Computer Systems Fundamentals

CSE 232 Computer Systems Fundamentals

Dr. Gamal Fahmy

Logic Gates

Boolean Algebra

Gray Coded Number

- Rule of thumb
  - 1 bit change per increment

BCD

0 00
1 01
2 10
3 11

0	00
1	01
2	11
3	10

0 000
1 001
2 011
3 010
4 110
5 111
6 101
7 100

0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

Digital Logic

- **There are three types of logic gates from which we can build all digital computers.** AND OR
- **They're all related to semantic logic operators** NOT
- **Consider:**

If Ahmed asks Tarek for Tennis AND Tarek accepts, then they'll go for a tennis game.

Digital Logic

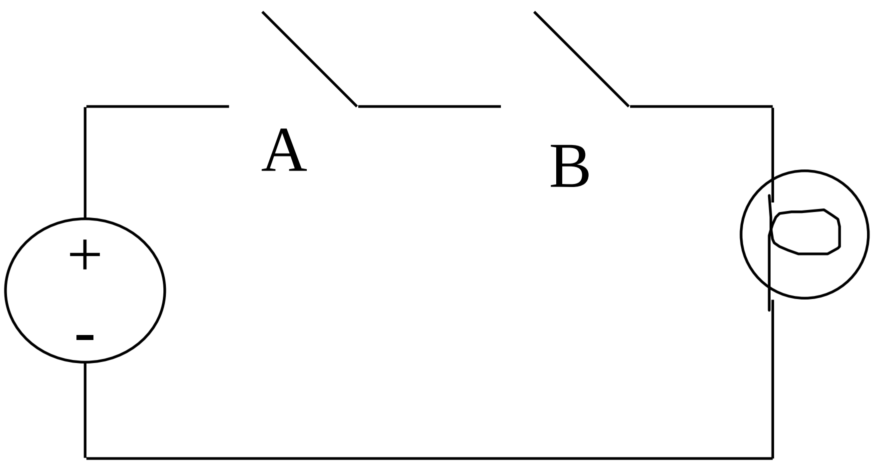
- **Let T(*) be the truth value operator:**
  - **g = T(They'll go to a tennis game)**
  - **A = T(Ahmed Asks Tarek)**
  - **~~B =~~ T(Tarek Accepts)**
- **g=A and B = A * B = A B**
  - **(If, Then provide the semantic equivalent of the equal sign.)**

Truth Table

A	B	g
0	0	0
0	1	0
1	0	0
1	1	1

Digital Logic

- **There is an electrical analog to the AND operator.**
- **True = Switch Closed   True = Light On**



AND Operator
Truth Table

A	B	g=A B
(F) 0	(F) 0	(F) 0
0	1	0
1	0	0
1	1	1

Digital Logic

Consider:

If Ahmed asks Tarek OR Mona Asks Tarek then, he’ll go to a tennis game.

Let T(*) be the truth value operator:

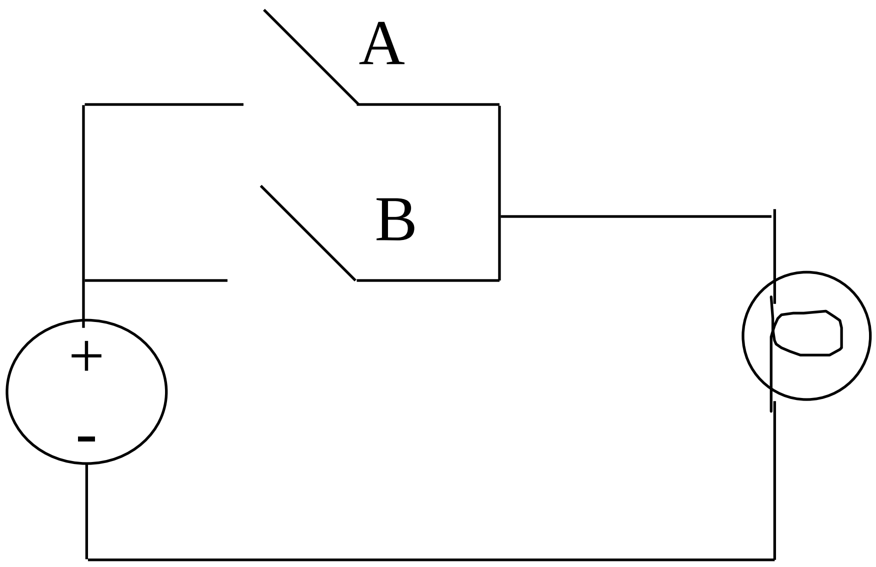
- **g = T (He’ll go to a tennis game)**
- **A = T (Ahmed Asks Tarek)**
- **B = T (Mona Asks Tarek)**

g=A OR B = A + B

A	B	g
0	0	0
0	1	1
1	0	1
1	1	1

Digital Logic

- **There is an electrical analog to the OR operator.**
- **True = Switch Closed   True = Light On**

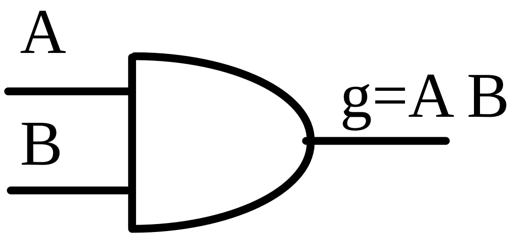


OR Operator
Truth Table

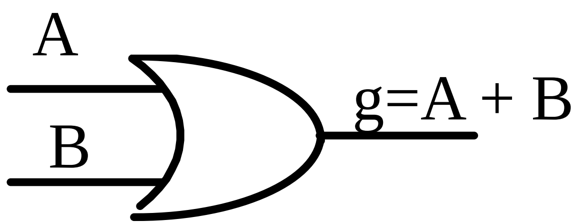
A	B	g=A + B
(F) 0	(F) 0	(F) 0
0	1	1
1	0	1
1	1	1

Gates

- **An AND gate has the electrical schematic:**

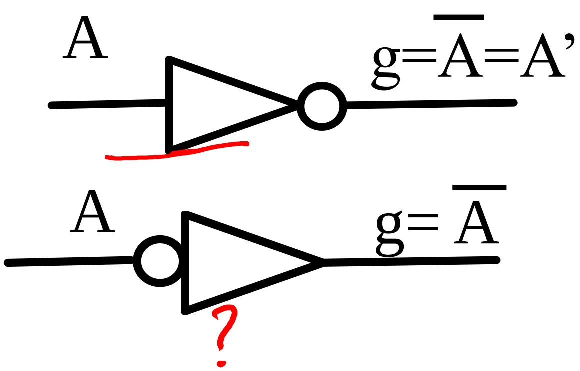


- **An OR gate has the electrical schematic:**



Gates

- **One Last Logical operator/gate - NOT**



NOT Operator
Truth Table

A	g = A'
0	1
1	0

- **All digital computers are built using ONLY these three gate types: AND, OR, Inverter**

