A.A. 2021-2022

Elementi di Elettronica (INF)

Prof. Paolo Crippa

Sistemi di Numerazione

Notazione Numerica Posizionale

Un numero è rappresentato da una sequenza di cifre, all'interno della quale il valore di ogni cifra dipende dalla sua posizione.

Rappresentazione Decimale: utilizza le 10 cifre "0, 1, 2, 3, 4, 5, 6, 7, 8, 9" (numeri arabici)

$$2546.13_{10} = 2\times10^{3} + 5\times10^{2} + 4\times10^{1} + 6\times10^{0} + 1\times10^{-1} + 3\times10^{-2}$$

Nella rappresentazione decimale, 10 è la **base** o **radice**

Notazione Numerica Posizionale

Rappresentazione Binaria (base 2)

cifre: 0, 1

$$10011_2 = 1\times 2^4 + 0\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0$$

Rappresentazione Ottale (base 8)

cifre: 0, 1, 2, 3, 4, 5, 6, 7

$$43912_8 = 4\times 8^4 + 3\times 8^3 + 9\times 8^2 + 1\times 8^1 + 2\times 8^0$$

Rappresentazione Esadecimale (base 16)

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

$$\mathrm{CA0F3}_{16} = \mathrm{C}\times 16^4 + \mathrm{A}\times 16^3 + 0\times 16^2 + \mathrm{F}\times 16^1 + 3\times 16^0$$

Conversioni

Binario ⇨ Ottale

$$10101101110_2 = \underbrace{010_2}_{2_8}\ \underbrace{101_2}_{5_8}\ \underbrace{101_2}_{5_8}\ \underbrace{110_2}_{6_8} = 2556_8$$

Binario ⇨ Esadecimale

$$10101101110_2 = \underbrace{0101_2}_{5_{16}}\ \underbrace{0110_2}_{6_{16}}\ \underbrace{1110_2}_{E_{16}} = 56E_{16}$$

Binario ⇨ Decimale

$$\begin{aligned}10101101110_2 = {} & 1\times 2^{10} + 0\times 2^9 + 1\times 2^8 + 0\times 2^7 + 1\times 2^6 + \\ & + 1\times 2^5 + 0\times 2^4 + 1\times 2^3 + 1\times 2^2 + 1\times 2^1 + 0\times 2^0 \\ = {} & 1024_{10} + 256_{10} + 64_{10} + 32_{10} + \\ & + 8_{10} + 4_{10} + 2_{10} \\ = {} & 1390_{10}\end{aligned}$$

Conversioni

Ottale ⇨ Binario

$$4133_8 = \underbrace{100_2}_{4_8}\ \underbrace{001_2}_{1_8}\ \underbrace{011_2}_{3_8}\ \underbrace{011_2}_{3_8} = 100001011011_2$$

Ottale ⇨ Esadecimale

$$4133_8 = 100001011011_2 = \underbrace{1000_2}_{8_{16}}\ \underbrace{0101_2}_{5_{16}}\ \underbrace{1011_2}_{B_{16}} = 85B_{16}$$

Ottale ⇨ Decimale

$$4133_8 = 4\times8^3 + 1\times8^2 + 3\times8^1 + 3\times8^0$$

$$= 4\times512 + 1\times64 + 3\times8 + 3\times1 = 2139_{10}$$

Conversioni

Esadecimale ⇨ Binario

$C1A0_{16} = 1100\ \ 0001\ \ 1010\ \ 0000_2$

Esadecimale ⇨ Ottale

$C1A0_{16} = 1100\ \ 0001\ \ 1010\ \ 0000_2$

$= 1\ 100\ 000\ 110\ 100\ 000_2 = 140640_8$

Esadecimale ⇨ Decimale

$C1A0_{16} = 12 \cdot 4096 + 1 \cdot 256 + 10 \cdot 16 + 0 \cdot 1$

$= 49568_{10}$

Conversioni

Decimale ⇨ Binario $218_{10} = 11011010_2$

218_{10} : 2 = 109 resto 0 (LSB)

: 2 = 54 resto 1

: 2 = 27 resto 0

: 2 = 13 resto 1

: 2 = 6 resto 1

: 2 = 3 resto 0

: 2 = 1 resto 1

: 2 = 0 resto 1 (MSB)

Conversioni

Decimale ⇨ Ottale $218_{10} = 332_8$

218_{10} : 8 = 27 resto 2 (LSD)

: 8 = 3 resto 3

: 8 = 0 resto 3 (MSD)

Decimale ⇨ Esadecimale $218_{10} = DA_{16}$

218_{10} : 16 = 13 resto 10 -> A_{16} (LSD)

: 16 = 0 resto 13 -> D_{16} (MSD)

