Combinational logic

- Switches
- Basic logic and truth tables
- Logic functions
- Boolean algebra
- Proofs by re-writing and by perfect induction

Switches: basic element of physical implementations

- Implementing a simple circuit (arrow shows action if wire changes to “1”):

  close switch (if A is “1” or asserted) and turn on light bulb (Z)

  open switch (if A is “0” or unasserted) and turn off light bulb (Z)

  \[ Z = A \]
Switches (cont’d)

- Compose switches into more complex ones (Boolean functions):

  \[
  Z = A \text{ and } B \\
  \]

  \[
  Z = A \text{ or } B \\
  \]

Switching networks

- Switch settings
  - determine whether or not a conducting path exists to light the light bulb

- To build larger computations
  - use the light bulb (output of the network) to set other switches (inputs to another network)
Transistor networks

- Modern digital systems are designed in CMOS technology
  - MOS stands for Metal-Oxide on Semiconductor
  - C is for complementary because there are both normally-open and normally-closed switches
- MOS transistors act as voltage-controlled switches
  - similar, though easier to work with than relays.

MOS transistors

- MOS transistors have three terminals: drain, gate, and source
  - they act as switches in the following way:
    - if the voltage on the gate terminal is (some amount) higher/lower than the source terminal then a conducting path will be established between the drain and source terminals

  \[
  \begin{align*}
  \text{n-channel} & : & \text{open when voltage at G is low} & \text{closed when:} & \text{voltage(G)} > \text{voltage (S)} + \varepsilon \\
  \text{p-channel} & : & \text{closed when voltage at G is low} & \text{opens when:} & \text{voltage(G)} < \text{voltage (S)} - \varepsilon
  \end{align*}
  \]
Most digital logic is CMOS

- CMOS logic gates are inverting
  - Easy to implement NAND, NOR, NOT while AND, OR, and Buffer are harder
Possible logic functions of two variables

- There are 16 possible functions of 2 input variables:
  - in general, there are $2^{2n}$ functions of n inputs

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>16 possible functions ($F_0$-$F_{15}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0000000011111111</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0000111100001111</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0011001100110011</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0101010101010101</td>
</tr>
</tbody>
</table>

X and Y \[ \equiv \ \text{not} (X \text{ nand } Y) \]

X \text{ or } Y \[ \equiv \ \text{not} (X \text{ nor } Y) \]

X \text{ not } \equiv \ \text{not} (X \text{ or } Y)

In fact, we can do it with only NOR or only NAND
- NOT is just a NAND or a NOR with both inputs tied together

\[
\begin{array}{c|c|c}
X & Y & X \text{ nor } Y \\
\hline
0 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\quad \begin{array}{c|c|c}
X & Y & X \text{ nand } Y \\
\hline
0 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

- and NAND and NOR are "duals", that is, it's easy to implement one using the other

\[
\begin{align*}
X \text{ nand } Y &= \text{not} (\text{not} X \text{ nor } \text{not} Y) \\
X \text{ nor } Y &= \text{not} (\text{not} X \text{ nand } \text{not} Y)
\end{align*}
\]
Boolean algebra

- An algebraic structure consists of
  - a set of elements $B$
  - binary operations $\{+, \cdot\}$
  - and a unary operation $\{\prime\}$
  - such that the following axioms hold:

1. the set $B$ contains at least two elements: $a, b$
2. closure: $a + b \in B$, $a \cdot b \in B$
3. commutativity: $a + b = b + a$, $a \cdot b = b \cdot a$
4. associativity: $a + (b + c) = (a + b) + c$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
5. identity: $a + 0 = a$, $a \cdot 1 = a$
6. distributivity: $a + (b \cdot c) = (a + b) \cdot (a + c)$, $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
7. complementarity: $a + a' = 1$, $a \cdot a' = 0$

George Boole – 1854

Logic functions and Boolean algebra

- Any logic function that can be expressed as a truth table can be written as an expression in Boolean algebra using the operators: $\prime$, $+$, and $\cdot$

$X, Y$ are Boolean algebra variables

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
<th>$X \cdot Y$</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>

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
<th>$X' + Y'$</th>
<th>$X' \cdot Y'$</th>
<th>$(X \cdot Y') + (X' \cdot Y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<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>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

$X = Y$

Boolean expression that is true when the variables $X$ and $Y$ have the same value and false, otherwise.
Axioms and theorems of Boolean algebra

- **identity**
  1. \( X + 0 = X \)
  1D. \( X \cdot 1 = X \)

- **null**
  2. \( X + 1 = 1 \)
  2D. \( X \cdot 0 = 0 \)

- **idempotency**
  3. \( X + X = X \)
  3D. \( X \cdot X = X \)

- **involution**
  4. \( (X')' = X \)

- **complementarity**
  5. \( X + X' = 1 \)
  5D. \( X \cdot X' = 0 \)

- **commutativity**
  6. \( X + Y = Y + X \)
  6D. \( X \cdot Y = Y \cdot X \)

- **associativity**
  7. \( (X + Y) + Z = X + (Y + Z) \)
  7D. \( (X \cdot Y) \cdot Z = X \cdot (Y \cdot Z) \)

- **distributivity**
  8. \( X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z) \)
  8D. \( X + (Y \cdot Z) = (X + Y) \cdot (X + Z) \)

