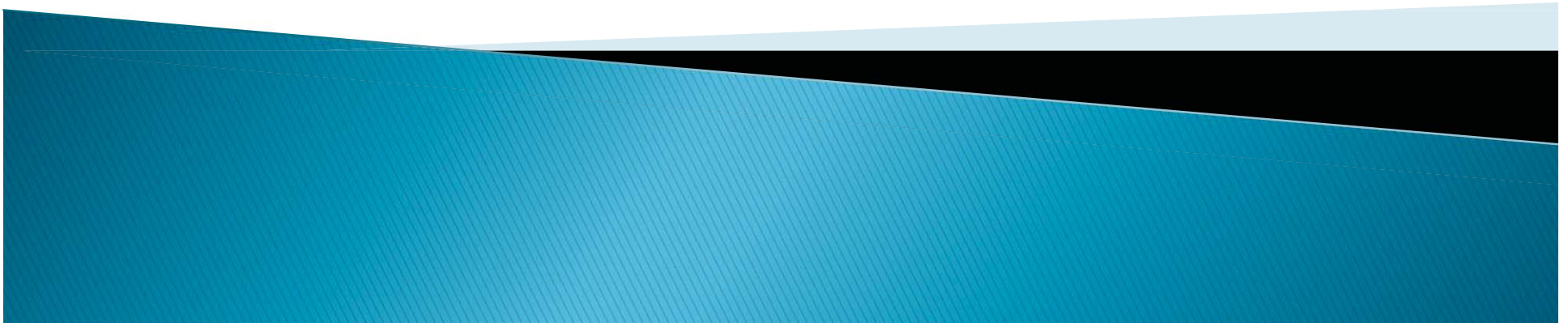


GATES AND CIRCUITS

The design and construction of
electrical circuits to implement
logical and arithmetical
operations.

CHAPTER 4



Chapter Goals

- ▶ Identify the basic gates and describe the behaviour of each;
- ▶ Describe how gates are implemented using transistors;
- ▶ Combine basic gates into circuits;
- ▶ Describe the behaviour of a gate or circuit using Boolean expressions, truth tables, and logic diagrams;



Chapter Goals

- ▶ Compare and contrast a half adder and a full adder;
- ▶ Explain how an S–R latch operates;
- ▶ Describe the characteristics of the four generations of integrated circuits.



Definitions

Two valued Logic (F/T, 0/1)

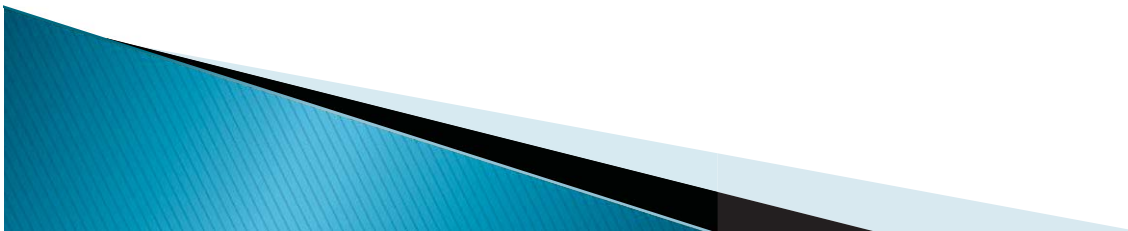
Allows formal decision on logical expressions

Gate

A device that performs a basic operation on electrical signals.

Circuits

Gates combined to perform more complicated tasks.



Two Valued Logic

Statements:

$A = \textit{My grandmother is alive}$

$B = \textit{My grandfather is alive}$

NOT $A = \textit{My grandmother is dead}$

A **AND** $B = \textit{Both my grandparents are alive}$

A **OR** $B = \textit{At least one of my grandparents is alive}$



Two Valued Logic

Statements:

