# ECE380 Digital Logic

Introduction to Logic Circuits:
Boolean algebra



## Axioms of Boolean algebra

- Boolean algebra: based on a set of rules derived from a small number of basic assumptions (***axioms***)

- 1a $0 \cdot 0 = 0$
- 1b $1 + 1 = 1$
- 2a $1 \cdot 1 = 1$
- 2b $0 + 0 = 0$
- 3a $0 \cdot 1 = 1 \cdot 0 = 0$
- 3b $1 + 0 = 0 + 1 = 1$
- 4a If $x = 0$ then $x' = 1$
- 4b If $x = 1$ then $x' = 0$

## Single-Variable theorems

- From the axioms are derived some rules for dealing with single variables
- 5a $x \cdot 0 = 0$
- 5b $x + 1 = 1$
- 6a $x \cdot 1 = x$
- 6b $x + 0 = x$
- 7a $x \cdot x = x$
- 7b $x + x = x$
- 8a $x \cdot x' = 0$
- 8b $x + x' = 1$
- 9 $x'' = x$

- Single-variable theorems can be proven by perfect induction
- Substitute the values $x=0$ and $x=1$ into the expressions and verify using the basic axioms



## Duality

- Axioms and single-variable theorems are expressed in pairs
  - Reflects the importance of ***duality***
- Given any logic expression, its dual is formed by replacing all + with ·, and vice versa and replacing all 0s with 1s and vice versa
  - f(a,b)=a+b dual of f(a,b)=a·b
  - f(x)=x+0 dual of f(x)=x·1
- The dual of any true statement is also true

## Two & three variable properties

- 10a. $x \cdot y = y \cdot x$ *Commutative*
- 10b. $x + y = y + x$

- 11a. $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ *Associative*
- 11b. $x + (y + z) = (x + y) + z$

- 12a. $x \cdot (y + z) = x \cdot y + x \cdot z$ *Distributive*
- 12b. $x + y \cdot z = (x + y) \cdot (x + z)$

- 13a. $x + x \cdot y = x$ *Absorption*
- 13b. $x \cdot (x + y) = x$



## Two & three variable properties

- 14a. $x \cdot y + x \cdot y' = x$ *Combining*
- 14b. $(x + y) \cdot (x + y') = x$

- 15a. $(x \cdot y)' = x' + y'$ *DeMorgan's*
- 15b. $(x + y)' = x' \cdot y'$ *Theorem*

- 16a. $x + x' \cdot y = x + y$
- 16b. $x \cdot (x' + y) = x \cdot y$

## Induction proof of $x+x'\cdot y=x+y$

- Use perfect induction to prove $x+x'\cdot y=x+y$

| $x$ | $y$ | $x'y$ | $x+x'y$ | $x+y$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |

equivalent ($x+x'y$ and $x+y$)



## Perfect induction example

- Use perfect induction to prove $(xy)'=x'+y'$

| $x$ | $y$ | $xy$ | $(xy)'$ | $x'$ | $y'$ | $x'+y'$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |

equivalent ($(xy)'$ and $x'+y'$)

## Proof (algebraic manipulation)

- Prove
  - (X+A)(X'+A)(A+C)(A+D)X = AX
  - (X+A)(X'+A)(A+C)(A+D)X
  - (X+A)(X'+A)(A+CD)X (using *12b*)
  - (X+A)(X'+A)(A+CD)X
  - (A)(A+CD)X (using *14b*)
  - (A)(A+CD)X
  - AX (using *13b*)



## Algebraic manipulation

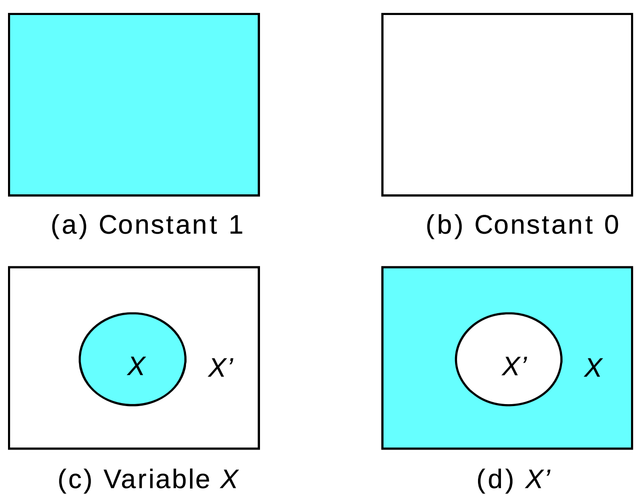
- Algebraic manipulation can be used to simplify Boolean expressions
  - Simpler expression => simpler logic circuit
- Not practical to deal with complex expressions in this way
- However, the theorems & properties provide the basis for automating the synthesis of logic circuits in CAD tools
  - To understand the CAD tools the designer should be aware of the fundamental concepts

