Switching Algebra (Chapter 2)

- Algebra represents (and can be used to manipulate) binary functions of binary inputs
- Manipulate: simplify (VERY nice machinery)

Boolean Algebra: Elements

- George Boole (1815-1864): “An investigation of the laws of thought” (1854)
- Boolean function: F(*vars*) = *expression*
- *Expression*: operators, variables, constants, groupings
  - Operators: Boolean + (OR), * (AND), and unary – (NOT)
  - Sometimes • used for AND
  - Sometimes ‘ used for NOT
  - Variables: Boolean-valued (0 and 1)
  - Constants: 0 and 1
  - Groupings: parentheses ( )

Definition of Boolean algebra

- The Boolean algebra is an algebra dealing with binary variables and logic operations.
  The variables are designated by letters of the alphabet, and the three basic logic operations are AND, OR, and NOT (complementation).
- A Boolean algebra is an algebraic structure <B,+,•,’> where B is a set of numerical elements, ‘ is a unary negation operator, and + and • are binary operators. This collection of stuff must satisfy the following **postulates** (or **axioms**).

Operators

- OR (+; binary “sum”) of 2 variables: output is 1 if either or both of the inputs is 1, otherwise 0
- AND (• or concatenation; binary product) of 2 variables: output is 1 if both of the inputs is 1, otherwise 0
- NOT (‘, overbar, ~; output is 1 if input is 0 and *vice versa*)

A	B	A+B	A•B
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

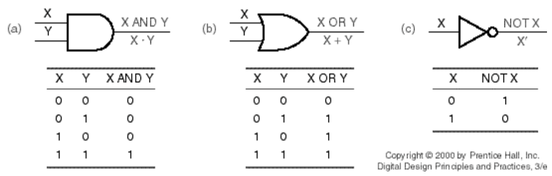
A	A′
0	1
1	0

Example Boolean functions

- $F_1(x,y,z) = x•y•z$
- $F_2(x,y,z) = x'•y'•z' + x'•y•z + x•y'$
- $F_3(x,y,z) = (x'+y)•(x'+z)$

- *Literal*: a variable or its complement
- *Product term*: literals connected by •
- *Expression*: RHS of a function

Punch line: Binary Logic and Gates

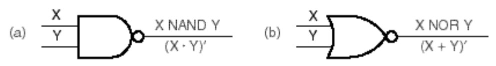


X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

X	NOT X
0	1
1	0

Punch line ctd.



X	Y	X NAND Y
0	0	1
0	1	1
1	0	1
1	1	0

X	Y	X NOR Y
0	0	1
0	1	0
1	0	0
1	1	0

Copyright © 2000 by Prentice Hall, Inc.
Digital Design Principles and Practices, 3/e

Identity properties

- +, OR, “disjunction”
  - 0 is the identity: ORing anything with 0 leaves it unchanged
  - **1**+0 = **1**, **0**+0 = **0**, 1+1 = 1
- •, AND, “conjunction”
  - 1 is the identity: ANDing anything with **1** leaves it unchanged
  - 1•**0** = **0**, 1•1 = **1**, 0•0 = 0

Postulates

Note: □ means “there exists”, □ means “for all”.

1. **Digital Abstraction:** □ x □ B = {0,1},
   - x=0 if x ≠ 1
   - x=1 if x ≠ 0
2. **(P5) Existence of complements and closure:** □ x □ B, □ x’ □ B (called the *complement* of x) such that
   - x + x’ = 1
   - x•x’ = 0

Postulates ctd.

3. **Truth tables for** •, +, ‘
   - 0+0=0, 0+1=1+0=1, 1+1=1
   - 0•0=0, 0•1=1•0=0, 1•1=1
   - 0’=1, 1’=0
4. **Closure:** ∀ x, y ∈ B,
   - x+y ∈ B
   - x•y ∈ B
   - x’ ∈ B

Postulates ctd.

5. **(P3) Existence of identity elements**
   - ∀ x ∈ B, ∃ 0 ∈ B such that x+0 = 0+x = x.
   - ∀ x ∈ B, ∃ 1 ∈ B such that x•1 = 1•x = x.
6. **(P1) Commutative laws:** ∀ x, y ∈ B,
   - x+y = y+x
   - x•y = y•x

Postulates ctd.

7. **(P2)Associative laws:** ∀ x, y, z ∈ B,
   - » x+(y+z) = (x+y)+z
   - » x•(y•z) = (x•y)•z
   - » ***Yields definition of n-input OR, AND***
8. **(P8) Distributive laws:** ∀ x, y, z ∈ B,
   - » x•(y+z) = x•y+x•z
   - » x+(y•z) = (x+y)•(x+z) *huh?*

The Duality Principle

- The dual of an expression is obtained by exchanging (• and +), and (1 and 0) in it.
- If a particular Boolean equation is valid, its dual is also valid, provided that the precedence of operations is not changed.
- One can replace • by + and + by • and 0 by 1 and 1 by 0 in an equality and the resulting equality remains true.
- The precedence of the operands must remain the same.
- Cannot exchange x with x’

Theorems of Boolean Algebra

- The postulates are basic axioms of the algebraic structure and need not be proven.
- Theorems must be proved from the postulates.

