

The Logic of Compound Statements

Application: Digital Logic Circuits

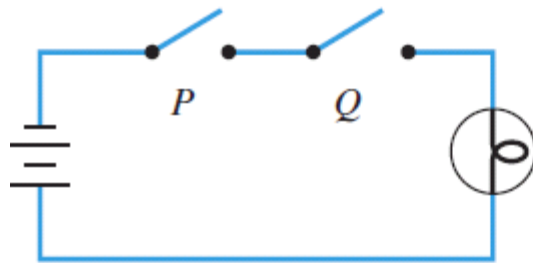
CSE 215, Foundations of Computer Science

Stony Brook University

<http://www.cs.stonybrook.edu/~cse215>

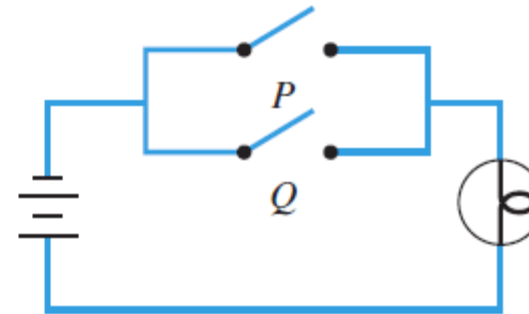
Application of Logic: Digital Logic Circuits

- Analogy between the operations of switching devices and the operations of logical connectives



Switches “in series”

Switches		Light Bulb
<i>P</i>	<i>Q</i>	State
closed	closed	on
closed	open	off
open	closed	off
open	open	off



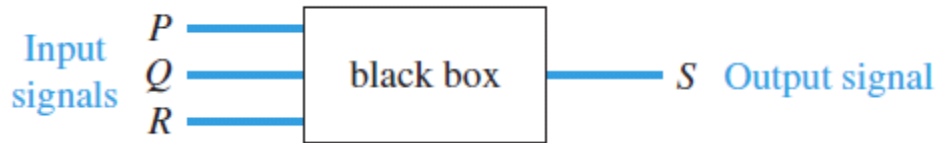
Switches “in parallel”

Switches		Light Bulb
<i>P</i>	<i>Q</i>	State
closed	closed	on
closed	open	on
open	closed	on
open	open	off

Binary digits (bits): we will use the symbols **1** and **0** instead of “on” (“closed” or True) and “off” (“open” or False)

Black Boxes and Gates


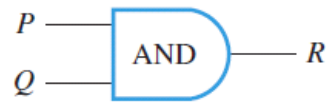
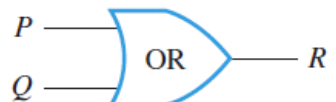
- Combinations of signal bits (1's and 0's) can be transformed into other combinations of signal bits (1's and 0's) by means of various circuits



An Input/Output Table

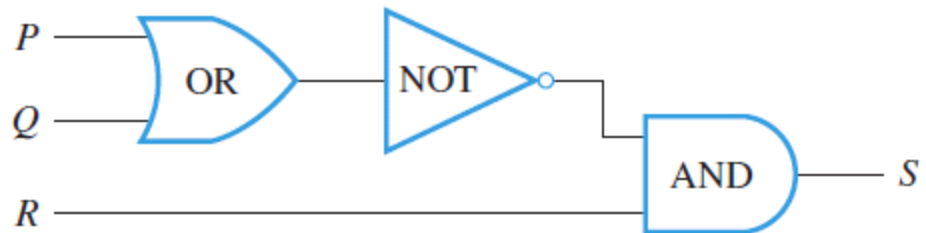
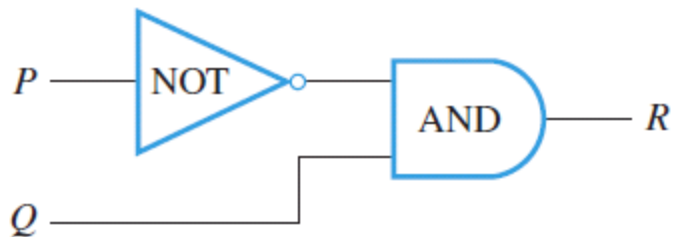
Input			Output
P	Q	R	S
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

- An efficient method for designing complicated circuits is to build them by connecting less complicated black box circuits: NOT-, AND-, and OR-gates.

