

Lezione D01 - Introduzione all'elettronica digitale

- 1) Analogico e digitale. Logica a due valori.
- 2) Funzioni logiche, tavole di verità, Legg. di De Morgan
- 3) Rappresentazione dei numeri

1) Analogico e digitale

- I segnali analogici possono assumere qualsunque valore
→ effetti rumore, attenuazione; non adatti per i calcoli
- Segnali digitali: due valori soltanto "0, 1"; "F, T"
Mappe tra parametri fisici e valore logico
Ad esempio: intervalli di tensione → 0, 1
luce accesa / spenta → 1, 0
Le mappe e' subtrattiva (famiglie di circuiti logici): stessa funzione logica può essere implementata facilmente in modi diversi.
→ Separazione tra progetto logico e implementazione.
- Elemento essenziale per costituire i circuiti:
interruttore controllato

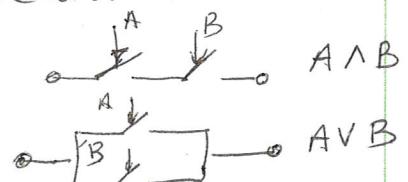


1906 - tubo a vuoto → 1946 - ENIAC

1948 - Transistor bipolare > circuiti integrati (1970)

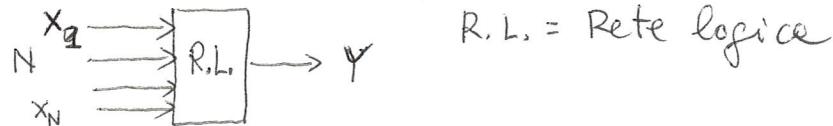
1959 - Transistor MOS > calcolatori moderni

Collegamenti logici booleani



2) Logica booleana

- Variabile booleana; può assumere i due valori
Non viene specificata l'implementazione. $B = \{0, 1\}$
- Funzione logica: una variabile booleana
che dipende da N ingressi (altre variabili)



$$Y = F(X_1 \dots X_N)$$

Si rappresenta con una "Tabelle di Verità"

	$X_1 \dots X_N$	Y	Colonna dei valori assunti delle funzione
tutti i valori possibili:	$\begin{cases} 0 \dots 0 \\ 0 \dots 1 \\ 1 \dots 0 \\ 1 \dots 1 \end{cases}$	$\begin{cases} 0 \\ 1 \\ 1 \\ 0 \end{cases}$	

- Logica combinatoria e sequenziale

Combinatoria: Y dipende esclusivamente dai
Valori attuali di $X_1 \dots X_N$
(funzione solo degli ingressi):

$$F: B^N \rightarrow B$$

Sequenziale: Y dipende dalla storia degli
ingressi (circuiti con memoria)

La rete logica ha uno "stato" interno
 S di cui dipende l'uscita

$$F: B^N \times S \rightarrow B$$

• Logica combinatoria

- Dati N ingressi, quante possibili funzioni esistono?
Tabelle di verità con 2^N righe

$X_1 \dots X_N$	\vee	\times	per ogni riga posso scegliere se 0 oppure 1
2^N			Funzioni distinte $2^{(2^N)}$

- 1 ingresso 4 funzioni = 2^2

A	Y_0	Y_1	Y_2	Y_3
0	0	1	0	1
1	0	0	1	1

\downarrow \downarrow \downarrow \downarrow
 $= 0$ $= \bar{A}$ $= A$ $= 1$

le funzioni costanti (0 e 1)
sono considerate "banalì"

