# Digital Systems

**Numbering Systems, Data Representation, Boolean Algebra, Basic Ciruits**

V.2.4 V.Lobo 2021

## Numbering Systems

Sistemas Digitais

- **DECIMAL**
  - SYMBOLS 0,1 .. 9
  - 1842 $\Rightarrow$ $1\times10^3 + 8\times10^2 + 4\times10^1 + 2\times10^0$
- **OCTAL**
  - SYMBOLS 0..7
  - 1634 $\Rightarrow$ $1\times8^3 + 6\times8^2 + 3\times8^1 + 4\times8^0$
- **HEXADECIMAL**
  - SYMBOLS 0.. 9,A,B,C,D,E,F
  - 5F1A0 $\Rightarrow$ $5\times16^4 + 15\times16^3 + 1\times16^2 + 10\times16^1 + 0\times16^0$
- **BINARY**
  - SYMBOLS 0,1
  - 10110 $\Rightarrow$ $1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$

The **POSITION** gives the importance, or **WEIGHT** to the digit

The **MOST SIGNIFICANT** digit (or bit)is on the left (**MSB**)
The **LEAST SIGNIFICANT** digit (or bit)is on the left (**LSB**)

V.Lobo @ EN

1

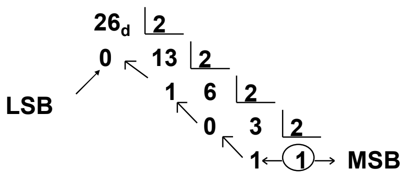
1

## Binary System

Sistemas Digitais

- **Importance of the Binary System**
  - Easy physical implementation
  - Implemented with hydraulic systems, electric systems, optical systems, etc.
- **Conversions:**
  - DECIMAL $\rightarrow$ BINARY
  - BINARY $\rightarrow$ DECIMAL

$26_d = 11010_b$




$1\,0\,1\,0\,0\,1\,1\,0_b = 166_d$

| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

+ + + 128+32+4+2=166

V.Lobo @ EN

2

2

## BASES THAT ARE POWERS OF 2



- **Bases that are powers of each other are easy to convert, since each digit of the higher power will only influence n digits of the other**
  - Octal — 1 octal digit = 3 binary digits
  - Hexadecimal — 1 hexa digit = 4 binary digits

```
      2       D       3
  1 0 1 1 0 1 0 0 1 1
 1    3     2     3
```

$2D3_H = 1011010011_b = 1323_{Oct.}$

- **Advantages**
  - They use less digits to represent a number
  - They are easier to read by human
  - They are widely used



## BINARY ARITHMETIC



- **Basically the same as what you know in decimal**
  - Numbers are summed digibit by digit
  - From one digit to the next (more significant) there my be a "CARRY"
  - 1 plus 1 is two ( i.e. $10_b$)
  - Example:

Adition →

$$\begin{array}{r} (11011)_2 \\ +\ (10011)_2 \\ \hline (101110)_2 \end{array} \qquad \begin{array}{r} (647)_{10} \\ +\ (537)_{10} \\ \hline (1184)_{10} \end{array}$$

Multiplication →

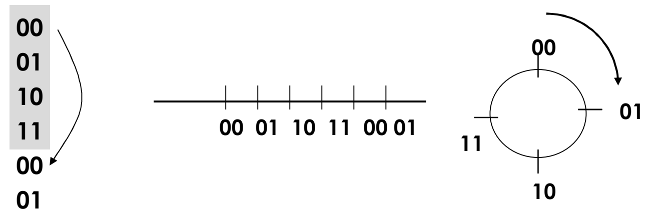
```
    1101          152
 ×   101        × 231
 -------        -----
    1101          152
   0000          456
  1101          304
 -------        -----
 1000001        35012
```

Only SHIFT, and COPY with a final sum !!!

## BINARY ARITHMETIC

Sistemas Digitais

- **In a machine, the number of digits is FINITE**
  - We cannot use as many digits as we want
    - →The number of digits is FIXED in a given machine
  - The is a MAXIMUM number that can be represented.




- **Consequence of the FINITE number of digits**
  - Numbers are NOT represented as a line, but as a CIRCLE !

V.Lobo @ EN

5

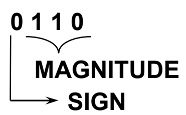
5

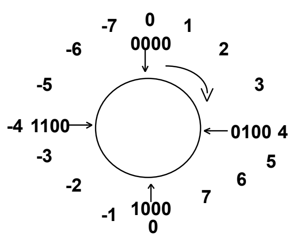
## REPRESENTING NEGATIVE NUMBERS

Sistemas Digitais

- **Problem:**
  - How can you represent a negative number without using the "-" symbol (*i.e.*using only 0 and 1)
  - Solution: use one of the POSTIONS to store the SIGN of the number
- **SIGN and MODULUS or magnitude** (signed integer)
  - The most significant bit is used for sign, the rest for magnitude
  - Sign = 0 => Positive (most "normal" case)
  - Sign = 1 => Negative
  - Exemples:

0100 = 4
1100 = -4
0010 = 2
1011 = -3




V.Lobo @ EN

6

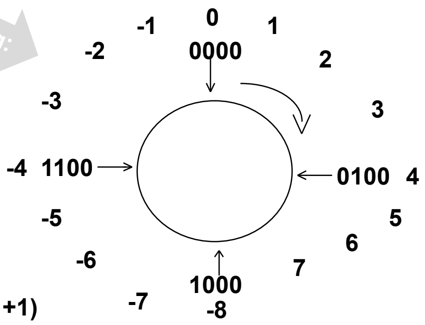
6

## 2's COMPLEMENT

Sistemas Digitais

- Basic idea
  - Make additions and subtractions easier

Base idea:




$N^{(2)} = M - N$

NUMBER 'N'

MODULUS (MÁX Nº of 1 +1)

NUMBER 'N' IN 2's COMPLEMENT

```
 100000
 -00110
 ~~1~~11010
```

Rule for computing 2's complemente
COMPLEMENT ALL DIGITS FROM THE 1st "1" onwards

00110 ($6_{10}$) $\Rightarrow$ 11010 ($-6_{10}$)

V.Lobo © EN

7

7

## 2's COMPLEMENT

Sistemas Digitais

- 2's complement
  - The MSB is used for sign (just as in SIGN+MODULUS)
  - Conversion positive/negative and vice-versa uses the 2's complemt transformation
  - Advantages
    - →You can rapidly see of a number is positive or negative
    - →There are no repeated numbers (no +0 and -0)
    - →Number -1 is right before 0 (good for counting)
    - →Additions and subtractions can use the standard algorithms

- Various algorithms for computing 2's complement
  1) Subtract the positive number from 10000.... ($2^N$)
  2) Start from the right, and leave all digits until the first "1". Leave that one, but change all bits from there onwards
  3) Complement all digits, and then sum 1.

V.Lobo © EN

8

8

## Non-Integer numbers

Sistemas Digitais

- Fixed point notation
  - Equivalent to a "shifted" integer
  - A pre-determined number of digits is used for the fractional part
  - Example:
    - →Given that a "2 fractional digit" fixed point notation is used
    - →2.5 = 1010
- Coefficients of the fractional part
  - Negative power of the base
  - $2^{-1}$ (=0.5) , $2^{-2}$ (=0.25), ….
- For very large or very small numbers: floating point

V.Lobo @ EN

9

9

## Floating point notation

Sistemas Digitais

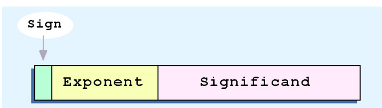
- Allow a very large RANGE
- They have less precision
- Representation:

Zero point….

$$+\ 0.43 \times 10^{+6} = 430000$$

Sign  Mantissa  Base  Exponent (w/sign)

- IEEE-754 standard (single precision floating point)
  - 32 bits – 1 sign bit, 8 exponent bits, 23 mantissa bits




V.Lobo @ EN

10

10

## OTHER BINARY CODES for NUMBERS

Sistemas Digitais

- Other notations, for specific purposes
  - To simplfy binary/decimal conversions
  - BCD - Binary coded decimal (natural, ou 8421)
    - →4 bits are used for each decimal digit
    - →6 positions are lost in each 16 binary numbers
  - Aiken (or 2421)
    - →The weights of the bits are 2421
    - →The unused binary number are "in the middle"
    - →Easy to see if a number is greater os smaller than 5
    - →It is "self-complementary"
  - Excess 3 (unweighted)
    - →Uses the "middle numbers" (3 to 13)
    - →It is "self-complementary"
  - 7421 – Minimizes the number of "1" used
    - →Reduces power consumption in certain systems

V.Lobo @ EN

11

11

## OTHER BINARY CODES for NUMBERS

Sistemas Digitais

| Dec. | BCD | AIKEN | EXC.3 | 7421 |
|---|---|---|---|---|
| 0 | 0000 | 0000 | 0011 | 0000 |
| 1 | 0001 | 0001 | 0100 | 0001 |
| 2 | 0010 | 0010 | 0101 | 0010 |
| 3 | 0011 | 0011 | 0110 | 0011 |
| 4 | 0100 | 0100 | 0111 | 0100 |
| 5 | 0101 | 1011 | 1000 | 0101 |
| 6 | 0110 | 1100 | 1001 | 0110 |
| 7 | 0111 | 1101 | 1010 | 1000 |
| 8 | 1000 | 1110 | 1011 | 1001 |
| 9 | 1001 | 1111 | 1100 | 1010 |

| Gray |
|---|
| 0000 |
| 0001 |
| 0011 |
| 0010 |
| 0110 |
| 0111 |
| 0101 |
| ... ... |
| ... ... |

- Gray code (reflected binary code)
  - Minimizes transitions (only 1 bit changes at each time)
  - Solves transient (or "spikes") due to transitions
    - →Physical converters (sensors)
  - It is a cyclic code
  - Can be seen as a numbering system or an method for ENUMERATION
  - Easy to convert to binary (XOR cascade)

V.Lobo @ EN

12

12

## Representing the world with 0's and 1's



- We can represent numbers in Binary
  - Natural Binary, two's complement, fixed point, floating point, BCD, etc…
- With numbers, we can represent:
  - TEXT
  - IMAGES
  - SOUND
  - Others
    - →Vibrations
    - →Smells…
    - →….whatever you want….

