit, you can find its output by tracing through the circuit gate by gate.



Example 2.4.1 – Determining Output for a Given Input

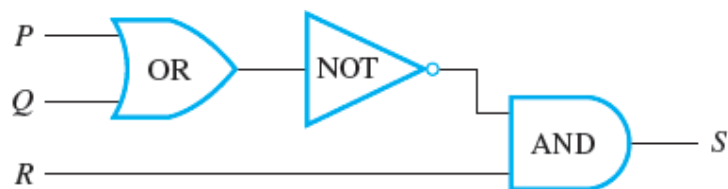
Indicate the output of the circuits shown below for the given input signals.

a.



Input signals: $P = 0$ and $Q = 1$

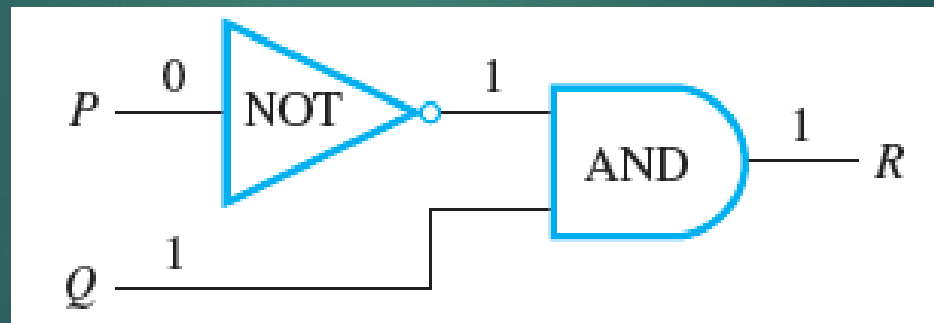
b.



Input signals: $P = 1$, $Q = 0$, $R = 1$

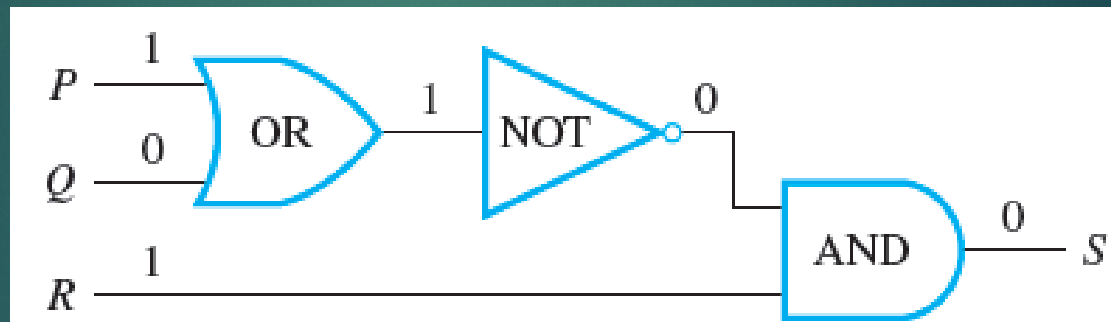
Example 2.4.1 – Solution (1/2)

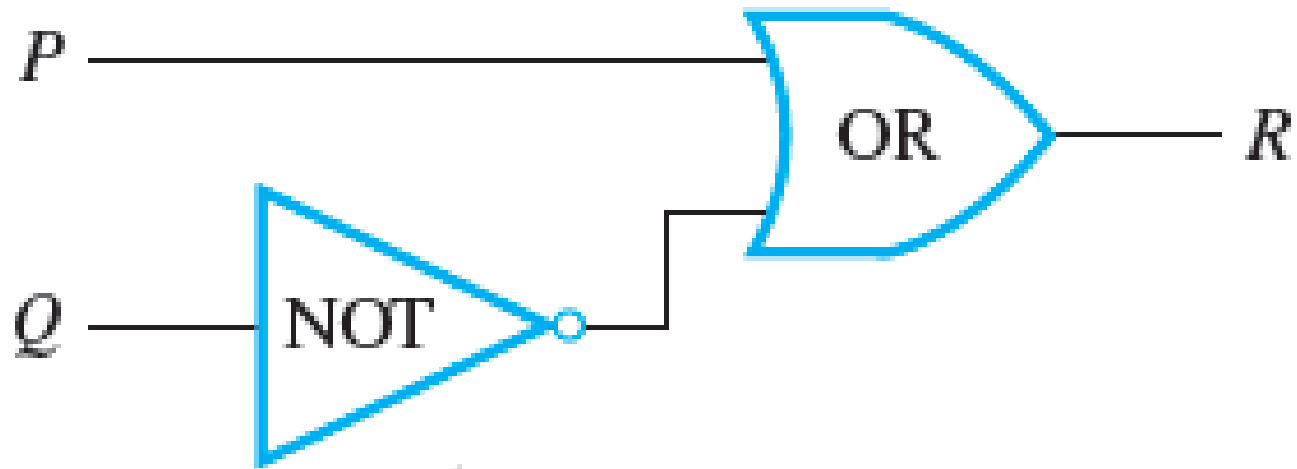
- ▶ a. Move from left to right through the diagram, tracing the action of each gate on the input signals. The NOT-gate changes $P = 0$ to a 1, so both inputs to the AND-gate are 1; hence the output R is 1.
- ▶ This is illustrated by annotating the diagram as shown below.