Identità = A
NOT = \bar{A}

• Notazione logica booleana

$$\text{NOT } A = \bar{A} = \sim A = \neg A$$

A	\bar{A}
0	1
1	0

$$A \text{ AND } B = A \cdot B = A \& B = A \wedge B$$

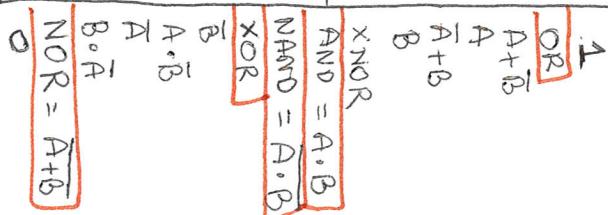
$$A \text{ OR } B = A + B = A | B = A \vee B$$

attenzione, non
è la somma
aritmetica

A	B	$A \cdot B$	$A + B$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

- Due ingressi $2^{(2^2)} = 2^4 = 16$ funzioni

A	B	Y ₀	Y ₁	Y ₂	Y ₃	Y ₄	Y ₅	Y ₆	Y ₇	Y ₈	Y ₉	Y ₁₀	Y ₁₁	Y ₁₂	Y ₁₃	Y ₁₄	Y ₁₅
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	1	1	1	0	0	0	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1



- Porte logiche = circuiti che realizzano una funzione logica \rightarrow Building blocks

1 ingresso A \rightarrow $Y = A$ identità

A \rightarrow $Y = \bar{A}$ NOT

Il cerchietto indica negazione

2 INGRESSI

A
B \rightarrow $A \cdot B$ AND

A
B \rightarrow $A + B$ OR

A
B \rightarrow $\overline{A \cdot B} = \overline{A} \cdot \overline{B}$ NAND

Building block
perché facile
da realizzare

A
B \rightarrow $\overline{A + B} = \overline{A} + \overline{B}$ NOR

XOR = OR ESCLUSIVO

A
B \rightarrow $A \oplus B$

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Con poche porte logiche elementari si possono realizzare tutte le funzioni logiche.

S' dimostra che basta i.e. NAND, oppure i.e. NOR, oppure AND + NOT ...

Metodologie:

- si parte dalle tabelle di verità

- si elettoro le funzioni in modo da esprimerele in termini di porte standard \rightarrow oramai

Foto
qui

Elaborazione logica: leggi di De Morgan e identità Booleane.

1) ASSOCIAZIONE

$$A + B + C = (A + B) + C = A + (B + C)$$

$$A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$$

2) COMMUTATIVITÀ

$$A + B = B + A ; A \cdot B = B \cdot A$$

3) DISTRIBUTIVITÀ

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

$$A + (B \cdot C) = (A + B) \cdot (A + C) \leftarrow \text{SORPRESA}$$

4) IDEMPOTENZA

$$A \cdot A = A ; A + A = A$$

5) ELEM. NEUTRO

$$A \cdot 1 = A ; A + 0 = A$$

6) ELEM. NULLO

$$A \cdot 0 = 0 ; A + 1 = 1$$

7) RIDONDANZA

$$A + (A \cdot B) = A ; A \cdot (A + B) = A$$

8) Invertibilità

$$\bar{\bar{A}} = A ; \bar{0} = 1$$

9) ELIMINAZIONE

$$A \cdot \bar{A} = 0 ; A + \bar{A} = 1$$

$$A + (\bar{A} \cdot B) = A + B ; A \cdot (\bar{A} + B) = A \cdot B$$

Leggi di De Morgan \rightarrow SIMMETRIA AND / OR

$$\overline{A \cdot B \cdot C \cdot D} = \bar{A} + \bar{B} + \bar{C} + \bar{D} \dots$$

$$\overline{A + B + C + D} = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D} \dots$$

permettono di pensare da OR AND

A	B	$\bar{A} \bar{B}$	$\bar{A} + \bar{B}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

- Esempio di applicazione al NAND di De Morgan

$$\bar{A} = \overline{A \cdot A} \quad A \rightarrow \text{NAND} \rightarrow \bar{A}$$

$$A \cdot B = \overline{\overline{A \cdot B}} = \overline{\overline{A} \oplus \overline{B}} \rightarrow \text{NAND} \rightarrow A \cdot B$$

$$A + B = \overline{\overline{A} \cdot \overline{B}} = \overline{A} \rightarrow \text{NAND} \rightarrow \overline{\overline{A} \cdot \overline{B}} = A + B$$

