

Computer Arithmetic

Written by HADIOUCHE Azouaou.

Disclaimer

This textbook follows the courses of Dr. OUDJIDA, some alternations of the course like additions and rearrangements occur for clarity, simplicity or personal preferences.

To separate the contents of the course to actual additions or out of context information, a black band will be added by its side like the one enclosing this comment.

USE IT AT YOUR OWN RISK!

Contents

Chapter: Classical logic	2
Boolean Algebra	2
Logic Gates & Digital Circuits	3
Transistors	4
Circuit Simplification & Reduction Methods	4
Chapter: Arithmetics	5
Numbers & Number Representations	5

Chapter 1

Classical logic

The most basic part of executing a computation on a machine is to describe the most basic information, which is true/false, and then compose them into a statement or a proposition.

informally, given a sentence, it is said to be a *statement* if

- its declarative, either affirmative or negative.
- its possible truth values are true or false.
- its verifiable in reality.

on those statements, we have some rules to give them truth values

- law of identity: $A = A$ is a true statement.
- law of non-contradiction: $\neg(A \wedge \neg A)$ is false statement.
- law of excluded middle: either $\neg A$ or A is true statement.

Now we will formalize calculations on boolean variables which is known as Boolean algebra, it will help us analyse and create circuits later on and also simplify them to have less components.

1.1. Boolean Algebra

Let $\mathbb{B} := \{0, 1\}$ denote the set of boolean values, which can be represented too with true/false. Any variable ta

Definition 1.1.1 (Boolean Variable): Let x be a variable, x is said to be a boolean variable if it can assume values in \mathbb{B} . Let $f : \mathbb{B}^n \rightarrow \mathbb{B}$ a map, f is called a boolean function.

Boolean functions will be the main study of Boolean algebra, how they can be written, expressed and modified without altering its values, the following operations will be useful for operating on boolean variables and construct functions.

Definition 1.1.2 (Boolean Operations): Let x, y be two boolean variables, we define the operations $+, \cdot, \neg$ to be the logical or, and, not respectively, which have the following truth tables.

y	x	\bar{x}	$x + y$	$x \cdot y$
0	0	1	0	0
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1

there are some other operations that are as follows

- $x \mid y = \bar{x} \cdot \bar{y}$.
- $x \otimes y = x \cdot \bar{y} + \bar{x} \cdot y$.
- $x \Rightarrow y = \bar{x} + y$.

Proposition 1.1.3 (Boolean Identities):

$\bar{\bar{x}} = x$	
$x + x = x$	$x \cdot x = x$
$x + 0 = x$	$x \cdot 1 = x$
$x + 1 = 1$	$x \cdot 0 = 0$
$x + y = y + x$	$x \cdot y = y \cdot x$
$x + (y + z) = (x + y) + z$	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
$x(y + z) = xz + yz$	$x + yz = (x + y) \cdot (x + z)$
$\bar{x + y} = \bar{x} \cdot \bar{y}$	$\bar{x \cdot y} = \bar{x} + \bar{y}$
$x + \bar{x} = 1$	$x \cdot \bar{x} = 0$

Definition 1.1.4 (Duality): Let $f : \mathbb{B}^n \rightarrow \mathbb{B}$ be a boolean function, we define the dual of f as the map $(x_1, \dots, x_n) \mapsto \overline{f(\bar{x}_1, \dots, \bar{x}_n)}$. We can obtain the dual of a function f by swapping $+$ with $\cdot, 0$ with 1 and keep the variables unchanged.

Definition 1.1.5 (Literal/Minterm/Maxterm): Let x_1, \dots, x_n be boolean variables and y_1, \dots, y_n such that $\forall i \in \llbracket 1, n \rrbracket, y_i = x_i \vee y_i = \overline{x_i}$.

- A literal is a proposition in the form of x or \overline{x} with x a boolean variable.
- A minterm of x_1, \dots, x_n is the product $y_1 \cdot y_2 \cdots y_n$.
- A maxterm of x_1, \dots, x_n is the sum $y_1 + y_2 \cdots + y_n$.

Definition 1.1.6 (Conjunctive/Disjunctive Normal Form): Let $f : \mathbb{B}^n \rightarrow \mathbb{B}$ be a boolean function, x_1, \dots, x_n boolean variables.

- DNF: $f(x_1, \dots, x_n) = \sum_{i=1}^k X_i$ where X_i are minterms.
- CNF: $f(x_1, \dots, x_n) = \prod_{i=1}^k X_i$ where X_i are maxterms.