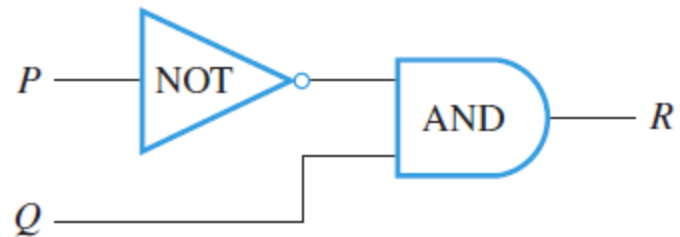
Type of Gate	Symbolic Representation	Action																		
NOT		<table border="1"> <thead> <tr> <th>Input</th> <th>Output</th> </tr> <tr> <th><i>P</i></th> <th><i>R</i></th> </tr> </thead> <tbody> <tr> <td>1</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> </tr> </tbody> </table>	Input	Output	<i>P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

Combinational Circuits

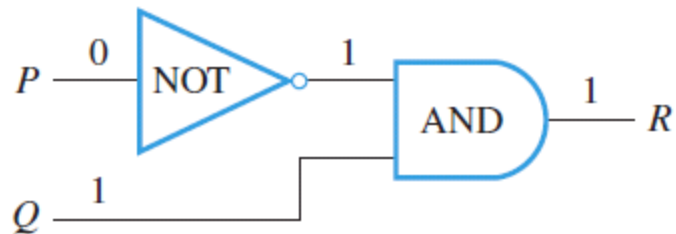
- Rules for a Combinational Circuit:
 - Never combine two input wires.
 - A single input wire can be split partway and used as input for two separate gates.
 - An output wire can be used as input.
 - No output of a gate can eventually feed back into that gate.
- Examples:



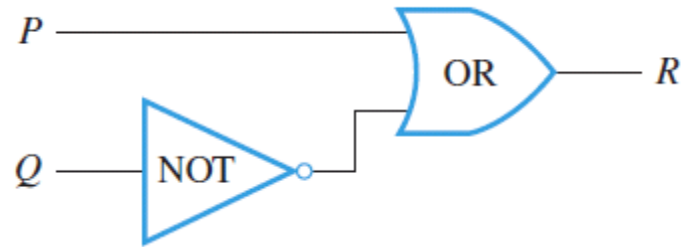
Determining Output for a Given Input



- Inputs: $P = 0$ and $Q = 1$



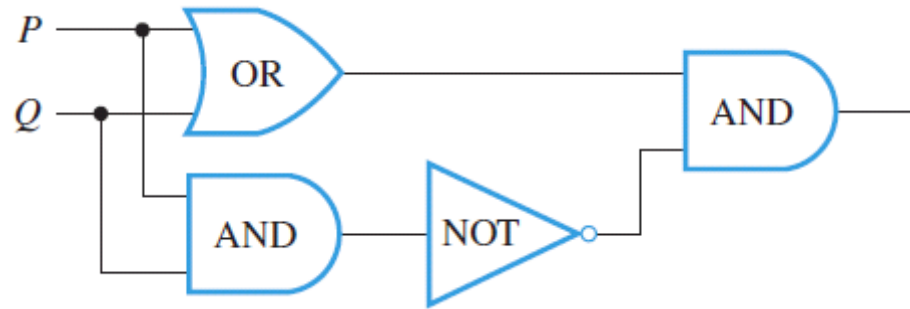
Constructing the Input/Output Table for a Circuit



