Computer Basics (COMPB 119) notes

Bachelor in Computer Science and Engineering

First year - First semester

Contents

1 Boolean Algebra and Logic Design

1.1 Analog vs. digital systems

Definition 1.1. An analog system is the one that works with analog signals. In this case, physical values are represented using analog signals, which are just infinite posible real values.

Definition 1.2. A digital system is the one that works with digital signals. Physical values are represented using digital signals that can take two possible values, which both are exclusive. A digital signal takes discrete values, in general, only 2.

Computers can't interpret and operate with analog signals, then, we should convert them to digital signals using an ADC (Analog to digital converter). To achieve this we can use the **binary numeral system** (discussed in the next section) to represent any number just with using two posible values.

1.2 Positional numeral systems

Definition 1.3. A positional numeral system is a system for representation of numbers by an ordered set of numeral symbols called digits whose value depends on its position in the actual number. For each position a unique symbol or a limited set of symbols is used.

Definition 1.4. In a positional numeral system, the radix, or base, is the number of unique digits, including the digit zero, used to represent numbers.

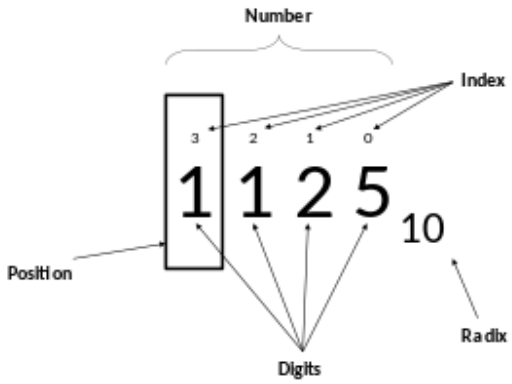

Figure 1: *Glossary of terms used in the positional numeral systems.*

In any standart positional numeral system, a number is conventionally written as $(x)_y$ with x as the string of digits and y as its base, although for the base 10 the subscript is usually assumed and omitted together with the pair of parenthesis.

Examples of commonly used positional numeral systems include the **decimal system** (base 10), the **binary system** (base 2) and the **hexadecimal system** (base 16).

Definition 1.5. In a positional numeral system with base b ($b > 1$), a string of digits $d_1 \dots d_n$ denotes the number $d_1 b^{n-1} + d_2 b^{n-2} + \dots + d_n b^0$, where $0 \leq d_i \leq b$.

In contrast to decimal, or base 10, which as a ones' place, tens' place, hundreds' place, and so on, base b would have a ones' place, then a b^1s' place, a b^2s' place, etc.

Proposition 1.6. A number with n digits with base b can represent b^n different numbers in the range $[0, b^n - 1]$.

1.2.1 Decimal numeral system

Definition 1.7. The decimal numeral system is a positional numeral system with base 10, meaning that it uses 10 digits from 0 to 9.

In decimal number system, the succesive positions to the left of the decimal point represents units, tens, hundreds, thousands and so on.

Each position represents a specific power of the base (10). For example, the decimal number 5374 consists of the digit 4 in the units position, 7 in the tens position, 3 in the hundreds position, and 5 in the thousands position, and by definition 1.5 its value can be written as

$$5 \cdot 10^3 + 3 \cdot 10^2 + 7 \cdot 10^1 + 4 \cdot 10^0 = 5000 + 300 + 70 + 4 = 5374 \tag{1}$$

Proposition 1.8. A number with n decimal digits from 0 to 9 can represent 10^n different numbers in the range $[0, 10^n - 1]$.

For example, when $n = 3$, $10^3 = 1000$ different numbers can be represented within the range $[0, 999]$.

1.2.2 Binary numeral system

Definition 1.9. The binary numeral system is a positional numeral system with base 2, which uses only two symbols: typically 0 and 1.

Each binary digit is referred to as a **bit**. A binary number made up of 4 binary digits, or 4 bits, receives the name **nibble**, and a group of two nibbles, or eight bits, defines what is called a **byte**.

