Lecture 1 Outline

Reminders to self:

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

- Last Lecture
  - Addition with 1's complement binary encoding
  - Binary codes
  - Started Boolean algebra – through AND operation
- Today's Lecture
  - Continue Boolean algebra – resume with OR operation

Handouts and Announcements

- Announcements
  - Homework Problems 2-2
    - Posted on Carmen yesterday (1/22)
    - Due in Carmen 11:25am, Monday 1/30
  - Homework Problem 1-7, 1-8, and 2-1 reminder
    - HW 1-7, 1-8 due: 11:25am Wednesday 1/25
    - HW 2-1 due: 11:59pm Thursday 1/26
  - Read for Wednesday: Pages 97-107

Handouts and Announcements

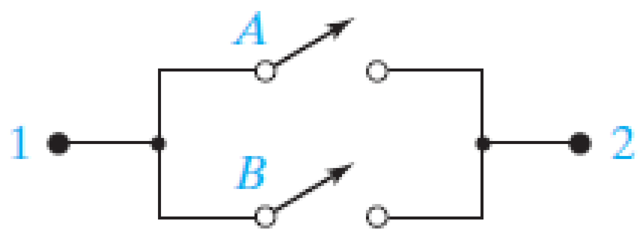
- Announcements
  - Mini-Exam 1 – **Final** Reminder
    - Available 5pm Monday 1/23 through 5:00pm Tuesday 1/24
    - Due in Carmen PROMPTLY at 5:00pm on 1/24
    - Designed to be completed in ~36 min, but you may use more
    - When planning your schedule:
      - I recommend building in 10-15 min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

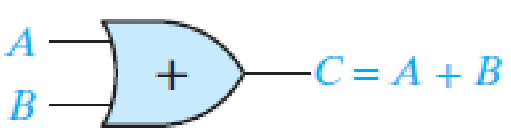
Boolean Algebra – Basic Operations

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



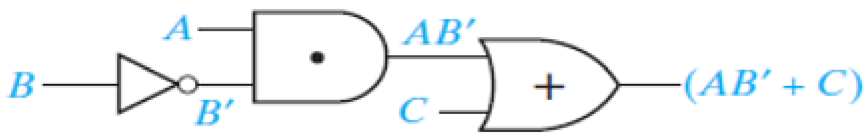


Note: the plus sign is often (actually usually) not shown. Shape identifies function.
(IEEE Std 91/91a-1991 does not include the plus)

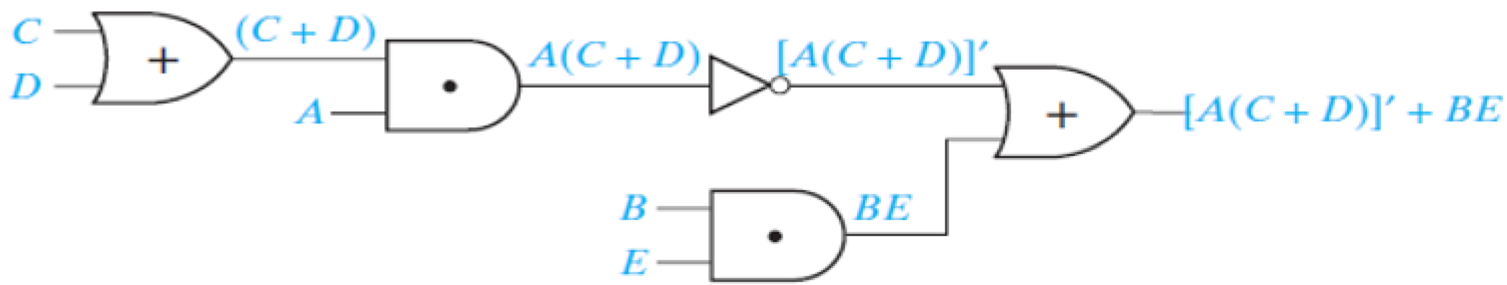
Boolean Operations and Truth Tables

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 $AB' + C$

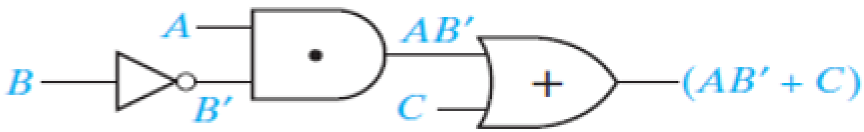


TABLE 2-1

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

Discuss order of filling input columns

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		

Basic Theorems

Single Variable Basic Theorems:

$$X + 0 = X \qquad\qquad X \cdot 1 = X$$

$$X + 1 = 1 \qquad\qquad X \cdot 0 = 0$$

denoting an element of a set which is unchanged in value when multiplied or otherwise operated on by itself

$$X + X = X \qquad\qquad X \cdot X = X$$

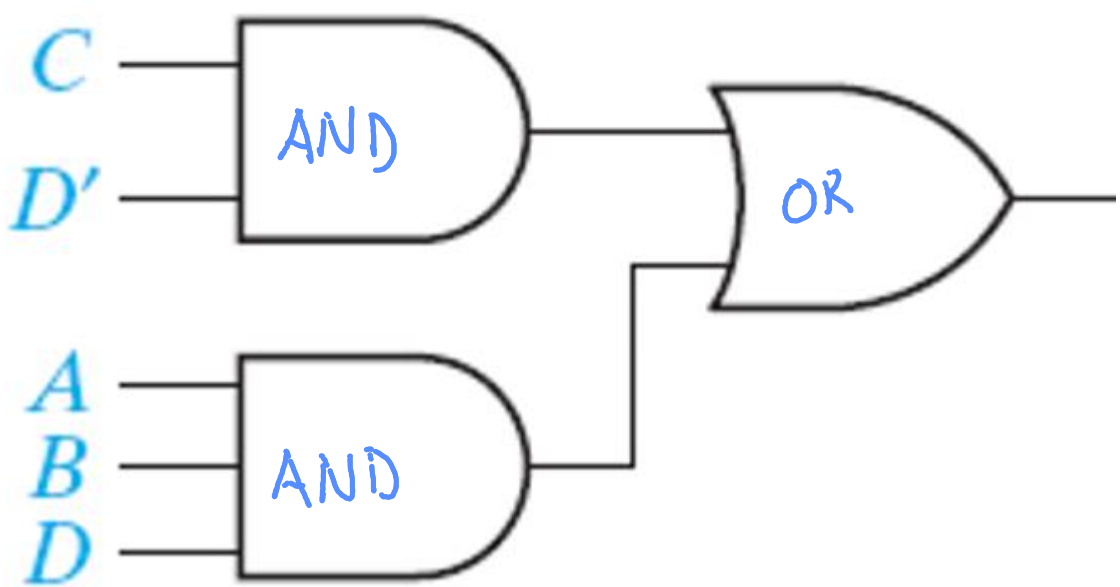
$$(X')' = X$$

$$X + X' = 1 \qquad\qquad X \cdot X' = 0$$

Boolean Algebra – Literals

Each appearance of a variable or its complement is called a "Literal"

- $C\overline{D} + ABD$
- Four variables
- Five literals



Boolean Algebra – Other Logic Gates

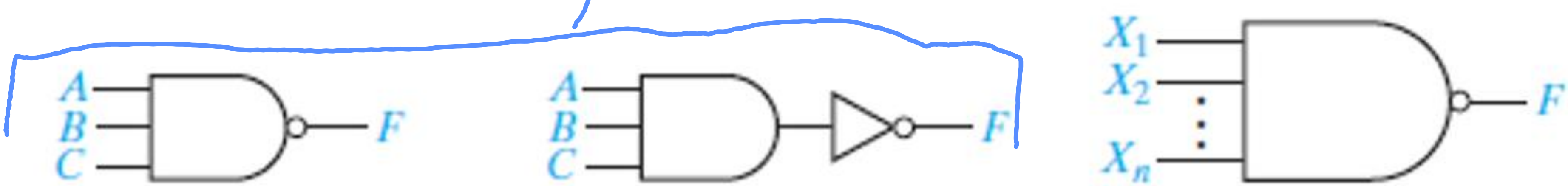
NAND: "NOT AND" (AND → NOT)

