

COMPUTER CODES AND BOOLEAN ALGEBRA

Application of Information and Communication Technologies

Dr. Muhammad Abdullah



Faculty of Computing and Information Technology (FCIT)
University of the Punjab, Lahore, Pakistan.

Data Types

- **Numeric Data** consists of only numbers 0, 1, 2, ..., 9
- **Alphabetic Data** consists of only the letters A, B, C, ..., Z, in both uppercase and lowercase, and blank character
- **Alphanumeric Data** is a string of symbols where a symbol may be one of the letters A, B, C, ..., Z, in either uppercase or lowercase, or one of the digits 0, 1, 2, ..., 9, or a special character, such as + - * / , . () = etc.

Computer Codes

- Computer codes are used for internal representation of data in computers
- As computers use binary numbers for internal data representation, computer codes use binary coding schemes
- In binary coding, every symbol that appears in the data is represented by a group of bits
- The group of bits used to represent a symbol is called a **byte**

Computer Codes

- As most modern coding schemes use 8 bits to represent a symbol, the term byte is often used to mean a group of 8 bits
- Commonly used computer codes are BCD, EBCDIC, and ASCII

BCD

- BCD stands for **B**inary **C**oded **D**ecimal
- It is one of the early computer codes
- It uses 6 bits to represent a symbol
- It can represent 64 (2^6) different characters

Coding of Alphabetic and Numeric Characters in BCD

Char	BCD Code		Octal
	Zone	Digit	
A	11	0001	61
B	11	0010	62
C	11	0011	63
D	11	0100	64
E	11	0101	65
F	11	0110	66
G	11	0111	67
H	11	1000	70
I	11	1001	71
J	10	0001	41
K	10	0010	42
L	10	0011	43
M	10	0100	44

Char	BCD Code		Octal
	Zone	Digit	
N	10	0101	45
O	10	0110	46
P	10	0111	47
Q	10	1000	50
R	10	1001	51
S	01	0010	22
T	01	0011	23
U	01	0100	24
V	01	0101	25
W	01	0110	26
X	01	0111	27
Y	01	1000	30
Z	01	1001	31

Coding of Alphabetic and Numeric Characters in BCD

Character	BCD Code		Octal Equivalent
	Zone	Digit	
1	00	0001	01
2	00	0010	02
3	00	0011	03
4	00	0100	04
5	00	0101	05
6	00	0110	06
7	00	0111	07
8	00	1000	10
9	00	1001	11
0	00	1010	12

BCD Coding Scheme Example

Example

Show the binary digits used to record the word BASE in BCD

Solution:

B = 110010 in BCD binary notation

A = 110001 in BCD binary notation

S = 010010 in BCD binary notation

E = 110101 in BCD binary notation

So the binary digits

<u>110010</u>	<u>110001</u>	<u>010010</u>	<u>110101</u>
B	A	S	E

will record the word BASE in BCD

BCD Coding Scheme Example

Example

Using octal notation, show BCD coding for the word DIGIT

Solution:

D = 64 in BCD octal notation

I = 71 in BCD octal notation

G = 67 in BCD octal notation

I = 71 in BCD octal notation

T = 23 in BCD octal notation

Hence, BCD coding for the word DIGIT in octal notation will be

<u>64</u>	<u>71</u>	<u>67</u>	<u>71</u>	<u>23</u>
D	I	G	I	T

EBCDIC

- EBCDIC stands for **E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode
- It uses 8 bits to represent a symbol
- It can represent 256 (2^8) different characters

Coding of Alphabetic and Numeric Characters in EBCDIC

Char	EBCDIC Code		Hex
	Digit	Zone	
A	1100	0001	C1
B	1100	0010	C2
C	1100	0011	C3
D	1100	0100	C4
E	1100	0101	C5
F	1100	0110	C6
G	1100	0111	C7
H	1100	1000	C8
I	1100	1001	C9
J	1101	0001	D1
K	1101	0010	D2
L	1101	0011	D3
M	1101	0100	D4