Proposition 1.10. A number with n binary digits, 0 and 1, can represent 2^n different numbers in the range $[0, 2^n - 1]$.

For instance, when $n = 3$, $2^3 = 8$ different numbers can be represented within the range $[0, 7]$.

Because of its straightforward implementation in digital electronic circuitry using logic gates, the binary system is used by almost all modern computers and computer-based devices.

Example 1.1. Convert 10101_2 from binary to decimal.
By definition 1.5 we get

$$1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 1^0 = 16 + 4 + 1 = 21 \tag{2}$$

Example 1.2. Convert 47_{10} from decimal to binary.

$$47 \quad \% \quad 2 = 1 \quad \rightarrow \quad 23 \quad \% \quad 2 = 1 \quad \rightarrow \tag{3}$$
$$\rightarrow \quad 11 \quad \% \quad 2 = 1 \quad \rightarrow \tag{4}$$
$$\rightarrow \quad 5 \quad \% \quad 2 = 1 \quad \rightarrow \tag{5}$$
$$\rightarrow \quad 2 \quad \% \quad 2 = 0 \quad \rightarrow \quad 2/2 = 1 \tag{6}$$

Putting the last quotient and the remainders in inverse order we get that 47_{10} is 101111_2 in binary.

1.2.3 Hexadecimal numeral system

Definition 1.11. The hexadecimal system is a positional numeral system with base 16, commonly used to abbreviate binary numbers.

The different equivalences in the hexadecimal, decimal and binary systems are shown in the following table.

Hexadecimal digit	Decimal equivalent	Binary equivalent
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Table 1: *Hexadecimal, decimal and binary equivalences.*

Remark. Note that every hexadecimal digit is a nibble (it's made up of 4 bits).

Example 1.3. Convert $4AF_{16}$ (or $0x4AF$) from hexadecimal to binary.
By looking at table 1 we know that

$$4 = 0100 \qquad A = 1010 \qquad F = 1111$$

Then, the hexadecimal number $4AF$ is 010010101111 in binary.

Example 1.4. Convert $0x4AF$ from hexadecimal to decimal.
By definition 1.5 between number systems can be do the following operation:

$$\begin{aligned} 4AF &= 4 \cdot 16^2 + 10 \cdot 16^1 + 15 \cdot 16^0 = 4 \cdot 2^8 + 160 + 15 \\ &= 4 \cdot 256 + 160 + 15 = 1199 \end{aligned}$$

1.3 Basic boolean algebra properties and laws

Definition 1.12. Boolean algebra is a mathematical tool used for analysis and synthesis of binary digital systems. It's named after George Bool, a british mathematician from the 19^{th} century.

Definition 1.13. A boolean variable is a digital signal which only has 1 out of 2 possible values in an instant, which are mutually exclusive. Those values are usually represented as 0 and 1, but another representation could be OFF and ON, or HIGH and LOW.

1.3.1 Boolean expressions and operations. Logic functions and gates

Definition 1.14. A logic function is a circuit accepting at least one input logical value and outputs a logical value.

In order to describe the working principle of a logic function a **truth table** is used. In this table is specified all the posible outputs for all possible input logic values. Also it's a graphical representation of all cases that can happen in an algebraic relation and its respective results.

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

Table 2: *Example of a truth table. In this case is a XOR logic gate.*

Definition 1.15. Logic gates are the implementation of most basic logic functions.

Amplifier / BUFFER gate

Definition 1.16. The amplifier, or most commonly named, the BUFFER, is the simplest logic gate. It expects one input (A) and gives one output value (Z).

It follows the logic equation $Z = A$, meaning the input received by the buffer is not changed at all, and its corresponding truth table is the following one.

A	Z
1	1
0	0

Table 3: *Truth table for a BUFFER gate.*

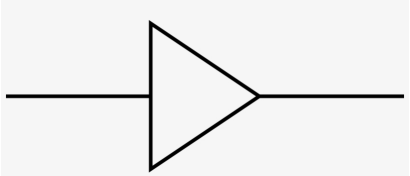

Figure 2: *Graphical representation of a BUFFER gate.*

NOT gate / Inverter

Definition 1.17. The NOT gate, also called an **inverter**, gets one input value (A) and gives the opposite output value (Z), following the logic equation $Z = \overline{A}$.

