

# INF1400 - Notes

Veronika Heimsbakk  
Department of Infomatics  
University of Oslo  
veronahe@student.matnat.uio.no

November 28, 2012

## Contents

<b>1</b>	<b>Note</b>	<b>2</b>
<b>2</b>	<b>Binary Numbers</b>	<b>2</b>
2.1	Number systems . . . . .	2
2.2	Number-base Conversions . . . . .	3
2.3	Compliment of Numbers . . . . .	4
2.4	Signed Binary Number . . . . .	4
2.5	Binary Storage and Registers . . . . .	5
2.6	Binary Logic . . . . .	5
2.7	Logic Gates . . . . .	5
<b>3</b>	<b>Boolean Algebra and Logic Gates</b>	<b>6</b>
3.1	Basic Definitions . . . . .	6
3.2	Boolean Functions . . . . .	6
3.3	Complement of a Function . . . . .	7
3.4	Minterms and Maxterms . . . . .	8
3.4.1	Conversion between Canonical Forms . . . . .	8
3.4.2	Standard Forms . . . . .	9
3.5	Other boolean expressions . . . . .	9
<b>4</b>	<b>Gate-Level Minimization</b>	<b>9</b>
4.1	Karnaugh Diagram . . . . .	11

<b>5</b>	<b>Combinational Logic</b>	<b>11</b>
5.1	Binary Adder . . . . .	12
5.1.1	Half Adder . . . . .	12
5.1.2	Full Adder . . . . .	13
5.2	Decoder . . . . .	13

## 1 Note

Note that encoders, latches and flip-flops are not described here.

## 2 Binary Numbers

Early computers were used for numeric computations, therefore the expression digital computers/system. Electric signals such as voltages and currents are the most common elements of how information are represented in a digital system. Transistors predominate in the circuiting that implement these signals. The signals are represented by two values, therefore binary. A binary digit, called bit, has two values: 0 and 1.

### 2.1 Number systems

The decimal number 7392 is a shorthand notation for:

$$7 \times 10^3 + 3 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$$

The binary number 11010.11 as shown from multiplication of the coefficients by powers of 2:

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 26.75_{10}$$

An octal number:

$$127.4_8 = 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1} = 87.5_{10}$$

A hexadecimal number:

$$B65F_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0 = 46687_{10}$$

In computer work there are some important numbers to remember:

A *byte* is equal to eight *bits* and can accommodate a keyboard character.

$2^{10}$	kilo	K
$2^{20}$	mega	M
$2^{30}$	giga	G
$2^{40}$	tera	T

Table 1: Some numbers to remember

Decimal base 10	Binary base 2	Octal base 8	Hexadecimal base 16
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Table 2: Numbers with different bases

## 2.2 Number-base Conversions

Decimal number: 41 is going to be converted to binary. Then we can read that the binary number for is:

$$41_{10} = a_5 \times a_4 \times a_3 \times a_2 \times a_1 \times a_0 = 101001_2$$

This by looking at the remainders (see Table 3). The first one is the least significant byte (LSB).

Binary numbers may be converted to hexadecimal number through grouping the binary number in groups of four.

$$10110001101011.1111 = 2C6B.F_{16}$$

This binary number gives 2C6B.F as hexadecimal number because 10 is 2, 1100 is 12 or C (since we are talking hex), 0110 is 6, 1011 is B and 1111 is

	Remainder	Coefficient
$41 : 2 = 20 + \frac{1}{2}$	$\frac{1}{2}$	$a_0 = 1$
$20 : 2 = 10 + 0$	0	$a_1 = 0$
$10 : 2 = 5 + 0$	0	$a_2 = 0$
$5 : 2 = 2 + \frac{1}{2}$	$\frac{1}{2}$	$a_3 = 1$
$2 : 2 = 1 + 0$	0	$a_4 = 0$
$1 : 2 = 0 + \frac{1}{2}$	$\frac{1}{2}$	$a_5 = 1$

Table 3: Conversion from decimal to binary

F. When we are grouping binary numbers, we always start with the most right group. Therefore is the first group only containing two characters.

