# Lecture 1 Outline

Reminders to self:

- ❑ Turn on lecture recording to Cloud
- ❑ Turn on Zoom microphone

- Last Lecture
  - Finished Binary Arithmetic
  - Representation of negative numbers in binary
  - Addition with 2's complement binary encoding
- Today's Lecture
  - Addition with 1's complement binary encoding
  - Binary codes
  - Start Boolean algebra

# Handouts and Announcements

- Announcements
  - Homework Problems 2-1
    - Already on Carmen – available at end of lecture
    - Due in Carmen 11:59pm, Thursday 1/26
      - HW 1-6: 11:25am, Monday 1/23
      - HW 1-7, 1-8: 11:25am Wednesday 1/25
  - Homework Problem 1-6, 1-7 and 1-8 reminder
    - HW 1-6 due: 11:25am, Monday 1/23
    - HW 1-7, 1-8 due: 11:25am Wednesday 1/25
  - Read for Monday: Pages 46-53, 66-70, 87, 94-97

# Handouts and Announcements

- Announcements
  - Mini-Exam 1 Reminder
    - Available 5pm Monday 1/23 through 5:00pm Tuesday 1/24
    - Due in Carmen PROMPTLY at 5:00pm on 1/24
    - Designed to be completed in ~36 min, but you may use more
    - When planning your schedule:
      - I recommend building in 10-15 min extra
      - To allow for downloading exam, signing and dating honor pledge, saving solution as pdf, and uploading to Carmen
    - I also recommend not procrastinating
  - Exam review topics available on Carmen

# Representation of Negative Numbers

## 1's compliment:

$$\bar{N} = (2^n - 1) - N$$

- Ex: $-5_{10} = (2^4 - 1) - 5 = 16 - 1 - 5 = 10_{10}$ $= 1010_2$
- Alternately, 1's complement can be found via bit-by-bit complement $+5_{10} = 0101_2 \rightarrow -5 = 1010$ 1's comp
- **End around carry:** In one's compliment addition
  - The last carry is not discarded as it is in 2's compliment
  - Rather, added to the $n$-bit sum in the position furthest to the right

# Example: 1's compliment Addition

- Addition of two positive numbers is identical to 2's compliment
- Not repeated here

**3.** Addition of positive and negative numbers (negative number with greater magnitude)

| | | |
|---|---|---|
| +5 | 0101 | |
| −6 | 1001 | |
| −1 | 1110 | (correct answer) |

**4.** Same as case 3 except positive number has greater magnitude

| | | | |
|---|---|---|---|
| −5 | | 1010 | |
| +6 | | 0110 | |
| | (1) | 0000 | |
| | └──→ | 1 | (end-around carry) |
| | | 0001 | (correct answer, *no* overflow) |

# Example 10: 1's compliment Addition (continued)

**5.** Addition of two negative numbers, $|\text{sum}| < 2^{n-1}$

$$\begin{array}{rrrl}
-3 & & 1100 & \\
\underline{-4} & & \underline{1011} & \\
 & (1) & 0111 & \\
 & \hookrightarrow & 1 & \text{(end-around carry)} \\
 & & \overline{1000} & \text{(correct answer, } \textit{no} \text{ overflow)}
\end{array}$$

**6.** Addition of two negative numbers, $|\text{sum}| \geq 2^{n-1}$

$$\begin{array}{rrrl}
-5 & & 1010 & \\
\underline{-6} & & \underline{1001} & \\
 & (1) & 0011 & \\
 & \hookrightarrow & 1 & \text{(end-around carry)} \\
 & & \overline{0100} & \text{(wrong answer because of overflow)}
\end{array}$$

# Example: 1's compliment Addition (continued)

## $n = 8$ example

**1.** Add −11 and −20 in 1's complement.

$+11 = 00001011 \qquad +20 = 00010100$

taking the bit-by-bit complement,

−11 is represented by 11110100 and −20 by 11101011

$$
\begin{array}{rr}
11110100 & (-11) \\
\underline{11101011} & \underline{+(-20)} \\
(1)\ 11011111 & \\
\underline{\longrightarrow 1} & \text{(end-around carry)} \\
11100000 = -31 &
\end{array}
$$