Example 2.4.1 – Solution

- b. The output of the OR-gate is 1 since one of the input signals, P , is 1. The NOT-gate changes this 1 into a 0, so the two inputs to the AND-gate are 0 and $R = 1$. Hence the output S is 0. The trace is shown below.





Example 2.4.2 – Constructing the Input/Output Table for a Circuit

CONSTRUCT THE INPUT/OUTPUT TABLE FOR THE FOLLOWING CIRCUIT.

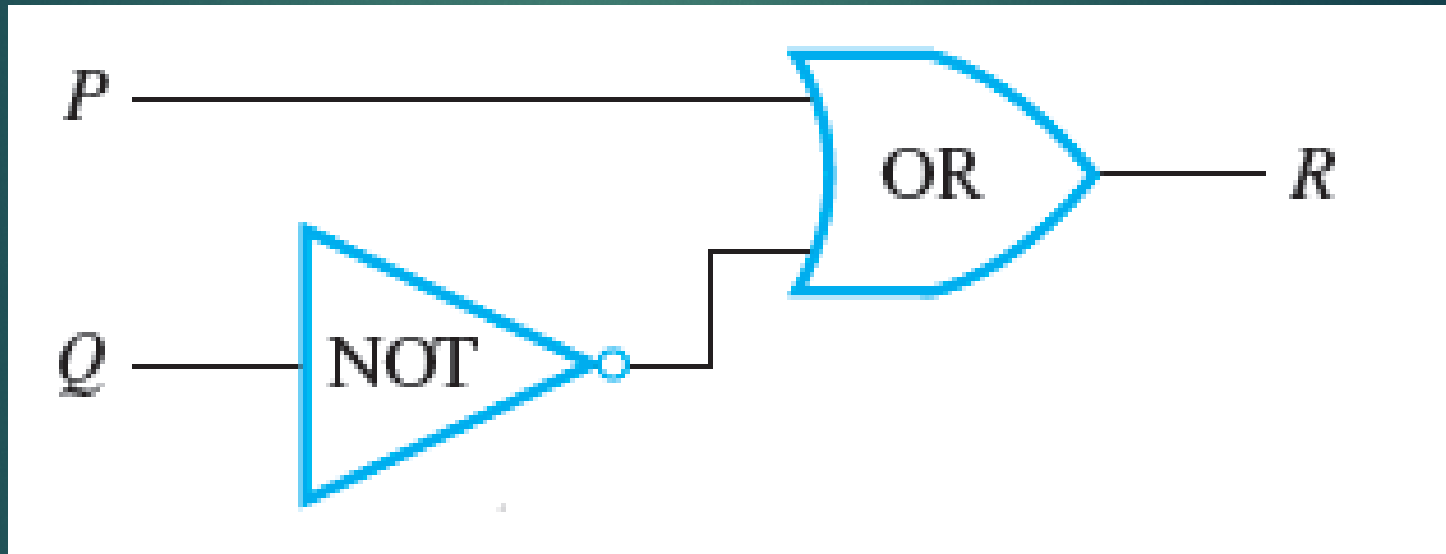
Example 2.4.2 – Solution


► List the four possible combinations of input signals, and find the output for each by tracing through the circuit.

Input		Output
<i>P</i>	<i>Q</i>	<i>R</i>
1	1	1
1	0	1
0	1	0
0	0	1

In Class Assignment

- Convert the circuit to a Boolean equation and find the truth table using Python





The Boolean Expression Corresponding to a Circuit

The Boolean Expression Corresponding to a Circuit (1/3)

In logic, variables such as p , q , and r represent statements, and a statement can have one of only two truth values: T (true) or F (false).

A statement form is an expression, such as $p \wedge (\sim q \vee r)$

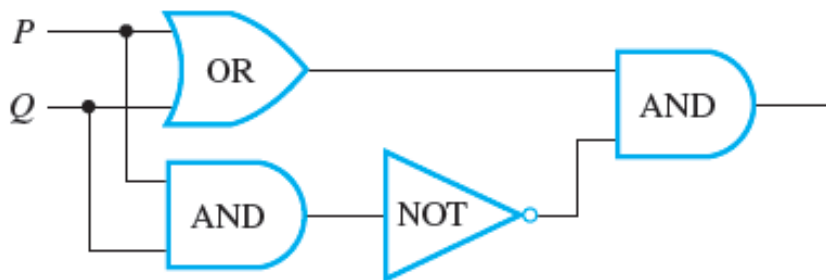
composed of statement variables and logical connectives.

