

MATHEMATICAL FOUNDATION OF DIGITAL CIRCUITS

Digital Circuits

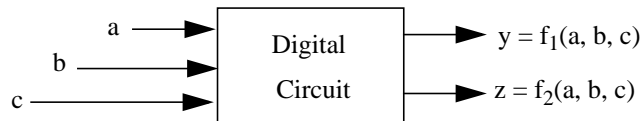
CS 2204 studies digital circuits. It focuses on the **theory, analysis** and **synthesis** (design) of digital circuits. The theory provides the mathematical foundation (i) to analyze and design digital circuits, (ii) to verify that a circuit performs what it is supposed to and (iii) to simplify (minimize) digital circuits.

The analysis means a digital circuit is given and we are asked to determine its **input-output relationship** (its purpose, operation, what it does). One studies a circuit and then states the input-output relationship (the purpose) of the circuit in text or on a truth table or on an operation table or on a state diagram or on an operation diagram. If the circuit is complex, one needs to employ block-based analysis to obtain the purpose.

The synthesis means an input-output relationship (purpose, operation) is given and we are asked to design the digital circuit. One studies the input-output relationship (the purpose) of a circuit which is in text or on an operation table or on an operation diagram and then designs the circuit. If the circuit is complex, one needs to employ block-based design to obtain the digital circuit.

It is clear that the input-output relationship is a very crucial element of digital circuit study. Another one is the block-based approach : Complex digital circuits are worked on piece by piece (block by block) to handle the large number of details. Blocks are analyzed/designed individually, then they are combined one by one and analyzed/designed and finally the whole circuit is analyzed/designed.

The input-output relationship and block-based approach treat the digital circuit as a black box with inputs and outputs in the beginning. It focuses on the relationship of outputs to its inputs : Every output must be described as a **function** of its inputs. We know from our experience that a function is a **mathematical entity** that precisely describes how an output is determined by its inputs. For example, in the figure below, the digital circuit shown as a black box has three inputs and two outputs (two digital functions) :



Functions f_1 and f_2 precisely indicate when outputs y and z are 1 and 0. That is, f_1 specifies the input-output relationship of y to inputs a , b and c and f_2 specifies the input-output relationship of z to inputs a , b and c . A function for a simple combinational circuit is represented in different forms, such as expressions, minterm lists, maxterm lists, truth tables, etc. For complex combinational circuits, we do not use these representations, but operation tables to specify the input-output relationship. Due to historical reasons, digital circuits are called **switching circuits**, digital circuit functions are called **switching functions** and the algebra is called **switching algebra**.

Switching Algebra

George Boole, in 1854, introduced a systematic treatment of logic and developed for this purpose an algebraic system now called **Boolean Algebra**. This algebraic system, ($S ; + ; \cdot ; - ; 0 ; 1$), consists of a set S of elements, **binary** operations “+” (called “plus”) and “ \cdot ” (called “dot”) and a **unary** operation “-” (called “complement”) and at least two elements “0” and “1” in set S .

E. V. Huntington in 1904 defined a **many-valued Boolean Algebra** with the following postulates (axioms) :

PI) There exists a set S of elements which satisfies the principle of **substitution** under the equivalence relation “=” that if $k = m$, then “ m ” may be substituted for “ k ” in any expression containing “ k ,” without affecting the validity of the expression.

PII) a) The set S is **closed** with respect to the “+” operation : if $k, m \in S$, then $(k + m) \in S$

PII) b) The set S is **closed** with respect to the “ \cdot ” operation : if $k, m \in S$, then $(k \cdot m) \in S$

PIIIa) $k + 0 = k$ 0 is the **identity** element with respect to “+”

PIIIb) $k \cdot 1 = k$ 1 is the **identity** element with respect to “.”

PIVa) $k + m = m + k$ “+” is **commutative**

PIVb) $k \cdot m = m \cdot k$ “.” is **commutative**

PVa) $k + (m \cdot p) = (k + m) \cdot (k + p)$ “+” is **distributive** over “.”

