Gates, Circuits, and Boolean Algebra

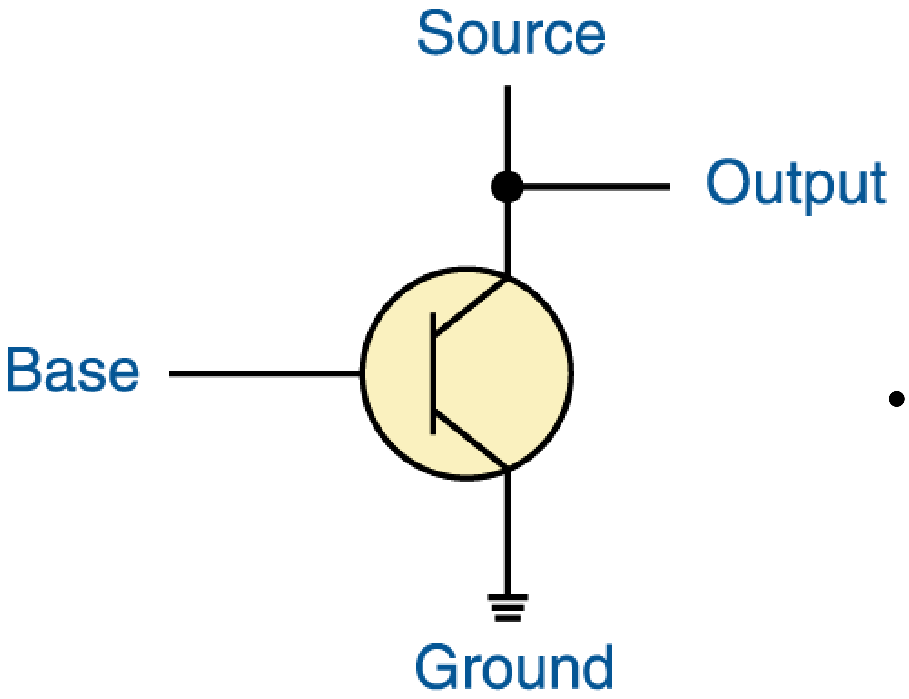
Computers and Electricity

- A **gate** is a device that performs a basic operation on electrical signals
- Gates are combined into **circuits** to perform more complicated tasks

Constructing Gates

- A **transistor** is a device that acts, depending on the voltage level of an input signal, either as a wire that conducts electricity or as a resistor that blocks the flow of electricity
  - A transistor has no moving parts, yet acts like a switch
  - It is made of a **semiconductor** material, which is neither a particularly good conductor of electricity, such as copper, nor a particularly good insulator, such as rubber

Constructing Gates



The connections of a transistor

- A transistor has three terminals
  - A source
  - A base
  - An emitter, typically connected to a ground wire
- If the electrical signal is grounded, it is allowed to flow through an alternative route to the ground (literally) where it can do no harm

Computers and Electricity

- There are three different, but equally powerful, notational methods for describing the behavior of gates and circuits
  - Boolean expressions
  - logic diagrams
  - truth tables

Computers and Electricity

- **Boolean algebra**: expressions in this algebraic notation are an elegant and powerful way to demonstrate the activity of electrical circuits

