# Capitolo 2
# Codifica binaria dell'informazione

2.1 – Rappresentazione dell'informazione
2.2 – Codifica di caratteri
2.3 – Codifica dei numeri
2.4 – Trasmissione dell'informazione
2.5 – Protezione dell'informazione

# 2.1
# Rappresentazione dell'informazione

# Simbolo, alfabeto e stringa

**Informazione** - **Stringa** di lunghezza finita formata da **simboli** appartenenti ad un **alfabeto** di definizione $A$:

$s_1\ s_2 \ldots s_i \ldots s_{n-1}\ s_n$ con $s_i \in A : \{a_1, a_2, .., a_m\}$

**Alfabeto A: insieme di informazioni "elementari"**

**Esempi:**

"testo" e caratteri

"numero" e cifre

"immagine" e pixel/colore-toni di grigio

"parlato" e fonemi

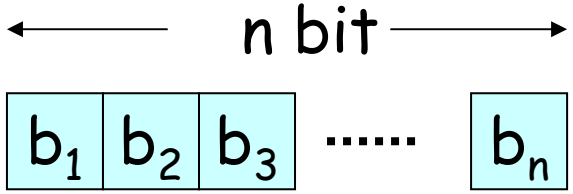
"musica" e note

"disegno" e pendenza/lunghezza di tratti

"misura" e posizione di un indice

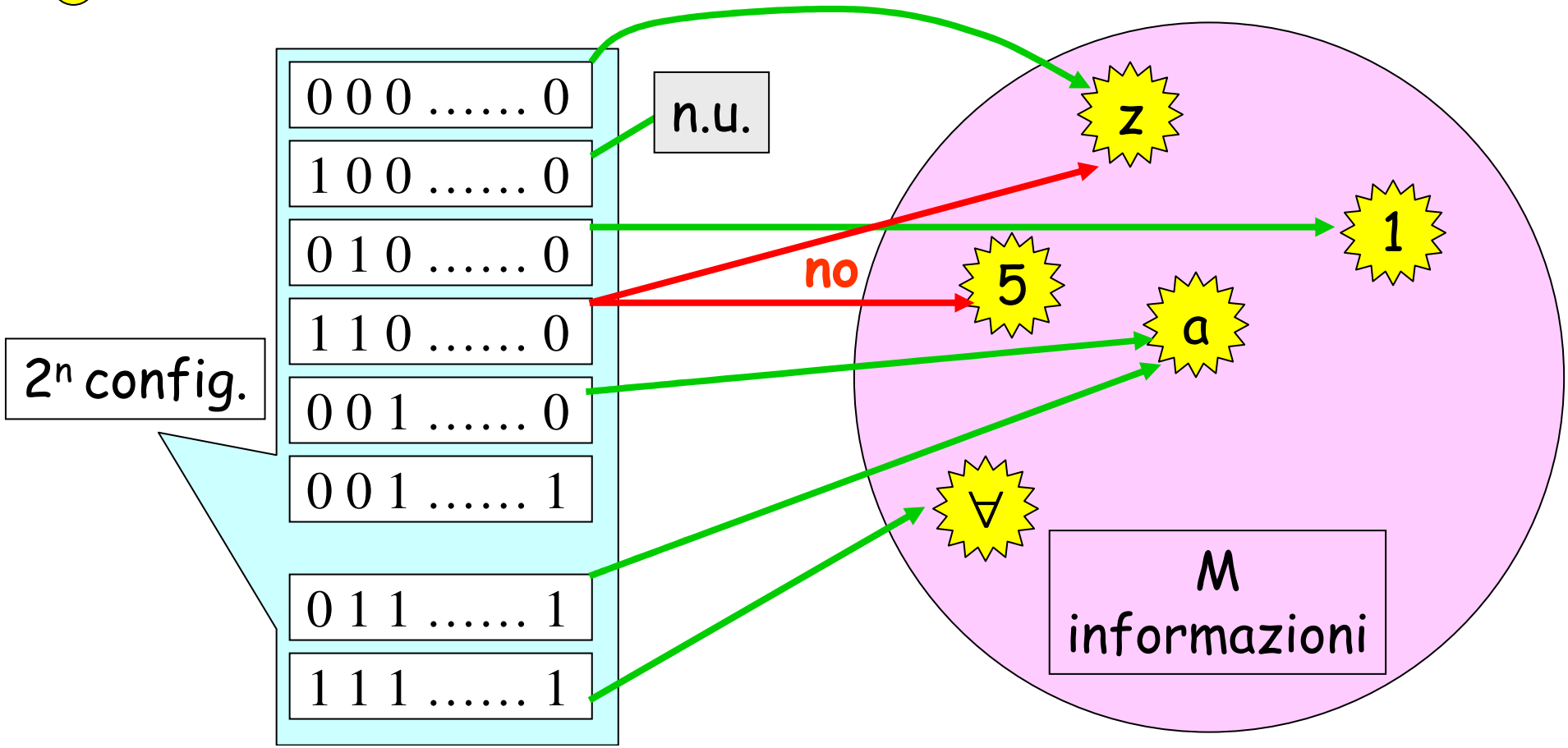
...

# La codifica binaria della informazione



# Codice binario

**Codice binario - 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

**Codice: rappresentazione convenzionale dell'informazione.**

La scelta di un codice è condivisa da sorgente e destinazione, ed ha due gradi di libertà:

- il numero n di bit (qualsiasi, purché $2^n \geq M$)
- l'associazione tra configurazioni e informazioni

A parità di n e di M, le associazioni possibili sono:

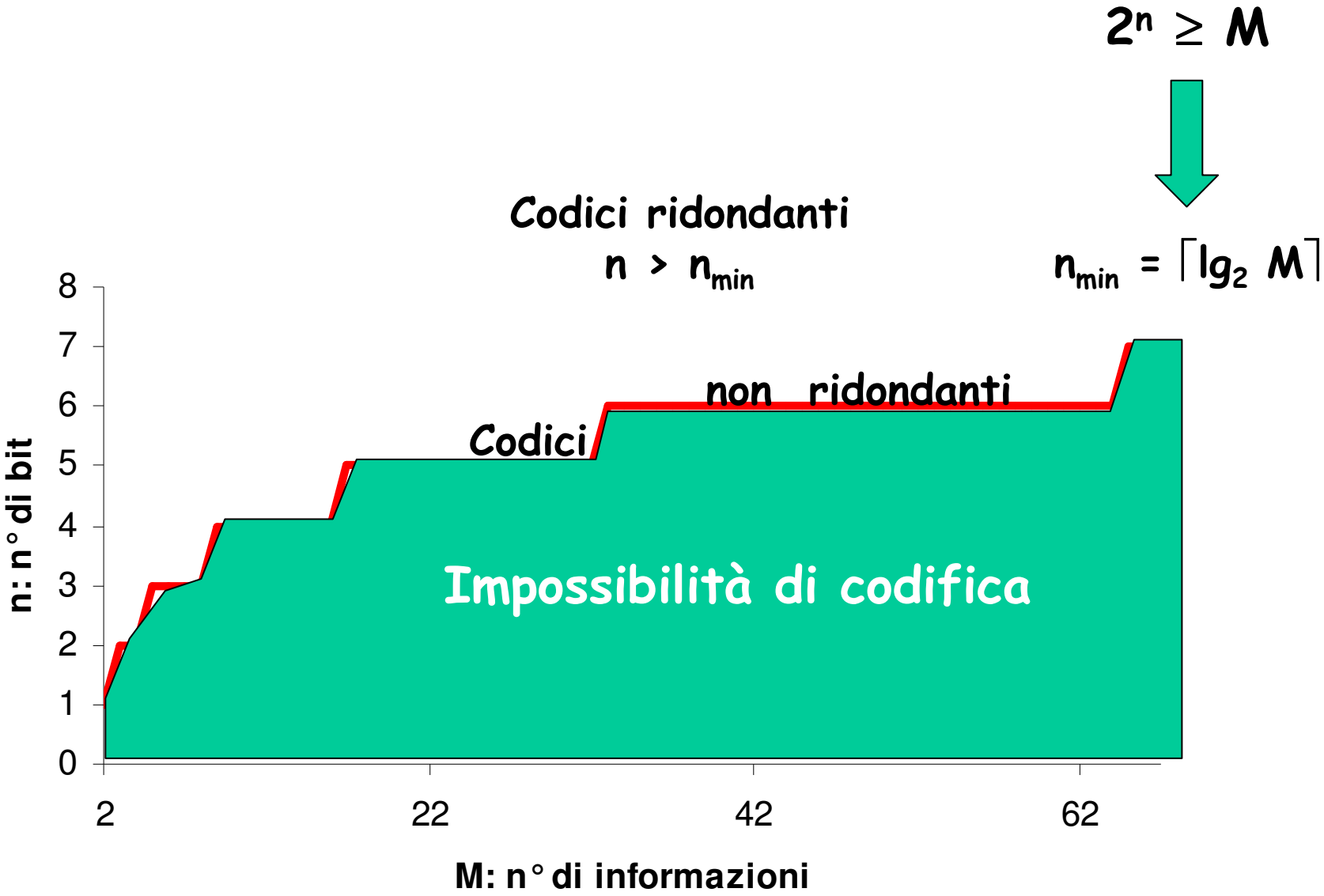
$$C = 2^n! / (2^n - M)!$$

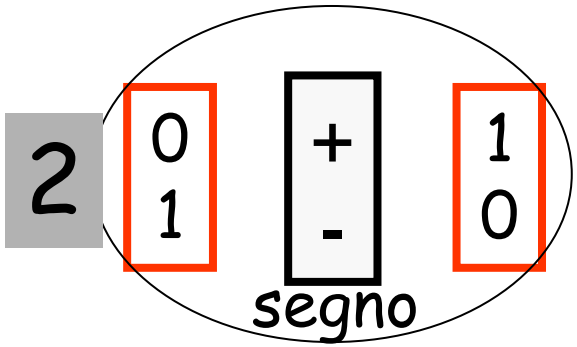
| | |
|---|---|
| n = 1, M = 2 | C = 2 |
| n = 2, M = 4 | C = 24 |
| n = 3, M = 8 | C = 40.320 |
| n = 4, M = 10 | C = 29.059.430.400 |

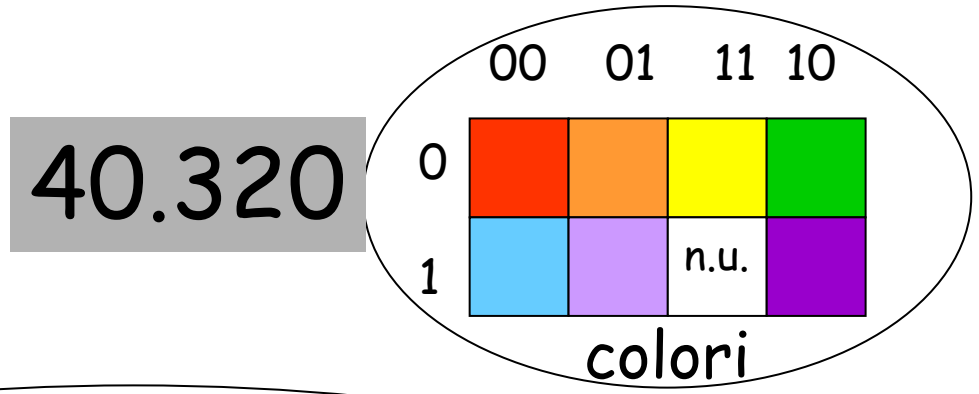
# Codici ridondanti e codici non ridondanti

$2^n \geq M$

$n_{min} = \lceil lg_2\ M \rceil$





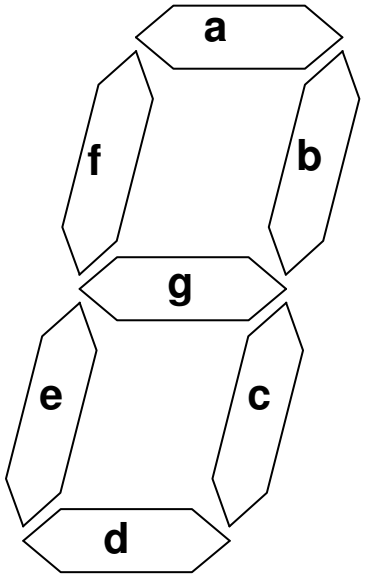



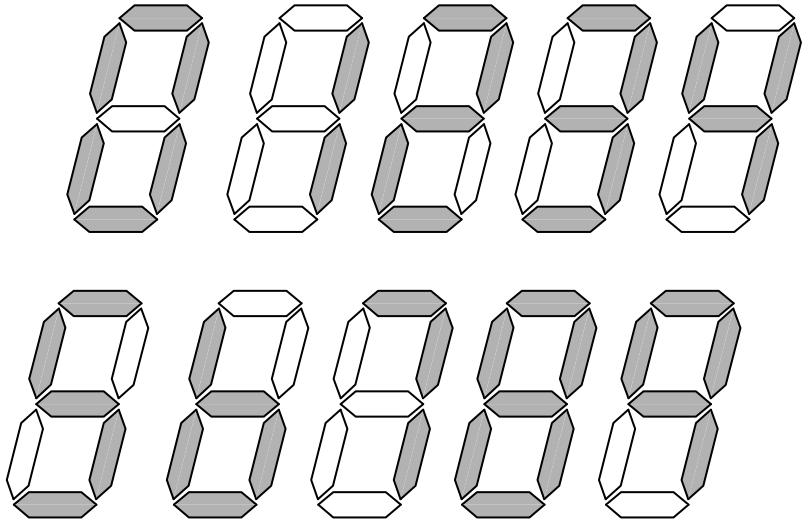

cifre decimali

altri
29.059.430.399
codici
a 4 bit

| BCD | | 7 segmenti | 1/10 |
|---|---|---|---|
| 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 |

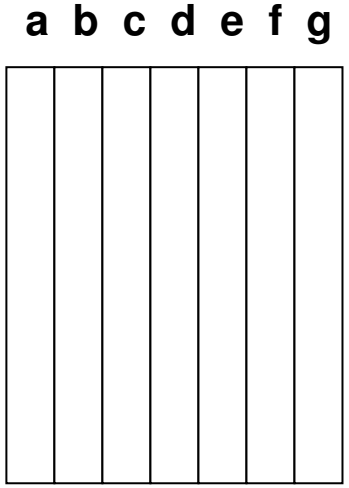
# Codice a 7 segmenti

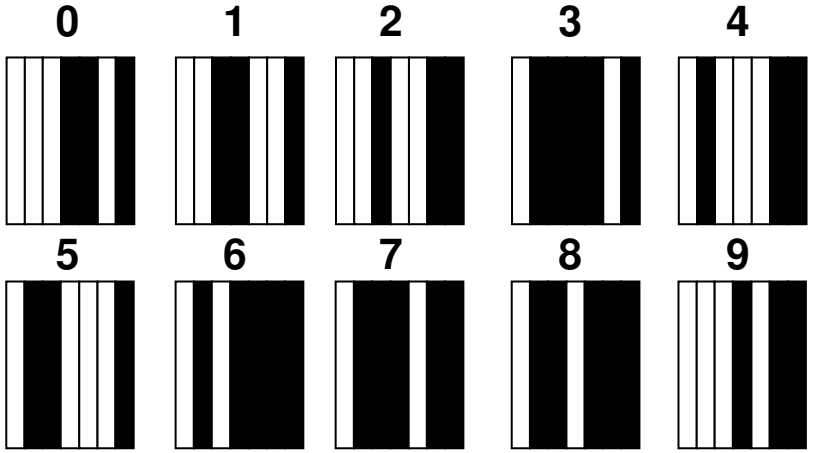


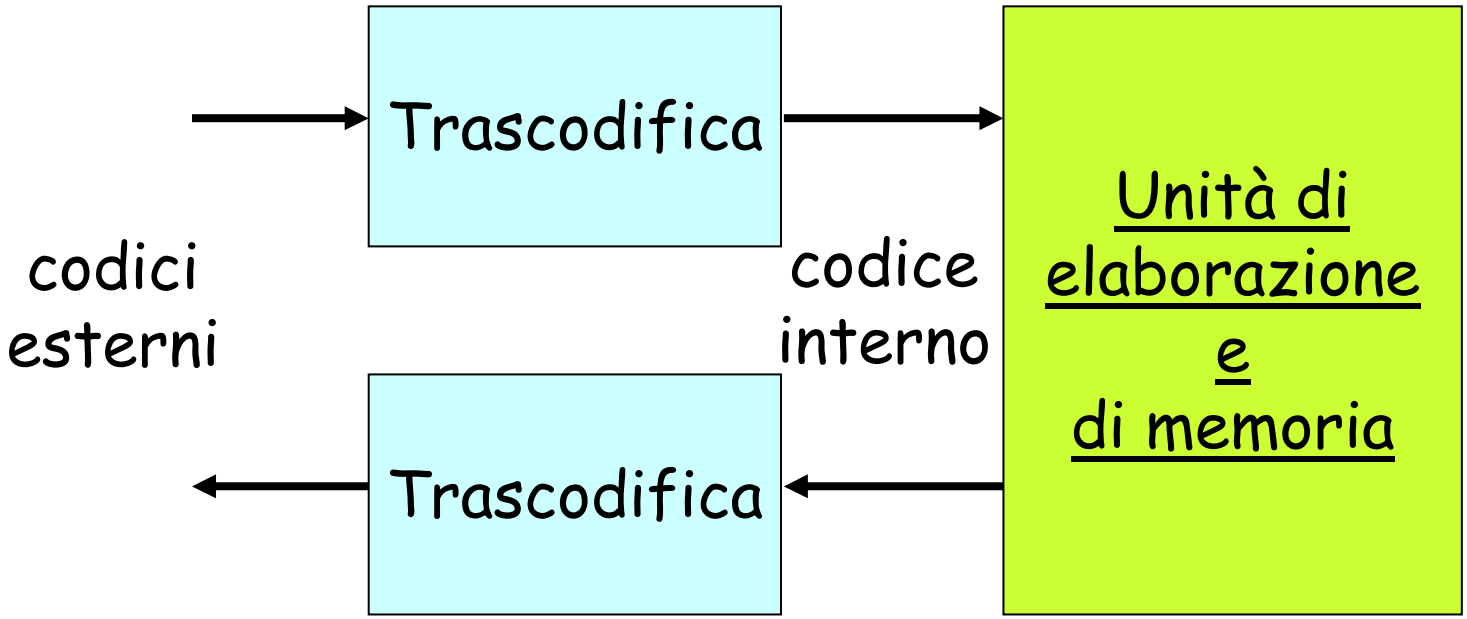
| | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|
| zero | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| uno | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| ecc. | | | | | | | |

# Universal Product Code



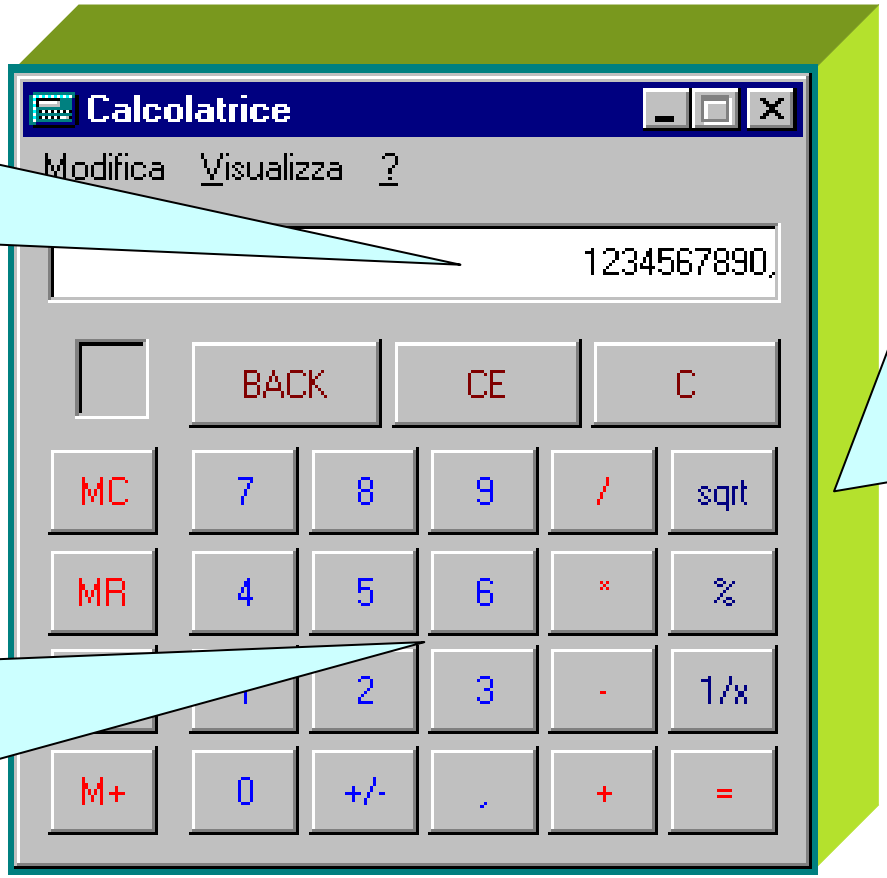



# La conversione di codice (trascodifica)




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

# La calcolatrice tascabile




Codice ridondante per la visualizzazione dei dati

Codice ridondante per la introduzione dei dati e dei comandi

Codice BCD per la rappresentazione interna dei numeri

# Codici proprietari e codici standard

**Codice proprietario** - Codice fissato da un Costruttore per mettere in comunicazione macchine di sua produzione.
L'uso di codici proprietari mira ad ottimizzare le prestazioni e a proteggere il mercato di certe macchine.