$A = \textit{My grandmother is alive}$

$B = \textit{My grandfather is alive}$

$A \text{ XOR } B = \textit{Exactly one of my grandparents is alive}$

$A \text{ NAND } B = \textit{My grandparents are not both alive}$

$A \text{ NOR } B = \textit{My grandparents are both dead}$



Describing Gates and Circuits

Boolean expressions

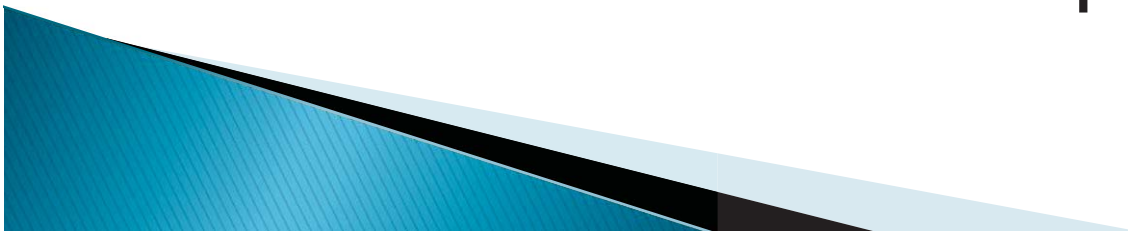
Uses Boolean algebra, a mathematical notation for expressing two-valued logic.

Logic diagrams

A graphical representation of a circuit; each gate has its own symbol.

Truth tables

A table showing all possible input values and the associated output values.



Gates

Six types of gates

- NOT
- AND
- OR
- XOR
- NAND
- NOR



NOT Gate

A **NOT** gate accepts one input signal (0 or 1) and returns the opposite signal as output

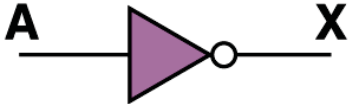
Boolean Expression	Logic Diagram Symbol	Truth Table						
$X = A'$		<table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	X	0	1	1	0
A	X							
0	1							
1	0							

Figure 4.1 Various representations of a NOT gate



AND Gate

An **AND** gate accepts two input signals if both are 1, the output is 1; otherwise the output is 0

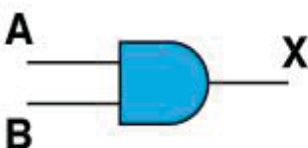
Boolean Expression	Logic Diagram Symbol	Truth Table															
$X = A \cdot B$		<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	X	0	0	0	0	1	0	1	0	0	1	1	1
A	B	X															
0	0	0															
0	1	0															
1	0	0															
1	1	1															

Figure 4.2 Various representations of an AND gate

OR Gate

An OR gate accepts two input signals
If both are 0, the output is 0;
otherwise, the output is 1