Computers and Electricity

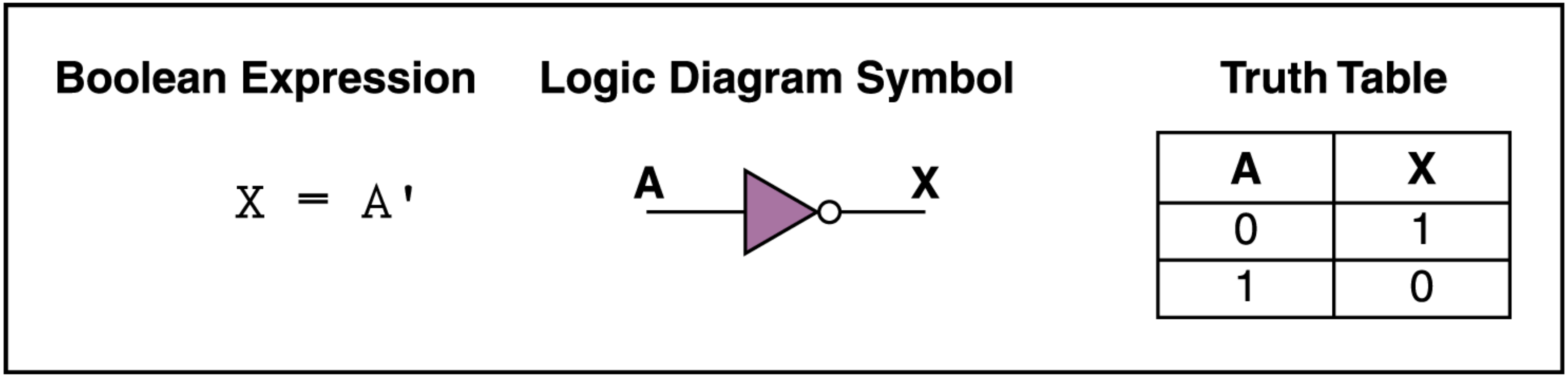
- **Logic diagram:** a graphical representation of a circuit
  - Each type of gate is represented by a specific graphical symbol
- **Truth table:** defines the function of a gate by listing all possible input combinations that the gate could encounter, and the corresponding output

Gates

- Let’s examine the processing of the following six types of gates
  - NOT
  - AND
  - OR
  - XOR
  - NAND
  - NOR
- Typically, logic diagrams are black and white, and the gates are distinguished only by their shape

NOT Gate

- A NOT gate accepts one input value and produces one output value



$X = A'$

A	X
0	1
1	0

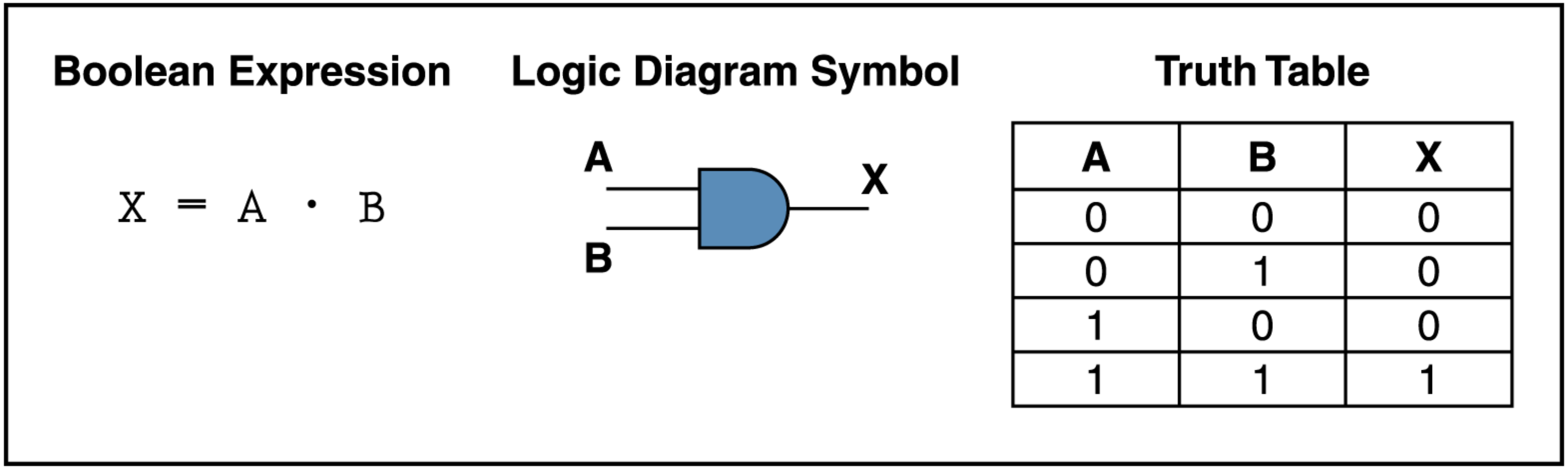
Various representations of a NOT gate

NOT Gate

- By definition, if the input value for a NOT gate is 0, the output value is 1, and if the input value is 1, the output is 0
- A NOT gate is sometimes referred to as an *inverter* because it inverts the input value

AND Gate

- An AND gate accepts two input signals
- If the two input values for an AND gate are both 1, the output is 1; otherwise, the output is 0