The other way around is like this:

$$678.124_8 = 110111011.001010100$$

$$306.D_{16} = 001100000110.1101$$

For octadecimal numbers we have to group up by three and three.

## 2.3 Compliment of Numbers

The 1's compliment of the binary number 1011000 is 0100111. The 1's compliment of a binary number is formed by changing 1's to 0's and 0's to 1's. The compliment of the compliment restores the number to its original value.

## 2.4 Signed Binary Number

Ordinary sign for positive numbers is + and – for negative numbers. Since the computer only handles with 1's and 0's, there is another way of finding negative numbers. The sign is here bit 0 for positive numbers and 1 for negative ones. Take for example the number 9 and representing it with eight bits. Then +9 would be 00001001. This is the only way to represent the number +9. However, there are three ways of representing the number –9:

1. signed-magnitude representation: 10001001
2. signed-1's-magnitude representation: 11110110
3. signed-2's-magnitude representation: 11110111

With signed-2's-magnitude representation you just add a 1 to the inverted bits. For example  $-5$  ( $+5_{10} = 0101_2$ ):

$$\begin{array}{r} \text{Inverted } 5 \quad 1010 \\ +0001 \\ \hline -5 \quad \quad \quad = 1011 \end{array}$$

## 2.5 Binary Storage and Registers

A *binary cell* is a device that possesses two stable states and is capable of storing one bit (0 or 1) of information. A *register* is a group of binary cells. A register with  $n$  cells may store any discrete quantity of information that contains  $n$  bits. A register with 16 cells can be in one of  $2^{16}$  possible states.

To process discrete quantities of information in binary form, a computer must be provided with devices that hold the data to be processed and with circuit elements that manipulate individual bits of information. The device most common used for holding data is a register.

## 2.6 Binary Logic

There are three basic logical operations: AND, OR and NOT. Do not con-

		AND	OR	NOT	NOT
x	y	$x \cdot y$	$x + y$	$x'$	$y'$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	1
1	1	1	1	0	0

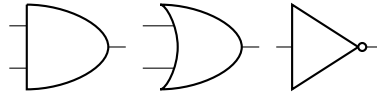
Table 4: Truth table for logic operations

fuse binary logic with binary arithmetic. A logic variable is either 1 or 0. For example in binary arithmetic, we have  $1 + 1 = 10$  ("one plus one is equal to two"), while binary logic, we have  $1 + 1 = 1$  ("one OR one is equal to one").

## 2.7 Logic Gates

Logic gates are electronic circuits that operate on one or more input signals to produce an output signal. These signals are voltages or currents which exist as analog signals having values e.g. 0 to 3 V. But in digital systems, analog signals are represented as 0 or 1.

Some gates may be:



This is AND, OR and NOT gates.

### 3 Boolean Algebra and Logic Gates

#### 3.1 Basic Definitions

A *set* of elements is any collection of objects, usually having a common property. E.g. if  $S$  is a set of elements, and  $x$  and  $y$  are objects. The notation  $x \in S$  means that  $x$  is a member of the set  $S$ . And  $y \notin S$  means that  $y$  is not a member of  $S$ . A set of elements is specified by braces:  $A = \{1, 2, 3, 4\}$ .