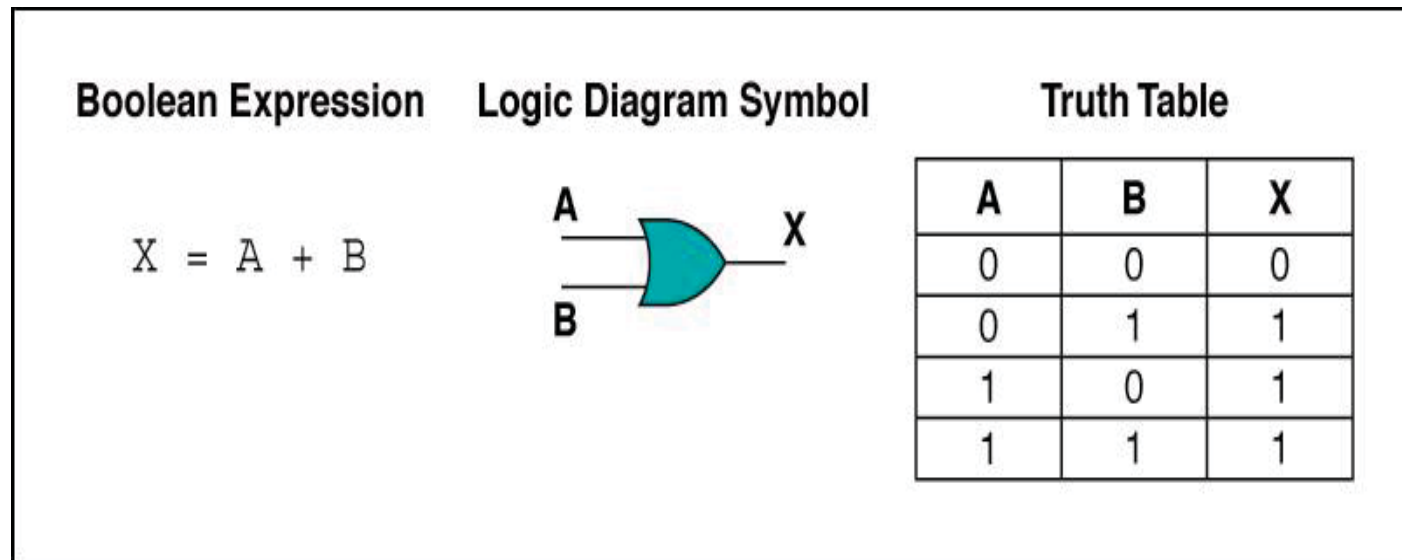


Figure 4.3 Various representations of a OR gate

XOR Gate

An **XOR** gate accepts two input signals

If both are the same, the output is 0;

otherwise, the output is 1

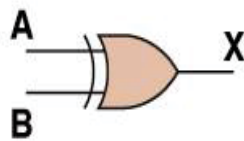
Boolean Expression	Logic Diagram Symbol	Truth Table															
$X = A \oplus B$		<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	X	0	0	0	0	1	1	1	0	1	1	1	0
A	B	X															
0	0	0															
0	1	1															
1	0	1															
1	1	0															

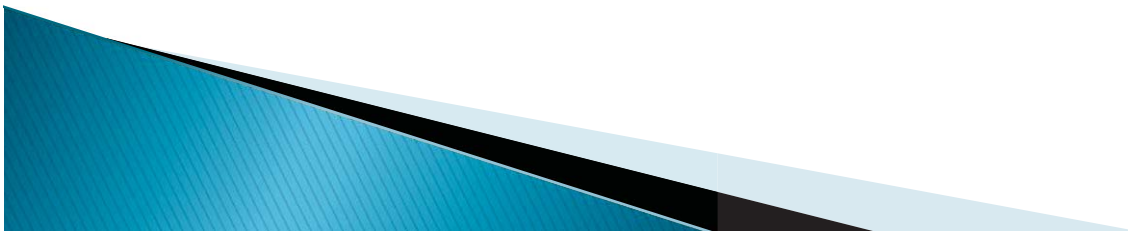
Figure 4.4 Various representations of an XOR gate

XOR Gate

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 is called the *exclusive OR*



NAND Gate

The **NAND** gate accepts two input signals
If both are 1, the output is 0;
otherwise, the output is 1

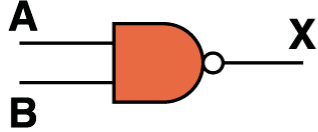
Boolean Expression	Logic Diagram Symbol	Truth Table															
$X = (A \cdot B)'$		<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	X	0	0	1	0	1	1	1	0	1	1	1	0
A	B	X															
0	0	1															
0	1	1															
1	0	1															
1	1	0															

Figure 4.5 Various representations of a NAND gate

NOR Gate

The **NOR** gate accepts two input signals

If both are 0, the output is 1; otherwise,
the output is 0

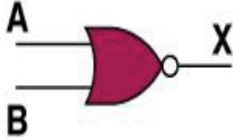
Boolean Expression	Logic Diagram Symbol	Truth Table															
$X = (A + B)'$		<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	X	0	0	1	0	1	0	1	0	0	1	1	0
A	B	X															
0	0	1															
0	1	0															
1	0	0															
1	1	0															

Figure 4.6 Various representations of a NOR gate

Review of Gate Processing

- ▶ A **NOT** gate **inverts** its single input
- ▶ An **AND** gate produces **1** if **both** input values are **1**
- ▶ An **OR** gate produces **0** if **both** input values are **0**
- ▶ An **XOR** gate produces **0** if input values are the **same**
- ▶ A **NAND** gate produces **0** if **both** inputs are **1**
- ▶ A **NOR** gate produces a **1** if both inputs are **0**