- To exchange information
  - NORMS (or FORMAT STANDARDS) are necessessay to interpret the 0s &1s
  - Well known formats: PDF, DOCX, JPG, XLS, MP3, etc,etc



## Binary codes- Alphanumeric codes



- Used to represent characters
  - ASCII code
    - →American Standard Code for Information Interchange
    - →Defines "normal","printable" characters, and control characters.
    - →Originally only 7 bits, but multiple 8 bit extensions with special characters
  - EBCDIC code (used only by IBM)
  - Unicode (16 bit extension to ASCII, with oriental characters)

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 16 | | 32 | | 48 | 0 | 64 | @ | 80 | P | 96 | ` | 112 | p |
| 1 | | 17 | | 33 | ! | 49 | 1 | 65 | A | 81 | Q | 97 | a | 113 | q |
| 2 | | 18 | DC2 | 34 | " | 50 | 2 | 66 | B | 82 | R | 98 | b | 114 | r |
| 3 | | 19 | DC3 | 35 | # | 51 | 3 | 67 | C | 83 | S | 99 | c | 115 | s |
| 4 | | 20 | DC4 | 36 | $ | 52 | 4 | 68 | D | 84 | T | 100 | d | 116 | t |
| 5 | | 21 | | 37 | % | 53 | 5 | 69 | E | 85 | U | 101 | e | 117 | u |
| 6 | | 22 | | 38 | & | 54 | 6 | 70 | F | 86 | V | 102 | f | 118 | v |
| 7 | BEL | 23 | | 39 | ' | 55 | 7 | 71 | G | 87 | W | 103 | g | 119 | w |
| 8 | BS | 24 | | 40 | ( | 56 | 8 | 72 | H | 88 | X | 104 | h | 120 | x |
| 9 | | 25 | | 41 | ) | 57 | 9 | 73 | I | 89 | Y | 105 | i | 121 | y |
| 10 | LF | 26 | | 42 | * | 58 | : | 74 | J | 90 | Z | 106 | j | 122 | z |
| 11 | | 27 | ESC | 43 | + | 59 | ; | 75 | K | 91 | [ | 107 | k | 123 | { |
| 12 | FF | 28 | | 44 | , | 60 | < | 76 | L | 92 | \ | 108 | l | 124 | \| |
| 13 | CR | 29 | | 45 | - | 61 | = | 77 | M | 93 | ] | 109 | m | 125 | } |
| 14 | SO | 30 | | 46 | . | 62 | > | 78 | N | 94 | ^ | 110 | n | 126 | ~ |
| 15 | SI | 31 | | 47 | / | 63 | ? | 79 | O | 95 | _ | 111 | o | 127 | |

Original ASCII

## Extended ASCII (8 bit code)

Sistemas Digitais

- **Code page 850** (international, ISO/IEC 8859-1, UTF-8)

| ASCII control characters | | | ASCII printable characters | | | | | | Extended ASCII characters | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | NULL | (Null character) | 32 | space | 64 | @ | 96 | ` | 128 | Ç | 160 | á | 192 | └ | 224 | Ó |
| 01 | SOH | (Start of Header) | 33 | ! | 65 | A | 97 | a | 129 | ü | 161 | í | 193 | ┴ | 225 | ß |
| 02 | STX | (Start of Text) | 34 | " | 66 | B | 98 | b | 130 | é | 162 | ó | 194 | ┬ | 226 | Ô |
| 03 | ETX | (End of Text) | 35 | # | 67 | C | 99 | c | 131 | â | 163 | ú | 195 | ├ | 227 | Ò |
| 04 | EOT | (End of Trans.) | 36 | $ | 68 | D | 100 | d | 132 | ä | 164 | ñ | 196 | ─ | 228 | õ |
| 05 | ENQ | (Enquiry) | 37 | % | 69 | E | 101 | e | 133 | à | 165 | Ñ | 197 | ┼ | 229 | Õ |
| 06 | ACK | (Acknowledgement) | 38 | & | 70 | F | 102 | f | 134 | å | 166 | ª | 198 | ã | 230 | µ |
| 07 | BEL | (Bell) | 39 | ' | 71 | G | 103 | g | 135 | ç | 167 | º | 199 | Ã | 231 | þ |
| 08 | BS | (Backspace) | 40 | ( | 72 | H | 104 | h | 136 | ê | 168 | ¿ | 200 | ╚ | 232 | Þ |
| 09 | HT | (Horizontal Tab) | 41 | ) | 73 | I | 105 | i | 137 | ë | 169 | ® | 201 | ╔ | 233 | Ú |
| 10 | LF | (Line feed) | 42 | * | 74 | J | 106 | j | 138 | è | 170 | ¬ | 202 | ╩ | 234 | Û |
| 11 | VT | (Vertical Tab) | 43 | + | 75 | K | 107 | k | 139 | ï | 171 | ½ | 203 | ╦ | 235 | Ù |
| 12 | FF | (Form feed) | 44 | , | 76 | L | 108 | l | 140 | î | 172 | ¼ | 204 | ╠ | 236 | ý |
| 13 | CR | (Carriage return) | 45 | - | 77 | M | 109 | m | 141 | ì | 173 | ¡ | 205 | ═ | 237 | Ý |
| 14 | SO | (Shift Out) | 46 | . | 78 | N | 110 | n | 142 | Ä | 174 | « | 206 | ╬ | 238 | ¯ |
| 15 | SI | (Shift In) | 47 | / | 79 | O | 111 | o | 143 | Å | 175 | » | 207 | ¤ | 239 | ´ |
| 16 | DLE | (Data link escape) | 48 | 0 | 80 | P | 112 | p | 144 | É | 176 | ░ | 208 | ð | 240 | ≡ |
| 17 | DC1 | (Device control 1) | 49 | 1 | 81 | Q | 113 | q | 145 | æ | 177 | ▒ | 209 | Ð | 241 | ± |
| 18 | DC2 | (Device control 2) | 50 | 2 | 82 | R | 114 | r | 146 | Æ | 178 | ▓ | 210 | Ê | 242 | ‗ |
| 19 | DC3 | (Device control 3) | 51 | 3 | 83 | S | 115 | s | 147 | ô | 179 | │ | 211 | Ë | 243 | ¾ |
| 20 | DC4 | (Device control 4) | 52 | 4 | 84 | T | 116 | t | 148 | ö | 180 | ┤ | 212 | È | 244 | ¶ |
| 21 | NAK | (Negative acknowl.) | 53 | 5 | 85 | U | 117 | u | 149 | ò | 181 | Á | 213 | ı | 245 | § |
| 22 | SYN | (Synchronous idle) | 54 | 6 | 86 | V | 118 | v | 150 | û | 182 | Â | 214 | Í | 246 | ÷ |
| 23 | ETB | (End of trans. block) | 55 | 7 | 87 | W | 119 | w | 151 | ù | 183 | À | 215 | Î | 247 | ¸ |
| 24 | CAN | (Cancel) | 56 | 8 | 88 | X | 120 | x | 152 | ÿ | 184 | © | 216 | Ï | 248 | ° |
| 25 | EM | (End of medium) | 57 | 9 | 89 | Y | 121 | y | 153 | Ö | 185 | ╣ | 217 | ┘ | 249 | ¨ |
| 26 | SUB | (Substitute) | 58 | : | 90 | Z | 122 | z | 154 | Ü | 186 | ║ | 218 | ┌ | 250 | · |
| 27 | ESC | (Escape) | 59 | ; | 91 | [ | 123 | { | 155 | ø | 187 | ╗ | 219 | █ | 251 | ¹ |
| 28 | FS | (File separator) | 60 | < | 92 | \ | 124 | \| | 156 | £ | 188 | ╝ | 220 | ▄ | 252 | ³ |
| 29 | GS | (Group separator) | 61 | = | 93 | ] | 125 | } | 157 | Ø | 189 | ¢ | 221 | ¦ | 253 | ² |
| 30 | RS | (Record separator) | 62 | > | 94 | ^ | 126 | ~ | 158 | × | 190 | ¥ | 222 | Ì | 254 | ■ |
| 31 | US | (Unit separator) | 63 | ? | 95 | _ | | | 159 | ƒ | 191 | ┐ | 223 | ▀ | 255 | nbsp |
| 127 | DEL | (Delete) | | | | | | | | | | | | | | |

Portugal: Code page 860…

V.Lobo @ EN

15

15

## Unicode (16 bit code)

Sistemas Digitais

- **Unicode allows 8,16, or 32 bits**
  - Latin, Greek, Cyrillic, Chinese, Japanese, Thai, Sanskrit, etc
- **Complete Unicode has 16 "Planes"**
  - →Plane 0 – Standard UTF-16: Basic Multilingual Sets
  - →Plane 1 – Supplementary multilingual
  - →Plane 2 – Ideographic
  - →Others: more ideographic, private, special purpose, etc.