PVb) $k \cdot (m + p) = (k \cdot m) + (k \cdot p)$ “.” is **distributive** over “+”

PVIa) For every element k of set S , there exists an element \bar{k} (**complement of k**) of set S , such that $k + (\bar{k}) = 1$

PVIb) For every element k of set S , there exists an element \bar{k} (**complement of k**) of set S , such that $k \cdot (\bar{k}) = 0$

Above, we do not mention the number of elements in S and how the operators manipulate the elements of S . In fact, one can formulate different Boolean Algebras depending on the number of elements in S and operator definitions. The **two-valued (two-element) Boolean Algebra** also known as **Switching Algebra** is one of the most widely known and studied algebras. It is the foundation of combinational circuits. Switching Algebra was developed by C. Shannon in 1938.

Switching Algebra is a subject of **Switching Theory** in Mathematics. Hence Switching Theory is the mathematical foundation of digital circuits. Switching Theory is concerned with how to represent, analyze and develop these switching functions that describe the input-output relationship of digital circuits.

Switching Algebra is defined by

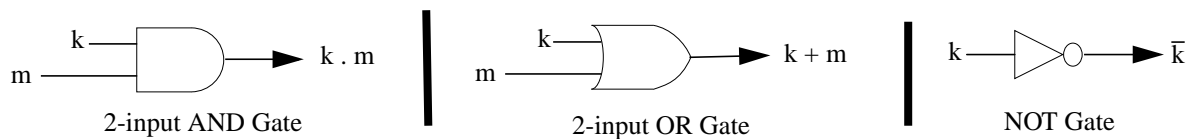
➤ Algebraic system (**0 ; 1 ; + ; . ; -**) and

➤ The six postulates above under the condition that

➔ The following operator rules on (0 ; 1) apply, i.e., the operator definitions :

<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="border-right: 1px solid black; padding: 2px 5px;">k</td> <td style="border-right: 1px solid black; padding: 2px 5px;">m</td> <td style="padding: 2px 5px;">$k.m$</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px 5px;">0</td> <td style="border-right: 1px solid black; padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">0</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px 5px;">0</td> <td style="border-right: 1px solid black; padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px 5px;">1</td> <td style="border-right: 1px solid black; padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">0</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px 5px;">1</td> <td style="border-right: 1px solid black; padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">1</td> </tr> </table>	k	m	$k.m$	0	0	0	0	1	0	1	0	0	1	1	1	Definition of the AND operator	<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="border-right: 1px solid black; padding: 2px 5px;">k</td> <td style="border-right: 1px solid black; padding: 2px 5px;">m</td> <td style="padding: 2px 5px;">$k+m$</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px 5px;">0</td> <td style="border-right: 1px solid black; padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">0</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px 5px;">0</td> <td style="border-right: 1px solid black; padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">1</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px 5px;">1</td> <td style="border-right: 1px solid black; padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">1</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px 5px;">1</td> <td style="border-right: 1px solid black; padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">1</td> </tr> </table>	k	m	$k+m$	0	0	0	0	1	1	1	0	1	1	1	1	Definition of the OR operator	<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="border-right: 1px solid black; padding: 2px 5px;">k</td> <td style="padding: 2px 5px;">\bar{k}</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">1</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> </tr> </table>	k	\bar{k}	0	1	1	0	Definition of the Complement (NOT, Invert) operator
k	m	$k.m$																																							
0	0	0																																							
0	1	0																																							
1	0	0																																							
1	1	1																																							
k	m	$k+m$																																							
0	0	0																																							
0	1	1																																							
1	0	1																																							
1	1	1																																							
k	\bar{k}																																								
0	1																																								
1	0																																								

Today, each operator above is directly implemented by **digital electronic circuits**. That is, a digital electronic circuit with transistors, capacitors, resistors and other electronic components is designed to perform the specific logic operation. Each such digital electronic circuit for an operator is called “**gate**.” The figure below shows the **schematic** symbols of the gates for the three operators above.