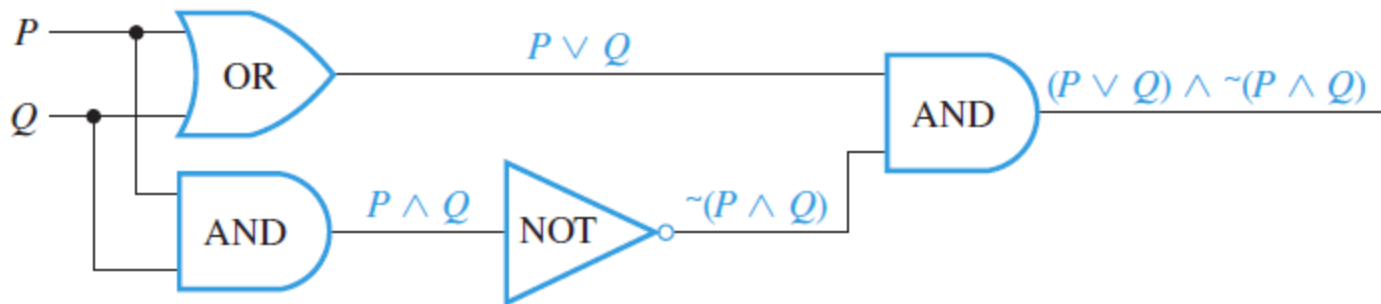
- List the four possible combinations of input signals, and find the output for each by tracing through the circuit.

Input		Output
P	Q	R
1	1	1
1	0	1
0	1	0
0	0	1

The Boolean Expression Corresponding to a Circuit

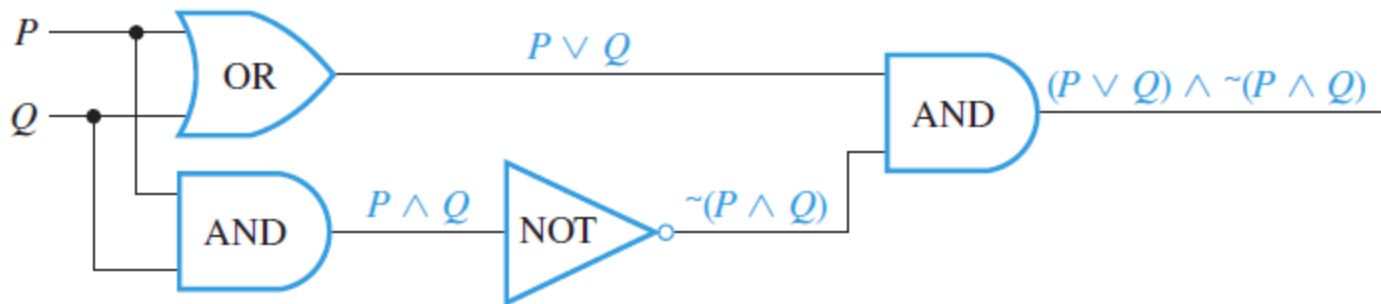
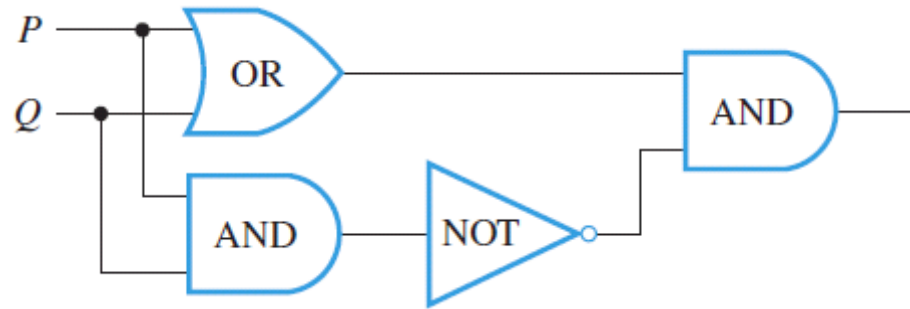


- Trace through the circuit from left to right:



- What is the result?