Any variable, such as a statement variable or an input signal, that can take one of only two values is called a **Boolean variable**. An expression composed of Boolean variables and the following connective operators:

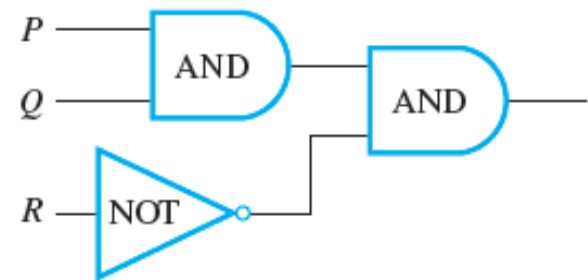
\sim , \wedge , and \vee is called a **Boolean expression**.

Example 2.4.3 – Finding a Boolean Expression for a Circuit

Find the Boolean expressions that correspond to the circuits shown below. A black dot indicates a soldering of two wires; wires that cross without a dot are assumed not to touch.



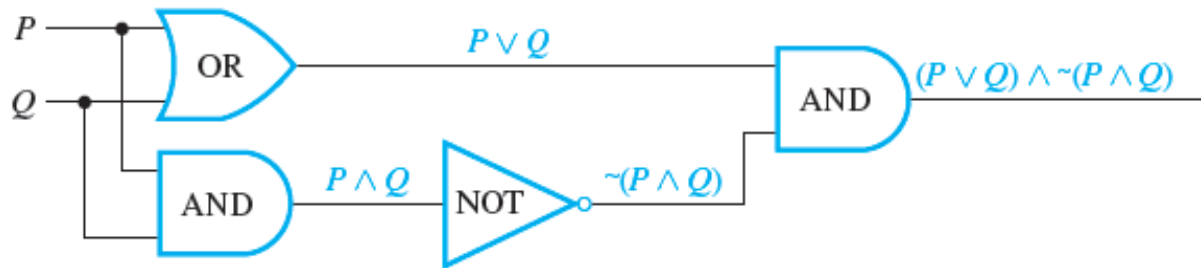
(a)



(b)

Example 2.4.3 – Solution (1/2)

- a. Trace through the circuit from left to right, indicating the output of each gate symbolically, as shown below.

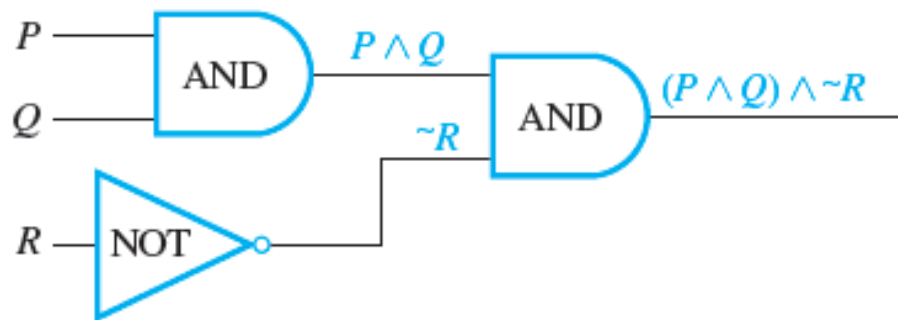


The final expression obtained,
exclusive or: P or Q but not both.

$(P \vee Q) \wedge \sim(P \wedge Q)$, or for

Example 2.4.3 – Solution (2/2)

b. The Boolean expression corresponding to the circuit is $(P \wedge Q) \wedge \sim R$, as shown below.



The Boolean Expression Corresponding to a Circuit (2/3)

Definition

A **recognizer** is a circuit that outputs a 1 for exactly one particular combination of input signals and outputs 0's for all other combinations.

The Boolean Expression Corresponding to a Circuit

P	Q	R	$(P \wedge Q) \wedge \sim R$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

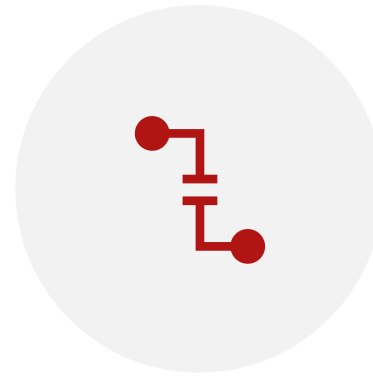
- Input/Output Table for a Recognizer

The Circuit Corresponding to a Boolean Expression

The Circuit Corresponding to a Boolean Expression (1/1)



THE FOLLOWING EXAMPLE SHOWS HOW TO CONSTRUCT A CIRCUIT CORRESPONDING TO A BOOLEAN EXPRESSION.



THE STRATEGY IS TO WORK FROM THE OUTERMOST PART OF THE BOOLEAN EXPRESSION TO THE INNERMOST PART, ADDING LOGIC GATES THAT CORRESPOND TO THE OPERATIONS IN THE EXPRESSION AS YOU MOVE FROM RIGHT TO LEFT IN THE CIRCUIT DIAGRAM.

