Combinational Logic Circuits

Markus Gasser — TU Wien — markus.gasser@tuwien.ac.at

Omid Mirmotahari — Universitetet i Oslo — omidmi@ifi.uio.no

August 21, 2016

Abstract

This paper will give a brief introduction into combinational logic circuits. This field covers the basic binary logic gates and their behavior and boolean algebra consisting of it's identities and standard forms to write expressions.

I. Binary Logic Gates

A *logic gate* is a basic binary circuit that manipulates binary information. In order to describe the manipulation that these components perform mathematically *Boolean algebra* can be used.

Binary logic deals with binary variables, variables that con only take two discrete values (1 and 0) and mathematical logic applied to these variables. *Logic gates* are electronic circuits with inputs and outputs that realize binary logic operations. A *truth table* is a table that lists all outputs of a logic operation with all possible combinations of input.

The basic logical operations, their respective gates and their truth tables are listed in the following enumeration.

1. *AND*. Denoted $Z = X \cdot Y$ or $Z = XY$, read ″Z is equal to X and Y″, depicted as a gate in figure 1 and defined through the truth table in table 1.

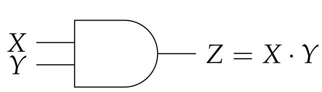

Figure 1: *AND gate*

2. *OR*. Denoted $Z = X + Y$, read ″Z is equal to X or Y″, depicted in figure 2 and defined through the truth table in table 2.

X	Y	$X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

Table 1: *AND truth table*

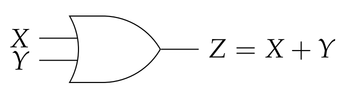

Figure 2: *OR gate*

X	Y	$X \cdot Y$
0	0	0
0	1	1
1	0	1
1	1	1

Table 2: *OR truth table*

3. *NOT*. Also called *inverter*, denoted $Z = \overline{X}$, read ″Z is equal to not X″, depicted in figure 3 and defined through the truth table in table 3.

4. *NAND* or NOT-AND. It is the complement of the AND, denoted by $Z = \overline{X \cdot Y}$, depicted in figure 4 and defined through table 4.

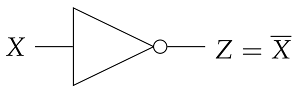

Figure 3: *NOT gate*

X	$\overline{X}$
0	1
1	0

Table 3: *NOT truth table*

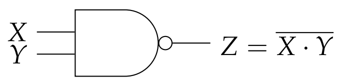

Figure 4: *NAND gate*

X	Y	$\overline{X \cdot Y}$
0	0	1
0	1	1
1	0	1
1	1	0

Table 4: *NAND truth table*

5. *NOR* or NOT-OR. It is the complement of the OR, denoted by $Z = \overline{X + Y}$, depicted in figure 5 and defined through table 5.

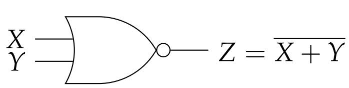

Figure 5: *NOR gate*

X	Y	$\overline{X + Y}$
0	0	1
0	1	0
1	0	0
1	1	0

Table 5: *NOR truth table*

6. *XOR* or exclusive-OR. Denoted by $Z = X\overline{Y} + \overline{X}Y = X \oplus Y$, depicted in figure 6 and defined trough table 6.

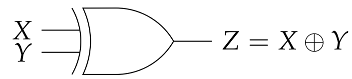

Figure 6: *XOR gate*

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

Table 6: *XOR truth table*

7. *XNOR* or exclusive-NOR. Denoted by $Z = XY + \overline{X}\,\overline{Y} = \overline{X \oplus Y}$, depicted in figure 7 and defined trough table 7.

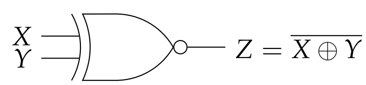

Figure 7: *XNOR gate*

X	Y	$\overline{X \oplus Y}$
0	0	1
0	1	0
1	0	0
1	1	1

Table 7: *XNOR truth table*

II. Boolean Algebra

Using *Boolean algebra* we can mathematically work with binary variables and operations. We distinguish between *Boolean expressions*, these are an algebraic expressions formed using binary variables and the constants 0 and 1, and *Boolean functions*, a equation consisting of binary variables followed by an equals sign and a Boolean expression.