Boolean Expression

$X = A \cdot B$

Logic Diagram Symbol

Truth Table

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Various representations of an AND gate

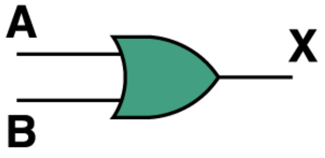
OR Gate

- If the two input values are both 0, the output value is 0; otherwise, the output is 1

Boolean Expression

$X = A + B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Various representations of a OR gate

XOR Gate

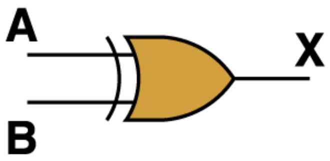
- XOR, or *exclusive* OR, gate
  - An XOR gate produces 0 if its two inputs are the same, and a 1 otherwise
  - Note the difference between the XOR gate and the OR gate; they differ only in one input situation
  - When both input signals are 1, the OR gate produces a 1 and the XOR produces a 0

XOR Gate

Boolean Expression

$X = A \oplus B$

Logic Diagram Symbol



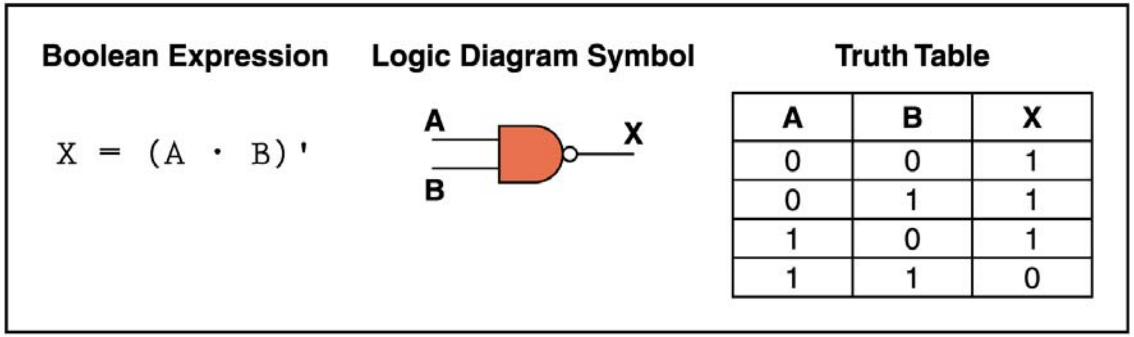
Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Various representations of an XOR gate

NAND and NOR Gates

- The NAND and NOR gates are essentially the opposite of the AND and OR gates, respectively