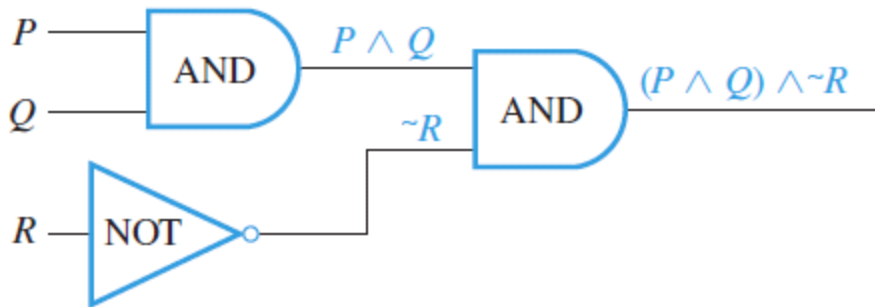
The Boolean Expression Corresponding to a Circuit



The result is: exclusive OR

Recognizer

- 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.
- Example:

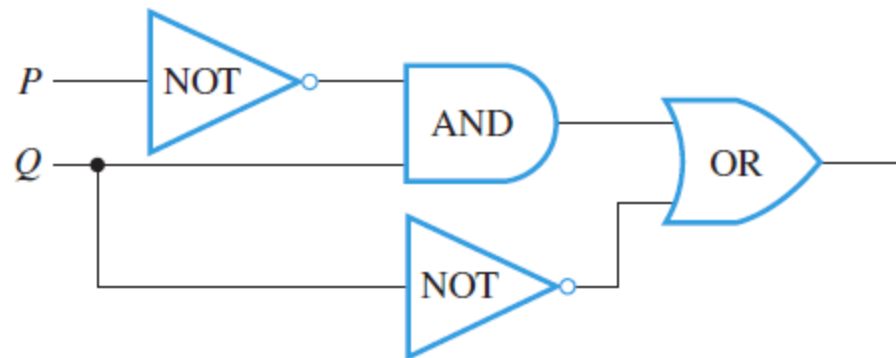


Input/Output Table for a Recognizer

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

The Circuit Corresponding to a Boolean Expression

1. Write the input variables in a column on the left side of the diagram
 2. Go from the right side of the diagram to the left, working from the outermost part of the expression to the innermost part
- Example: $(\sim P \wedge Q) \vee \sim Q$



Find a Circuit That Corresponds to an Input/Output Table

1. Construct a Boolean expression with the same truth table
 - identify each row for which the output is 1 and construct an and expression that produces a 1 for the exact combination of input values for that row