Example 2.4.4 – Constructing Circuits for Boolean Expressions

Construct circuits for the following Boolean expressions.

a. $(\sim P \wedge Q) \vee \sim Q$

b. $((P \wedge Q) \wedge (R \wedge S)) \wedge T$

Example 2.4.4 – Solution (1/4)

a. Write the input variables in a column on the left side of the diagram. Since the last operation executed when evaluating

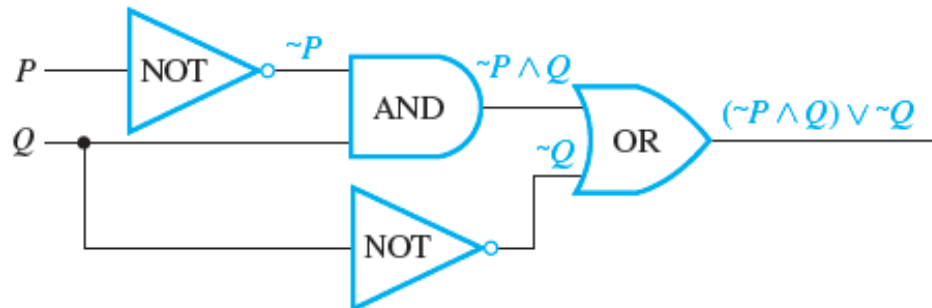
$(\sim P \wedge Q) \vee \sim Q$ is \vee , put an OR-gate at the extreme right of the diagram.

One input to this gate is $\sim P \wedge Q$, so draw an AND-gate to the left of the OR-gate and show its output coming into the OR-gate. Since one input to the AND-gate is $\sim P$, draw a line from P to a NOT-gate and from there to the AND-gate. Since the other input to the AND-gate is Q , draw a line from Q directly to the AND-gate.

Example 2.4.4 – Solution (2/4)

The other input to the OR-gate is $\sim Q$, so draw a line from Q to a NOT-gate and from the NOT-gate to the OR-gate.

The circuit you obtain is shown below.



Example 2.4.4 – Solution (3/4)

b. To start constructing this circuit, put one AND-gate at the extreme right to correspond to the \wedge , which is the final operation between $((P \wedge Q) \wedge (R \wedge S))$ and T .

To the left of that gate put the AND-gate corresponding to the \wedge between $P \wedge Q$ and $R \wedge S$.

To the left of that gate put the two AND-gates corresponding to the \wedge 's between P and Q and between R and S .

Example 2.4.4 – Solution (4/4)

The circuit is shown in Figure 2.4.4.

