CSE 370 Lecture 3

Wrapping up 2’s Complement. Starting Boolean Algebra.

Lecture 2 Recap:

- Hexadecimal has 16 symbols, Decimal 10, Octal 8, Binary has 2
- Learnt the kung-fu required to switch between bases
- Binary in digital systems:
  - Finite and Fixed word length
  - How do we decide on a word length for our system?
  - \( \log_2 x = \text{No. of bits we will need} \)
- What about Negative Numbers?
  - Sign Magnitude
  - One’s complement
  - 2’s complement

Some Observations about 2’s complement

- Range: \(-2^{N-1} \) to \(2^{N-1} - 1\)

- The weird number: most negative number

- Trick: Fast way to do 2’s complement
Major topic: Combinational logic

- Axioms and theorems of Boolean algebra
- Logic functions and truth tables
  - AND, OR, Buffer, NAND, NOR, NOT, XOR, XNOR
- Gate logic
  - Networks of Boolean functions
- Canonical forms
  - Sum of products and product of sums
- Simplification
  - Boolean cubes and Karnaugh maps
  - Two-level simplification

Boolean Logic/Algebra

- Notation for writing down precise logical statements (in propositional logic)
- Primitives: true, false, variables
- Connectives: NOT, AND, OR, IMPLIES, ...
- (Almost) all memoryless digital circuits can be seen as Boolean algebra expressions
- Understanding Boolean logic helps us design “simpler” circuits, both by hand and automatically
- \((A \text{ AND } B) \text{ OR } (\text{NOT } A \text{ AND } B)\) AND \(A\)
- Equivalent to: \(A \text{ AND } B\)
Boolean algebra

- A Boolean algebra comprises...
  - A set of elements B
  - Binary operators \{+ , \} 
  - A unary operation \{ ' \}

- ...and the following axioms
  - 1. The set B contains at least two elements \{a, b\} with \(a = b\)
  - 2. Closure: \(a + b\) is in B \(a * b\) is in B
  - 3. Commutative: \(a + b = b + a\) \(a * b = b * a\)
  - 4. Associative: \((a + (b + c)) = (a + b) + c\) 
    \((a * (b * c)) = (a * b) * c\)
  - 5. Identity: \(a + 0 = a\) \(a * 1 = a\)
  - 6. Distributive: \((a + (b * c)) = (a + b) * (a + c)\)
    \((a * (b + c)) = (a * b) + (a * c)\)
  - 7. Complementarity: \(a + a' = 1\) \(a * a' = 0\)

Digital (binary) logic is a Boolean algebra

- Substitute
  - \{0, 1\} for B
  - AND for \(\cdot\) Boolean Product
  - OR for + Boolean Sum
  - NOT for \('\)

- All the axioms hold for binary logic

- Definitions
  - Boolean function
    - \(\) Maps inputs from the set \{0,1\} to the set \{0,1\}
  - Boolean expression
    - An algebraic statement of Boolean variables and operators
Logic functions and Boolean algebra

- Any logic function that is expressible as a truth table can be written in Boolean algebra using +, *, and '

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</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>

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</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>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X'</th>
<th>Y'</th>
<th>X*Y</th>
<th>X'*Y'</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Two key concepts

- Duality (a meta-theorem— *a theorem about theorems*)
  - All Boolean expressions have logical duals
  - Any theorem that can be proved is also proved for its dual
  - Replace: • with +, + with •, 0 with 1, and 1 with 0
  - Leave the variables unchanged

- de Morgan’s Theorem
  - Procedure for complementing Boolean functions
  - Replace: • with +, + with •, 0 with 1, and 1 with 0
  - Replace all variables with their complements

Useful laws and theorems

<table>
<thead>
<tr>
<th>Identity:</th>
<th>X + 0 =</th>
<th>Dual:</th>
<th>X • 1 =</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null:</td>
<td>X + 1 =</td>
<td>Dual:</td>
<td>X • 0 =</td>
</tr>
<tr>
<td>Idempotent:</td>
<td>X + X =</td>
<td>Dual:</td>
<td>X • X =</td>
</tr>
<tr>
<td>Involution:</td>
<td>(X')' =</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complementarity: X + X' =</td>
<td>Dual:</td>
<td>X • X' =</td>
<td></td>
</tr>
<tr>
<td>Commutative:</td>
<td>X + Y =</td>
<td>Dual:</td>
<td>X • Y =</td>
</tr>
<tr>
<td>Associative:</td>
<td>(X+Y)+Z=</td>
<td>Dual:</td>
<td>(X•Y)•Z=</td>
</tr>
<tr>
<td>Distributive:</td>
<td>X•(Y+Z)=</td>
<td>Dual:</td>
<td>X+(Y•Z)=</td>
</tr>
<tr>
<td>Uniting:</td>
<td>X•Y+X•Y'=X</td>
<td>Dual:</td>
<td>(X+Y)•(X+Y')=X</td>
</tr>
</tbody>
</table>
Useful laws and theorems (con’t)

Absorption: \( X + X \cdot Y = X \)  
Dual: \( X \cdot (X + Y) = X \)

Absorption (#2): \( (X + Y') \cdot Y = X \cdot Y \)  
Dual: \( (X \cdot Y') + Y = X + Y \)

De Morgan’s: \( (X + Y + ...) = X' \cdot Y' + ... \)  
Dual: \( (X \cdot Y + ...) = X' + Y' + ... \)

Duality: \( (X + Y + ...) = X \cdot Y + ... \)  
Dual: \( (X \cdot Y + ...) = X + Y + ... \)

Multiplying & factoring: \( (X + Y) \cdot (X' + Z) = X \cdot Z + X' \cdot Y \)  
Dual: \( X \cdot Y + X' \cdot Z = (X + Z) \cdot (X' + Y) \)

Consensus: \( (X \cdot Y) \cdot (Y \cdot Z) \cdot (X' \cdot Z) = X \cdot Y + X' \cdot Z \)  
Dual: \( (X + Y) \cdot (Y + Z) \cdot (X' + Z) = (X + Y) \cdot (X' + Z) \)

Proving theorems

- **Example 1:** Prove the uniting theorem-- \( X \cdot Y + X \cdot Y' = X \)

- **Example 2:** Prove the absorption theorem-- \( X + X \cdot Y = X \)
Logic simplification

- Use the axioms to simplify logical expressions
  - Why? To use less hardware

- Example: A two-level logic expression
  \[ Z = A'B'C + AB'C' + AB'C + ABC' + ABC \]