Lecture 1 Outline

Reminders to self:

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

- Last Lecture
  - – Continued Boolean algebra – through XOR operation
- Today's Lecture
  - – Continue Boolean algebra – resume at Distributive Law
  - – Foundations for K-Maps
    - Minterms
    - Maxterms

Handouts and Announcements

- Announcements
  - Homework Problems 2-3
    - Posted on Carmen yesterday (1/24)
    - Due in Carmen 11:59pm, Tuesday 1/31
  - Homework Problems 2-1 and 2-2 reminder
    - HW 2-1 due: 11:59pm Thursday 1/26
    - HW 2-2 due: 11:25am Monday 1/30
  - Participation Quizzes 1&2
    - Quiz 1 available 11:10am today, Quiz 2 12:25pm today
    - Due 24 hrs later, but available +24hrs with late penalty
    - 15min time limit – clock starts when you start
  - Read for Friday: Pages 123, 134-143

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)
  - $X + YZ = (X+Y)(X+Z)$ $= XX + XZ + XY + YZ = X + XZ + XY + YZ$
  - This is the "dual" of the first distributive law $= X(1 + Z + Y) + YZ$
- "Duality" concept satisfied in Boolean algebra $= X + YZ$
  - 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:
  - $F = X + X' = 1 \;\Rightarrow\; \text{DUAL}(F) \longrightarrow XX' = 0$ (Complementarity Laws)
  - $G = X + X = X \;\Rightarrow\; \text{DUAL}(G) \longrightarrow XX = X$ (Idempotent Laws)
  - $H = X + 0 = X \Rightarrow\; \text{DUAL}(H) \longrightarrow X \cdot 1 = X$ (Operations with 0 and 1)
  - $K = X + 1 = 1 \Rightarrow\; \text{DUAL}(K) \longrightarrow X \cdot 0 = 0$ (Operations with 0 and 1)
  - $L = X + Y = Y + X \Rightarrow\; \text{DUAL}(L) \longrightarrow XY = YX$ (Commutative Law)
  - 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:

$\cancel{F = \overline{AB} \neq \bar{A}\bar{B}}$

$\cancel{F = \overline{A + B} \neq \bar{A} + \bar{B}}$

- 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'
0	0	1	1
0	1	1	0
1	0	0	1
1	1	0	0

Boolean Algebra – DeMorgan's Law

DeMorgan's Examples:

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

$$\overline{F} = \overline{(A+\overline{B})(C+D)} = \overline{(A+\overline{B})} + \overline{(C+D)} = \overline{A}B + \overline{C}\,\overline{D}$$

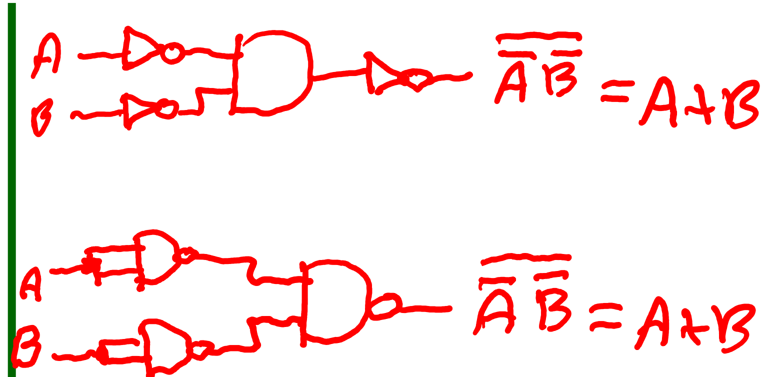
$\bar{G} = A\bar{C}D + \bar{B}C$ Find G

$$G = \overline{\overline{G}} = \overline{A\overline{C}D + \overline{B}C} = \overline{(A\overline{C}D)}\;\overline{(\overline{B}C)} = (\overline{A}+C+\overline{D})(B+\overline{C})$$