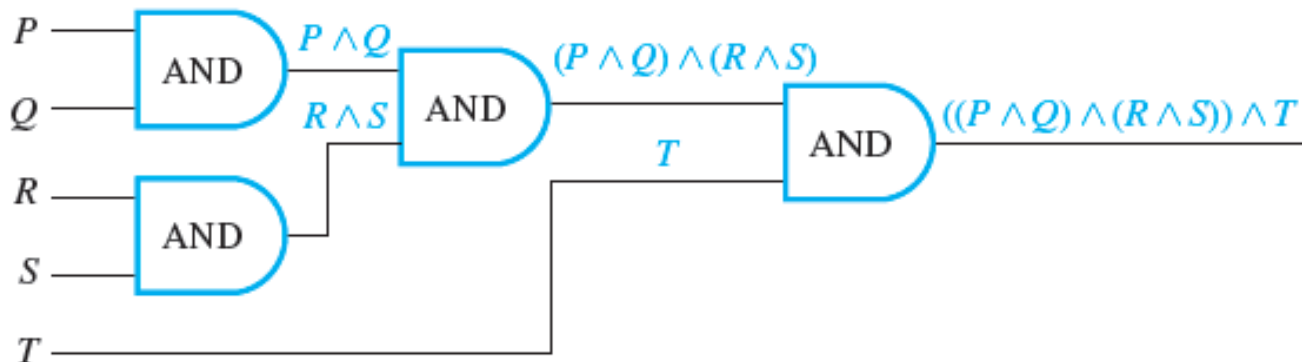
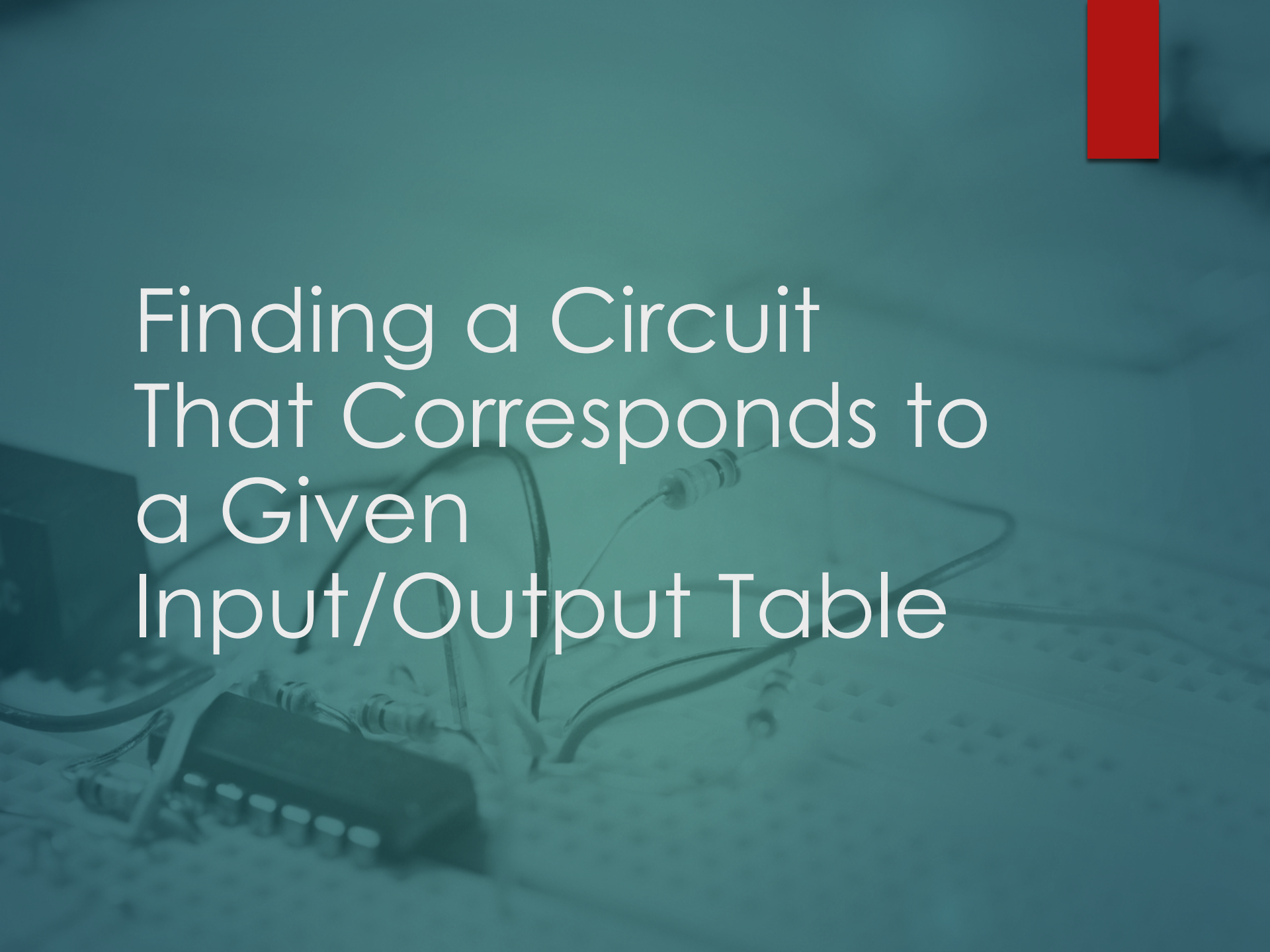


Figure 2.4.4



Finding a Circuit That Corresponds to a Given Input/Output Table

Example 2.4.5 – *Designing a Circuit for a Given Input/Output Table*

- Design a circuit for the following input/output table:

Input			Output
<i>P</i>	<i>Q</i>	<i>R</i>	<i>S</i>
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

Example 2.4.5 – Solution (1/4)

First construct a Boolean expression with this table as its truth table. To do this, identify each row for which the output is 1—in this case, the rows 1, 3, and 4.

For each such row, construct an *and* expression that produces a 1 (or true) for the exact combination of input values for that row and a 0 (or false) for all other combinations of input values.

For example, the expression for row 1 is $P \wedge Q \wedge R$ because $P \wedge Q \wedge R$ is 1 if $P = 1$ and $Q = 1$ and $R = 1$, and it is 0 for all other values of P , Q , and R .

Example 2.4.5 – Solution (2/4)

The expression for row 3 is $P \wedge \sim Q \wedge R$ because $P \wedge \sim Q \wedge R$ is 1 if $P = 1$ and $Q = 0$ and $R = 1$, and it is 0 for all other values of P , Q , and R . Similarly, the expression for row 4 is $P \wedge \sim Q \wedge \sim R$.

