Boolean Algebra and Logic Gates
Chapter 2

EECE 256
Dr. Sidney Fels
Steven Oldridge

Topics

- Definitions of Boolean Algebra
- Axioms and Theorems of Boolean Algebra
  - two valued Boolean Algebra
- Boolean Functions
  - simplification
- Canonical forms
  - minterm and maxterms
- Other logic gates

Boolean Algebra

- Allows us to define and simplify functions of binary variables
- Important for designers to create complex circuits
  - functions of computer
  - ASIC devices
  - programmable logic
  - determine machine state transitions

Boolean Algebra

- Adheres to the laws of an algebra
  - closure
  - associative
  - commutative
  - identity
  - inverse
  - distributive

  \+ for addition (0 is identity)

  ● for multiplication (1 is identity)

Axioms of Boolean Algebra

- closure for + and ●
- Identity:

  x + 0 =   x ● 1 =
- commutative

  x + y =   x ● y = y ● x
- distributive

  x (y + z) =   x + (y z) =

Axioms of Boolean Algebra

- Complement
  - x + x' = 1   x ● x' = 0
- two elements for Two-Valued Boolean Algebra

  0 and 1; 0 != 1

  AND = ●, OR = + , NOT = inverse
  - check with Truth tables and you'll see it meets all the axioms
- switching algebra (Shannon, 1928)
  - basis of all digital computers
- Precedence:
  - parentheses, NOT, AND, OR

Theorems and Properties of Boolean Algebra

Table 2.1
Postulates and Theorems of Boolean Algebra

Postulate 2 identity	(a)	$x + 0 = x$
Postulate 5 complement	(a)	$x + x' = 1$
Theorem 1 idempotent	(a)	$x + x = x$
Theorem 2 0 and 1 ops	(a)	$x + 1 = 1$
Theorem 3, involution		$(x')' = x$
Postulate 3, commutative	(a)	$x + y = y + x$
Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$
Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$
Theorem 6, absorption	(a)	$x + xy = x$

Theorems and Properties of Boolean Algebra

Table 2.1
Postulates and Theorems of Boolean Algebra

Postulate 2 identity	(a)	$x + 0 = x$
Postulate 5 complement	(a)	$x + x' = 1$
Theorem 1 idempotent	(a)	$x + x = x$
Theorem 2 0 and 1 ops	(a)	$x + 1 = 1$
Theorem 3, involution		$(x')' = x$
Postulate 3, commutative	(a)	$x + y = y + x$
Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$
Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$
Theorem 6, absorption	(a)	$x + xy = x$

Duality: interchange 0 for 1 and AND and OR

Theorems and Properties of Boolean Algebra

Table 2.1
Postulates and Theorems of Boolean Algebra

Postulate 2 identity	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
Postulate 5 complement	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Theorem 1 idempotent	(a)	$x + x = x$	(b)	$x \cdot x = x$
Theorem 2 0 and 1 ops	(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$
Theorem 3, involution		$(x')' = x$		
Postulate 3, commutative	(a)	$x + y = y + x$	(b)	$xy = yx$
Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$	(b)	$x(yz) = (xy)z$
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$	(b)	$x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$
Theorem 6, absorption	(a)	$x + xy = x$	(b)	$x(x + y) = x$

Duality: interchange 0 for 1 and AND and OR

Theorems used to simplify complex functions of binary variables

Useful Theorems

- Simplification Theorems:
  - X • Y + X • Y' =
  - X + X• Y =
  - (X + Y') • Y =
- **DeMorgan's Law:**
  - (X + Y)' =
- Theorems for Multiplying and Factoring:
  - (X + Y) • (X' + Z) = X •Z + X' •Y
- Proofs by algebra complicated
  - use truth tables instead