OR

A OR B = g

$g_1 = A+B$

A AND B = g_2

$g_2 = A.B$

A NOT

= NOT A = g_3

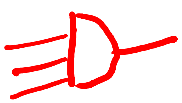
$g_3 = \overline{A}$

Digital Logic

- **All of our truth tables had 4 input combinations**
  - **2 Ways to pick the first input**
  - **2 Ways to pick the second input**
  - **2^2 Input Combinations**
  - **How many input combinations for a 3 input gate?**

Digital Logic

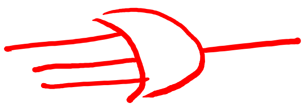
- **3 Input AND/OR Truth Tables**

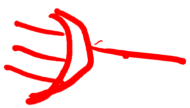


3-input AND
Truth Table

A	B	C	Y
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

3-Input OR
Truth Table

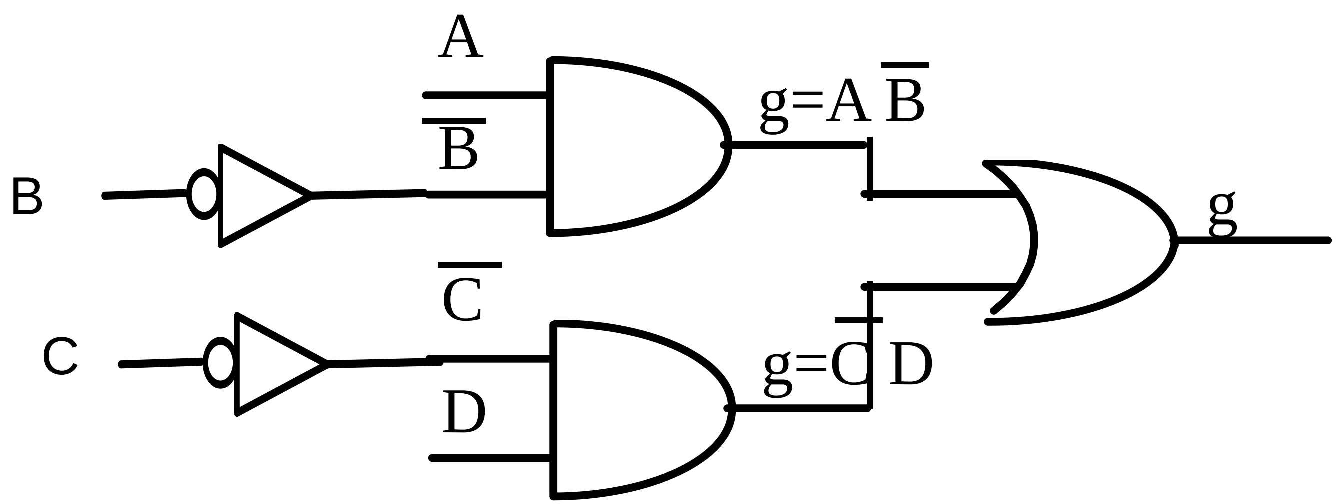




A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

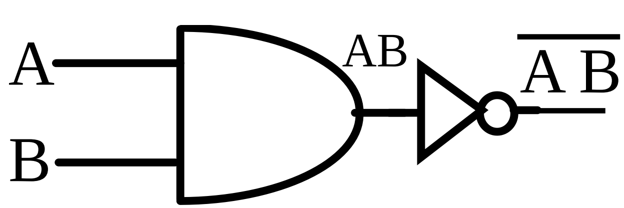
Digital Logic

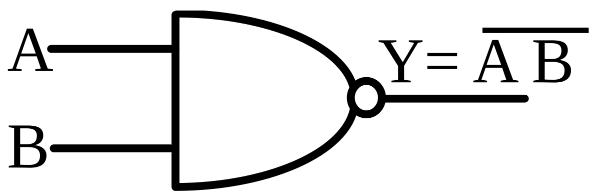
- Draw a Schematic for $g=A\overline{B}+\overline{C}D$



Digital Logic

NAND Gate=NOT (AND)





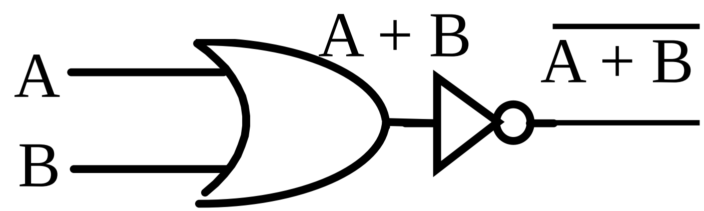
OR ≠ NAND

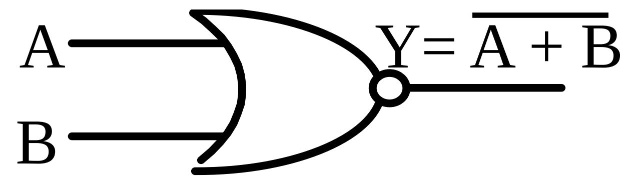
NAND
Truth Table

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

A+B
0
1
1
1

NOR Gate = NOT(OR)



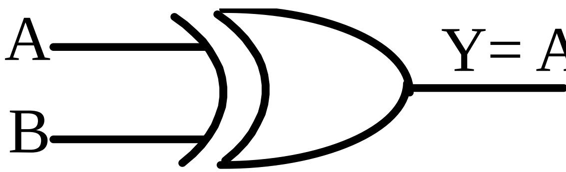


NOR
Truth Table

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

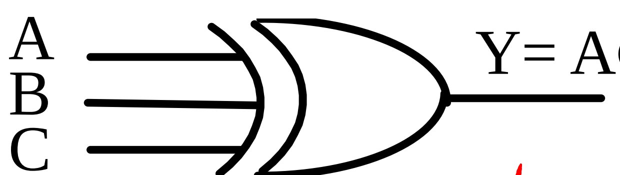
Digital Logic

- Exclusive OR:Output = 1 if odd number of inputs = 1



$Y = A \oplus B = A\bar{B} + \bar{A}B$

A	B	
0	0	0
0	1	1
1	0	1
1	1	0



$Y = A \oplus B \oplus C$

3-Input XOR Truth Table

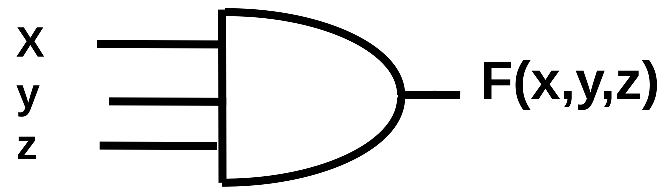
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$Y = A\bar{B} + \bar{A}B$