Now any Boolean expression with the given table as its truth table has the value 1 in case $P \wedge Q \wedge R = 1$, or in case $P \wedge \sim Q \wedge R = 1$,

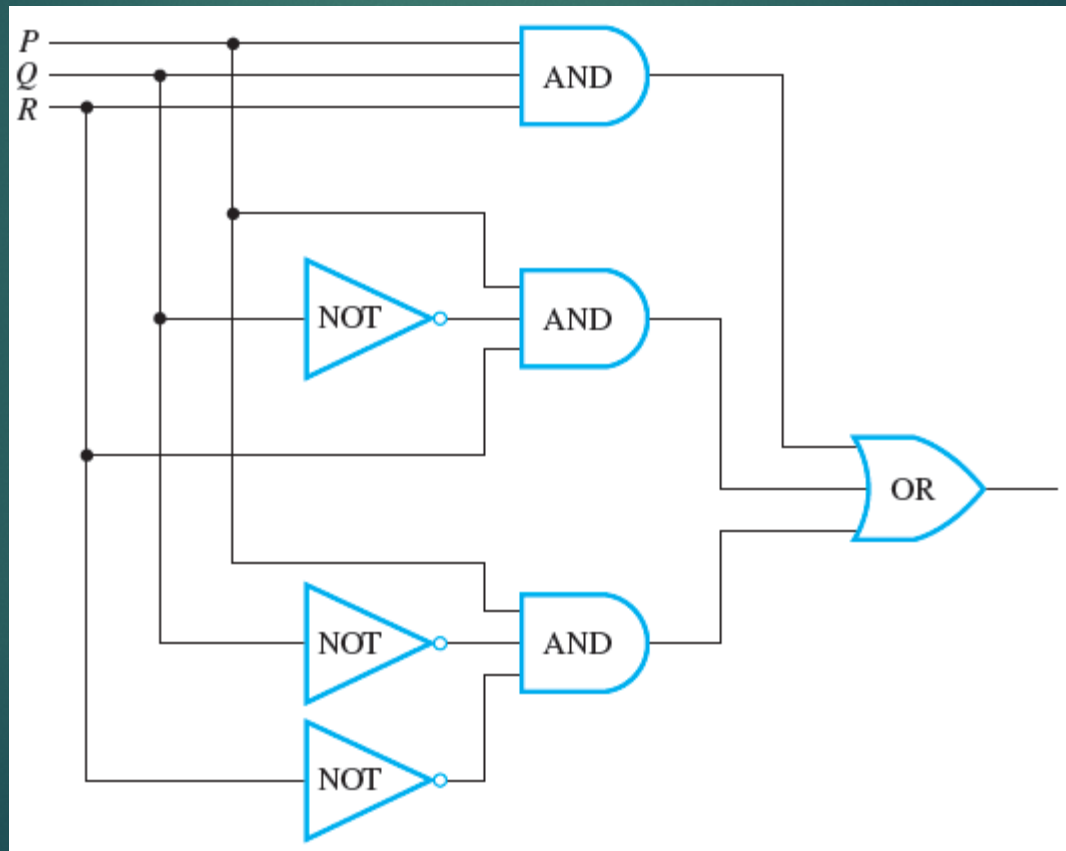
or in case $P \wedge \sim Q \wedge \sim R = 1$, and in no other cases. It follows that a Boolean expression with the given truth table is:

$$(P \wedge Q \wedge R) \vee (P \wedge \sim Q \wedge R) \vee (P \wedge \sim Q \wedge \sim R).$$

2.4.5

Example 2.4.5 – Solution (3/4)

► The circuit corresponding to this expression has the diagram shown in Figure 2.4.7.



► Figure 2.4.7

Example 2.4.5 – Solution (4/4)

Observe that expression (2.4.5) is a disjunction of terms that are themselves conjunctions in which one of P or $\sim P$, one of

Q or $\sim Q$, and one of R or $\sim R$ appear. Such expressions are said to be in **disjunctive normal form** or **sum-of products form**.

$$(P \wedge Q \wedge R) \vee (P \wedge \sim Q \wedge R) \vee (P \wedge \sim Q \wedge \sim R).$$

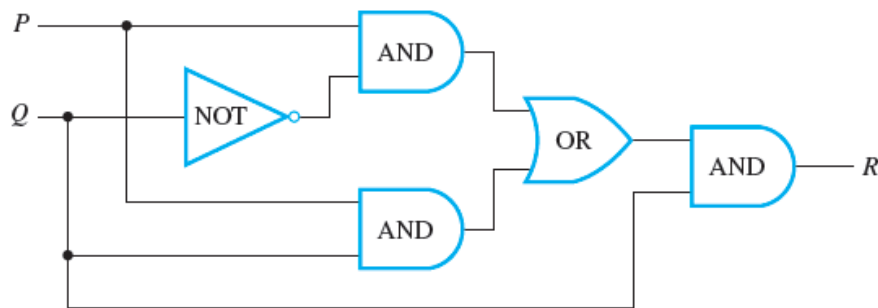
2.4.5

The background of the slide features a blurred photograph of an electronic breadboard circuit. A black integrated circuit (IC) is visible in the lower-left foreground, with several thin wires connected to its pins and other points on the breadboard. The entire image is covered with a semi-transparent teal overlay. In the top-right corner, there is a solid red vertical rectangular bar.

Simplifying Combinational Circuits

Simplifying Combinational Circuits (1/3)

Consider the two combinational circuits shown in Figure 2.4.8.