Char	EBCDIC Code		Hex
	Digit	Zone	
N	1101	0101	D5
O	1101	0110	D6
P	1101	0111	D7
Q	1101	1000	D8
R	1101	1001	D9
S	1110	0010	E2
T	1110	0011	E3
U	1110	0100	E4
V	1110	0101	E5
W	1110	0110	E6
X	1110	0111	E7
Y	1110	1000	E8
Z	1110	1001	E9

Coding of Alphabetic and Numeric Characters in EBCDIC

Character	EBCDIC Code		Hexadecimal Equivalent
	Digit	Zone	
0	1111	0000	F0
1	1111	0001	F1
2	1111	0010	F2
3	1111	0011	F3
4	1111	0100	F4
5	1111	0101	F5
6	1111	0110	F6
7	1111	0111	F7
8	1111	1000	F8
9	1111	1001	F9

EBCDIC Coding Example

Example

Using binary notation, write EBCDIC coding for the word BIT. How many bytes are required for this representation?

Solution:

B = 1100 0010 in EBCDIC binary notation

I = 1100 1001 in EBCDIC binary notation

T = 1110 0011 in EBCDIC binary notation

Hence, EBCDIC coding for the word BIT in binary notation will be

<u>11000010</u>	<u>11001001</u>	<u>11100011</u>
B	I	T

3 bytes will be required for this representation because each letter requires 1 byte (or 8 bits)

ASCII

- ASCII stands for **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.
- ASCII is of two types – ASCII-7 and ASCII-8
- ASCII-7 uses 7 bits to represent a symbol and can represent 128 (2^7) different characters
- ASCII-8 uses 8 bits to represent a symbol and can represent 256 (2^8) different characters
- First 128 characters in ASCII-7 and ASCII-8 are same

Coding of Alphabetic and Numeric Characters in ASCII

Character	ASCII-7 / ASCII-8		Hexadecimal Equivalent
	Zone	Digit	
0	0011	0000	30
1	0011	0001	31
2	0011	0010	32
3	0011	0011	33
4	0011	0100	34
5	0011	0101	35
6	0011	0110	36
7	0011	0111	37
8	0011	1000	38
9	0011	1001	39

Coding of Alphabetic and Numeric Characters in ASCII

Character	ASCII-7 / ASCII-8		Hexadecimal Equivalent
	Zone	Digit	
A	0100	0001	41
B	0100	0010	42
C	0100	0011	43
D	0100	0100	44
E	0100	0101	45
F	0100	0110	46
G	0100	0111	47
H	0100	1000	48
I	0100	1001	49
J	0100	1010	4A
K	0100	1011	4B
L	0100	1100	4C
M	0100	1101	4D

Coding of Alphabetic and Numeric Characters in ASCII

Character	ASCII-7 / ASCII-8		Hexadecimal Equivalent
	Zone	Digit	
N	0100	1110	4E
O	0100	1111	4F
P	0101	0000	50
Q	0101	0001	51
R	0101	0010	52
S	0101	0011	53
T	0101	0100	54
U	0101	0101	55
V	0101	0110	56
W	0101	0111	57
X	0101	1000	58
Y	0101	1001	59
Z	0101	1010	5A

ASCII Table

Left Digit(s)	Right Digit	ASCII									
		0	1	2	3	4	5	6	7	8	9
0		NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
1		LF	VT	FF	CR	SO	SI	DLE	DC1	DC2	DC3
2		DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
3		RS	US	□	!	“	#	\$	%	&	'
4		()	*	+	,	-	.	/	0	1
5		2	3	4	5	6	7	8	9	:	;
6		<	=	>	?	@	A	B	C	D	E
7		F	G	H	I	J	K	L	M	N	O
8		P	Q	R	S	T	U	V	W	X	Y
9		Z	[\]	^	_	`	a	b	c
10		d	e	f	g	h	i	j	k	l	m
11		n	o	p	q	r	s	t	u	v	w
12		x	y	z	{		}	~	DEL		

ASCII-7 Coding Scheme

Example

Write binary coding for the word BOY in ASCII-7. How many bytes are required for this representation?