(P4) Null elements

- $x+1 = 1$
- $x \cdot 0 = 0$
- Prove by **perfect induction**: consider all values of x and verify equality.
  - $0+1 = 1$, $1+1 = 1$
  - $0 \cdot 0 = 0$, $1 \cdot 0 = 0$ (complement)

(P6) Idempotency

- x+x = x
- x•x = x

- Prove by perfect induction
  - 0+0 = 0
  - 1+1 = 1

  QED
- By duality, x•x = x holds.

(P7) Involution

- (x')' = x
- **Proof:** perfect induction again
- (0')' = (1)' = 0

  QED

Distributive laws (again)

Distributivity: $\forall$ x, y, z $\in$ B,

» x•(y+z) = x•y + x•z

» x + (y•z) = (x+y)•(x+z)

Provable via perfect induction

(P11) DeMorgan's laws

- THESE ARE IMPORTANT
- $(x+y)' = x' \cdot y'$
- $(x \cdot y)' = x' + y'$

(P12) Absorption (Covering)

- $x + x \cdot y = x$
- $x \cdot (x+y) = x$
- **Proof:**

$$\begin{aligned} x + x \cdot y &= x \cdot 1 + x \cdot y \\ &= x \cdot (1+y) \\ &= x \cdot (y + 1) \\ &= x \cdot 1 \\ &= x \end{aligned}$$

QED (second part true by duality)

(P13) Consensus

- $xy + x'z + yz = xy + x'z$
- $(x+y) \cdot (x'+z) \cdot (y+z) = (x+y) \cdot (x'+z)$
- **Proof:**

$$\begin{aligned} xy + x'z + yz &= xy + x'z + (x+x')yz \\ &= xy + x'z + xyz + x'yz \\ &= (xy + xyz) + (x'z + x'zy) \\ &= xy + x'z \end{aligned}$$

QED.

Truth table

- Enumerates all possible variable values (2^n combinations for n variables) and lists the value of the function for each set of input values.
- The truth tables for some functions named $F_1(x,y,z)$, $F_2(x,y,z)$, and $F_3(x,y,z)$ are to the right.

x	y	z		F_1	F_2	F_3
0	0	0		0	1	1
0	0	1		0	0	1
0	1	0		0	0	1
0	1	1		0	1	1
1	0	0		0	1	0
1	0	1		0	1	0
1	1	0		0	0	0
1	1	1		1	0	1

Truth Table ctd.

- Truth table: a unique representation of a Boolean function (except for the interchange of rows, whose order is arbitrary).
- If two functions have identical truth tables, the functions are equivalent.
- Truth tables can be used to prove equality theorems.
- However, the size of a truth table grows exponentially with the number of variables involved, hence unwieldy. This motivates the use of Boolean Algebra.

Boolean expressions are not unique

- Unlike truth tables, expressions representing a Boolean function are NOT unique.
- Example
  - $F(x,y,z) = x'\bullet y'\bullet z' + x'\bullet y\bullet z' + x\bullet y\bullet z'$
  - $G(x,y,z) = x'\bullet y'\bullet z' + y\bullet z'$
- The corresponding truth tables are to the right.

x	y	z		F	G
0	0	0		1	1
0	0	1		0	0
0	1	0		1	1
0	1	1		0	0
1	0	0		0	0
1	0	1		0	0
1	1	0		1	1
1	1	1		0	0

Not unique ctd.

- These two functions have identical truth tables, so F( ) and G( ) represent the same function.
- F( ) = G( ), even though their expressions appear different.

Algebraic Manipulation

- Boolean algebra is a useful tool for simplifying digital circuits.
- Why do it? Simpler is cheaper and smaller.
- Example: Simplify F= x'yz + x'yz' + xz.

  F = x'yz + x'yz' + xz

  = x'y(z+z') + xz

  = x'y•1 + xz

  = x'y + xz

Algebraic Manipulation ctd.

- Example: Prove

  x'y'z' + x'yz' + xyz' = x'z' + yz'
- **Proof:**

  x'y'z'+ x'yz'+ xyz'

  = x'y'z' + x'yz' + x'yz' + xyz'

  = x'z'(y'+y) + yz'(x'+x)

  = x'z'•1 + yz'•1

  = x'z' + yz'

  QED

Complement of a Function

- The complement of a function is obtained by interchanging AND and OR operations and complementing each variable and constant.
- Or change 1s to 0s and 0s to 1s in the truth table column containing F's value.
- One should not confuse the *complement* of a function with the *dual* of a function.

