Overview

- You have read in “Introduction to Logic Design”
  - Chapter 1: Introduction excluding number systems
  - Chapter 2: Boolean algebra / Combinational Circuits

- First part:
  - Information and digital abstraction
  - Combinational devices and networks
  - Implementation

- Second part:
  - Boolean algebra
  - Proof techniques
  - Switching algebra

Information

- For example: voltage (s)
  - Continuous level
    - Between 0.23 and 0.28 becomes 0.25 V
  - Discrete level
    - Either 0.0 or 2.5 V
  - Binary level

How much information?

- Definition: The amount of information is defined to be
  the base 2 logarithm of all (equally probable) possibilities

- Example: Voltage (s)
  - Continuous level
    - Log₂(0.6713456278) = 3.46 bits
  - Discrete level
    - Log₂(0.25) = 2 bits
  - Binary level
    - Log₂(2) = 1 bit
representation of discrete variables

example:

to distinguish 128 characters for printing (ascii)
we need \( \log_2{128} \) = 7 bits

- measure 128 different voltages
- send serially 7 binary levels + synchronisation
- send parallel 7 binary levels + sync-line

digital abstraction

the reliable translation between a discrete variable and
the approximate value of a continuous variable (voltages)

digital abstraction

a combinational device is an element that has

- one or more input terminals for binary valued signals
- one output terminal for a binary valued signal
- a functional specification, detailing the value at the output
  for each possible combination of input values
- a timing specification, containing at least an upper bound on the evaluation delay,
  that is, the maximum time needed by the device
to produce a valid output for any combination of inputs

gates

a gate is a combinational device

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

functional specification

transfer characteristic

captures the static behavior

diagram of a circuit in response

to input states, i.e. the output
after long enough waiting!

dynamic behavior is the
output over time in response to an input change

logic 0 input voltage

logic 1 input voltage

but this is an abstraction!

combinational devices

more complex combinational devices

can be built from simpler devices

a combinational composition:

a network is a combinational device

- each node is itself a combinational device
- every arc (directed edge) either starts from a designated
  input or from exactly one output terminal of a device
- the circuit contains no directed cycles
**combinational devices**

More complex combinational devices can be built from simpler devices. A combinational composition:

- Each node is itself a combinational device.
- Every arc (directed edge) either starts from a designated input or from exactly one output terminal of a device.
- The circuit contains no directed cycles.

A network is a combinational device if it consists of interconnected devices such that:

- Each node is itself a combinational device.
- Every arc (directed edge) either starts from a designated input or from exactly one output terminal of a device.
- The circuit contains no directed cycles.

Combinational devices can be hierarchies.

**number of devices**

How many different combinational devices do exist with \( n \) input terminals?

- For one input terminal: the truth table has 2 rows for 2 values each \( \rightarrow 4 \)
- For two input terminals: the truth table has 4 rows for 2 values each \( \rightarrow 16 \)
- For three input terminals: the truth table has 8 rows for 2 values each \( \rightarrow 256 \)
- For \( n \) input terminals: the truth table has \( 2^n \) rows for 2 values each \( \rightarrow 2^{2^n} \)

Ignores timing and implementation specifics.

**exponential growth**

- Truth table for 64 variables (e.g., an adder for two 32 bit numbers)
- Computer processing 1 billion lines per second (approximately today’s technology)
- 500+ years needed for processing the table !!!

**MOS transistors / switching networks**

<table>
<thead>
<tr>
<th>nMOS transistor</th>
<th>pMOS transistor</th>
<th>complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>gate</td>
<td>gate</td>
<td></td>
</tr>
<tr>
<td>drain</td>
<td>source</td>
<td></td>
</tr>
<tr>
<td>source</td>
<td>drain</td>
<td></td>
</tr>
</tbody>
</table>

When voltage between gate and source exceeds threshold, the transistor is open; otherwise it is closed. Switch!