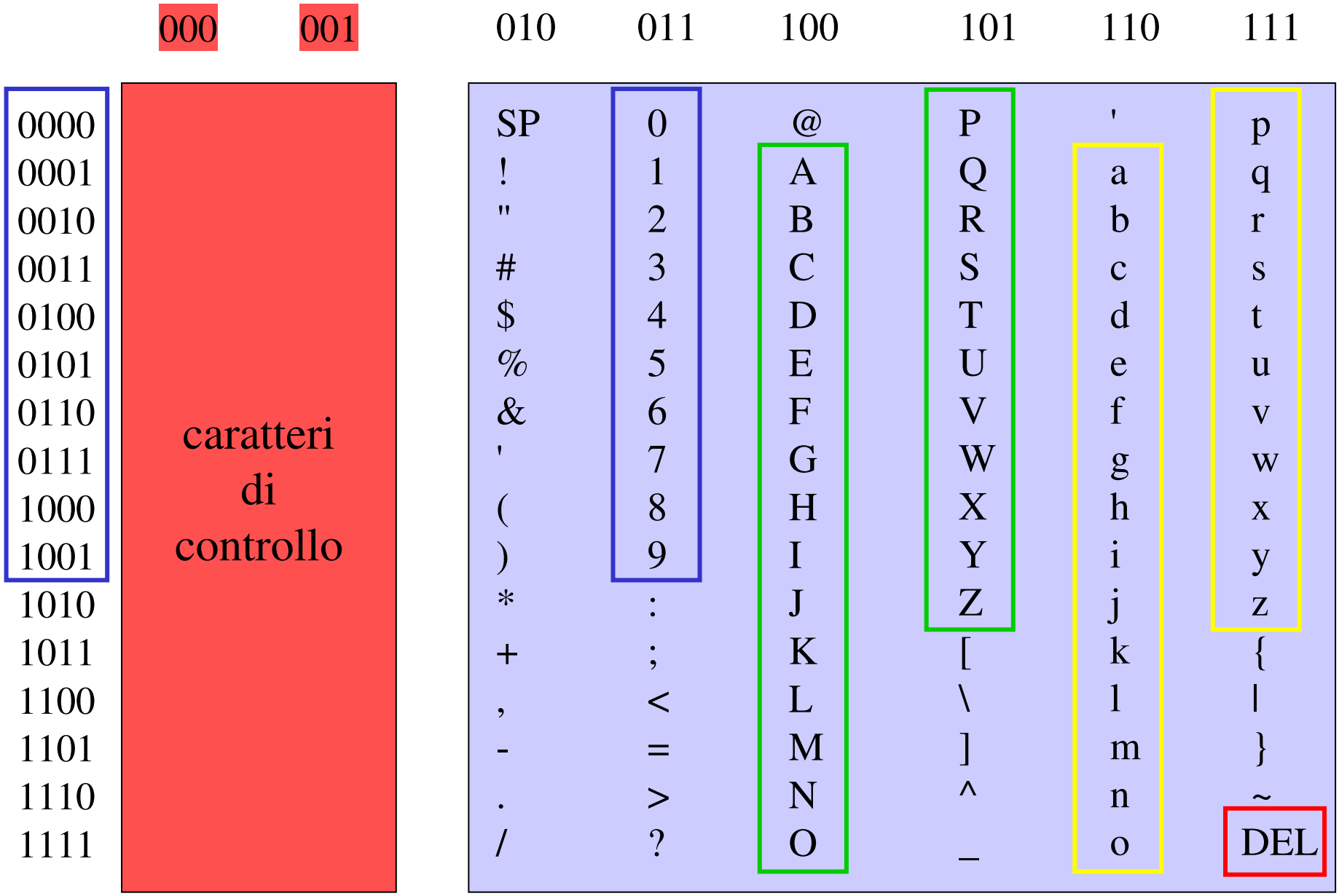
*Esempi: Linguaggio Assembler, Periferiche, Telecomando TV*

**Codice standard** - Codice fissato da norme internazionali (***de iure***) o dal Costruttore di una macchina ampiamente utilizzata sul mercato (***de facto***).
L'uso di codici standard nelle unità di I/O consente di collegare macchine realizzate da Costruttori diversi.

*Esempi: Stampanti e Calcolatori, Calcolatori e Calcolatori*

# 2.2
# La codifica dei caratteri

# Il codice ASCII a 7 bit (1967)



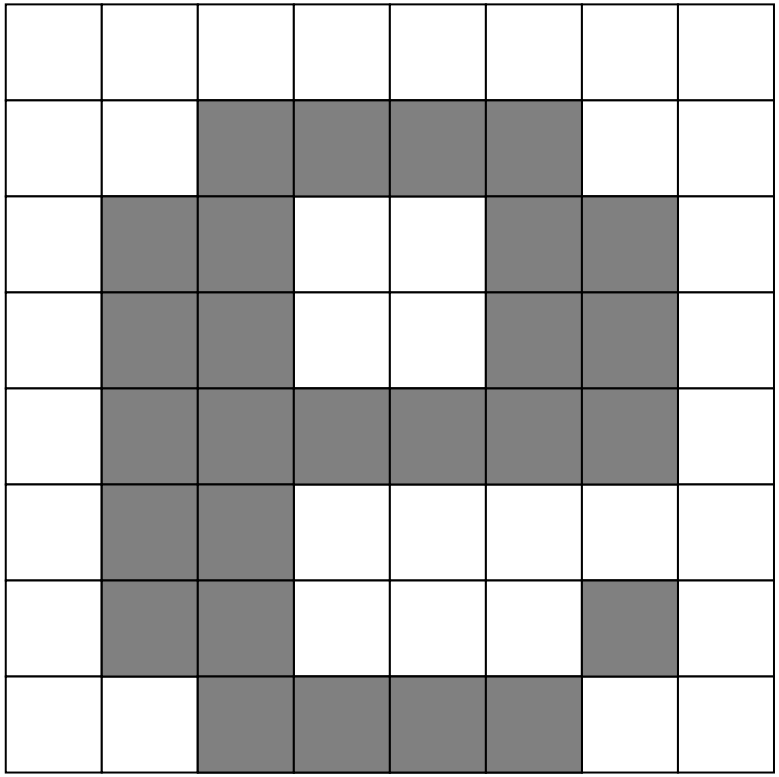
# Codice ASCII esteso (8 e 16 bit)


Lo standard Unicode (16 bit) codifica
in binario i caratteri di tutte le lingue !

# *Bit map*: un codice d'uscita ridondante per simboli alfanumerici

Stampanti ad impatto: *ASCII*

Stampanti laser, a getto, monitor: BITMAP



Bianco/nero:
1 pixel, 1 bit

Tonalità:
1 pixel, 8 bit

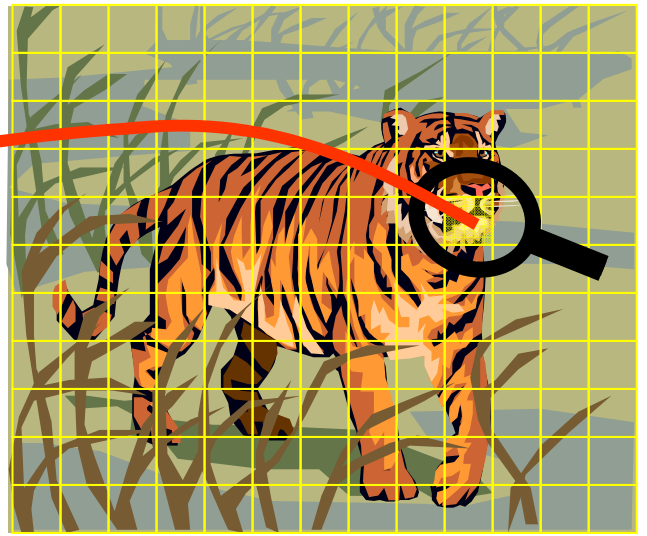
Colori RGB:
1 pixel, 3x8 bit

Font

Matrice di pixel ("picture element"): ad es. 8x8

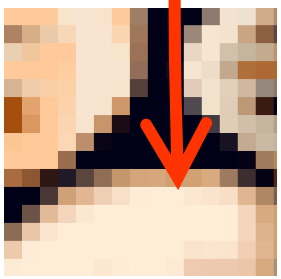
# Codifica di immagini

Scena reale



480

720



R ∈ {0,1,2,..,254,255}
G ∈ {0,1,2,..,254,255}
B ∈ {0,1,2,..,254,255}

24 bit/pixel

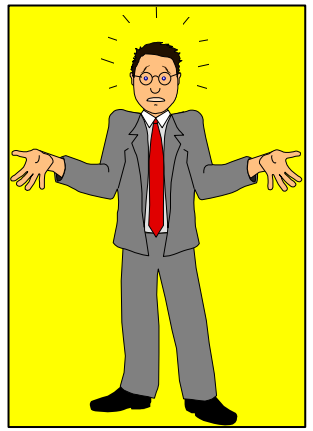
Immagine digitalizzata: 720 * 480 * 24 = 8.294.400 ≅ 8 Mbits

# Codifica di immagini video

In tale ottica, la registrazione di un film della durata di 2 ore comporta una capacità di memoria pari a (30 frame/s):

2 * 60 * 60 * 30 * 8 Mbits = 1728 Gbits

Essendo la capacità di un DVD pari a 37.6 Gbits,
è sufficiente (!?) allo scopo dotarsi di 46 DVD.
A meno che …



# Tecniche di compressione

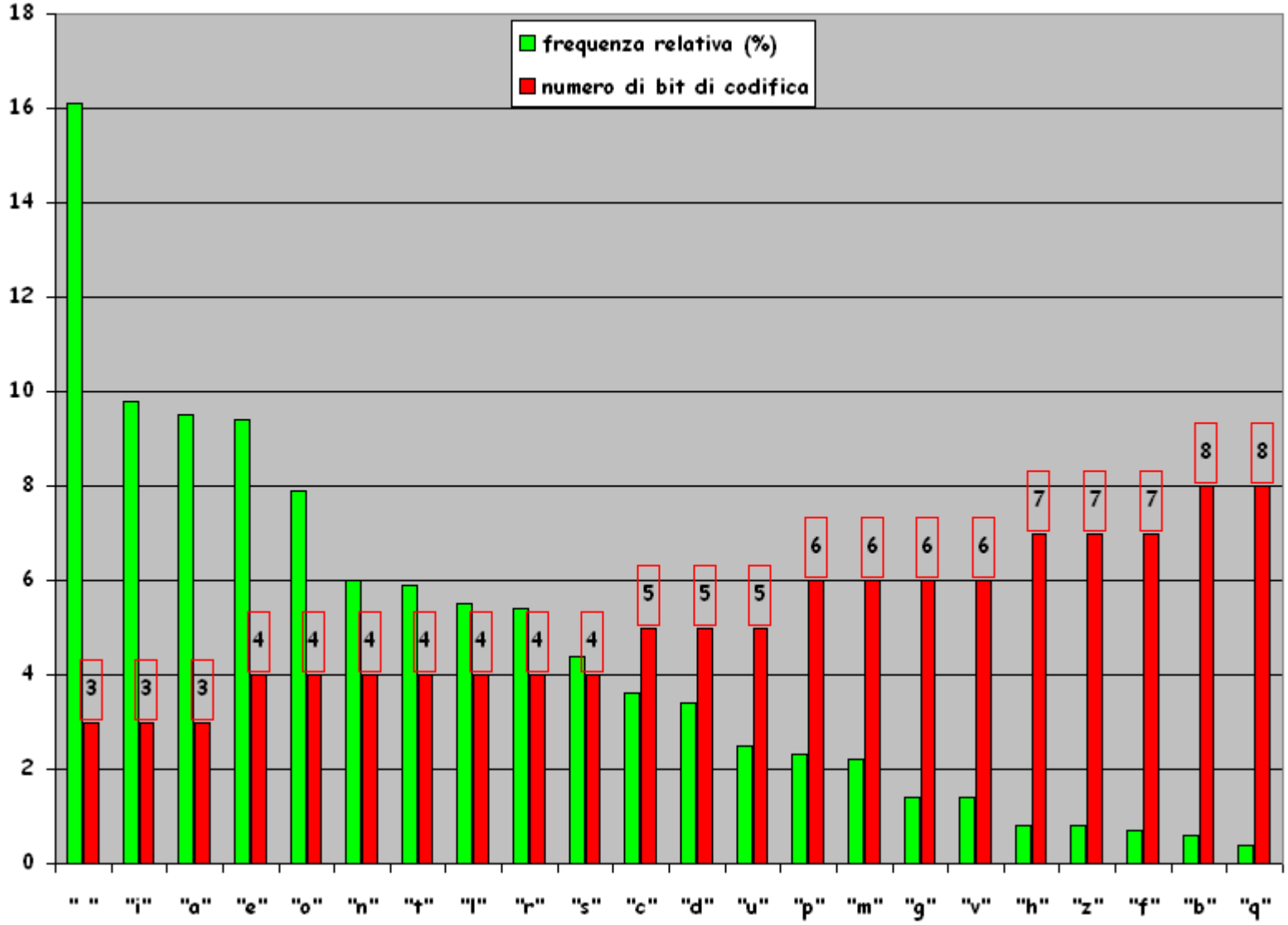
Tipologie:
con perdita di informazione (lossy compression)
senza perdita di informazione (lossless compression)

Contesto tipico, rispettivamente:
elaborazione di immagini
elaborazione di testi

Efficienza o fattore di compressione:

$$\frac{\text{Size [Original (Uncompressed) Info]}}{\text{Size [Compressed Info]}}$$

# Il codice (lossless) di Huffman




Libro (qualsiasi) scritto in lingua italiana: $FC = 5 / \sum_{s_i \in A} n(s_i)\, f(s_i) = 1.25$

# 2.3
# La codifica dei numeri

# Rappresentazione dei numeri

- Esterna: BCD, ASCII, Unicode
- Interna: Sistema di numerazione in base 2

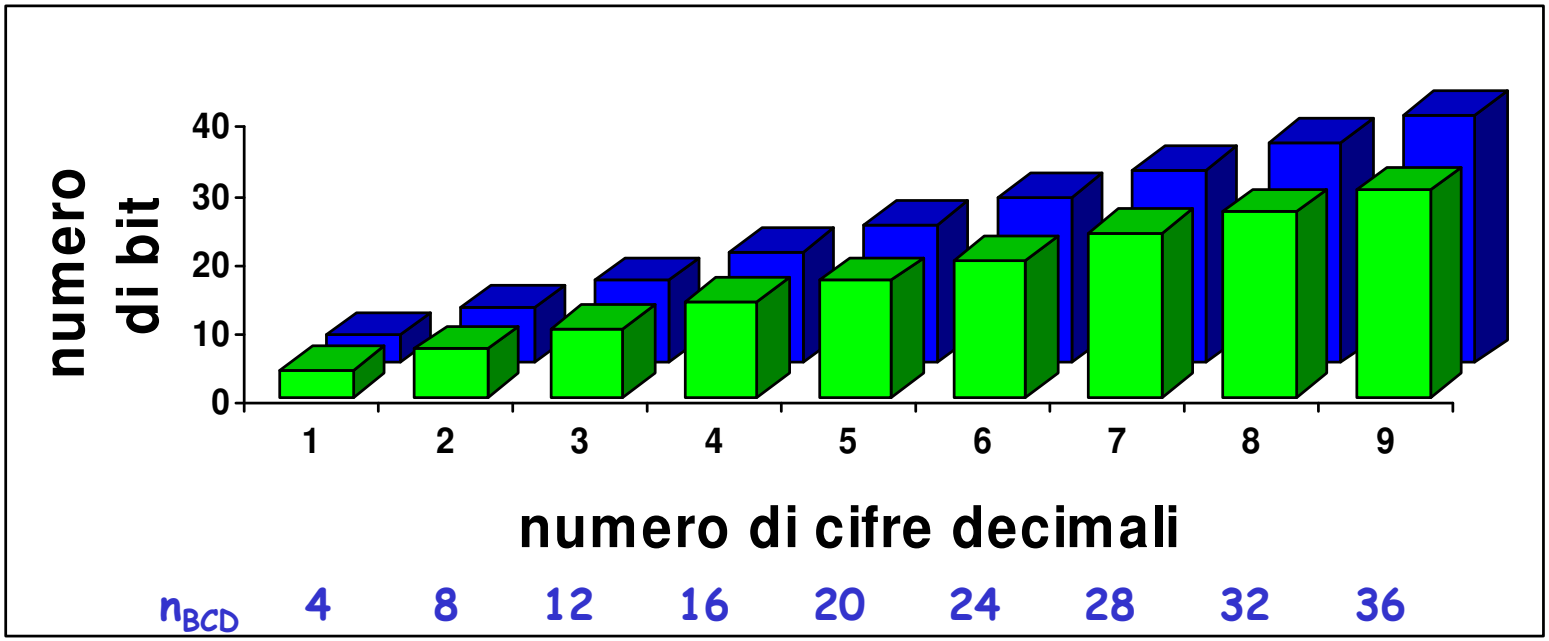
Il numero minimo di bit ($n_{min}$) necessario per rappresentare l'insieme dei $10^k$ (M) numeri interi non negativi di k cifre decimali ($\{0, 1, \ldots, 10^k-1\}$) è:

$$n_{min} = \lceil lg_2 M \rceil = \lceil lg_2 10^k \rceil = \lceil k * lg_2 10 \rceil = \lceil k * 3{,}32 \rceil$$

Il codice BCD, ancorché irridondante dal punto di vista della rappresentazione delle singole cifre decimali, comporta un numero di bit decisamente maggiore:

$$n_{BCD} = k * \lceil lg_2 10 \rceil = k * \lceil 3{,}32 \rceil = k * 4 > n_{min},\ k = 2, 3, \ldots .$$

La ridondanza è tanto più significativa quanto più elevato è il valore di k.




| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $n_{BCD}$ | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 |
| $n_{min}$ | 4 | 7 | 10 | 14 | 17 | 20 | 24 | 27 | 30 |
| $n_{BCD} - n_{min}$ | 0 | 1 | 2 | 2 | 3 | 4 | 4 | 5 | 6 |

# Sistemi di numerazione

Un sistema di numerazione è definito da:

- un **insieme di simboli elementari**;
- un **insieme di regole** che stabiliscono le modalità **di rappresentazione** di grandezze numeriche in termini di simboli elementari;
- un **insieme di regole** che stabiliscono le modalità **di elaborazione** di grandezze numeriche espresse in notazione simbolica.