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**

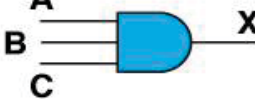
Boolean Expression	Logic Diagram Symbol	Truth Table																																				
$X = A \cdot B \cdot C$		<table><tr><th>A</th><th>B</th><th>C</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></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
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																																			

Figure 4.7 Various representations of a three-input AND gate

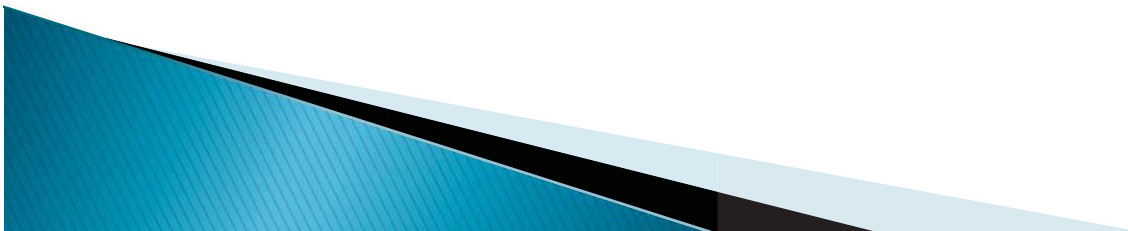
Constructing Gates

Transistor

A device that acts either as a wire that conducts electricity or as a resistor that blocks the flow of electricity, depending on the voltage level of an input signal

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 nor a particularly good insulator



Constructing Gates

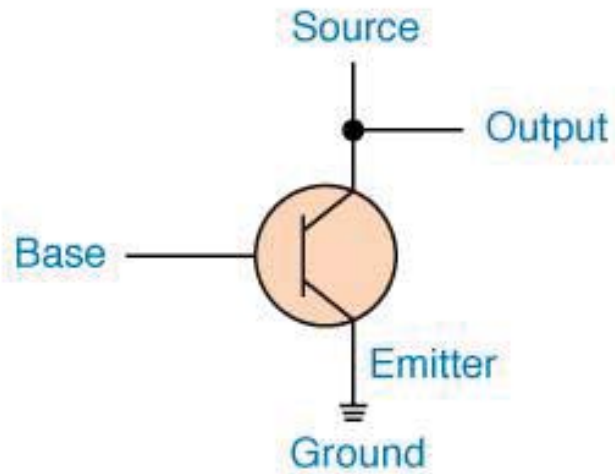


Figure 4.8 The connections of a transistor

A transistor has three terminals

- A source
- A base
- An emitter, typically connected to a ground wire

If the Base is “on”, then the transistor Emits, and is Earthed.

If the Base is “off”, then the transistor becomes an insulator and the Output is on.