Vediamo che è possibile esprimere TUTTE le funzioni logiche in termini di

NOT, OR, AND \rightarrow NAND PORTA UNIVERSALE

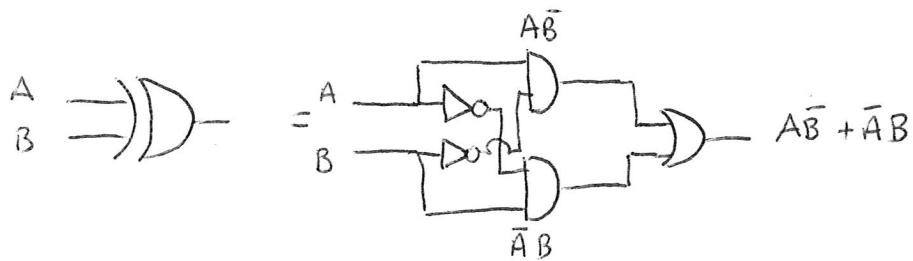
Anche NOR è porta universale

$$\bar{A} = \overline{A + A} \quad A \cdot B = \overline{\overline{A} \cdot \overline{B}} = \overline{\overline{A} + \overline{B}} \quad A + B = \overline{\overline{A} + \overline{B}}$$

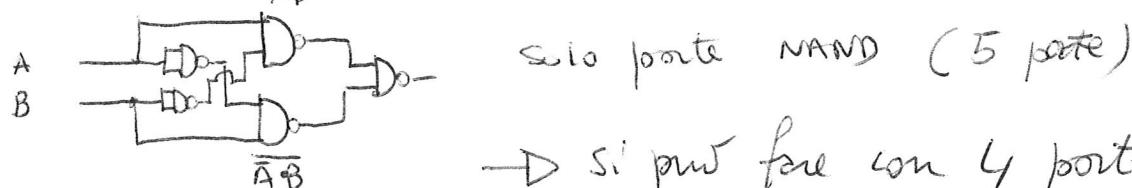
- Esempio XOR

"uno o l'altro, ma non entrambi" =

$$\begin{aligned} A \oplus B &= (A + B) \cdot (\overline{A} \cdot \overline{B}) = (A + B)(\overline{A} + \overline{B}) = \\ &= A\overline{A} + B\overline{A} + A\overline{B} + B\overline{B} = A\overline{B} + \overline{A}B \end{aligned}$$



$$\overline{A}\overline{B} + \overline{A}B = \overline{\overline{A}\overline{B} + \overline{A}B} = \overline{(\overline{A}\overline{B}) \cdot (\overline{A}B)}$$



3) Rappresentazione digitale dei numeri

→ operazioni realizzate con porte logiche su bit.

- Codici binari: rappresentazione con sequenze di 0, 1 (ciascun elemento è un bit)

$$\text{Numeri in base 2: } \begin{cases} b_0 \rightarrow 2^0 \\ b_1 \rightarrow 2^1 \\ \vdots \\ b_{N-1} \rightarrow 2^{N-1} \end{cases} \quad N \text{ bit}$$

$$x = \sum_{i=0}^{N-1} b_i 2^i$$

↑ bit i-esimo
= 0/1

Esempio: 10110 : 5 bit -

Most significant " bit msg LSB = Least significant bit

$$1 \times 2^4 + 1 \times 2^3 + 1 \times 2^1 = 16_d + 8_d + 2_d = 22_{dec}$$

N.B. i bit si contano da 0 a N-1

* Se ho N bit posso esprimere i numeri da 0 a $2^N - 1$

$$2^N - 1 = \sum_{i=0}^{N-1} (1) 2^i : \text{DIM. } 2^0 - 1 = 1 = \sum_{i=0}^0 2^i = 1 \rightarrow \text{SI}$$