Switching networks:

- Combination is conducting if \( a \) AND \( b \) is high.
- Combination is conducting if \( a \) OR \( b \) is high.
- Combination is conducting if \( a \) is high AND \( b \) is low.
gate implementations

overview

- you have read in “introduction to logic design”
  - chapter 1: introduction excluding number systems
  - chapter 2: boolean algebra / combinational circuits

- first part:
  - information and digital abstraction
  - combinational devices and networks
  - implementation

- second part:
  - boolean algebra
  - proof techniques
  - switching algebra

algebraic structures

an algebraic structure is a set over which operations are defined

- properties are fixed by postulating axioms
- the system of axioms should be preferably minimal, consistent and complete
- axioms are assumed; theorems have to be proven
- proof techniques
  - producing a counterexample
  - (direct) derivation
    - a sequence of steps that are (applications of)
      either axioms or already derived theorems
  - reduction to equality
  - reduction to absurdity
    - assume the contrary and derive a contradiction
  - exhaustive enumeration
  - induction
    - prove a small case, and prove the case \( n \) by assuming the case \( n-1 \)

George Boole and Claude E. Shannon

- 1847: G.Boole, "Mathematical analysis of logic"
- 1854: G.Boole, "An investigation in the laws of thought"
- 1854: A. De Morgan, "Formal logic"
- 1904: E.V. Huntington, "Sets of independent postulates for the algebra of logic"
- 1910: P. Ehrenfest, "Review of L.Couturat's L'algebre de la logique"
- 1938: C.E. Shannon, "A symbolic analysis of relay and switching circuits"
A set $B$ with two operations $+$ and $\cdot$ is a boole algebra iff:

- it has at least two distinct elements
- the set $B$ is closed under $+$ and $\cdot$
- both operations have an identity element
- both operations are commutative
- each element has a complement
- each operation distributes over the other

**Axioms:**

- $\exists x, y \in B \left[ x \neq y \right]$
- $\forall x, y \in B \left[ x \cdot y = y \cdot x \right]$
- $\forall x, y, z \in B \left[ (x \cdot y) + (x \cdot z) = x \cdot (y + z) \right]$
- $\forall x \in B \left[ x + x = x \right]$
- $\forall x \in B \left[ x \cdot 1 = x = 1 \cdot x \right]$
- $\forall x \in B \left[ x + 0 = x = 0 + x \right]$

**Proof techniques:** derivation

- $x \cdot x = x$ to prove: $x + x = x$
- $\exists x, y \in B \left[ x \neq y \right]$ proof: $x = x + 0$
- $\forall x, y \in B \left[ x \cdot y = y \cdot x \right]$ $\Rightarrow$ $x = (x + x) \cdot (x + x')$
- $\forall x, y, z \in B \left[ (x + y) + (x \cdot z) = x + (y + z) \right]$

**Theorems:**

- $\forall x, y, z \in B \left[ x + z = x + (y + z) \right]$
- $\forall x, y, z \in B \left[ x \cdot (y \cdot z) = (x \cdot y) \cdot (x \cdot z) \right]$
proof techniques: reduction to absurdity

axioms:

\[ \exists_{x,y \in B}[x \neq y] \]

prove that each operand has exactly one identity

\[ \forall_{x,y \in B}[x \cdot y = B \land x + y \in B] \]

proof: suppose \( l_1 \) and \( l_2 \) are both identities for \( \cdot \) and \( l_1 \neq l_2 \)
then

\[ l_2 \cdot l_1 = l_2 = l_1 \cdot l_2 \]

\[ l_1 \cdot l_2 = l_2 = l_2 \cdot l_1 \]

\( l_1 = l_2 \)

contradiction!!!!

\[ \forall_{x \in B} \exists_{x' \in B}[x + x' = 1 \land x \cdot x' = 0] \]

\[ \forall_{x,y \in B}[x \cdot (y + z) = (x \cdot y) + (x \cdot z)] \]

\[ \forall_{x,y \in B}[x + (y \cdot z) = (x + y) \cdot (x + z)] \]

duality

the dual of a proposition concerning a boole algebra is the proposition obtained by exchanging the operators and exchanging the identity elements

we prove \( x \cdot 0 = 0 \) by derivation as follows:
\[
\begin{align*}
  x \cdot 0 &= (x \cdot 0) + 0 = (x \cdot 0) + (x \cdot 0') \\
  &= (x \cdot 0') + (x \cdot 0) = x \cdot (x' + 0) = x \cdot x' = 0 \text{ which gives us by duality: } x + 1 = 1
\end{align*}
\]

