Chapter 03 Part 03
Coding of information

Coding of information (text, sound, image, video, etc).

Coding of information

- Concept of Information Encoding
- Information is encoded in binary and exclusively in binary.
- This is why we covered the binary numbering system in the first chapter.
- However, this alone is not sufficient: it is necessary to be able to encode numbers, text, images, sound, videos, and various other computer objects.

Coding of information

- **II.1 –** Binary Encodings
- Definition: a code is a set of symbols (such as the alphabet of a language) representing useful information.
- In computing, these symbols are limited to the two objects, namely '0' and '1'. Therefore, in this field, all information is represented in the form of binary configurations. Whether it is text, images, sound, video, or simply numbers, binary encoding is used.

Pure Binary

- Pure binary is also referred to as natural binary. This encoding has already been discussed in the context of Chapter 1 addressing numbering systems. In this encoding, each positive integer is associated with its corresponding value according to the binary numbering system. Thus, with n bits, we can encode values ranging from [0 et 2^n].
- Exemple :

  With 6 bits : $(35)_{10} = (100011)_2$

Gray Code

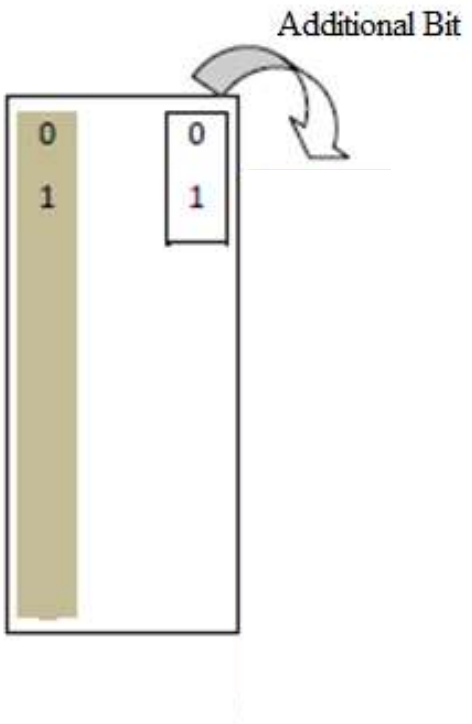
- Gray Code
- Gray code is used when a binary numerical progression without transitory problems is desired.
- It is also employed in Karnaugh maps used during the design of logical circuits.
- It is constructed in such a way that starting from the digit 0, each consecutive number differs from the immediate previous one by only one digit (bit).

Gray Code

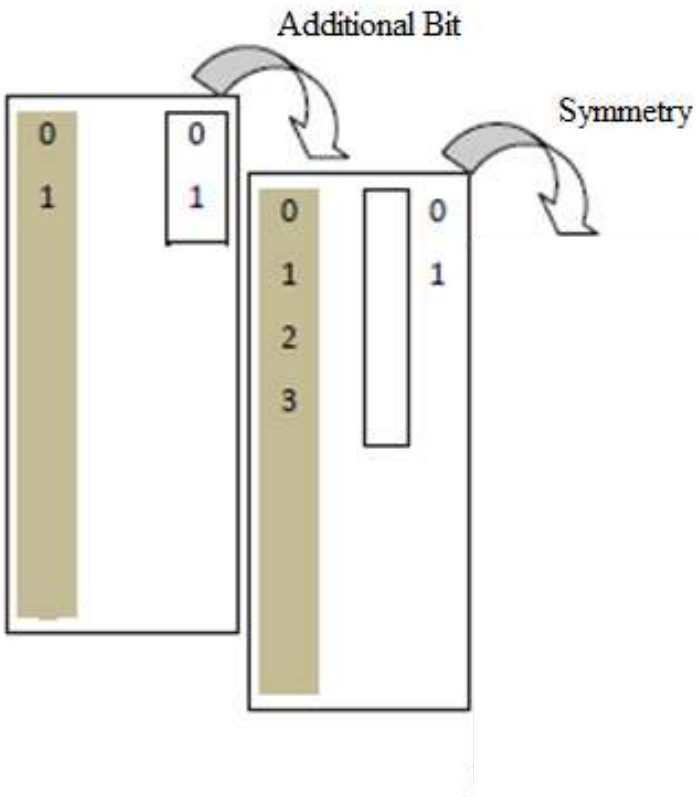
- One of the methods for constructing Gray code is called the Reflected Binary Code or REFLEX code method. Here is its principle:

1. Establish an initial code: zero is coded as 0, and one is coded as 1.
2. Then, each time an additional bit is needed,
3. Reflect the numbers already obtained (like a reflection in a mirror).
4. Add a 1 to the beginning of the new numbers and a zero to the old ones.