Complementation: example

- Find G(x,y,z), the complement of
  F(x,y,z) = x'yz' + x'y'z
- G = F' = (x'yz' + x'y'z)'
  = (x'yz')'•(x'y'z)' *DeMorgan*
  = (x+y'+z)•(x+y+z') *DeMorgan*

Dual: example

- Find H(x,y,z), the dual of
  F(x,y,z) = x'yz' + x'y'z
- H = (x'+y+z') (x'+y'+ z)

Canonical and Standard Forms

- We are preparing to consider formal techniques for the simplification of Boolean functions.
- Simplified functions in turn lead to "less expensive" logic circuits.

Maxterm and Minterm

- *Minterm*: For *n* variables, the minterm is a product (•, AND) term that contains each variable exactly once, in complemented or uncomplemented form.
- In minterm m_j, a variable is complemented if its value in the binary equivalent of *j* is 0.
- Must know the names and order of the variables! $F(x,y,z) = m_1 + m_4 + m_6$ means something specific.

Maxterm and Minterm ctd.

- Maxterm: For *n* variables, the maxterm is a sum (+, OR) term which contains each variable exactly once, in complemented or uncomplemented form.
- In maxterm M_j, a variable is complemented if its value in the binary equivalent of *j* is 1.
- Variable names and order is important: $F(a,b,c) = M_0 \bullet M_3 \bullet M_5$ means something specific.

Truth Table notation for minterms, maxterms

- Minterms and Maxterms are easy to denote using a truth table.
- Example (3 variables)

x	y	z		Minterm	Maxterm
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 Forms

- Any Boolean function f( ) can be expressed as a *unique* **sum** of **min**terms (order of variables fixed).
- The minterms included are those m_j such that f( ) = 1 in row *j* of the truth table for f( ).
- Any Boolean function f( ) can be expressed as a unique **product** of **max**terms (order of variables fixed).
- The maxterms included are those M_j such that f( ) = 0 in row *j* of the truth table for f( ).

Canonical forms

- Any function, therefore, has two canonical forms
  - Canonical Sum-Of-Products (sum of minterms)
  - Canonical Product-Of-Sums (product of maxterms)

Example

- Truth table for $f_1(a,b,c)$ at right
- The canonical sum-of-products form for f_1 is
  $f_1(a,b,c) = a'b'c + a'bc' + ab'c' + abc'$
- The canonical product-of-sums form for f_1 is
  $f_1(a,b,c)= (a+b+c)\bullet(a+b'+c')\bullet(a'+b+c')\bullet(a'+b'+c')$.

a	b	c	f_1
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	1
1	1	1	0

Shorthand: $\sum$ and $\prod$

- $f_1(a,b,c) = \sum m(1,2,4,6)$, where $\sum$ indicates that this is a sum-of-products form, and m(1,2,4,6) indicates that the minterms to be included are m_1, m_2, m_4, and m_6.
- $f_1(a,b,c) = \prod M(0,3,5,7)$, where $\prod$ indicates that this is a product-of-sums form, and M(0,3,5,7) indicates that the maxterms to be included are M_0, M_3, M_5, and M_7.

Conversion Between Canonical Forms

- Replace $\sum$ with $\prod$ (or *vice versa*) and replace those *j*s that appeared in the original form with those that do not.
- Example:

$$\begin{aligned} f_1(a,b,c) &= a'b'c + a'bc' + ab'c' + abc' \\ &= m_1 + m_2 + m_4 + m_6 \\ &= \textstyle\sum(1,2,4,6) \\ &= \textstyle\prod(0,3,5,7) \\ &= (a+b+c)\bullet(a+b'+c')\bullet(a'+b+c')\bullet(a'+b'+c') \end{aligned}$$

Standard Forms

- Standard forms are like canonical forms except that not all variables need appear in the individual product (SOP) or sum (POS) terms.
- Example:
  $f_1(a,b,c) = a'b'c + bc' + ac'$
  is a standard sum-of-products form
- $f_1(a,b,c) = (a+b+c)\bullet(b'+c')\bullet(a'+c')$
  is a standard product-of-sums form.

Conversion of SOP from standard to canonical form

- Expand non-canonical terms by inserting equivalent of 1 in each missing variable:
  $(x + x') = 1$
- Remove duplicate minterms
- $f_1(a,b,c) = a'b'c + bc' + ac'$
  $= a'b'c + (a+a')bc' + a(b+b')c'$
  $= a'b'c + abc' + a'bc' + abc' + ab'c'$
  $= a'b'c + abc' + a'bc + ab'c'$

Conversion of POS from standard to canonical form

- Expand noncanonical terms by adding 0 in terms of missing variables (*e.g.*, $xx' = 0$) and using the distributive law
- Remove duplicate maxterms
- $f_1(a,b,c) = (a+b+c)\bullet(b'+c')\bullet(a'+c')$
  $= (a+b+c)\bullet(aa'+b'+c')\bullet(a'+bb'+c')$
  $= (a+b+c)\bullet(a+b'+c')\bullet(a'+b'+c')\bullet$
  $(a'+b+c')\bullet(a'+b'+c')$
  $= (a+b+c)\bullet(a+b'+c')\bullet(a'+b'+c')\bullet(a'+b+c')$