Se vero per N → vero N+1: $2^{N+1} - 1 = 2(2^N) - 1 =$

$$= 2 \left(\sum_{i=0}^{N-1} 2^i + 1 \right) - 1 = \sum_{i=0}^{N-1} 2^{i+1} + 2^0 = \sum_{j=0}^N 2^j \quad \text{c.v.d.}$$

- Operazioni: Somma $0+0=0$ $0+1=1$ $1+1=10_b$

$$\begin{array}{r} 110 \\ 29 \\ \hline 139 \end{array} \quad \begin{array}{r} 11111 \\ 1101110 + \\ \hline 011101 \end{array}$$

riporto = CARRY

Moltiplicazione: $\times 2_d \rightarrow \times 10_b \rightarrow$ shift a sx

Divisione: $\div 2_d \rightarrow \div 10_b \rightarrow$ shift a dx

- Convenzione: si prendono i successivi resti delle divisioni intere per 2 (o per la base)

500/2

Esempio

$56_d / 2$	Q	R	
$28_d / 2$	14_d	0	$0 \leftarrow \text{LSB}$
$14_d / 2$	7	0	$32 \ 16 \ 8 \ 4 \ 2 \ 1$
$7_d / 2$	3	1	$111000_b = 56_d$
$3_d / 2$	1	1	
$1_d / 2$	0	1	$\leftarrow \text{MSB}$

Altre basi utili:

ottale (base 8) : $[0 - 7] = 2^3$

esadecimale (base 16) : $[0 - 9, A, B, C, D, E, F] = 2^4$

spesso indicata con $\underline{\underline{O_x}} 2A37$ per esempio

4 bit \rightarrow 1 cifra esadecimale \rightarrow Usatissimo.

3 bit \rightarrow 1 cifra ottale

$$\underline{\underline{O_x}} 3B = \underline{\underline{O_b}} 00111011$$

3 B = 11dec

hex $\underline{\underline{O_x}} F = 15_{\text{dec}}$ 4 bit

$O_x FF = 255_d$ 8 bit

$O_x FFFF = 65535_d$ 16 bit

x	$d=2^x$	
0	1	256
1	2	512
2	4	1024
3	8	2048
4	16	4096
5	32	8192
6	64	16384
7	128	32768
		65536

potenze di 2 de
surre e memorie.

a	b	hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

$\hookrightarrow K$

4) Rappresentazione dei numeri negativi

Concepto fondamentale: mappa (approssimata o limitata) di oggetti fisici o concettuali verso elementi computabili.

Fen. fisico $\xrightarrow{\text{num. reale}}$ numero computer
Approx Approx.

↳ 1) Precisione dello strumento

2) Interpretazione del modello

ad es. se misuro solo $|\psi|^2 \in \mathbb{R}$
non conosco $\psi \in \mathcal{C}$

Le mappature e' essenziale in tutti i comp'

DNA \rightarrow sequenze aminoacidi
Immagine \rightarrow sequenze di pixel
(Logica, Teorema) \rightarrow codone di Gödel

base delle dimostrazione che qualunque sistema logico e' incompleto, cioè contiene proposizioni indecidibili

\rightarrow D. Hofstadter, Gödel, Escher, Bach.

• Una volta copia le mappature sostituiamo continuamente oggetto per rappresentazione.
 \rightarrow mantenere la coscienza di quel che si fa.

Per noi: numero \leftrightarrow sequenze di bit.