Addizione di Numeri Binari

C_{in}	X	Y	C_{out}	S
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			1 0 1 1 1 1 1 1 0
X		191	1 0 1 1 1 1 1 1
Y	+	153	1 0 0 1 1 0 0 1
X+Y		344	1 0 1 0 1 1 0 0 0

Sottrazione di Numeri Binari

b_{in}	X	Y	b_{out}	d
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			0 0 1 1 1 1 1 1 0
X		224	1 1 1 0 0 0 0 0
Y	-	43	0 0 1 0 1 0 1 1
X -Y		181	1 0 1 1 0 1 0 1

Rappresentazione di Numeri Negativi

Bit di segno : + ⇨ 0 - ⇨ 1

$$0 \mid 1\,0\,1\,1\,1\,0\,1_2 = +\ 93_{10}$$

$$1 \mid 1\,1\,1\,1\,1\,1\,1_2 = -\ 127_{10}$$

Per sommare (sottrarre) numeri con segno, si deve:

- valutare il segno
- se il segno è uguale si sommano
- se il segno è diverso si sottrae il più piccolo dal più grande e si dà il segno del più grande.

Rappresentazione in Complemento a Radice

$$\overline{D} = r^n - D$$

$$\overline{D} = \left[\left(r^n - 1 \right) - D \right] + 1$$

$$\overline{D} = \left[\left(\underbrace{m\,m \;\cdots\; m\,m}_{n} \right) - D \right] + 1 \qquad \text{con} \quad m = r - 1$$

Complemento della cifra $\overline{d} = r - 1 - d$

$$\overline{D} = \overline{d_{n-1}}\;\overline{d_{n-2}} \cdots \overline{d_1}\;\overline{d_0} + 1$$

Complemento a 10 (*r = 10*; *n = 6*)

731892 ⇨ 1000000 - 731892 = 268108

(999999 - 731892) + 1 = 268107 + 1 = 268108

Complemento a 2

num. negativo ⇨ MSB = 1 num. positivo ⇨ MSB = 0

con n bit: rappresentabili da $-\left(2^{n-1}\right)$ a $+\left(2^{n-1}-1\right)$

-4	$100 = \overline{1}\,\overline{0}\,\overline{0} + 1 = 011 + 1$
-3	$101 = \overline{0}\,\overline{1}\,\overline{1} + 1 = 100 + 1$
-2	$110 = \overline{0}\,\overline{1}\,\overline{0} + 1 = 101 + 1$
-1	$111 = \overline{0}\,\overline{0}\,\overline{1} + 1 = 110 + 1$
0	000
1	001
2	010
3	011

n = 3

Esempi di Complemento a 2

$53_{10} = 00110101_2$ *n = 8*

$\Downarrow$ $\overline{\text{bits}}$

11001010

$+1$

$\overline{11001011_2} = -53_{10}$

$1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1_2$

$\downarrow\ \ \downarrow\ \ \ \ \downarrow\ \ \ \ \downarrow\ \ \downarrow$

$-2^7\ \ 2^6\ \ \ \ 2^3\ \ \ \ 2^1\ \ 2^0$

$-128 + 64 + 8 + 2 + 1 = -53_{10}$

La conversione in decimale di un numero in complemento a 2 si calcola considerando che il peso di MSB è di -2^{n-1} se MSB = 1

n = 4

$\overline{0000}$ = 1111 + 1 = 0000 il riporto è ignorato

Il numero -2^{n-1} non ha un corrispondente positivo

$-8_{10} = 1000$ $\overline{1000} = 0111 + 1 = 1000 = -8_{10}$

Complemento a Radice Diminuita

$$\overline{D} = r^n - 1 - D$$

Complemento a 9 ($r = 10$; $n = 6$)

$\overline{731892}$ = 999999 - 731892 = 268107

Complemento a 1

num. negativo ⇨ MSB = 1 num. positivo ⇨ MSB = 0

con n bit: rappresentabili da $-\left(2^{n-1}-1\right)$ a $+\left(2^{n-1}-1\right)$

Complemento a 1

-3	100
-2	101
-1	110
-0	111
0	000
1	001
2	010
3	011

esempio

$$53_{10} = 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1_2$$

$$\Downarrow$$

$$1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0_2$$

$$\downarrow \quad \downarrow \qquad \downarrow \qquad \downarrow$$

$$-(2^7-1) \quad 2^6 \qquad 2^3 \qquad 2^1$$

$$-127 \quad +64 \qquad +8 \qquad +2 = -53_{10}$$

La conversione in decimale di un numero in complemento a 1 si calcola considerando che il peso di MSB è di $-(2^{n-1}-1)$ se MSB=1