Constructing Gates

The easiest gates to create are the NOT, NAND, and NOR gates

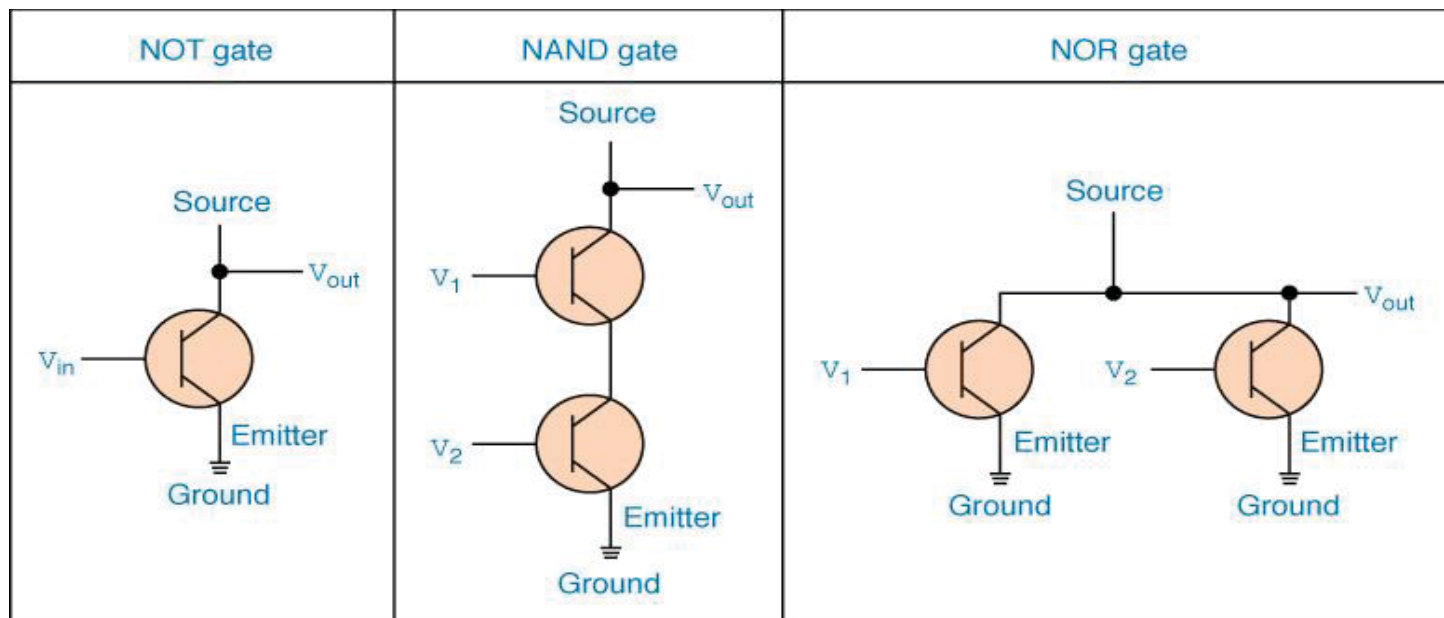


Figure 4.9 Constructing gates using transistors

Circuits

Combinational circuit

The input values explicitly determine the output

Sequential circuit

The output is a function of the input values and the existing state of the circuit

