Introduction to Binary Logic and Boolean Algebra

Antoine Lavault[1,2]

[1]Apeira Technologies

[2]UMR CNRS 9912 STMS, IRCAM, Sorbonne Université

18 octobre 2022

Contents

What is going to happen ?

What you should know, use and understand by the end of the class :

- Boolean logic and friends
- Logic gates, synchronous and asynchronous logic.
- Programmable Logical Devices : PAL, CPLD, FPGA
- Circuit description in VHDL

Analog vs. Digital

Digital electronics is defined as opposed to analog electronics.

- Analog : the signal is continuous.
- Digital : the signal is discrete, either in time (sampling) or in value (discretization)

And that's about it.
First, let's talk about binary, octal and hexadecimal.

Radices and bases

Let b be a positive integer such that $b \geq 1$. Then every positive integer a can be expressed uniquely in the form :

$$a = r_m b^m + r_{m-1} b^{m-1} + \cdots + r_1 b + r_0, \tag{1}$$

where m is a non-negative integer and the r's are integers such that

$$0 < r_m < b; \tag{2}$$

and

$$0 \leq r_i < b; \text{for } i = 0, 1, \cdots, m-1 \tag{3}$$

We say the number a is expressed in base b and is noted a_b.

Binary = base 2, Hexadecimal = base 16.
For bases greater than 10, the numbers are replaced by letters : a for 10, b for 11 etc...

Exemple

$3 = 1 * 2^1 + 1 * 2^0 = 3 * 16^0 = 10_2 = 3_{16}$
$24 = 1 * 2^4 + 1 * 2^3 = 1 * 16 + 8 * 16^0 = 11000_2 = 18_{16}$
$1465 = 5 * 16^2 + 11 * 16^1 + 9 * 16^0 = 5b9_{16}$

Note : hexadecimal numbers are also noted as $0x\cdots$ (e.g $0x5b9$) and binary numbers $0b\cdots$

Exercise

$\underbrace{1010}_{a}\,\underbrace{0111}_{7} = 0xa7$ is a byte i.e 8 binary digits (or bits)
What is the hexadecimal value of 0b0100 0101 ?

Contents

A binary variable is a variable with only two possible states.

Définition

A binary variable *a* is one that takes its values in $\mathcal{B} = \{0, 1\}$

A binary variable can be associated to logical values : True and False. Most of the time, True =1 and False = 0.

Définition

On $\mathcal{B}$, we can define two binary operations called conjunction, noted $\wedge$, and disjunction, noted $\vee$, and an unary operation called negation, noted $\neg$.
For x and y in $\mathcal{B}$, we have :

- $x \wedge y = 1$ if $x = y = 1, x \wedge y = 0$ otherwise
- $x \vee y = 0$ if $x = y = 0, x \vee y = 1$ otherwise
- $\neg x = 1$ if $x = 0, x = 0$ otherwise

Analogy

These operations are suspiciously similar to the intersection, the union and the complement in set theory.

Common names

Conjonction = AND, Disjonction = OR, Negation = NOT

For x, y in $\mathcal{B}$,

- $x \wedge y = x \cdot y$
- $x \vee y = x + y$
- $\neg x = \bar{x}$
- Exclusive OR : x XOR $y = x \oplus y$
- Exclusive OR : x XOR $y = x \oplus y$

WARNING : + and . are not operations modulo 2 here

And the related truth table,

x	y	$x \cdot y$	$x + y$	$x \oplus y$
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	1	1	0

Table – Truth table for basic binary operations on binary variables

Boolean algebra satisfies many of the same laws as ordinary algebra when one matches up with addition and with multiplication. In particular the following laws are common to both kinds of algebra :