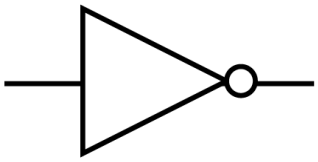

Figure 3: *Graphical representation of a NOT logic gate.*

A	Z
1	0
0	1

Table 4: *Truth table for a NOT gate.*

AND gate / Boolean multiplication

Definition 1.18. The AND gate gets at least two input values (A, B) and gives the one output value (Z). The output value Z will be true, $Z = 1$, if and only if both inputs A and B are simultaneously true.

The logic equation corresponding to the AND gate is $Z = A \cdot B$. Looking at this equation we could see the AND gate as a multiplication operation.

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

Table 5: *Truth table for an AND gate.*

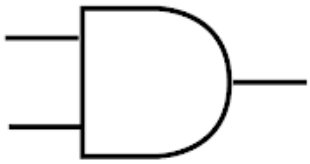

Figure 4: *Graphical representation of an AND logic gate.*

OR gate / Boolean addition

Definition 1.19. The OR gate gets a minimum of two input values (A, B) and gives the one output value (Z). The output value Z will be true, $Z = 1$, if at least one of the inputs A and B is true.

Then, the logic equation corresponding to the OR gate is $Z = A + B$. Looking at this equation we could see the OR gate as an addition operation.

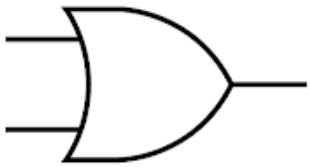

Figure 5: *Graphical representation of an OR logic gate.*

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

Table 6: *Truth table for an OR gate.*

NAND gate / Inverted AND gate

Definition 1.20. A NAND gate is a type of logic gate that gets at least two input values (A, B) and gives one output value (Z), which is 1 if at least one of the inputs is 0.

This logic gate can be though as a combination of an AND and a NOT gates, i.e. an inverted AND gate, and its corresponding logic equation is $Z = \overline{A \cdot B}$.

Note. $Z = \overline{A \cdot B}$ is not the same as $Z = \overline{A} \cdot \overline{B}$.

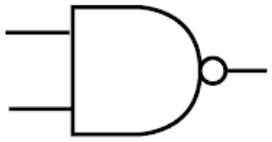

Figure 6: *Graphical representation of a NAND logic gate.*

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

Table 7: *Truth table for a NAND gate.*

NOR gate / Inverted OR gate

Definition 1.21. A NOR gate is a type of logic gate that gets at least two input values (A, B) and gives one output value (Z), which is 1 if both inputs are 0, and is 0 if at least one of the inputs is 1.

The corresponding logic equation for a NOR gate is $Z = \overline{A + B}$.

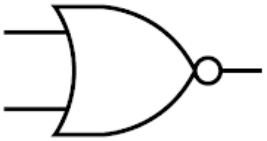

Figure 7: *Graphical representation of a NOR logic gate.*

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

Table 8: *Truth table for a NOR gate.*

XOR gate / OR-exclusive gate

Definition 1.22. A XOR gate, or OR-exclusive gate, is a type of logic gate that gets at least two input values (A, B) and gives one output value (Z), which is 1 if and only if one input is 1.

The XOR gate can be seen as a comparator of bits (if both bits are equal the output is 0); and its corresponding logic equation is $Z = A \oplus B$.

Figure 8: *Graphical representation of a XOR logic gate.*

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

Table 9: *Truth table for a XOR gate.*

XNOR gate / Inverted OR-exclusive gate

Definition 1.23. A XNOR gate is a type of logic gate that gets two input values (A, B) and gives one output (Z), which is 0 if and only if one input is 0.

The corresponding logic equation for a XNOR gate is $Z = \overline{A \oplus B}$.