Tabella Riassuntiva

Decimale	Compl. a 2	Compl. a 1	Modulo e Segno	Eccesso $2^{m-1} = 4$
-4	100	---	---	000
-3	101	100	111	001
-2	110	101	110	010
-1	111	110	101	011
0	000	000 o 111	100 o 000	100
1	001	001	001	101
2	010	010	010	110
3	011	011	011	111

Addizione in Complemento a 2

Segue le stesse regole dell'addizione ordinaria (ignorando il riporto che eccede l'MSB).

$$\begin{array}{rr} +2 & 0010 \\ +3 & +0011 \\ \hline +5 & 0101 \end{array}$$

$$\begin{array}{rr} +7 & 0111 \\ -4 & +1100 \\ \hline +3 & 0011 \end{array}$$

$$\downarrow$$

$$carry = 1$$

Overflow (Addizione Compl. a 2)

Si ha **overflow** quando la somma eccede il range consentito:

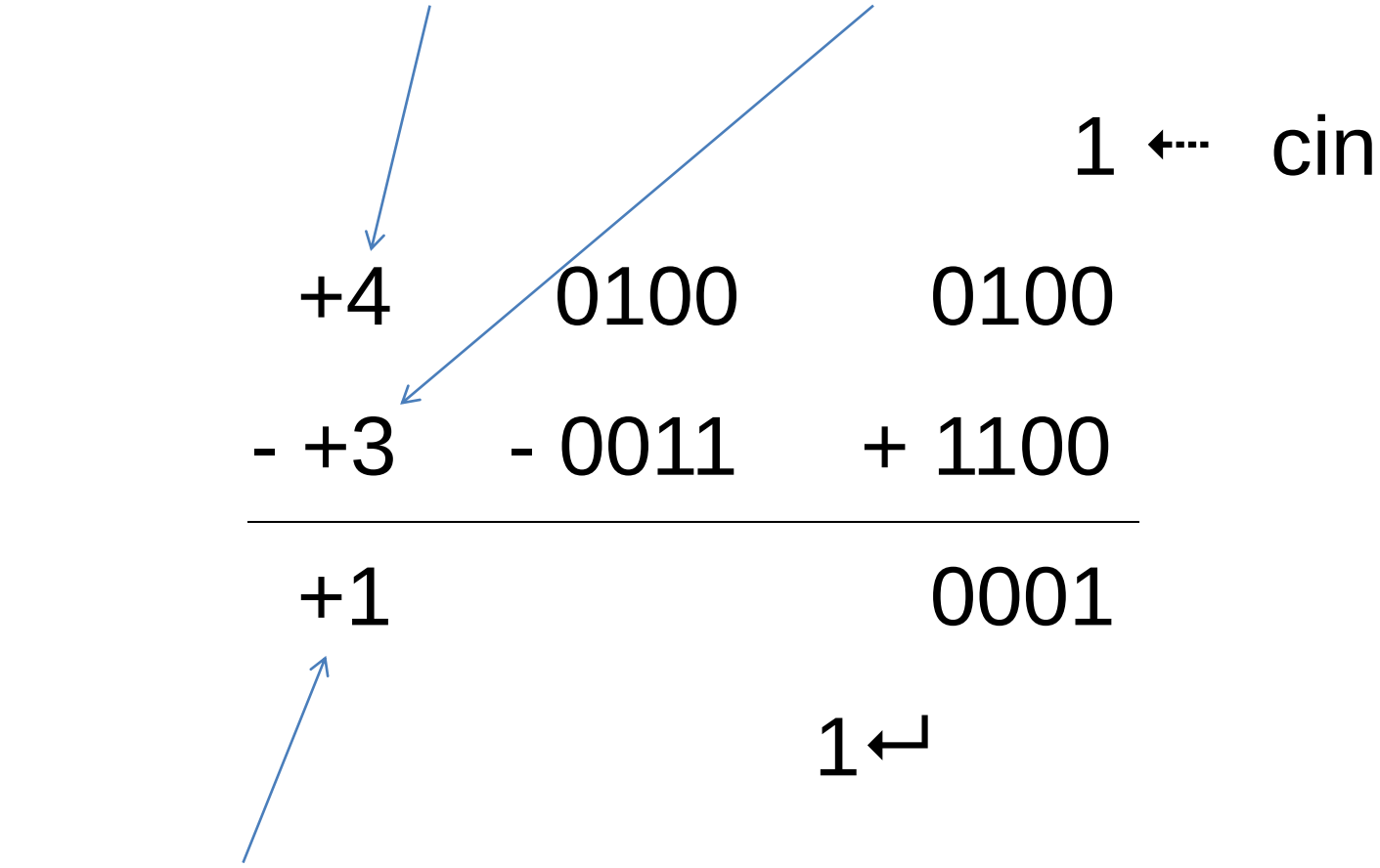
ciò avviene se il segno dei due addendi è lo stesso, e se il segno della somma è differente dal segno degli addendi.