- Associativity of $\vee$: $x \vee (y \vee z) = (x \vee y) \vee z$
- Associativity of $\wedge$: $x \wedge (y \wedge z) = (x \wedge y) \wedge z$
- Commutativity of $\vee$: $x \vee y = y \vee x$
- Commutativity of $\wedge$: $x \wedge y = y \wedge x$
- Distributivity of $\wedge$ over $\vee$: $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
- Identity for $\vee$: $x \vee 0 = x$
- Identity for $\wedge$: $x \wedge 1 = x$
- Annihilator for $\wedge$: $x \wedge 0 = 0$
- Double Negation : $\neg(\neg x) = x$
- Complementation : $x \vee \neg x = 1$ and $x \wedge \neg x = 0$

The following does not hold in a Boolean algebra :

$$\begin{aligned}(-x)(-y) &= xy \\ (-x)+(-y) &= -(x+y)\end{aligned} \tag{4}$$

Insted, we have the De Morgan laws :

$$\begin{aligned}&\text{De Morgan 1} \quad \neg x \wedge \neg y = \neg(x \vee y) \\ &\text{De Morgan 2} \quad \neg x \vee \neg y = \neg(x \wedge y)\end{aligned} \tag{5}$$

We also have the Consensus Theorem : for $x, y, z \in \mathcal{B}$

$$xy \vee \bar{x}z \vee yz = xy \vee \bar{x}z \tag{6}$$

As an exercice, let's prove the following :

- De Morgan laws :

$$\begin{aligned} &\text{De Morgan 1} \quad \neg x \wedge \neg y = \neg(x \vee y) \\ &\text{De Morgan 2} \quad \neg x \vee \neg y = \neg(x \wedge y) \end{aligned}$$

- Consensus Theorem : for $x, y, z \in \mathcal{B}$

$$xy \vee \bar{x}z \vee yz = xy \vee \bar{x}z$$

Contents

Outline

A problem

Problem

A tank is filled by 2 valves : V_1 and V_2. We consider 3 levels :

- Warning (W)
- Bottom (B)
- top (T)

When the level is below W, V_1 et V_2 are opened.
When the level is between W and B, only V_1 is opened.
When the level is between B and T, only V_2 is opened.
When the level is greater than T, we close the pipe.

How to deal with it ?

- Make a drawing (visualization)
- Make the truth table

Reframing the problem

Problem

A tank is filled by 2 valves : V_1 and V_2. We consider 3 levels :

- Warning (W)
- Bottom (B)
- top (T)

When the level is below W, V_1 et V_2 are opened.
When the level is between W and B, only V_1 is opened.
When the level is between B and T, only V_2 is opened.
When the level is greater than T, we close the pipe.

Analysis

We can model the state of the valves and the state of the sensors by binary variables. Here, the sensors are controlling the valves : the sensors are the input of a system whose outputs are the valve states.
Note : not all states are physically attainable.

Towards a truth table

w	b	t	v_1	v_2
0	0	0	1	1
0	0	1	X	X
0	1	0	X	X
0	1	1	X	X
1	0	0	1	0
1	0	1	X	X
1	1	0	0	1
1	1	1	0	0

Table – Truth table for the introductory valve problem

When the output is not physically meaningful, the X symbol (called "do not care") is used.
Naively, we have :

$$\begin{aligned} v_1 &= \overline{wbt} + w\overline{bt} \\ v_2 &= \overline{wbt} + wb\bar{t} \end{aligned} \tag{7}$$

Outline

Définition

For a Boolean algebra B, a minterm is a function from B^n to$\mathrm{B}$ such that it exists an unique $(a_1, \cdots, a_n) \in \mathrm{B}^n$ with :

$$minterm(a_1, \cdots, a_n) = 1 \tag{8}$$

Hands-on interpretation

A minterm is a Boolean expression resulting in 1 for the output of a single input, and 0s otherwise.