Boolean Expression

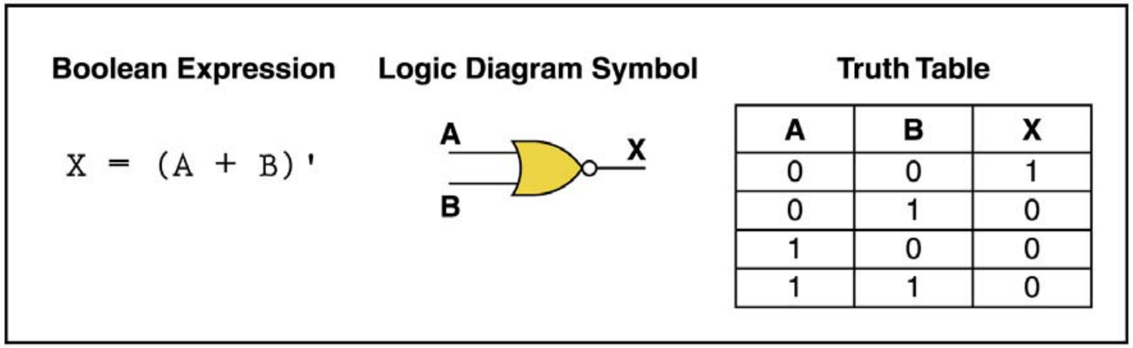
$X = (A \cdot B)'$

Logic Diagram Symbol

Truth Table

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

Various representations of a NAND gate



Boolean Expression

$X = (A + B)'$

Logic Diagram Symbol

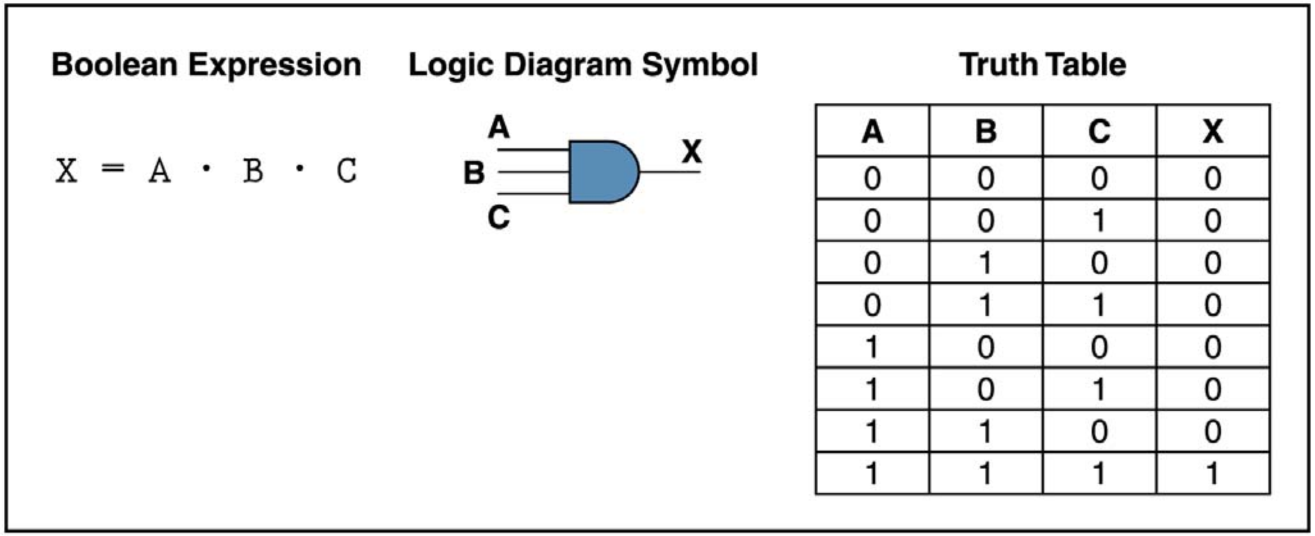
Truth Table

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Various representations of a NOR gate

Gates with More Inputs

- Gates can be designed to accept three or more input values
- A three-input AND gate, for example, produces an output of 1 only if all input values are 1



Boolean Expression

$X = A \cdot B \cdot C$

Logic Diagram Symbol

Truth Table

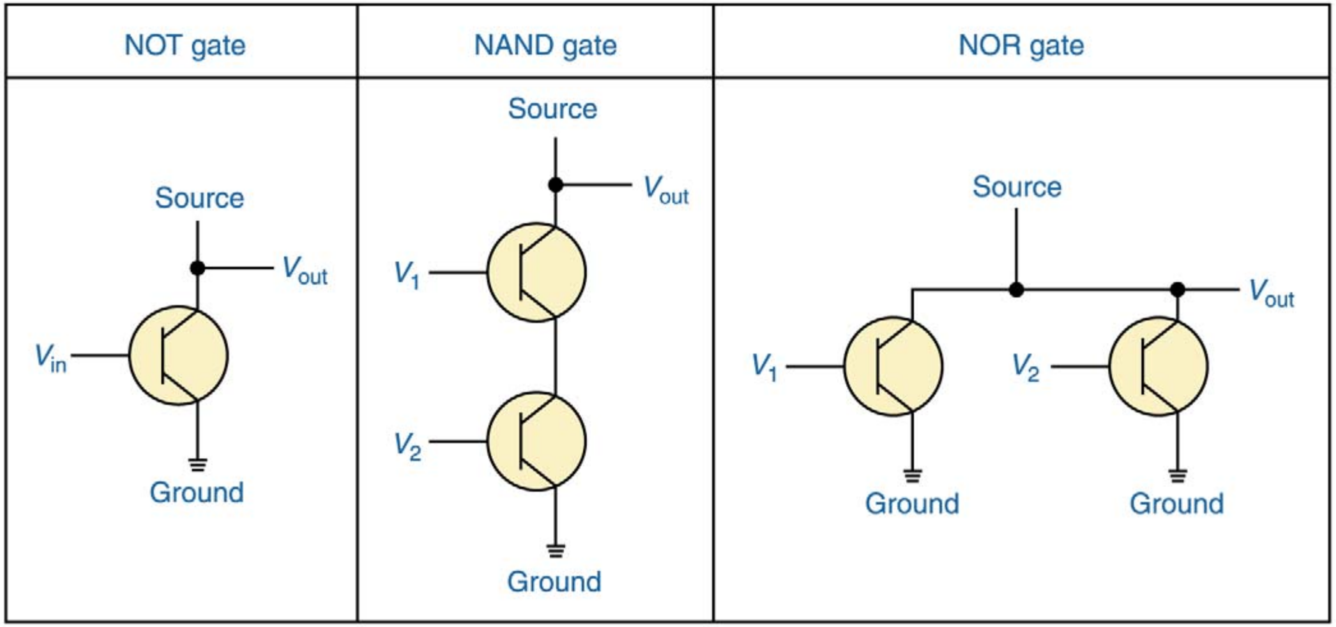
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Various representations of a three-input AND gate

