2 Basic logic gates

2.1 Objectives

The fundamental principles and properties of digital circuits are presented from a technological perspective. The basic logic gates are introduced and analyzed using the operations from boolean algebra. Given the priority of the boolean operators, the computation of the truth table for a boolean expression is detailed. The most common properties of boolean algebra are listed and several operational conversions are explained at gate level as well as implementing new functionalities.

2.2 Fundamental technological aspects

The basic logic gates are the primary circuits used in logic design. They have simple functionality and contain a small number of transistors or semiconductors. They implement the basic operations in boolean algebra: NOT (INV), OR, AND. Gates implementing combined functionality such as NOR (NOT OR), NAND (NOT AND), XOR (eXclusive OR) and XNOR (eXclusive NOR) are also considered basic gates due to the simplicity of their electronic compound.

From a technological perspective there are two common categories of gates, the TTL (Transistor-Transistor Logic) and the CMOS (Complementary Metal-Oxide Semiconductor). The most important parameters defining the two families are related to tolerance against variations, switching speed and power consumption.

- *tolerance against variations* – according to values in Figure 2. 1 the CMOS circuits have an extended voltage interval for state 0, which means a better tolerance against oscillations around small values and a smaller chance for an uncontrolled exit outside this state, as compared to TTL circuits.

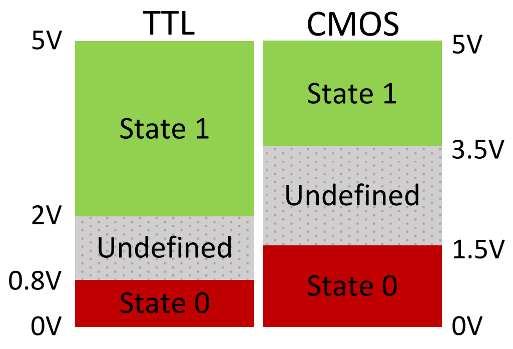


Figure 2. 1 Voltage level association with logic states

- *switching speed* – this parameter represents the reaction time when a state transition should occur on output due to changes of values on one or more inputs. Usually, the CMOS gates are slower than their counterpart.

- *power consumption* – the TTL circuits are power hungry, hence mobile devices running on batteries or accumulators are equipped with CMOS gates, for a longer life period.

Circuits from the TTL family will be studied in detail. In dedicated electronic catalogues they can be identified by their code composed of more digits. The initial two digits are either 74 or 54. Considering the properties mentioned earlier several categories of TTLs can be identified:

- standard.
- low power.
- high-speed.
- Schottky (the fastest).
- combinations (e.g., Low-Power Schottky).

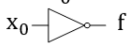
2.3 The basic logic gates

The basic logic gates implement the functionality of boolean operators with one or two inputs (variables) and can be extended to more inputs. Their expression, the truth tables and the associated graphic symbol for logic diagrams are listed in Table 2. 1.

Table 2. 1 The basic logic gates

NOT (INVerter)

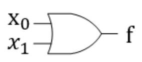
$f = \overline{x_0}$



x_0	f
0	1
1	0

OR

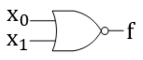
$f = x_1 + x_0$



x_1	x_0	f
0	0	0
0	1	1
1	0	1
1	1	1

NOR (NOT OR)

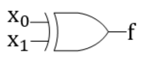
$f = \overline{x_1 + x_0}$



x_1	x_0	f
0	0	1
0	1	0
1	0	0
1	1	0

XOR (eXclusive OR)

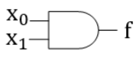
$f = x_1 \oplus x_0$



x_1	x_0	f
0	0	0
0	1	1
1	0	1
1	1	0

AND

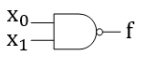
$f = x_1 \cdot x_0$



x_1	x_0	f
0	0	0
0	1	0
1	0	0
1	1	1

NAND (NOT AND)

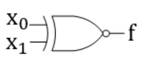
$f = \overline{x_1 \cdot x_0}$



x_1	x_0	f
0	0	1
0	1	1
1	0	1
1	1	0

XNOR (eXclusive NOR)

$f = \overline{x_1 \oplus x_0} = x_1 \odot x_0$



x_1	x_0	f
0	0	1
0	1	0
1	0	0
1	1	1

An analysis of the symbols shows that logic inversion is usually represented by a small circle and the mathematical operator is represented by a bar on top of the inverted boolean expression. Also, you should not confuse the + operator with addition, nor the · operator with multiplication from arithmetic.

From the truth tables, the effect of the logic gates can be summarized as follows:

- the NOT gate inverts the input.
- the result of OR is 1 if at least one input is 1 – the equivalent for max function.
- the result of AND is 1 if all inputs are 1 – the equivalent for min function.
- NOR is the inversion of OR.

- NAND is the inversion of AND.
- the result of XOR is 1 when both inputs are not equal.
- the result of XNOR is 1 when both inputs are identical.
- XNOR is the inversion of XOR.

The logic gates are associated unique codes (Table 2. 2) which identify the boolean operation.