| Basic | | Supplementary | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Plane 0 | | Plane 1 | | Plane 2 | | Plane 3 | Planes 4–13 | Plane 14 | Planes 15–16 |
| **0000–FFFF** | | 10000–1FFFF | | 20000–2FFFF | | 30000–3FFFF | 40000–DFFFF | E0000–EFFFF | F0000–10FFFF |
| **Basic Multilingual Plane** | | **Supplementary Multilingual Plane** | | **Supplementary Ideographic Plane** | | **Tertiary Ideographic Plane** | ***unassigned*** | **Supplementary Special-purpose Plane** | **Supplement-ary Private Use Area planes** |
| BMP | | SMP | | SIP | | TIP | — | SSP | SPUA-A/B |
| 0000–0FFF | 8000–8FFF | 10000–10FFF | 18000–18FFF | 20000–20FFF | 28000–28FFF | 30000–30FFF | | E0000–E0FFF | *15: SPUA-A* F0000–FFFFF |
| 1000–1FFF | 9000–9FFF | 11000–11FFF | | 21000–21FFF | 29000–29FFF | 31000–31FFF | | | |
| 2000–2FFF | A000–AFFF | 12000–12FFF | 1A000–1AFFF | 22000–22FFF | 2A000–2AFFF | | | | *16: SPUA-B* 100000–10FFFF |
| 3000–3FFF | B000–BFFF | 13000–13FFF | 1B000–1BFFF | 23000–23FFF | 2B000–2BFFF | | | | |
| 4000–4FFF | C000–CFFF | 14000–14FFF | 1C000–1CFFF | 24000–24FFF | 2C000–2CFFF | | | | |
| 5000–5FFF | D000–DFFF | | 1D000–1DFFF | 25000–25FFF | 2D000–2DFFF | | | | |
| 6000–6FFF | E000–EFFF | 16000–16FFF | 1E000–1EFFF | 26000–26FFF | 2E000–2EFFF | | | | |
| 7000–7FFF | F000–FFFF | 17000–17FFF | 1F000–1FFFF | 27000–27FFF | 2F000–2FFFF | | | | |

V.Lobo @ EN

16

16

## Images



- **General idea**
  - Divide the image in "small squares" or "Picture Elements"
    - → PIXEL
  - Each PIXEL can occupy 1 bit (0,1 => black & white) or multiple bits, so encode colors, intensities, etc
- **Raster Formats**
  - BMP (Windows Bitmap)
    - → 24 bits (3 Bytes), representing 8 bit R,G,B, components for each pixel.
    - → No compression
  - TIFF (Tagged Image File Format)
    - → 24 or 32 bits
    - → Uses lossless compression (LZW)
  - JPEG (Joint Photographic Experts Group)
    - → Uses lossy compression. Compression can be adjusted, RFC 1341
  - (RAW) – No header, fancy format, or compression: just bitmaps
- **Other formats**
  - Vector formats (*e.g.* PCX, CAD, etc)
  - Others (GIF, PNG, CGM, SVG, JPG(2000), TGA, PDF, CDR, EPS, ODG, WMF, XPS, VML, XPS, DXF, PIC)



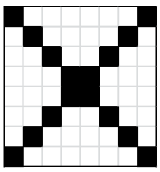


## Example of an image format



- **Consider the follwing image:**
- **It's (raw) coding, with 1 bit color depth, would be:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |



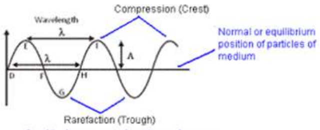
- **Represented another way:**
  - In bits:
    - →10000001010000100010010000011000000110000010010001000010 10000001
  - In 8 bit numbers represented in:
    - →Decimal: 129,66,36,24,24,36,66,129
    - →Hexadecimal: 81,42,24,18,18,24,42,81
  - With a header: (3 bytes for specifying color depth, 2 bytes for width, 2 bytes for hight: 00 00 01 00 08 00 08 81 42 24 18 18 24 43 81.

## Sound

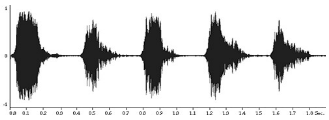
Sistemas Digitais

- **General idea**
  - Sound is composed of variations in pressure
  - At each point in time there is a (relative) value for pressure
    - → Number of different possible values for pressure determines the "quantification error"
  - Periodically one has to re-sample the pressure value
    - → Frequency with which pressure is samples determines the frequencies can observe (hear) in the sound. The highest frequency (Nyquist Frequency) is half the sampling frequency.

- **Sound formats**
  - WAV
    - → Keeps the pressure values
  - MP3
    - → Compresses the information
    - → Computes the "Fourier Transform", and keeps only the most important coeficients (lossy compression)
  - FLAC
    - → Lossless compression
    - → High quality audio
  - MIDI
    - → Which notes to play (≅ "vector sound")



Graphical representation of sound wave



V.Lobo © EN

19

19

## ERRORS

Sistemas Digitais

- **What is an error ?**
  - A 1 that by mistake changes to 0 or vice-versa

- **Transmission errors**
  - Wires and radio link have interferences

- **Degradation of magnetic, optical , or electric media**
  - Dust, scratches, "old age", charge loss, etc

- **Faulty circuits**

010100 ⟶ 010000

V.Lobo © EN

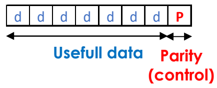
20

20

## ERRORS

Sistemas Digitais

- Solutions
  - Use redundant information to confirm data
    - →Send the same information multiple times !
    - →Send more information that strictly necessary
  - Parity bits
    - →A single bit will allow the detection of an odd number of errors
    - →Even, odd, mark, and space parity
    - →Byte by byte parity
    - →Vertical parity




  - Checksums
    - →e.g.: sum all bytes modulus 256 / compute XOR of all bytes
  - Correcting codes
    - →Hamming 5/3 code
    - →All correcting codes have a high overhead

V.Lobo @ EN

21

21

Sistemas Digitais

# Boolean Algebra

**Mathematical rules**
**for manipulating the 0's e 1's**
**that we use to represent the world**

V.Lobo @ EN

22

22

## Boolean Algebra



- **FORMAL definition**

$\{ U , + , . \}$

U = Finite set of symbols

+,· = Operations (named sum and product)

1 → $a + b \in U$
$a . b \in U$

2 → $a + b = b + a$
$a . b = b . a$

3 → $a + 0 = a$
$a . 1 = a$

4 → $a ( b + c ) = a b + a c$
$a + b c = ( a + b ) ( a + c )$

5 → $a + X = 1$
$a . X = 0$

$X \equiv \bar{a}$ ( complement )



## Utility of Boolean Algebra for Digital Systems



- **Let U = {0,1}**
  - The set U contains only 2 (binary) values
  - We can implement this system with electrical, electronic, optical, hydraulic devices, amongst others !
- **Addition Operation**
  - Also know as logical OR
- **Product Operation**
  - Also know as logical AND
- **Complement Operation**
  - Simple NEGATION

We use binary because it is easy to make machines that recognize 2 states

$U = \{ 0 , 1 \}$

+ = "OR" ( logical OR)

. = "AND" ( logical AND )

Complement = "NOT" ( logical not)

may be written as $\mathrm{not}(x)$, $\bar{x}$, $!x$, $\sim x$, $\neg x$

## THEOREMS

Sistemas Digitais

- Direct consequences of the axioms, they are the base for all operations we will do
- Duality Principle
  - If a given proposition is true, then changing ANDs by ORs (and vice-versa), and 0s by 1s (and vice-versa), the resulting proposition is also true
- 1 – Absorbing element (Null or Dominance law)
  - $A . 0 = 0 \qquad A + 1 = 1$
- 2 – Neural element (Identity law)
  - $A . 1 = A \qquad A + 0 = A$
- 3 – Idempotent law
  - $A . A = A \qquad A + A = A$

V.Lobo @ EN

25

25

## THEOREMS

Sistemas Digitais

- 4 – Inverse law
  - $A . \overline{A} = 0 \qquad A + \overline{A} = 1$
- 5 – Double complement law
  - $A = \overline{\overline{A}}$
- 6 – Comutative law
  - $A . B = B . A \qquad A + B = B + A$
- 7 – Associative law
  - $A . B . C = ( A . B ) C = A . ( B . C )$
  - $A + B + C = ( A + B ) + C = A + ( B + C )$
- 8 – DeMorgan's law

$$\overline{A . B} = \overline{A} + \overline{B} \qquad \overline{A + B} = \overline{A} . \overline{B}$$

V.Lobo @ EN

26

26

## THEOREMS

Sistemas Digitais

- **9 – Distributive law**
  - $A \cdot (B + C) = A \cdot B + A \cdot C$
  - $A + B\,C = (A + B) \cdot (A + C)$
- **10 – Absorption law**
  - $A + A\,B = A$ $\qquad A\,(A + B) = A$
- **11 -**
  - $A\,B + A\,\overline{B} = A$ $\qquad (A + B) \cdot (A + \overline{B}) = A$
- **12 -**
  - $A + \overline{A}B = A + B$ $\qquad A \cdot (\overline{A} + B) = A \cdot B$
- **13 – Inclued term law**
  - $A\,B + \overline{A}\,C + B\,C = A\,B + \overline{A}\,C$
  - $(A + B)(\overline{A} + C)(B + C) = (A + B)(\overline{A} + C)$

