Capitolo 2
Codifica binaria dell'informazione

2.1 - Rappresentazione
2.2 – Codifica di caratteri
2.3 – Codifica dei numeri
2.4 - Trasmissione
2.5 - Protezione

2.1
Rappresentazione

Alfabeti e simboli

La rappresentazione dell'informazione

<u>Informazione</u> - Stringa di lunghezza finita formata da simboli s_i appartenenti ad un alfabeto di definizione A:

$s_1\, s_2\, s_3 \dots\, s_i \dots\dots\, s_{n-1}\, s_n$ **con** $s_i \in \mathtt{A}\!:\!\{\mathtt{a}_1,\ \mathtt{a}_2,\ \dots,\ \mathtt{a}_m\}$

Esempi:

"testo" e caratteri

"numero" e cifre

"musica" e note

"immagine", pixel e toni di grigio

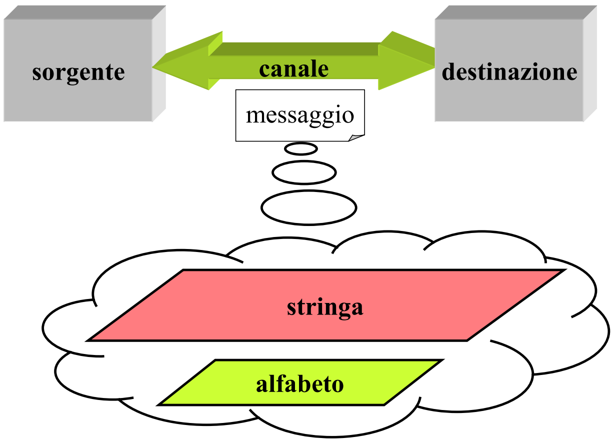
"disegno" e pend./lung. di tratti

"misura" e posizione di un indice

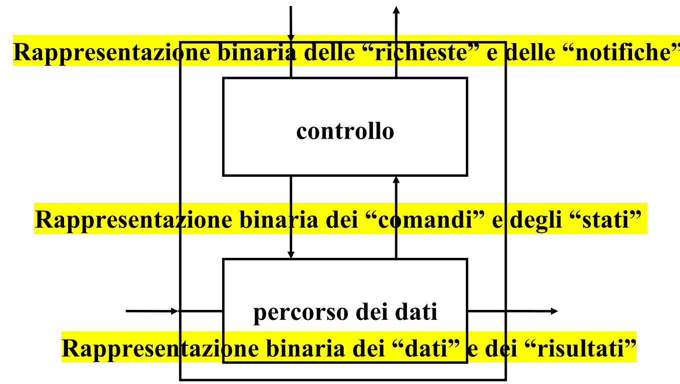
"parlato" e fonemi

<u>Simboli di informazione</u> - Variabili a cui può essere assegnato come valore uno qualsiasi degli elementi dell'alfabeto A

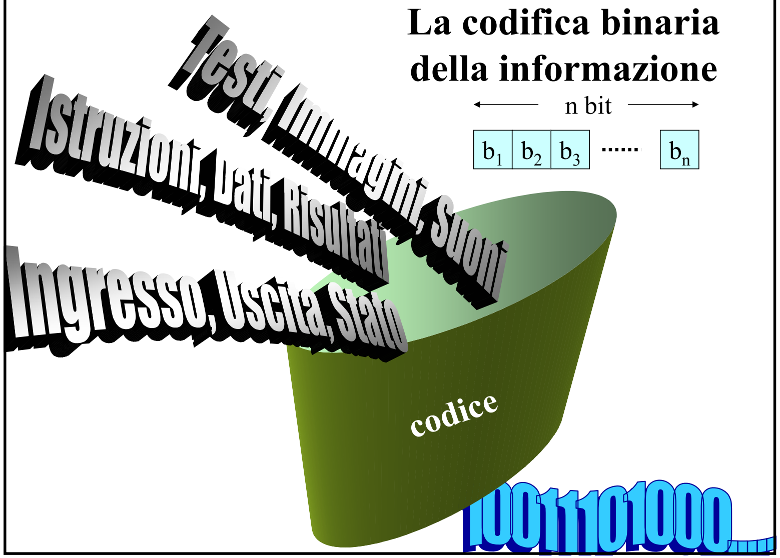
Il modello



Rappresentazione binaria dell’informazione



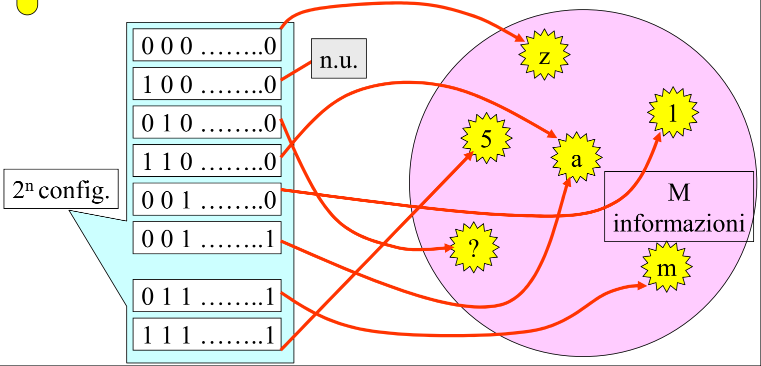
La codifica binaria della informazione



Codici binari

Codice binario

<u>Codice binario</u> - Funzione dall'insieme delle 2^n configurazioni di n bit ad un insieme di M informazioni (*simboli alfanumerici*, *colori, eventi*, *stati interni*, ecc.).
Condizione necessaria per la codifica: $2^n \geq M$