$-\ 7$	1001
$+-\ 6$	$+1010$
-13	$+3 = 0011$

$+\ 5$	0101
$++\ 6$	$+0110$
$+11$	$-5 = 1011$

Sottrazione in Complemento a 2

Si somma il minuendo con il sottraendo complementato a 2



Sottrazione in Complemento a 2

1 ⇠ cin

+3	0011	0011
- +4	- 0100	+ 1011
-1		1111

Sottrazione in Complemento a 2

		1 ⇠	cin
+2	0010		0010
- -3	- 1101	+	0010
+5			0101

Overflow (Sottrazione Compl. a 2)

L'overflow nella sottrazione si può rilevare esaminando il segno del minuendo e del sottraendo complementato usando la stessa regola dell'addizione.

		1 ⇠ cin	
-4	1100	1100	
- +7	- 0111	+ 1000	
-11		0101	= +5
		1↵	

Moltiplicazione: Shift and Add


```
   11   moltiplicando
 x 13   moltiplicatore
-------
   33
  11
-------
  143   prodotto
```

```
  1011
 x 1101
------
  1011
 0000
1011
1011
--------
10001111
```

```
  1011
 x 1101
------
  1011
 0000
-------
 01011
 1011
-------
110111
 1011
-------
10001111
```

Moltiplicazione: Shift and Add


```
   11
 x 13
-----
   33
  11
-----
  143
```

```
   1011
 x 1101
------
   1011
  0000
 1011
1011
--------
10001111
```

```
   1011
 x 1101
------
  01011
 00000
------
 001011
 101100
------
 0110111
 1011000
--------
10001111
```

Estensione di Segno di Numeri in Complemento a 2

100,01 = 1111100,01000

010,01 = 0000010,01000

-4	100
-3	101
-2	110
-1	111
0	000
1	001
2	010
3	011

-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Moltiplicazione in Complemento a 2: Estensione di Segno

-5

x -3

+15

1011

x 1101

11111011

00000000

11111011

11101**00**

11100111

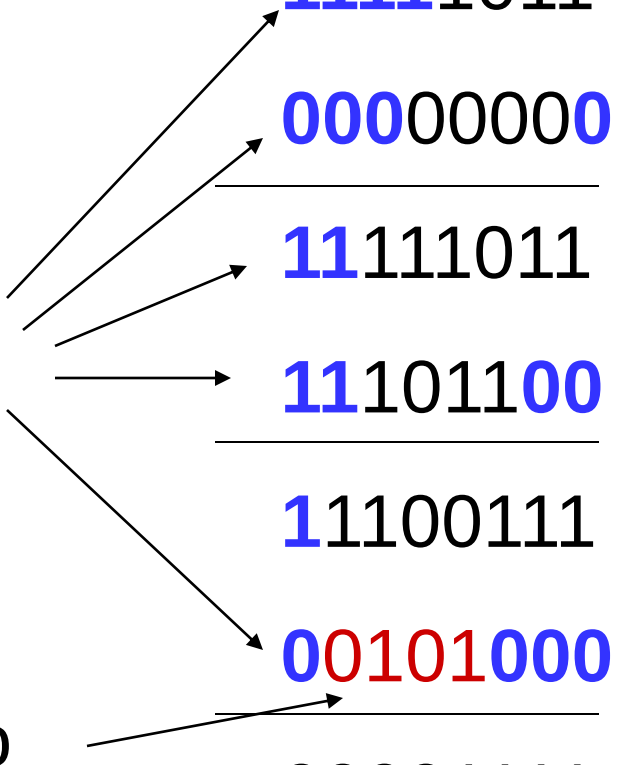
00101000

00001111

1↵

estensione di segno

Ultimo passo:
moltiplicando shiftato
e complementato a 2



Moltiplicazione in Complemento a 2: Estensione di Segno

2 x -3 = - 6

010 x 101

000010

000000

000010

1110**00**

111010

estensione di segno

moltiplicando shiftato e complementato a 2

-3 x 2 = - 6

101 x 010

000000

11101**0**

111010

00000**0**

111010

estensione di segno

Divisione

217 : 11 = 19 resto 8

217	11
- 11	19
107	
- 99	
8	

11011001	1011
1011	10011
0101	
0000	
1010	
0000	
10100	
1011	
10011	
1011	
1000	resto

Si usano algoritmi specifici per la divisione di numeri in complemento a 2

Divisione

217 : 11 = 19 resto 8

217	11	
- 11	19	
107		
- 99		
8		

11011001	1011	quoziente
1011		-> 1
00101001		
1011		-> 0
00101001		
1011		-> 0
00101001		
1011		-> 1
00010011		
1011		-> 1
1000	-> resto	