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. (use DeMorgan's - **Sketch**)
- 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) **(Sketch OR using 2-input NAND)**



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

Main steps

1. Find switching function that describes desired behavior
2. Find simplified Boolean algebraic expression
3. Realize simplified expression using available logic elements

Finding the switching function

- Logic design problems often stated in terms of one or more English sentences
- First step in designing logic circuit is to translate sentences into Boolean equations
  - Break down each sentence into phrases, and
  - Associate a Boolean variable with each phrase
- If a phrase can have a value of true or false, then we can represent that phrase by a Boolean variable
- Phrases can be
  - either true or false, or
  - have no truth value

Finding the switching function

Example

- Statement: *The alarm will ring iff the alarm switch is turned on and the door is not closed, or it is after 6 p.m. and the window is not closed.*
- Break into the following phrases with Boolean variables A, B, C, D and Z:

$$\underbrace{\text{The alarm will ring}}_{Z} \quad \text{iff} \quad \underbrace{\text{the alarm switch is on}}_{A} \quad \text{and}$$

$$\underbrace{\text{the door is not closed}}_{B'} \quad \textit{or} \quad \underbrace{\text{it is after 6 P.M.}}_{C} \quad \text{and}$$

$$\underbrace{\text{the window is not closed.}}_{D'}$$

$A = 1$ if alarm switch is on
$B = 1$ if door is closed ($B' = 1$ if door NOT closed)
$C = 1$ if after 6pm
$D = 1$ if window is closed ($D' = 1$ if window NOT closed)

To the Logic Circuit

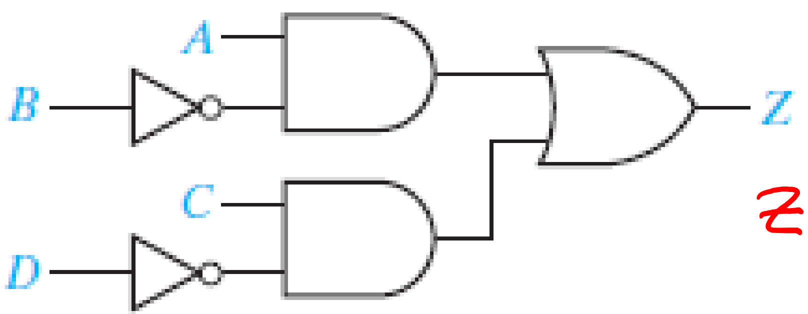
Example

$\underbrace{\text{The alarm will ring}}_{Z}$ iff $\underbrace{\text{the alarm switch is on}}_{A}$ $\boxed{\text{and}}$
$\underbrace{\text{the door is not closed}}_{B'}$ $\boxed{or}$ $\underbrace{\text{it is after 6 P.M.}}_{C}$ $\boxed{\text{and}}$
$\underbrace{\text{the window is not closed.}}_{D'}$

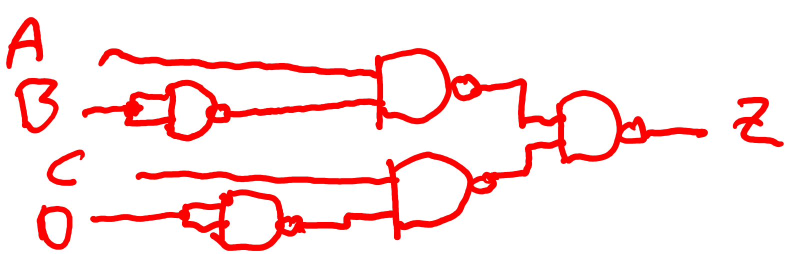
$A = 1$ if alarm switch is on
$B = 1$ if door is closed
$C = 1$ if after 6pm
$D = 1$ if window is closed

$$Z = AB' + CD'$$

- Already as simple as possible
- But could re-arrange if different gates available
- e.g. NAND