Numeri negativi

- a) segno + valore
 - b) complemento a 1
 - c) complemento a 2
- $\left. \begin{array}{l} \\ \\ \end{array} \right\} MSB = 1 \text{ per numeri negativi}$

a) Segno + valore (8 bit) $N=8 \text{ bit}$

$$5_{dec} = 00000101 + 0 \div 2^{7-n-1} \text{ positivi}$$

$$-5_{dec} = \underline{10000101}, -1 \div -(2^{7-n}-1) \text{ negativi}$$

lo \emptyset ha due rappresentazioni 0_{dec} e -0_{dec} .
no regole addizione

b) Complemento a 1: nego classum bit

$$5_{dec} = 00000001 + 0 \text{ è somma}$$

$$-5_{dec} = \underline{11111010}$$

$$0_{dec} = \underline{11111111} \rightarrow \text{due rappresentazioni per lo } \emptyset$$

c) Complemento a 2

nego tutti i bit e sommo 1

$$x = \sum b_i 2^i$$

$$-x = \sum \overline{b_i} 2^i + 1$$

$$\begin{array}{r} 0 \times 0 \quad 5 \\ 5_{dec} = 00000|0101 + \\ -5_{dec} = 11111|0111 \\ \hline 0 \times F \quad DB \\ \hline \underbrace{(1)00000000}_0 \end{array}$$

carry
butte re nò N fissato!

0	$\overline{\downarrow}$	
1	-1	$\rightarrow 0x FF$
2	-2	$0x FE$
3	-3	F0
4	-4	FC
5	-5	FB
6	-6	FA
.	.	.
.	.	.

$$127 = 7F \quad -128 = 0x 80$$

$-2^{N-1} -1, 0, \dots (2^{N-1} - 1)$
negativi positivi
Antmetice funzione \rightarrow

OVERFLOW
Attenzione all'range!
 $A = 1010 \rightarrow 0101 +$

$$\begin{array}{r} 0x58 + \\ 0x32 \\ \hline 0x8A = -0x76 \end{array}$$

$$\overline{0110} = 6$$

Altre rappresentazioni

Codice GRAY:

- rappresentazione dei numeri in cui 1 solo bit cambia tra un numero ed il successivo
- Esempio $7 \rightarrow 8$ $0111 \rightarrow 1000$
4 bit cambiano \rightarrow assorbimento, rumore
- Non è univoco, ma il più usato è quello costituito per riflessioni

N ₆		N
0	0 0 0 0	0
1	0 0 0 1	1
3	0 0 1 1	2
2	0 0 1 0	3
6	0 1 1 0	4
7	0 1 1 1	5
5	0 1 0 1	6
4	1 1 0 0	7
C	1 1 0 0	8
D	1 1 0 1	9
F	1 1 1 1	A
E	1 1 1 0	B
A	1 0 1 0	C
B	1 0 1 1	D
9	1 0 0 1	E
8	1 0 0 0	F

Codice BCD = Binary coded decimal

\rightarrow rappresenta una cifra decimale in un "ibble", cioè 4 bit \rightarrow (spreco di bit) \rightarrow utile x display.

$$\text{Esempio } 137_{\text{dec}} = 0001|0011|0111|$$

$$0x89 = 00000|1000|1001|$$

CODIFICA DEI CARATTERI

Anche i caratteri vengono rappresentati con delle stringhe di bit.

Storicamente molte rappresentazioni diverse

IBM → EBCDIC (1963) 8-bit

ASCII → American standard code for information interchange (1968)

7-bit → 128 simboli

0x00 - 0x1F → caratteri di controllo

0x20 - 0x7E → stampabili
(lettere, numeri, punzecchiature)

0x7F - DELETE

Oggi: UNICODE - 21-bit

"plane" code

0x00 0x0000 - 0xFFFF

;

0x10

~~~~~

16 bit

} MILIONI DI CARATTERI

Per entrare l'enorme spazio di spazio se di meno i caratteri base → Ricodifica UTF-8, UTF-16, UTF-32

UTF-8 = Unicode Transformation Format

rappresenta i codici UNICODE in 1, 2, 3, o 4 bytes

1 byte 0xx~~x~~xxxx → codice ASCII (un file ASCII è UTF-8)

2 byte 110~~x~~xxxx 10~~y~~yy~~z~~zz → 0000xxxxyy<sup>UNICODE</sup>yyzzzz

3 byte 1110xxxx 10y~~yy~~yy 10zzzzzz

etc. → xxxx yy~~yy~~yy zzzzzz