Table 2. 2 Codes associated to TTL basic logic gates

Gate	Code
NAND	7400
NOR	7402
NOT	7404
AND	7408
OR	7432
XOR	7486
XNOR	74266

Note: In Project Navigator the basic logic gates are grouped in category *Logic* and can be searched by their name (AND, OR, INV etc.). INV replaces NOT.

Observation: An alternative implementation for NOR is to connect the output of OR to the input of NOT. This solution is less efficient since it uses two gates, therefore doubles the resources used. Also, the speed decreases because the propagation path covers both gates, and the power consumption increases. Nevertheless, the preferred solution is the use of dedicated logic gates implementing the behavior described by the truth table of NOR. The situation is similar for NAND, XOR and XNOR.

2.4 Generating the truth table for a boolean expression

The computation of an expression is based on the priority of operators presented in Table 2. 3. The results are computed for all combinations of values associated to the variables. For *n* variables there are 2^n combinations.

Considering the expression $\overline{a \cdot b} + c$ the first operation will be the NAND over a and b, followed by the operation OR over the previous result and the variable c. The computations will use the specific truth tables of the operations. According to this rule the intermediary values and results are presented in Table 2. 4.

Table 2. 3 Priority of operators in boolean algebra

Operator	Priority
()	high
$\overline{}$	↕
•	
$\oplus$, $\odot$	
+	low

Table 2. 4 Computing the truth table for $\overline{a\cdot b}+c$

a b c		$\overline{a\cdot b}+c$
0 0 0	:	$\overline{0\cdot 0}+0=1+0=\mathbf{1}$
0 0 1	:	$\overline{0\cdot 0}+1=1+1=\mathbf{1}$
0 1 0	:	$\overline{0\cdot 1}+0=1+0=\mathbf{1}$
0 1 1	:	$\overline{0\cdot 1}+1=1+1=\mathbf{1}$
1 0 0	:	$\overline{1\cdot 0}+0=1+0=\mathbf{1}$
1 0 1	:	$\overline{1\cdot 0}+1=1+1=\mathbf{1}$
1 1 0	:	$\overline{1\cdot 1}+0=0+0=\mathbf{0}$
1 1 1	:	$\overline{1\cdot 1}+1=0+1=\mathbf{1}$

a b c	$\overline{a\cdot b}+c$
0 0 0	1
0 0 1	1
0 1 0	1
0 1 1	1
1 0 0	1
1 0 1	1
1 1 0	0
1 1 1	1

2.5 Designing the logic diagram for an expression

The connection of gates in a logic diagram follows the operator priorities. Applying the priorities from Table 2. 3 on expression $f=b+a\cdot\overline{a+(b+a)}$, the result can be calculated using the following steps:

1. P1 = $b+a$ (the OR inside brackets)
2. P2 = $\overline{a+P1}$ (NOR operation)
3. P3 = $a\cdot P2$ (AND operation)
4. f = $b+P3$ (OR operation)

Considering the gates propagate the values from inputs on the left side to the output on the right side the resulting logic diagram is presented in Figure 2. 2.

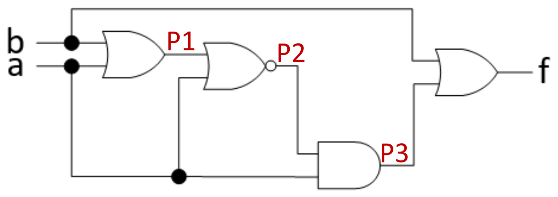


Figure 2. 2 The logic diagram for $f=b+a\cdot\overline{a+(b+a)}$

2.6 Properties of boolean algebra

Two boolean functions are considered identical if they return the same result for all possible combination of input values.

Using the tables for the basic logic gates the following properties can be demonstrated true:

- Commutative law: $a+b=b+a$; $a\cdot b=b\cdot a$
- Associative law: $(a+b)+c=a+(b+c)$; $(a\cdot b)\cdot c=a\cdot(b\cdot c)$
- Distributive law: $a+(b\cdot c)=(a+b)\cdot(a+c)$; $a\cdot(b+c)=(a\cdot b)+(a\cdot c)$
- Identity element: $a\cdot 1=a$; $a+0=a$
    - Consequences of the identity element: $a+1=1$; $a\cdot 0=0$

- Idempotent law: $a + a = a$; $a \cdot a = a$
- Double negation law: $\bar{\bar{a}} = a$
- XOR operation: $a \oplus b = \bar{a} \cdot b + a \cdot \bar{b}$
- XNOR operation: $a \odot b = a \cdot b + \bar{a} \cdot \bar{b}$
- De Morgan: $\overline{a + b} = \bar{a} \cdot \bar{b}$; $\overline{a \cdot b} = \bar{a} + \bar{b}$

2.7 Transforming logic gates and generating new behaviors

It is possible to generate various functionalities using the logic gates and the properties of boolean algebra. A common example is the reduction of inputs for AND, NAND, OR and NOR gates. A 4-input AND implementing expression $a \cdot b \cdot c \cdot d$ can be redesigned to implement $a \cdot b$ using any of the following properties: $a \cdot b = a \cdot 1 \cdot b \cdot 1 = a \cdot b \cdot a \cdot b = a \cdot b \cdot b \cdot b = a \cdot a \cdot a \cdot b = \cdots$, etc. In these expressions, the value 1 represents the power source (5V) and is called VCC (Voltage Common Collector). It is available in Project Navigator, in category *General*. Several solutions implementing a 2-input AND using a 4-input AND are highlighted in Figure 2. 3. The solutions are similar for AND, and NAND.