# Binary codes

- Although most large computers work internally with binary numbers, the input-output equipment generally uses decimal numbers.
- Because most logic circuits only accept two-valued signals, the decimal numbers must be coded in terms of binary signals.
- In the simplest form of binary code, each decimal digit is replaced by its binary equivalent. For example, 937.25 is represented by:

Binary Coded Decimal (BCD)

$$\begin{array}{cccccc} 9 & 3 & 7 & . & 2 & 5 \\ 1001 & 0011 & 0111 & . & 0010 & 0101 \end{array}$$

# Binary codes

Bit weights

| Decimal Digit | 8-4-2-1 Code (BCD) | 6-3-1-1 Code | Excess-3 Code | 2-out-of-5 Code | Gray Code |
|---|---|---|---|---|---|
| 0 | 0000 | 0000 | 0011 | 00011 | 0000 |
| 1 | 0001 | 0001 | 0100 | 00101 | 0001 |
| 2 | 0010 | 0011 | 0101 | 00110 | 0011 |
| 3 | 0011 | 0100 | 0110 | 01001 | 0010 |
| 4 | 0100 | 0101 | 0111 | 01010 | 0110 |
| 5 | 0101 | 0111 | 1000 | 01100 | 1110 |
| 6 | 0110 | 1000 | 1001 | 10001 | 1010 |
| 7 | 0111 | 1001 | 1010 | 10010 | 1011 |
| 8 | 1000 | 1011 | 1011 | 10100 | 1001 |
| 9 | 1001 | 1100 | 1100 | 11000 | 1000 |

Only one bit flips at each digit

Good for input from electro-mechanical switches

look at $3 \to 4$ in BCD
Rotary Encoder

$8-4-2-1+11_2$
Aka XS-3

2 of 5 bits are 1
Error Checking

Historically: mechanical adding machines, cash registers, and early computers and electronic calculators

# Binary codes

Codes starting 000 & 001 are for control functions: e.g.

"end of transmission" (0000100)

or

"file separator" (0011100)