	<b>a</b>	<b>b</b>
Postulate 2	$x + 0 = x$	$x \cdot 1 = x$
Postulate 5	$x + x' = 1$	$x \cdot x' = 0$
Theorem 1	$x + x = x$	$x \cdot x = x$
Theorem 2	$x + 1 = 1$	$x \cdot 0 = 0$
Theorem 3, involution	$(x')' = x$	
Theorem 3, commutative	$x + y = y + x$	
Theorem 4, associative	$x + (x + z) = (x + y) + z$	$x(yz) = (xy)z$
Theorem 4, distributive	$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	$(x + y)' = x'y'$	$(xy)' = x' + y'$
Theorem 6, absorption	$x + xy = x$	$x(x + y) = x$

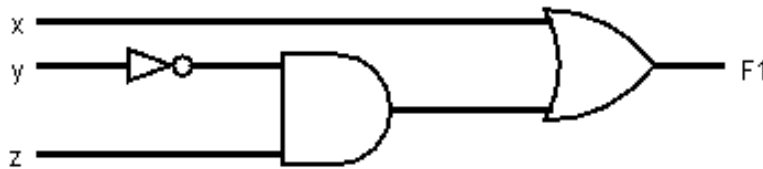
Table 5: Postulates and theorems

#### 3.2 Boolean Functions

Boolean algebra deals with binary variables and logical operators. The binary variables is either 1 or 0. Consider the function:  $F_1 = x + y'z$  The truth table for this function for this will be: The truth table for function  $F_1$  read as "x OR NOT y AND z". The number of rows in the truth table is  $2^n$ , where  $n$  is the number of variables in the function. A boolean function may be transformed into a circuit.

x	y	z	$F_1$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Table 6: Truth table for function  $F_1$



The schematic expresses the relationship between the output of the circuit and its inputs. When the function is in its algebraic form, it may be expressed in a variety of ways, all of which have equivalent logic.

By manipulating a boolean expression, it is sometimes possible to obtain a simpler expression for the same function, and then reduce the number of gates in the circuit and the number of inputs to the gate. For example:

$$F_2 = x'y'z + x'yz + xy'$$

May be reduced by doing this:

$$F_2 = x'y'z + x'yz + xy' = x'z(y' + y) + xy' = x'z + xy'$$

By means of a truth table it is possible to verify that the two expressions are equivalent

### 3.3 Complement of a Function

The compliment of a function  $F$  is  $F'$ . The compliment may be derived algebraically though DeMorgan's theorems. These DeMorgan's theorems can be generalized as follows:

$$(A + B + C + D + \dots + F)' = A'B'C'D' \dots F'$$

$$(ABC \dots F)' = A' + B' + C' + D' + \dots + F'$$

The generalized form of DeMorgan's theorems states that the complement of a function is obtained by interchanging AND and OR operations and complementing each literal.

### 3.4 Minterms and Maxterms

Since each variable may appear in either form, there are four possible combinations of  $x$  and  $y$ :  $x'y'$ ,  $x'y$ ,  $xy'$  and  $xy$ . Each of these four AND terms is called *minterm*, or a standard product.  $n$  variables may be combined to  $2^n$  minterms. Each minterm is obtained from an AND term of the  $n$  variables.

Maxterms is the  $n$  variables forming an OR term. With each variable being primed or unprimed, this provides  $2^n$  possible combinations of maxterms, or standard sums. Boolean functions expressed as a sum of

x	y	z	Minterm		Maxterm	
			Term	Designation	Term	Designation
0	0	0	$x'y'z'$	$m_0$	$x + y + z$	$M_0$
0	0	1	$x'y'z$	$m_1$	$x + y + z'$	$M_1$
0	1	0	$x'yz'$	$m_2$	$x + y' + z$	$M_2$
0	1	1	$x'yz$	$m_3$	$x + y' + z'$	$M_3$
1	0	0	$xy'z'$	$m_4$	$x' + y + z$	$M_4$
1	0	1	$xy'z$	$m_5$	$x' + y + z'$	$M_5$
1	1	0	$xyz'$	$m_6$	$x' + y' + z$	$M_6$
1	1	1	$xyz$	$m_7$	$x' + y' + z'$	$M_7$

Table 7: Minterms and maxterms for three variables

minterms or product of maxterms are said to be in canonical form.