Maxterm

Définition

For a Boolean algebra $\mathbb{B}$, a maxterm is a function from $\mathbb{B}^n$ to $\mathbb{B}$ such that it exists an unique $(a_1, \cdots, a_n) \in \mathbb{B}^n$ with :

$$maxterm(a_1, \cdots, a_n) = 0 \tag{9}$$

Hands-on interpretation

A maxterm is a Boolean expression resulting in 0 for the output of a single input, and 1s otherwise.

e_1	e_2	e_3	Minterm	Maxterm
0	0	0	$\bar{e}_1\bar{e}_2\bar{e}_3$	$e_1 + e_2 + e_3$
0	0	1	$\bar{e}_1\bar{e}_2 e_3$	$e_1 + e_2 + \bar{e}_3$
0	1	0	$e_1\bar{e}_2\bar{e}_3$	$e_1 + \bar{e}_2 + e_3$
0	1	1	$\bar{e}_1 e_2 e_3$	$e_1 + \bar{e}_2 + \bar{e}_3$
1	0	0	$e_1\bar{e}_2\bar{e}_3$	$\bar{e}_1 + e_2 + e_3$
1	0	1	$e_1\bar{e}_2 e_3$	$\bar{e}_1 + e_2 + \bar{e}_3$
1	1	0	$e_1 e_2\bar{e}_3$	$\bar{e}_1 + \bar{e}_2 + e_3$
1	1	1	$e_1 e_2 e_3$	$\bar{e}_1 + \bar{e}_2 + \bar{e}_3$

Table – Minterms and maxterms in a 3 value case

We derived previously this pair of equations :

$$
\begin{aligned}
v_1 &= \overline{wbt} + w\overline{bt} \\
v_2 &= \overline{wbt} + wb\bar{t}
\end{aligned}
\tag{10}
$$

This set of equations for the valve control example is the minterm expression of the system, where the "do not care" condition is set to 0.
However, we don't know if this is optimal.

Outline

LSB and MSB

Définition

In computing, the LSB (least significant bit) is the bit corresponding to the lowest power of 2 in the binary decomposition of the integer.
Similarly, the most significant bit (MSB) is the bit corresponding to the highest power of 2 in the binary decomposition of the integer (bound to a 1).

Exemple

For 1001_b, the MSB is 1 and the LSB is 1. For 1000_b, the MSB is 1 and the LSB is 0. For 0110_b, both the MSB and the LSB are 0.

Gray Code

Définition

The reflected binary code (RBC), also known as reflected binary (RB) or Gray code (after Frank Gray), is an ordering of the binary numeral system such that two successive values differ in only one bit (binary digit).

Property

The Hamming distance between two consecutive words is 1.

Decimal	Binary	Gray	Decimal of Gray
0	0000	0000	0
1	0001	0001	1
2	0010	0011	3
3	0011	0010	2
4	0100	0110	6
5	0101	0111	7
6	0110	0101	5
7	0111	0100	4
8	1000	1100	12
9	1001	1101	13
10	1010	1111	15
11	1011	1110	14
12	1100	1010	10
13	1101	1011	11
14	1110	1001	9
15	1111	1000	8

Table – Gray Code, 4-bit case

Karnaugh Map

Définition

A Karnaugh map is a diagram consisting of a rectangular array of squares each representing a different combination of the variables of a Boolean function.

Property

A Karnaugh map uses the Gray code to label its rows and columns : two adjacent squares differs by the complementation of one and only one variable.

		b	
		0	1
a	0	0	1
	1	1	0

Figure – Karnaugh map for $f = a \oplus b$

Karnaugh maps and simplification of expressions

The following map is for the function $f = \bar{x}\bar{y}z + x\bar{y}\bar{z} + x\bar{y}z + xy\bar{z} + xyz$

$x \backslash yz$	00	01	11	10
0	0	1	0	0
1	1	1	1	1

By finding groups of 2 adjacent 1s, we can simplify. For instance the green square represent an expression independant of x, $\bar{y}z$. Similarly, the red rectangle is bound to an expression independant of y and z, so x.
All in all, we have $f = x + \bar{y}z$