- $Z = \overline{AB}$ (A NAND B)

$F = \overline{ABC}$

- At transistor level
  - Require fewer components
  - Faster

A	B	$Z = \overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0



(a) Three-input NAND gate (b) NAND gate equivalent (c) *n*-input NAND gate

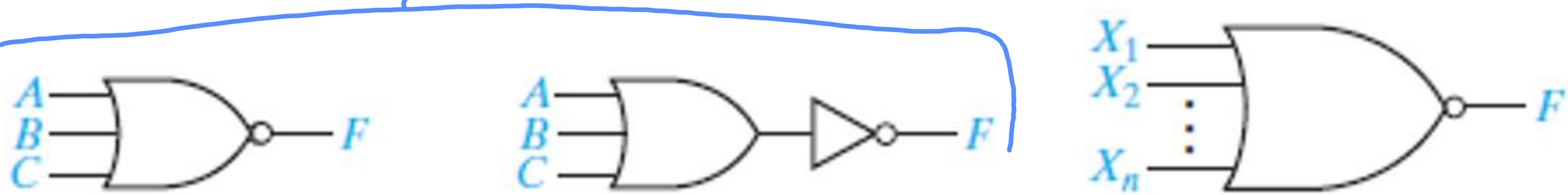
Boolean Algebra – Other Logic Gates

NOR: "NOT OR" (OR → NOT)

- $Z = \overline{A + B}$ (A NOR B)

A	B	$Z = \overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

- $F = \overline{A + B + C}$

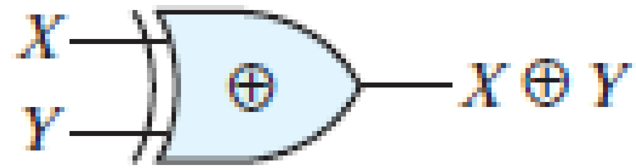


(a) Three-input NOR gate (b) NOR gate equivalent (c) n-input NOR gate

Boolean Algebra – Other Logic Gates

XOR: "eXclusive OR"

- Operator symbol is $\oplus$
- Logic gate symbol for XOR $\Longrightarrow X \oplus Y$



- $Z = A \oplus B \Rightarrow \quad Z = 1$ if $A = 1$ or $B = 1$, but not if both $= 1$
- Truth table

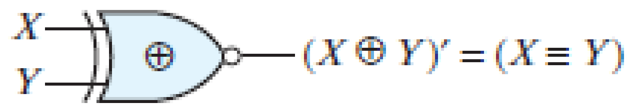
A	B	$Z = X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

- $Z = A \oplus B = A\bar{B} + \bar{A}B$

Boolean Algebra – Other Logic Gates

XNOR: "eXclusive NOR"

- Complement of XOR $\Longrightarrow \overline{X \oplus Y}$
- Logic gate symbol for XNOR $\Longrightarrow \overline{X \oplus Y}$



- $Z = \overline{A \oplus B} \quad \Rightarrow \quad Z = 1$ iff $A = B$ $\therefore$ also known as Equivalence
- Truth table

A	B	$Z = \overline{X \oplus Y}$
0	0	1
0	1	0
1	0	0
1	1	1

- $Z = \overline{A \oplus B} = AB + \bar{A}\bar{B}$

Boolean Algebra – Basic Laws

Commutative & Associative Laws:

- Many basic laws of ordinary algebra also apply to Boolean algebra
- Commutative:
  - Order variables written does not affect result of AND & OR operations
  - $XY = YX$
  - $X + Y = Y + X$
- Associative:
  - Result of AND & OR operations independent of which variables we associate together first
  - $(XY)Z = X(YZ) = XYZ$
  - $(X + Y) + Z = X + (Y + Z) = X + Y + Z$
  - XOR:
    - $A \oplus B \oplus C = (A \oplus B) \oplus C = A \oplus (B \oplus C)$
    - Output = 1 for odd # of 1s