**TABLE 1-3** ASCII Code

| | ASCII Code | | | | | | | | ASCII Code | | | | | | | | ASCII Code | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Character | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ | Character | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ | Character | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ |
| space | 0 | 1 | 0 | 0 | 0 | 0 | 0 | @ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | ' | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| ! | 0 | 1 | 0 | 0 | 0 | 0 | 1 | A | 1 | 0 | 0 | 0 | 0 | 0 | 1 | a | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| " | 0 | 1 | 0 | 0 | 0 | 1 | 0 | B | 1 | 0 | 0 | 0 | 0 | 1 | 0 | b | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| # | 0 | 1 | 0 | 0 | 0 | 1 | 1 | C | 1 | 0 | 0 | 0 | 0 | 1 | 1 | c | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| $ | 0 | 1 | 0 | 0 | 1 | 0 | 0 | D | 1 | 0 | 0 | 0 | 1 | 0 | 0 | d | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| % | 0 | 1 | 0 | 0 | 1 | 0 | 1 | E | 1 | 0 | 0 | 0 | 1 | 0 | 1 | e | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| & | 0 | 1 | 0 | 0 | 1 | 1 | 0 | F | 1 | 0 | 0 | 0 | 1 | 1 | 0 | f | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| ' | 0 | 1 | 0 | 0 | 1 | 1 | 1 | G | 1 | 0 | 0 | 0 | 1 | 1 | 1 | g | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| ( | 0 | 1 | 0 | 1 | 0 | 0 | 0 | H | 1 | 0 | 0 | 1 | 0 | 0 | 0 | h | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| ) | 0 | 1 | 0 | 1 | 0 | 0 | 1 | I | 1 | 0 | 0 | 1 | 0 | 0 | 1 | i | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| * | 0 | 1 | 0 | 1 | 0 | 1 | 0 | J | 1 | 0 | 0 | 1 | 0 | 1 | 0 | j | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| + | 0 | 1 | 0 | 1 | 0 | 1 | 1 | K | 1 | 0 | 0 | 1 | 0 | 1 | 1 | k | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| , | 0 | 1 | 0 | 1 | 1 | 0 | 0 | L | 1 | 0 | 0 | 1 | 1 | 0 | 0 | l | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| – | 0 | 1 | 0 | 1 | 1 | 0 | 1 | M | 1 | 0 | 0 | 1 | 1 | 0 | 1 | m | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| . | 0 | 1 | 0 | 1 | 1 | 1 | 0 | N | 1 | 0 | 0 | 1 | 1 | 1 | 0 | n | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| / | 0 | 1 | 0 | 1 | 1 | 1 | 1 | O | 1 | 0 | 0 | 1 | 1 | 1 | 1 | o | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | P | 1 | 0 | 1 | 0 | 0 | 0 | 0 | p | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | Q | 1 | 0 | 1 | 0 | 0 | 0 | 1 | q | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | R | 1 | 0 | 1 | 0 | 0 | 1 | 0 | r | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | S | 1 | 0 | 1 | 0 | 0 | 1 | 1 | s | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | T | 1 | 0 | 1 | 0 | 1 | 0 | 0 | t | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | U | 1 | 0 | 1 | 0 | 1 | 0 | 1 | u | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | V | 1 | 0 | 1 | 0 | 1 | 1 | 0 | v | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | W | 1 | 0 | 1 | 0 | 1 | 1 | 1 | w | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 8 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | X | 1 | 0 | 1 | 1 | 0 | 0 | 0 | x | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 9 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | Y | 1 | 0 | 1 | 1 | 0 | 0 | 1 | y | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| : | 0 | 1 | 1 | 1 | 0 | 1 | 0 | Z | 1 | 0 | 1 | 1 | 0 | 1 | 0 | z | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| ; | 0 | 1 | 1 | 1 | 0 | 1 | 1 | [ | 1 | 0 | 1 | 1 | 0 | 1 | 1 | { | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| < | 0 | 1 | 1 | 1 | 1 | 0 | 0 | \ | 1 | 0 | 1 | 1 | 1 | 0 | 0 | \| | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| = | 0 | 1 | 1 | 1 | 1 | 0 | 1 | ] | 1 | 0 | 1 | 1 | 1 | 0 | 1 | } | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| > | 0 | 1 | 1 | 1 | 1 | 1 | 0 | ^ | 1 | 0 | 1 | 1 | 1 | 1 | 0 | ~ | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| ? | 0 | 1 | 1 | 1 | 1 | 1 | 1 | — | 1 | 0 | 1 | 1 | 1 | 1 | 1 | delete | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