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

$$P \wedge Q \wedge R$$

$$P \wedge \sim Q \wedge R$$

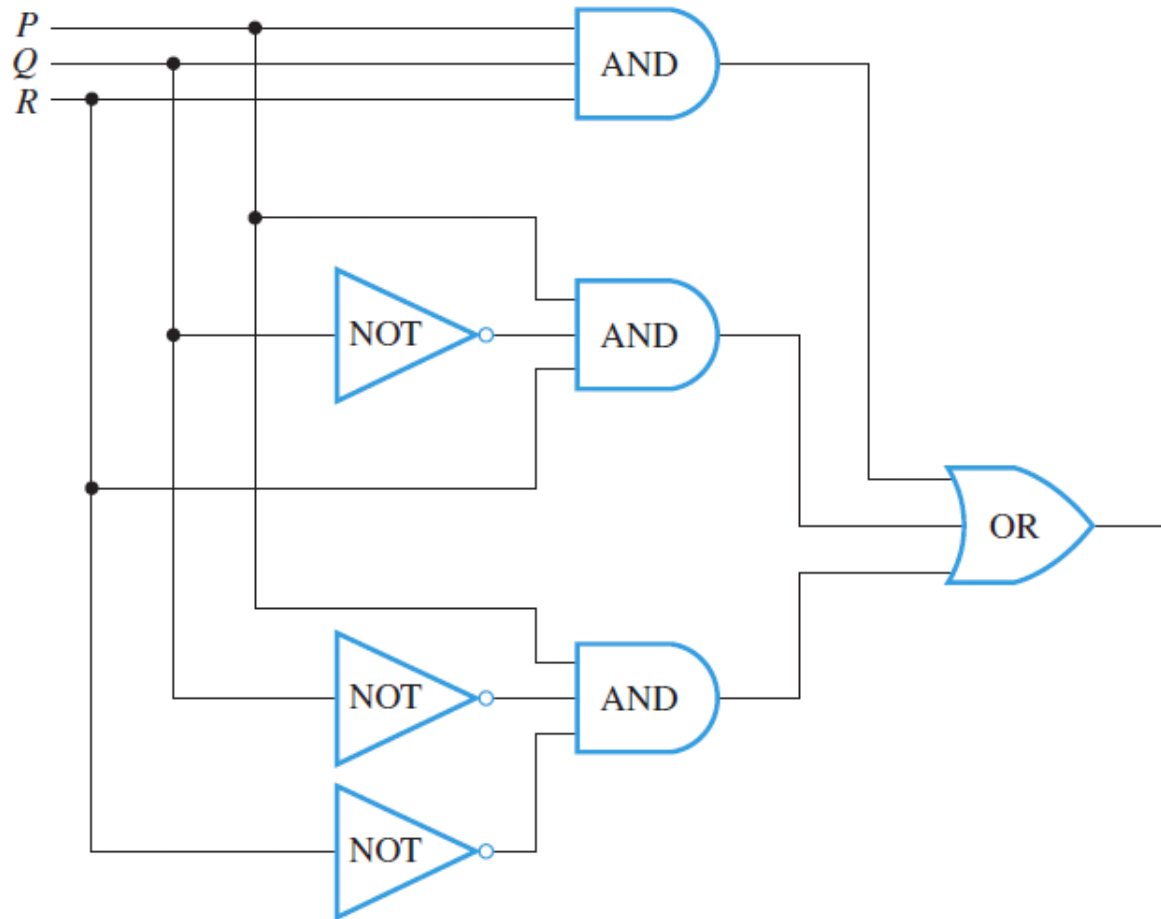
$$P \wedge \sim Q \wedge \sim R$$

$$\text{Result: } (P \wedge Q \wedge R) \vee (P \wedge \sim Q \wedge R) \vee (P \wedge \sim Q \wedge \sim R)$$

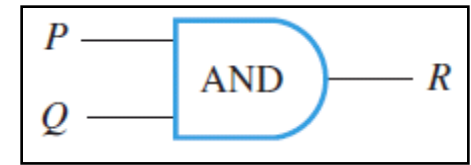
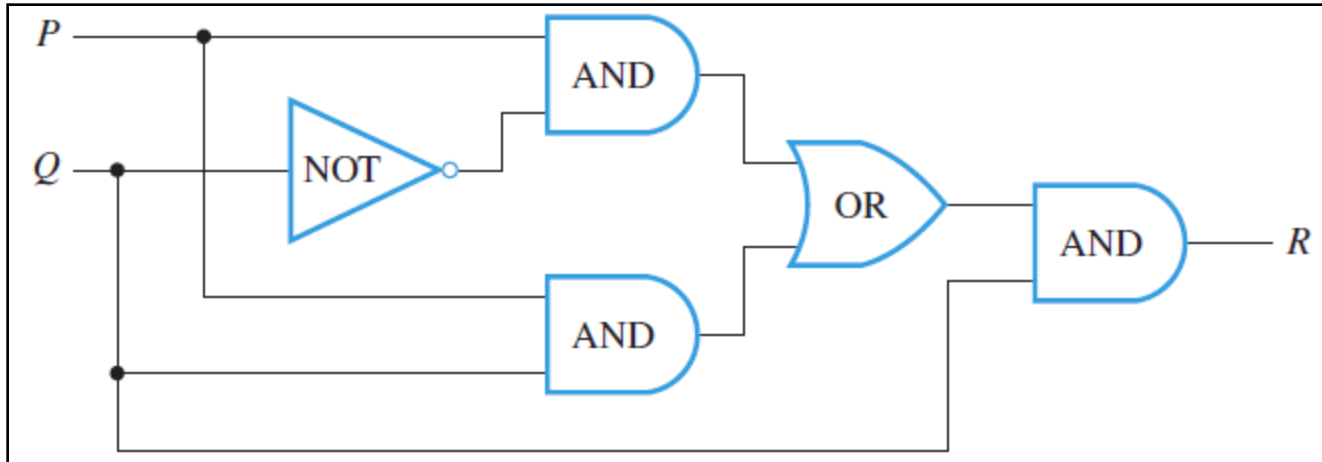
disjunctive normal form

Find a Circuit That Corresponds to an Input/Output Table

2. Construct the circuit for: $(P \wedge Q \wedge R) \vee (P \wedge \sim Q \wedge R) \vee (P \wedge \sim Q \wedge \sim R)$