Si usano algoritmi specifici per la divisione di numeri in complemento a 2

Notazione in Virgola Fissa

0110.011	0110.011
+ 1000.100	x 1000.100
= 1110.111	00110110.001100

Le operazioni si svolgono come se fossero numeri interi

Notazione in virgola mobile

$$N = \pm M \times B^{\pm E}$$

M = mantissa B = base E = esponente

Rappresentazione Non Unica

Lo stesso numero ha diverse rappresentazioni:
per esempio 16 può essere rappresentato da

$$a = 4 \times 2^2 \quad b = 8 \times 2^1 \quad \mathrm{c} = 2 \times 2^3 \quad \mathrm{d} = 32 \times 2^{-1}$$

per una rappresentazione unica si definisce la
rappresentazione normalizzata:
la mantissa è un numero con il ***MSB sempre ad 1***.

La mantissa è un numero positivo o negativo in virgola fissa.
Standard IEEE: rappresentazione in modulo e segno.
La posizione della virgola deve rimanere costante per tutti i numeri e le operazioni aritmetiche.
Tre possibili soluzioni:

- 1XXXXXXX.0
- 0.1XXXXXXX
- 1.XXXXXXXX (è lo standard IEEE)

esempio con normalizzazione 0.1XXXXX
32 valori rappresentabili fra 0.03125 e 7, distribuzione non uniforme

	0.100	0.101	0.110	0.111
	M = 0.5	M = 0.625	M = 0.75	M = 0.875
E = - 4	0.03125	0.0390625	0.046875	0.0546875
E = - 3	0.0625	0.078125	0.09375	0.109375
E = - 2	0.125	0.15625	0.1875	0.21875
E = - 1	0.25	0.3125	0.375	0.4375
E = 0	0.5	0.625	0.75	0.875
E = 1	1	1.25	1.5	1.75
E = 2	2	2.5	3	3.5
E = 3	4	5	6	7

esempio con normalizzazione 0.1XXXXX

32 valori rappresentabili fra 0.125 e 1.875

	0.1000	0.1001	0.1010	0.1011	0.1100	0.1101	0.1110	0.1111
	M = 0.5	M = 0.5625	M = 0.625	M = 0.6875	M = 0.75	M = 0.8125	M = 0.875	M = 0.9375
E = - 2	0.125	0.140625	0.15625	0.171875	0.1875	0.203125	0.21875	0.234375
E = - 1	0.25	0.28125	0.3125	0.34375	0.375	0.40625	0.4375	0.46875
E = - 0	0.5	0.5625	0.625	0.6875	0.75	0.8125	0.875	0.9375
E = 1	1	1.125	1.25	1.375	1.5	1.625	1.75	1.875

ne < nm

Le Eccezioni

Alcuni valori vengono riservati per indicare situazioni particolari.

– ***Overflow*** (E = max, M = 0)

- somma di due numeri con E superiore al range ammesso
- divisione per zero

– ***Not a Number (NaN)*** (E = max, M $\neq$ 0)

- 0 / 0
- radice di un numero negativo

– ***Underflow*** (0, oppure un valore convenzionale)

- numero minore del valore minimo rappresentabile

Esempio

segno (1-bit)	esponente polarizzato (n-bit)	mantissa (m-bit)

esponente *polarizzato* ottenuto sommando una costante uguale al valore minimo cosicché e ≥ 0

Esempio $(5.5)_{10} = (101.1)_2$

Normalizzazione: $(101.1)_2 = (1.011)_2 x 2^2$

$1x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} = (1x2^0 + 0x2^{-1} + 1x2^{-2} + 1x2^{-3}) \times 1x2^2$

esempio n = 7, m = 8
bias = 64 ⇨ e= 0000010 + 1000000 (bias)

segno	esponente	mantissa
0	100 0010	0110 0000

Standard IEEE 754 – 1985

Lo standard IEEE per il calcolo in virgola mobile "*IEEE Standard for Binary Floating-Point Arithmetic* (**ANSI/IEEE Std 754-1985**)" noto anche come "**IEC 60559:1989**, *Binary floating-point arithmetic for microprocessor systems"* è lo standard più diffuso nel campo del calcolo automatico e adottato dai principali fabbricanti di microprocessori

Successivamente è stato introdotto lo standard *"IEEE Standard for Radix-Independent Floating-Point Arithmetic* (**IEEE 854-1987**)*"* per generalizzare l' IEEE 754-1985 in modo da coprire sia l'aritmetica decimale che binaria

E' stata pubblicata nell'agosto 2008 una revisione dello standard che va sotto il nome "*IEEE Standard for Floating-Point Arithmetic* (**IEEE 754-2008**)" che sostituisce (includendoli) sia lo standard IEEE 754-1985 che il successivo IEEE 854-1987