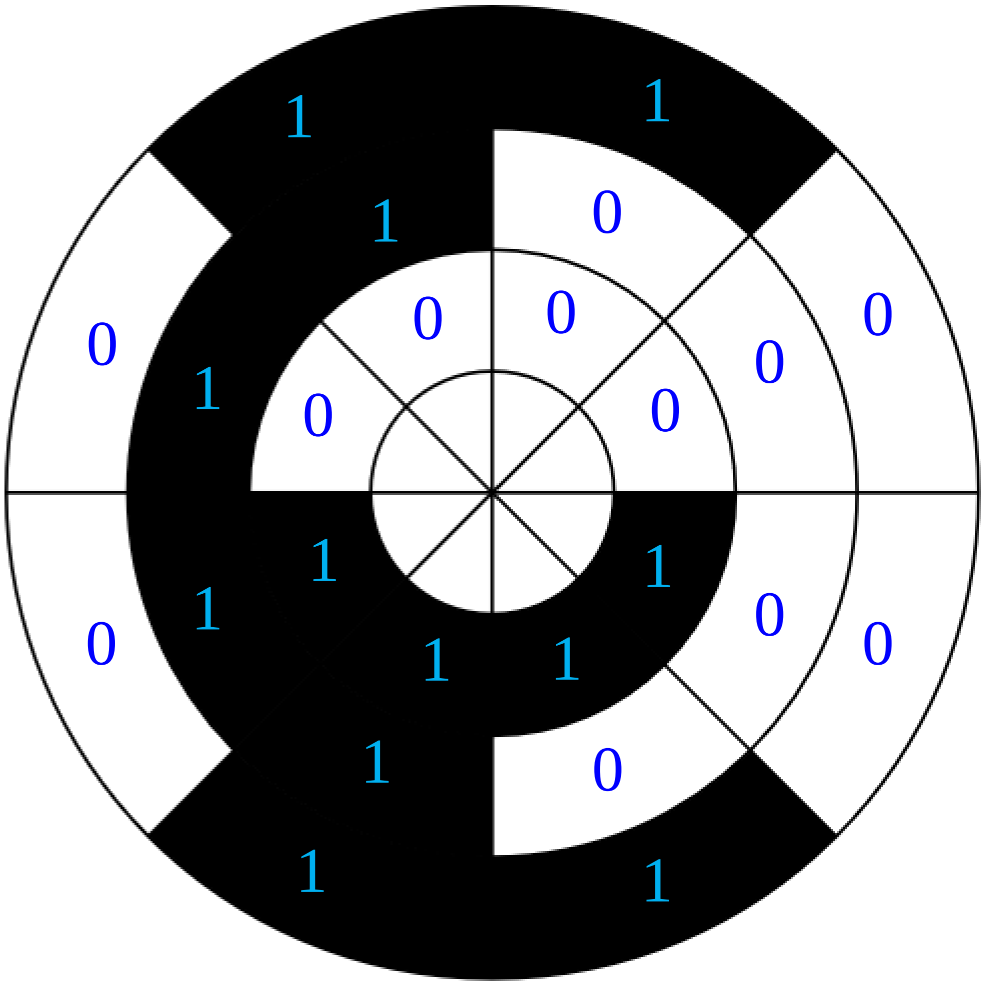


Modern extensions have many more codes

e.g. Unicode 12.1 standard has 137,994 characters

- Modern scripts,
- Historic scripts,
- Emoji, etc.

# Binary codes




## 3-bit Rotary Encoder

- 8 positions
- 3-bit Gray code
- Often done optically
- Let light beam through or block light
- Three source-detector pairs

By jjbeard - Replacement for Image:Encoder_disc.png, Public Domain, https://commons.wikimedia.org/w/index.php?curid=801435

# Switching to Chapter 2 now: Boolean Algebra

## Learning Objectives

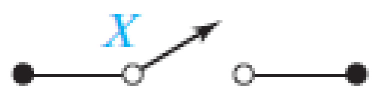
- Understand basic operations and laws of Boolean algebra
- Relate operations and laws to circuits composed of AND gates, OR gates, INVERTERS and switches
- Prove laws in switching algebra using a truth table
- Apply laws to manipulation of algebraic expressions including:
  - obtaining a sum of products or product of sums,
  - simplifying an expression, and/or
  - finding the complement of an expression

# Introduction

- All switching devices we will use are two-state devices, so we will emphasize the case in which all variables assume only one of two values
- Boolean variables, such as $X$ or $Y$, will be used to represent input or output of switching circuit
- Symbols "0" and "1" represent the two values any variable can take on
- These represent states in a logic circuit, and do not have numeric value.
- Logic gate:
  - 0 usually represents range of low voltages, and
  - 1 represents range of high voltages
- Switch circuit:
  - 0 represents open switch, and
  - 1 represents closed
- 0 and 1 can be used to represent the two states in any binary valued system.