A	B	$A\bar{B}$	$\bar{A}B$	Y
0	0	0	0	0
0	1	0	1	1
1	0	1	0	[illegible]
1	1	0	0	[illegible]

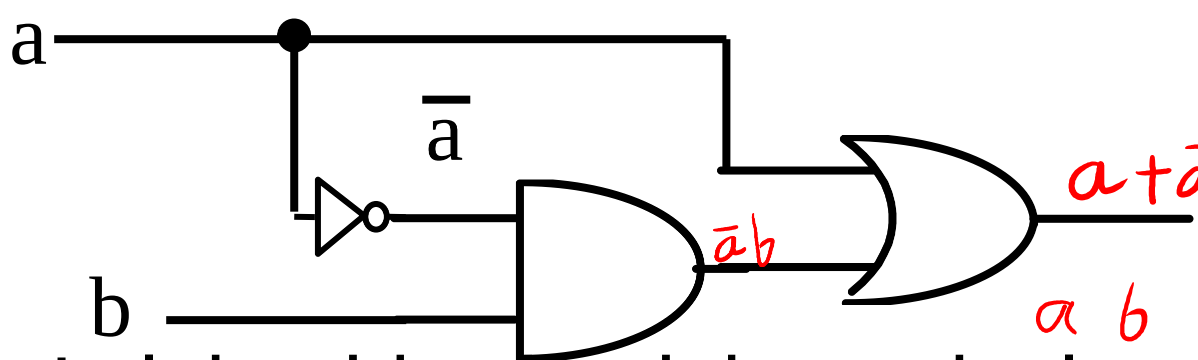
Boolean Algebra

1. It is representing inputs/ variables x,y,z and output as function on the inputs F(x,y,z)
2. + is OR
3. ● is AND
4. ‘ is NOT



Boolean Algebra

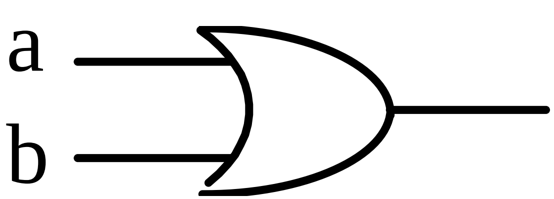
- Why study Boolean Algebra? Consider:



$a+\bar{a}b =$

a	b	a	$\bar{a}b$	$a+\bar{a}b$
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	1	0	1

- I claim this circuit is equivalent to:



- Could use a truth table.
- For a 10 Variable function need 2^{10} lines.

$= A+B$

Boolean Algebra

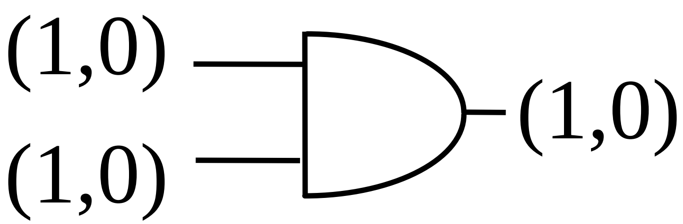
- George Boole (1815-1864), a mathematician sought to formalize logic using mathematical notation.
- He developed a consistent set of laws that were sufficient to define a new type of algebra: Boolean Algebra cf. Linear Algebra
- Many of the rules are the same as Linear Algebra. Some are different.

Boolean Algebra

- There are 7 fundamental laws, that tell us what operations are valid in Boolean algebra.

1. ***Field***: Boolean algebra operates over a field of numbers B with only two elements {0,1}.
2. ***Closure***: For every a, b in B,

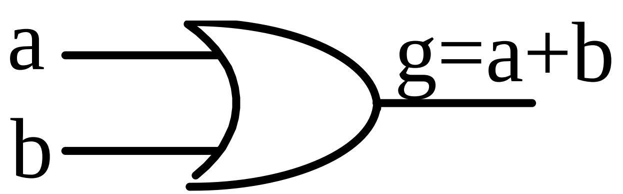
- $a + b$ is in B
- $a*b$ is in B

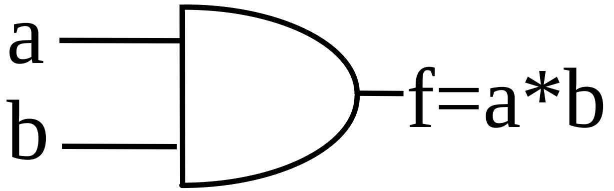


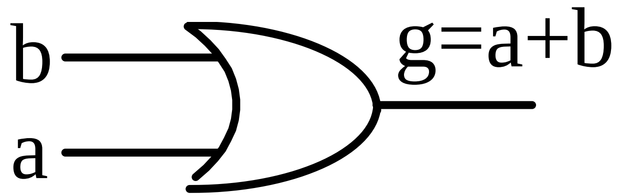
Boolean Algebra

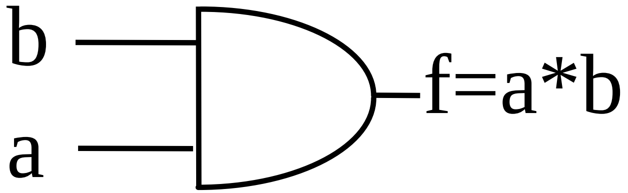
3. **_Commutative laws_**: For every a, b in B,