V.Lobo @ EN

27

27

## Demonstrations

Sistemas Digitais

- **Using truth tables**
  - Show for EVERY possible case.
  - Truth tables for AND and OR functions

| A | B | S = A.B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| A | B | S = A+B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

V.Lobo @ EN

28

28

## Example:

Sistemas Digitais

- Prove that $A \cdot (\bar{A} + B) = A \cdot B$

| A | B | $\bar{A}$ | $\bar{A}+B$ | $A.(\bar{A}+B)$ | $A.B$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |

V.Lobo @ EN

29

29

## Functions of 2 variables

Sistemas Digitais

- How many real functions are there, with 1 or 2 variables ?
- How many Boolean functions are there, with 2 variables ?
  - It is a finite number !
  - 2 variables ⇒ 4 input combinations ⇒ $2^4$=16 functions
  - 3 of them stem from the axiomatic definition of the algebra:
    - →AND ( E, . )
    - →OR ( OU, + )
    - →NOT ( NEG, ¯ )
  - Other functions are commonly used : XOR, NAND, NOR
- Physical implementation
  - Mechanical systems (leavers, cranks, toothed wheels, etc,)
  - Hydraulic systems (still used in explosive environments)
  - Electrical systems (relays)
  - Electronic systems (transistors, diodes, integrated circuits)
    - →By far the most commonly used ! (today)

V.Lobo @ EN

30

30

## Boolean functions

Sistemas Digitais

- How many Boolean functions exist ?
  - For a given number os input variables, it is finite !
    - →1 variable ⇒ 2 possible inputs ⇒ $2^2$=4 functions (which ?)
    - →2 variables ⇒ 4 possible inputs ⇒ $2^4$=16 functions
  - 3 functions are required by the axiomatic definition:
    - →AND ( . )
    - →OR ( + )
    - →NOT ( ¯ )
  - Other commonly used 2 variable functions: XOR, NAND, NOR
- Physical implementation
  - Mechanical systems (gears, levers)
  - Hydraulic systems (specially in dangerous, explosive environments)
  - Electrical systems (with relays)
  - Electronic systems (transistors, diodes, integrated circuits)
    - →By far the most (and only) used in practice: efficient and cheap

V.Lobo @ EN 31

31

Sistemas Digitais

# Physical implementation of Boolean Algebra Functions

**Electric circuits that can perform the Boolean operations (AND, OR, NOT), thus manipulate binary information, thus can be used to build computing machines**

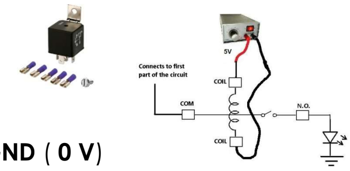
V.Lobo @ EN 32

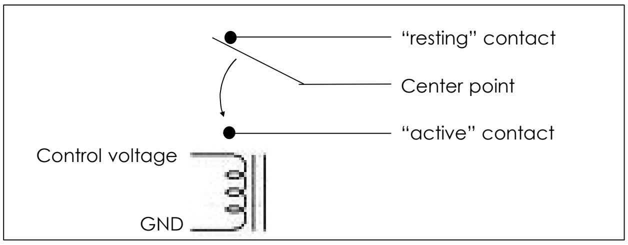
32

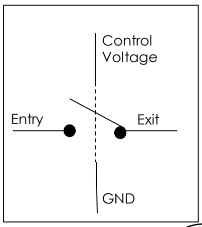
## Examoples of physical implementations

Sistemas Digitais

- **Objective:**
  - Build a **device** (a machine) that can perform the necessary operations, or **logical functions** of Boolean algebra
- Exemple
  - We can use electric **relays**
  - They are electrically controlled **Switches**
  - The logical "0" can be represented by **GND** ( **0 V**)
  - The logical "1" can be represented by $\mathbf{V_{cc}}$ (for example **5 V**)




V.Lobo @ EN

33

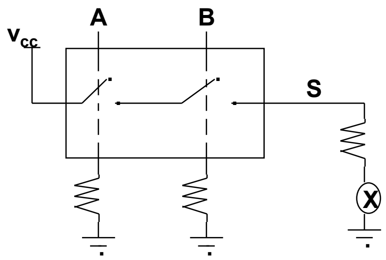
33

## Physical implementation with switches

Sistemas Digitais

- **"AND" GATE with relays**

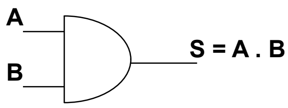
**Physical implementation**




**Truth table**

| A | B | A.B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**Logic symbol**




V.Lobo @ EN

34

34

## Physical implementation with switches

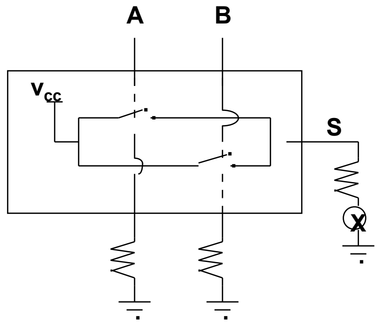
Sistemas Digitais

- “OR” GATE with relays

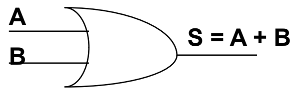
Truth table

| A | B | A+B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Physical implementation




Logic symbol






## Physical implementation with switches

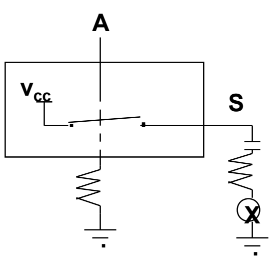
Sistemas Digitais

- “NOT” GATE with relays

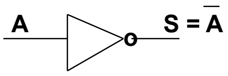
Truth table

| A | $\overline{A}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

Physical implementation




Logic symbol

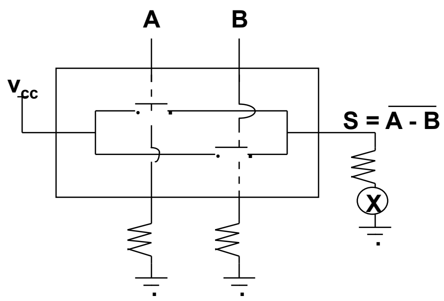


## Physical implementation with switches



- **"NAND" GATE with relays**

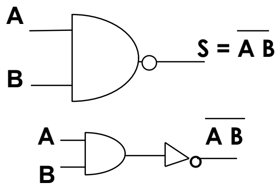
**Physical implementation**




**Truth table**

| A | B | $\overline{A.B}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Logic symbol**



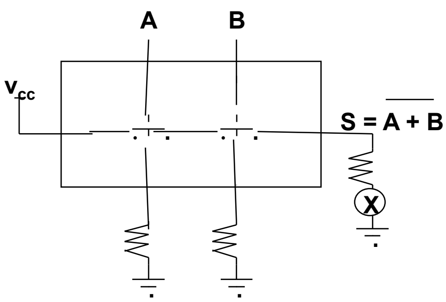



## Physical implementation with switches



- **"NOR" GATE with relays**

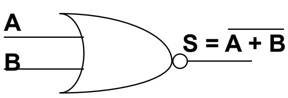
**Physical implementation**




**Truth table**

| A | B | $\overline{A+B}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

**Logic symbol**



## Physical implementation with switches

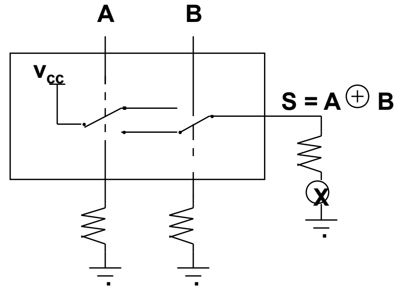
Sistemas Digitais

- "XOR" GATE with relays

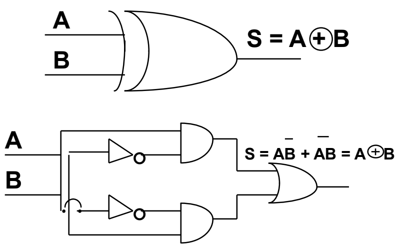
Truth table

| A | B | A⊕B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Physical implementation




Logic symbol




V.Lobo @ EN

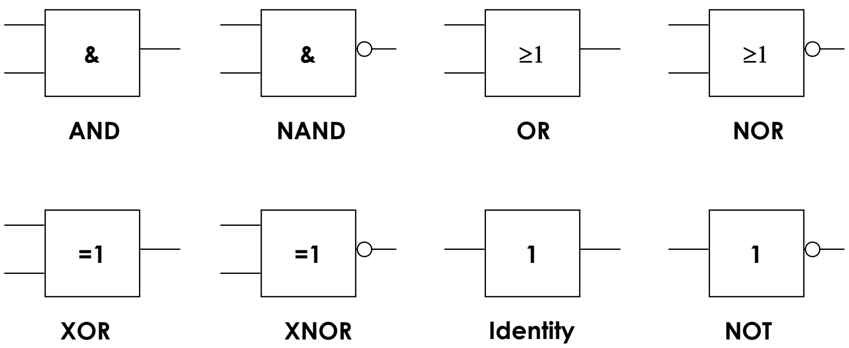
39

39

## Other symbols

Sistemas Digitais

- ANSI Y.32.14 standard
  - Simplifies the graphical representation of gates
  - Less pretty, but more efficient




V.Lobo @ EN

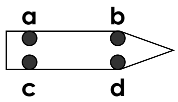
40

40

## Problems

Sistemas Digitais

- Fire alarm problem
  - Imagine you have two digital fire sensors (1=fire, 0=no fire), and a LED that should be turned on when there is a fire in any of the sensors. Project a logical circuit to actuate the LED based on the inputs.
- Ship's sentinels problem
  - Imagine that in a given ship there are 4 guard points where sentinels should be stationed: two on each side, one on the bow another on the stern. In each of those stations there is a sensor that sends a "1" when someone is there, and a "0" otherwise. In the officers wardroom we need 2 LEDS: a red one that turns on whenever there is no one on one of the sides, and a yellow one when there are only 2 sentinels on the watch. Project a logical circuit to actuate the LEDs based on the inputs.