Constructing Gates

It turns out that, because the way a transistor works, the easiest gates to create are the NOT, NAND, and NOR gates. Source is a consistent voltage source.

$V_{out} = V_{in}$’; $V_{out} =(V_1V_2)$’; $V_{out} =(V_1+V_2)$’



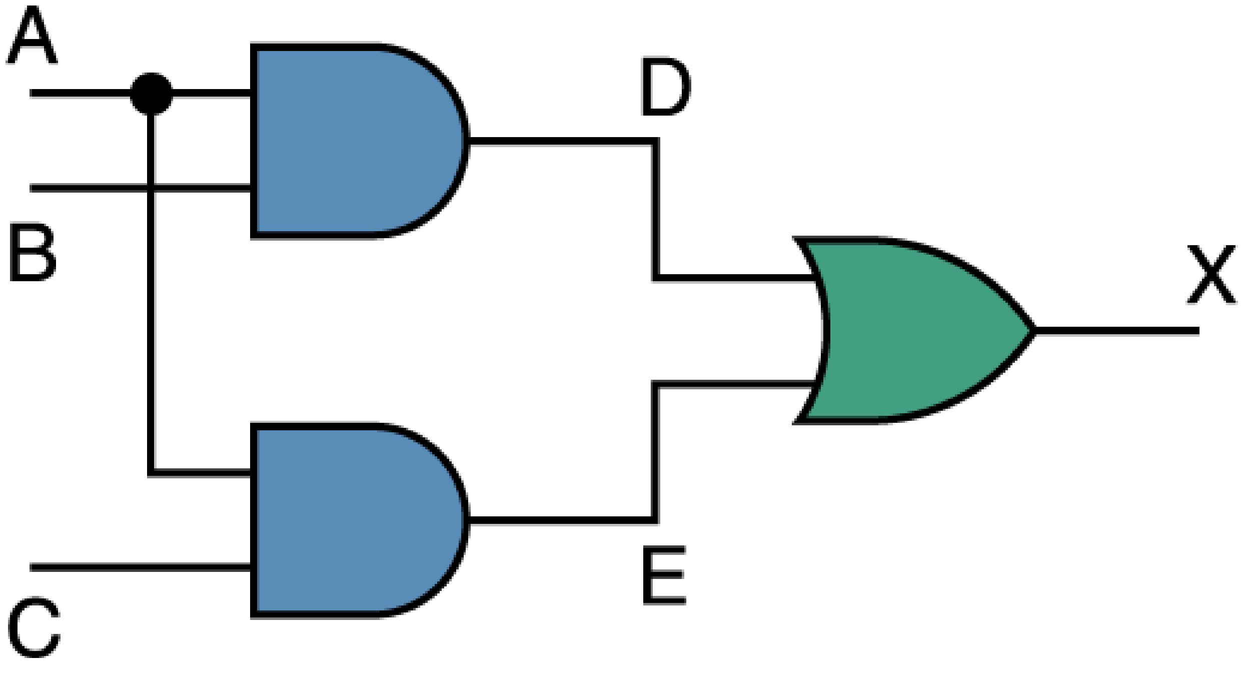
Constructing gates using transistors

Circuits

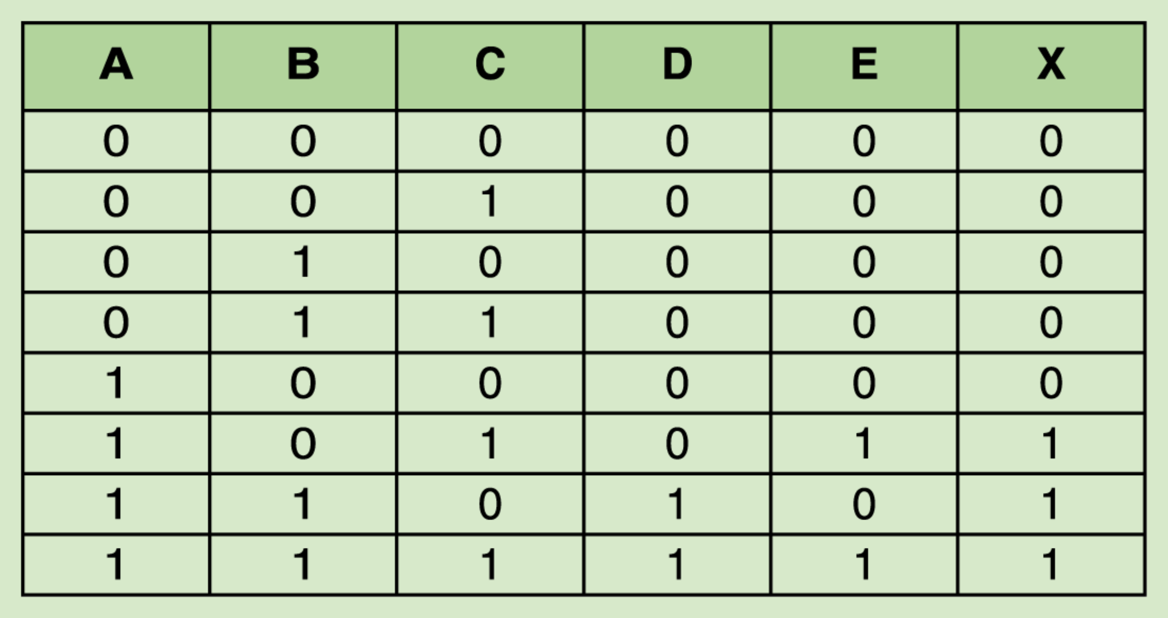
- Two general categories
  - In a **combinational circuit**, the input values explicitly determine the output
  - In a **sequential circuit**, the output is a function of the input values as well as the existing state of the circuit