Proprietà di un codice

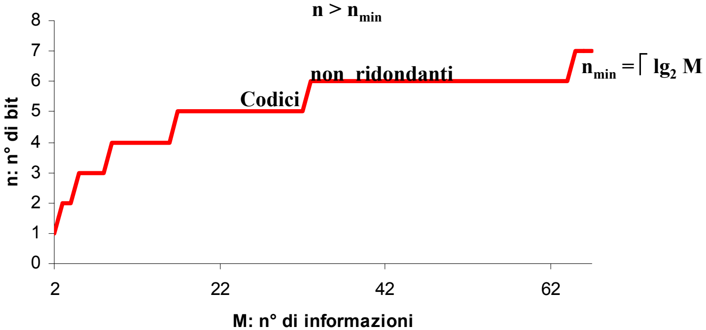
Il codice è una rappresentazione convenzionale dell'informazione.

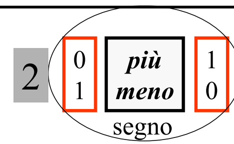
La scelta di un codice è condivisa da sorgente e destinazione ed ha due gradi di libertà:
- **il numero di bit (qualsiasi, a patto che sia $\boxed{2^n \geq M}$)**
- **l'associazione tra configurazioni e informazioni; a parità di n e di M le associazioni possibili sono**

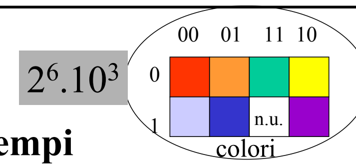
$$\boxed{C = 2^n! / (2^n-M)!}$$

n = 1, M = 2	C = 2
n = 2, M = 4	C = 24
n = 3, M = 8	C = 64.320
n = 4, M = 10	C = 29.000.000.000

Codici ridondanti e non ridondanti







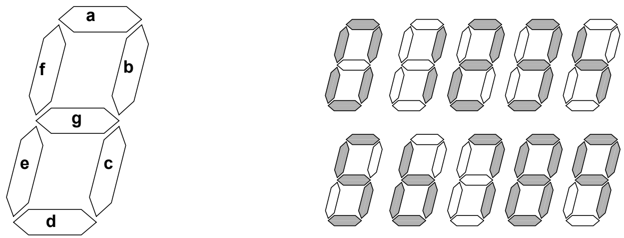
Esempi

Cifre decimali

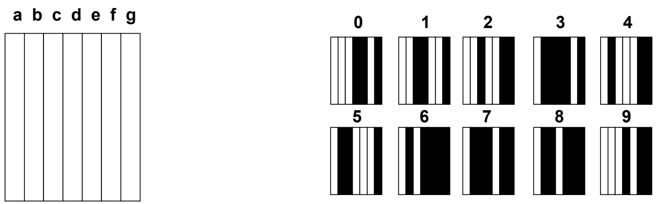
Altri 29 miliardi di codici a 4 bit	BCD		7 segmenti	uno su dieci
	0000	*zero*	1111110	1000000000
	0001	*uno*	0110000	0100000000
	0010	*due*	1101101	0010000000
	0011	*tre*	1111001	0001000000
	0100	*quattro*	0110011	0000100000
	0101	*cinque*	1011011	0000010000
	0110	*sei*	0011111	0000001000
	0111	*sette*	1110000	0000000100
	1000	*otto*	1111111	0000000010
	1001	*nove*	1110011	0000000001

N.B. 1= acceso

Codice a 7 segmenti

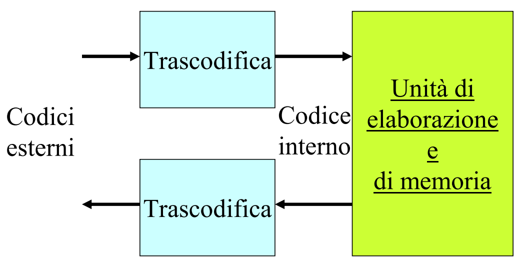


Universal Product Code



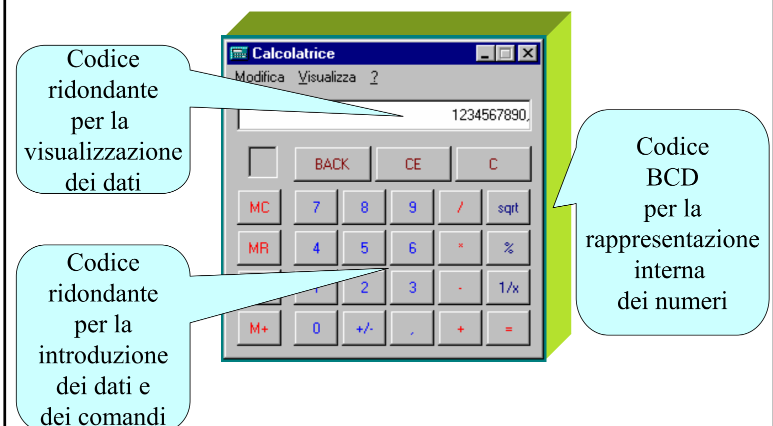
Trascodifica

La trascodifica sul percorso dei dati



Il codice interno è di norma non ridondante per minimizzare il n° di bit da elaborare e da memorizzare.
Il codice esterno è di norma ridondante, per semplificare la generazione e la interpretazione delle informazioni, e standard, per rendere possibile la connessione di macchine (o unità di I/O) fatte da Costruttori diversi.