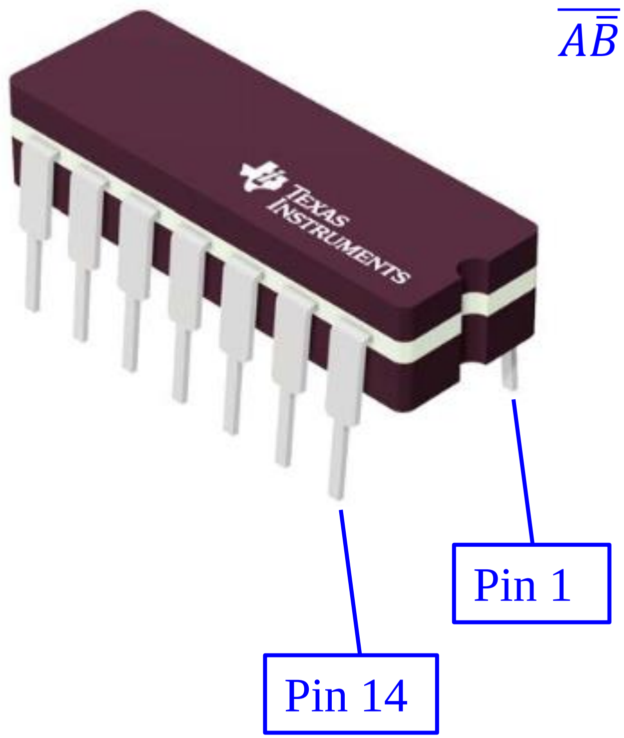


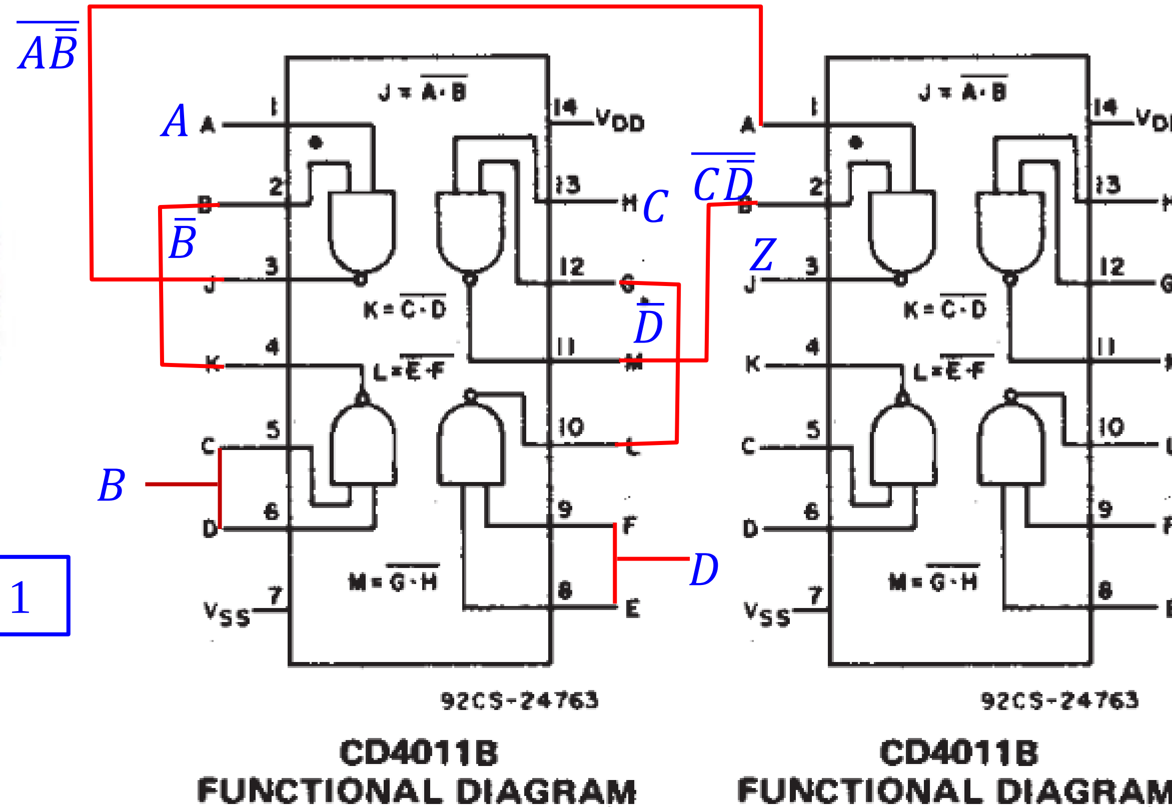
$$Z = A\overline{B} + C\overline{D} = \overline{\overline{A\overline{B} + C\overline{D}}}$$
$$= \overline{(\overline{A\overline{B}})(\overline{C\overline{D}})}$$