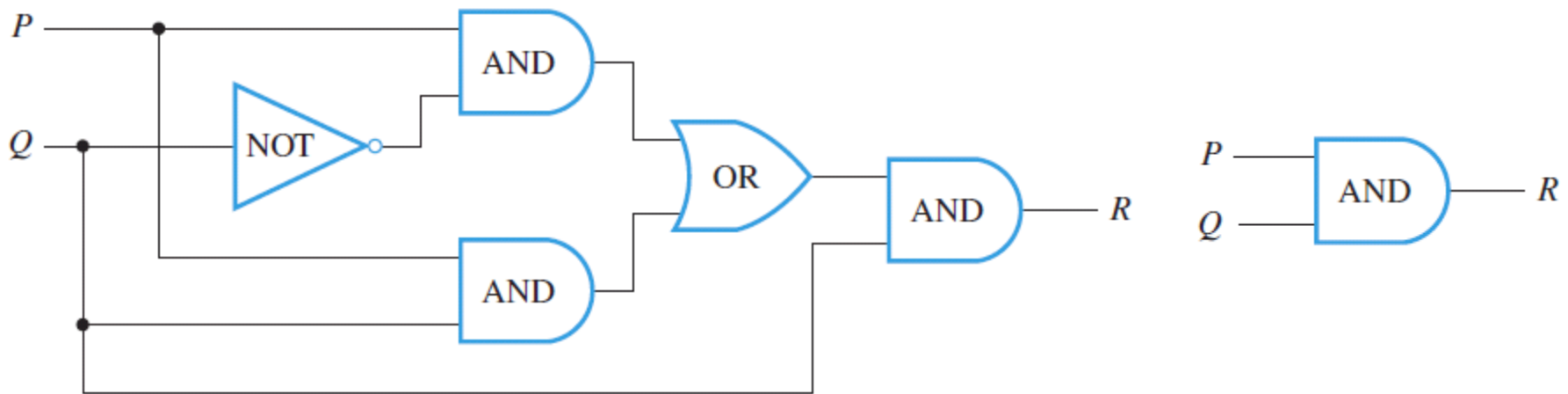
Equivalent Combinational Circuits



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

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



1. Find the Boolean expressions for each circuit.
2. Show that these expressions are logically equivalent.

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

$$\equiv (P \wedge (\sim Q \vee Q)) \wedge Q \quad \text{by the distributive law}$$

$$\equiv (P \wedge (Q \vee \sim Q)) \wedge Q \quad \text{by the commutative law for } \vee$$

$$\equiv (P \wedge T) \wedge Q \quad \text{by the negation law}$$

$$\equiv P \wedge Q \quad \text{by the identity law.}$$

NAND and NOR Gates

- A NAND-gate is a single gate that acts like an AND-gate followed by a NOT-gate

- it has the logical symbol: $|$
(called **Sheffer stroke**)

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



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

- A NOR-gate is a single gate that acts like an OR-gate followed by a NOT-gate

- it has the logical symbol: \downarrow
(called **Peirce arrow**)

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



Input		Output
<i>P</i>	<i>Q</i>	$R = P \downarrow Q$
1	1	0
1	0	0
0	1	0
0	0	1

Rewriting Expressions Using the Sheffer Stroke

- Any Boolean expression is equivalent to one written entirely with Sheffer strokes or entirely with Peirce arrows

$$\begin{aligned}\sim P &\equiv \sim(P \wedge P) && \text{by the idempotent law for } \wedge \\ &\equiv P \mid P && \text{by definition of } \mid.\end{aligned}$$

$$\begin{aligned}P \vee Q &\equiv \sim(\sim(P \vee Q)) && \text{by the double negative law} \\ &\equiv \sim(\sim P \wedge \sim Q) && \text{by De Morgan's laws} \\ &\equiv \sim((P \mid P) \wedge (Q \mid Q)) && \text{by the above } \sim P \equiv P \mid P \\ &\equiv (P \mid P) \mid (Q \mid Q) && \text{by definition of } \mid\end{aligned}$$