V.Lobo @ EN

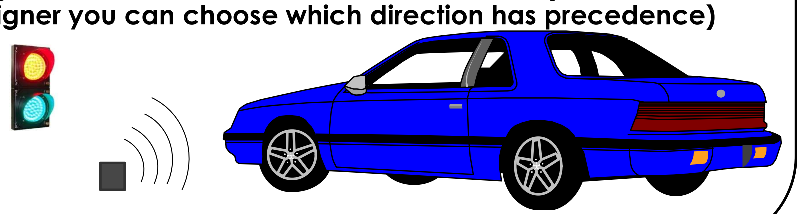
41

41

## Problems

Sistemas Digitais

- Traffic lights "on demand"
  - Imagine that at a given location, there are is a chokepoint a two-way road, where only one car pass at each time. Two traffic lights are available to install there, each with only two lamps: a green one, and a red one. There area also two sensors available, that send a digital signal "1" when a car is in front of it, and "0" when there is nothing there. If there is a car on one side, and none in the other, the traffic light should be green for that car (and red in the other direction). If there are no cars, then the lights should all be red. If there are cars in both directions, then in should be green in one direction and red in the other (and as designer you can choose which direction has precedence)



V.Lobo @ EN

42

42

## Sufficiency of the NAND function

Sistemas Digitais

- How many gates are necessary to generate any Boolean function ?
  - 3 functions are necessary to implemente the basic axioms, so those are sufficient for all the algebra:
    - →AND, OR, NOT
  - If one can implement these three functions using another one, that function can generate any Boolean function you may want

- Sufficiency of NAND
  - NOT(A) = A NAND A
  - A AND B = (A NAND B) NAND (A NAND B)
  - A OR B = (A NAND A) NAND (B NAND B)

V.Lobo @ EN

43

43

## Physical implementations

Sistemas Digitais

- Logical Families
  - Allow direct conections between the ouput of a gate and the input of another gate
  - Possible exemples: switches, realys, mechanical system, etc..

- Electronic logical families
  - DTL ,RTL – Easy to undersand, implemente, and explain
  - ECL - Very fast, power hungry, used only is ultra-fast processing
  - TTL - Most common in desktop devices
    - →Cheap, easy to use, good compromise of characteristics. Most common families are the 74xx, and 54xx for ruggedized devices.
  - CMOS - Most common in portable devices
    - →Very low consumption, tolerant to various levels of supply voltage, allow very dense integrated circuits (4000 family, and others)

V.Lobo @ EN

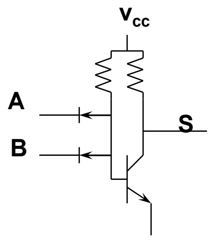
44

44

## DTL

Sistemas Digitais

- **Diode-Transistor Logic**
  - Uses diodes and transistors
  - Example: NAND gate
  - Hypothesis 1 : A=0V or B=0V (logical "0")
    - →Diodes conduct (ON state)
    - →Voltage at the base of the transistor is aprox. = 0V
    - →The transistor switches OFF
    - →The output resistor acts as "pull-up", and the output voltage S is aprox = 5V
  - Hypothesis 1: A = B = 5V (logical "1")
    - →Diodes do NOT conduct (OFF state)
    - →Voltage at the base of the transistor is aprox. = 5V
    - →The output voltage S is aprox. = 0V




V.Lobo © EN

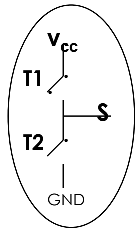
45

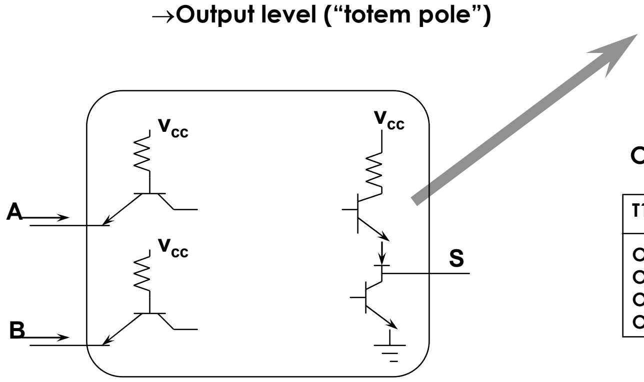
45

## TTL

Sistemas Digitais

- **Transistor-Transistor Logic**
  - Bipolar Junction Transistors
  - Let us look only at:
    - →Input level (inverted díods)
    - →Output level ("totem pole")




**Output level**

| T1 | T2 | Saida |
|---|---|---|
| ON | OFF | $V_{CC}$ |
| OFF | ON | GND |
| OFF | OFF | Tri-State |
| ON | ON | Bumm! |

V.Lobo © EN

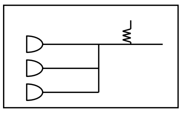
46

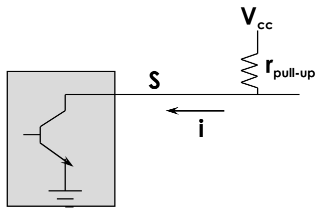
46

## TTL

Sistemas Digitais

- **Open-Collector Gates**
  - Output level only has a transístor (connected to GND)
  - The gate can only force the logic value "0"
  - There mus be na external PULL-UP resistor, to force the logical "1"
  - We may use it implemente a "WIRED-AND", connecting various O.C. ouputs






**TRANSISTOR ON**
Ouput voltage is 0, and the current is $V_{cc}/R_{pull-up}$

**TRANSISTOR OFF**
The current is 0, thus the voltage is $V_{cc}$

V.Lobo @ EN

47

47

## Characteristics of LOGICAL FAMILIES

Sistemas Digitais

- **TTL**
  - Easy to build and use. Widely available,
  - Robust and reliable
  - Low cost
  - Reasonable energy consumption
  - 74xxyy and 54xxyy families
    - →54xx has military specifications: large range of operating temperatures/humidity/vibration conditios, pins optimized for Electro-Magnetic Compatibility
    - →Variations 74S , 74LS , 74L , 74H, 74HCT (power, speed)
- **CMOS, NMOS and PMOS**
  - Field effect transistors
  - Very low consumption
  - Slow and sensitive to static electiricity
  - Flexible with power levels
  - 40xx family

V.Lobo @ EN

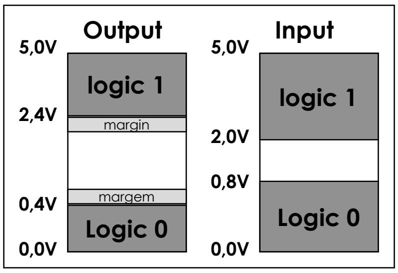
48

48

## Characteristics of LOGICAL FAMILIES (TTL)

Sistemas Digitais

- **FAN-OUT**
  - Number of gates that can be connected at the ouput
  - Specified in number of gates (assuming a standard consumption for each famility(, or in current ( mA).
- **FAN-IN**
  - Currenct injected / consumed in the inputs
- **Noise margin**
  - Tolerance between levels
  - Logical 0 is not quite 0V
  - Logical 1 in not quite 5V




**Nota:**
What is noise ?
What are it's effects ?
What are the soures of noise ?
How can it be minimized ?

V.Lobo @ EN

49

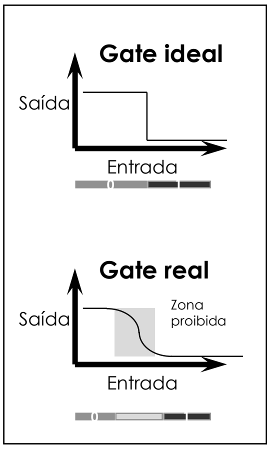
49

## Characteristics of LOGICAL FAMILIES

Sistemas Digitais

- **Transfer function**
  - Transitions 0 → 1 → 0 are not perfect
  - Example: NOT gate
- **Propagation time**
  - Gates takes a certain time until the ouptput reflects the current inputs
  - Propagation delays when transitioning from 0 to 1 is normally different the propagation delays transitioning from 1 to 0
- **Dissipation**
  - Gates need current and heat up in the process
  - The heat produced in normally proportional to the clock frequency

GATE NOT




V.Lobo @ EN

50

50

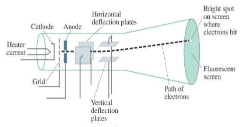
## DISPLAYS – Available Technologies

Sistemas Digitais

- Gas discharge Indicadors
  - →Electron valves
- Leds – Light Emitting Diodes
  - →Low consumption
  - →Easy interface
  - →Great Variety

- Liquid Crystal Displays (LCD)
  - →Very low consumption
  - →Changes in polarization (caused by changes in the electrical field) stop light from being reflected