I sistemi di numerazione si distinguono in:

- **sistemi non posizionali** (sistema di numerazione romano),
- **sistemi posizionali** (sistema di numerazione decimale).

1667 MDCLXVII

# Sistema di numerazione posizionale in base b (b≥2)

1) Rappresentazione:

$(N_b) = (a_{n-1} \dots a_0, a_{-1} \dots a_{-m})_b$ $\quad a_k \in \{0, 1, \dots, b-1\}, \ \forall\ k$

2) Valore:

$$(N_b) = (a_{n-1}\, b^{n-1} + \dots + a_0\, b^0 + a_{-1}\, b^{-1} + \dots + a_{-m}\, b^{-m})_b$$

# I sistemi di numerazione binario, ottale, esadecimale

base=2, {0,1}

base=8, {0,1,2,3,4,5,6,7}

base=16, {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

Conversione di rappresentazione (base)
B → B' (B, B' = 2, 8, 16):

B = 2 → B' = 8, 16

$N_2$ = 101011000100

101-011-000-100 → $N_8$ = 5304

1010-1100-0100 → $N_{16}$ = *AC4*

B = 8, 16 → B' = 2

$N_8$ = 5236 → $N_2$ = 101010011110

$N_{16}$ = E82 → $N_2$ = 111010000010

| $N_{10}$ | $N_2$ | $N_8$ | $N_{16}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 2 | 10 | 2 | 2 |
| 3 | 11 | 3 | 3 |
| 4 | 100 | 4 | 4 |
| 5 | 101 | 5 | 5 |
| 6 | 110 | 6 | 6 |
| 7 | 111 | 7 | 7 |
| 8 | 1000 | 10 | 8 |
| 9 | 1001 | 11 | 9 |
| 10 | 1010 | 12 | A |
| 11 | 1011 | 13 | B |
| 12 | 1100 | 14 | C |
| 13 | 1101 | 15 | D |
| 14 | 1110 | 16 | E |
| 15 | 1111 | 17 | F |
| 16 | 10000 | 20 | 10 |
| … | … | … | … |

# Conversione di base: $B \rightarrow B'$ ($\forall$ B, B' ≥ 2) …

$$(N)_B = (I,F)_B = (a_{n-1} \ldots a_0, a_{-1} \ldots a_{-m})_B \rightarrow (N)_{B'} = (?,?)_{B'}$$

**Metodo di conversione polinomiale:**

$$(N)_{B'} = (a_{n-1} B^{n-1} + \ldots + a_0 B^0 + a_{-1} B^{-1} + \ldots + a_{-m} B^{-m})_{B'}$$

Il metodo di conversione polinomiale si avvale delle regole di rappresentazione e di elaborazione del sistema di numerazione in base B', e pertanto è convenientemente utilizzabile per operare la conversione dalla base B = $\forall$ (di norma 2) alla base B' = 10.

Esempio: B = 2 $\rightarrow$ B' = 10

$$(100110,01)_2 = (0 \cdot 1 + 1 \cdot 2 + 1 \cdot 4 + 0 \cdot 8 + 0 \cdot 16 + 1 \cdot 32 + 0 \cdot 0{,}5 + 1 \cdot 0{,}25)_{10} = (38{,}25)_{10}$$

Esempio: B = 3 $\rightarrow$ B' = 10

$$(122)_3 = (2 \cdot 1 + 2 \cdot 3 + 1 \cdot 9)_{10} = (17)_{10}$$

# … Conversione di base: $B \rightarrow B'$ ($\forall$ B, B' ≥ 2) …

$(N)_B = (I,F)_B = (a_{n-1} \dots a_0, a_{-1} \dots a_{-m})_B \rightarrow (N)_{B'} = (?,?)_{B'}$

## Metodo di conversione iterativa:

$(N)_{B'} = (c_{i-1} \dots c_0, c_{-1} \dots c_{-f})_{B'}$

$$
\begin{aligned}
(I)_B &= (c_{i-1} B'^{i-1} + \dots + c_1 B' + c_0)_B \\
&= ((c_{i-1} B'^{i-2} + \dots + c_1) B' + c_0)_B \\
&= (I' B' + c_0)_B \\
&\rightarrow c_0 = \text{parte frazionaria di } (I/B')_B, \\
&\quad\; c_1 = \text{parte frazionaria di } (I'/B')_B, \\
&\quad\; \dots
\end{aligned}
$$

$$
\begin{aligned}
(F)_B &= (c_{-1} B'^{-1} + c_{-2} B'^{-2} + \dots + c_{-f} B'^{-f})_B \\
&= ((c_{-1} + c_{-2} B'^{-1} + \dots + c_{-f} B'^{-f+1}) B'^{-1})_B \\
&= ((c_{-1} + F') B'^{-1})_B \\
&\rightarrow c_{-1} = \text{parte intera di } (F \cdot B')_B, \\
&\quad\; c_{-2} = \text{parte intera di } (F' \cdot B')_B, \\
&\quad\; \dots
\end{aligned}
$$

# … Conversione di base: $B \rightarrow B'$ ($\forall$ B, B' ≥ 2) …

Il metodo di conversione iterativa si avvale delle regole di rappresentazione e di elaborazione del sistema di numerazione in base B, e pertanto è convenientemente utilizzabile per operare la conversione dalla base B = 10 alla base B' = $\forall$ (di norma 2).

Esempio: B = 10 $\rightarrow$ B' = 2

$(131,75)_{10} = (10000011,11)_2$

| I | : 2 = | I' | + | $c_{k\ (k = 0, 1, \ldots)}$ |
|---|---|---|---|---|
| 131 | | 65 | | 1 |
| 65 | | 32 | | 1 |
| 32 | | 16 | | 0 |
| 16 | | 8 | | 0 |
| 8 | | 4 | | 0 |
| 4 | | 2 | | 0 |
| 2 | | 1 | | 0 |
| 1 | | 0 | | 1 |

| F | · 2 = | $c_{-k\ (k=1,2,\ldots)}$ | + | F' |
|---|---|---|---|---|
| 0,75 | | 1 | | 0,5 |
| 0,5 | | 1 | | 0 |

# … Conversione di base: $B \rightarrow B'$ ($\forall$ B, B' ≥ 2)

Al contrario di quanto avviene per la parte intera, il procedimento di conversione della parte frazionaria può non terminare in un numero finito di iterazioni. In tal caso la conversione è da intendersi completata al raggiungimento della precisione desiderata, con un eventuale arrotondamento dell'ultima cifra significativa.

Esempio: B = 10 → B' = 2

$(\ldots,8)_{10} = (\ldots\ldots,11001101)_2$

| F | · 2 = | $c_{-k\ (k=1,2,\ldots)}$ | + | F' |
|---|---|---|---|---|
| 0,8 | | 1 | | 0,6 |
| 0,6 | | 1 | | 0,2 |
| 0,2 | | 0 | | 0,4 |
| 0,4 | | 0 | | 0,8 |
| 0,8 | | … | | … |

B, B' ≠ 10: B → 10 → B'

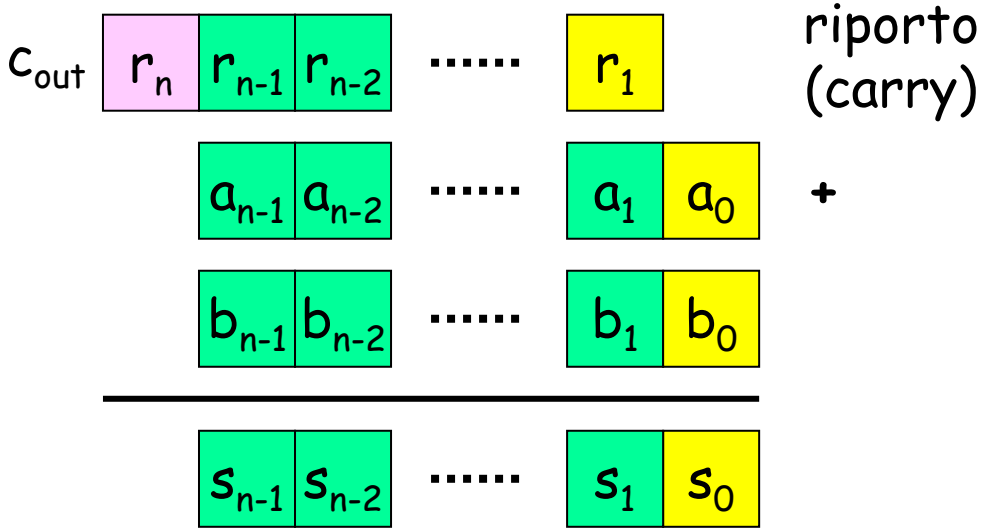
Esempio: B = 3 → B' = 16

$(122)_3 = (17)_{10} = (11)_{16}$

# Operazioni aritmetiche

# Addizione ...

$S = A + B$ $A, B, S$: n-bit unsigned integer $0 \le A, B, S \le 2^n - 1$




| $r_i$ | $a_i$ | $b_i$ | $r_{i+1}$ | $s_i$ |
|---|---|---|---|---|
| 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 |

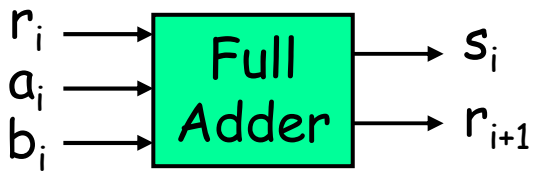
$C_{out} = 0$ ↓ $S \le 2^4 - 1$

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | | |
| | 0 | 1 | 0 | 1 | A | $(5)_{10}$ |
| | 1 | 0 | 0 | 1 | B | $(9)_{10}$ |
| | 1 | 1 | 1 | 0 | S | $(14)_{10}$ |

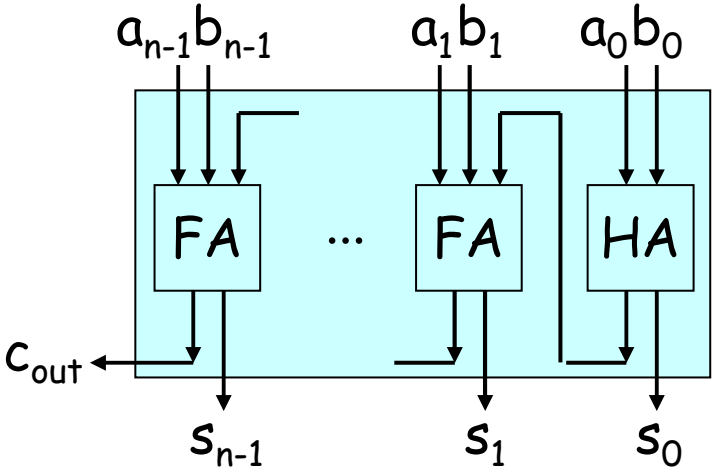
$C_{out} = 1$ ↓ $S > 2^4 - 1$

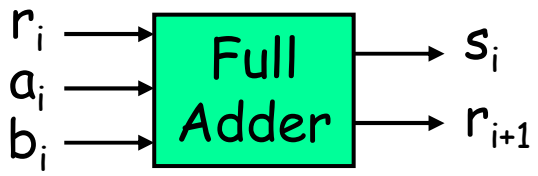
| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | | |
| | 1 | 0 | 0 | 0 | A | $(8)_{10}$ |
| | 1 | 0 | 0 | 1 | B | $(9)_{10}$ |
| | 0 | 0 | 0 | 1 | S | $(17)_{10}$ |

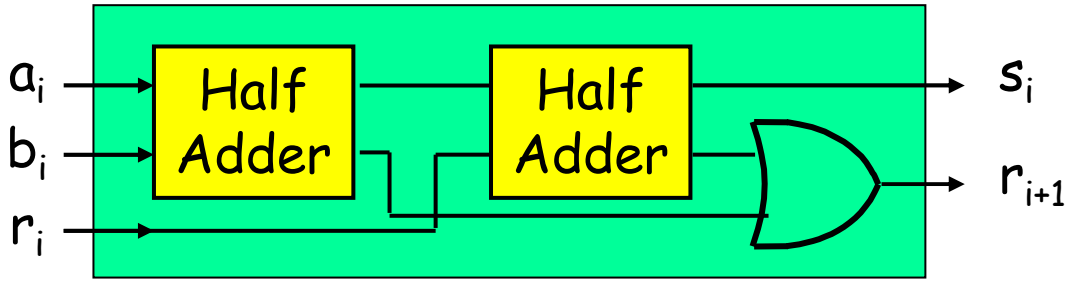
**Esempi n=4**

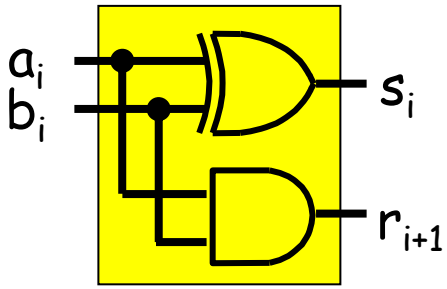
# … Addizione …

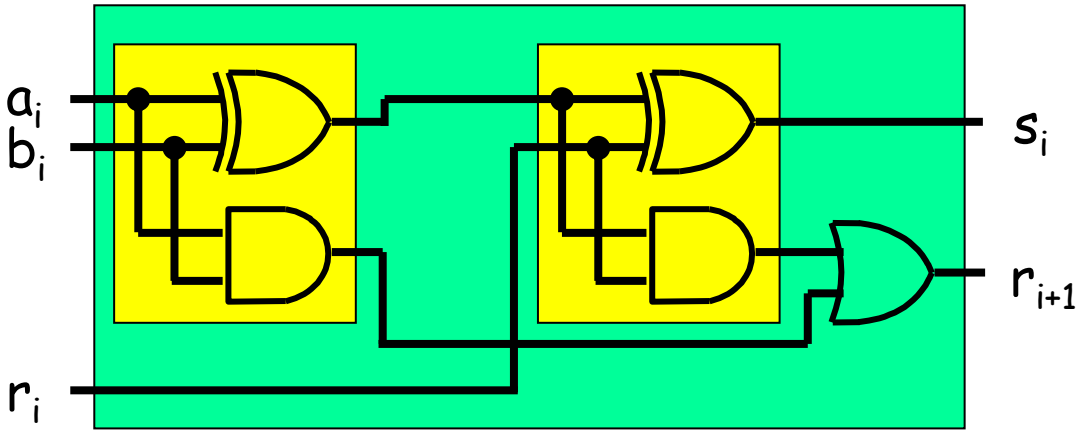

$a_{n-1}b_{n-1}$
$a_1b_1$
$a_0b_0$
FA
…
FA
HA
$c_{out}$
$s_{n-1}$
$s_1$
$s_0$


$r_i$
$a_i$
$b_i$
Full Adder
$s_i$
$r_{i+1}$

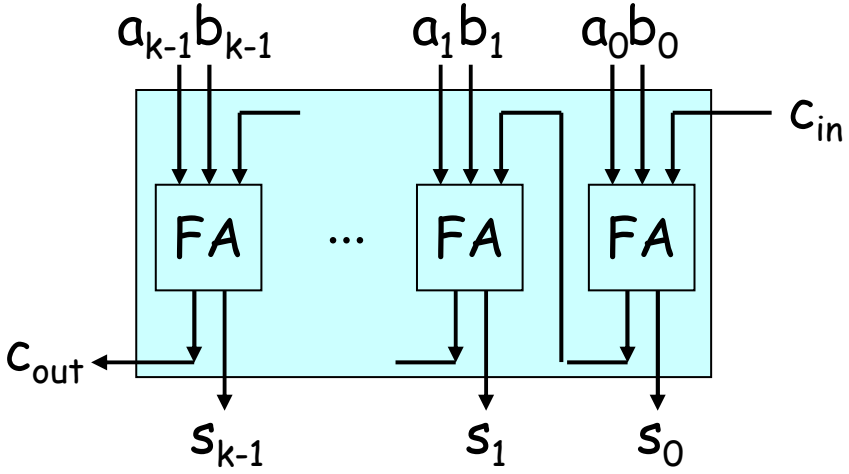

$a_i$
$b_i$
$r_i$
Half Adder
Half Adder
$s_i$
$r_{i+1}$

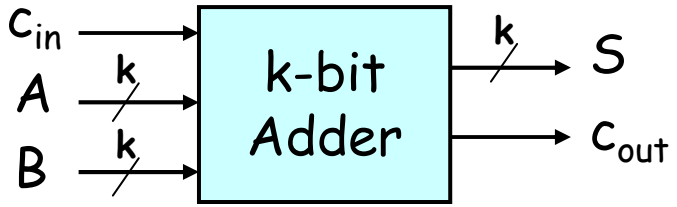

$a_i$
$b_i$
Half Adder
$s_i$
$r_{i+1}$


$a_i$
$b_i$
$s_i$
$r_{i+1}$


$a_i$
$b_i$
$r_i$
$s_i$
$r_{i+1}$

# … Addizione




**E se k < n ?**

esempio:
k = 8, n = 16

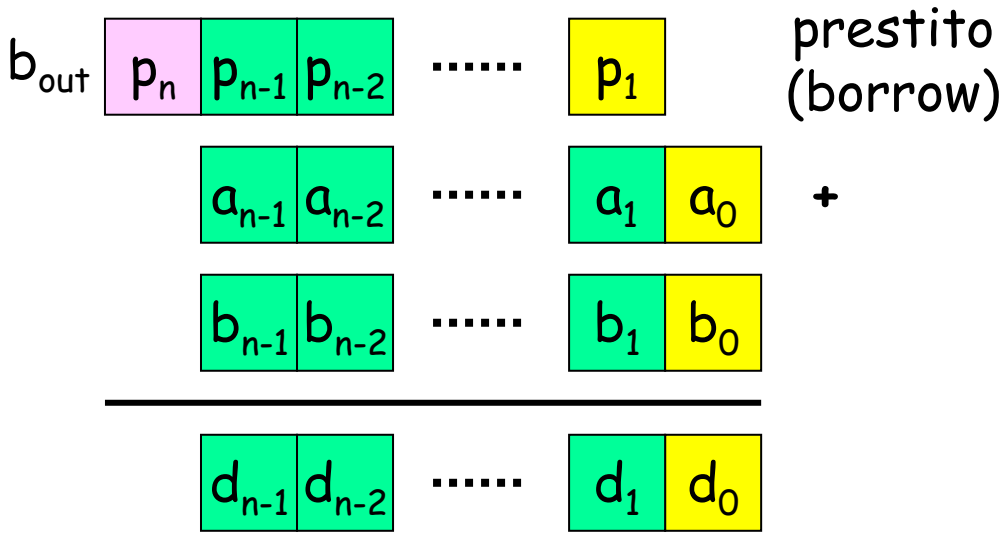
$R = OP_1 + OP_2$

$OP_1$, $OP_2$, R:
16-bit UINT

```
CLC                          /* clear carry-out                    */
LOAD   A, OP1 LSB            /* first load operands                */
LOAD   B, OP2 LSB            /* least significant bytes,           */
ADC                          /* perform addition (cin=cout)        */
STORE  RLSB, S               /* and save the result;               */
LOAD   A, OP1 MSB            /* then load operands                 */
LOAD   B, OP2 MSB            /* most significant bytes,            */
ADC                          /* perform addition (cin=cout)        */
STORE  RMSB, S               /* and save the result;               */
JC     Error                 /* if carry-out then …                */
```

