Boolean Algebra
Outline

• Basic Gates in Digital Circuit
• Boolean Algebra : Definitions, Axioms
• Named, Simplification & Consensus Theorems
• Duality Principle, Shannon's Expansion
• Proof : Using Truth Table, Using Theorem
• Boolean function: Representation, Canonical form
Boolean Algebra

• Computer hardware using binary circuit greatly simply design

• Binary circuits: To have a conceptual framework to manipulate the circuits algebraically

• George Boole (1813-1864): developed a mathematical structure
  — To deal with binary operations with just two values.
Basic Gates in Binary Circuit

- Element 0: “FALSE”. Element 1: “TRUE”.
- ‘+’ operation “OR”, ‘*’ operation “AND” and ‘’ operation “NOT”.

<table>
<thead>
<tr>
<th></th>
<th>OR</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>AND</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>NOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
OR Gate

• ‘+’ operation “OR”

<table>
<thead>
<tr>
<th>OR</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ R = X + Y \]

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>R = X OR Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ 1 + Y = 1 \]
**AND Gate**

- ‘*’ operation “AND”

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>R = X * Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

0 * Y = 0
NOT Gate

• ‘ operation “NOT”

<table>
<thead>
<tr>
<th>X</th>
<th>R=X’</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Boolean Algebra Defined

• Boolean Algebra $B : 5$-tuple
  \[ \{ B, +, *, ', 0, 1 \} \]
• $+$ and $*$ are binary operators,
• $'$ is a unary operator.
Boolean Algebra Defined

• **Axiom #1: Closure**
  
  If a and b are Boolean
  
  \[(a + b)\] and \[(a * b)\] are Boolean.

• **Axiom #2: Cardinality**
  
  if \(a\) is Boolean then \(a'\) is Boolean

• **Axiom #3: Commutative**
  
  \[(a + b) = (b + a)\]
  
  \[(a * b) = (b * a)\]
Boolean Algebra Defined

• *Axiom #4: Associative*: If \( a \) and \( b \) are Boolean

\[
(a + b) + c = a + (b + c) \\
(a * b) * c = a * (b * c)
\]

• *Axiom #6: Distributive*

\[
a * (b + c) = (a * b) + (a * c) \\
a + (b * c) = (a + b) * (a + c)
\]
Boolean Algebra Defined

• **Axiom #5: Identity Element** :
  • B has identity to + and *
    
    0 is identity element for + : \( a + 0 = a \)
    
    1 is identity element for * : \( a * 1 = a \)

• **Axiom #7: Complement Element**

  \[ a + a' = 1 \]
  \[ a * a' = 0 \]
Terminology

• Juxtaposition implies * operation:

\[ ab = a \ast b \]

• Operator order of precedence is:

\[ ( ) > ' > * > + \]

\[ a + b c = a + (b \ast c) \neq (a + b) \ast c \]

\[ ab' = a(b') \neq (a \ast b)' \]
# Named Theorems

<table>
<thead>
<tr>
<th>Theorem</th>
<th>Equation 1</th>
<th>Equation 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idempotent</td>
<td>(a + a = a)</td>
<td>(a \ast a = a)</td>
</tr>
<tr>
<td>Boundedness</td>
<td>(a + 1 = 1)</td>
<td>(a \ast 0 = 0)</td>
</tr>
<tr>
<td>Absorption</td>
<td>(a + (a \ast b) = a)</td>
<td>(a \ast (a+b) = a)</td>
</tr>
<tr>
<td>Associative</td>
<td>((a+b)+c= a+(b+c))</td>
<td>((a \ast b) \ast c= a \ast (b \ast c))</td>
</tr>
<tr>
<td>Involution</td>
<td>((a')' = a)</td>
<td></td>
</tr>
<tr>
<td>DeMorgan’s</td>
<td>((a+b)' = a' \ast b')</td>
<td>((a \ast b)' = a' + b')</td>
</tr>
</tbody>
</table>
Simplification Theorem

• Uniting:
  \[ XY + XY' = X \quad \text{X(Y+Y')}=X.1=X \]
  \[ (X + Y)(X + Y') = X \quad \text{XX+XY'+YX+YY'}=X+X(Y+Y')+0=X \]

• Absorption:
  \[ X + XY = X \quad \text{X(1+Y)}=X.1=X \]
  \[ X(X + Y) = X \quad \text{XX+XY=X+XY=X} \]