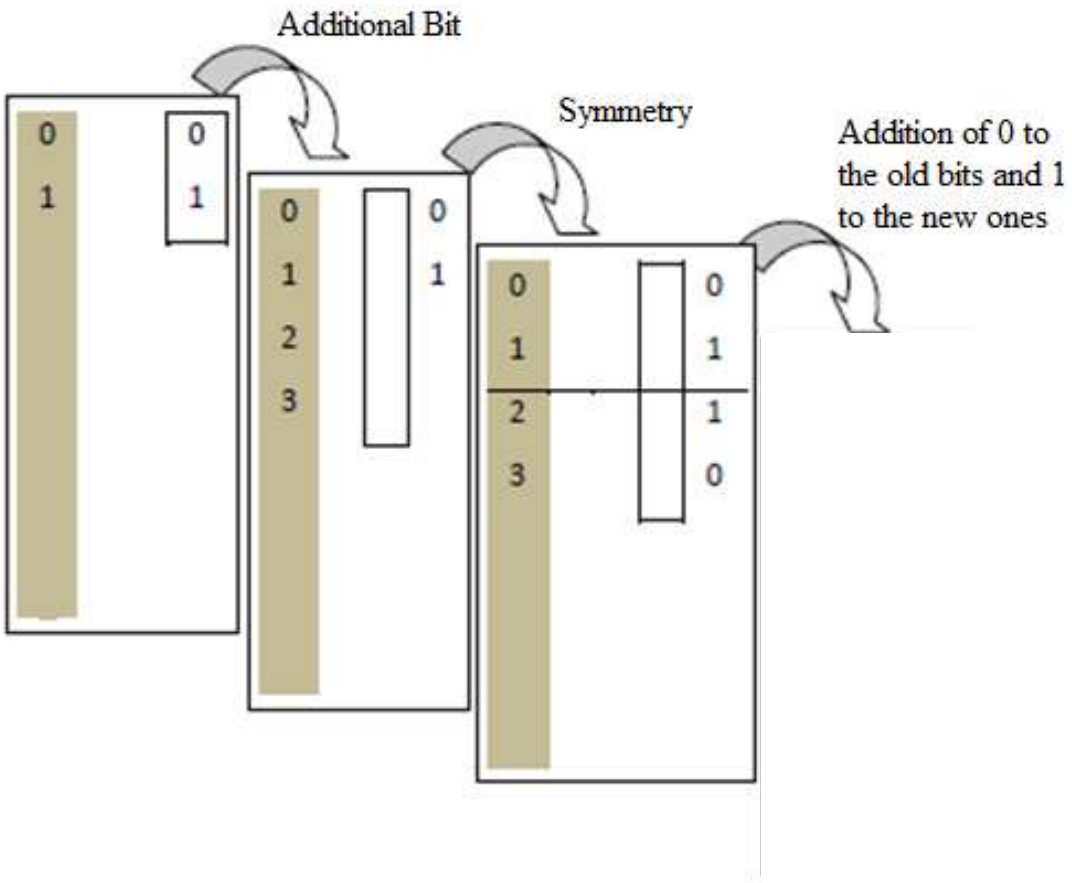
Gray Code



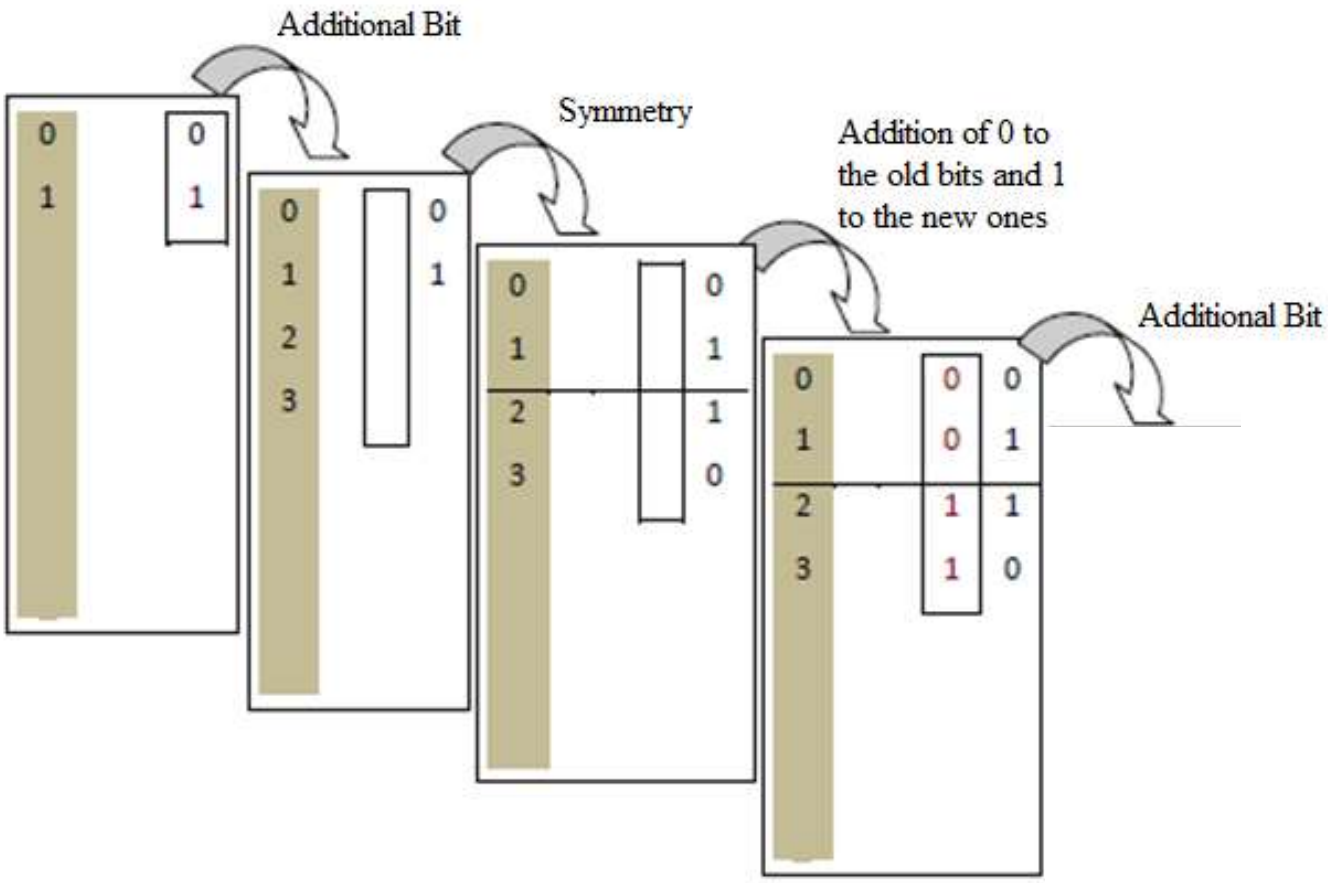
Gray Code



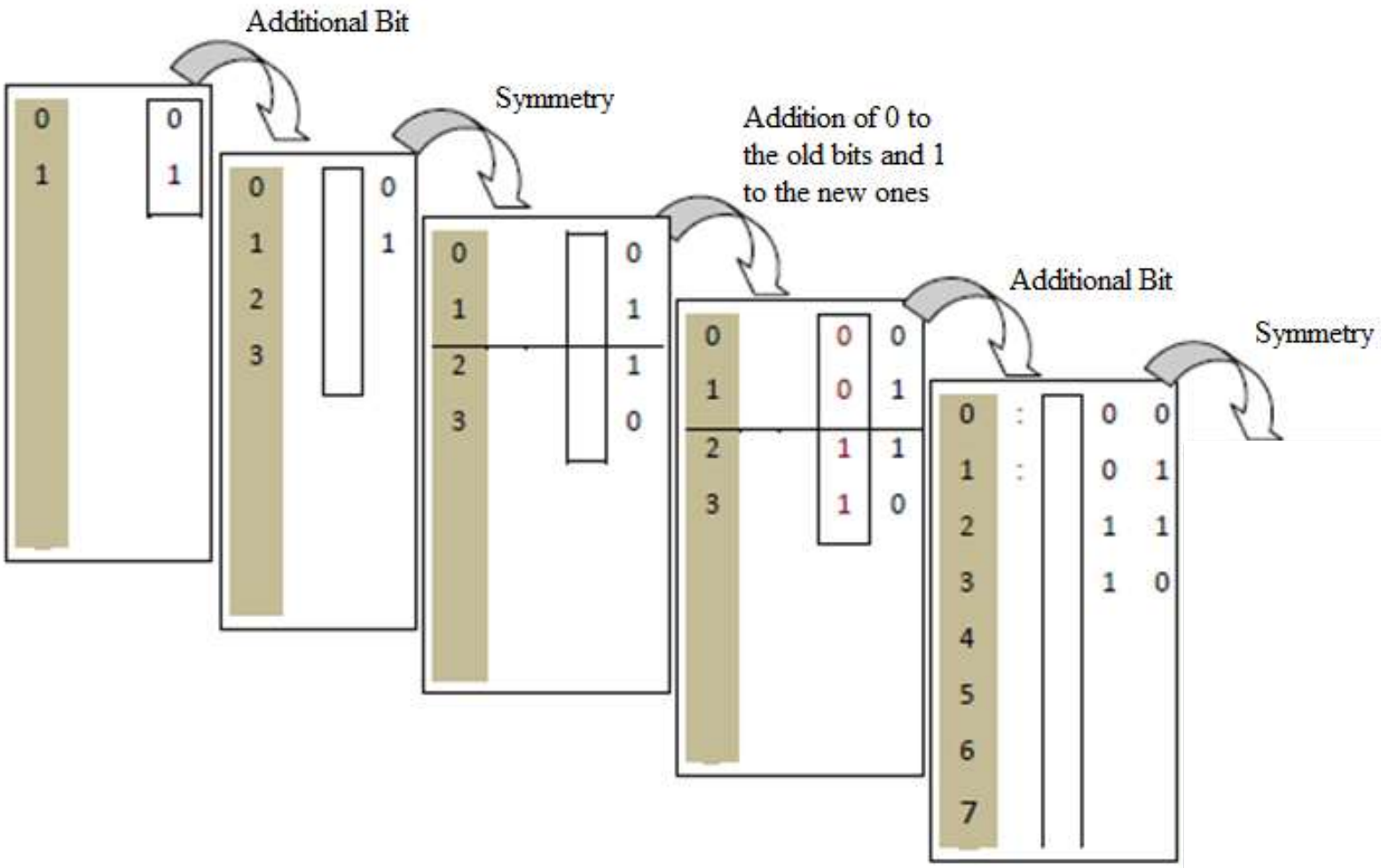
Gray Code


Additional Bit
Symmetry
Addition of 0 to
the old bits and 1
to the new ones
0
1
0
1
0
1
2
3
0
1
0
1
0
1
2
3
1
0