# Sottrazione ...

$D = A - B$ A, B, D: n-bit unsigned integer $0 \le A, B, D \le 2^n-1$




| $p_i$ | $a_i$ | $b_i$ | $p_{i+1}$ | $d_i$ |
|---|---|---|---|---|
| 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 |

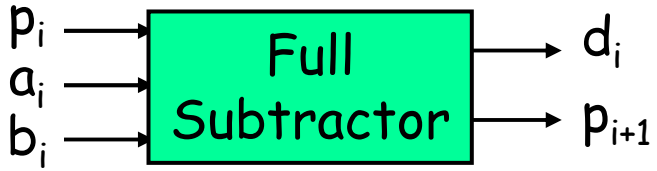
$b_{out} = 0$ ↓ $D \ge 0$

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | | |
| 1 | 0 | 0 | 1 | A | $(9)_{10}$ |
| 0 | 1 | 0 | 1 | B | $(5)_{10}$ |
| 0 | 1 | 0 | 0 | D | $(4)_{10}$ |

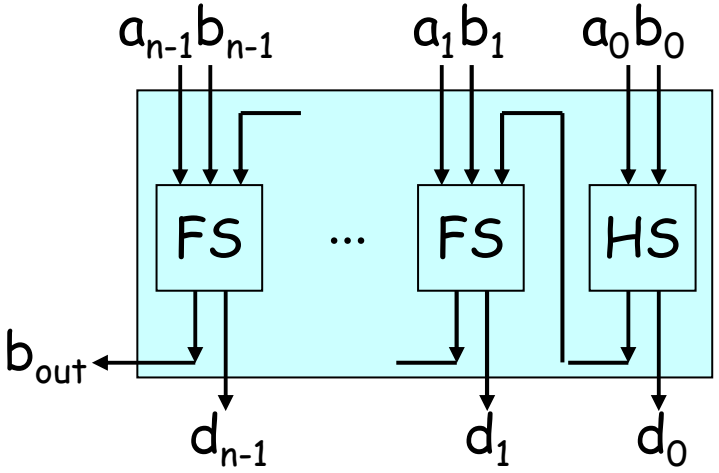
$b_{out} = 1$ ↓ $D < 0$

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | | |
| 1 | 0 | 0 | 0 | A | $(8)_{10}$ |
| 1 | 0 | 0 | 1 | B | $(9)_{10}$ |
| 1 | 1 | 1 | 1 | D | ??? |

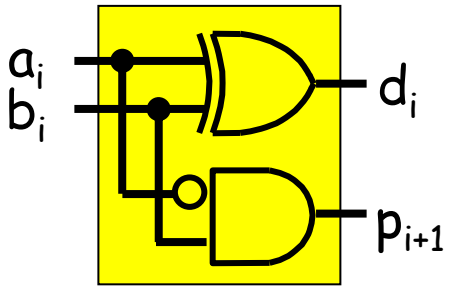
**Esempi n=4**

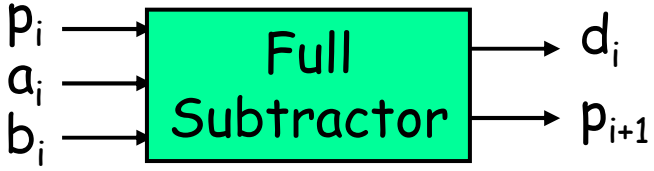



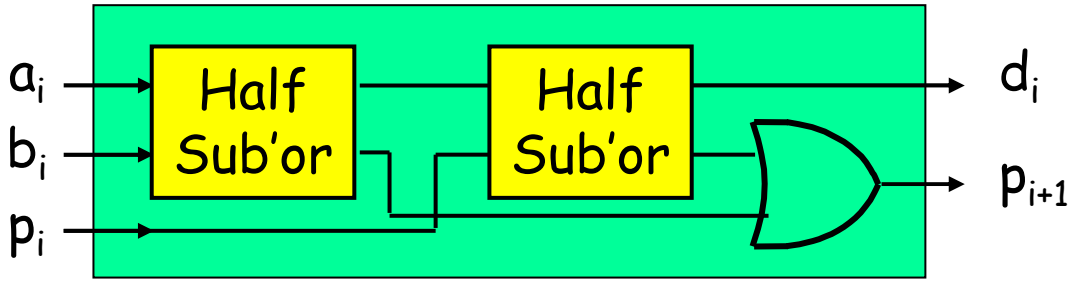
# … Sottrazione …

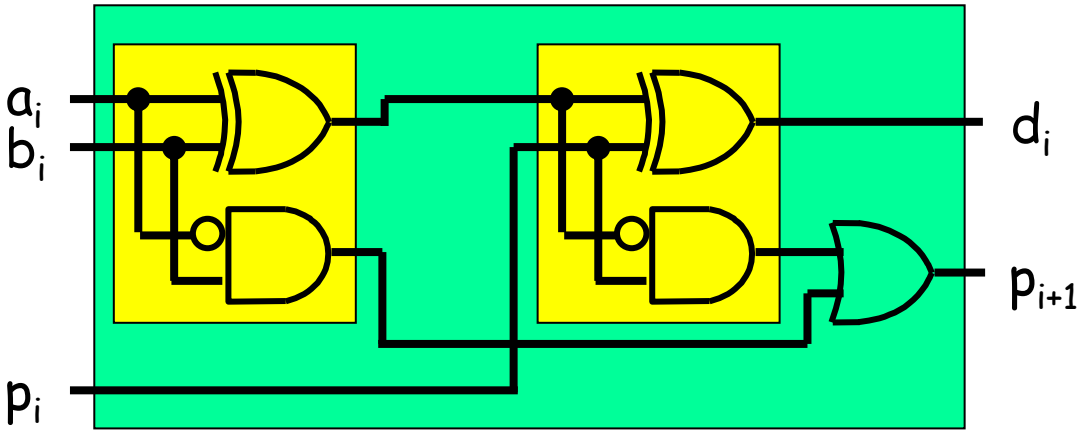




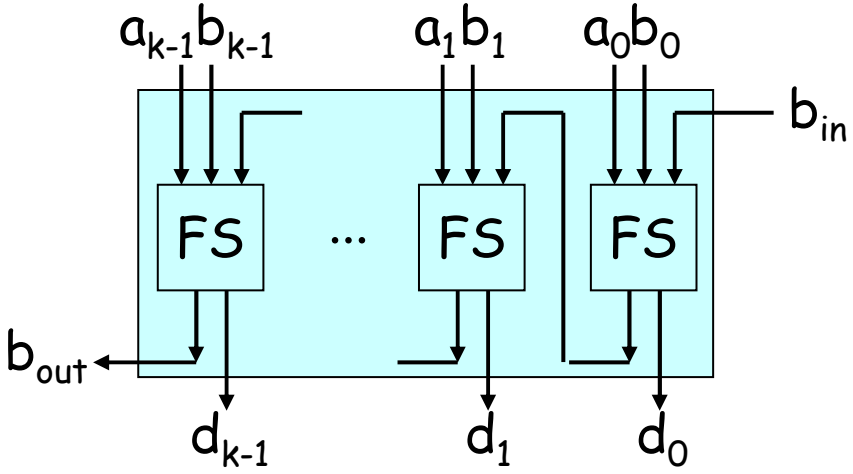


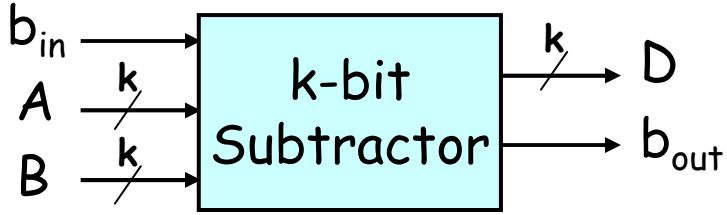



# … Sottrazione




**E se k < n ?**

esempio:
k = 8, n = 16

R = $OP_1$ - $OP_2$

$OP_1$, $OP_2$, R:
16-bit UINT

```
CLB                  /* clear borrow-out                          */
LOAD  A, OP1 LSB     /* first load operands                       */
LOAD  B, OP2 LSB     /* least significant bytes,                  */
SBB                  /* perform subtraction (bin=bout)            */
STORE RLSB, D        /* and save the result;                      */
LOAD  A, OP1 MSB     /* then load operands                        */
LOAD  B, OP2 MSB     /* most significant bytes,                   */
SBB                  /* perform subtraction (bin=bout)            */
STORE RMSB, D        /* and save the result;                      */
JB    Error          /* if borrow-out then …                      */
```

# Rappresentazione dei numeri relativi

$A = a_{n-1}\, a_{n-2} \ldots a_1\, a_0$ n-bit signed integer

## Segno e valore assoluto

$a_{n-1}$ (0:+, 1:-) $\quad a_{n-2} \ldots a_1\, a_0$ $\quad -(2^{n-1}-1) \le A \le 2^{n-1}-1$

**n = 4**

$(+5)_{10} \rightarrow A = 0101$

$(-5)_{10} \rightarrow A = 1101$

## Complemento a 2

$^2(-A) = 2^n - {}^2(A) = \text{not } {}^2(A) + 1 \quad -2^{n-1} \le A \le 2^{n-1}-1$

**n = 4**

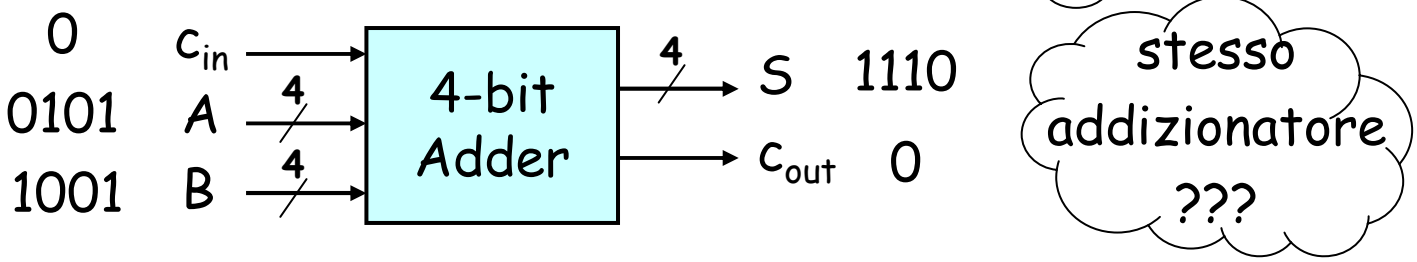
$(+5)_{10} \rightarrow A = 0101$

$(-5)_{10} \rightarrow A = 1010 + 1 = 1011$

**n = 4**

| | $a_3$ | $a_2$ | $a_1$ | $a_0$ | |
|---|---|---|---|---|---|
| +7 | 0 | 1 | 1 | 1 | +7 |
| +6 | 0 | 1 | 1 | 0 | +6 |
| +5 | 0 | 1 | 0 | 1 | +5 |
| +4 | 0 | 1 | 0 | 0 | +4 |
| +3 | 0 | 0 | 1 | 1 | +3 |
| +2 | 0 | 0 | 1 | 0 | +2 |
| +1 | 0 | 0 | 0 | 1 | +1 |
| +0 | 0 | 0 | 0 | 0 | +0 |
| -7 | 1 | 1 | 1 | 1 | -1 |
| -6 | 1 | 1 | 1 | 0 | -2 |
| -5 | 1 | 1 | 0 | 1 | -3 |
| -4 | 1 | 1 | 0 | 0 | -4 |
| -3 | 1 | 0 | 1 | 1 | -5 |
| -2 | 1 | 0 | 1 | 0 | -6 |
| -1 | 1 | 0 | 0 | 1 | -7 |
| -0 | 1 | 0 | 0 | 0 | -8 |

# Addizione di numeri relativi




A, B: 4-bit unsigned integers

$A = (5)_{10}$ $B = (9)_{10}$ $S = (14)_{10} = A + B$ **OK**

A, B: 4-bit signed integers

**Complemento a 2**

$A = (+5)_{10}$ $B = (-7)_{10}$ $S = (-2)_{10} = A + B$ **OK**

**Segno e valore assoluto**

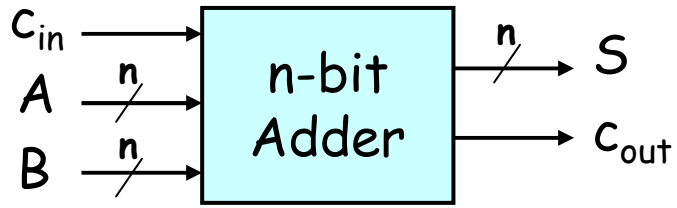
$A = (+5)_{10}$ $B = (-1)_{10}$ $S = (-6)_{10} \neq A + B = (+4)_{10}$ **NO**

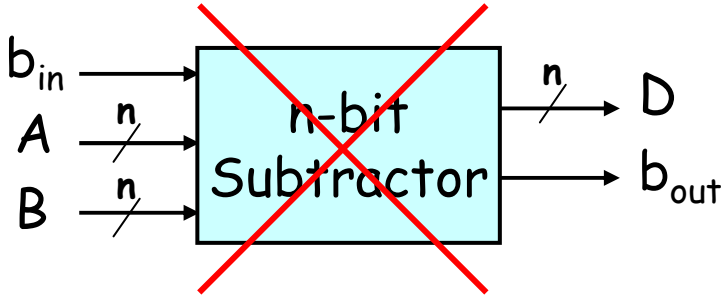
# Sottrazione di numeri relativi

A, B: n-bit 2'complement signed integers

L'operazione di sottrazione è riconducibile ad un'operazione di addizione, previa complementazione del sottraendo:

$$A - B = A + (-B)$$



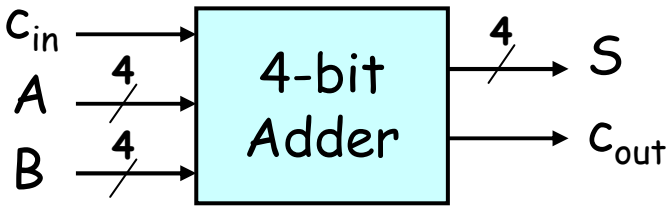

n = 4

$A = (+5)_{10}$ $B = (+2)_{10}$ $A - B = (+5 + (-2))_{10} = (+3)_{10}$

0101 0010 0101 + (-0010) = 0101 + 1101 + 1 = 1 0011

??? OK

# Carry-out & Overflow




$c_{out}$ evidenzia correttamente
la non rappresentabilità
del risultato mediante n bit
solo nel caso di unsigned integers

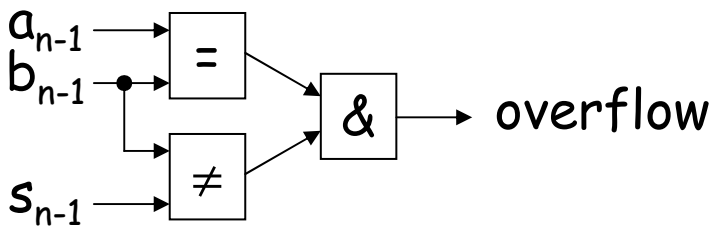
L'analoga indicazione
nel caso di signed integers
è derivabile dal confronto
del segno degli operandi
e del risultato:

| $s_{n-1}$ \ $a_{n-1}$ $b_{n-1}$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| 0 | OK | OK | OK | NO |
| 1 | NO | OK | OK | OK |

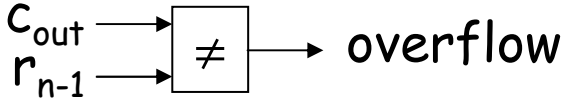
A, B, S: 4-bit
unsigned / signed integers