# Boolean Algebra – Basic Operations

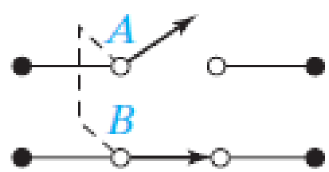
- The basic operations of Boolean ( Switching ) algebra are called
  - AND,
  - OR, and
  - complement (or inverse )
- To apply switching algebra to a switch circuit, each switch contact is labeled with a variable




$X = 0 \rightarrow$ switch open as drawn
$X = 1 \rightarrow$ switch closed Connection made

- NC (normally closed) and NO (normally open) contacts are always in opposite states.

Dashed line indicates "ganged" operation



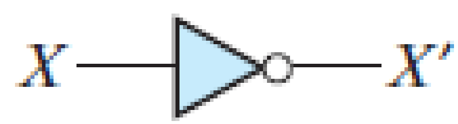

NO contact
NC contact

- If variable $X$ is assigned to NO contact, then $X'$ will be assigned for NC (the prime denotes complementation)

# Boolean Algebra – Basic Operations

## Complementation / Inversion:

- Prime ($'$) denotes complementation
  - $0' = 1$ and $1' = 0$
- For a switching variable, $X$:
  - $X' = 1$ if $X = 0$, and
  - $X' = 0$ if $X = 1$
- Complementation is also called inversion
- An inverter (gate implementing inversion) is represented as shown here, where circle at output denotes inversion




- Triangle symbol without the circle would be a buffer
  - No logic function: Input = $X$, Output = $X$
  - Might do things like, provide propagation delay , boost output current, etc.

# Boolean Algebra – Basic Operations

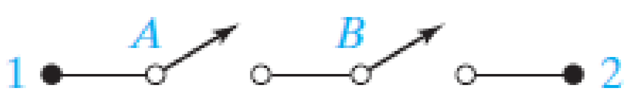
## Series Switching Circuits / AND Operation:

Truth Table

| $A$ | $B$ | $C = A \cdot B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

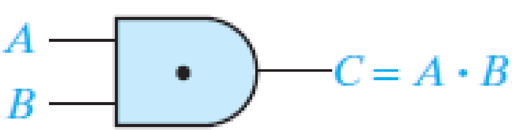
- Operation defined by this truth table is called AND
- Written algebraically as $C = A \cdot B$
- We will usually write $AB$ instead of $A \cdot B$
- AND operation also referred to as logical (or Boolean) multiplication

Switch Circuit Diagram




Either switch open
$C = 0 \rightarrow$ open circuit between terminals 1 and 2
$C = 1 \rightarrow$ closed circuit between terminals 1 and 2
Both switches closed

Logic Gate Diagram



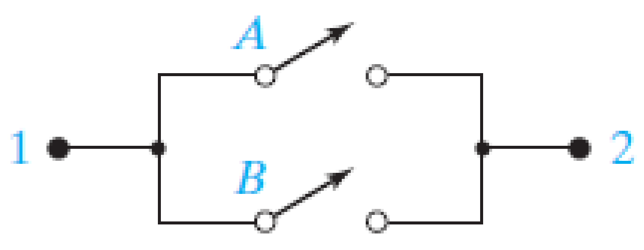

Note: the dot is often (actually usually) not shown.
Shape identifies function.
(IEEE Std 91/91a-1991 does not include the dot)

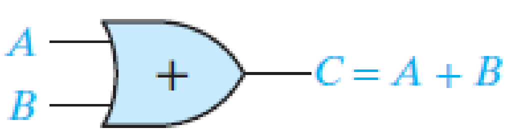
# Boolean Algebra – Basic Operations

## Parallel Switching Circuits / Operation:

| $A$ | $B$ | $C = A + B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

- Operation defined by this truth table is called
- Written algebraically as $C = A + B$
- OR operation also referred to as logical (or Boolean) addition



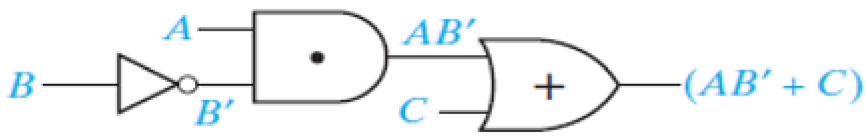




Note: the plus sign is often (actually usually) not shown. Shape identifies function.
(IEEE Std 91/91a-1991 does not include the plus)