- $a + b = b + a$
- $ab = ba$
  - » Similar to Linear Algebra





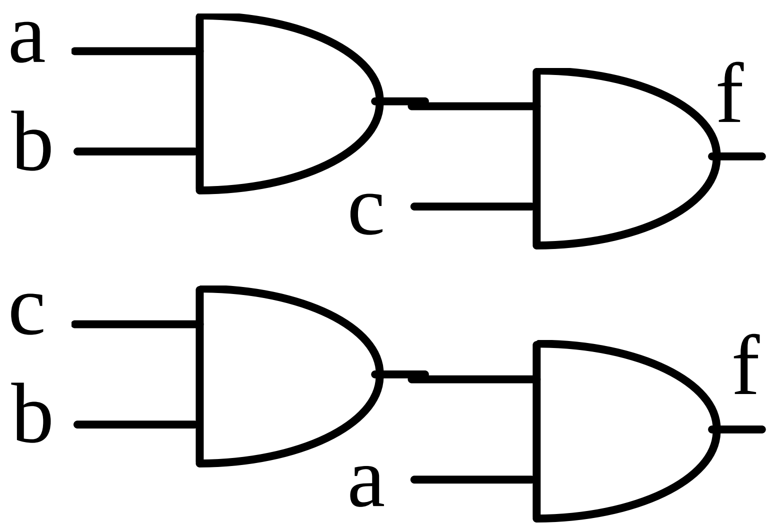




Boolean Algebra

4.***Associative laws***: For every a, b, c in B,

- $(a + b) + c = a + (b + c) = a + b + c$
- $(ab)\ c = a\ (bc) = abc$
  - » Similar to Linear Algebra



Boolean Algebra

5. ***Distributive laws***: **For every *a*, *b*, *c* in *B*,**

- **$a + (bc) = (a + b)(a + c)$ ('+' Distributes Over '*')**

  (NOT Similar to Linear Algebra.)

- **$a(b + c) = (ab) + (ac)$ ('*' Distributes Over '+')**

  (Similar to Linear Algebra.)

6. ***Complement***: **For each *a* in *B*, there exists an element *a* in *B* (the complement of *a*) such that:**

- ***a.*** **$a + \overline{a} = 1$ (Similar to Mult. Inverse of Linear Algebra)**
- **b.** **$a*\overline{a}= 0$ (Similar to Additive. Inverse of Linear Algebra)**

Boolean Algebra

$a + 0 = a$
$a + 1 = 1$
$a . 0 = 0$
$a . 1 = a$

- *7.* ***Identity***
- a. There exists an identity element with respect to {+}, designated by 0, such that *a* + 0 = *a* for every *a* in *B*.
- (Similar to Additive Ident. of Linear Algebra)

OR Operator
Truth Table

A	B	g=A + B
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	1	1

AND Operator
Truth Table

A	B	g=A B
0	0	0
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

b. There exists an identity element with respect to {*}, designated by 1, such that *a**1 = *a* for every a in *B*.

(Similar to Multiplicative. Ident. of Linear Algebra)

Summary

Commutative

$a + b = b + a$ $\quad$ $ab = ba$

Associative

$(a + b) + c = a + (b + c)$

$(ab)\ c = a\ (bc)$

Distributive

$a + (bc) = (a + b)(a + c)$

$a(b + c) = (ab) + (ac)$

Identity

$a + 0 = a$ $\quad$ $a*1 = a$

Complement

$a + \overline{a} = 1$ $\quad$ $a * \overline{a} = 0$

Or with 1 $\quad$ ***And with 0***

$a + 1 = 1$ $\quad$ $a *0= 0$

Idempotent

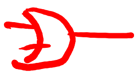
$a + a = a$ $\quad$ $a * a = a$

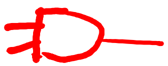
Boolean Algebra

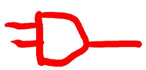
Duality Theorem: The Dual of every theorem is a valid theorem. The dual is constructed by interchanging:

$1 \longleftrightarrow 0$

$+ \longleftrightarrow *$

$X+1=1$ 

$X.1=X$ 

$X.0=0$ 

Or With 1: X+1=1 $X+0=X$

Dual of Or With 1: X*0=0 (And With 0)

Idempotent: X+X=X

Dual of Idempotent: X*X=X

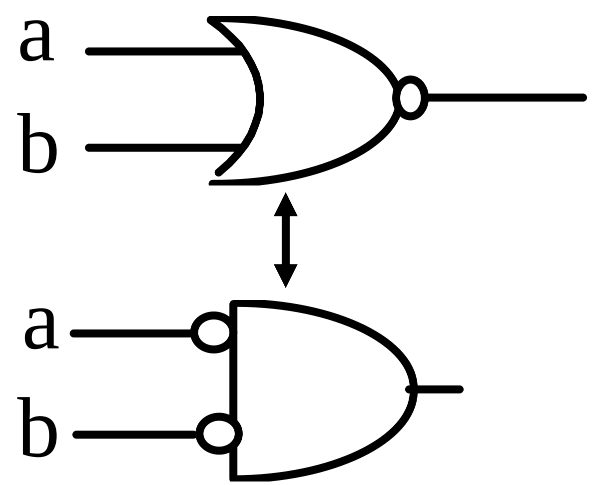
One last useful Theorem: DeMorgan's Law

Boolean Algebra

Gate Equivalency

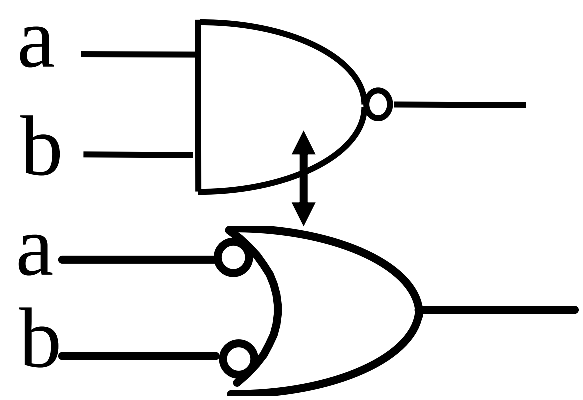
- **DeMorgan's Law**

$$\overline{a} \bullet \overline{b} = \overline{a + b}$$



- **Dual of DeMorgan's Law**

$$\overline{a} + \overline{b} = \overline{a \bullet b}$$



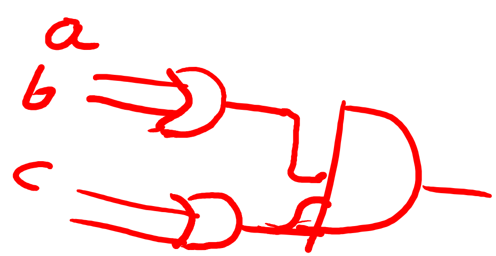
$\bar{a} \cdot \bar{b} = \overline{a+b}$