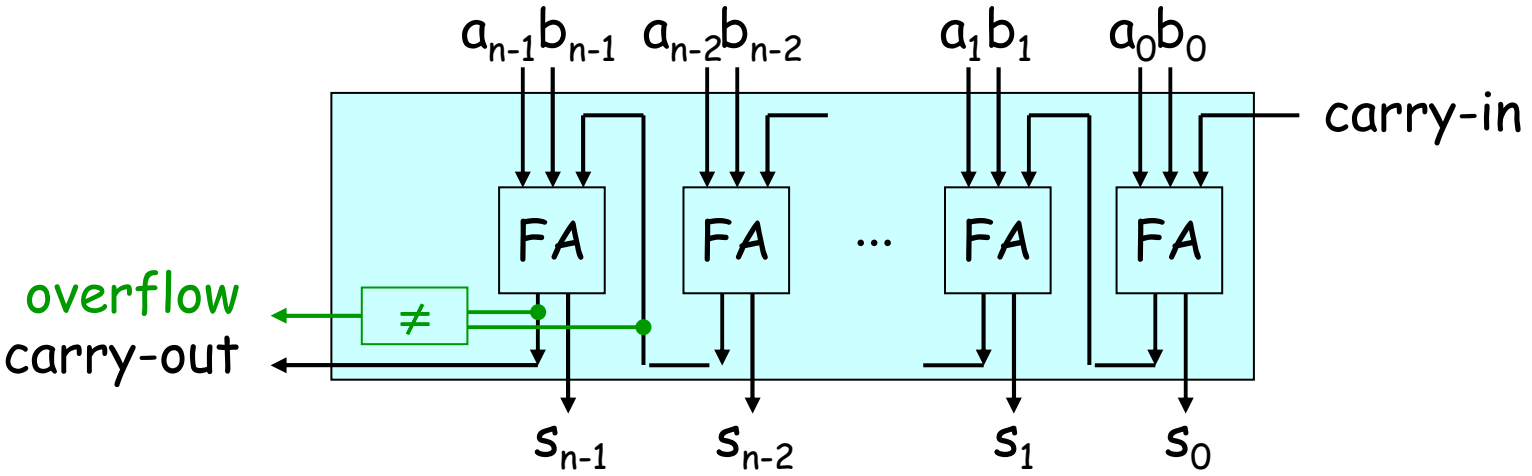
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| OK | OK | 0 | 0 | 0 | 0 | | | |
| $(6)_{10}$ | $(+6)_{10}$ | | 0 | 1 | 1 | 0 | A | |
| $(1)_{10}$ | $(+1)_{10}$ | | 0 | 0 | 0 | 1 | B | |
| $(7)_{10}$ | $(+7)_{10}$ | | 0 | 1 | 1 | 1 | S | |
| OK | NO | 0 | 1 | 1 | 1 | | | |
| $(7)_{10}$ | $(+7)_{10}$ | | 0 | 1 | 1 | 1 | A | |
| $(1)_{10}$ | $(+1)_{10}$ | | 0 | 0 | 0 | 1 | B | |
| $(8)_{10}$ | $(-8)_{10}$ | | 1 | 0 | 0 | 0 | S | |
| OK | OK | 1 | 0 | 0 | 1 | | | |
| $(9)_{10}$ | $(-7)_{10}$ | | 1 | 0 | 0 | 1 | A | |
| $(9)_{10}$ | $(-7)_{10}$ | | 1 | 0 | 0 | 1 | B | |
| $(2)_{10}$ | $(+2)_{10}$ | | 0 | 0 | 1 | 0 | S | |
| OK | NO | 1 | 1 | 1 | 1 | | | |
| $(15)_{10}$ | $(-1)_{10}$ | | 1 | 1 | 1 | 1 | A | |
| $(1)_{10}$ | $(+1)_{10}$ | | 0 | 0 | 0 | 1 | B | |
| $(0)_{10}$ | $(+0)_{10}$ | | 0 | 0 | 0 | 0 | S | |

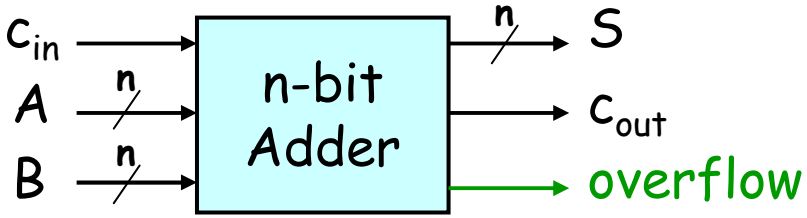
# Signed & Unsigned Integers: +/-




o più semplicemente



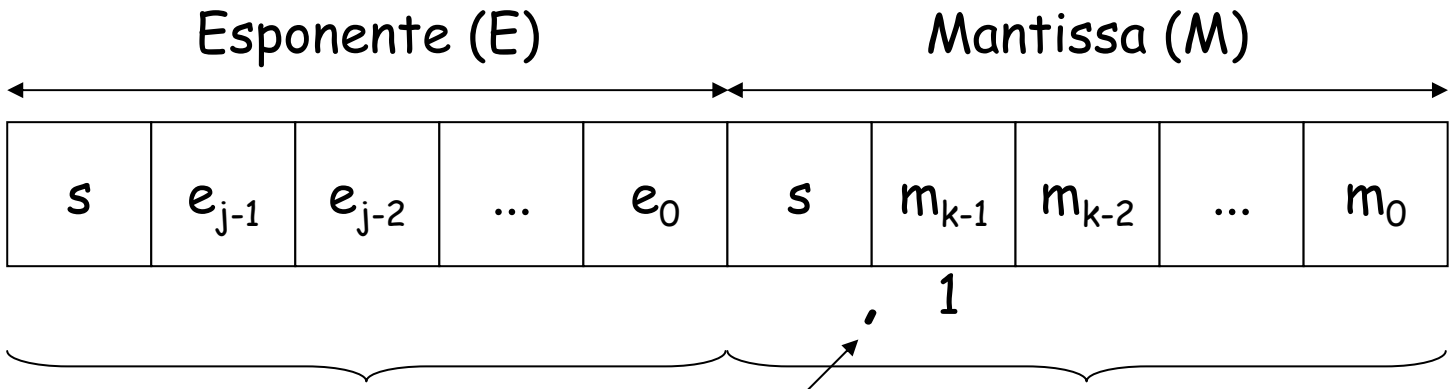




unsigned integers

signed integers

# Rappresentazione dei numeri razionali

Notazione scientifica: $N_2 = (M \cdot 2^E)_2$




Es.:
32 bit

7-bit
(complemento a 2)

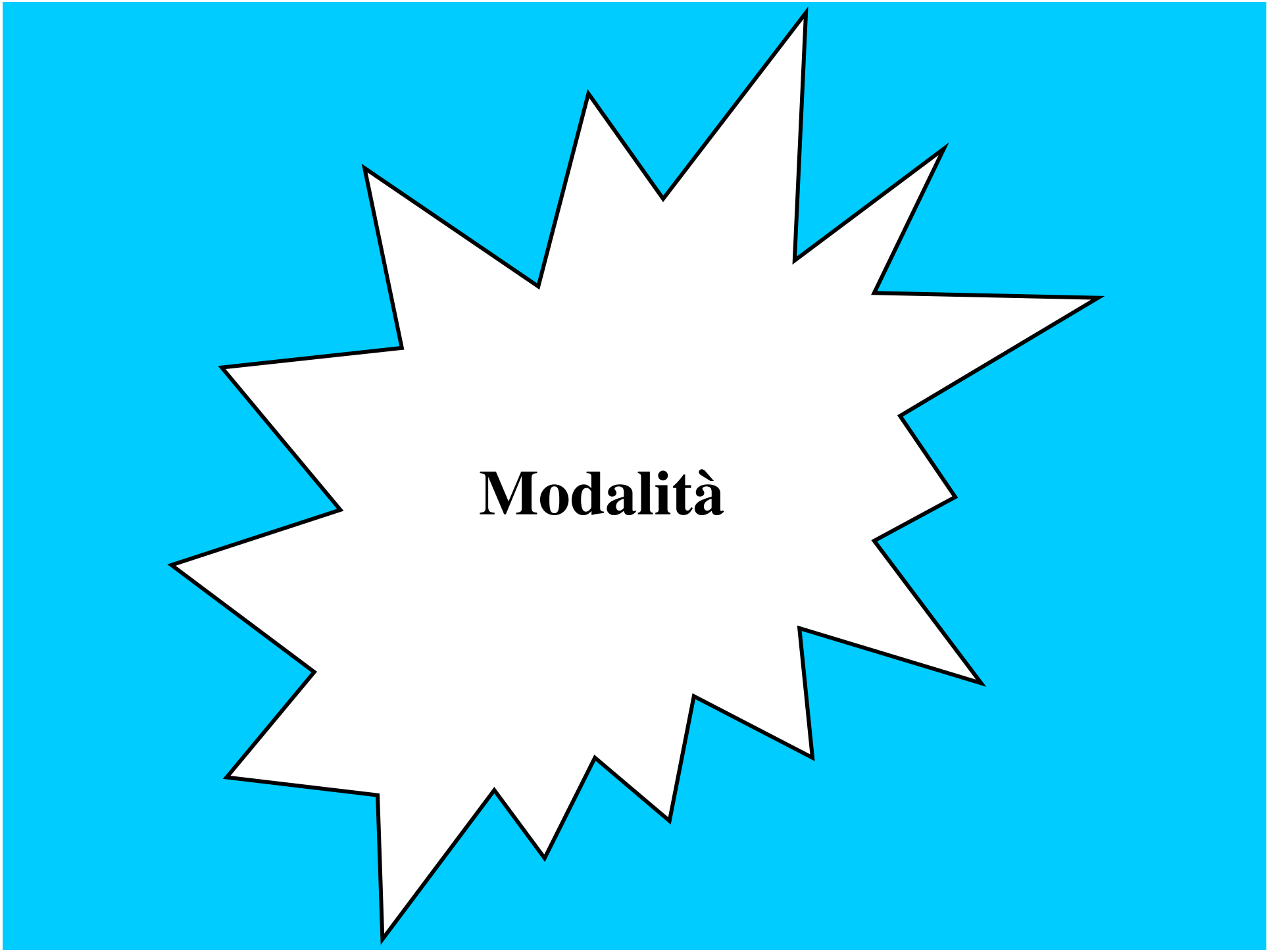
25-bit
(segno e valore assoluto)

notazione frazionaria normalizzata:
$0{,}5 \leq M < 1$ (tranne lo zero (32 "0"))

$E_{min} = -2^6 = -64$
$E_{max} = 2^6 - 1 = 63$
$M_{min} = 0{,}5$
$M_{max} = 1 - 2^{-24}$

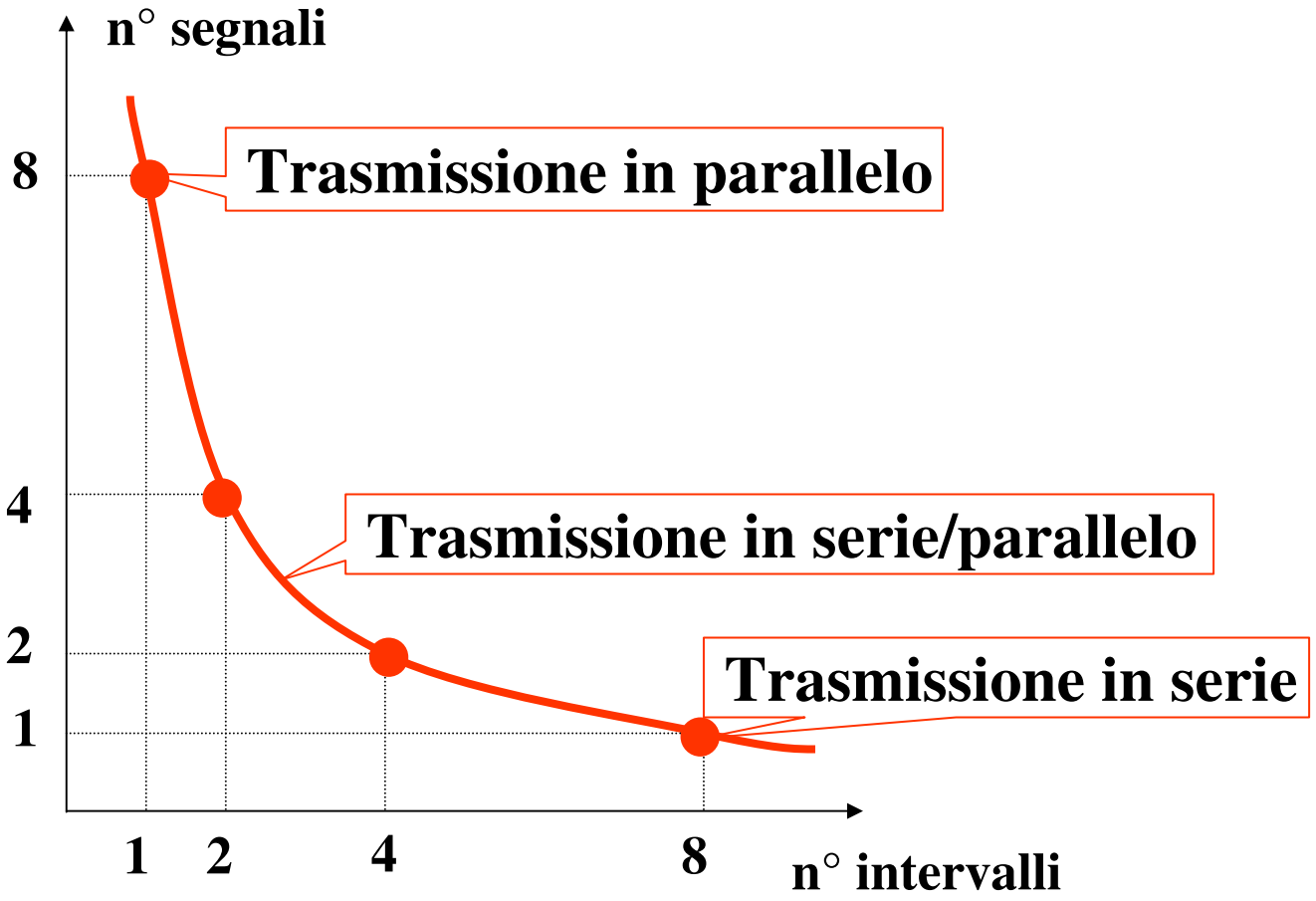
$\pm (0{,}5 \cdot 2^{-64} \div (1 - 2^{-24}) \cdot 2^{63}) \cong \pm (2{,}7 \cdot 10^{-20} \div 0{,}9 \cdot 10^{19})$

# 2.4
# Trasmissione


Modalità

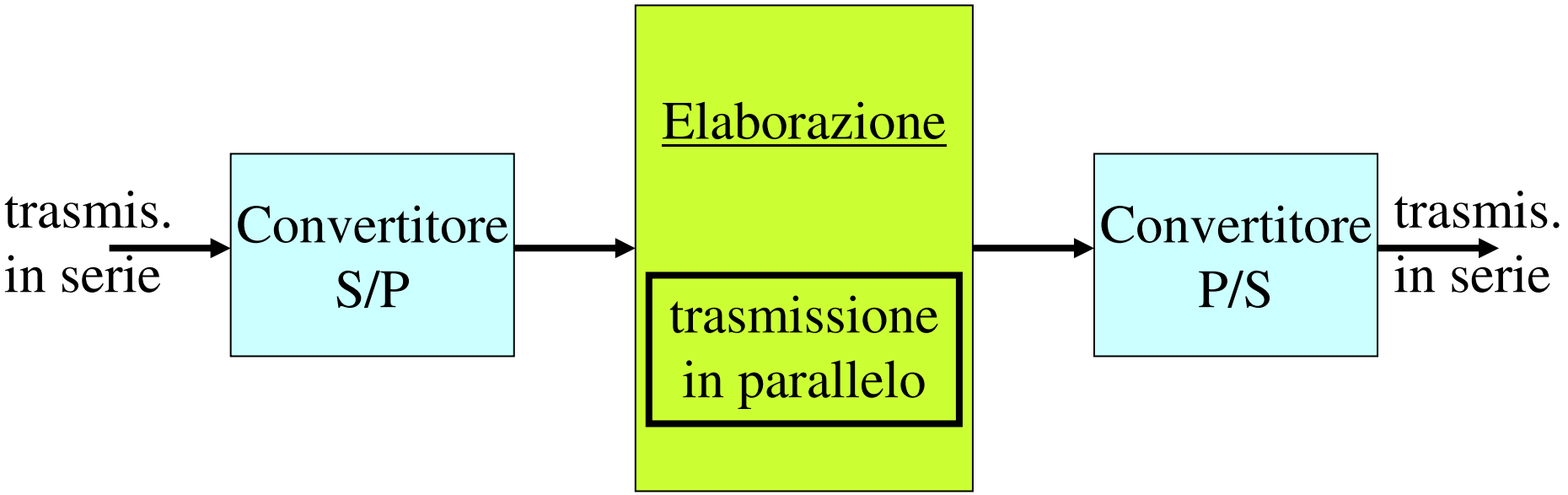
# Modalità di trasmissione dei bit: compromesso spazio/tempo

Es.: Codice a 8 bit




Esempio: processori Intel

# Modalità di trasmissione dei bit: convertitori S/P e P/S

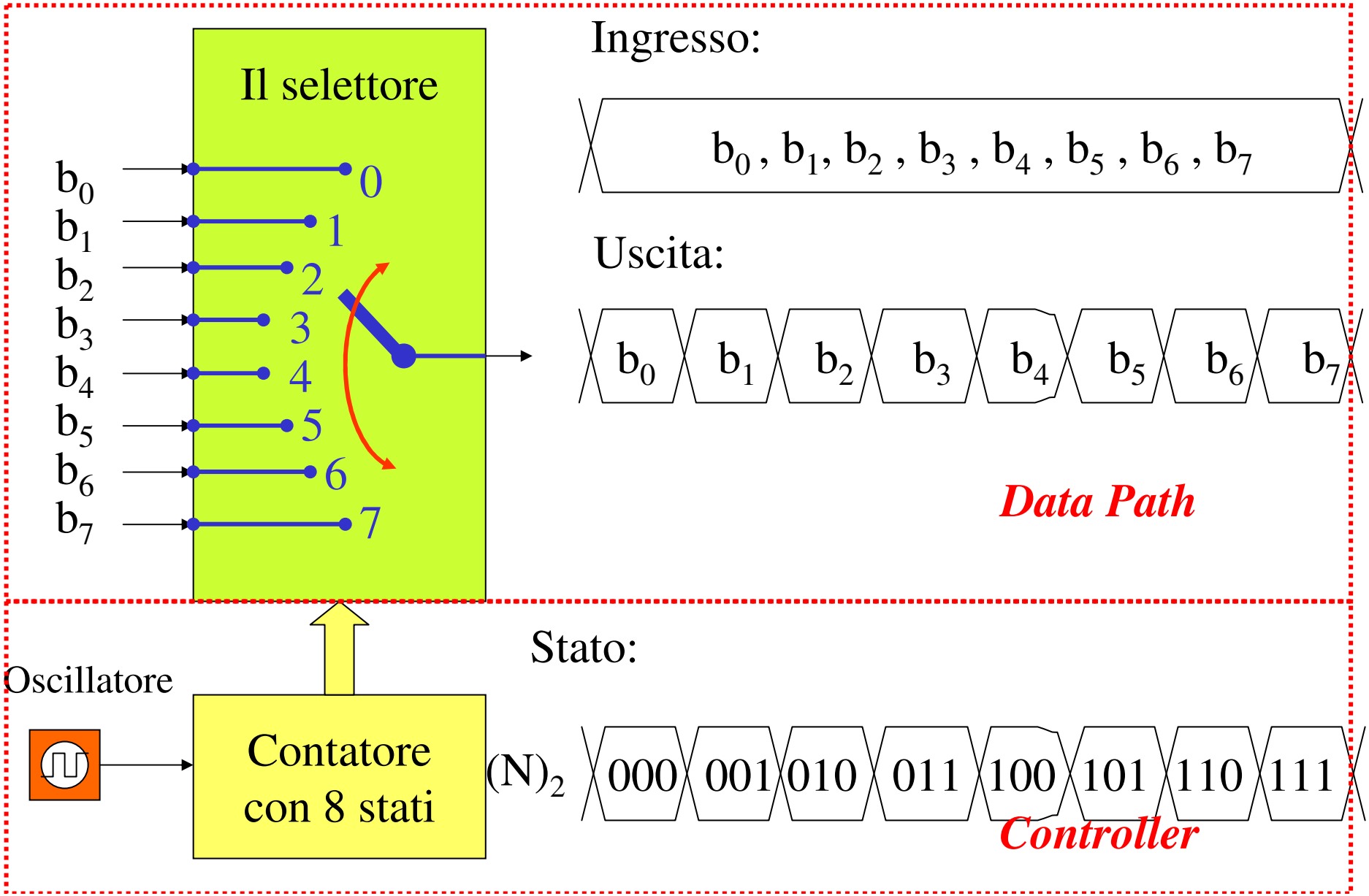



**La <u>modalità di trasmissione all'interno</u> della macchina è di norma in parallelo (per massimizzare la velocità di elaborazione)**