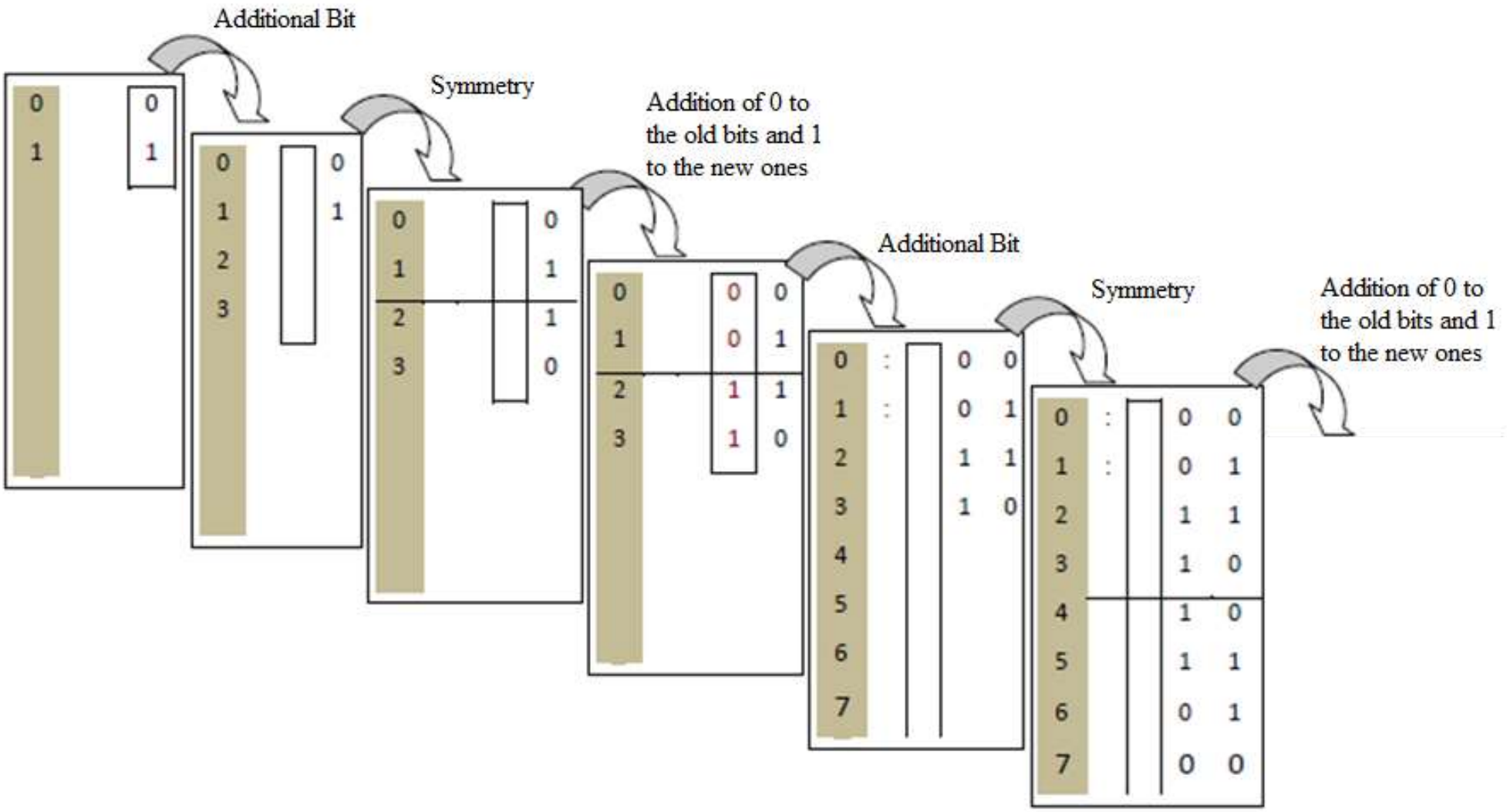
Gray Code


Additional Bit
Symmetry
Addition of 0 to
the old bits and 1
to the new ones
Additional Bit
0
1
0
1
0
1
2
3
0
1
0
1
0
1
2
3
0
1
0
1
0
1
2
3
0
0
0
1
1
1
1
0