Proposition 1.1.7:

1. the dual of a DNF is a CNF and vice versa.
2. Every boolean function can be written with only the defined connectives.
3. Every boolean function can be expressed only using one of those sets $\{+, -\}, \{\cdot, -\}, \{| \}$, we call them a complete set of connectives.
4. Any boolean function can be written in the CNF or DNF.

Proof.

1. A minterm is of the form $y_1 \dots y_n$ then its dual is $y_1 + \dots + y_n$ which is a maxterm, now if f is in a DNF then it is the sum of minterms, the dual will become a product of maxterms which is a CNF, its easy to verify the rest.
2. Let $f : \mathbb{B}^n \rightarrow \mathbb{B}$ a boolean function, we define $g : \mathbb{B}^n \rightarrow \mathbb{B}$ as follows $(x_1, \dots, x_n) \mapsto \sum_{i=1}^{2^n} \sigma_i(x_1, \dots, x_n) \cdot f(x_1, \dots, x_n)$ where $\sigma_i(x_1, \dots, x_n)$ is defined as if we take the i in base 2, $i = i_n i_{n-1} \dots i_1 i_0$ then $\sigma_i(x_1, \dots, x_n) = y_1 \dots y_n$ and if $i_j = 1$ then $y_j = x_j$ otherwise $y_j = \overline{x_j}$. Notice that $\sigma_i(x_1, \dots, x_n) = 1$ if and only if $\overline{x_n \dots x_1}^2 = i$. So we have for $(x_1, \dots, x_n) \in \mathbb{B}^n, g(x_1, \dots, x_n) = f(x_1, \dots, x_n)$ thus f can be written only with the connectives.
3. To prove this statement, we just need to use the fact that any function can be written using only $+, \cdot, -$.

- We have by De Morgans laws that $x \cdot y = \overline{\overline{x} + \overline{y}}$ thus $+, -$ is enough to express every function.

- Same can be used to express $x + y = \overline{\overline{x} \cdot \overline{y}}$.

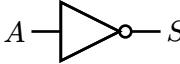
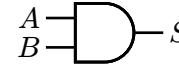
- Now we can use the NAND to write everything, notice that $x|x = \overline{x}$ and $(x|y)|(x|y) = x \cdot y$ thus we use the previous statement and we get that $\{| \}$ is a complete set of connectives.

4. Notice that in the first statement we proved that any boolean function can be written in the DNF, using the same function σ_i we can construct a DNF, it will be of the form $g : (x_1, \dots, x_n) \mapsto \prod_{i=1}^{2^n} \overline{f(x_1, \dots, x_n)} \cdot \overline{\sigma_i(x_1, \dots, x_n)}$.

□

1.2. Logic Gates & Digital Circuits

To be able to use and/or/not in circuits, we introduce the most basic circuit components called logic gates, as the table below shows

Not	And	Or					
							
A	S	A	B	S	A	B	C
0	1	0	0	0	0	0	0
1	0	0	1	1	0	1	0
		1	0	1	1	0	0
		1	1	1	1	1	1

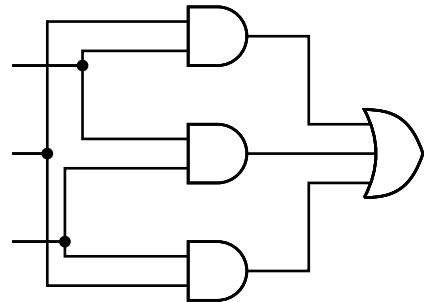
Thus we can use these to represent some circuits that behave as logical circuits called digital circuits.

Example:

- A committee of n individuals decide issues for an organization. Each individual votes either yes or no for each proposal that arises. The proposal is passed if it receives at least p votes. Its easy to notice that the solution is equal to

$$y = \sum_{1 \leq i_1 < \dots < i_p \leq n} x_{i_1} x_{i_2} \dots x_{i_p}$$

where x_1, \dots, x_n represent the votes of the individuals and y the proposal passing. In case, $n = 3$ and $p = 2$ we get



1.3. Transistors

One of the biggest advancements in our modern world is the creation of a transistor, in principle the idea is simple, a transistor is simply an electrically controlled valve. We use this to materialize

1.3.1. Circuit Simplification & Reduction Methods

- 1. Karnaugh Maps
- 2. Quine-McCluskey Method

Chapter 2

Arithmetics

Computer arithmetic is the process of using algorithms for doing basic operations on numbers like addition multiplication... etc. To be able to do those operations, a representation of the numbers is needed, which will be the first part of the course.

2.1. Numbers & Number Representations

Multiple numeral systems were developed