we prove one absorption law:

to prove: \( x \cdot (x + y) = x \)

proof:
\[
\begin{align*}
  x \cdot (x + y) &= (x + 0) \cdot (x + y) = x + (0 \cdot y) = x + 0 = x
\end{align*}
\]

then, by duality, we also have: \( x + (x \cdot y) = x \)
is associativity an axiom?
preferably not, because it can be derived from the other axioms!
lemma: \( \forall_{a,b,c \in B}[b \cdot a = c \cdot a \land b \cdot a' = c \cdot a' \rightarrow b = c] \)
proof: \( b = b \cdot 1 = b \cdot (a + a') = (b \cdot a) + (b \cdot a') = (c \cdot a) + (c \cdot a') = c \cdot (a + a') = c \)

use the premisses

**Switching algebra**

if \(|B| = 2\), then \(B\) has to be \(\{0, 1\}\), with of course \(0' = 1\) and \(1' = 0\)
and further, by the identity axioms:
\(x + y = 1 \iff x = 1 \lor y = 1\)
\(x \cdot y = 1 \iff x = 1 \land y = 1\)

note:

**CHECK THE AXIOMS!!!**

the boole algebra with set cardinality 2 is called switching algebra

the model

switching algebra is a boole algebra, because

<table>
<thead>
<tr>
<th>(x \in B)</th>
<th>(y \in B)</th>
<th>(\neg x \in B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>(0)</td>
<td>(x)</td>
</tr>
<tr>
<td>(\neg x)</td>
<td>(x)</td>
<td>(1)</td>
</tr>
<tr>
<td>(\neg x)</td>
<td>(\neg x)</td>
<td></td>
</tr>
</tbody>
</table>
the model

switching algebra is a boole algebra, because

<table>
<thead>
<tr>
<th>left</th>
<th>x+y</th>
<th>x+z</th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>y•z</th>
<th>right</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>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</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>1</td>
<td>1</td>
<td>1</td>
<td>0</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>0</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>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

— distributivity

switching algebra

if |B|=2, then B has to be \{0,1\}, with of course 0' = 1 and 1' = 0

and further, by the identity axioms:

\[ x + y = 1 \iff x = 1 \vee y = 1 \]
\[ x \cdot y = 1 \iff x = 1 \wedge y = 1 \]

CAUTION:

these two properties are only valid in a 2-element boole algebra

switching algebra is an adequate formal system for analyzing and manipulating combinational networks

the operations •, + and ' of a 2-element boole algebra have to correspond to an AND, OR and NOT gate respectively

proof techniques: exhaustive enumeration

proofs by exhaustive enumeration consist of checks for all cases
e.g. the de morgan laws for switching algebra: \((x + y)' = x' \cdot y'\)

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>x+y</th>
<th>(x+y)'</th>
<th>x</th>
<th>y</th>
<th>x•y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

by duality we immediately have the other de morgan law!

useful when there is a small number of cases to check and for many base cases for inductive proofs

proofs by induction

we prove one of the de morgan laws
(and obtain the other one by the duality principle)

to prove: \(\left(\sum_{i=1}^{n} x_i\right)' = \prod_{i=1}^{n} (x_i)'\)

already proven: \(\left(\sum_{i=1}^{2} x_i\right)' = \prod_{i=1}^{2} (x_i)'\)

(base case)

(Ind. Hypothesis)

assumed: \(\left(\sum_{i=1}^{n-1} x_i\right)' = \prod_{i=1}^{n-1} (x_i)'\)

by definition

proof: by definition

base case

\(\left(\sum_{i=1}^{n} x_i\right)' = \left(\sum_{i=1}^{n-1} x_i\right)' + x_n\)

(Induction Hypothesis)

shorthand for repeated application of the + operator

\[\left(\sum_{i=1}^{n} x_i\right)' = \prod_{i=1}^{n} (x_i)'\]

shorthand for repeated application of the • operator

\[\prod_{i=1}^{n} x_i = \sum_{i=1}^{n} (x_i)\]