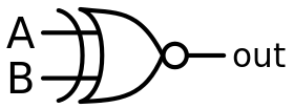

Figure 9: *Graphical representation of a XNOR logic gate.*

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	1

Table 10: *Truth table for a XNOR gate.*

1.3.2 Boolean algebra properties and dual equations

Name	Property	Dual
Indentity	$B \cdot 1 = B$	$B + 0 = B$
Annulment	$B \cdot 0 = 0$	$B + 1 = 1$
Idempotent	$B \cdot B = B$	$B + B = B$
Double negation	$\overline{\overline{B}} = B$	$\overline{\overline{B}}$
Complement	$B \cdot \overline{B} = 0$	$B + \overline{B} = 1$
Commutative	$B \cdot C = C \cdot B$	$B + C = C + B$
Associative	$(B \cdot C) \cdot D = B \cdot (C \cdot D)$	$(B + C) + D = B + (C + D)$
Distributive	$(B \cdot C) + (B \cdot D) = B \cdot (C + D)$	$(B + C) \cdot (B + D) = B + (C \cdot D)$

Table 11: *Basic properties of boolean algebra.*

Definition 1.24. (De Morgan's laws). De Morgan's laws let us the operations in the expression we are working with.

$$\left(\overline{B_0 \cdot B_1 \cdot \ldots \cdot B_{n-1}}\right) = \left(\overline{B_0} + \overline{B_1} + \ldots + \overline{B_{n-1}}\right) \tag{7}$$

$$\left(\overline{B_0 + B_1 + \ldots + B_{n-1}}\right) = \left(\overline{B_0} \cdot \overline{B_1} \cdot \ldots \cdot \overline{B_{n-1}}\right) \tag{8}$$

1.4 Logic circuits and functions

Definition 1.25. The combination of different logic gates by taking the output from one gate and using it as the input to another gate is called a **logic circuit**.

Logic circuits enables computers to do more complex operations than they could accomplish with just a single gate.

A logic circuit is composed of inputs, outputs, its functional specification and its temporal specification (delay).

Note. Any logic function can be expressed using AND, OR and NOT gates.

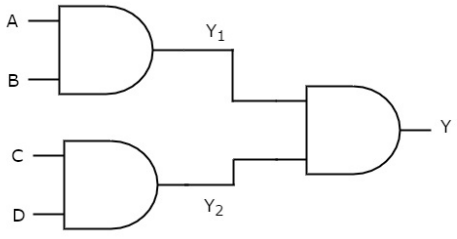

Figure 10: *Example of a logic circuit that takes four inputs (A, B, C, D) and gives one output (Y).*

The functional specification for the logic circuit in figure 10 is

$$Y_1 = (A, B) \qquad Y_2 = (C, D) \qquad Y\,(Y_1, Y_2) \tag{9}$$

Definition 1.26. Combinational logic circuits are the ones in which the output state depends exclusively on input states, i.e. it's a circuit without memory.

Definition 1.27. Sequential logic circuits are the ones in which the output state also depends on the previous state of the system, i.e. it's a circuit with memory.

Definition 1.28. A logic function is a boolean expression that relates logic variables directly or complemented by the use of AND and OR operations.

Logic functions are represented as logic circuits with 2 levels whose **canonical form** can be

- Sum of Products of all variables or their conjugates: Sum of Minterms or SOP circuits (Sum of Products).
- Product of Sums of all variables or their conjugates: Product of Maxterms or POS circuits (Products of Sums).

Note. All boolean equations can be represented as a sum of minterms (SOP).

Definition 1.29. Each row of a truth table is a minterm. A minterm is a product (AND) of variables and their complement. Each minterm is true for that row, and only for that row.

Function is constructed as the sum (OR) of minterms whose output is true. It's then a sum of products.

A	**B**	**Y**	**Minterm**	**Index**
0	0	0	$\overline{A} \cdot \overline{B}$	0
0	1	1	$\overline{A} \cdot B$	1
1	0	0	$A \cdot \overline{B}$	2
1	1	1	$A \cdot B$	3

Table 12: $Y = F\,(A, B) = \overline{A}B + AB$.