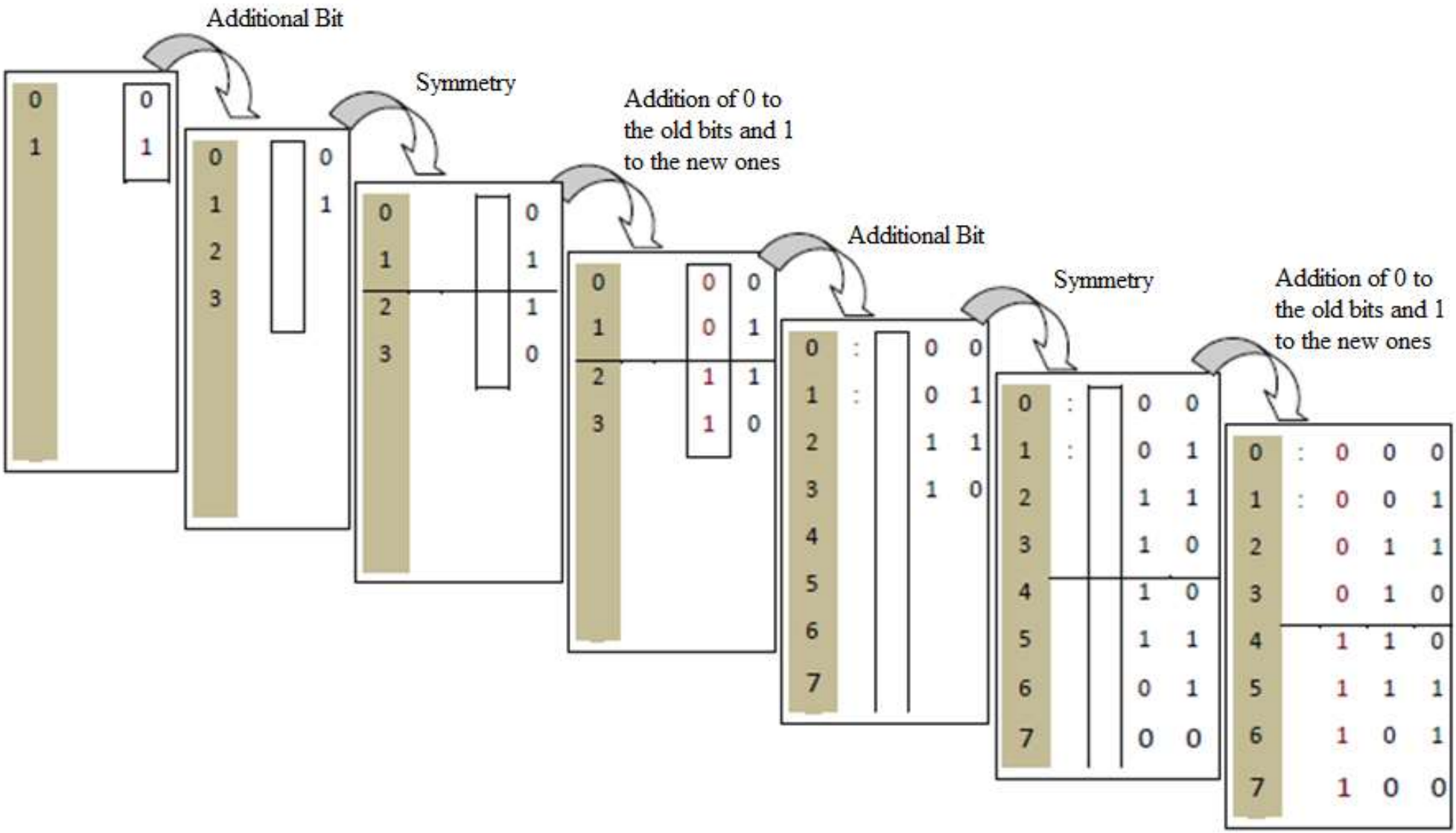
Gray Code


Additional Bit
Symmetry
Addition of 0 to
the old bits and 1
to the new ones
Additional Bit
Symmetry

Gray Code


Additional Bit
Symmetry
Addition of 0 to the old bits and 1 to the new ones
Additional Bit
Symmetry
Addition of 0 to the old bits and 1 to the new ones

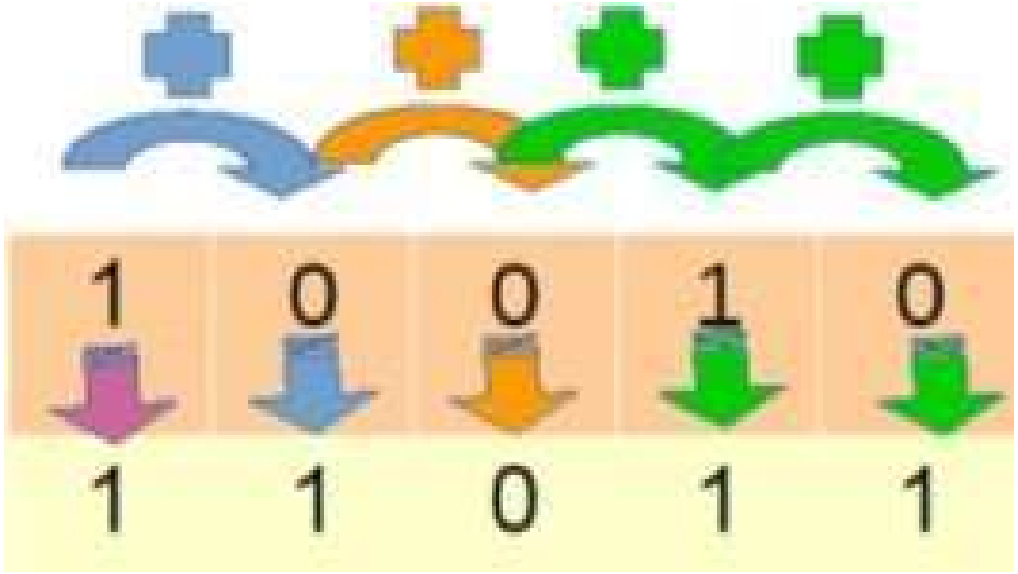
Gray Code


Additional Bit
Symmetry
Addition of 0 to the old bits and 1 to the new ones
Additional Bit
Symmetry
Addition of 0 to the old bits and 1 to the new ones

Gray Code

- There are other methods for calculating Gray code. The first bit on the left remains the same.

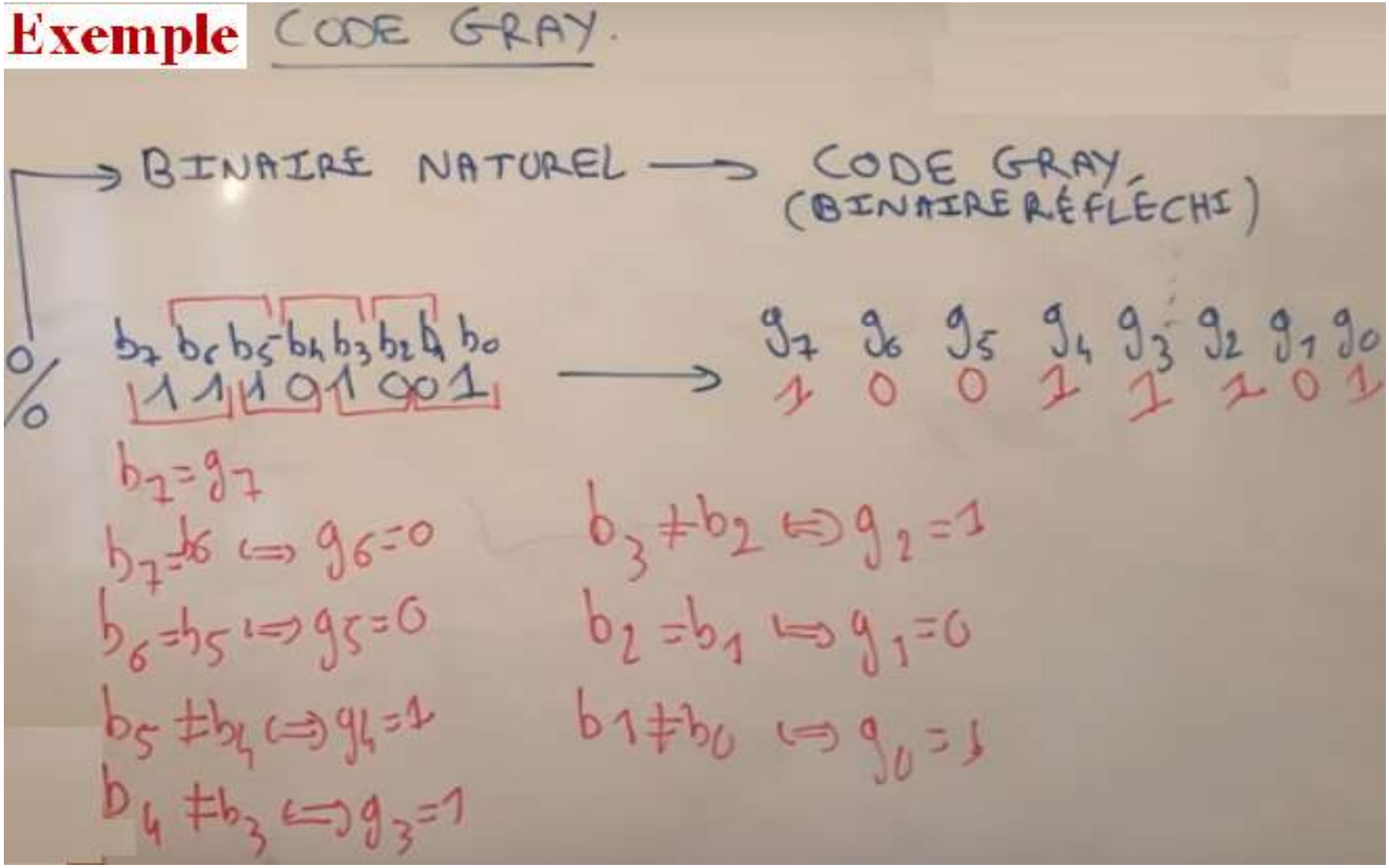
Then, from left to right, sum the adjacent bits without carryover.