La calcolatrice tascabile



Esempio : un editor

tastiera
↓
buffer di input
↓
memoria principale
↓
memoria video
↓
schermo

uno su enne

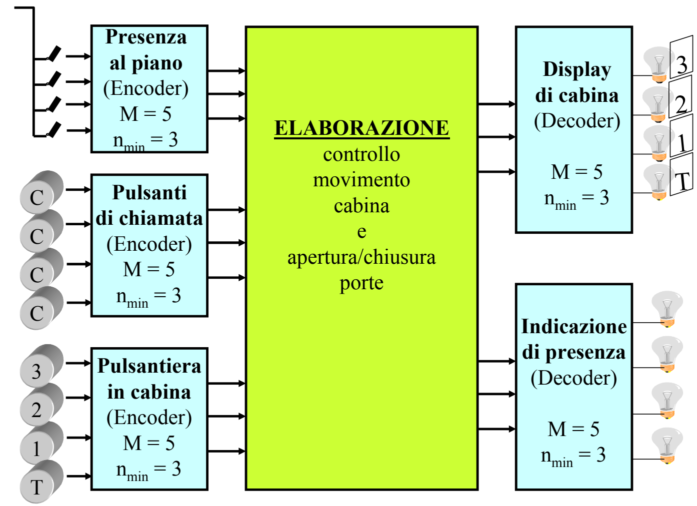
scan-code

ASCII

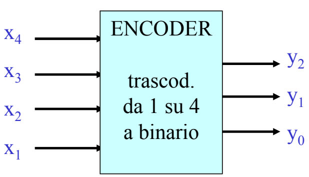
font/bit-map

Riga- Colonna - Colore

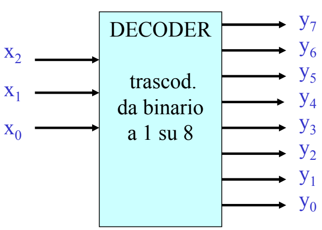
Input/output di un ascensore



Le due trascodifiche



x_4	x_3	x_2	x_1	y_2	y_1	y_0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	1	0	0	0	1	1
1	0	0	0	1	0	0



x_2	x_1	x_0	y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Codici proprietari e standard

Codice proprietario - Codice fissato da un Costruttore per mettere in comunicazione apparati da lui realizzati

• L'uso di codici proprietari ottimizza le prestazioni e protegge il mercato di certe apparecchiature.

Esempi: Linguaggio Assembler, Periferiche, Telecomando TV

Codice standard - Codice fissato da norme internazionali (***de iure***) o dal costruttore di una macchina utile per tutti gli altri (***de facto***).

• L'uso di codici standard nelle unità di I/O consente di collegare macchine fatte da costruttori diversi

Esempi: Stampanti e Calcolatori, Calcolatori e Calcolatori

2.2 La codifica dei caratteri

La codifica Morse

	t_0	t_1	t_2	t_3
E	•			
T	–			
A	•	–		
I	•	•		
N	–	•		
M	–	–		
O	–	–	–	
S	•	•	•	
R	•	–	•	
G	–	–	•	
W	•	–	–	
U	•	•	–	
K	–	•	–	

	t_0	t_1	t_2	t_3
D	–	•	•	
F	•	•	–	•
H	•	•	•	•
B	–	•	•	•
X	–	•	•	–
V	•	•	•	–
C	–	•	–	•
Y	–	•	–	–
L	•	–	•	•
J	•	–	–	–
Z	–	–	•	•
Q	–	–	•	–
P	•	–	–	•

Caratteristiche:

- Lunghezza variabile
- Stringhe separate da pause

- Efficiente per l'uso da parte di operatori umani
- Difficoltoso il progetto di ricetrasmettitori automatici

Stringhe di uguale lunghezza (Baudot: 5 bit)

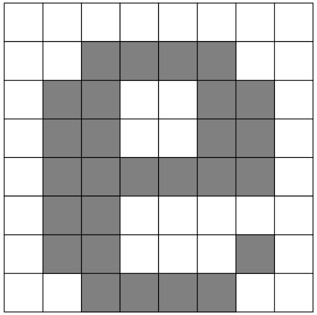
I 96 simboli di "testo" (ASCII a 7 bit)

	000	001	010	011	100	101	110	111
0000	caratteri di controllo		SP	0	@	P	'	p
0001			!	1	A	Q	a	q
0010			"	2	B	R	b	r
0011			#	3	C	S	c	s
0100			$	4	D	T	d	t
0101			%	5	E	U	e	u
0110			&	6	F	V	f	v
0111			'	7	G	W	g	w
1000			(	8	H	X	h	x
1001			)	9	I	Y	i	y
1010			*	:	J	Z	j	z
1011			+	;	K	[	k	{
1100			,	<	L	\	l	|
1101			-	=	M	]	m	}
1110			.	>	N	^	n	~
1111			/	?	O	_	o	DEL

Codice ASCII esteso (8 bit)

N.B. - Lo standard Unicode (16 bit) consente di rappresentare diversi sottoinsiemi di caratteri.