Solution:

B = 1000010 in ASCII-7 binary notation

O = 1001111 in ASCII-7 binary notation

Y = 1011001 in ASCII-7 binary notation

Hence, binary coding for the word BOY in ASCII-7 will be

<u>1000010</u>	<u>1001111</u>	<u>1011001</u>
B	O	Y

Since each character in ASCII-7 requires one byte for its representation and there are 3 characters in the word BOY, 3 bytes will be required for this representation

ASCII-8 Coding Scheme

Example

Write binary coding for the word SKY in ASCII-8. How many bytes are required for this representation?

Solution:

S = 01010011 in ASCII-8 binary notation

K = 01001011 in ASCII-8 binary notation

Y = 01011001 in ASCII-8 binary notation

Hence, binary coding for the word SKY in ASCII-8 will be

<u>01010011</u>	<u>01001011</u>	<u>01011001</u>
S	K	Y

Since each character in ASCII-8 requires one byte for its representation and there are 3 characters in the word SKY, 3 bytes will be required for this representation

Unicode

- **Why Unicode:**
 - No single encoding system supports all languages
 - Different encoding systems conflict
- **Unicode features:**
 - Provides a consistent way of encoding multilingual plain text
 - Defines codes for characters used in all major languages of the world
 - Defines codes for special characters, mathematical symbols, technical symbols, and diacritics

Unicode

- **Unicode features (continued):**
 - Capacity to encode as many as a million characters
 - Assigns each character a unique numeric value and name
 - Reserves a part of the code space for private use
 - Affords simplicity and consistency of ASCII, even corresponding characters have same code
 - Specifies an algorithm for the presentation of text with bi-directional behavior
- **Encoding Forms**
 - UTF-8, UTF-16, UTF-32

Collating Sequence

- Collating sequence defines the assigned ordering among the characters used by a computer
- Collating sequence may vary, depending on the type of computer code used by a particular computer
- In most computers, collating sequences follow the following rules:
 1. Letters are considered in alphabetic order
($A < B < C \dots < Z$)
 2. Digits are considered in numeric order
($0 < 1 < 2 \dots < 9$)

Sorting in EBCDIC

Example

Suppose a computer uses EBCDIC as its internal representation of characters. In which order will this computer sort the strings 23, A1, 1A?

Solution:

In EBCDIC, numeric characters are treated to be greater than alphabetic characters. Hence, in the said computer, numeric characters will be placed after alphabetic characters and the given string will be treated as:

$$A1 < 1A < 23$$

Therefore, the sorted sequence will be: A1, 1A, 23.

Sorting in ASCII

Example

Suppose a computer uses ASCII for its internal representation of characters. In which order will this computer sort the strings 23, A1, 1A, a2, 2a, aA, and Aa?

Solution:

In ASCII, numeric characters are treated to be less than alphabetic characters. Hence, in the said computer, numeric characters will be placed before alphabetic characters and the given string will be treated as:

$$1A < 23 < 2a < A1 < Aa < a2 < aA$$

Therefore, the sorted sequence will be: 1A, 23, 2a, A1, Aa, a2, and aA

Boolean Algebra and Logic Circuits

Boolean Algebra

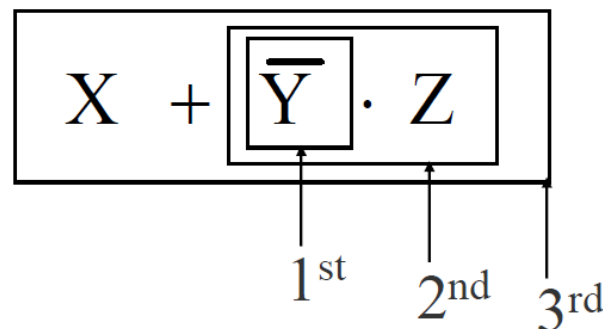
- An algebra that deals with binary number system
- George Boole (1815-1864), an English mathematician, developed it for:
 - Simplifying representation
 - Manipulation of propositional logic
- In 1938, Claude E. Shannon proposed using Boolean algebra in design of relay switching circuits
- Provides economical and straightforward approach
- Used extensively in designing electronic circuits used in computers