Boolean Algebra – Basic Laws

Develop $A \oplus B \oplus C = (A \oplus B) \oplus C = A \oplus (B \oplus C)$ truth table

A	B	C	$A\oplus B$	$(A\oplus B)\oplus C$	$B\oplus C$	$A\oplus(B\oplus C)$
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	1	1
0	1	1	1	0	0	0
1	0	0	1	1	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	0
1	1	1	0	1	0	1

Boolean Algebra – Basic Laws

Distributive Law of Boolean Algebra:

- Ordinary Distributive Law:
  - $X(Y + Z) = XY + XZ$
- Second Distributive Law (not valid for ordinary algebra)
  - $X + YZ = (X + Y)(X + Z)$
  - This is the "   " of the first distributive law
- "   " concept satisfied in Boolean algebra
  - Given a Boolean algebra expression
    - Interchange all constants 1 and 0
    - Interchange AND and OR operations
    - Variables and complements unchanged
- A more direct algebraic proof of the second distributive law is shown on page 45 of the textbook

Boolean Algebra – DeMorgan's Laws

Duality Examples:

- Interchange all constants 1 and 0
- Interchange AND and OR operations
- Variables and complements unchanged
- Examples:
  - $F = X + X' = 1 \;\Rightarrow\; \text{DUAL}(F) \longrightarrow XX' = 0$
  - $G = X + X = X \;\Rightarrow\; \text{DUAL}(G) \longrightarrow XX = X$
  - $H = X + 0 = X \Rightarrow\; \text{DUAL}(H) \longrightarrow X \cdot 1 = X$
  - $K = X + 1 = 1 \Rightarrow\; \text{DUAL}(K) \longrightarrow X \cdot 0 = 0$
  - $L = X + Y = Y + X \Rightarrow\; \text{DUAL}(L) \longrightarrow XY = YX$
  - OR & AND forms of Associative Law also related by duality
  - Distributive Laws by duality shown on previous slide

Boolean Algebra – DeMorgan's Law

DeMorgan's Law:

- The NOT operation isn't distributable by normal means
- Special rule $\Longrightarrow$ DeMorgan's Law to find the complement
  1. Take the DUAL
  2. Complement each literal
- First form of DeMorgan's Law: $\overline{X+Y} = \bar{X}\bar{Y}$
- Second form of DeMorgan's Law: $\overline{XY} = \bar{X} + \bar{Y}$
- Note: Two forms are duals of each other
- Truth table proof of DeMorgan's Laws:

X	Y	X'	Y'	$X+Y$	$(X+Y)'$	$X'Y'$	XY	$(XY)'$	$X'+Y'$
0	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1
1	1	0	0	1	0	0	1	0	0

Boolean Algebra – DeMorgan's Law

DeMorgan's Examples:

$F = (A + \bar{B})(C + D)$ Find $\bar{F}$

$\bar{G} = A\bar{C}D + \bar{B}C$ Find G

Boolean Algebra – Completeness

Functional Completeness:

- A set of logic operations is said to be functionally complete if any Boolean function can be expressed in terms of this set of operations
- The set {NOT, AND, OR} is functionally complete
- Similarly, the set {NOT, NAND, NOR} is functionally complete
- But OR can be realized using NOT & AND, etc.
- Can implement any Boolean expression with just
  - {NOT, AND}, or
  - {NOT, OR}, or
  - {NOT, NAND}, or
  - {NOT, NOR}
- But a NAND gate (or NOR gate) with inputs tied together is a NOT
- Any Boolean expression can be implemented with just NAND gates (or just NOR gates)

Boolean Algebra – Simplification

- Boolean algebra laws and theorems can be used to algebraically simplify expressions into forms that are more readily implemented with logic gates
- Theorems and algebraic techniques useful for simplification and proving validity are covered in greater depth in some sections of Chapters 2 and 3
- But we are going to learn a graphical technique for reducing Boolean Expressions
  - Karnaugh Maps
  - Often abbreviated K-Maps