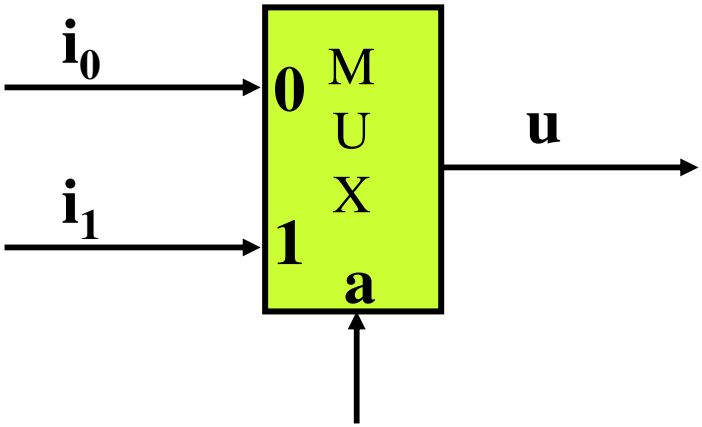
**La <u>modalità di trasmissione all'esterno</u> della macchina è di norma in serie (per minimizzare la complessità del supporto fisico)**

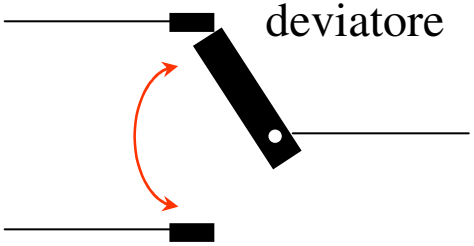
Esempi: interfaccia di tastiera, interfaccia video

# La conversione P/S di un byte



# La serializzazione di due bit



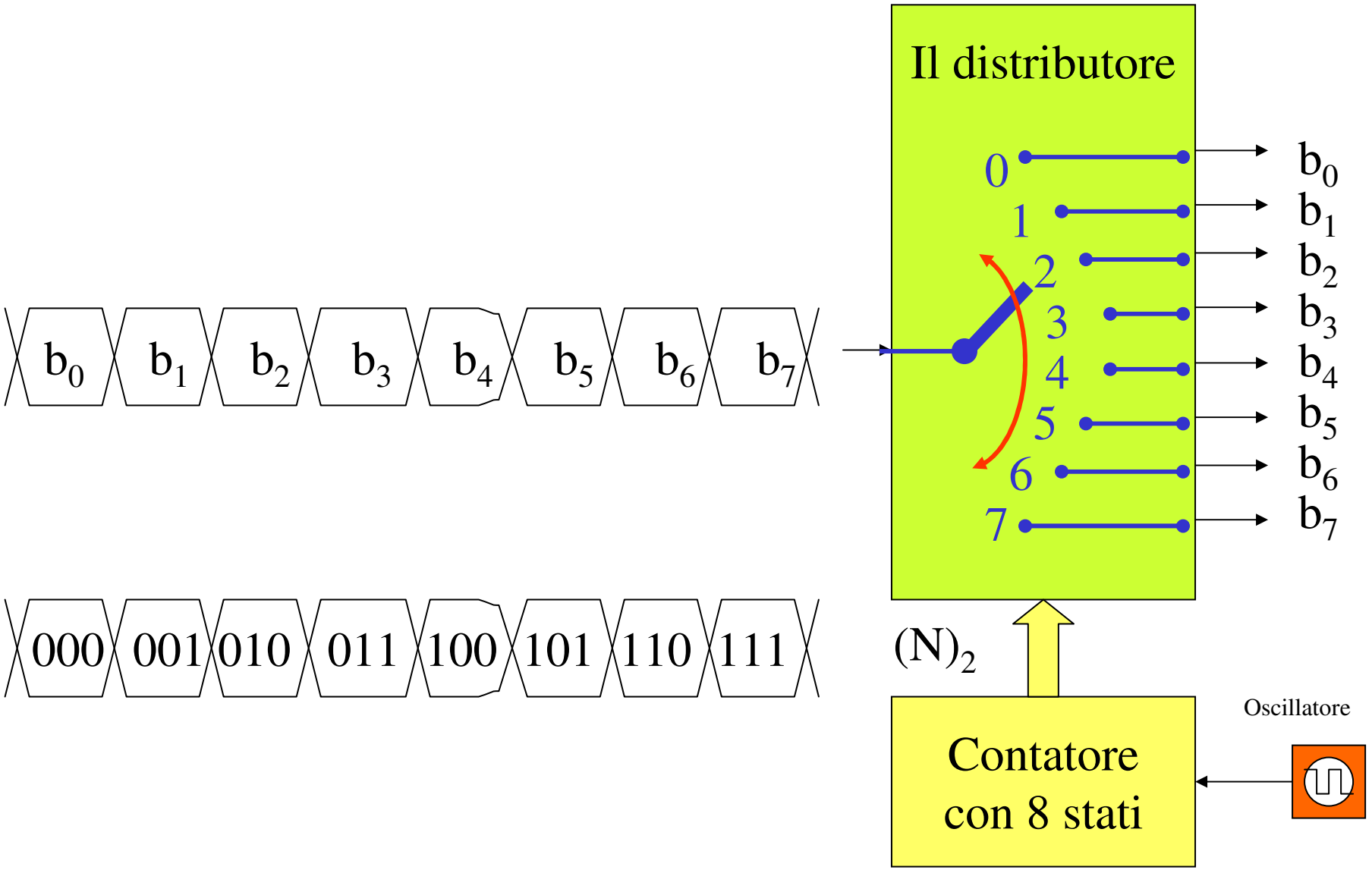




| a | $i_0$ | $i_1$ | u |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

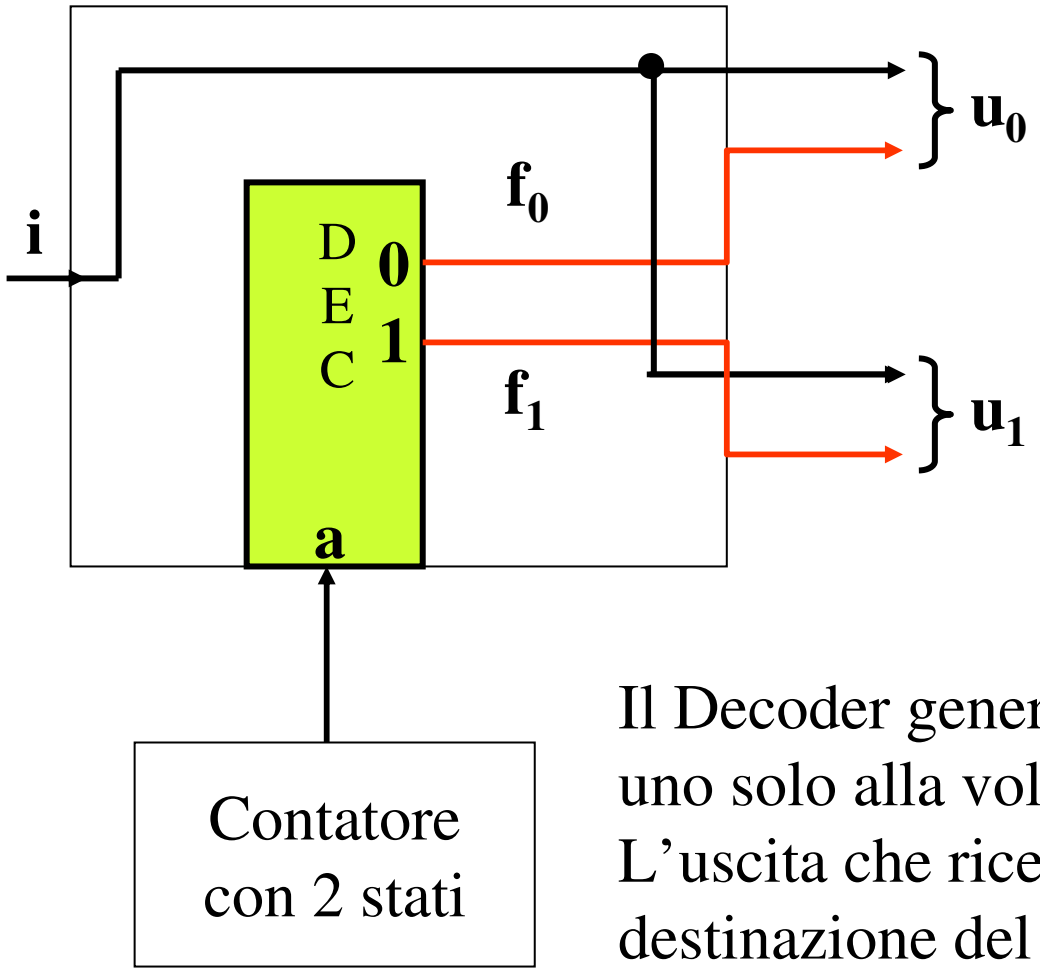
***se*** **a=0** ***allora*** **$u=i_0$**

***altrimenti*** **$u=i_1$**

# Conversione S/P di un byte

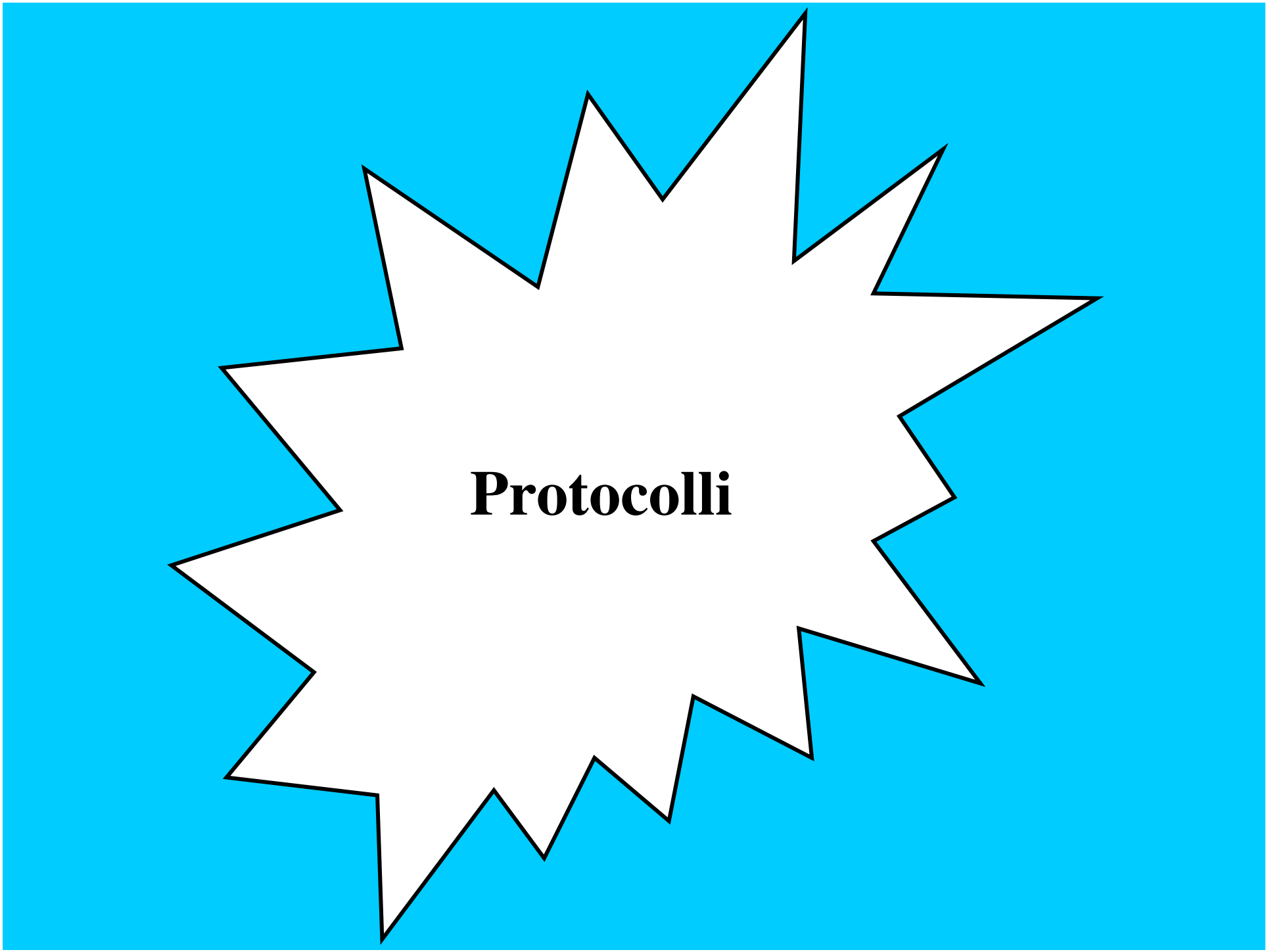


# La distribuzione di due bit

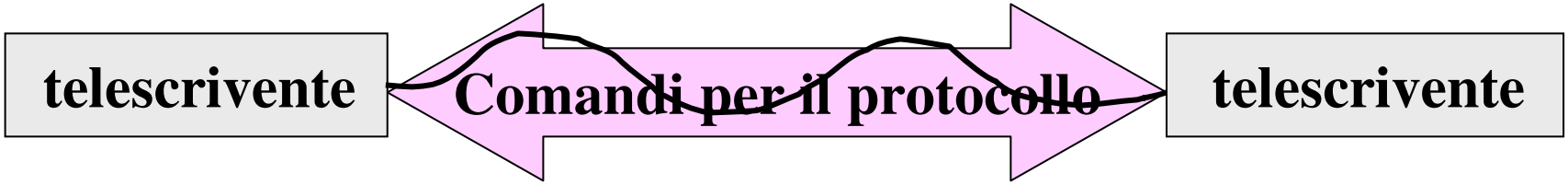



| a | $f_0$ | $f_1$ |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 0 | 1 |

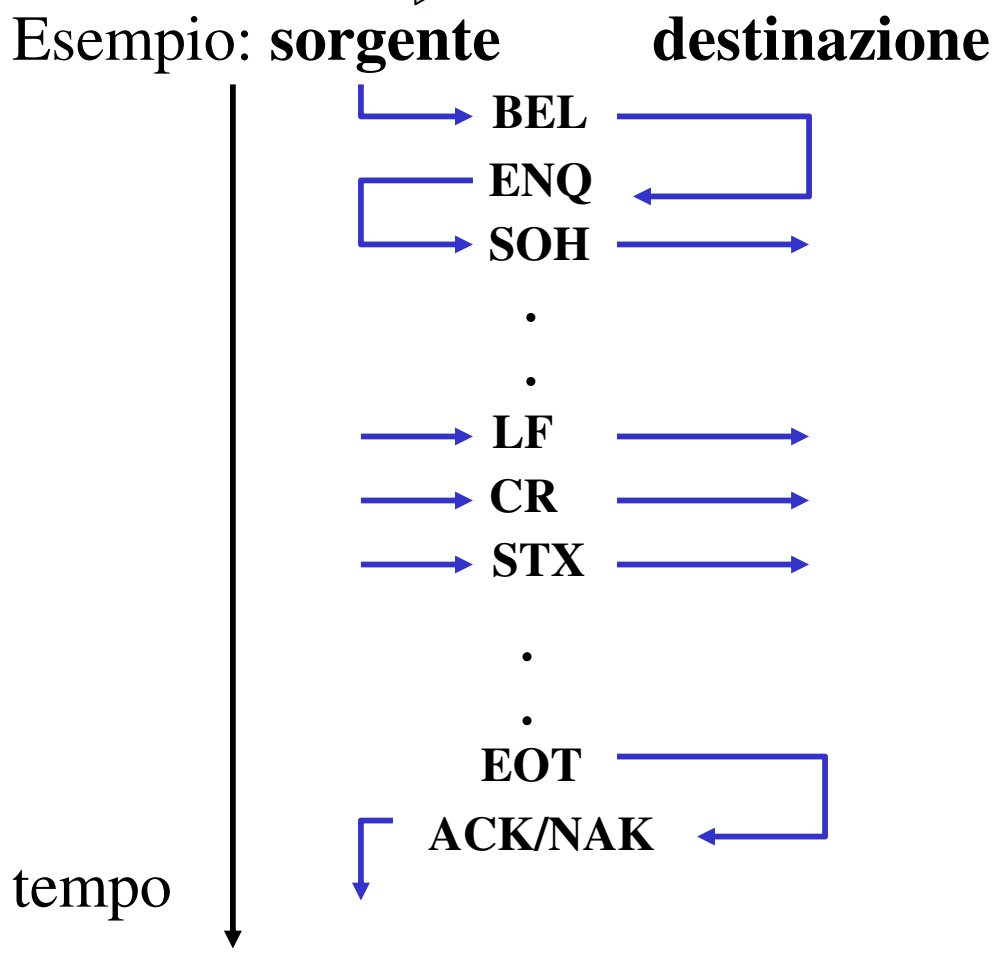
Il Decoder genera **2** "flag di validità", di cui uno solo alla volta ha valore 1.
L'uscita che riceve tale valore è la destinazione del bit d'ingresso **i**


Protocolli

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



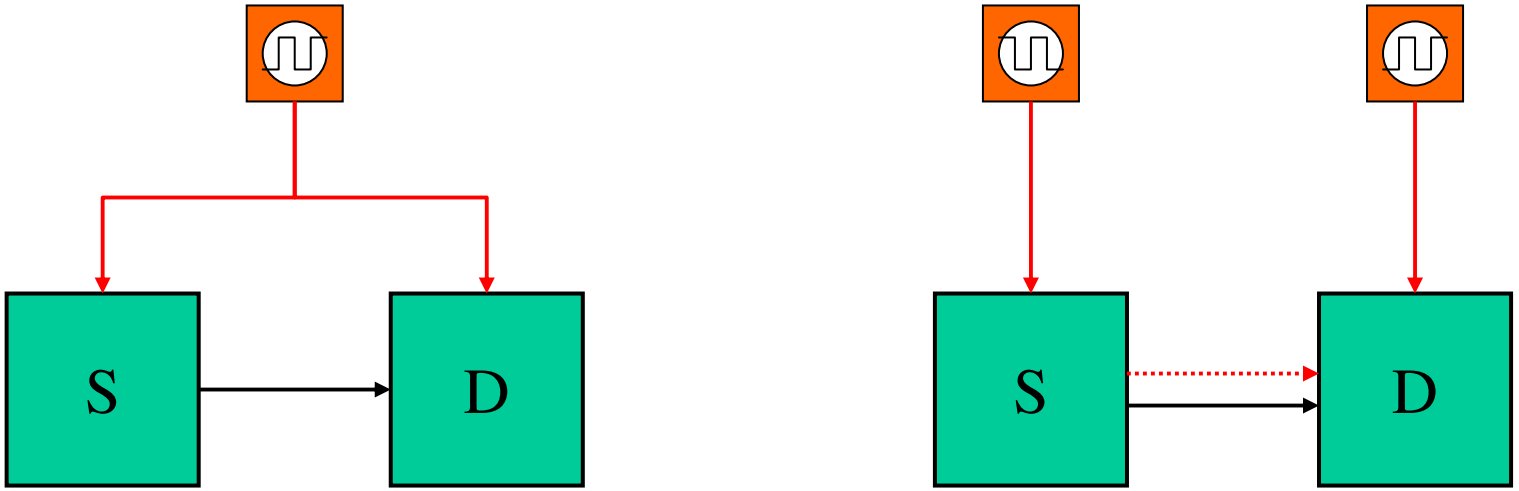

| | 000 | 001 |
|---|---|---|
| 0000 | NUL | DLE |
| 0001 | SOH | DC1 |
| 0010 | STX | DC2 |
| 0011 | ETX | DC3 |
| 0100 | EOT | DC4 |
| 0101 | ENQ | NAK |
| 0110 | ACK | SYN |
| 0111 | BEL | ETB |
| 1000 | BS | CAN |
| 1001 | HT | EM |
| 1010 | LF | SUB |
| 1011 | VT | ESC |
| 1100 | FF | FS |
| 1101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