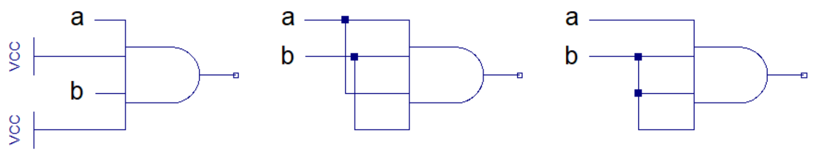


Figure 2. 3 Solutions for implementing a 2-input AND using a 4-input gate

The equivalent properties of OR operation use the constant 0, which is the ground GND (0V) of the circuit: $a + b = a + 0 + b + 0 = a + b + a + b = a + b + b + b = a + a + a + b = \cdots$, etc. Several solutions for input reduction are showcased in Figure 2. 4. They are similar for NOR. GND is also available in category *General*.

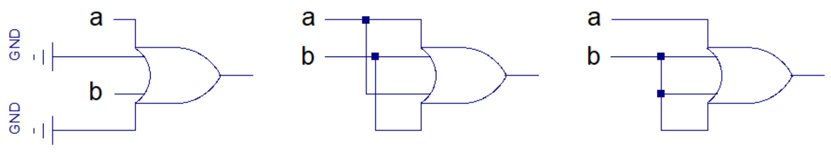


Figure 2. 4 Solutions for implementing a 2-input OR using a 4-input gate

An inverter (NOT) can be implemented from a 2-input NAND or NOR by the following expressions: $\bar{a} = \overline{a \cdot 1} = \overline{a \cdot a} = \overline{a + a} = \overline{a + 0}$ (Figure 2. 5).

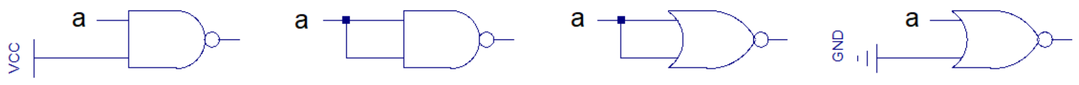


Figure 2. 5 Implementing NOT from other gates

Based on De Morgan laws, an OR can be implemented from a NAND and two inverters. An AND can be implemented from a NOR and two inverters. They respect the following properties: $a \cdot b = \overline{\bar{a} + \bar{b}}$, $a + b = \overline{\bar{a} \cdot \bar{b}}$ (Figure 2. 6).

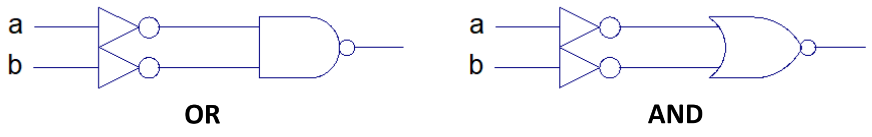
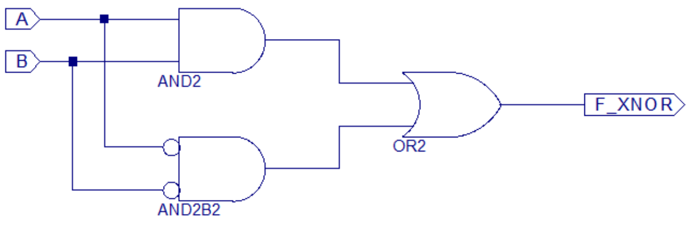


Figure 2. 6 Alternative solutions for implementing OR (left) and AND (right)

2.8 Assignments

1. Add one diagram to the project *ttl_env*, which implements the basic logic gates NOT, NOR and XOR with two inputs. Use the board and the corresponding tables from Table 2. 1 to test their functionality.

2. Add the following logic diagram to the project. The diagram implements the expression $a \cdot b + \bar{a} \cdot \bar{b}$. Use the board and the table of XNOR to demonstrate the equality $a \odot b = a \cdot b + \bar{a} \cdot \bar{b}$.



3. For each expression below calculate the truth table, design the corresponding logic diagram, and test its behavior on the board against the truth table. Do not apply any changes to the expressions. Finalize each point before stepping to the next.

   a. $\bar{a} \cdot b + \bar{c}$
   b. $a \cdot c + \overline{b \cdot c}$
   c. $a + \overline{b \cdot c}$
   d. $\overline{a + \bar{b}} \cdot c$
   e. $\overline{(a + b) \cdot \overline{a + c}}$
   f. $a + \overline{b \cdot \overline{a + c}}$