Exemple:

$(10010)_2=(11011)_{Gray}$

Gray Code


Exemple
CODE GRAY.
BINAIRE NATUREL
CODE GRAY
(BINAIRE RÉFLÉCHI)
%
b7 b6 b5 b4 b3 b2 b1 b0
1 1 1 0 1 0 0 1
g7 g6 g5 g4 g3 g2 g1 g0
1 0 0 1 1 1 0 1
b7 = g7
b7 ≠ b6 ⇔ g6 = 0
b6 = b5 ⇔ g5 = 0
b5 ≠ b4 ⇔ g4 = 1
b4 ≠ b3 ⇔ g3 = 1
b3 ≠ b2 ⇔ g2 = 1
b2 = b1 ⇔ g1 = 0
b1 ≠ b0 ⇔ g0 = 1

Binary Coded Decimals

- The information processed by the computer is not always converted according to the binary numbering system. Indeed, this information can be manipulated in the form of binary coded decimal (BCD).
- Several codes are defined for this purpose:
  - BCD (Binary Coded Decimal) or DBC (Decimal Coded Binary) or also 8421 code
  - Excess-3 code
  - 2 out of 5 code (or parity code)

1 - BCD Code

- BCD (Binary Coded Decimal)
- is the most commonly used decimal code in electronics.
- It contains code words that represent the natural binary translation (on 4 bits) of each of the ten decimal digits.
- Principle: Associate a binary code (4 bits) with each decimal digit.

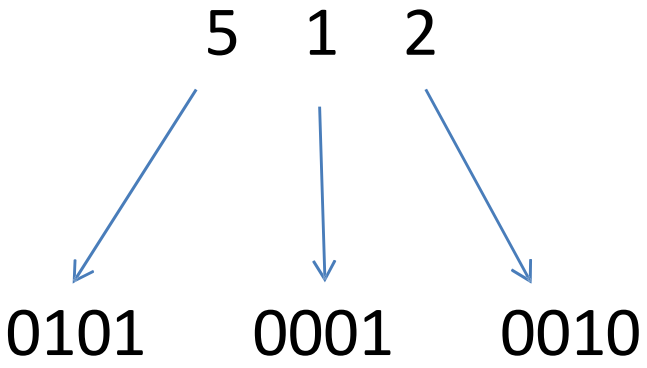
1 - BCD Code

- Exemple :

We have the number 512 in decimal, which is equivalent to : $(512)_{10} = (1000000000)_2$.

In BCD code, this number will be encoded as follows : $(512)_{10} = (0101\ 0001\ 0010)_{BCD}$

The code conversion is established as follows :



1 - BCD Code

We can observe that there are unused binary configurations. Since the number of configurations that can be represented with 4 bits is 16, we deduce that there are 6 unused configurations (between 10 and 16).

decimal digit	BCD code	
0	0000	
1	0001	
2	0010	
3	0011	
4	0100	
5	0101	
6	0110	
7	0111	
8	1000	
9	1001	
/	1010	unused
/	1011	
/	1100	
/	1101	
/	1110	
/	1111	

1 - BCD Code

Le code BCD ou DCB

Poids binaires 8	4	2	1	Dec	Hex
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	8
1	0	0	1	9	9
1	0	1	0	10	A
1	0	1	1	11	B
1	1	0	0	12	C
1	1	0	1	13	D
1	1	1	0	14	E
1	1	1	1	15	F

Zone BCD

Décimal
Codé en
Binaire

La valeur ne doit pas dépasser 9 en décimal ou 1001 en binaire. Sinon il s'agirait d'un code hexadécimal !

1 - BCD Code

- Note
- For addition in BCD code, if there is a carry where the result does not belong to BCD (the result is > 9), we must add $6=(0110)_2$

- **If** the result does not contain any unauthorized configurations (between 10 and 16), then it is correct.
- **Else**, the value 6 (0110) must be added to the result for each unauthorized configuration.