Bit map: un codice ridondante per simboli alfanumerici



Matrice di pixel: ad es. 8x8

Bianco/nero:
1 pixel, 1 bit

Tonalità:
1 pixel, 8 bit

Colori RGB:
1 pixel, 24 bit

Font

2.3 La codifica dei numeri

Numeri in base B

1) Rappresentazione:

$a_{n-1} \ldots\ldots a_0 , a_{-1} \ldots\ldots a_{-m}$

$a_i \in \{0, 1, \ldots, (B-1)\}$

2) Valore:

$$(N)_B = (a_{n-1} .B^{n-1} + \ldots + a_0 .B^0 + a_{-1} .B^{-1} + a_{-2} .B^{-2} + \ldots a_{-m} .B^{-m})$$

Il sistema di numerazione in base 2 (il caso dei numeri naturali < 2^n)

n bit

b_{n-1} b_{n-2} ...... b_1 b_0

$$(N)_2 = b_{n-1} .2^{n-1} + b_{n-2} .2^{n-2} + \ldots + b_0 .2^0$$

N_{10}	N_2	N_{10}	N_2
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

Lunghezza della stringa in base 2 e in base 10

Dato un numero decimale con m cifre

$$0 \le (N)_{10} \le 10^m - 1$$

per la sua rappresentazione binaria deve essere $2^n > 10^m$ e quindi

$$n = \lceil (m \times \log_2 10) \approx \lceil (3{,}32\ m)$$

Conversione di base

Conversioni da base 2 a base 10

ESEMPIO: 100110

0 +
2 +
4 +
0 +
0 +
32 =
38

Conversione da base 2 a base 10

$(N)_{10} = (b_{n-1} .2^{n-1} + b_{n-2} .2^{n-2} + \ldots + b_1 .2^1 + b_0 .2^0)_{10}$

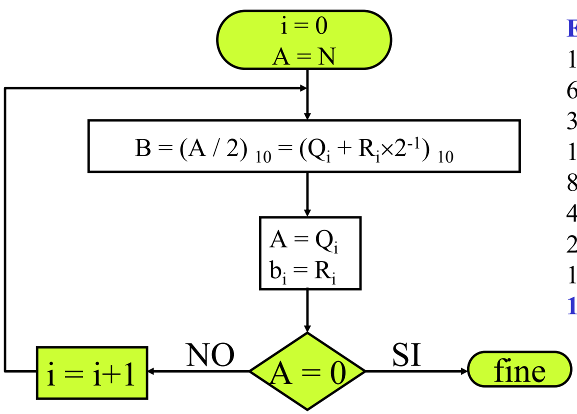
Osservazione:

$(N)_{10} = (b_{n-1} .2^{n-1} + b_{n-2} .2^{n-2} + \ldots + b_1 .2^1 + b_0 .2^0)_{10}$

$(N)_{10}/2 = (b_{n-1} .2^{n-2} + b_{n-2} .2^{n-3} + \ldots + b_1 .2^0) + (b_0 .2^{-1})_{10}$

$= \quad Q \quad + \quad R.\ 2^{-1}$

Conversione di un numero naturale N da base 10 a base 2



ESEMPIO: 131

131/2 = 65 +1.0,5
65 = 32 + 1.0,5
32/2 = 16 + 0
16/2 = 8 + 0
8/2 = 4 + 0
4/2 = 2 + 0
2/2 = 1 + 0
1/2 = 0 + 1.0,5
10000011

Altre rappresentazioni di numeri binari

- **Sistema esadecimale**: B =16
  cifre: 0,1,..,9,a,b,c,d,e,f
  codice binario: 0 = 0000, 1 = 0001, …, f = 1111
  n° di bit per cifra: 4

ESEMPIO: 11000100 → 1100-0100 → C4

- **Sistema ottale**: B = 8,
  cifre: 0, 1, …,7
  codice OCTAL: 0 = 000, …, 7 = 111
  n° di bit per cifra: 3

ESEMPIO: 11000100 → 11-000-100 → 304

Operazioni aritmetiche

Addizione (carry)

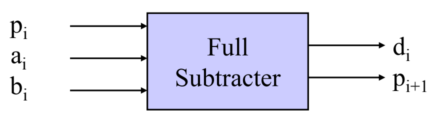
0+0 = 00
0+1 = 01
1+0 = 01
1+1 = 10

r_n	r_{n-1}	r_{n-2}	......	r_1	**0**	
	a_{n-1}	a_{n-2}	......	a_1	a_0	+
	b_{n-1}	b_{n-2}	......	b_1	b_0	
s_n	s_{n-1}	s_{n-2}	......	s_1	s_0	

r_i a_i b_i	r_{i+1}	s_i
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

Sottrazione (borrow)