Some algebraic proofs

Proving Theorems via axioms of Boolean Algebra:

e.g., Prove: X •Y + X • Y' = X

e.g., Prove: X + X • Y = X

e.g., Prove: (X + Y) • (X' + Z) = X •Z + X' •Y

Some algebraic proofs

Proving Theorems via axioms of Boolean Algebra:

e.g., Prove: X •Y + X • Y' = X

LHS = X (Y + Y') distributive
= X(1) complement
= X = RHS identity

e.g., Prove: X + X • Y = X

LHS = X (1+Y) distributive
= X(1) identity
= X = RHS

Some algebraic proofs

e.g., Prove: (X + Y) • (X' + Z) = X •Z + X' •Y
LHS = (X+Y)X' + (X+Y)Z distributive
= XX' + YX' + XZ + YZ distributive
= 0 + X'Y + XZ + YZ complement, associative, distributive
= X'Y(Z + Z') + XZ (Y + Y') + YZ (X + X') identity/complement
= X'YZ + X'YZ' + XYZ + XY'Z + XYZ + X'YZ distributive, associative
= XZ(Y+Y') + X'Y(Z+Z') idempotent, associative, distributive
= XZ + X'Y = RHS complement

Some proofs using truth tables

DeMorgan's Law
(X +Y)' = X' • Y'

X	Y	X'	Y'	(X+Y)	(X+Y)'	X'*Y'
0	0	1	1			
0	1	1	0			
1	0	0	1			
1	1	0	0			

(X • Y)' = X' + Y'

X	Y	X'	Y'	(X •Y)	(X*Y)'	X'+Y'
0	0	1	1			
0	1	1	0			
1	0	0	1			
1	1	0	0			

Some proofs using truth tables

DeMorgan's Law
(X +Y)' = X' • Y'

X	Y	X'	Y'	(X+Y)	(X+Y)'	X'*Y'
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

(X • Y)' = X' + Y'

X	Y	X'	Y'	(X •Y)	(X*Y)'	X'+Y'
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

DeMorgan's Thereom

Example:

Z = A'B'C + A'BC + AB'C + ABC'

Z' = (A+B+C') • (A+B'+C') • (A'.....

Boolean Functions

- Now, we have everything to make Boolean Functions
  - F = f(x,y,z...) where x, y, z etc. are binary values (0,1) with Boolean operators
  - circuits can implement the function
  - algebra used to simplify the function to make it easier to implement

Example

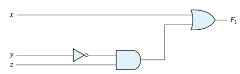
- $F_1 = x + y'z$

x	y	z	y'z	x+y'z
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Example

- $F_1 = x + y'z$

x	y	z	y'z	x+y'z
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	0	1
1	0	1	1	1
1	1	0	0	1
1	1	1	0	1



Simplification allows for different implementations

- F = AB + C(D + E) = requires 3 levels of gates

2-level implementation

- F = AB + C(D + E) = AB + CD + CE

Canonical Forms

- Express all Boolean functions as one of two canonical forms
  - enumerates all combinations of variables as either
    - Sum of Products, i.e., m1 + m2 + m3 … etc
    - Product of Sums, i.e., M1 • M2 • M3 … etc
  - each variable appears in normal form (x) or its complement (x')
  - if it is a product it is called a MINTERM
  - if is is a sum it is called a MAXTERM
  - n variables -> 2^n MINTERMS or MAXTERMS

Canonical Forms

Table 2.3
Minterms and Maxterms for Three Binary Variables

			Minterms		Maxterms	
x	**y**	**z**	**Term**	**Designation**	**Term**	**Designation**
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

Canonical Form Example

Table 2.4
Functions of Three Variables

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Canonical Form Example

Table 2.4
Functions of Three Variables

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1 m1	0
0	1	0	0	0
0	1	1	0	1 m3
1	0	0	1 m4	0
1	0	1	0	1 m5
1	1	0	0	1 m6
1	1	1	1 m7	1 m7

Canonical Form Example: Sum of Products (Minterms)

- So we can read off of TT directly
- Sum of products is sum of Minterms

$$\sum m_i$$

F1 = m1 + m4 + m7
= x'y'z + xy'z' + xyz

F2 = m3 + m5 + m6 + m7
= x'yz + xy'z + xyz' + xyz

Canonical Form Example

Table 2.4
Functions of Three Variables

x	y	z	Function f_1	Function f_2
0	0	0	0 M0	0 M0
0	0	1	1	0 M1
0	1	0	0 M2	0 M2
0	1	1	0 M3	1
1	0	0	1	0 M4
1	0	1	0 M5	1
1	1	0	0 M6	1
1	1	1	1	1