Standard IEEE 754 – 1985

	Precisione			
	Singola	**Singola Estesa**	**Doppia**	**Doppia Estesa**
bit mantissa	23	≥ 31	52	≥ 63
e_{max}	127	≥ 1023	1023	≥ 16383
e_{min}	-126	≤ -1022	-1022	≤ -16382
Bias	127	Non def.	1023	Non def.
bit esponente	8	≥ 11	11	≥ 15
Bit totali	32	≥ 43	64	≥ 79

Bit di segno : + ⇨ 0 - ⇨ 1

Standard IEEE 754 – 1985: Singola Precisione 32 bit

$$\mathrm{E}=0, \quad \mathrm{m}=0 \quad \rightarrow \quad a=(-1)^{\mathrm{s}}\times 0$$

$$\mathrm{E}=255, \quad \mathrm{m}=0 \quad \rightarrow \quad a=(-1)^{\mathrm{s}}\times \infty$$

$$\mathrm{E}=255, \quad \mathrm{m}\neq 0 \quad \rightarrow \quad a=\mathrm{NaN}$$

$$0<\mathrm{E}<255 \quad \rightarrow \quad a=(-1)^{\mathrm{s}}\times 2^{(\mathrm{E}-127)}\times(1.\,\mathrm{m})$$

$$\mathrm{E}=0,\ \mathrm{m}\neq 0 \quad \rightarrow \quad a=(-1)^{\mathrm{s}}\times 2^{(-126)}\times(0.\,\mathrm{m})$$

Standard IEEE 754 – 1985: Doppia Precisione 64 bit

segno (1-bit)	esponente polarizzato (11-bit)	mantissa (51-bit)

0 1 11 12 63

$$E = 0, \quad m = 0 \quad \rightarrow \quad a = (-1)^S \times 0$$

$$E = 2047, \quad m = 0 \quad \rightarrow \quad a = (-1)^S \times \infty$$

$$E = 2047, \quad m \neq 0 \quad \rightarrow \quad a = \mathrm{NaN}$$

$$0 < E < 2047 \quad \rightarrow \quad a = (-1)^S \times 2^{(E-1023)} \times (1.\,m)$$

$$E = 0, \; m \neq 0 \quad \rightarrow \quad a = (-1)^S \times 2^{(-1022)} \times (0.\,m)$$

Standard IEEE 754 – 1985: Esempi

0 1000 0000 0000 0000 0000 0000 000 =

$$+1\times 2^{(128-127)}\times 1.0 = 2$$

0 1000 0001 1010 0000 0000 0000 000 =

$$+1\times 2^{(129-127)}\times 1.101 = 6.5$$

1 1000 0001 1010 0000 0000 0000 000 =

$$-1\times 2^{(129-127)}\times 1.101 = -6.5$$

0 0000 0000 1000 0000 0000 0000 000 =

$$+1\times 2^{(-126)}\times 0.1 = 2^{(-127)}$$

Codici Binari per Numeri Decimali

Codici Binari: insieme di stringhe a n-bit in cui ogni stringa rappresenta un numero o un altro simbolo.

- Il ***codice BCD*** (binary-coded decimal) rappresenta le 10 cifre decimali con i bit da 0000 a 1001.

Le parole da 1010 a 1111 non vengono usate.

Ogni parola a 4-bit viene sostituita con una cifra decimale.

Viene usata una stringa a 4-bit per il segno. 0000 = +, 1001 = -

Addizione: è come la somma di numeri a 4-bit

Se il risultato eccede 1001, si deve correggere sommando + 6.

```
    0 1 0 1
+   1 0 0 1
-------------
    1 1 1 0
+   0 1 1 0   correzione
-------------
0 0 0 1 0 1 0 0
```

Codici BCD e 2421

Il codice BCD è un codice pesato viene anche detto **Codice - 8421** : $1001 = 1x8 + 0x4 + 0x2 + 1x1 = 9_{10}$

Codice - 2421 : $1001 = 1x2 + 0x4 + 0x2 + 1x1 = 3_{10}$

Il vantaggio di questo codice è che è autocomplementante $\Rightarrow$ il complemento a 9 si può ottenere complementando i singoli bit.