- Cathode Ray Tubes
  - →Electrons projected against a phosphorous screen




V.Lobo @ EN

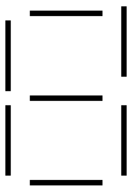
51

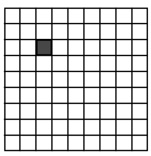
51

## DISPLAYS – Physical Layout

Sistemas Digitais

- Discreet elements
  - 7 Segment Diplays
  - More segments
- Matrix displays
  - Variable resolution





V.Lobo @ EN

52

52

## Boolean Functions

Sistemas Digitais

- S = F (A)    S = F (A,B,C, ...)
  - Where A,B,C... can assume values 0 or 1
- For a given number of functions, the number of functions is FINITE (limited)
  - They can be numbered
  - Example: Functions with 1 variable:

| FUNCTION | INPUTS 0 | INPUTS 1 | NAME | BOOLEAN expression |
|---|---|---|---|---|
| S0 | 0 | 0 | Zero | 0 |
| S1 | 0 | 1 | Identity | A |
| S2 | 1 | 0 | Negation | !A (or $\overline{A}$) |
| S3 | 1 | 1 | Unity | 1 |

V.Lobo @ EN

53

53

## Functions with two variables

Sistemas Digitais

| FUNCTION | INPUTS 00 ,01,10,11 | NAME | BOOLEAN Expression | NOTATION | DUAL | COMPLEMENT |
|---|---|---|---|---|---|---|
| S0 | 0 0 0 0 | Zero | 0 | | 15 | 15 |
| S1 | 0 0 0 1 | And | A.B | A.B | 7 | 14 |
| S2 | 0 0 1 0 | Inibition or Nix | A.B* | | 11 | 13 |
| S3 | 0 0 1 1 | Identity (to A) | A | | 3 | 12 |
| S4 | 0 1 0 0 | Inibition or Nix | A*.B | | 13 | 11 |
| S5 | 0 1 0 1 | Identity (to B) | B | | 5 | 10 |
| S6 | 0 1 1 0 | eXclusive OR | A*.B+A.B* | $A \oplus B$ | 9 | 9 |
| S7 | 0 1 1 1 | OR (Inclusive) | A+B | A+B | 1 | 8 |
| S8 | 1 0 0 0 | NOR (or Dagger) | (A+B)* | $\overline{A+B}$ | 14 | 7 |
| S9 | 1 0 0 1 | Equivalence | A.B+A*.B* | $A \equiv B$ | 6 | 6 |
| S10 | 1 0 1 0 | NOT (Negation) | B* | | 10* | 5 |
| S11 | 1 0 1 1 | Implication | A+B* | $B \Rightarrow A$ | 2 | 4 |
| S12 | 1 1 0 0 | Not (Negação) | A* | | 12 | 3 |
| S13 | 1 1 0 1 | Implication | A*+B | $\underline{A \Rightarrow} B$ | 4 | 2 |
| S14 | 1 1 1 0 | NAND (or Stroke) | (A.B)* | A . B | 8 | 1 |
| S15 | 1 1 1 1 | UNITY | 1 | | 0 | 0 |

V.Lobo @ EN

54

54

## Dual and complemente functions



- **DUAL function**
  - G is dual of F iff G(A)= (F(A*))*
  - (X* is the dual of X if you Exchange 1 for 0, + for . , and vice-versa)
  - Exemples
    - →The dual of AND is OR
    - →The dual of negation, is negation itself
- **COMPLEMENT function**
  - G is the complement of F iff G(A)=!F(A)
    - →The complement of AND is NAND
    - →The complement of NOT is Identity



## Canonical Forms



- **How can you identify a function in a normalized way?**
  - Analytical expressions can have many forms
  - Truth tables are very long
  - Canonical forms
    - →The truth table, for each input has either 0 or 1
    - →The lines of the truth table can be ordered so that the inputs form increasing binary numbers
    - →A function can be identified by the lines where the result is "1" (or where the result is 0)

$A.\overline{B}+\overline{A}.B$=A.!B+!A.B=A⊕B

| A | B | S |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Lines 1 e 2 have "1"

Inputs are ordered (0,1,2,3)

$F_{1,2}$

## CANONICAL FORMS

Sistemas Digitais

- **How can we identify lines in the truth table ?**
  - Each line is a PRODUCT of all free variables
- **MINTERMS**
  - Products that include ALL FREE VARIABLES
  - Correspond to lines in the truth table, numbered according to the binary code that the variables form
  - To determine the binary code, affirmed variables are represented with "1", and negated variables are represented with "0"
  - Example: $A\bar{B}CD$ is numbered 1011, *i.e.* eleven
- **MAXTERMS**
  - Sums that include ALL FREE VARIABLES
  - They are duals of the minterms (but are simpler if there are frew "0")
  - $M_i \Rightarrow m_{2^n-1-i}$

V.Lobo @ EN

57

57

## CANONICAL FORMS

Sistemas Digitais

- **1st CANONICAL FORM**
  - Sum of minterms
  - Example: XOR function
    - →XOR(A,B) = $A \cdot \bar{B} + \bar{A} \cdot B = m_1 + m_2 = \sum m_{(1,2)}$
  - Exercises:
    - →What is the truth table of the 3-variable function $\sum m_{(0,5,7)}$ ?
    - →What is the 1st canonical form of the 3-variable OR function ?
- **2nd CANONICAL FORM**
  - Product of maxterms
  - Example: XOR function
    - →XOR(A,B) = $(\bar{A} + \bar{B}) \cdot (A + B) = M_0 + M_3 = \prod M_{(0,3)}$
  - Exercises:
    - →What is the truth table of the 3-variable function $\Pi\, M_{(0,5,7)}$ ?
    - →What is the 2nd canonical form of the 3-variable OR function ?

V.Lobo @ EN

58

58

## Recipe for solving problems with combinatory circuits

Sistemas Digitais

1) **Obtain a Boolean function that solves the problem:**
   - Using analytical methods (describe the problem with boolean logic)
   - Specify the desired outputs in a truth table
     - →Obtain the minterms/maxterms and you have the boolean

2) **Simplify the Boolean function using**
   - Analytical manipulation of the expression
   - Karnaugh Maps (Venn diagram based method)
   - Quine-McCluscky or other mathematical optimization method

3) **Implement the circuit (on discreet logic or programable logic devices)**
   - Choose the integrated circuits that have the desired gates
     - →You may need to change the function obtained in 2 to minimize the number of integrated circuits used
   - Draw the circuit layout (or use CAE software to do so)

V.Lobo @ EN

59

59

Sistemas Digitais

**OPTIONAL**

**A few notes on Karnaugh simplification**

V.Lobo @ EN

60

60

## Example

Sistemas Digitais

- **Step 1 for the ship sentinels:**
  - Analitical Method: L= a.b.!c.!d + a.!b.!c.!d + !a.b.!c.d + ....
  - Truth table:

**Minterms:**
0,1,2,3,4,8,12

A(a,b,c,d) = $\Sigma$ (0,1,2,3,4,8,12)

A(a,b,c,d) = !a.!b.!c.!d + !a....

| a b c d | A | |
|---|---|---|
| 0 0 0 0 | 1 | $\bar{a}.\bar{b}.\bar{c}.\bar{d}$ |
| 0 0 0 1 | 1 | |
| 0 0 1 0 | 1 | |
| 0 0 1 1 | 1 | |
| 0 1 0 0 | 1 | |
| 0 1 0 1 | 0 | |
| 0 1 1 0 | 0 | |
| 0 1 1 1 | 0 | |
| 1 0 0 0 | 1 | |
| 1 0 0 1 | 0 | |
| 1 0 1 0 | 0 | |
| 1 0 1 1 | 0 | |
| 1 1 0 0 | 1 | $a.b.\bar{c}.\bar{d}$ |
| 1 1 0 1 | 0 | |
| 1 1 1 0 | 0 | |
| 1 1 1 1 | 0 | |

V.Lobo @ EN

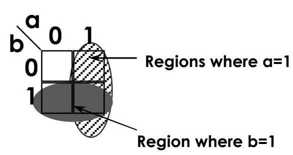
61

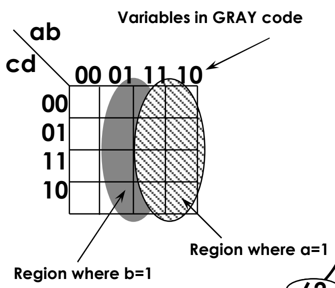
61

## KARNAUGH MAPS

Sistemas Digitais

- **A Karnaugh map is a way of writing the truth table**
  - Put the truth table in a 2-dimensional "map"
- **Each square differs from its neighbours in just 1 bit**
  - Hamming distance to all neighbours = 1


V.Lobo @ EN

62

62

## KARNAUGH MAPS

Sistemas Digitais

- **Método gráfico, baseado nos diagramas de Venn, que permite detectar adjacências**
  - 1) Escrever o mapa usando código reflectido, de modo a que 2 quadrículas contíguas diferem em apenas 1 bit.
  - 2 ) Cada quadrado corresponde a uma linha da tabela de verdade => corresponde a um mintermo da expressão se for 1
  - 3 ) Como na tabela 2 quadrados contíguos diferem apenas numa das variáveis, podemos escrevê-los como $\prod x_i \cdot y$ e $\prod x_i \cdot \bar{y}$
  - 4 ) Se dois quadrados contíguos forem 1, podemos representá-los como $\prod x_i \cdot y + \prod x_i \cdot \bar{y} = \prod x_i \cdot (y + \bar{y}) = \prod x_i$ , de onde se conclui que podemos ignorar a variável que troca de valor