$\overline{a \cdot b} = \bar{a} + \bar{b}$

I only have NAND

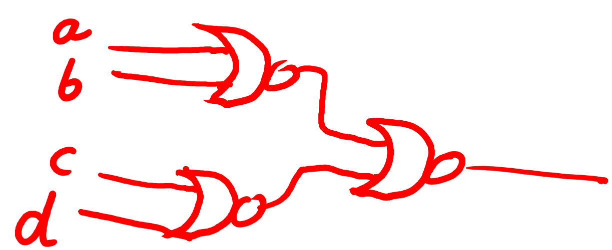
NOR

$F = (a+b)(c+d)$

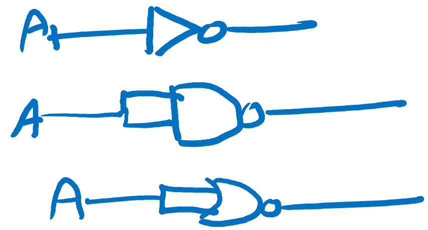


$= \overline{\overline{(a+b)(c+d)}}$

$= \overline{\overline{a+b} + \overline{c+d}}$



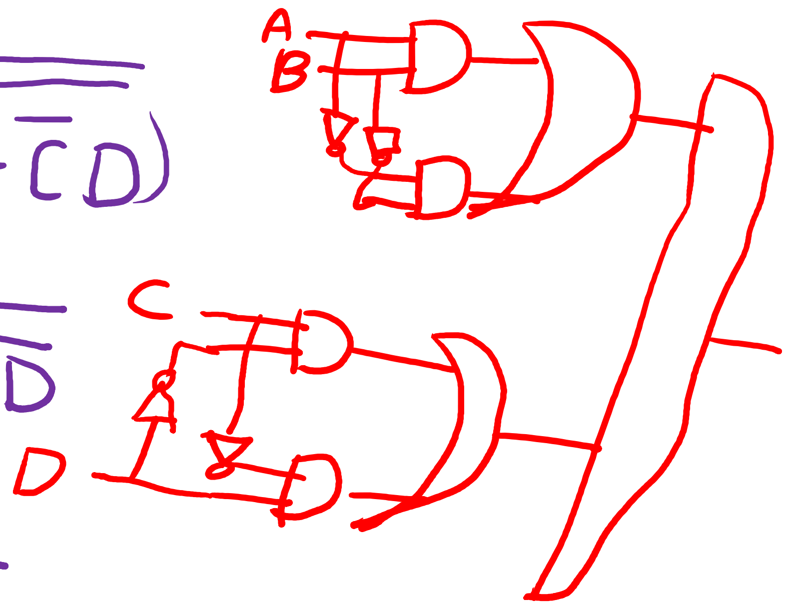
$\bar{A} = \overline{A \cdot A} = \overline{A + A}$



$\overline{\bar{a} \cdot \bar{b}} = a + b$

$\overline{a \cdot b} = \bar{a} + \bar{b}$

$\rightarrow (AB + \bar{A}\bar{B})(C\bar{D} + \bar{C}D)$ using NAND



$\left(\overline{\overline{AB + \bar{A}\bar{B}}}\right)\overline{\overline{(C\bar{D} + \bar{C}D)}}$

$\rightarrow \overline{\overline{AB} \cdot \overline{\bar{A}\bar{B}}} \; \overline{\overline{C\bar{D}} \cdot \overline{\bar{C}D}}$

$\rightarrow = \overline{\overline{\overline{\overline{AB}\;\overline{\bar{A}\bar{B}}}\;\overline{\overline{C\bar{D}}\;\overline{\bar{C}D}}}}$

Boolean Algebra

Q:Why is this useful?

A:It allows us to build functions using only one gate type.

Q:Why do we use NAND/NOR gate to build functions rather than AND/OR logic?

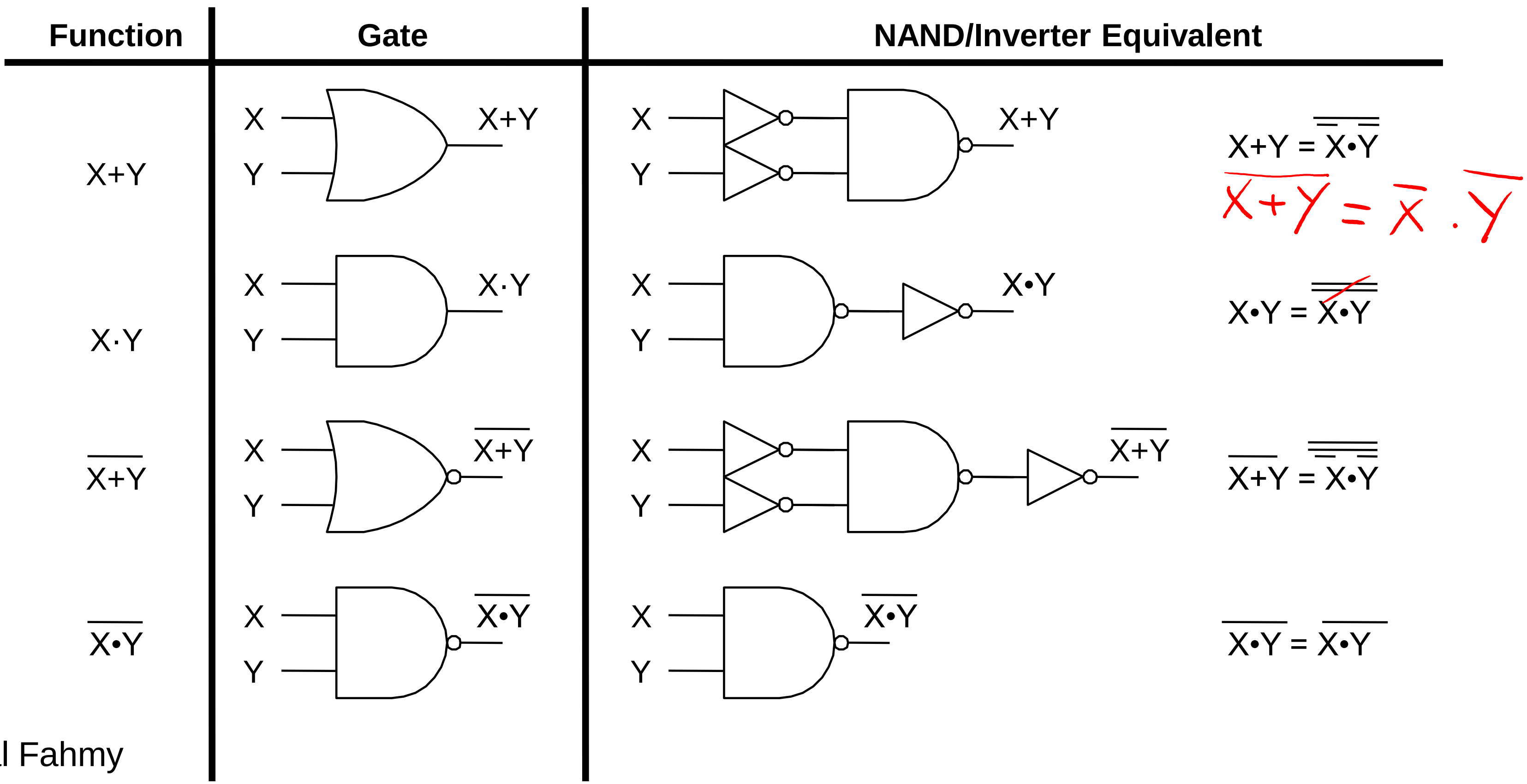
A:NAND/NOR gates are physically smaller and faster.