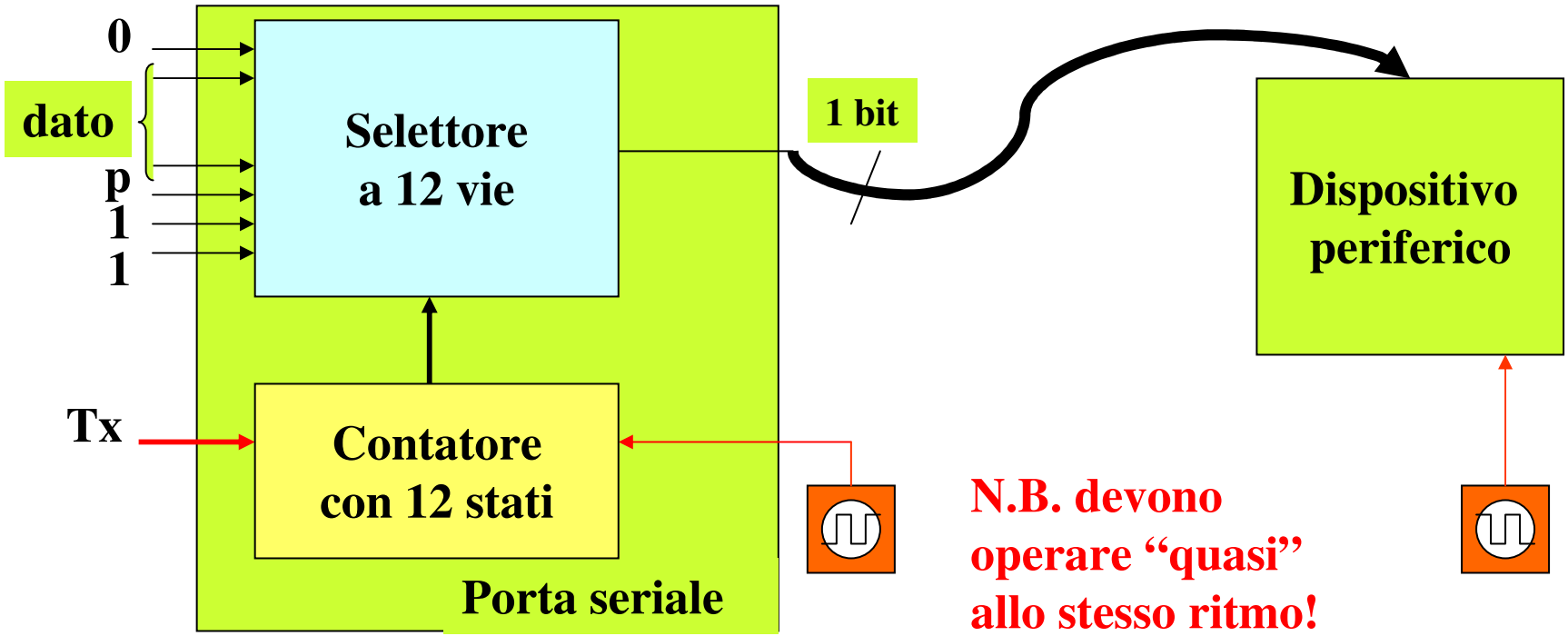
# Sincronizzazione

**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




# 2.5
# Protezione

# Disturbi e Guasti

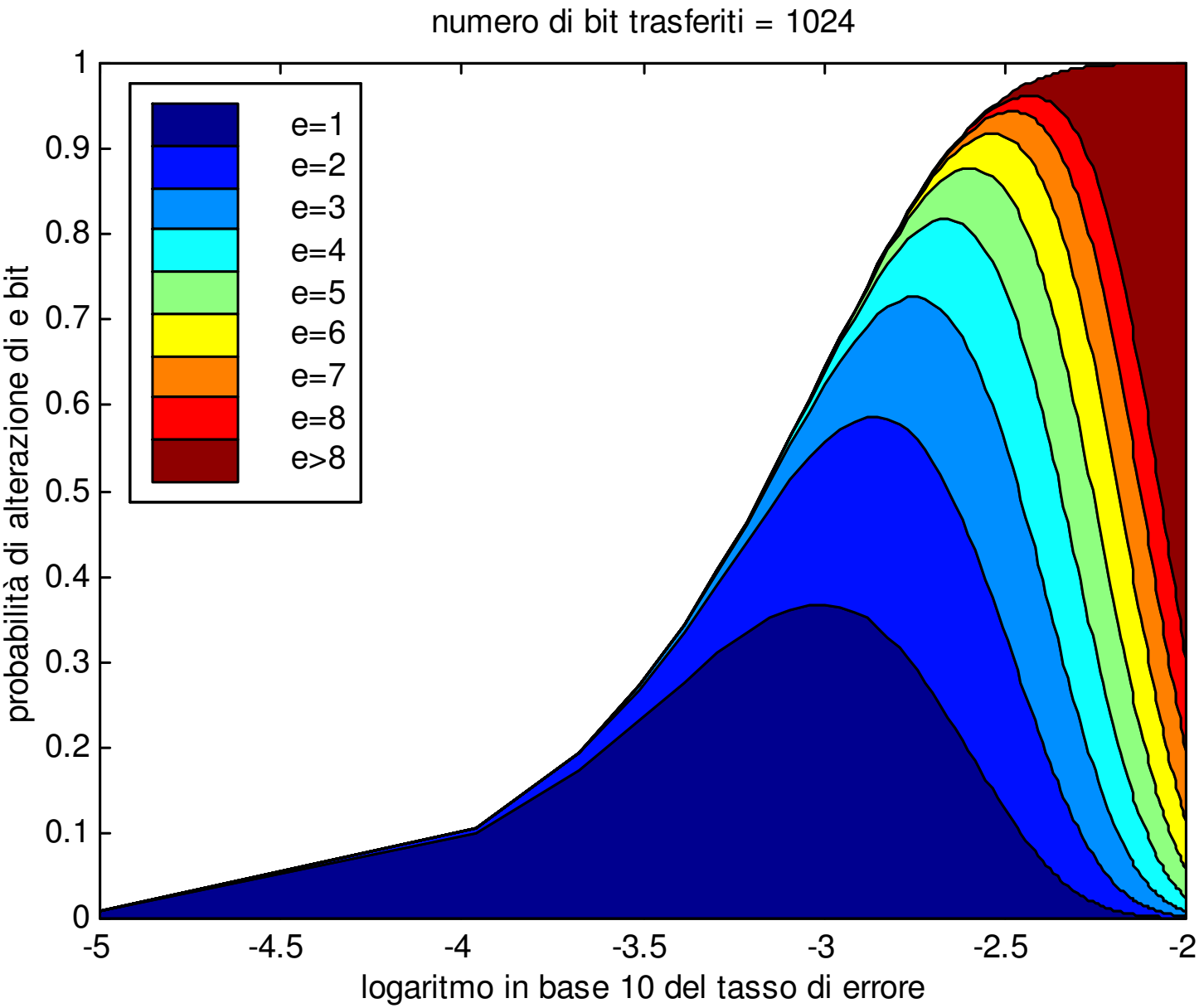

- *linea di trasmissione*
- *unità di memoria*

Agendo opportunamente a livello di realizzazione fisica del canale, si può formulare l'ipotesi che l'alterazione di un bit (o errore) nell'ambito di una stringa di **n** bit sia un evento aleatorio
a) indipendente dalla posizione del bit nella stringa,
b) caratterizzato da una probabilità di occorrenza **p** (tasso di errore).
Conseguentemente la probabilità che si abbiano **e** errori è data da:

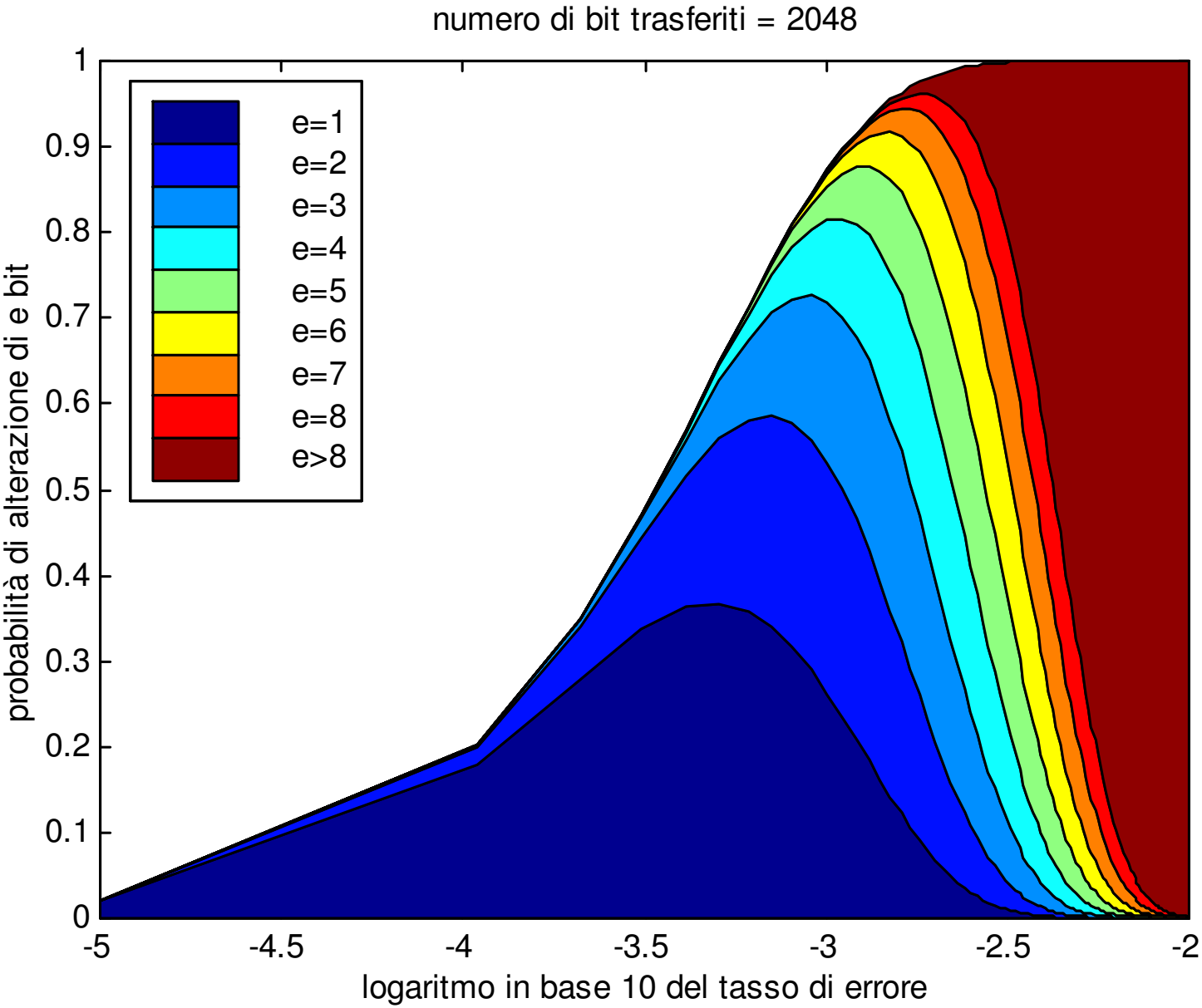
$$P_e = \binom{n}{e} \cdot p^e \cdot (1-p)^{n-e}$$

**Se $n \cdot p << 1 \rightarrow P_0 >> P_1 >> P_2 >> \ldots$**

# L'ipotesi degli errori indipendenti

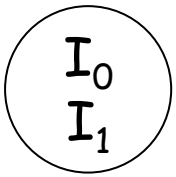


# L'ipotesi degli errori indipendenti

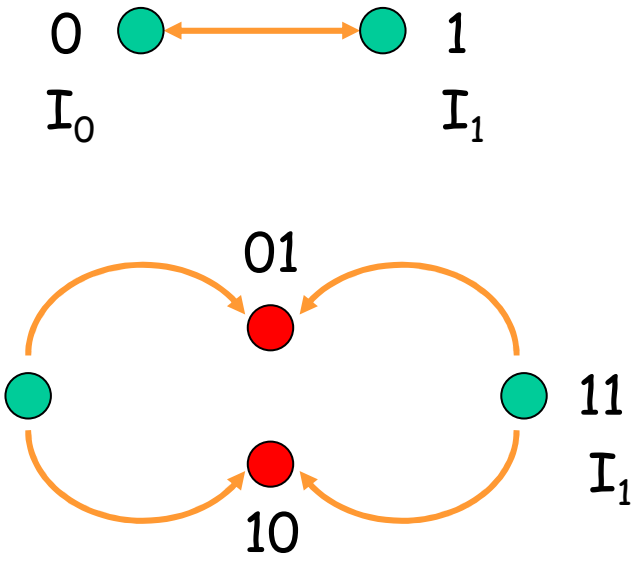


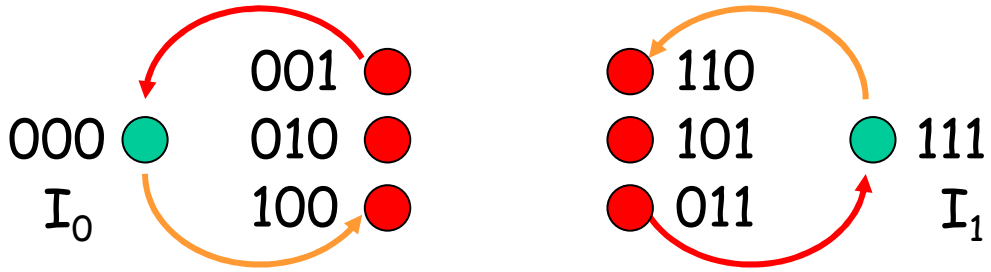
# Rilevazione/Correzione di errori singoli (——)

| | R | C |
|---|---|---|
| $n = 1 = n_{min}$ | NO | NO |
| $n = 2$ | SI | NO |
| $n = 3$ | SI | SI |




M = 2




se $P_1 >> P_2$: correzione errore (——)

# Distanza minima di un codice

**Distanza fra due configurazioni binarie A, B di n bit:** D(A,B)
numero di bit omologhi in A, B con valore diverso.

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

**Distanza minima di un codice C:** $D_{MIN}(C)$
valore minimo della distanza tra due qualsiasi configurazioni utilizzate dal codice C.

- I codici non ridondanti hanno $D_{MIN} = 1$.
- I codici ridondanti possono avere $D_{MIN} > 1$.

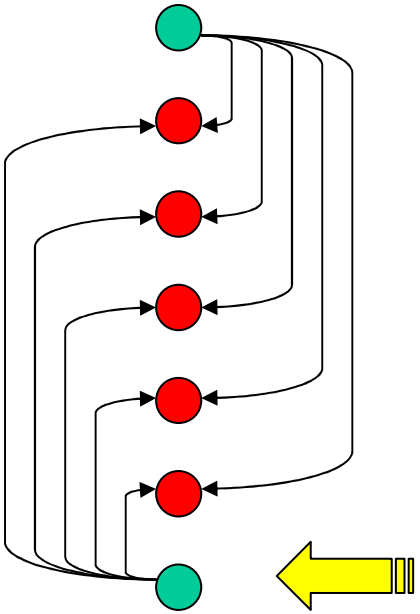
Esempi:
$D_{MIN}$ (Codice BCD) = 1
$D_{MIN}$ (Codice 1/10) = 2; $D_{MIN}$ (Codice 7-Segmenti) = 1

# Error(s) Detecting/Correcting Codes

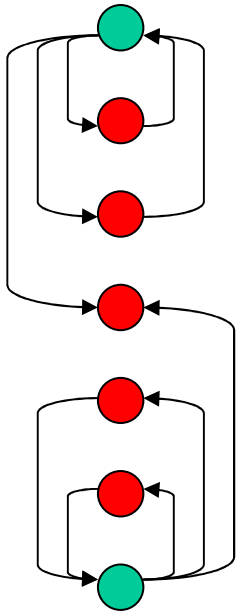
Un codice con $D_{MIN}$

**R+1** consente la rilevazione di **R** errori

**R+C+1 (R ≥ C)** consente la correzione di **C** errori e la rilevazione di **R** errori



| $D_{MIN}$ | EDC | ECC | |
|---|---|---|---|
| 1 | - | - | - |
| 2 | 1 | - | 1 |
| 3 | 2 | 1 | - |
| 4 | 3 | 1 | 2 |
| 5 | 4 | 2 | - |
| 6 | 5 | 2 | 3 |
| ... | ... | ... | ... |