- **REGRA:**
  - 1 ) Formar quadrados ou rectângulos com $2^m$ quadrículas
  - 2 ) Pôr na expressão só as variáveis que se mantêm constantes

V.Lobo @ EN

63

63

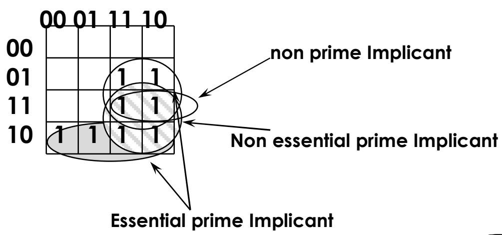
## KARNAUGH MAPS

Sistemas Digitais

- **Groups of minterms are combined into MINTERMS**
  - An IMPLICANT has a short expression for minterms that differ in 1 bit
  - PRIME Implicant
    - →An Implicant that cannot be further enlarged
  - ESSENCIAL Implicant
    - →An Implicant this the ONLY PRIME to cover a given minterm

**Problems:**

1. Sentinels
2. Seven segment BCD decoder
3. Traffic lights
4. Safety for citadel doors




V.Lobo @ EN

64

64

## KARNAUGH MAPS

Sistemas Digitais

- **"Don't Care"**
  - Correspond to cases where we don't care what happens to the input: it can be 1 or 0,
    - →Usually cases of inputs that cannot occur for some reason
  - Represented in the Karnaugh map with "X"
  - Depending on what is more convenient for simplification, we can assume they are either 1 or 0
  - Example: 7 segment BCD decoder (ex: middle segment)

| | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | x | 1 |
| 01 | 0 | 1 | x | 1 |
| 11 | 1 | 0 | x | x |
| 10 | 1 | 1 | x | x |

Some X are simplified to 1, others to 0

V.Lobo @ EN

65

65

Sistemas Digitais

# Basic Combinational Circuits

**Encoders and Decoders**

**Multiplexers and Demultiplexers**

**Comparators**

**Addition and Subtraction Circuits**

**Arithmetic and Logic Units (ALU)**

V.Lobo @ EN

66

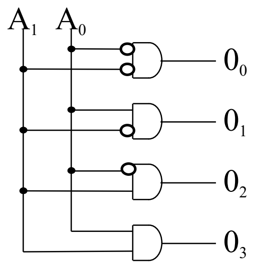
66

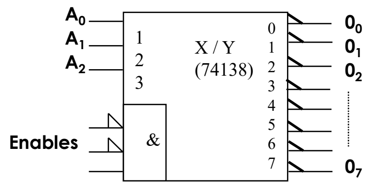
## Decoders

Sistemas Digitais

- **Objective: Decode a coded value**
- **Example: Given a two bit number, actuate on 1 and only 1 output (the one coded in the input=**
- **Implementation:**
  - Using elementary gates
  - With dedicated circuits

| Inputs | | Outputs | | | |
|---|---|---|---|---|---|
| $A_1$ | $A_0$ | $0_0$ | $0_1$ | $0_2$ | $0_3$ |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |




V.Lobo @ EN

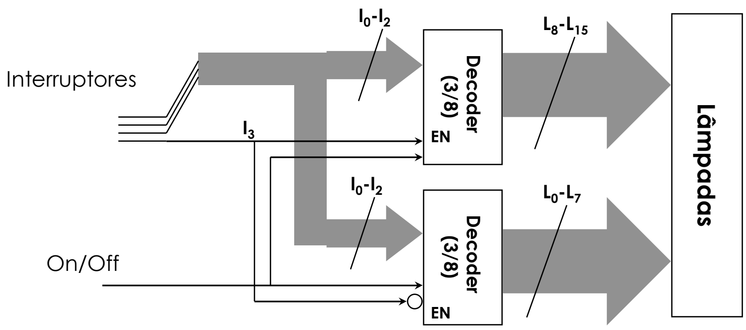
67

67

## Decoders

Sistemas Digitais

- **ENABLE inputs**
  - When they are off, the outputs are "off" (usually "0") or in TRI-STATE
  - We may use them to associate decoders
- **Example**
  - We want to turn on one of 16 lamps, using 4 control lines




V.Lobo @ EN

68

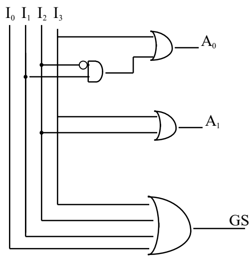
68

## Encoders

Sistemas Digitais

- **Objective: Preform the inverse of a decoder**
  - From a set of inputs (of which only one is active) generate a code to identify which one is active
  - Example: from the 4 wires coming from 4 different sensors, generate the number of the one that is active

- **Implementation:**
  - With logical gates
  - With dedicated integrated circuits




| Inputs | | | | Outputs | | |
|---|---|---|---|---|---|---|
| I3 | I2 | I1 | I0 | A1 | A0 | GS |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | x | 0 | 1 | 1 |
| 0 | 1 | x | x | 1 | 0 | 1 |
| 1 | x | x | x | 1 | 1 | 1 |

V.Lobo @ EN

69

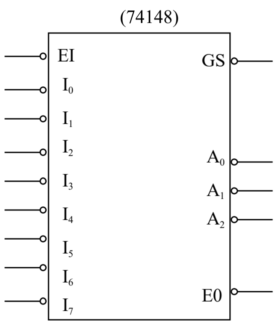
69

## Encoders

Sistemas Digitais

- **Group Signal (GS)**
  - Indicates that there is at least one active input
  - Important to separate between the situation when input number 0 is active (code=0,GS=1), and when no input is active (code=0, GS=0)

- **Enable**
  - When it is turned off, all output ate set to inactive




Nota: No 74148
todas as entradas
e saídas são
activas a 0

V.Lobo @ EN

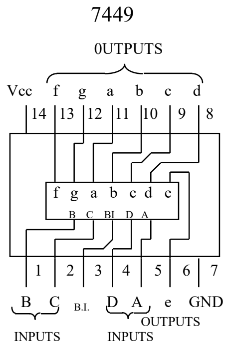
70

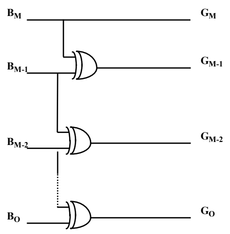
70

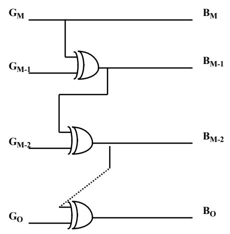
## Code converters and decoders

Sistemas Digitais

- **7 segment decoders**
  - Convert a 4 bit code (in natural binary or BCD) into the signals necessary to activate a 7 segment display)
- **BCD decoders**
  - Exemples: 74184,74185
- **Gray code enconders/decoders**
  - Implemented with a XOR ladder




V.Lobo @ EN

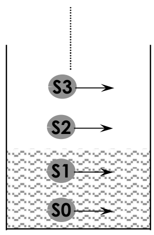
71

71

## Applications

Sistemas Digitais

- **Fuel level indicator problem**
  - We whish to know the fuel level in a fuel tank that is not easily accessible
  - We have16 sensors, that indicate if there is fuel at that level or not.
    Note: For an easier version use only 10 sensors
  - We have a 7 segment display to see the level in the tanks.
  - We have encoders and decoders (available with 16 to 4 and 4 to 16 configurations, BCD to 7Segments, etc).
  - We want the level indicator in a dashboard far from the tank
    →Obtain the full circuit diagram, including sensors, coding system, transmission lines, decoders, and displays
- **Solve the same problem, but using only the encoders/decoders used in the previous class**
  (8 to 3 encoders and 3 to 8 decoders)




V.Lobo @ EN

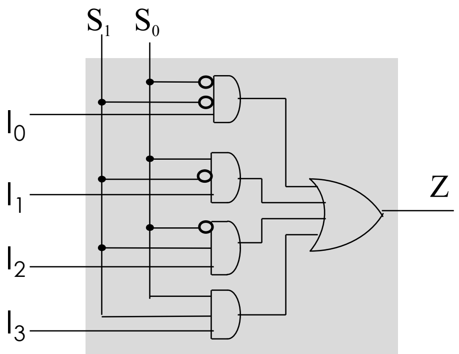
72

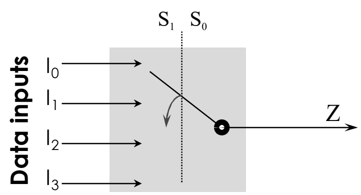
72

## Multiplexers (Mux)

Sistemas Digitais

- **Objective: Select one from a set of inputs**
  - Also called DATA-SELECTORS or simply "MUX"
- **Implementation:**
  - With logical gates
  - With dedicated integrated circuits




| Inputs $S_1$ | $S_0$ | Output $Z$ |
|---|---|---|
| 0 | 0 | $I_0$ |
| 0 | 1 | $I_1$ |
| 1 | 0 | $I_2$ |
| 1 | 1 | $I_3$ |