- As with gates, we can describe the operations of entire circuits using three notations
  - Boolean expressions
  - logic diagrams
  - truth tables

Combinational Circuits

- Gates are combined into circuits by using the output of one gate as the input for another



Combinational Circuits



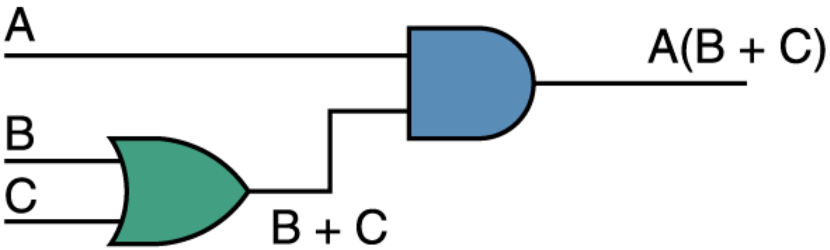
A	B	C	D	E	X
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

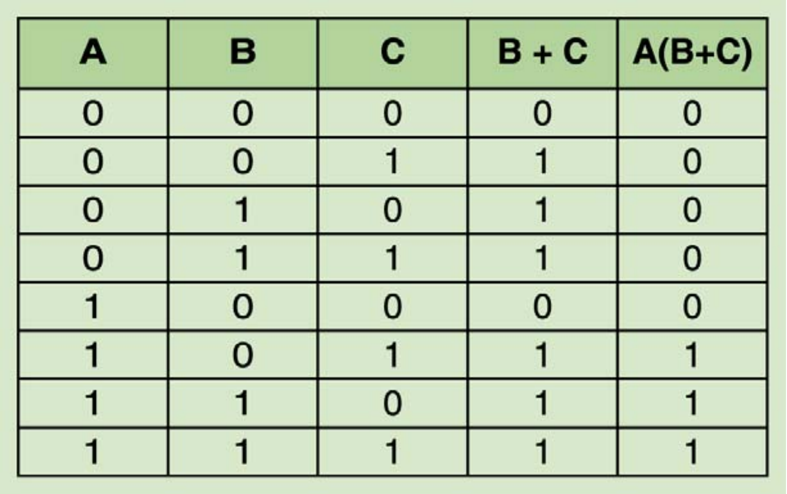
- Because there are three inputs to this circuit, eight rows are required to describe all possible input combinations
- This same circuit using Boolean algebra:

$$(AB + AC)$$

Now let's go the other way; let's take a Boolean expression and draw

- Consider the following Boolean expression: $A(B + C)$

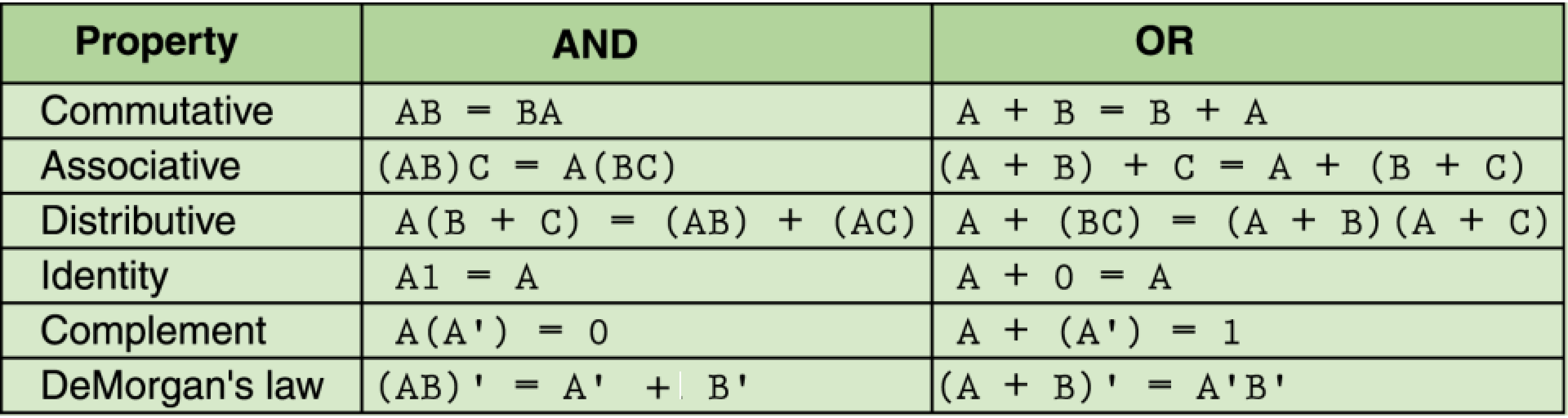




A	B	C	B + C	A(B+C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

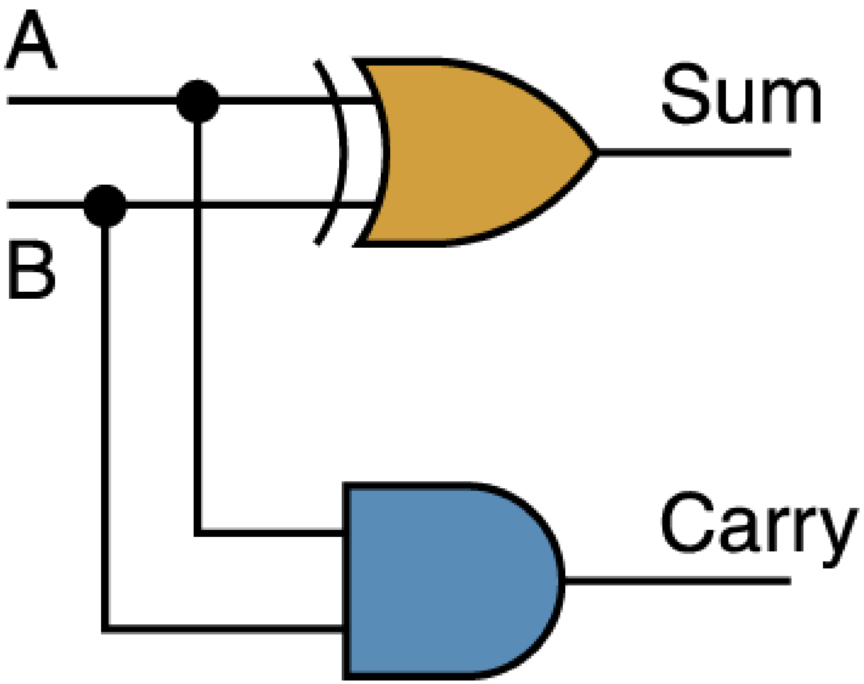
- Now compare the final result column in this truth table to the truth table for the previous example
  - They are identical