Canonical Form Example: Product of Sums (Maxterms)

- So we can read off of TT directly
- Product of sums is product of Maxterms

$$\prod M_i$$

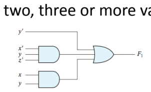
F1 = M0 • M2 • M3 • M5 • M6
= (x + y + z)(x + y’ + z)(x + y’ + z’)(x’ + y + z’)(x’+y’+z)
F2 = M0•M1•M2•M4
= (x+y+z)(x+y+z’)(x+y’+z)(x’+y+z)

Converting between them

- You can use complement and deMorgan’s theorem
  - if F=m1 + m3 + m5 i.e. Σ(1, 3, 5) then
  - F’ = m0 + m2 + m4 + m6 +m7
    - F = (m0 + m2 + m4 + m6 +m7)’
  - use DeMorgan’s now to get Product of Sum
    - F = Π(0,2,4,6,7)
- Remember to include all Minterms/Maxterms
  - n variables, 2^n terms

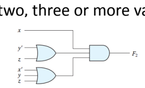
Standard form

- Sum of Products with one, two, three or more variables in product form
  - F1 = y’ + xy + x’yz’



(a) Sum of Products

- Product of Sum with one, two, three or more variables in sum form
  - F2 = x • (y’+z) • (x’+y+z’)



(b) Product of Sums

- Notice: canonical and standard form are 2-level implementations
  - but may have many inputs for gate
    - called fan-in; limited by pins on IC and manufacturing

Other logical operations

- for 2 input gates, you can have 16 different logic operations 2^{2^n} where n = 2

Table 2.7
Truth Tables for the 16 Functions of Two Binary Variables

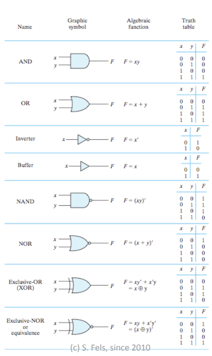
x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Other logical operations

Table 2.8
Boolean Expressions for the 16 Functions of Two Variables

Boolean Functions	Operator Symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x + y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y, then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x, then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

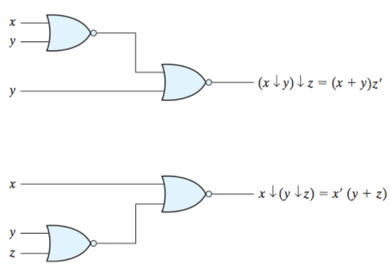
Digital Logic Gates



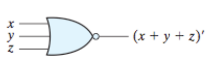
Extending to multiple inputs

- works fine for
  - AND, OR; no problem – commute and associate
  - NAND, NOR – commute but don't associate ☹
    - so, be careful when using them cascaded
    - instead:
      - define multi-input NAND as multi-input AND that is inverted at the end
        - » x NAND y NAND z = (xyz)'
      - define multi-input NOR as multi-input OR that is inverted at the end
        - » x NOR y NOR Z = (x+y+z)'

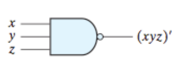
Extending to multiple inputs



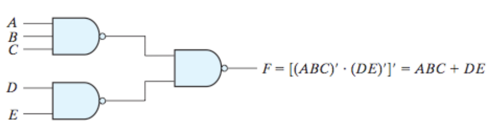
Extending to multiple inputs



(a) 3-input NOR gate



(b) 3-input NAND gate



(c) Cascaded NAND gates

Summary

- two-valued Boolean algebra supports
  - switching logic
  - simplification postulates and theorems
  - digital logic gates
- Truth tables can be used to define function
- Canonical and standard forms make it easy to create functions that can be implemented
- finite number of 2 input gates
  - easy to implement larger complex functions
- 2 input gates can be extended to multiple inputs