Q:How does Gate Equivalency help when drawing schematics with one gate type?

A:Let's look at three approaches.

Boolean Algebra

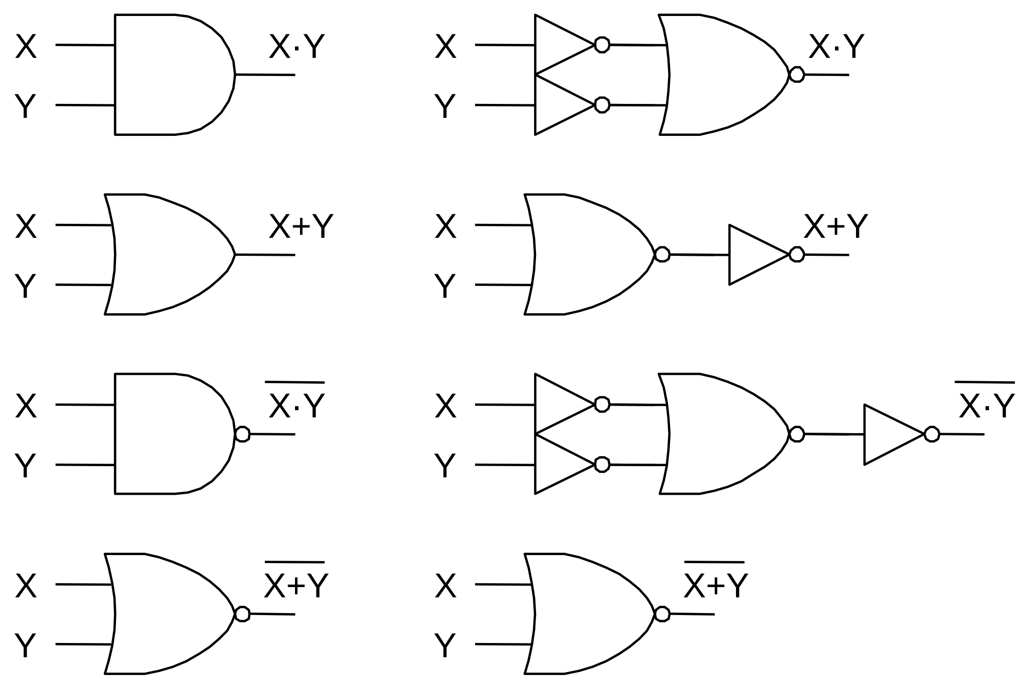
- **One approach is to construct AND/OR logic and replace each gate with it's equivalent using the figures shown below.**



Boolean Algebra

- **The drawback to such an approach is that you may use more than the minimal number of gates.**
- **The advantage is that the approach is simple.**

Function	Gate	NOR/Inverter Equivalent	
$X \cdot Y$			$X \cdot Y = \overline{\overline{X}+\overline{Y}}$
$X+Y$			$X+Y = \overline{\overline{X+Y}}$
$\overline{X \cdot Y}$			$\overline{X \cdot Y} = \overline{\overline{\overline{X}+\overline{Y}}}$
$\overline{X+Y}$			$\overline{X+Y} = \overline{X+Y}$

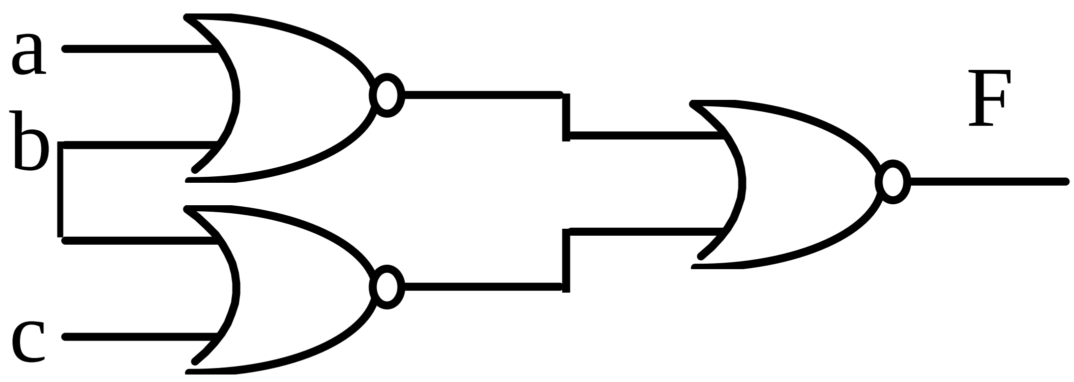


Boolean Algebra

- **There are two traditional approaches that yield less costly realizations:**
  - **EX: Using ONLY NOR gates, draw a schematic for the following function: F=(a+b)(b+c)**