Fundamental concepts of Boolean Algebra

- Use of Binary Digit
 - Boolean equations can have either of two possible values, 0 and 1
- Logical Addition
 - Symbol '+', also known as '**OR**' operator, used for logical addition. Follows law of binary addition
- Logical Multiplication
 - Symbol '.', also known as '**AND**' operator, used for logical multiplication. Follows law of binary multiplication
- Complementation
 - Symbol '-', also known as '**NOT**' operator, used for complementation. Follows law of binary complement

Operator Precedence

- Each operator has a precedence level
- Higher the operator's precedence level, earlier it is evaluated
- Expression is scanned from left to right
- First, expressions enclosed within parentheses are evaluated
- Then, all complement (NOT) operations are performed
- Then, all '.' (AND) operations are performed
- Finally, all '+' (OR) operations are performed



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	$A(A') = 0$	$A + (A') = 1$
De Morgan's law	$(AB)' = A' \text{ OR } B'$	$(A + B)' = A'B'$

Boolean Functions

- A Boolean function is an expression formed with:
 - Binary variables
 - Operators (OR, AND, and NOT)
 - Parentheses, and equal sign
- The value of a Boolean function can be either 0 or 1
- A Boolean function may be represented as:
 - An algebraic expression, or
 - A truth table

Boolean Algebraic Expression

$$W = X + \bar{Y} \cdot Z$$

- Variable W is a function of X , Y , and Z , can also be written as $W = f(X, Y, Z)$
- The RHS of the equation is called an **expression**
- The symbols X , Y , Z are the **literals** of the function
- For a given Boolean function, there may be more than one algebraic expressions

Truth Table

X	Y	Z	W
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

- The number of rows in the table is equal to 2^n , where n is the number of literals in the function
- The combinations of 0s and 1s for rows of this table are obtained from the binary numbers by counting from 0 to $2^n - 1$

Gates and Circuits

Computers and Electricity

- **Gate**
 - A device that performs a basic operation of electrical signals
- **Circuits**
 - Gates combined to perform more complicated tasks

Computers and Electricity

How do we describe the behavior of 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

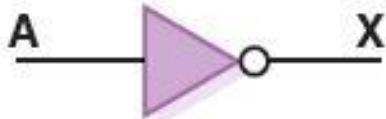
Gates

Six types of gates

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

NOT Gate

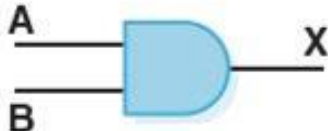
A NOT gate accepts one input signal (0 or 1) and returns the complementary (opposite) signal as output

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							

AND Gate

An AND gate accepts two input signals

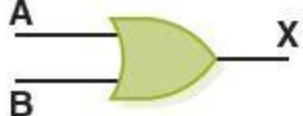
If both are 1, the output is 1; otherwise, the output is 0

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															

OR Gate

An OR gate accepts two input signals


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

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>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>1</td></tr></table>	A	B	X	0	0	0	0	1	1	1	0	1	1	1	1
A	B	X															
0	0	0															
0	1	1															
1	0	1															
1	1	1															

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															

XOR Gate

Note the difference between the **XOR** gate and the **OR** gate; they differ only in one input situation

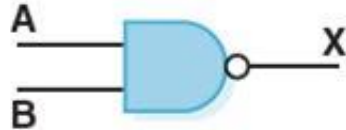
When both input signals are 1, the OR gate produces a 1 and the XOR produces a 0

XOR is called the *exclusive OR* because its output is 1 if (and only if):

- *either* one input *or* the other is 1,
- *excluding* the case that they both are

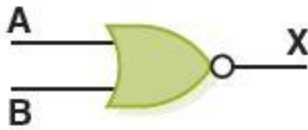
NAND Gate

The NAND (“NOT of AND”) 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															

NOR Gate

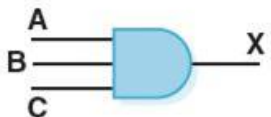
The NOR (“NOT of OR”) gate accepts two inputs. If both are 0, the output is 1; otherwise, the output is 0.