0-0 = 0
0-1 = n.a.
1-0 = 1
1-1 = 0



p_n	p_{n-1}	p_{n-2}	......	p_1	**0**	
0	a_{n-1}	a_{n-2}	......	a_1	a_0	-
0	b_{n-1}	b_{n-2}	......	b_1	b_0	
d_n	d_{n-1}	d_{n-2}	......	d_1	d_0	

p_i a_i b_i	p_{i+1}	d_i
0 0 0	0	0
0 0 1	1	1
0 1 0	0	1
0 1 1	0	0
1 0 0	1	1
1 0 1	1	0
1 1 0	0	0
1 1 1	1	1

Moltiplicazione

0.0 = 0
0.1 = 0
1.0 = 0
1.1 = 1

$$
\begin{aligned}
P &= Y.X \\
&= Y.(x_{n-1}.2^{n-1} + \ldots + x_1.2^1 + x_0.2^0) \\
&= (Y.x_{n-1}).2^{n-1} + \ldots + (Y.x_1).2^1 + (Y.x_0).2^0 \\
&= P_{n-1}.2^{n-1} + \ldots + P_1.2^1 + P_0.2^0
\end{aligned}
$$

$$P_i = (Y.x_i) = (y_{n-1}.x_i).2^{n-1} + \ldots + (y_0.x_i).2^0$$

Moltiplicazione (shift and add)

				y_3	y_2	y_1	y_0	×
				x_3	x_2	x_1	x_0	
				x_0y_3	x_0y_2	x_0y_1	x_0y_0	+
			x_1y_3	x_1y_2	x_1y_1	x_1y_0	**0**	+
		x_2y_3	x_2y_2	x_2y_1	x_2y_0	**0**	**0**	+
	x_3y_3	x_3y_2	x_3y_1	x_3y_0	**0**	**0**	**0**	+
p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0	

Rappresentazione dei numeri razionali

- Come coppia di interi (più un bit per il segno)

i_n	i_{n-1}	...	i_0	f_1	f_2	...	f_n

s	i_n	i_{n-1}	...	i_0	f_1	f_2	...	f_n

- Notazione *scientifica*

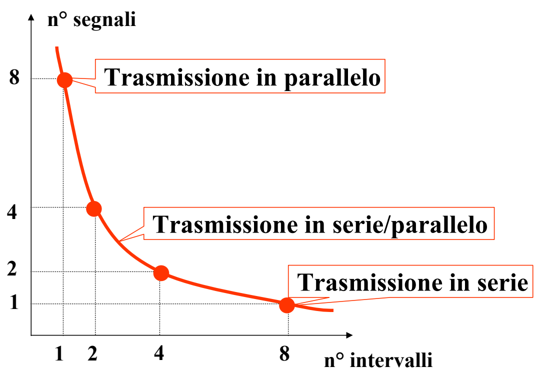
s	m_n	m_{n-1}	...	m_0	s	e_n	e_{n-1}	...	e_0

2.4
Trasmissione

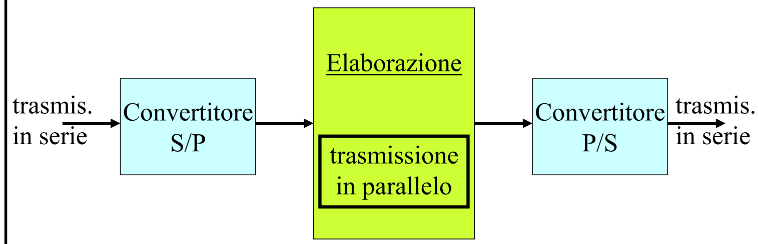
Modalità

Modalità di trasmissione dei bit: compromesso spazio/tempo

Es.: Codice a 8 bit



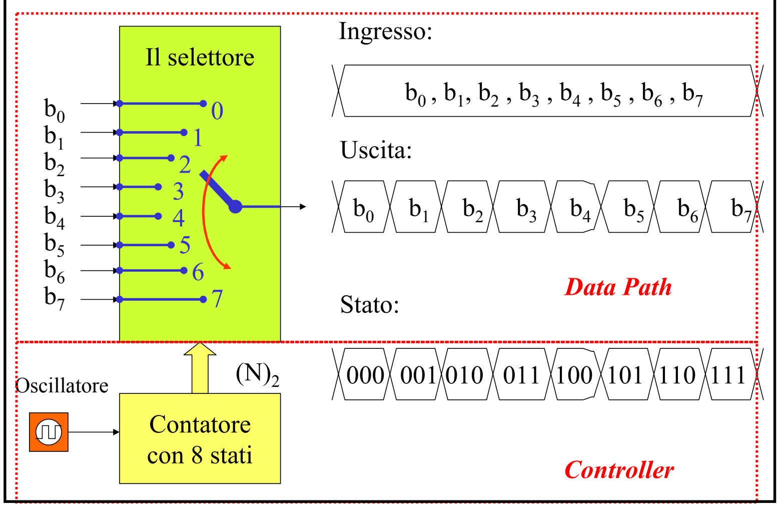
Modalità di trasmissione dei bit: le unità di conversioni S/P e P/S



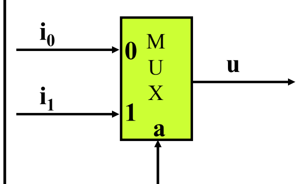
La modalità di trasmissione all'interno della macchina è di norma in parallelo (per massimizzare la velocità di elaborazione)

