Reference: W:\Lib\MathCAD\Default\defaults.mcd

**Boolean algebra**

Signal processing with electronic circuits:

A) Directly, using Kirchhoff’s laws and IV characteristics
   - e.g. addition, subtraction, gain, integration, log, solve differential equations, ...
   - direct interface to sensor
   - relatively simple circuits
   - fast, low power (but only for relatively low "accuracy")
   - very problem specific (limited configurability with potentiometers, switches, ...)
   - limited accuracy (typically <16 Bits, see later)
   - no good memory

B) Digital computation
   - wires carry few limited (usually 2) values
   - highly configurable (often programmable)
   - arbitrary accuracy, immune to "noise"
   - good memory
   - maps to very efficient electronic circuit implementations
   - cannot deal with analog sensor outputs
   - slow / high power for demanding applications
     (moving targets, e.g. digital video now possible, was not 10+ years ago)

C) Invent your own
   - objectives
     - solve interesting problems
     - efficient implementation
   - e.g.
     - molecular computer
     - quantum computer

**Digital computation - Boolean algebra**

Digital computers can add, subtract, multiply ... but not directly. They realize these functions from simpler elements that perform boolean functions.

Why? These boolean functions map to efficient electronic circuits.

**Boolean variables:**

only 2 levels, usually called 0 and 1

Physical implementations: e.g.
- smoke signal; semaphore (old train signals)
- VDD / VSS for 1 and 0 <- most current digital computers use this scheme
- positive / negative current (industrial control)
- light / dark (fiberoptics)
- ...

A signal allowing only 2 choices (e.g. 0/1) carries **1 Bit of information.**

**Example:**
- 1-Bit memory stores 1 in 2 possibilities
- 8-Bit memory stores 1 in $2^8 = 256$ choices, e.g. 0 ... 255 or 256 colors, ...
**Binary numbers:**

<table>
<thead>
<tr>
<th>Binary</th>
<th>1011</th>
<th>is decimal</th>
<th>(1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 11)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>13</td>
<td>convert to binary:</td>
<td>(\text{mod}(13, 2) = 1 \quad \text{trunc}\left(\frac{13}{2}\right) = 6)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(\text{mod}(6, 2) = 0 \quad \text{trunc}\left(\frac{6}{2}\right) = 3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(\text{mod}(3, 2) = 1 \quad \text{trunc}\left(\frac{3}{2}\right) = 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(\text{mod}(1, 2) = 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is binary</td>
<td>1101 (read mod from bottom)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>check:</td>
<td>(1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13)</td>
</tr>
</tbody>
</table>

**Boolean operations:**

a) logic NOT (inversion): \(y = \overline{x}\)

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

b) logic OR \(z = x + b\)

<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>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**boolean algebra!**

c) logic AND \(z = x \cdot y\)
combinations, symbols (bubble is inversion):

**AND**

\[
\begin{array}{c|cc}
    y & 0 & 1 \\
    \hline
    x & 0 & 0 \\
        & 1 & 1
\end{array}
\]

**OR**

\[
\begin{array}{c|cc}
    y & 0 & 1 \\
    \hline
    x & 0 & 1 \\
        & 1 & 1
\end{array}
\]

**XOR**

\[
\begin{array}{c|cc}
    y & 0 & 1 \\
    \hline
    x & 0 & 1 \\
        & 1 & 0
\end{array}
\]

Figure 1. Truth tables

**Figure 2. Logic gates**

**Figure 3. De Morgan equivalents**

**Figure 4. Venn diagrams**

**Example:** arithmetic (addition) with boolean algebra

binary 1-bit adder slice ... chain \(N\) for \(N\)-bit adder

<table>
<thead>
<tr>
<th>inputs</th>
<th>outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)</td>
<td>(B)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
S = \overline{A} \cdot \overline{B} \cdot C_{in} + \overline{A} \cdot B \cdot \overline{C_{in}} + A \cdot \overline{B} \cdot \overline{C_{in}} + A \cdot B \cdot C_{in}
\]

\(C_{out} = \ldots\)  

circuit diagram ... simplify? (use computer)