#### 3.4.1 Conversion between Canonical Forms

The complement of a function expressed as the sum of minterms equals the sum of minterms missing from the original function. E.g. the function<sup>1</sup>:

$$F(A, B, C) = \sum(1, 4, 5, 6, 7)$$

This function has a complement that may be expressed as:

$$F'(A, B, C) = \sum(0, 2, 3) = m_0 + m_2 + m_3$$

<sup>1</sup> $\sum$  means sum and  $\prod$  means product.



If we took the complement of  $F'$  by DeMorgan's theorem, we obtain  $F$  in a different form:

$$F = (m_0 + m_2 + m_3)' = m'_0 \cdot m'_2 \cdot m'_3 = M_0 M_2 M_3 = \prod(0, 2, 3)$$

Therefore it is clear that following relation holds:

$$m'_j = M_j$$

### 3.4.2 Standard Forms

The *sum of products* is a boolean expression containing AND terms called product terms, with one or more literals each. The sum denotes the ORing of these terms.

A *product of sums* is a boolean expression containing OR terms, called sum terms. Each term may have any number of literals. The product denotes ANDing of these terms.

## 3.5 Other boolean expressions

Boolean function	Operator symbol	Name	Comment
$xy$	$x \cdot y$	AND	$x$ and $y$
$xy'$	$x/y$	Inhibition	$x$ , but not $y$
$x'y$	$y/x$	Inhibition	$y$ , but not $x$
$xy' + x'y$	$x \oplus y$	Exclusive-OR	$x$ or $y$ , but not both
$x + y$	$x + y$	OR	$x$ or $y$
$(x + y)'$	$x \downarrow y$	NOR	Not-OR
$xy + x'y'$	$(x \oplus y)'$	Equivalence	$x$ equals $y$
$x + y'$	$x \subset y$	Implication	If $y$ , then $x$
$x' + y$	$x \supset y$	Implication	If $x$ , then $y$
$(xy)'$	$x \uparrow y$	NAND	Not-AND

Table 8: Other operators

## 4 Gate-Level Minimization

*Gate-level minimization* is the design task of finding an optimal gate-level implementation.

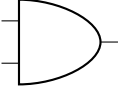
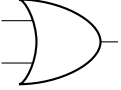
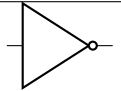
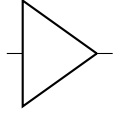
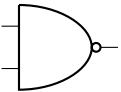
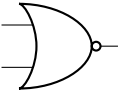
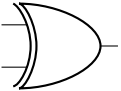
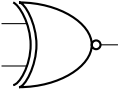
Name	Symbol	Function	Truth table		
AND		$F = x \cdot y$	x	y	F
			0	0	0
			0	1	0
			1	0	0
OR		$F = x + y$	x	y	F
			0	0	0
			0	1	1
			1	0	1
Inverter		$F = x'$	x	F	
			0	1	
			1	0	
			1	0	
Buffer		$F = x$	x	F	
			0	0	
			1	1	
			1	1	
NAND		$F = (xy)'$	x	y	F
			0	0	1
			0	1	1
			1	0	1
NOR		$F = (x + y)'$	x	y	F
			0	0	1
			0	1	0
			1	0	0
XOR		$F = xy' + x'y = x \oplus y$	x	y	F
			0	0	0
			0	1	1
			1	0	1
XNOR		$F = xy + x'y' = (x \oplus y)'$	x	y	F
			0	0	1
			0	1	0
			1	0	0
			1	1	1

Table 9: Logic gates

## 4.1 Karnaugh Diagram

This is a diagram made up of squares, with each square representing one minterm of the function that is to be minimized. K-diagram for the function:  $m_1 + m_2 + m_3 = x'y + xy' + xy = x + y$ :

		$y$	
		0001	
$x$	00	0	1
	01	1	1

In a three-variable K-diagram the minterms are arranged, not in a binary sequence, but in a sequence similar to the Grey code. The characteristic is that only one bit changes in value from one adjacent column to the next. For example, the square assigned to  $m_5$  corresponds to row 1 and column 01. When these two numbers are concatenated, they give the binary number 101, which is equivalent with the decimal number 5,  $m_5 = xy'z$ .