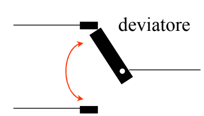
La modalità di trasmissione all'esterno della macchina è di norma in serie (per minimizzare la complessità del supporto fisico)

La conversione P/S di un byte



La serializzazione di due bit



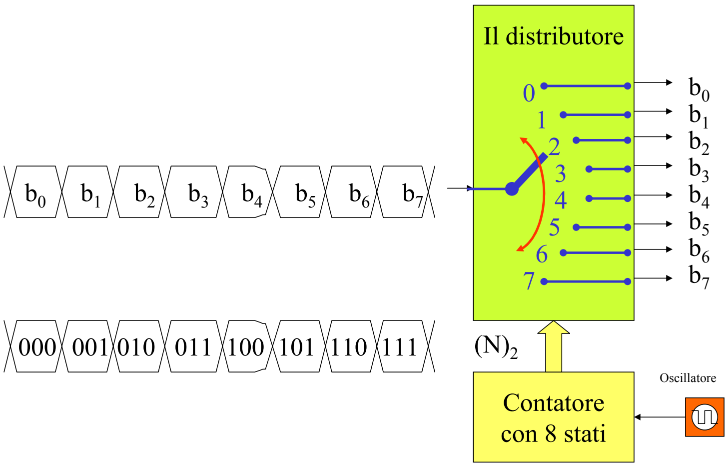


a	i_0	i_1	u
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

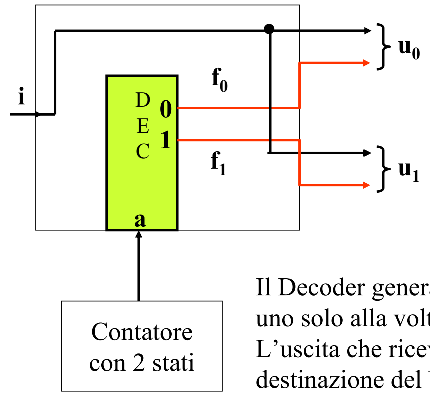
***se* a=0 *allora* u=i_0**

***altrimenti* u=i_1**

Conversione S/P di un byte



La distribuzione di due bit

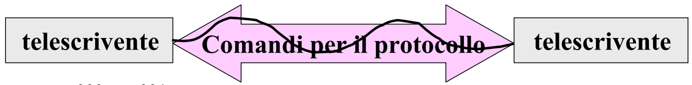


a	f_0	f_1
0	1	0
1	0	1

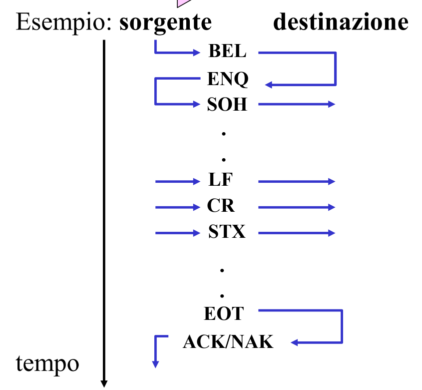
Il Decoder genera **2** “flag di validità”, di cui uno solo alla volta ha valore 1.
L’uscita che riceve tale valore è la destinazione del bit d’ingresso **i**

Protocolli

Modalità di controllo (ASCII a 7 bit) : codifica dei comandi e protocollo di scambio



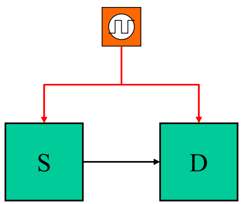
	000	001
0000	**NUL**	**DLE**
0001	**SOH**	**DC1**
0010	**STX**	**DC2**
0011	**ETX**	**DC3**
0100	**EOT**	**DC4**
0101	**ENQ**	**NAK**
0110	**ACK**	**SYN**
0111	**BEL**	**ETB**
1000	**BS**	**CAN**
1001	**HT**	**EM**
1010	**LF**	**SUB**
1011	**VT**	**ESC**
1100	**FF**	**FS**
1101	**CR**	**GS**
1110	**SO**	**RS**
1111	**SI**	**US**

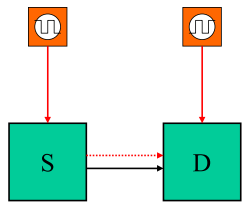


Sincronizzazione

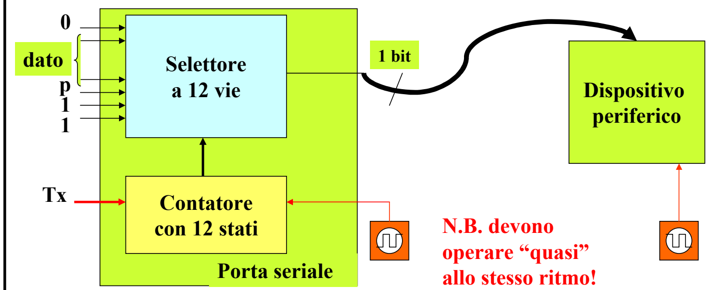
La destinazione deve sapere in quali istanti di tempo i valori presenti sul canale sono significativi.
Si hanno due casi:

"accoppiamento stretto" ***"accoppiamento lasco"***





Comunicazione asincrona di un byte: il protocollo RS232

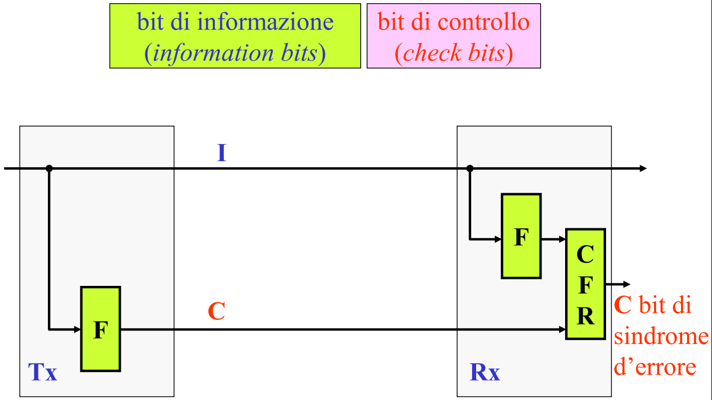


Disturbi e Guasti

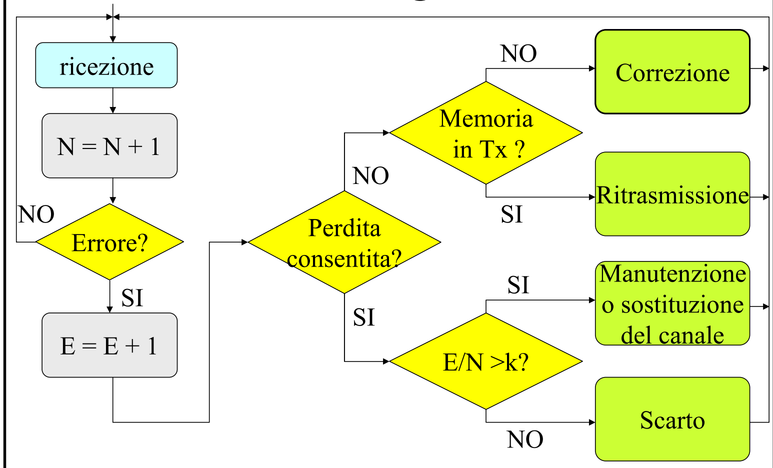
Obiettivo: riconoscere alla destinazione le configurazioni modificate.
Condizione necessaria: **il codice deve essere ridondante**
- se arriva una configurazione "non utilizzata" la destinazione ha la **certezza** della sua non integrità;
- se arriva una configurazione "utilizzata" la destinazione ha solo una certa **probabilità** che sia integra.

Ulteriore condizione - Le configurazioni non utilizzate devono essere le modifiche più probabili delle configurazioni utilizzate.

Codici separabili: rilevazione di errori



Gestione degli errori



Codici con rilevazione di errori

L'ipotesi degli errori indipendenti

Consideriamo una stringa di ***n*** bit e supponiamo che l'evento di modifica di un bit (o *errore*) da parte di un disturbo
a) sia indipendente dalla posizione del bit nella stringa;
b) si verifichi con probabilità pari a ***p*** *(tasso di errore)*.
La probabilità che la stringa ricevuta contenga ***e*** errori è data da:

$$P_e = \binom{n}{e} \cdot p^e \cdot (1-p)^{n-e}$$

Esempio:
p = *1 %*
N.B. molto alto!

n	P_0	P_1	P_2	P_3
8	*92,27 %*	*7,46 %*	*0,26 %*	*0,005 %*
16	*85.14 %*	*13,76 %*	*1,04 %*	*0,049 %*

Per n = 8 le modifiche più probabili riguardano un solo bit
Per n = 16 le modifiche più probabili riguardano uno o due bit

Distanza minima di un codice

Distanza fra due configurazioni binarie di n bit: **D(A,B) - Numero di bit omologhi che hanno valore diverso.**

Esempi: D(100,101) = 1; D(011,000) = 2; D(010,101) = 3

Distanza minima di un Codice C: **DMIN (C) - Valore minimo della distanza tra due qualsiasi delle configurazioni utilizzate.**

Esempi: DMIN (Codice ASCII) = 1; DMIN (Codice semaforo) = 2

- **I codice non ridondanti hanno DMIN=1.**
- **I codice ridondanti possono avere DMIN > 1.**

Esempio: DMIN (UPC) = 2

Distanza minima e rilevazione degli errori

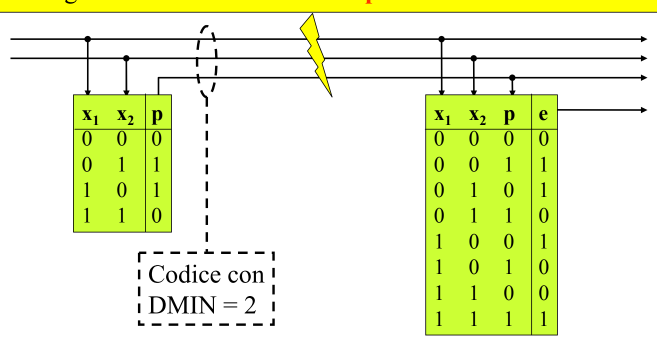
- Un codice per la rilevazione di tutti i possibili errori singoli, o **SEDC (Single Error Detection Code)**, deve non utilizzare tutte le configurazioni che distano "uno" da ciascuna delle configurazioni utilizzate.