Contents

Outline

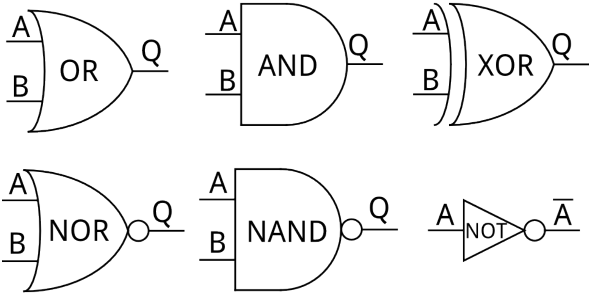

Figure – Logic Gates, ANSI style

Property

Every logic circuit can be written with only NAND (Not And) gates. Every logic circuit can be written with only NOR (Not Or) gates.

Note : NOT can be seen as a NAND or a NOR gate whose inputs are the same.

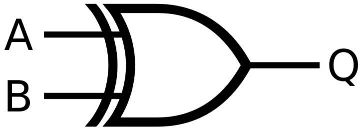

Figure – XOR gate

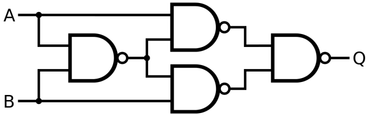

Figure – XOR gate, NAND representation

For circuits with a large number of variables, the synthesis using the previously mentioned method does not provide a Boolean expression.
In this case, the complex problem should be broken down into smaller (most likely identical) simple problems.

Let us consider a circuit, having two numbers x and y, coded on n bits as its inputs and whose outputs are $S_x = X > Y$ and $S_y = Y > X$.
We can break down the problem in a way where each bit of the inputs can be compared.
If we denote, a_i and b_i the cascaded inputs, α_i and β_i the cascaded outputs and x_i,y_i the i-th bit of x and y respectively, we have :

$$\begin{aligned} \alpha_i &= a_i + b_i x_i y_i \\ \beta_i &= b_i + a_i y_i x_i \\ a_{n-1} &= b_{n-1} = 0 \end{aligned} \tag{11}$$

Exercice

Find the NAND representation of such a block.

Outline

Définition

A decoder is a combinational circuit that has n input lines and a maximum of 2^n output lines. One of these outputs will be active High based on the combination of inputs present, when the decoder is enabled. That means the decoder detects a particular code. The outputs of the decoder are nothing but the min terms of n input variables lines, when it is enabled.

Exemple (74LS146)

1 :4 (1 to 4) decoder means 2 bits as its input and 4 output lines. 00_b is mapped to output 0, 01_b is mapped to output 1 and so on and so forth.

Encoders/Multiplexers

Définition

An Encoder is a combinational circuit that performs the reverse operation of the Decoder. It has a maximum of 2^n input lines and n output lines. It will produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2^n input lines with n bits.

Définition

A Multiplexer is a combinational circuit that has maximum of 2^n data inputs, n selection lines and a single output line. One of these data inputs will be connected to the output based on the values of selection lines.
Since there are *s*election lines, there will be 2^n possible combinations of zeros and ones. So, each combination will select only one data input. Multiplexer is also called as "Mux".

Définition

Tri-state logic allows an output or input pin/pad to assume a high impedance state (noted Z), effectively removing the output from the circuit, in addition to the 0 and 1 logic levels.

Three-state buffers can also be used to implement efficient multiplexers, especially those with large numbers of inputs. To avoid floating pins, pull-up or pull-down resistors might be added.
Three-state buffers are essential to the operation of a shared electronic bus.

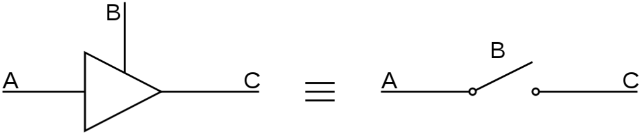

Figure – Tri-state buffer, effectively equivalent to a controlled switch

Outline

4 Applications to logic gates and circuits
- Logic gates
- Common logic functions
- Arithmetic Circuits

Representation of numbers

With n Boolean variables, we have 2^n possibilities. The most useful representations are :

- Unsigned integers
- Signed integers
- Two's complement

The 2^n combinations are used to represents integers from 0 to $2^n - 1$.
An operation is not guaranteed to be within the range of values. This case is called overflow.
Let us consider the following example to see how the overflow is handled. On 4 bits, the maximum encoded unsigned integer is 15.
If we compute $5 + 13$, the results on 4 bits should be $0010_b = 2$. The overflow will be signaled by a supplementary carry bit.