(a)



(b)

Figure 2.4.8

Simplifying Combinational Circuits

If you trace through circuit (a), you will find that its input/output table is

This is the same as the input/output table for circuit (b). Thus these two circuits do the same job in the sense that they transform the same combinations of input signals into the same output signals.

Input		Output
<i>P</i>	<i>Q</i>	<i>R</i>
1	1	1
1	0	0
0	1	0
0	0	0

Simplifying Combinational Circuits

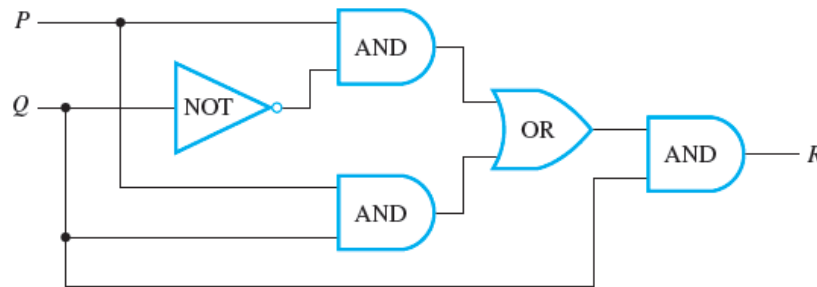
- ▶ Yet circuit (b) is simpler than circuit (a) in that it contains many fewer logic gates. Thus, as part of an integrated circuit, it would take less space and require less power.

Definition

Two digital logic circuits are **equivalent** if, and only if, their input/output tables are identical.

Example 2.4.6 – Showing That Two Circuits Are Equivalent

Find the Boolean expressions for each circuit in Figure 2.4.8. Use Theorem 2.1.1 to show that these expressions are logically equivalent when regarded as statement forms.



(a)



(b)

Figure 2.4.8

Example 2.4.6 – Solution (1/2)

The Boolean expressions that correspond to circuits (a) and (b) are $((P \wedge \sim Q) \vee (P \wedge Q)) \wedge Q$ and $P \wedge Q$, respectively. By Theorem 2.1.1,

$$((P \wedge \sim Q) \vee (P \wedge Q)) \wedge Q$$

$$\equiv (P \wedge (\sim Q \vee Q)) \wedge Q$$

by the distributive law

$$\equiv (P \wedge (Q \vee \sim Q)) \wedge Q$$

by the commutative law for \vee

$$\equiv (P \wedge \mathbf{t}) \wedge Q$$

by the negation law

$$\equiv P \wedge Q$$

by the identity law.

Example 2.4.6 – Solution (2/2)

It follows that the truth tables for $((P \wedge \sim Q) \vee (P \wedge Q)) \wedge Q$ and $P \wedge Q$ are the same. Hence the input/output tables for the circuits corresponding to these expressions are also the same, and so the circuits are equivalent.