We describe the circuit operations using

- Boolean expressions

- Logic diagrams

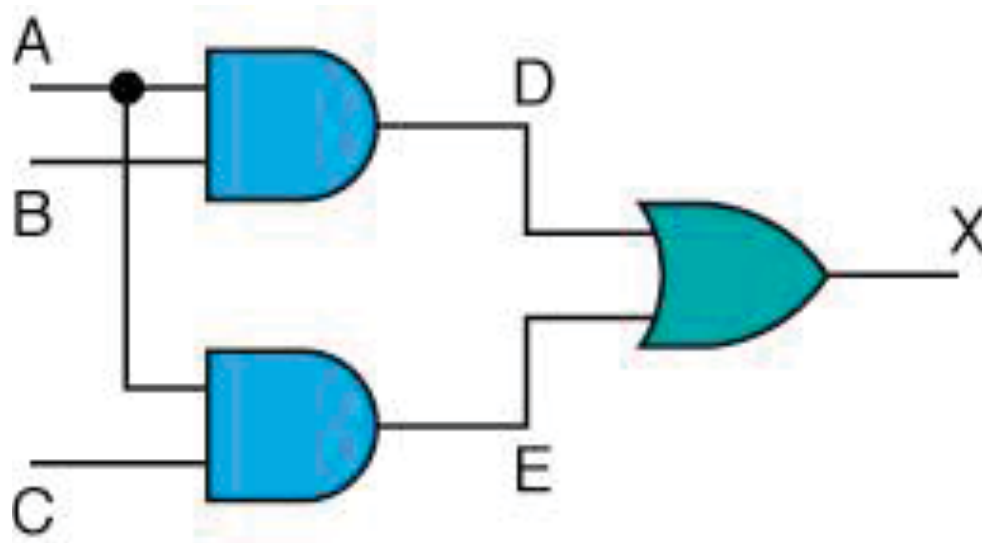
- Truth tables

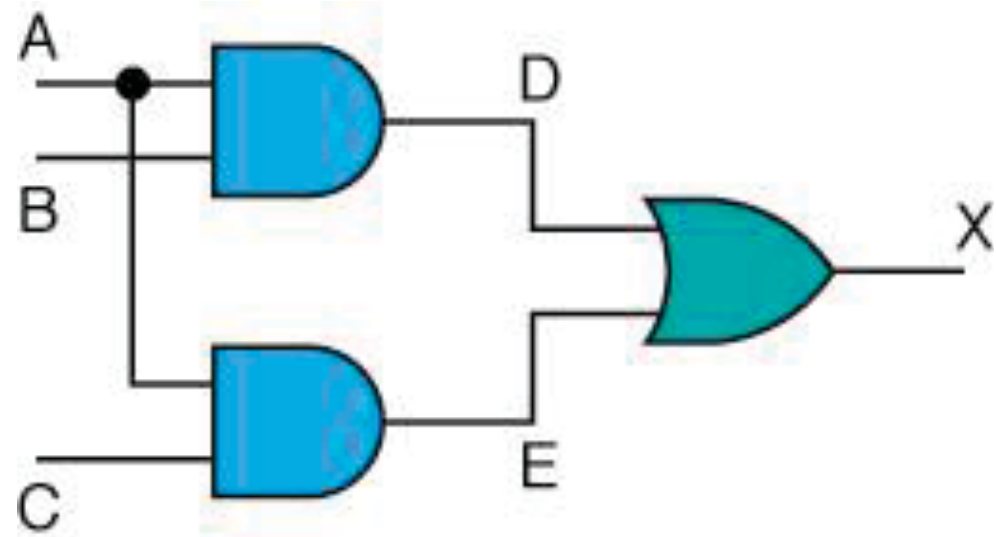


Combinational Circuits

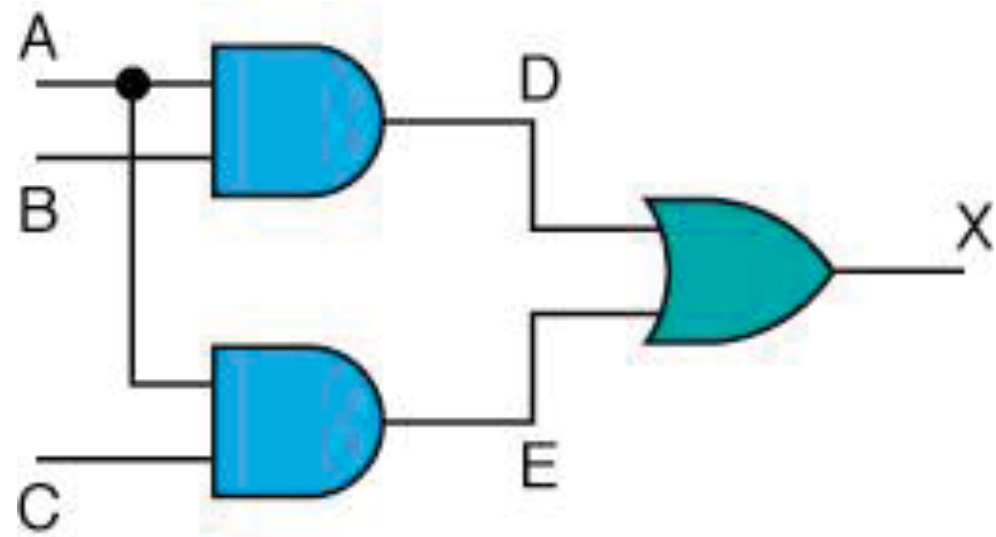
Gates are combined into circuits by using the output of one gate as the input for another