AND and OR gates can have **any number of inputs**, as long as there are at least two inputs. However, a NOT gate always has a **single input**.



There are **other** operators, such as **NAND**, **NOR**, **EXOR** and **EXNOR**, that are also implemented by gates. We will also study them.

Today, on a chip electronic circuits with **transistors** implement several gates to millions of gates. There are chips, such as microprocessor chips, with billions of transistors.

Chip densities have increased at the rate of **Moore's Law** since the 1960s :

➤ **The number of transistors on a chip doubles every two years.**

It is forecast that doubling every two years may **not** be sustainable and that it may be doubling every three years or so.

Since 1938, theorems have been developed in Switching Algebra. These theorems follow and satisfy the postulates and operator definitions given above :

TI) Idempotency :

a) $k + k = k$

b) $k \cdot k = k$

TII) Null elements

a) $k + 1 = 1$

b) $k \cdot 0 = 0$

TIII) Absorption :

a) $k + (k \cdot m) = k$

b) $k \cdot (k + m) = k$

TIV) Involution : $\overline{\overline{k}} = k$

TV) Associativity :

a) $k + (m + p) = (k + m) + p = k + m + p$

b) $k \cdot (m \cdot p) = (k \cdot m) \cdot p = k \cdot m \cdot p$

TVI)

a) $k + ((\overline{k}) \cdot m) = k + m$

b) $k \cdot ((\overline{k}) + m) = k \cdot m$

TVII) DeMorgan's theorems :

a) $\overline{(k + m)} = \overline{k} \cdot \overline{m}$

b) $\overline{(k \cdot m)} = \overline{k} + \overline{m}$

TVIII) Consensus theorem :

a) $(k \cdot m) + ((\overline{k}) \cdot p) + (m \cdot p) = (k \cdot m) + ((\overline{k}) \cdot p)$

b) $(k + m) \cdot ((\overline{k}) + p) \cdot (m + p) = (k + m) \cdot ((\overline{k}) + p)$

The duality principle :

A postulate or theorem can be obtained from another postulate or theorem by interchanging the binary operators “+” and “.” and the identity elements “0” and “1.”

Example :

“ $k + 1 = 1$ ” is a theorem. One can obtain its dual by interchanging “+” with “.” and “1” with “0”. Then, the new theorem is “ $k . 0 = 0$ ”.

Similarly, if “ $k . 0 = 0$ ” is given, one can then obtain “ $k + 1 = 1$ ” as the dual of it.

The AND operator symbol :

In order to reduce the number of symbols in expressions, we will **not** show the “.” symbol between variables. We will imply there is an AND operation between them.

Example :

$$a.b.c = a b c$$

Precedence rules :

In order to reduce the number of parentheses, we have the following set of precedence rules that indicates which subexpression or operator to evaluate next :

- ⇨ Evaluate the expression inside a pair of parentheses
- ⇨ Evaluate NOT
- ⇨ Evaluate AND
- ⇨ Evaluate OR

Example :

$$(\bar{a}).(((b.(\bar{c})) + (b.c)) + ((a.c).(b + (\bar{b})))) = \bar{a}(b \bar{c} + bc) + ac(b + \bar{b})$$

Finally : the Wakerly book classifies theorems with respect to

- ⇨ single-variable
- ⇨ two- and three-variable
- ⇨ n-variable.

One can obtain an n-variable theorem from a single-variable or two- or three-variable theorem, by using the principle of substitution together with the use of other postulates and theorems.

An n-variable theorem holds true since AND, OR, NAND, NOR, EXOR and EXNOR gates can have more than two inputs.

Example :

⇨ The n-variable version of “ $k + k = k$ ” is

$$\text{⇨ “} k + k + k + \dots + k = k \text{”}$$

⇨ This is an OR gate with “n” inputs :