Approach #1: Use DeMorgan's Law:

$$F=\overline{\overline{(a+b)(b+c)}}$$

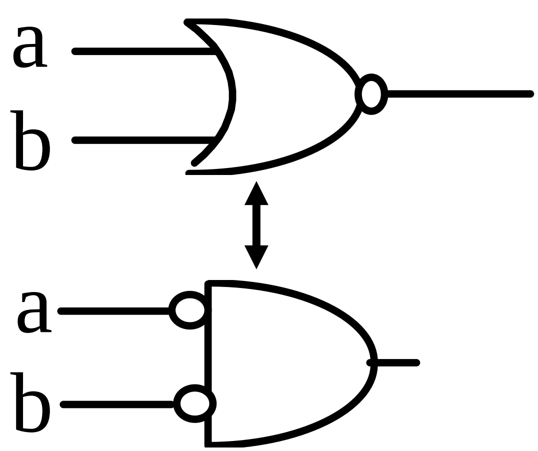


Boolean Algebra

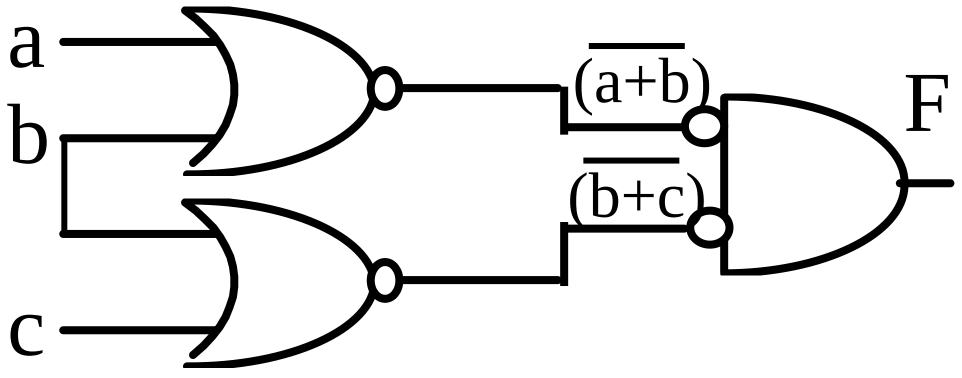
- Drawbacks to using DeMorgan’s Laws:
  - Some functions require many manipulations to get into proper form.
  - During these many manipulations it is easy to make mistakes.

Boolean Algebra

EX: Using ONLY NOR gates, draw a schematic for the following function: $F=(a+b)(b+c)$



Approach #2:Use Gate Equivalency (Draw gate form that you need. Account for inversions.)



Note that the inversions cancel and we're left with AND/OR LOGIC.

Boolean Algebra

- General Approach for building functions with 2-input NAND gates:
  - 1. Break function into two pieces.
  - 2. If the pieces form a:
    - AND/NAND function ===> build function with NAND+Inverters/NAND.
    - OR function ===>build with NAND (OR with inverted inputs)
    - NOR ($F=\overline{X+Y}$) function ===>Work with complement of the function ($F = \overline{X+Y}$), which is an OR, then complement F to get $\overline{F}$.
  - 3. Recursively apply these rules

$\overline{a}\cdot\overline{b} = \overline{a+b}$

$\overline{ab} = \overline{a}+\overline{b}$

علي + محمود only NOR

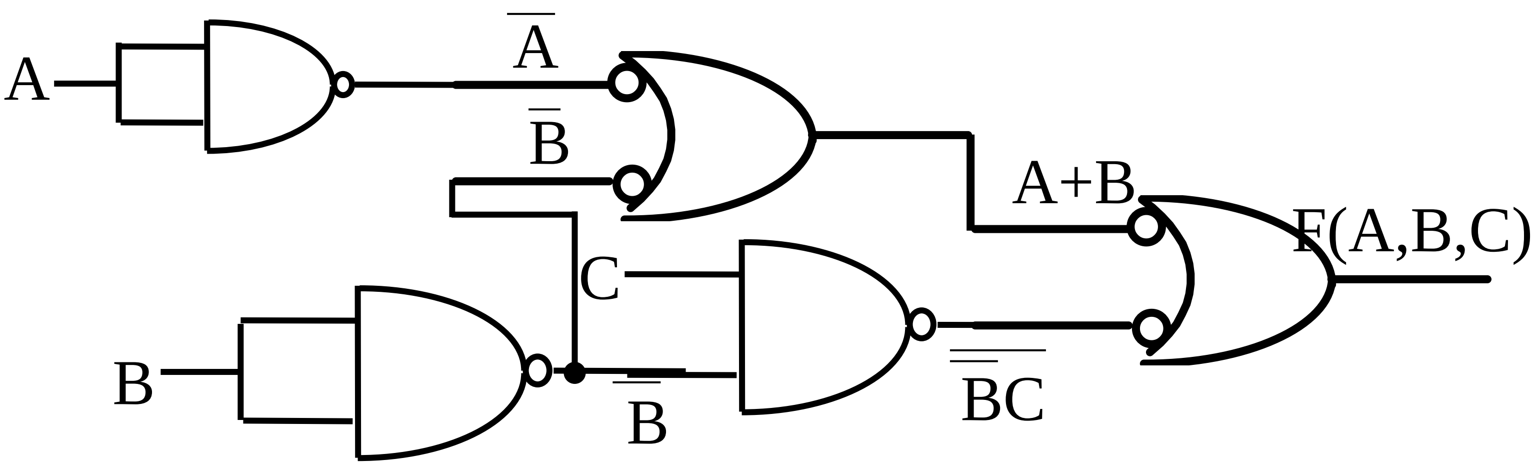
$\overline{\overline{\text{علي} + \text{محمود}}}$

only NAND

$\overline{\overline{\text{علي}}\cdot\overline{\text{محمود}}} = \overline{\overline{\text{علي} + \text{محمود}}}$

Boolean Algebra

- Ex: Assume you have the forms:
  - $F=\overline{(A+B)} + \overline{B}C$
- & want to find a NAND gate implementation.
- Break the function in the following way.
  - $F(A\ .\ .\ .\ Z)=\overline{g(A\ .\ .\ .\ Z) +}\ h(A\ .\ .\ .\ Z)$.



Boolean Algebra

- General Approach for building function with 2-input NOR gates:
  - 1. Break function into two pieces.
  - 2. If the pieces form an:
    - OR/NOR function ===> build function with NOR+Inverters/NOR
    - AND function ===> build function with NOR (AND with inverted inputs)
    - NAND ($F=\overline{X*Y}$) function ===>Work with complement of the function (F), which is an AND, then complement F to get F.
  - 3. Recursively apply these rules.