To the Logic Circuit

TI CD4011B – Quad 2 Input CMOS NAND

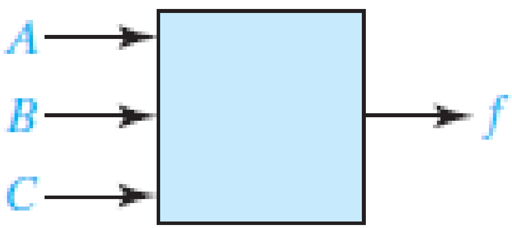




Logic Design via Truth Table

Example: Consider a three-input, one output system where

- A, B, C are inputs that represent digits of binary number N, and
- f is the output such that
  - $f = 1$ if $N \geq 011_2$ and
  - $f = 0$ if $N < 011_2$



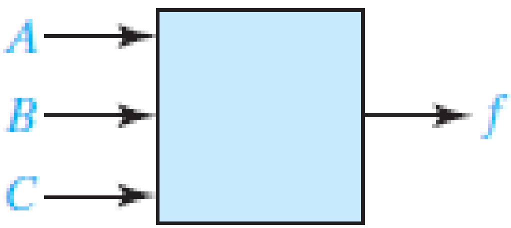
(a)

A	B	C	f	f′
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

(b)

OR these to find the expression that yields $f = 1$

Logic Design via Truth Table



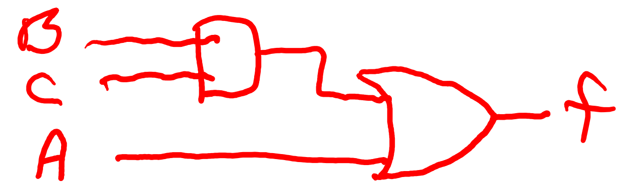
(a)

A	B	C	f	f′
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

(b)

OR these to find the expression that yields $f = 1$

Sketch circuit



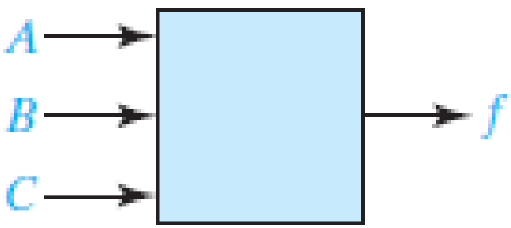
$$f = A'BC + AB'C' + AB'C + ABC' + ABC$$

$$f = A'BC + AB'\underbrace{(C' + C)}_{1} + AB\underbrace{(C' + C)}_{1}$$

Can be simplified to $f = A'BC + \underbrace{AB' + AB}_{A\underbrace{(B' + B)}_{1}} = A'BC + A = A + BC$

Elimination theorem

Logic Design via Truth Table



(a)

A	B	C	f	f'
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

(b)

Alternatively, could write f' by OR-ing

$$f' = A'B'C' + A'B'C + A'BC'$$

Then use DeMorgan's to find f

Alternatively, could write f based on the 0's of the function

$$f = (A + B + C)(A + B + C')(A + B' + C)$$

Also simplifies to $f = A + BC$