$$(A.B) + (A.C)$$

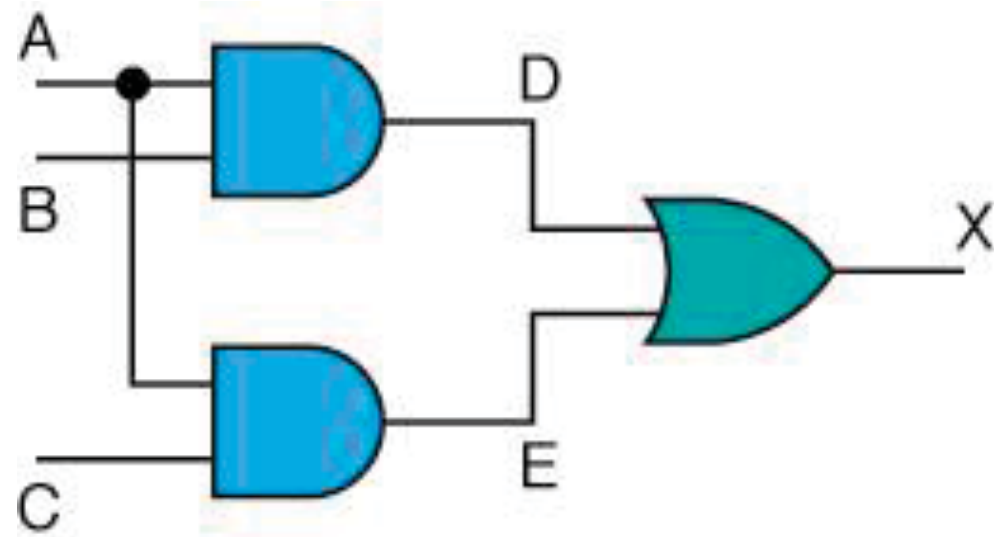




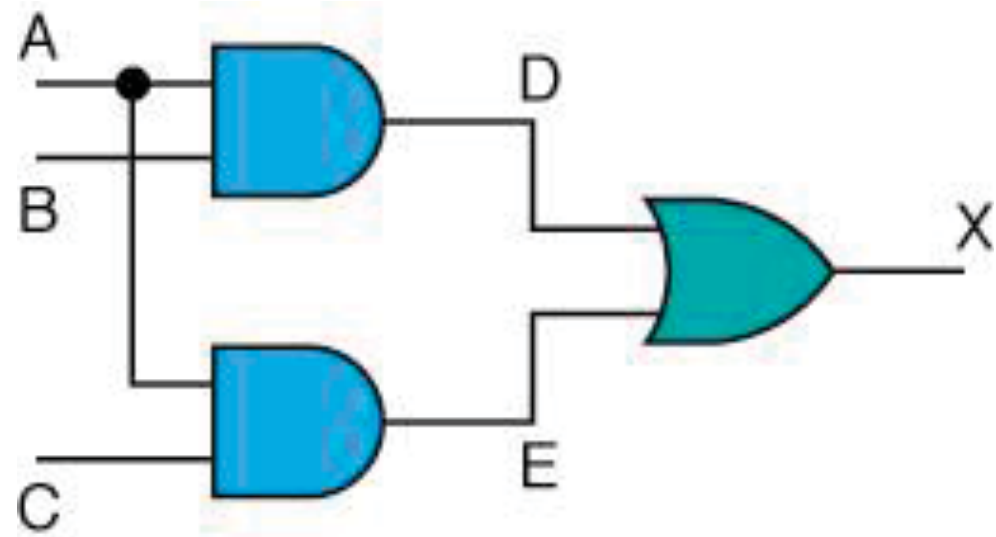
A	B	C	D	E	X
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			