Boolean Algebra

$\bar{a}\,\bar{b} = \overline{a+b}$

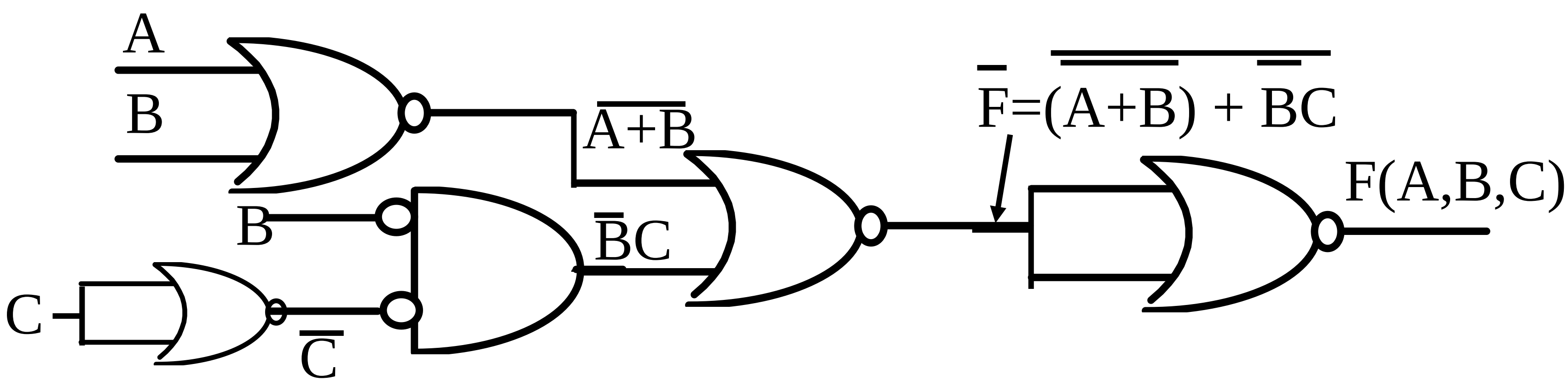
$\overline{ab} = \bar{a} + \bar{b}$

- EX: Assume you have the forms:
  - $F = \overline{(A+B)} + \bar{B}C$

$\bar{B}C = \overline{B + \bar{C}}$

- **Want to find a NOR gate implementation.**

- Break the function in the following way.
  - $F(A\ldots Z) = \overline{g(A\ldots Z)} + h(A\ldots Z).$



Boolean Algebra

The Uniting Theorem:

Minimize the following function:

$$F = A\overline{B} + AB$$

$$F = A(\overline{B} + B) \quad \text{Distr.}$$

$$F = A * 1 \quad \text{Compl..}$$

$$F = A \quad \text{Ident..}$$

A	B	F
0	0	0
0	1	0
1	0	1
1	1	1

These two 1's are physically and address adjacent.

Defn: Two 1's are address adjacent if their 'addresses' differ in only one bit position.

SOP → $F = A\overline{B} + AB$

Sum of Products

Boolean Algebra

We can use this connection between adjacency and the Uniting Thm.to visually implement the uniting Theorem:

Rule of Thumb(Min SOP): Eliminate the literal with CHANGING Address bit and write Canonical SOP for group of 1's using remaining literals.

A	B	F
0	0	0
0	1	0
1	0	1
1	1	1

$\}\ F=A$

Boolean Algebra

TEAMS: Find Min. SOP form for the following Truth Table:

A	B	F
0	0	1
0	1	0
1	0	1
1	1	0

$$F = \overline{A}\,\overline{B} + A\overline{B}$$

Q? Are the 1's address adjacent?

Boolean Algebra

By analogy we can generate a rule of thumb for generating a Minimum POS form.

Rule of Thumb(Min POS): Eliminate the literal with CHANGING Address bit and write Canonical POS for group of 0's using remaining literals.

Boolean Algebra

Ex: Find Min. POS form for the following Truth Table:

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

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

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

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

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

Boolean Algebra

Ex: Find Min. POS form for the following Truth Table:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

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

SOP and POS Forms

- We need to be able to translate a truth table description into a logical expression. Two forms are used:
- Sum-of-Products and Product-of-Sums Canonical Form
- Consider first Sum-of-Products Canonical Form Using an Example:

SOP Canonical Form

Term #	A	B	C	F	F1	F2	F3	F4	F1+F2+F3+F4
0	0	0	0	1	1	0	0	0	1
1	0	0	1	1	0	1	0	0	1
2	0	1	0	0	0	0	0	0	0
3	0	1	1	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0
5	1	0	1	1	0	0	1	0	1
6	1	1	0	1	0	0	0	1	1
7	1	1	1	0	0	0	0	0	0

- Construct Each Function F1, F2, etc.

$F1 = \overline{A}\,\overline{B}\,\overline{C}$ □ $F2 = \overline{A}\,\overline{B}\,C$ □ $F3 = A\,\overline{B}\,C$

□ $F4 = A\,B\,\overline{C}$

$$F = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + A\overline{B}\,C + AB\overline{C}$$

SOP Canonical Form

Term #	A	B	C	F
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

- Minterm: "Product" term containing all literals used by the function.
- Minterm Short-hand Notation:

$$F = \sum m(0,1,5,6)$$

$$F = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + A\overline{B}C + AB\overline{C}$$

POS Canonical Form

Term #	A	B	C	F	FA	FB	FC	FD	FA*FB*FC*FD
0	0	0	0	1	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1
2	0	1	0	0	0	1	1	1	0
3	0	1	1	0	1	0	1	1	0
4	1	0	0	0	1	1	0	1	0
5	1	0	1	1	1	1	1	1	1
6	1	1	0	1	1	1	1	1	1
7	1	1	1	0	1	1	1	0	0

- Construct Each Function FA, FB, etc.
- $FA = A + \overline{B} + C$
- $FB = A + \overline{B} + \overline{C}$
- $FC = \overline{A} + B + C$
- $FD = \overline{A} + \overline{B} + \overline{C}$
- $F = (A + \overline{B} + C)\,(A + \overline{B} + \overline{C})\,(\overline{A} + B + C)\,(\overline{A} + \overline{B} + \overline{C})$

POS Canonical Form

Term #	A	B	C	F
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

- Maxterm: "Sum" term containing all literals used by the function.
- Maxterm Short-hand Notation:

$$F = \sum m(0,1,5,6) \qquad F = \Pi M(2,3,4,7)$$

- $F=(A+\overline{B}+C)\,(A+\overline{B}+\overline{C})\,(\overline{A}+B+C)\,(\overline{A}+\overline{B}+\overline{C})$
- Compare MinTerm and MaxTerm notation. What do you observe about the union of the integer arguments?

Home work

Problems:

Homework 3,4,5