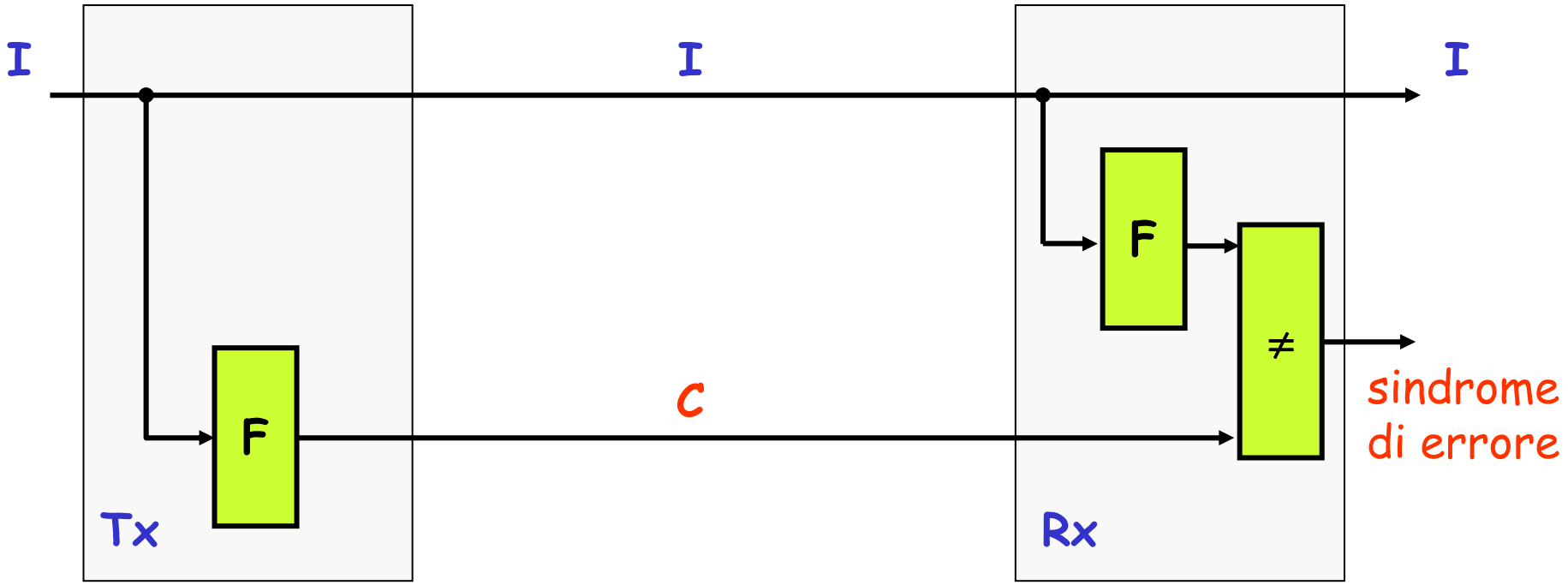


| n° max di errori contemporanei: | rilevabili | rilevabili e correggibili | rilevabili soltanto |
|---|---|---|---|

# Codici separabili: rilevazione di errori singoli

| "I" bit di informazione (*information bits*) | "C" bit di controllo (*check bits*) |
|---|---|

$$\forall\ I \rightarrow C = 1$$

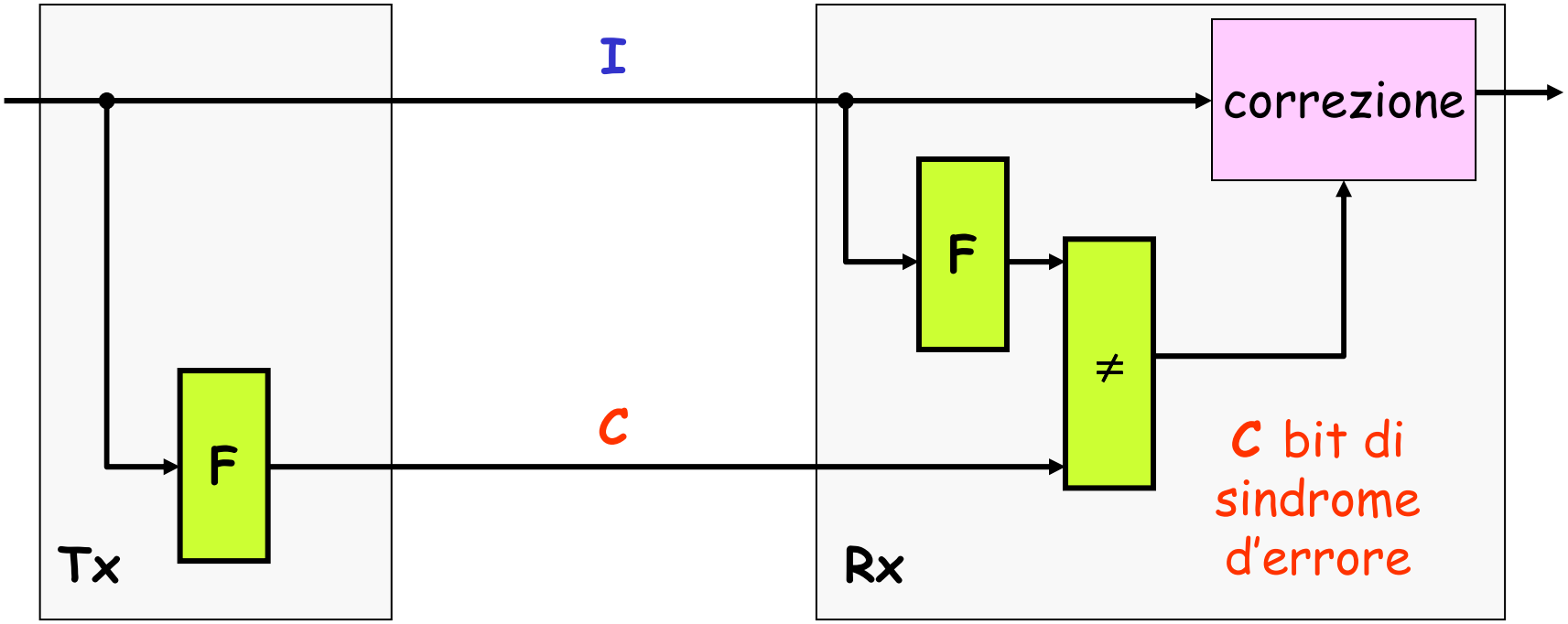



Le due configurazioni 1, 0 della **sindrome di errore** identificano rispettivamente la presenza e l'assenza di un errore singolo.

# Codici separabili: correzione di errori singoli

$$\forall\ I \rightarrow C : 2^C \geq I + C + 1$$

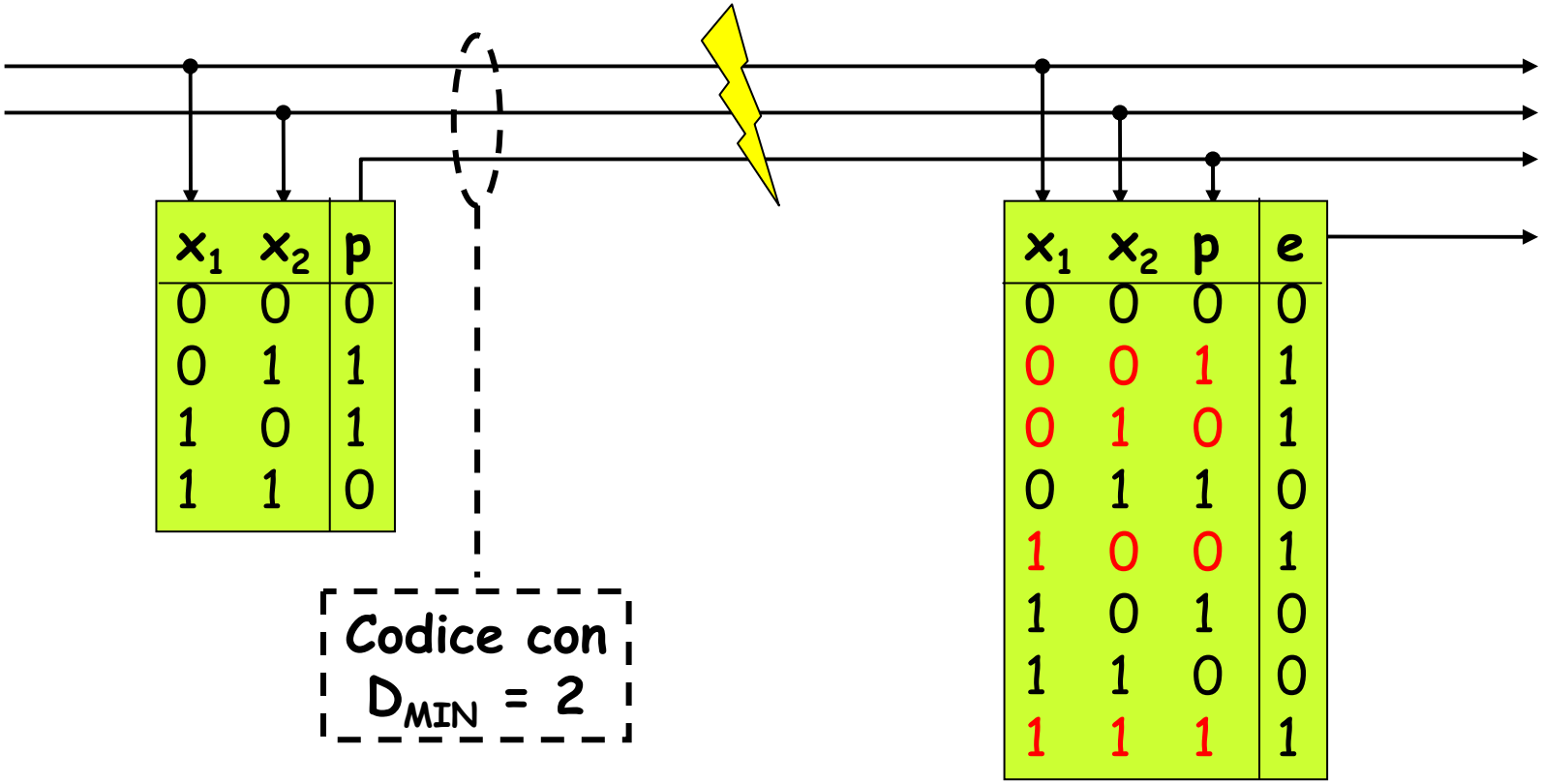
| I | 1 | 2 | 3 | 4 | 5 | 6 | 7 | … |
|---|---|---|---|---|---|---|---|---|
| C | 2 | 3 | 3 | 3 | 4 | 4 | 4 | … |




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).

# SINGLE EDC: ∀ codice + bit di parità

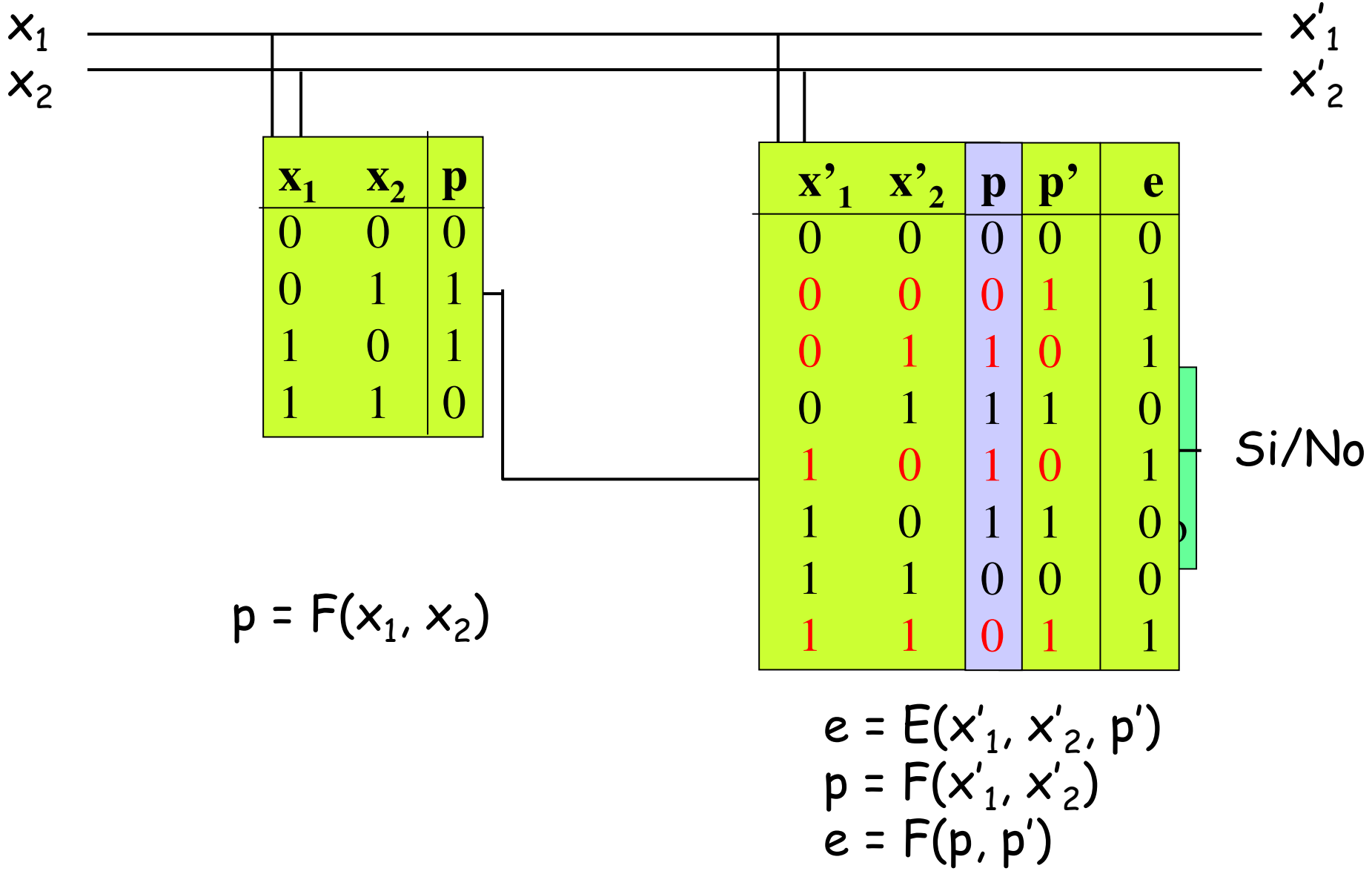
**Bit di parità p** - bit che la sorgente aggiunge ad una stringa di bit di codifica al fine di renderne **pari** il n° di "uni".
**Errore di parità e** - bit che la destinazione pone a 1 se e solo se riceve una configurazione con un numero **dispari** di "uni".




| $x_1$ | $x_2$ | 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 della parità e della sindrome d'errore




| $x_1$ | $x_2$ | p |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| $x'_1$ | $x'_2$ | p | p' | e |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |

# SINGLE ECC: il codice di Hamming

$$I = 4: \; 2^C \geq I + C + 1 \rightarrow C = 3 \rightarrow I + C = 7$$

| M=10 info | I=4 (BCD) $X_3X_2X_1X_0$ | C=3 $H_4H_2H_1$ |
|---|---|---|
| 0 | 0000 | 000 |
| 1 | 0001 | 111 |
| 2 | 0010 | 110 |
| 3 | 0011 | 001 |
| 4 | 0100 | 101 |
| 5 | 0101 | 010 |
| 6 | 0110 | 011 |
| 7 | 0111 | 100 |
| 8 | 1000 | 011 |
| 9 | 1001 | 100 |

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| $H_1$ | $H_2$ | $X_3$ | $H_4$ | $X_2$ | $X_1$ | $X_0$ |

**Tx**

$$H_1 = X_3 \oplus X_2 \oplus X_0$$
$$H_2 = X_3 \oplus X_1 \oplus X_0$$
$$H_4 = X_2 \oplus X_1 \oplus X_0$$

**Rx**

$$E_1 = X_3' \oplus X_2' \oplus X_0' \oplus H_1'$$
$$E_2 = X_3' \oplus X_1' \oplus X_0' \oplus H_2'$$
$$E_4 = X_2' \oplus X_1' \oplus X_0' \oplus H_4'$$

Se $E_4 = E_2 = E_1 = 0$

**OK**

In caso contrario:

$E_4 E_2 E_1$ = indice del bit affetto da errore e quindi da correggere (complementare)

**NO**

# Esempi:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| $H_1$ | $H_2$ | $X_3$ | $H_4$ | $X_2$ | $X_1$ | $X_0$ |

**Tx**

"9": $X_3X_2X_1X_0 = 1001$

$H_1 = X_3 \oplus X_2 \oplus X_0 = 0$

$H_2 = X_3 \oplus X_1 \oplus X_0 = 0$

$H_4 = X_2 \oplus X_1 \oplus X_0 = 1$

$X_3X_2X_1X_0H_4H_2H_1 = 1001100$

## Caso 1: nessun errore

$X_3'X_2'X_1'X_0'H_4'H_2'H_1' = 1001100$

**Rx**

$E_1 = X_3' \oplus X_2' \oplus X_0' \oplus H_1' = 0$

$E_2 = X_3' \oplus X_1' \oplus X_0' \oplus H_2' = 0$

$E_4 = X_2' \oplus X_1' \oplus X_0' \oplus H_4' = 0$

$E_4\,E_2\,E_1 = 000 \rightarrow X_k = X_k',\ \forall\ k$

$X_3X_2X_1X_0 = 1001$ ("9") **OK**

## Caso 2: errore singolo

$X_3'X_2'X_1'X_0'H_4'H_2'H_1' = 0001100$

$E_1 = X_3' \oplus X_2' \oplus X_0' \oplus H_1' = 1$

$E_2 = X_3' \oplus X_1' \oplus X_0' \oplus H_2' = 1$

$E_4 = X_2' \oplus X_1' \oplus X_0' \oplus H_4' = 0$

$E_4\,E_2\,E_1 = 011 \rightarrow X_3 = \text{not } X_3'$

$X_k = X_k',\ \forall\ k \neq 3$

$X_3X_2X_1X_0 = 1001$ ("9") **OK**

# Esempi:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| $H_1$ | $H_2$ | $X_3$ | $H_4$ | $X_2$ | $X_1$ | $X_0$ |

**Tx**

"9": $X_3X_2X_1X_0 = 1001$

$H_1 = X_3 \oplus X_2 \oplus X_0 = 0$

$H_2 = X_3 \oplus X_1 \oplus X_0 = 0$

$H_4 = X_2 \oplus X_1 \oplus X_0 = 1$

$X_3X_2X_1X_0H_4H_2H_1 = 1001100$

**Caso 3: errore doppio**

$X_3'X_2'X_1'X_0'H_4'H_2'H_1' = 0011100$

**Rx**

$E_1 = X_3' \oplus X_2' \oplus X_0' \oplus H_1' = 1$

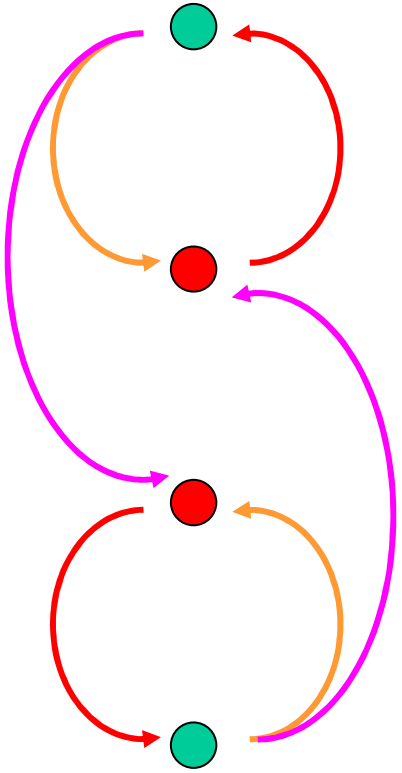
$E_2 = X_3' \oplus X_1' \oplus X_0' \oplus H_2' = 0$

$E_4 = X_2' \oplus X_1' \oplus X_0' \oplus H_4' = 1$

$E_4\,E_2\,E_1 = 101 \rightarrow X_2 = \text{not } X_2'$

$X_k = X_k',\ \forall\ k \neq 2$

$X_3X_2X_1X_0 = 0111$ ("7") **NO**



errore singolo (——)

errore doppio (——)

correzione errore (——)