	00	01	11	10
0	$m_0$	$m_1$	$m_3$	$m_2$
1	$m_4$	$m_5$	$m_7$	$m_6$

Table 10: Three variable K-diagram

Any two adjacent squares differ by only one variable, which is primed in one square and unprimed in the other. For example  $m_5$  and  $m_7$ . Variable  $y$  is primed in  $m_5$  and unprimed in  $m_7$ .

$$m_5 + m_7 = xy'z + xyz = xz(y' + y) = xz$$

The two squares differ by the variable  $y$ , which can be removed when the sum of the two minterms is formed.

## 5 Combinational Logic

A combinational circuit consists of an interconnection of logic gates. For  $n$  variables, there are  $2^n$  possible combinations of the binary inputs. For each possible input combination, there is one possible value for each output variable. This can be illustrated with a truth table. The diagram of a combinational circuit has logic gates with no feedback paths or memory elements.

The design of combinational circuits starts from the specification of the design objective and culminates in a logic circuit diagram or set of boolean functions from which the logic diagram can be obtained. This is the design procedure:

1. Determine the required number of inputs and outputs and assign a symbol to each.
2. Derive the truth table that defines the required relationship between inputs and outputs.
3. Obtain the simplified boolean function for each output as a function of the input variables.
4. Draw the logic diagram and verify the correctness of the design (e.g. by simulation).

## 5.1 Binary Adder

A combinational circuit that preforms the addition of two bits is called a *half adder*, one that preforms this with three bits is a *full adder*.

### 5.1.1 Half Adder

This circuit needs two inputs and two outputs. Simplified boolean func-

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Table 11: Truth table for half adder

tions for the output variables:

$$S = x \oplus y$$

$$C = xy$$

S stands for sum and C stands for carry. The C output is 1 only when both inputs are 1. The sum output represent the least significant bit of the sum.

### 5.1.2 Full Adder

Addition of  $n$ -bit binary numbers requires the use of a full adder. The process proceeds on a bit-by-bit basis, right to left, beginning with the LSB. A full adder is a combinational circuit that forms the arithmetic sum of three bits. Three inputs and two outputs. Simplified functions:

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

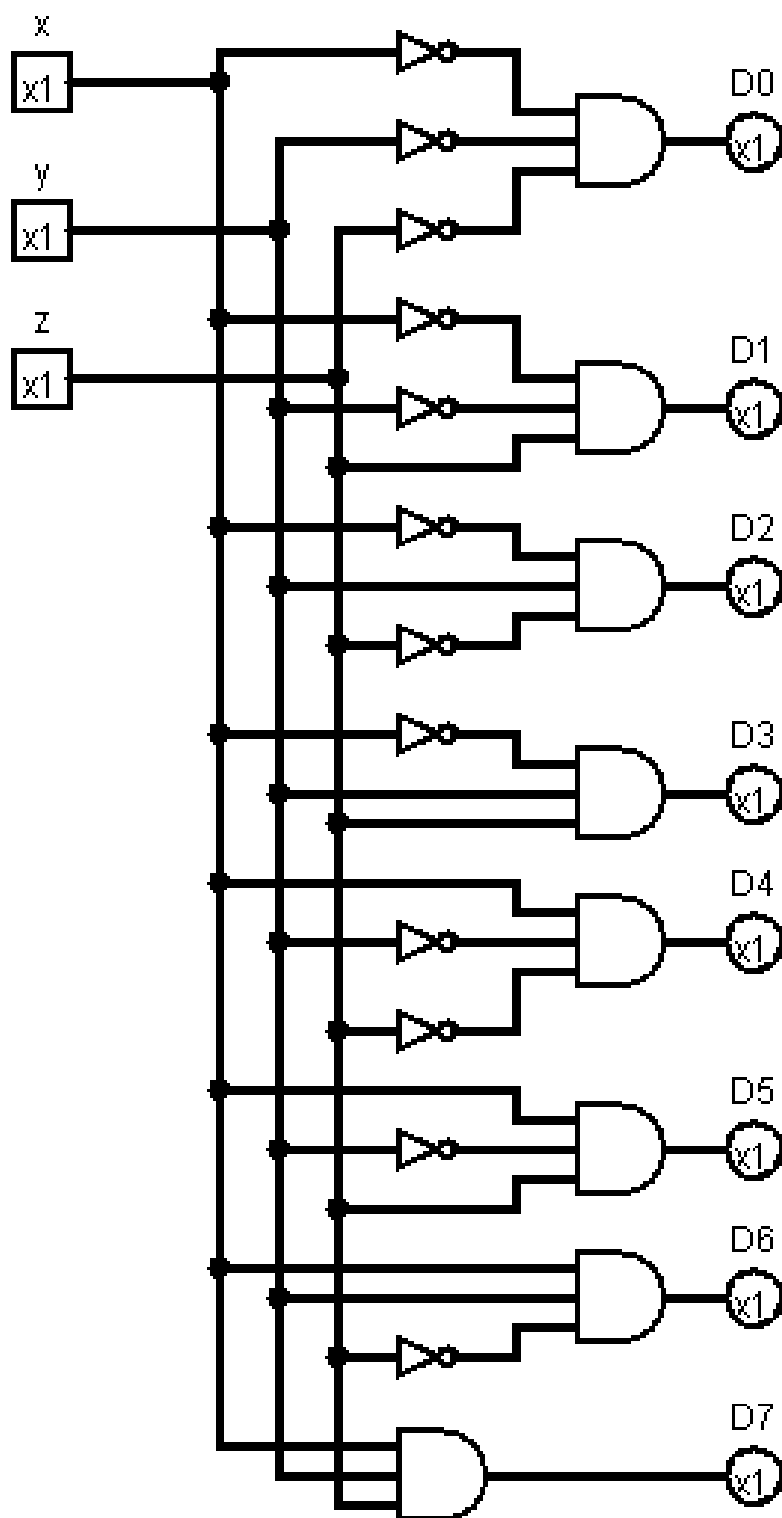
Table 12: Truth table for full adder

$$S = x'y'z + x'yz' + xy'z' + xyz$$

$$C = xy + xz + yz$$

## 5.2 Decoder

A decoder is a combinational circuit that converts binary information from  $n$  input lines to a maximum of  $2^n$  unique output lines. If the  $n$ -bit coded information has unused combinations, the decoder may have fewer than  $2^n$  outputs. With three inputs gives eight outputs - all minterms.



A decoder with enable input can function as a *demultiplexer* – a circuit that receives information from a single line and directs it to one of  $2^n$  possible output lines.

## List of Tables

1	Some numbers to remember . . . . .	3
2	Numbers with different bases . . . . .	3
3	Conversion from decimal to binary . . . . .	4
4	Truth table for logic operations . . . . .	5
5	Postulates and theorems . . . . .	6
6	Truth table for function $F_1$ . . . . .	7
7	Minterms and maxterms for three variables . . . . .	8
8	Other operators . . . . .	9
9	Logic gates . . . . .	10
10	Three variable K-diagram . . . . .	11
11	Truth table for half adder . . . . .	12
12	Truth table for full adder . . . . .	13

## References

- [1] Redaelli, Massimo A. *CircuitikZ* September 11, 2011. <ftp://ftp.dante.de/tex-archive/graphics/pgf/contrib/circuitikz/doc/latex/circuitikz/circuitikzmanual.pdf>, downloaded: November 22, 2012.
- [2] Ciletti, Michael D. and Mano, Morris M. *Digital Design, 5th ed.* Pearson Education Limited.