# Venn diagrams

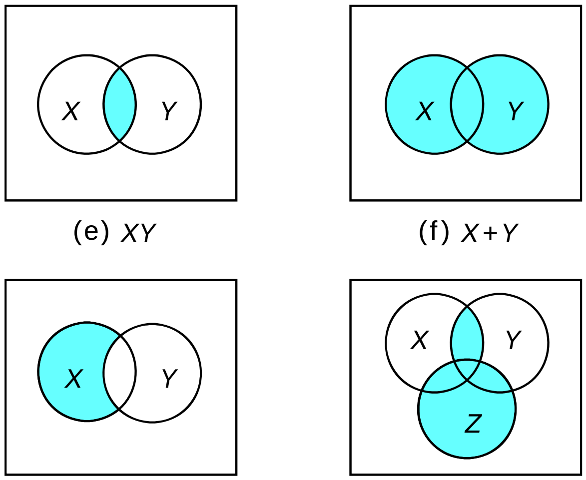
- Venn diagram: graphical illustration of various operations and relations in an algebra of sets
- A set *s* is a collection of elements that are members of *s* (for us this would be a collection of Boolean variables and/or constants)
- Elements of the set are represented by the area enclosed by a contour (usually a circle)



# Venn diagrams



# Venn diagrams

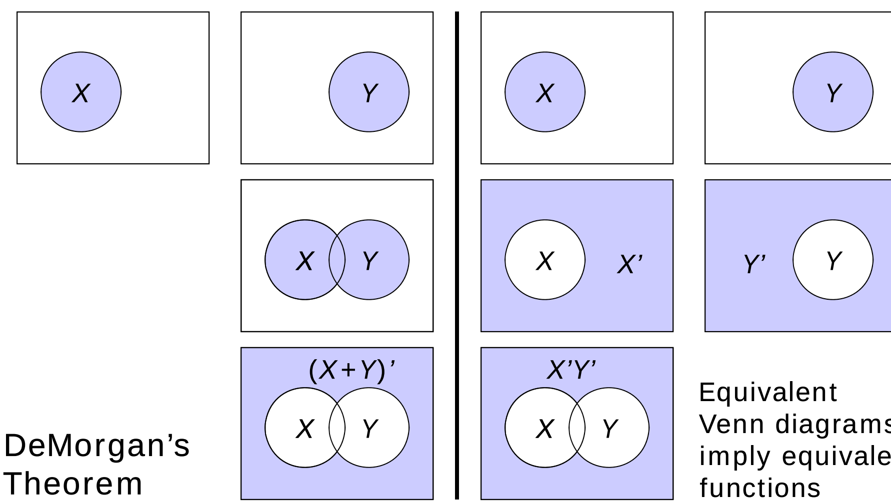



(e) $XY$ (f) $X+Y$

(g) $XY'$ (h) $XY+Z$



# Venn diagrams $(x+y)' = x'y'$




DeMorgan's Theorem

Equivalent Venn diagrams imply equivalent functions

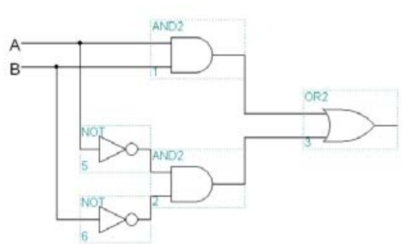
## Notation and terminology

- Because of the similarity with arithmetic addition and multiplication operations, the **OR** and **AND** operations are often called the ***logical sum*** and ***product*** operations
- The expression
  - ABC+A’BD+ACE’
  - Is a sum of three product terms
- The expression
  - (A+B+C)(A’+B+D)(A+C+E’)
  - Is a product of three sum terms



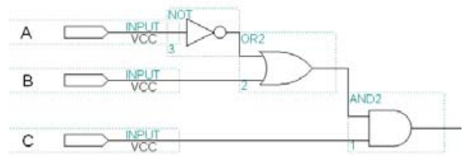
## Precedence of operations

- In the absence of parentheses, operations in a logical expression are performed in the order
  - NOT, AND, OR
- Thus in the expression AB+A’B’, the variables in the second term are complemented before being ANDed together. That term is then ORed with the ANDed combination of A and B (the AB term)



# Precedence of operations

- Draw the circuit diagrams for the following
  - f(a,b,c)=(a'+b)c




  - f(a,b,c)=a'b+c