73

V.Lobo @ EN

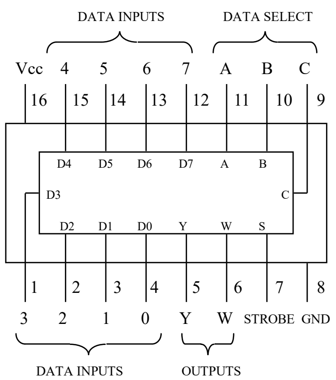
73

## Multiplexers (Mux)

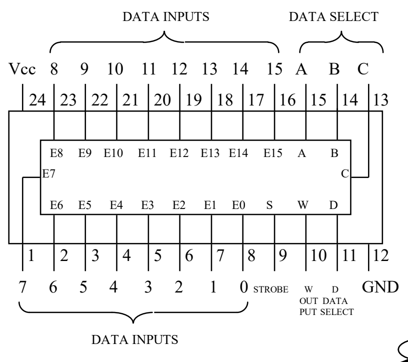
Sistemas Digitais

- **Multiplexers (in the 74xx TTL logical family)**
  - They usually have an ENABLE input, also called STROBE
    - → The STROBE signal is normally generated by the circuit that requires the information
  - 74151 - 8 bits for data inputs, 3 bits for selection and 1 bit for strobe
  - 74150 - 16 data inputs entradas, 4 selection inputs, and strobe

74151




74150




74

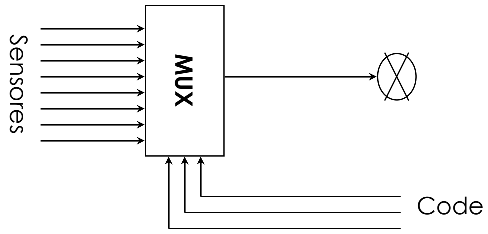
V.Lobo @ EN

74

## MUX - Applications

Sistemas Digitais

- **Multiplexing for pooling**
  - We may “interrogate” a set of sensors, sending the code of the one we want to see.




- **Multiplexing “buses”**
  - We may share a communication line, by alternating the data we pass through them with a multiplexer
  - We may use a clock and a counter so cycle through the various possible inputus

V.Lobo @ EN

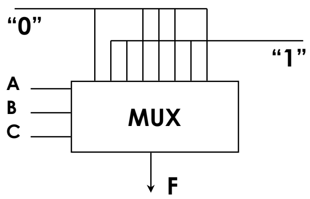
75

75

## MUX - Applications

Sistemas Digitais

- **Generate any Boolean function with a MUX**
  - A MUX is an OR of ANDs, and each AND corresponds to a MINTERM of the selection inputs, with one (and only one) of the data inputs.
  - To select a given MINTERM, we only need to set the corresponding data input to “1”
  - Advantages: with only 1 integrated circuit (an little project development) we can generate any function. If the inputs are variable, we can change the function in “run-time”
  - Disadvantages: we use more gates than necessary
  - Exemple:
    - →Implement the following function: F(A,B,C)=Σ(1,2,6)




V.Lobo @ EN

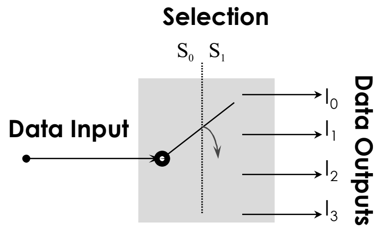
76

76

## Demultiplexing



- **It is the inverse operation of multiplexing**
  - Sent an input to one (and only one) of the outputs.
  - The outputs that are not connected to input are "inactive" (normally "0")
  - We can implement a DEMUX using a DECODER if we connect the selection inputs to the decoder's data inputs, and the input to the decoder's enable.




- **Aplications**
  - Project a system to transmit data from 16b sensors over just 6 wires



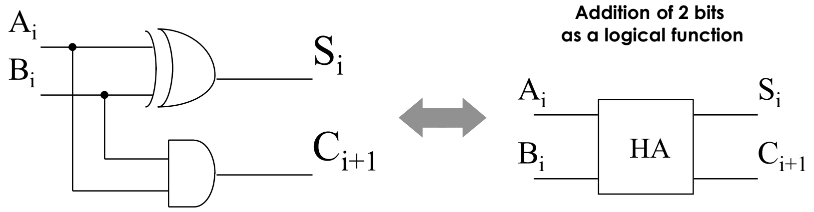
## Addition Circuits



- **Combinatory circuits for additions/subtractions**
  - Additions and subtractions can also be made using sequential circuits
- **Half adder**
  - Adds two bits (without a carry input)
  - Can only add numbers with 1 bit
  - May generate a "carry" bit

| A B | Sum | Carry |
|---|---|---|
| 0 0 | 0 | 0 |
| 0 1 | 1 | 0 |
| 1 0 | 1 | 0 |
| 1 1 | 0 | 1 |

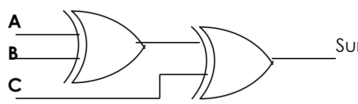
Addition of 2 bits as a logical function

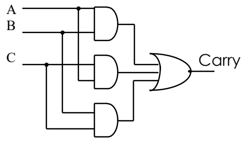


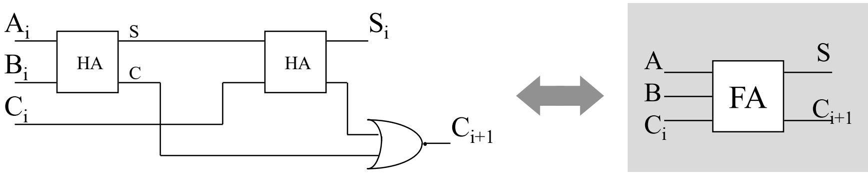
## Addition Circuits

Sistemas Digitais

- **Full adder**
  - Adds with bits with carry ⇒ adds three bits
  - Can be done directly as a logical function with 3 input variables
  - Can be done with two half-adders (sum 2 bits, and the sum carry)
  - Which is best ?




V.Lobo @ EN

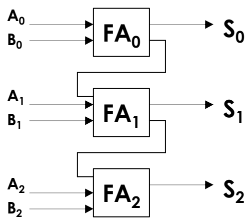
79

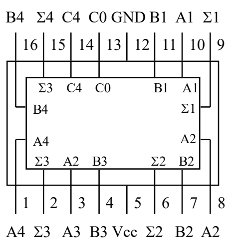
79

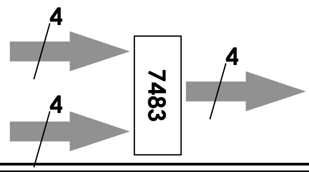
## Adding multiple bit numbers

Sistemas Digitais

- **Cascades of full adders**
  - Delay problems due to the carries
- **2 level adders**
  - Implement *n* bit additions as *n* functions with *2n+1* variables
  - Require a lot of hardware (gates)
- **Mixed solution**
  - Use genarate (G) e carry propagate (P) (look ahead) signals to fast-forward the carry
- **TTL 7483 integrated circuit**




V.Lobo @ EN

80

80

## Subtraction Circuits

Sistemas Digitais

- **Dedicated subtractors**
  - Implement the subtraction directly as a Boolean Funcion
- **Using adder circuits**
  - Subtracting is summing the negative of a number (*i.e.* the 2's complement)
  - The 2's complement may be obtained by complementing all bits and them summing 1
- **BCD Adders**
  - Correct the sum to be valid in BCD:
  - Generate a "carry" whenever the result exceeds 9, and adjust the value (summing 6)
  - Exercise:
    - →Design a BCD adder using 2 common adders, a multiplexer (MUX) and some control logic. It must generate the correct carry signal, and add 6 whenever the result is between 9 and 15.

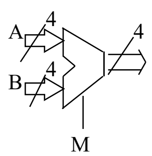
81

81

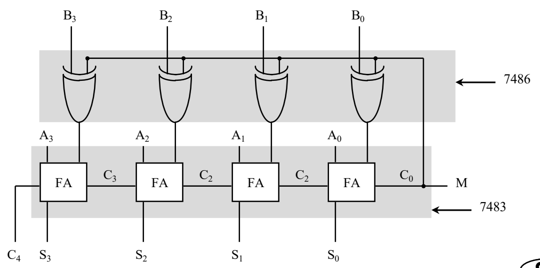
## ALU (Arithmetic and Logic Units)

Sistemas Digitais

- **Objective**
  - Perform various different arithmetic or logic operations using a single circuit, with different control signals
- **Structure**
  - They normally have 2 inputs each with *n* bits, know as operands.
  - They have one (or more) control input to specify which operation is done
- **Example:**
  - →A ALU to perform sums and subtractions of 4 bit numbers




M=0 Sum
M=1 Subtact



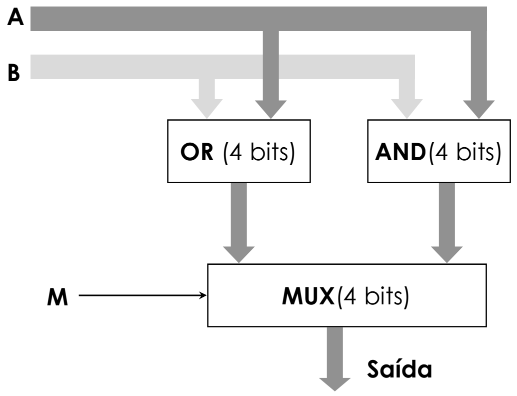

82

82

## ALU (Arithmetic and Logic Units)

Sistemas Digitais

- **Problem:**
  - **Project an ALU that can compute F(A,B) = A AND B, and F(A,B) = A OR B, depending whether M=0 or 1, with A and B being 4 bit words**
- **Solution:**




Another problem:

Project an ALU that can perform sums, subtractions, ANDs, or ORs

V.Lobo @ EN

83

83