Properties of Boolean Algebra



Property	AND	OR
Commutative	$AB = BA$	$A + B = B + A$
Associative	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive	$A(B + C) = (AB) + (AC)$	$A + (BC) = (A + B)(A + C)$
Identity	$A1 = A$	$A + 0 = A$
Complement	$A(A') = 0$	$A + (A') = 1$
DeMorgan's law	$(AB)' = A' + B'$	$(A + B)' = A'B'$

Adders



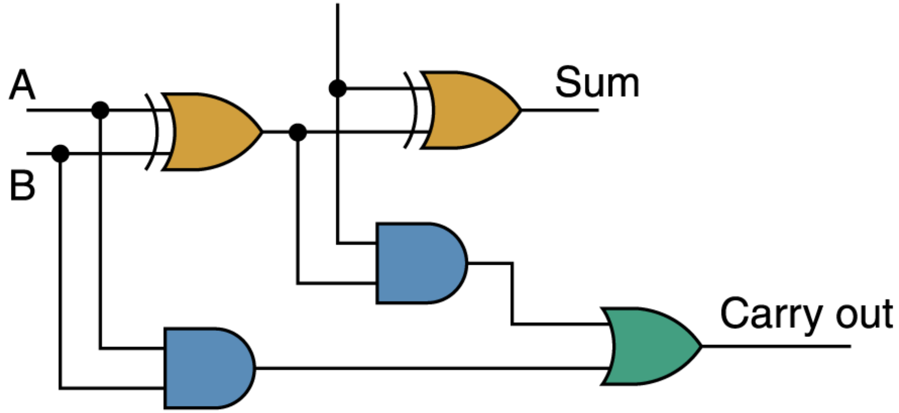
- Circuit diagram representing a half adder
- Two Boolean expressions:

$$\text{sum} = A \oplus B$$
$$\text{carry} = AB$$

Adders

- A circuit called a **full adder** takes the carry-in value into account

Logic Diagram



Truth Table

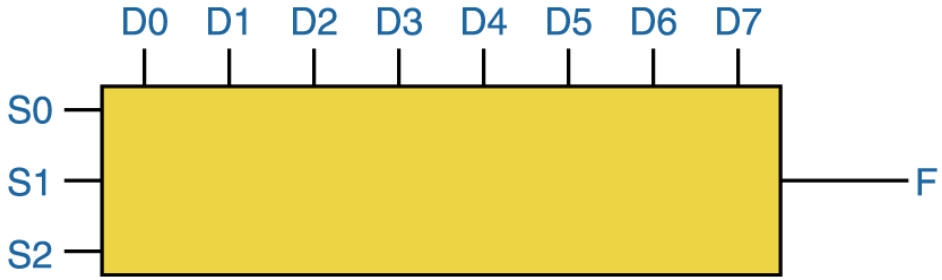
A	B	Carry-in	Sum	Carry-out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A full adder

Multiplexers

- **Multiplexer** is a general circuit that produces a single output signal
  - The output is equal to one of several input signals to the circuit
  - The multiplexer selects which input signal is used as an output signal based on the value represented by a few more input signals, called *select signals* or *select control lines*

Multiplexers



A block diagram of a multiplexer with three select control lines

S0	S1	S2	F
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

- The control lines S0, S1, and S2 determine which of eight other input lines (D0 through D7) are routed to the output (F)

Circuits as Memory

- Digital circuits can be used to store information
- These circuits form a sequential circuit, because the output of the circuit is also used as input to the circuit