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															

Gates with More Inputs

Some gates can be generalized 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																																			

Review of Gate Processing

Gate	Behavior
NOT	Inverts its single input
AND	Produces 1 if all input values are 1
OR	Produces 0 if all input values are 0
XOR	Produces 0 if both input values are the same
NAND	Produces 0 if all input values are 1
NOR	Produces 1 if all input values are 0

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

Transistors are used to build logical gates



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

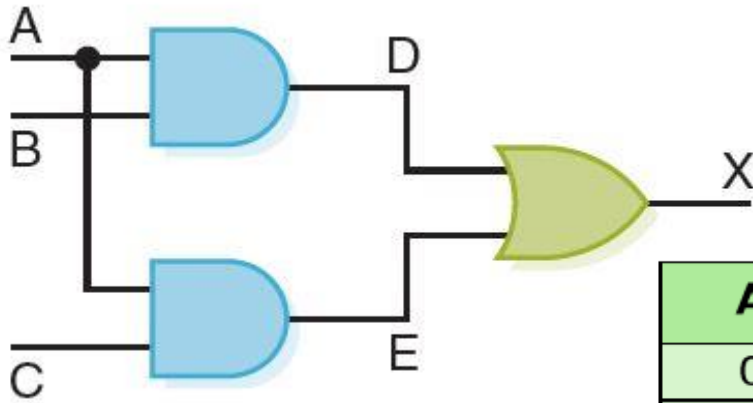
- Boolean expressions

- 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	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

Adders

At the digital logic level, addition is performed in binary

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

Adders

The result of adding two binary digits could produce a *carry value*

Recall that $1 + 1 = 10$
in base two

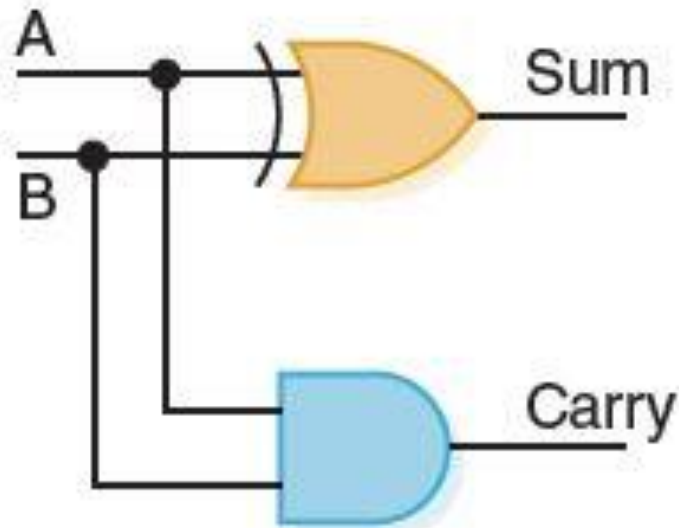
Half adder

A circuit that computes the sum of two bits and produces the correct carry bit

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth table

Adders



Circuit diagram
representing
a half adder

Boolean expressions

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

$$\text{carry} = AB$$

Adders

Full adder

A circuit that takes the carry-in value into account

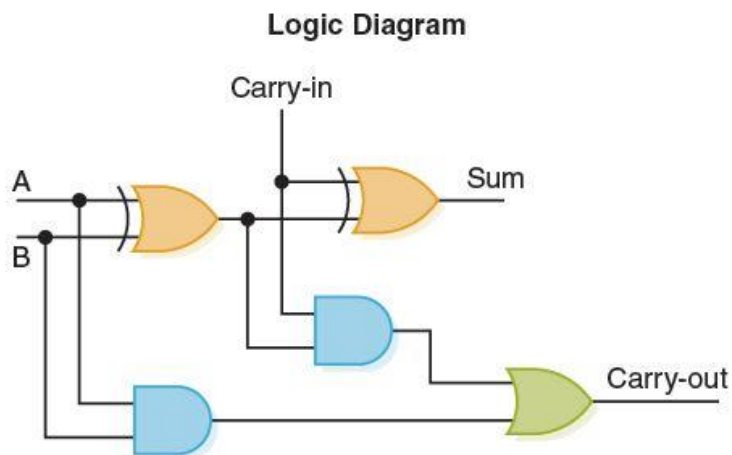


FIGURE 4.10 A full adder

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