- **uniting**
  9. \( X \cdot Y + X \cdot Y' = X \)
  9D. \( (X + Y) \cdot (X + Y') = X \)

- **absorption**
  10. \( X + X \cdot Y = X \)
  10D. \( X \cdot (X + Y) = X \)
  11. \( (X + Y') \cdot Y = X \cdot Y \)
  11D. \( (X \cdot Y') + Y = X + Y \)

- **factoring**
  12. \( (X + Y) \cdot (X' + Z) = X \cdot Z + X' \cdot Y \)
  12D. \( X \cdot Y + X' \cdot Z = (X + Z) \cdot (X' + Y) \)

- **concensus**
  13. \( (X \cdot Y) + (Y \cdot Z) + (X' \cdot Z) = X \cdot Y + X' \cdot Z \)
  13D. \( (X + Y) \cdot (Y + Z) \cdot (X' + Z) = (X + Y) \cdot (X' + Z) \)

- **de Morgan’s**
  14. \( (X + Y + \ldots)' = X' \cdot Y' \cdot \ldots \)
  14D. \( (X \cdot Y \cdot \ldots)' = X' + Y' + \ldots \)

- **generalized de Morgan’s**
  15. \( f'(X_1, X_2, \ldots, X_n, 0, 1, +, \cdot) = f(X_1', X_2', \ldots, X_n', 1, 0, \cdot, +) \)
Axioms and theorems of Boolean algebra (cont’d)

- **Duality**
  - a dual of a Boolean expression is derived by replacing
    \( \cdot \) by \(+\), \( + \) by \( \cdot \), \( 0 \) by \( 1 \), and \( 1 \) by \( 0 \), and leaving variables unchanged
  - any theorem that can be proven is thus also proven for its dual!
  - a meta-theorem (a theorem about theorems)

- **duality:**
  16. \( X + Y + \ldots \leftrightarrow X \cdot Y \cdot \ldots \)

- **generalized duality:**
  17. \( f(X_1,X_2,\ldots,X_n,0,1,+,\cdot) \leftrightarrow f(X_1,X_2,\ldots,X_n,1,0,\cdot,+) \)

- **Different than deMorgan’s Law**
  - this is a statement about theorems
  - this is not a way to manipulate (re-write) expressions

---

**Proving theorems (rewriting)**

- **Using the laws of Boolean algebra:**
  - e.g., prove the theorem: \( X \cdot Y + X \cdot Y' = X \)
    - distributivity (8)
    - complementarity (5)
    - identity (1D)
  - \( X \cdot Y + X \cdot Y' = X \cdot (Y + Y') \)
  - \( X \cdot (Y + Y') = X \cdot (1) \)
  - \( X \cdot (1) = X \)

  - e.g., prove the theorem: \( X + X \cdot Y = X \)
    - identity (1D)
    - distributivity (8)
    - identity (2)
    - identity (1D)
  - \( X + X \cdot Y = X \cdot 1 + X \cdot Y \)
  - \( X \cdot 1 + X \cdot Y = X \cdot (1 + Y) \)
  - \( X \cdot (1 + Y) = X \cdot (1) \)
  - \( X \cdot (1) = X \)
Activity

- Prove consensus theorem using the laws of Boolean algebra:
  - \((X \cdot Y) + (Y \cdot Z) + (X' \cdot Z) = X \cdot Y + X' \cdot Z\)

<table>
<thead>
<tr>
<th>Identity</th>
<th>Complementarity</th>
<th>Distributivity</th>
<th>Commutativity</th>
<th>Factoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>((X \cdot Y) + (Y \cdot Z) + (X' \cdot Z))</td>
<td>((X \cdot Y) + (X' + X) \cdot (Y \cdot Z) + (X' \cdot Z))</td>
<td>((X \cdot Y) + (X' \cdot Y \cdot Z) + (X \cdot Y \cdot Z) + (X' \cdot Z))</td>
<td>((X \cdot Y) + (X \cdot Y \cdot Z) + (X' \cdot Y \cdot Z) + (X' \cdot Z))</td>
<td>((X \cdot Y) \cdot (1 + Z) + (X' \cdot Z) \cdot (1 + Y))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Null</th>
<th>Identity</th>
</tr>
</thead>
<tbody>
<tr>
<td>((X \cdot Y) \cdot (1) + (X' \cdot Z) \cdot (1))</td>
<td>((X \cdot Y) + (X' \cdot Z))</td>
</tr>
</tbody>
</table>

Proving theorems (perfect induction)

- Using perfect induction (complete truth table):
  - e.g., de Morgan's:
    - \((X + Y)' = X' \cdot Y'\)
      - NOR is equivalent to AND with inputs complemented
    - \((X \cdot Y)' = X' + Y'\)
      - NAND is equivalent to OR with inputs complemented
A simple example: 1-bit binary adder

- Inputs: A, B, Carry-in
- Outputs: Sum, Carry-out

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Cin</th>
<th>Cout</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<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>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<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>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Cout = $A' B' Cin + A' B Cin' + A B' Cin' + A B Cin$

$S = A' B' Cin + A' B Cin' + A B' Cin' + A B Cin$

W'10 CSE370 - I - Boolean Algebra