$9_{10} = 1111_{2421}$ complemento ⇨ $0000_{2421} = 0_{10}$

$5_{10} = 1011_{2421}$ complemento ⇨ $0100_{2421} = 4_{10}$

Altri Codici Binari

Codice Eccesso 3

E’ autocomplementante

Si ottiene dal codice BCD sommando 3 = 0011_2

Codice Biquinary

Usa 7 bit :

primi due bit = 01 per il range 0-4

primi due bit = 10 per il range 5-9

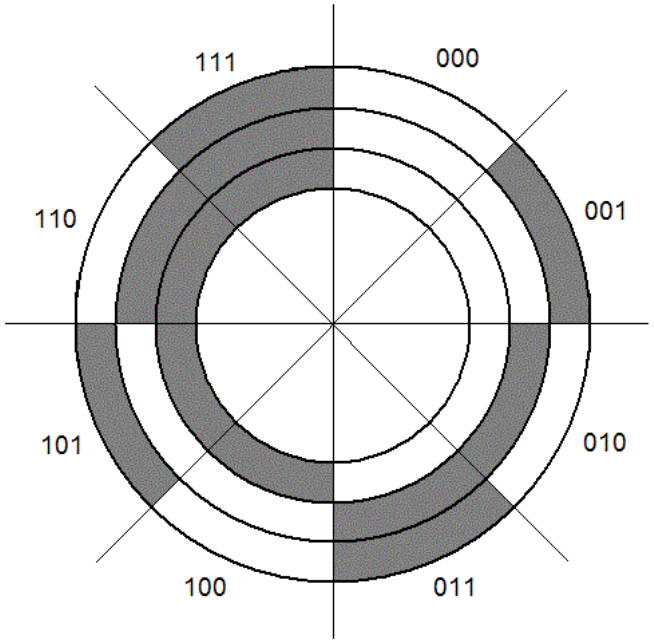
Codice 1-out-of-10

Usa 10 bit : uno solo è uguale a 1 gli altri sono 0

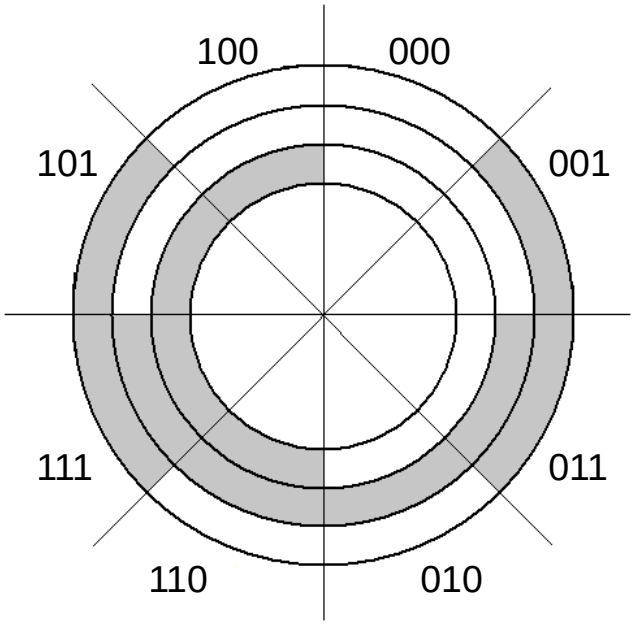
Tabella di Conversione fra Codici Binari

Decimale	BCD	2421	Eccesso-3	Biquinary	1-out-of-10
0	0000	0000	0011	0100001	1000000000
1	0001	0001	0100	0100010	0100000000
2	0010	0010	0101	0100100	0010000000
3	0011	0011	0110	0101000	0001000000
4	0100	0100	0111	0110000	0000100000
5	0101	1011	1000	1000001	0000010000
6	0110	1100	1001	1000010	0000001000
7	0111	1101	1010	1000100	0000000100
8	1000	1110	1011	1001000	0000000010
9	1001	1111	1100	1010000	0000000001
Num. di code-word non utilizzate	6	6	6	2^7-10=118	2^{10}-10=1014

Codice Gray



Bin.	Dec.	Gray
000	0	000
001	1	001
010	2	011
011	3	010
100	4	110
101	5	111
110	6	101
111	7	100