Signed Integers - Sign–magnitude

The simplest representation uses the most significant bit as a sign bit, 0 for positive number and 1 for negative numbers. The $n - 1$ other bits represent the absolute value of the number as an unsigned integer.

A problem with "naive" signed integers is we have to treat the sign bit before interpreting the rest of the number.
A number $0 \leq m < 2^{n-1}$ coded on n bits will be represented just like before.
For $2^{n-1} \leq m < 0$, m will be represented as $2^n - |m|$.

Exemple

On four bits signed, $5 = 0101$ and $-5 \rightarrow 1011$ $(2^4 - 5) = 11 > 2^3 = 8$

To add two number a, b coded in two's complement and noted α, β, we have four cases :

- $a, b > 0 : \alpha = a, \beta = b, \alpha + \beta = a + b$
- $a, b < 0$ and $|b| \leq a : \alpha = a, \beta = 2^n - |b|, \alpha + \beta = 2^n + a - |b| \equiv a - |b|$
- $a, b < 0$ and $|b| > a : \alpha = a, \beta = 2^n - |b|, \alpha + \beta = 2^n - (|b| - a) \equiv a - |b|$
- $a, b < 0$:
  $\alpha = 2^n - |a|, \beta = 2^n - |b|, \alpha + \beta = 2^n + 2^n - (|a| + |b|) \equiv 2^n - (|a| + |b|)$
  which is the representation of $-(|a| + |b|)$

Remark : for a number a, its negative in two's complement is the inversion of all the bits $+1$.
Proof : If $a \geq 0$, $\bar{a} = 2^n - |a| - 1, \bar{a} + 1 = 2^n - |a| \equiv -a$, if $a < 0$, similar.

Exemple

$-5 = 1011, \overline{-5} = 0100, \overline{-5} + 1 = 0101 \equiv 5$
$5 = 0101, \overline{5} = 1010, \overline{5} + 1 = 1011 \equiv -5$

Full Adder

- Ports : Inputs A,B ; Output S (Sum), $C_{in/out}$ (Carry)
- $S = A \oplus B \oplus C_i, \; C_{out} = (AB) + (C_{in}(A \oplus B))$

Exercise

Proof of the boolean equations above.

A	B	Cin	Cout	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 – Truth table of full adder

Arithmetic logic unit

Example : 74181 IC = `https://www.ti.com/lit/ds/symlink/sn54s181.pdf`

- Ports : Inputs A,B ; Output Y
- Opcode : which operations to do = sum, difference, logic etc...
- Status : carry bit, parity, overflow...

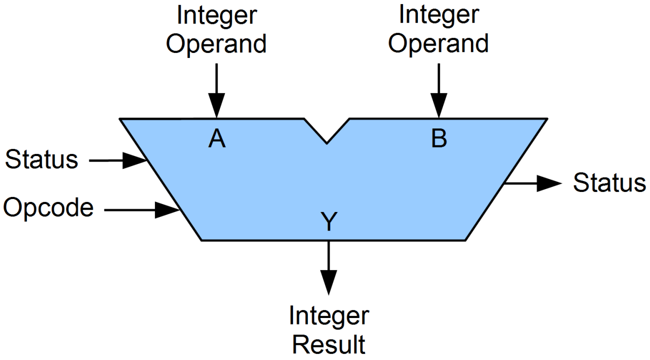

Figure – Arithmetic logic unit

Contents

Conclusion

What we have seen :

- Boolean algebra
- Operations on Boolean algebras
- Karnaugh map for logic equation simplification
- Logic gates
- Common logic circuits