Simplifying Expressions in Python

```
from sympy import *

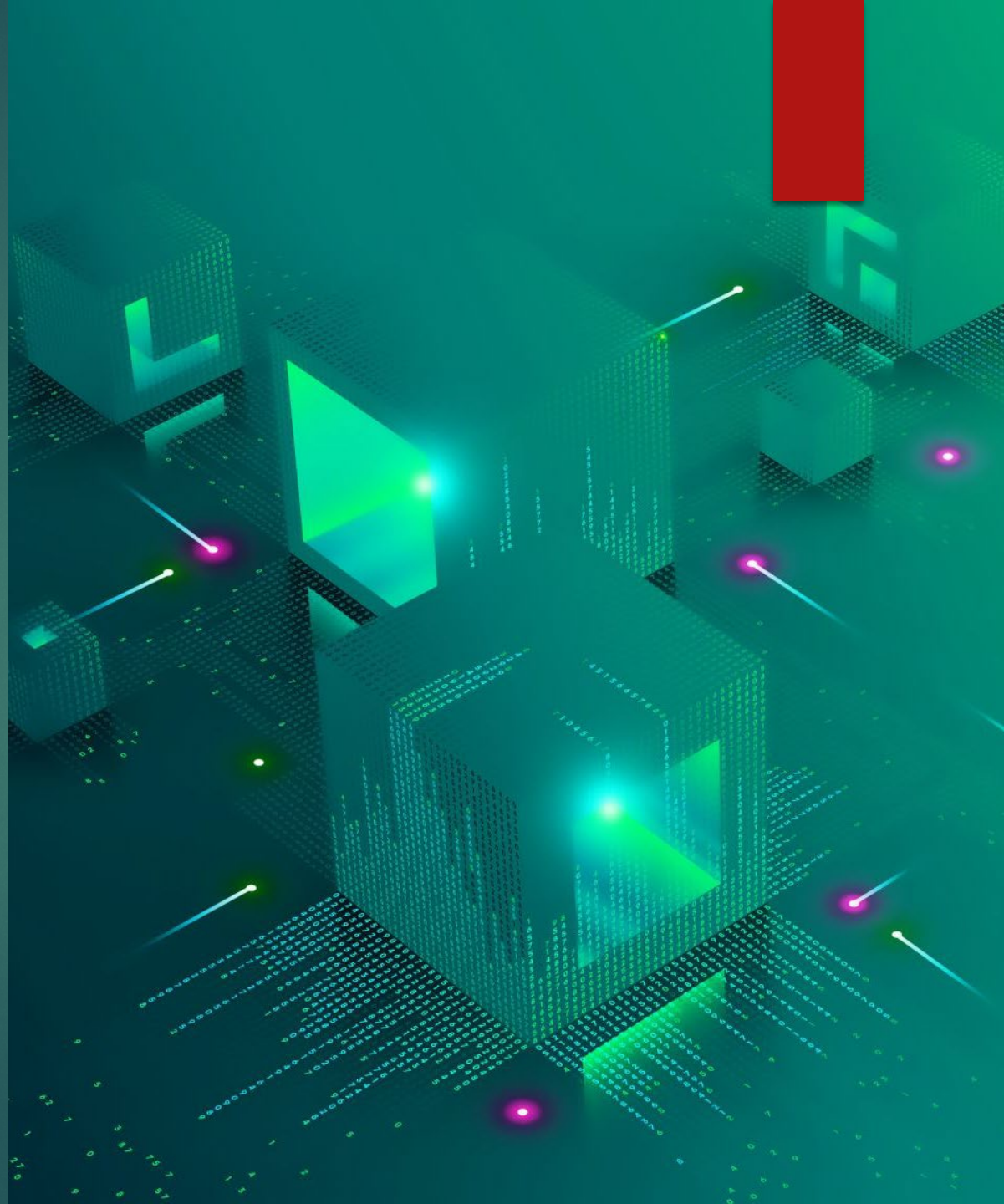
# Define logical variables
A, B, C = symbols('A B C')

# Example logical expression using 'and' (&) and 'or' (|)
expr = (A & B) | (A & ~B) # Equivalent to A

# Simplify the logical expression
simplified_expr = simplify(expr)

print("Simplified Expression:", simplified_expr)
```

NAND and NOR Gates



NAND and NOR Gates (1/3)



Two gates not previously introduced are particularly useful for this: NAND-gates and NOR-gates. A NAND-gate is a single gate that acts like an AND-gate followed by a NOT-gate. A NOR-gate acts like an OR-gate followed by a NOT-gate.

Thus the output signal of a NAND-gate is 0 when, and only when, both input signals are 1, and the output signal for a NOR-gate is 1 when, and only when, both input signals are 0.

NAND and NOR Gates (2/3)

The logical symbols corresponding to these gates are $|$ (for NAND) and \downarrow (for NOR), where $|$ is called a **Sheffer stroke** and \downarrow is called a **Peirce arrow**.

Thus $P|Q \equiv \sim(P \wedge Q)$ and $P \downarrow Q \equiv \sim(P \vee Q)$.

Type of Gate	Symbolic Representation	Action																		
NAND		<table><tr><th colspan="2">Input</th><th>Output</th></tr><tr><th>P</th><th>Q</th><th>$R = P Q$</th></tr><tr><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>1</td></tr></table>	Input		Output	P	Q	$R = P Q$	1	1	0	1	0	1	0	1	1	0	0	1
		Input		Output																
		P	Q	$R = P Q$																
		1	1	0																
		1	0	1																
		0	1	1																
0	0	1																		
NOR		<table><tr><th colspan="2">Input</th><th>Output</th></tr><tr><th>P</th><th>Q</th><th>$R = P \downarrow Q$</th></tr><tr><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td></tr></table>	Input		Output	P	Q	$R = P \downarrow Q$	1	1	0	1	0	0	0	1	0	0	0	1
		Input		Output																
		P	Q	$R = P \downarrow Q$																
		1	1	0																
		1	0	0																
		0	1	0																
0	0	1																		

NAND and NOR Gates (3/3)

THE TABLE SUMMARIZES
THE ACTIONS OF NAND
AND NOR GATES.

Example 2.4.7 – Rewriting Expressions Using the Sheffer Stroke

Use Theorem 2.1.1 and the definition of Sheffer stroke to show that

a. $\sim P \equiv P|P$ and b. $P \vee Q \equiv (P|P)|(Q|Q).$

Example 2.4.7 – Solution

a. $\sim P \equiv \sim(P \wedge P)$

$$\equiv P|P$$

by the idempotent law for \vee

by definition of $|$.

b. $P \vee Q \equiv \sim(\sim(P \vee Q))$

$$\equiv \sim(\sim P \wedge \sim Q)$$

by the double negative law

by De Morgan's laws

$$\equiv \sim((P|P) \wedge (Q|Q))$$

by part (a)

$$\equiv (P|P)|(Q|Q)$$

by definition of $|$.