A Boolean equation determines the logical relationship between binary variables and can

1.	$X + 0 = X$	2.	$X \cdot 1 = X$	
3.	$X + 1 = 1$	4.	$X \cdot 0 = 0$	
5.	$X + X = X$	6.	$X \cdot X = X$	
7.	$X + \overline{X} = 1$	8.	$X \cdot \overline{X} = 0$	
9.	$\overline{\overline{X}} = X$			
10.	$X + Y = Y + X$	11.	$XY = YX$	Commutativity
12.	$X + (Y + Z) = (X + Y) + Z$	13.	$X(YZ) = (XY)Z$	Associativity
14.	$X(Y + Z) = XY + XZ$	15.	$X + YZ = (X + Y)(X + Z)$	Distributivity
16.	$\overline{X + Y} = \overline{X} \cdot \overline{Y}$	17.	$\overline{X \cdot Y} = \overline{X} + \overline{Y}$	DeMorgan's Theorem

Table 8: *Basic identities of Boolean algebra.*

be evaluated using a truth table. A Boolean function can be equivalently expressed by a circuit diagram consisting of logic gates. These circuits are often called *combinational logic circuits*.

i. Basic Identities

Table 8 lists the basic identities of Boolean algebra.

It is important to note that the equations in the right column are the *dual* of the equations in the left column and vice versa, meaning that the can be obtained from each other by interchanging ORs and ANDs and 0s and 1s.

By using these identities one can conduct *algebraic manipulation* to simplify arbitrary boolean expressions and equations.

To obtain the *complement of a function* $\overline{F}$ of F algebraically DeMorgan's theorem can be used, by interchanging AND and OR operations and complementing each variable and constant.

III. Standard Forms

Since equivalent algebraic functions can be written in different ways there are *standard forms*, specific ways of writing these expressions consisting of *product terms* (e.g. $XY\overline{Z}$) and *sum terms* (e.g. $X + Y + \overline{Z}$).

i. Minterms and Maxterms

A *minterm* is a a product term in which all the variables appear exactly once (either complemented or uncomplemented). It represents exactly one combination of binary variable values in the truth table. Minterms are usually denoted by m_j where j denotes the decimal representation of the binary combination corresponding to the minterm.

A *maxterm* is a sum term in which all variables appear exactly once. It is usually denoted by M_j. The minterm and maxterm with the same subscript are complements of each other, $M_j = \overline{m_j}$.

A Boolean function can be expressed by forming the logical sum of all the minterms that produce a 1 in the function. This expression is called the *sum of minterms* and can be abbreviated with the $\sum$ symbol representing the logical sum (e.g. $F(X, Y, Z) = XYZ + \overline{X}YZ = m_3 + m_7 = \sum(3, 7)$).

By complementing the sum of minterms, or equivalently forming the logical product of all the maxterms that produce a 0 in the function, we obtain the *product of maxterms*, abbreviated by the symbol $\prod$ representing the logical product (e.g. $F(X, Y, Z) = (X + Y + Z)(\overline{X} + Y + Z) = M_0 + M_4 = \prod(0, 4)$).

ii. Sum of Products

Once the sum of minterms is obtained from the truth table of a function it can be simplified using the identities in table 8. The resulting simplified expression is called *sum of products*. The realization with gate logic consists of a group of AND gates followed by a single OR gate and is thus called *two-level implementation.*

iii. Product of Sums

Equivalently to the sum of products the *product of sums* is obtained by simplifying the product of maxterms. The gate structure consists of a group of OR gates followed by a single AND gate.

IV. Summary

We introduced the basics of combinational logic including the most important parts of Boolean Algebra. This includes the combinational operations AND, OR, NOT, NAND, NOR, XOR and XNOR, together with their definition, their gate symbols and their expressions. Furthermore we introduced the basics of Boolean algebra such as basic identities including DeMorgan's theorem and the standard forms with minterms and maxterms resulting in the sum of products and the product of sums.

References

[Kime and Mano, 2004] Kime, Charles R and Mano, M Morris (2004). Logic and computer design fundamentals. *Pearson Education*.