A	B	C	D	E	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	1		
1	1	1	1		



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

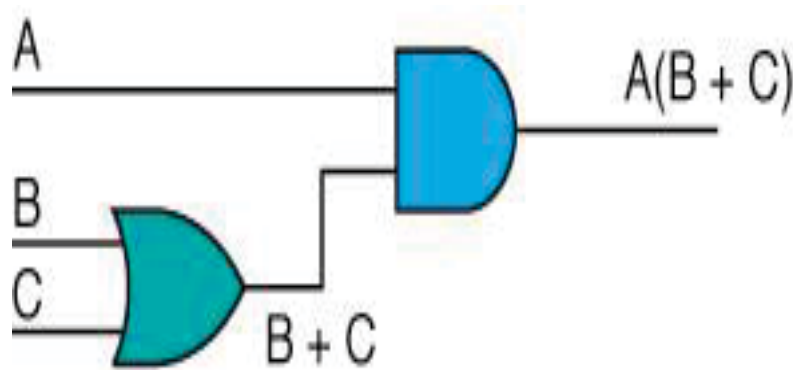


A	B	C	D	E	X
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

Combinational Circuits

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

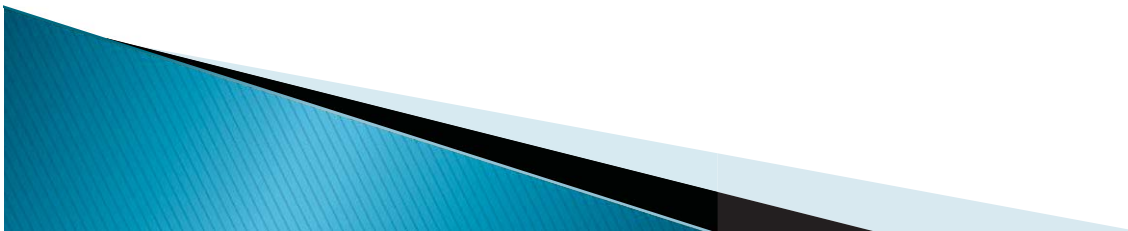
Combinational Circuits

Circuit equivalence

Two circuits that produce the same output for identical input

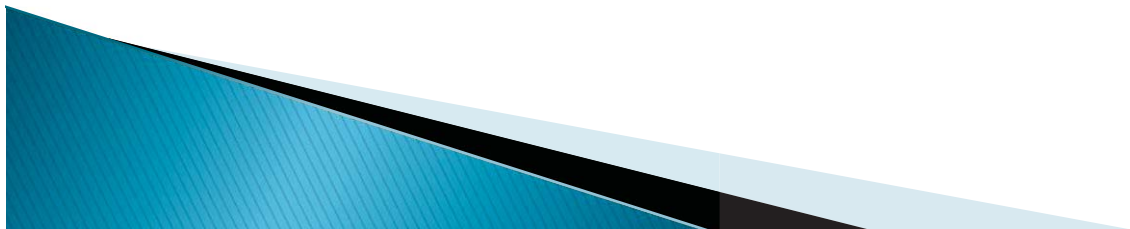
Boolean algebra allows us to apply provable mathematical principles to help design circuits

$A.(B + C) = A.B + A.C$ (distributive law) so circuits must be equivalent



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	$(AB)' = A' + B'$	$(A + B)' = A'B'$



Proving Boolean Relations

We can prove these rules by setting out all the values in a truth table and showing that the truth table for each side is the same.

For example take the Distributive Law:

$$A + (BC) = (A + B).(A + C)$$



$$A + (BC) = (A + B) \cdot (A + C)$$

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



$$A + (BC) = (A + B) \cdot (A + C)$$

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



$$A + (BC) = (A + B) \cdot (A + C)$$

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



$$A + (BC) = (A + B) \cdot (A + C)$$

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



$$A + (BC) = (A + B) \cdot (A + C)$$

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



$$A + (BC) = (A + B) \cdot (A + C)$$

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



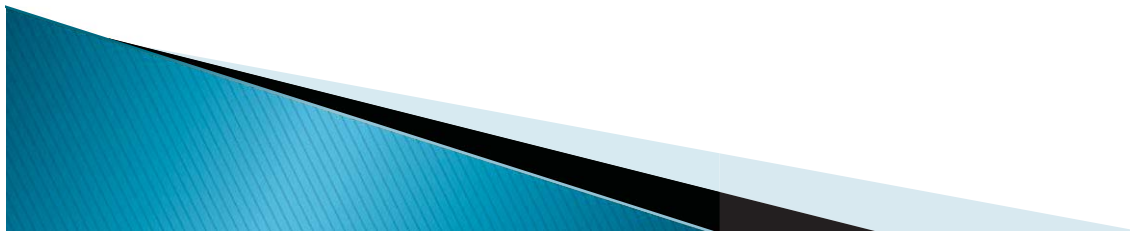
$$A + (BC) = (A + B) \cdot (A + C)$$

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



De Morgan's Law $(A+B)' = A'B'$

A	B	A+B	$(A+B)'$	A'	B'	A'B'
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0



De Morgan's Law $(A+B)' = A'B'$

A	B	A+B	$(A+B)'$	A'	B'	$A'B'$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0



Adders

At the digital logic level, addition is performed in binary

Addition operations are carried out by special circuits called, appropriately, **adders**