Un codice SED deve dunque avere almeno **DMIN = 2** .

- Un codice per la rilevazione di modifiche su k bit deve avere almeno **DMIN = k+1** .

Il bit di parità : una semplice modalità per ottenere la rilevazione di errori singoli

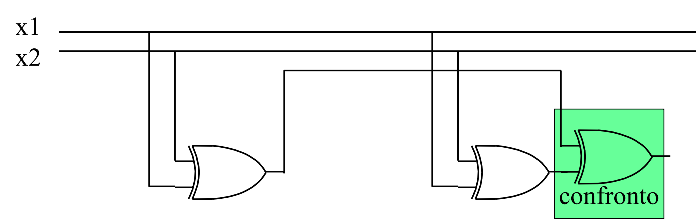
Bit di parità **p** - bit che la sorgente aggiunge ad una stringa di bit di codifica al fine di renderne **pari** il n° di "uni".
Errore di parità **e** - bit che la destinazione pone a 1 se e solo se riceve una configurazione con un numero **dispari** di "uni".



x_1	x_2	p
0	0	0
0	1	1
1	0	1
1	1	0

x_1	x_2	p	e
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Calcolo del bit di parità



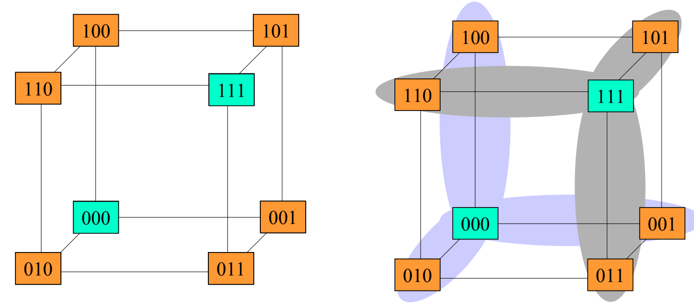
p = F(x1, x2)

e = F(x1, x2, p)
= F(F(x1, x2), p)

- Funzione composta
- Disposizione in serie

Codici con correzione di errori

La correzione di errori singoli (esempio)



Tx trasmette
o NO = 000 o SI = 111

A causa dei disturbi Rx può ricevere una terna qualsiasi

Se P_1>> P_2 ogni configurazione errata può essere corretta

Distanza minima e correzione degli errori

Il codice dell'esempio precedente ha DMIN=3.

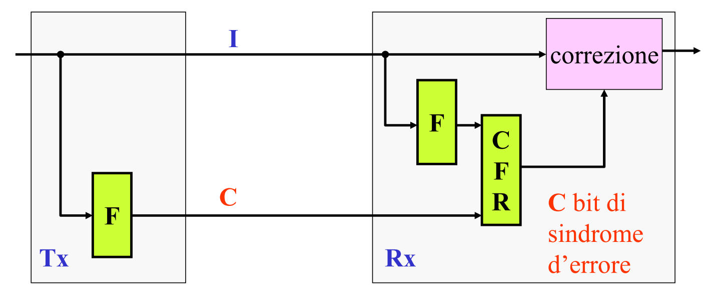
• Ogni **SECC** (<u>S</u>ingle <u>E</u>rror <u>C</u>orrection <u>C</u>ode) deve avere **DMIN ≥ 3**.

• Un codice con **DMIN = 2k+1** rileva **2k** errori e può correggerne fino a **k** .

Di solito <u>si corregge un solo bit</u> e si usa la ridondanza introdotta per valutare la "qualità" del canale (manutenzione/sostituzione)

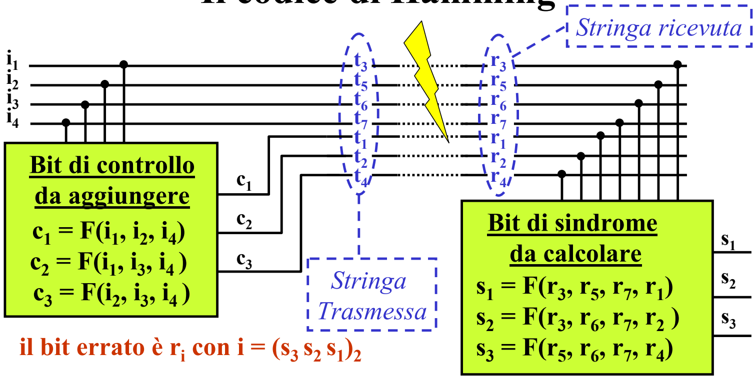
Codici separabili: correzione di errori

Hamming (Bell Labs, 1950) ha dimostrato che per correggere gli errori singoli su informazioni codificate con **I** bit occorrono **C** bit di controllo tali che **$2^C \geq I + C + 1$** .



Le 2^C configurazioni delle **sindromi di errore** devono indicare se non c'è errore (1 situazione) e se c'è, dov'è (I + C situazioni).

Il codice di Hamming



il bit errato è r_i con $i = (s_3\, s_2\, s_1)_2$

ES.:	I = 0001	R=0011
	$c_1 = 1$	$s_1 = 0$
	$c_2 = 1$	$s_2 = 1$
	$c_3 = 1$	$s_3 = 1$