• Adsorption
  \[ (X + Y')Y = XY, \quad XY' + Y = X + Y \quad \text{XY+YY'}=XY+0=XY \]
Principle of Duality

• Dual of a statement S is obtained
  • By interchanging * and +
  • By interchanging 0 and 1
  • By interchanging for all x by x’ also valid
    —(for an expression)
• Dual of \((a*1)*(0+a’) = 0\) is \((a+0)+(1*a’) = 1\)

\[
(a+b)’ = a’ * b’ \\
(a+b) * 1 = (a’*b’) + 0
\]
Consensus Theorem

• $XY + X'Z + YZ = XY + X'Z$

  $XY + X'Z + YZ$
  $= xy + x'z + (x + x')yz$
  $= xy + x'z + xyz + x'yz$
  $= xy + xyz + x'z + x'yz$
  $= xy(1 + z) + x'z(1 + y)$
  $= xy + x'z$

  Consensus (collective opinion) of $X.Y$ and $X'.Z$ is $Y.Z$

• $(X + Y)(X' + Z)(Y + Z) = (X + Y)(X' + Z)$
Shannon Expansion

• \( F(X, Y, Z) = X \cdot F(1,Y,Z) + X' \cdot F(0, Y, Z) \)

Example:

\( XY + X'Z + YZ \)

\[ = X \cdot (1 \cdot Y + 0 \cdot Z + YZ) + X' \cdot (0 \cdot Y + 1 \cdot Z + YZ) \]
\[ = X \cdot (Y + YZ) + X' \cdot (Z + YZ) \]
\[ = X \cdot (Y(1 + Z)) + X' \cdot (Z(1 + Y)) \]
\[ = X(Y \cdot 1) + X'(Z \cdot 1) \]
\[ = XY + X'Z \]
\( \text{N-bit Boolean Algebra} \)

- Single bit to \( \text{n-bit} \) Boolean Algebra
- Let \( a = 1101010, b = 1011011 \)
  - \( a + b = 1101010 + 1011011 = 1111011 \)
  - \( a \times b = 1101010 \times 1011011 = 1001010 \)
  - \( a' = 1101010' = 0010101 \)
Proof by Truth Table

• Consider the distributive theorem:
  \[ a + (b * c) = (a + b) * (a + c) \]
  Is it true for a two bit Boolean Algebra?

• Can prove using a truth table
  – How many possible combinations of \( a, b, \) and \( c \) are there?

• Three variables, each with two values
  \[-2\times2\times2 = 2^3 = 8\]
## Proof by Truth Table

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>b*c</th>
<th>a+(b*c)</th>
<th>a+b</th>
<th>a+c</th>
<th>(a+b)*(a+c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Proof using Theorems

- Use the properties of Boolean Algebra to proof

\[(x + y)(x + x) = x\]

- Warning, make sure you use the laws precisely

<table>
<thead>
<tr>
<th>((x + y)(x + x))</th>
<th>Given</th>
</tr>
</thead>
<tbody>
<tr>
<td>((x + y)x)</td>
<td>Idempotent</td>
</tr>
<tr>
<td>(x(x + y))</td>
<td>Commutative</td>
</tr>
<tr>
<td>(x)</td>
<td>Absorption</td>
</tr>
</tbody>
</table>
Converting to Boolean Equations

• Convert the following English statements to a Boolean equation

  – Q1. a is 1 and b is 1.
     • Answer: F = a AND b = ab
  – Q2. either of a or b is 1.
     • Answer: F = a OR b = a+b
  – Q3. both a and b are not 0.
     • Answer:
       – (a) Option 1: F = NOT(a) AND NOT(b) = a’b’
       – (b) Option 2: F = a OR b = a+b
  – Q4. a is 1 and b is 0.
     • Answer: F = a AND NOT(b) = ab’
Example: Converting a Boolean Equation to a Circuit of Logic Gates

- Q: Convert the following equation to logic gates:
  \[ F = a \text{ AND NOT}(b \text{ OR NOT}(c)) \]

(a)

(b)
Some Circuit Drawing Conventions

No

![Diagram of circuit drawing conventions showing 'No' example with 'X', 'Y', and 'F'.](image1)

Yes

![Diagram of circuit drawing conventions showing 'Yes' example with connections and labels.](image2)
Thanks