1 - BCD Code

- **Exemple:** perform the following addition : 16+25

	16	0001 0110
+	25	0010 0101

1 - BCD Code

- **Exemple:** perform the following addition : 16+25

```
    16          0001 0110
+   25          0010 0101
=               0011 1011    > 9
```

1 - BCD Code

- **Exemple:** perform the following addition : 16+25

	16	0001 0110	
+	25	0010 0101	
=		0011 1011	> 9
	+	0110	+6
		0100 0001	

1 - BCD Code

- **Exemple:** perform the following addition : 137+99

1 - BCD Code

- **Exemple:** perform the following addition : 137+99

$$
\begin{array}{rl}
137 = & 0001 \mid 0011 \mid 0111 \\
+\ 99 = & +\ 0000 \mid 1001 \mid 1001
\end{array}
$$

1 - BCD Code

- **Exemple:** perform the following addition : 137+99

$$
\begin{array}{rl}
137= & 0001 \mid 0011 \mid 0111 \\
+\quad 99 \ + & \underline{0000 \mid 1001 \mid 1001} \\
 & 0001 \mid 1101 \mid 0000
\end{array}
$$

1 - BCD Code

- **Exemple:** perform the following addition : 137+99

$$
\begin{array}{rl}
137= & 0001\,|\,0011\,|\,0111 \\
+\quad 99\ + & \underline{0000\,|\,1001\,|\,1001} \\
 & 0001\,|\,1101\,|\,0000 \\
 & \quad +\ \underline{0110\,|\,0110} \\
 &
\end{array}
$$

1 - BCD Code

- **Exemple:** perform the following addition : 137+99

$$
\begin{array}{rl}
137 = & 0001 \mid 0011 \mid 0111 \\
+\ 99 = & +\ 0000 \mid 1001 \mid 1001 \\
\hline
 & 0001 \mid 1101 \mid 0000 \\
 & +\ 0110 \mid 0110 \\
\hline
 & 0010 \mid 0011 \mid 0110 \\
= & \quad 2 \qquad 3 \qquad 6
\end{array}
$$

Excess-3 Code

This code closely resembles the BCD code.
The principle is based on the idea
of associating each decimal digit
with its binary equivalent added to **3**.

- **Exemple :**

512 in decimal is equal to

(0101 + 0011) (0001 + 0011) (0010 + 0011)

resulting in (1000 0100 0101) Excess-3.

The following table provides

the Excess-3 equivalent of decimal digits.

decimal digit	BCD Code	Excess-3 Code
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Excess-3 Code

- Note
- In the Excess-3 code: For addition,
- If there is a carry, we must add **3 $(0011)_2$**
- Else, we subtract **3 $(-0011)_2$**

45 =	0011 |	0111 |	1000	
+ 90	+ 0011 |	1100 |	0011	
	0111 |	0011 |	1011	
	-0011 |	+0011 |	- 0011	
	= 0100 |	0110 |	1000	
	= 1	3	5	

137 =	0100 |	0110 |	1010
+ 90	+ 0011 |	1100 |	1100
	1000 |	0011 |	0110
	-0011 |	+0011 |	+ 0011
	= 0101 |	0110 |	1001
	= 2	3	6

2 out of 5 Code

This code is, in fact, an error-correcting code. We arrange to encode the digits on 5 bits, but ensuring that each binary configuration has only two bits set to 1. The following table provides the equivalent of decimal digits in the 2 out of 5 code:

dicimal digit	BCD Code	Excess-3 Code	2 out of 5 Code
0	0000	0011	00011
1	0001	0100	00101
2	0010	0101	00110
3	0011	0110	01001
4	0100	0111	01010
5	0101	1000	01100
6	0110	1001	10001
7	0111	1010	10010
8	1000	1011	10100
9	1001	1100	11000

Representation of characters

- In fact, the goal is to encode the letters of the alphabet, including numbers and keyboard control characters.
- The choice of a code depends on the country and language.
- Various codes are used:
- ASCII (7 bits),
- Extended ASCII (8 bits),
- Unicode (16 bits), etc.

Representation of characters

- The ASCII Code: is a character encoding standard - American Standard Code for Information Interchange. It is a 7-bit encoding, providing 128 different codes:
  - 0 to 31: control characters (line feed, etc.)
  - 65 to 90: uppercase letters
  - 97 to 122: lowercase letters
- In this code, accented characters (é, à, etc.) are not represented, which is why it is limited to a few languages (especially English)

Representation of characters

bin	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0001	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0110	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

- The characters marked with a blue background cannot be displayed; they are control characters. To read the table, associate the character with the following binary code: row number_column number. For example, the code for 'a' is: 01100001 $=(97)_{10}$

Representation of characters

- The need to represent texts containing characters not present in the ASCII table, such as those from the Latin alphabet used in French like 'à,' 'é,' or 'ç,' requires the use of a different encoding than ASCII.
- Extended ASCII: To represent more languages, especially Western languages like French, the code has been extended from 7 to 8 bits.
- Unicode Encoding: This is a 16-bit encoding that allows the representation of all characters specific to various languages. It is increasingly being adopted.

Representation of characters

- UTF-8 Encoding In practice, for each letter, it occupies 2 bytes (16 bits).
- This is wasteful because several languages share the same letters (French, English, etc.), especially unaccented French alphabet letters.
- To optimize this, we use UTF-8: A text in UTF-8 is simple: it is entirely in ASCII, and whenever a character from Unicode is needed, a special character is used to signal, 'attention, the following character is in Unicode.

Sound Encoding

- Sound is naturally represented in the form of an analog signal. The encoding of such a signal involves representing it in binary to transmit it to the computer, which can only store and process information in binary.
- The analog signal of sound can be determined by a continuous function represented by frequency and amplitude.
- To convert from an analog signal to a binary representation, this process is called sound digitization,

Sound Encoding

- Sampling involves capturing the amplitude value of the analog signal at constant repetitive time intervals. The amplitude value is encoded with n bits at each sampling period. The sampling period is crucial as it has an impact on the fidelity of the digitized sound. It must be set in such a way that human perception does not detect differences between the original signal and the digitized one.

