INF2270 — Spring 2011

Lecture 1: Digital Representation

UNIVERSITETET I OSLO

content

Background

Binary Numbers

Boolean Algebra

content

Background

Binary Numbers

Boolean Algebra

Digital Electronics

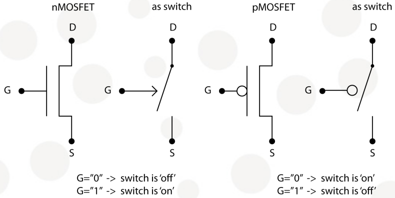
- Digital electronics is everywhere: PCs, cell phones, DVD and MP3 players, cameras, cars, fridges ...
- 'Digital' comes from latin 'digitus' meaning 'finger' which we often use for discrete counting
- Digital representation: discontinuous/discrete representation (discrete numbers/integers/$\mathbb{Z}$) as opposed to analog/continuous

A Short History of Computers

- The Analytical Engine by Charles Babbage in 1837 (mechanical)
- G. Stibitz' Model-K (1937) and K. Zuse's Z3 (1941) (electro mechanical, relays)
- ENIAC, 1946 (vacuum tubes)
- Computer revolution with transistors, first used to build a computer in 1953 (ca. 600 transistors)

Transistors

- Transistors: electronic switches (connecting 'drain' and 'source') controlled by an electronic signal ('gate'-voltage)



  - Down to 28nm (0.000000028m) long, and hundreds of millions on a single integrated circuit. The Intel Core^{tm} i7 Quad Extreme (Bloomfield) has 731 millioner transistorer in a 45nm-technology (2008)
  - Transistors need in the order of hundreds of pico-second to switch

Binary Representation

- Switches obviously have two states: ’on’ and ’off’
- Thus they are suited to represent digital numbers in the binary system (rather than the decimal that we are used to)
- The two states can also be used to represent the two states in predicate logic/Boolean algebra: ’true’ and ’false’, respectively ’1’ and ’0’

content

Background

Binary Numbers

Boolean Algebra

Binary Numbers

'10010' as a binary number means:

$1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 16 + 2 = 18$

just as '18' as a decimal number means:

$1 \times 10^1 + 8 \times 10^0$

content

Background

Binary Numbers

Boolean Algebra

Boolean Algebra

- Boolean algebra/logic is a set of operations defined on variables that have only two values: ’0’ and ’1’
- There are three basic opeartoions: NOT, AND, OR
- NOT a can be written as $\neg a$, $\bar{a}$ or a’
- a AND b can be written as $a \wedge a$ or $a \times b$ (not to be mixed up with normal multiplication)
- a OR b can be written as $a \vee a$ or $a + b$ (not to be mixed up with normal addition)

Truth Tables

a	ā
0	1
1	0

a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

Derived Boolean Functions

$a \text{ XOR } b = (a \wedge \bar{b}) \vee (\bar{a} \wedge b)$

$a \text{ XNOR } b = (a \wedge b) \vee (\bar{a} \wedge \bar{b})$

$a \text{ NAND } b = \overline{a \wedge b}$

$a \text{ NOR } b = \overline{a \vee b}$

Rules that Apply

$a \wedge b \vee c = (a \wedge b) \vee c$	$a \vee b \wedge c = a \vee (b \wedge c)$	(priority)
$a \wedge b = b \wedge a$	$a \vee b = b \vee a$	(commutativity)
$(a \wedge b) \wedge c = a \wedge (b \wedge c)$	$(a \vee b) \vee c = a \vee (b \vee c)$	(associativity)
$\bar{\bar{a}} = a$		(involution)
$a \wedge \bar{a} = 0$	$a \vee \bar{a} = 1$	(completeness)
$a \wedge a = a$	$a \vee a = a$	(idempotency)
$a \wedge 1 = a$	$a \vee 0 = a$	(boundedness)
$a \wedge 0 = 0$	$a \vee 1 = 1$	(boundedness)
$a \wedge (a \vee b) = a$	$a \vee (a \wedge b) = a$	(absorbtion)
$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$	$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$	(distributivity)
$\overline{a \vee b} = \bar{a} \wedge \bar{b}$	$\overline{a \wedge b} = \bar{a} \vee \bar{b}$	(deMorgan)

Checking deMorgan's Theorem

for NOR:

a	b	$\overline{a \vee b}$
0	0	1
0	1	0
1	0	0
1	1	0

a	b	$\bar{a} \wedge \bar{b}$
0	0	1
0	1	0
1	0	0
1	1	0

Examples: simplify

$a \wedge b \vee a \wedge \bar{b} = ?$

$a \wedge b \wedge c \vee \bar{a} \wedge b \wedge c \vee \bar{a} \wedge b \wedge \bar{c} \wedge (a \vee c) = ?$

Solution 1

$$
\begin{aligned}
& a \wedge b \vee a \wedge \bar{b} \\
= \;& a \wedge (b \vee \bar{b}) \\
= \;& a \wedge 1 \\
= \;& a
\end{aligned}
$$

Solution 2

$$
\begin{aligned}
& a \wedge b \wedge c \vee \bar{a} \wedge b \wedge c && \vee && \bar{a} \wedge b \wedge \bar{c} \wedge (a \vee c) \\
= \;& (a \vee \bar{a}) \wedge b \wedge c && \vee && \bar{a} \wedge b \wedge \bar{c} \wedge a \;\vee\; \bar{a} \wedge b \wedge \bar{c} \wedge c \\
= \;& 1 \wedge b \wedge c && \vee && 0 \vee 0 \\
= \;& b \wedge c
\end{aligned}
$$

Completnes of NAND and NOR

Either the NAND or the NOR function are actually sufficient to implement any Boolean function

$$\bar{a} = \overline{a \wedge a} = \overline{a \vee a} \tag{1}$$

$$a \wedge b = \overline{\overline{a \wedge b}} = \overline{\bar{a} \vee \bar{b}} \tag{2}$$

$$a \vee b = \overline{\bar{a} \wedge \bar{b}} = \overline{\overline{a \vee b}} \tag{3}$$

Examples: representation with truth tables (1/2)

$F=(a \wedge b) \vee (\bar{a} \wedge \bar{b})$

a	b	F
0	0	1
0	1	0
1	0	0
1	1	1

Examples: representation with truth tables (2/2)

$$G = \underbrace{a \wedge b \wedge c}_{x_1} \vee \underbrace{\bar{a} \wedge c}_{x_2} \vee \underbrace{a \wedge \bar{b}}_{x_3}$$

a	b	c	x_1	x_2	x_3	G
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	0	0	0	0
0	1	1	0	1	0	1
1	0	0	0	0	1	1
1	0	1	0	0	1	1
1	1	0	0	0	0	0
1	1	1	1	0	0	1

Representing Boolean Functions

1. expression with variables and operators
2. truth table
3. graphically with logic gates