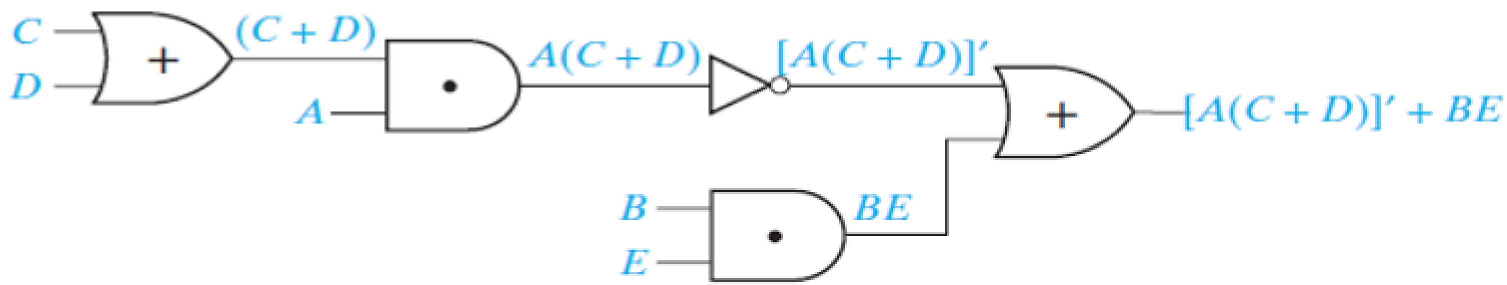
# Boolean Operations and Truth Tables

## Examples: Boolean Expressions & Corresponding Diagrams

- Expressions
  - $AB' + C$




  - $[A(C + D)]' + BE$




- Order of operations:
  1.
  2.
  3.
  4.

For the second expression,
if $A = B = D = 1$ and $C = E = 0$ then
$[A(C + D)]' + BE =$

# Boolean Operations and Truth Tables

- Expression $AB' + C$

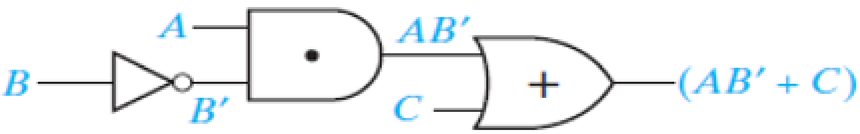



TABLE 2-1



Discuss order of filling input columns

| A B C | $B'$ | $AB'$ | $AB' + C$ | $A + C$ | $B' + C$ | $(A + C)(B' + C)$ |
|---|---|---|---|---|---|---|
| 0 0 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 0 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 1 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 1 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 0 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 0 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 1 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 1 1 | 0 | 0 | 1 | 1 | 1 | 1 |

## Equal Boolean Expressions:

Two Boolean expressions are said to be equal if they have the same value for every possible combination of the variables

# Boolean Algebra – Basic Operations

## A bit more about Complementation / Inversion:

- Also known at the operation
- Our textbook uses the prime mark to indicate inversion
  - $X' = 1$ if $X = 0$, and
  - $X' = 0$ if $X = 1$
- It is very common to see an overbar mark used for inversion
  - $\bar{X} = 1$ if $X = 0$, and
  - $\bar{X} = 0$ if $X = 1$
- Looking at the same two expressions from a few slides ago:
  - $AB' + C \quad \Leftrightarrow \quad A\bar{B} + C$
  - $[A(C + D)]' + BE \quad \Leftrightarrow \quad \overline{A(C + D)} + BE$

# Crossovers vs. Connections

Wires in circuit schematics: 1) Sometime branch. 2) Sometimes they cross without connecting

| | Connected | Not Connected |
|---|---|---|
| Preferred | | |
| Accepted | | |
| But see sometimes | | |
| Archaic | | |