Sound Encoding

- Sampling

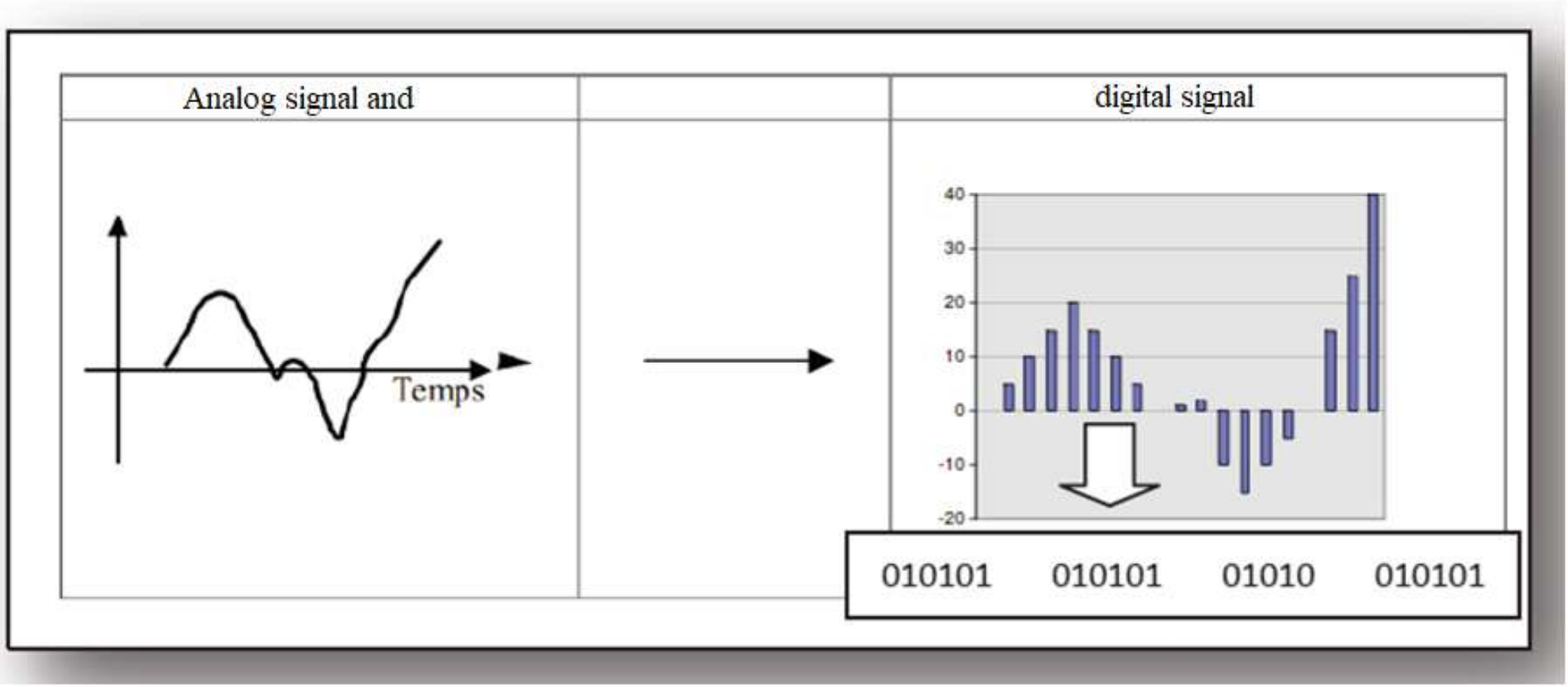


Image Encoding

- There are two categories of image encoding:
- Vector images:

The image is encoded using a set of mathematical formulas.

- Bitmap (Raster) images:

The image is encoded as an array of points. This encoding is used to digitally represent photos.

Image Encoding

- The image is described point by point.
- The points of an image are called pixels.
- Each pixel is described by a number indicating its color.
- The image is therefore represented by a series of numbers.
- The encoding of the image is done by successively writing the bits corresponding to each pixel, line by line, starting from the bottom-left pixel. The encoding is simple, but the bitmap image occupies a lot of memory: the smaller the pixels, the more numerous they are! This explains the need for compression.

Image Encoding

- Three parameters define a bitmap image:
  1. The number of columns
  2. The number of rows
  3. The number of colors per pixel
- The first two parameters determine what is called the resolution of the image. For example, 800x600 pixels.
- The last parameter determines what is called the profundity of the image. It defines the colors of the image pixel by pixel.

Image Encoding

- How to encode color?
- Based on the three primary colors: Red (R), Green (G), and Blue (B), abbreviated as RGB.
- From these three primary colors, all other colors can be generated, ranging from black to white and everything in between. In terms of storing and processing pixel colors, a fixed number of bits is assigned to each of the red, green, and blue colors.

Video Encoding

- A video is a sequence of images played at a certain rate. These images can be of different natures and formats, including drawings, graphics, and photos. Animations can have different characteristics depending on whether they come from a series of photos or drawings, accompanied by an audio sequence or not. Videos consist of two synchronized parts: an audio part and a video part (images).

Video Encoding

- The number of frames to be displayed per second.
- When it is >10, an animation effect is visible.
- Below 10 frames per second, the animation will appear jerky.
- At 25 frames per second, the animation approaches reality, and the human eye will not be able to see imperfections in the animation.
- In cinema, the standard frame rate is 24 frames per second. In television, the European PAL system (or SECAM in France) operates at 25 frames per second. In the United States and Japan, the NTSC standard is 30 frames per second.