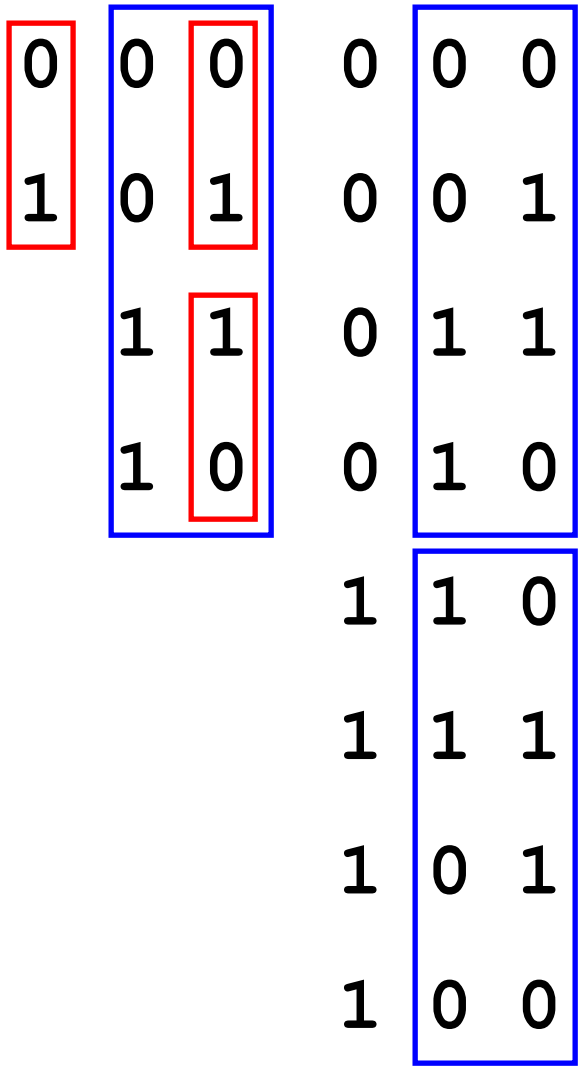


Nel codice Gray si hanno variazioni di un solo bit

Costruzione del Codice Gray a n-bit

Algoritmo

1. Il codice Gray a 1 bit ha solo 2 parole, 0 e 1.
2. Le prime 2^n parole del codice a (n+1)-bit sono uguali a quelle del codice a n-bit scritte nello stesso ordine, con uno “0” come (n+1)-bit.
3. Le restanti 2^n parole sono uguali a quelle del codice a n-bit ma scritte in ordine inverso, con un “1” come (n+1)-bit.



Codice ASCII (standard No. X3.4-1998 American National Standards Institute)

Il codice comunemente usato per i caratteri è il codice **ASCII**

"American Standard Code for Information Interchange"

Per esempio la parola

"Yahoo"

è rappresentata da:

1011001 1100001 1101000 1101111 1101111

		$b_6\,b_5\,b_4$ (colonna)							
	Riga	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
$b_3b_2b_1b_0$	**Hex**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0000	0	NUL	DLE	SP	0	@	P	‘	p
0001	1	SOH	DC1	!	1	A	Q	a	q
0010	2	STX	DC2	“	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB	’	7	G	W	g	w
1000	8	BS	CAN	(	8	H	X	h	x
1001	9	HT	EM	)	9	I	Y	i	y
1010	A	LF	SUB	*	:	J	Z	j	z
1011	B	VT	ESC	+	;	K	[	k	{
1100	C	FF	FS	,	<	L	\	l	|
1101	D	CR	GS	-	=	M	]	m	}
1110	E	SO	RS	.	>	N	^	n	~
1111	F	SI	US	/	?	O	_	o	DEL

Codice ASCII: Tabella dei Simboli

Control Code

NUL	Null	**DLE**	Data link escape
SOH	Start of heading	**DC1**	Device control 1
STX	Start of text	**DC2**	Device control 2
ETX	End of text	**DC3**	Device control 3
EOT	End of trasmission	**DC4**	Device control 4
ENQ	Enquiry	**NAK**	Negative acknowledge
ACK	Acknowledge	**SYN**	Synchronize
BEL	Bell	**ETB**	End transmitted block
BS	Backspace	**CAN**	Cancel
HT	Horizontal tab	**EM**	End of medium
LF	Line feed	**SUB**	Substitute
VT	Vertical tab	**ESC**	Escape
FF	Form feed	**FS**	File separator
CR	Carriage return	**GS**	Group separator
SO	Shift out	**RS**	Record separator
SI	Shift in	**US**	Unit separator
SP	Space	**DEL**	Delete or rubout

Codici per Azioni, Condizioni e Stati

n - differenti azioni, o condizioni, o stati possono essere rappresentate con un codice a b-bit

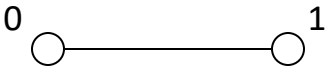
b è il più piccolo intero maggiore di $\log_2(n)$

Esempio: Controllo di un semaforo. Gli stati del semaforo sono codificati con una parola a 3-bit.

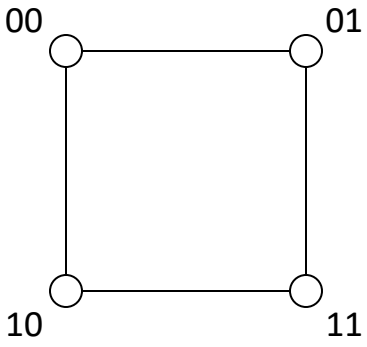
State	NS Green	NS Yellow	NS Red	EW Green	EW Yellow	EW Red	Code
NS go	ON	off	off	off	off	ON	000
NS wait	off	ON	Off	off	off	ON	001
NS delay	off	off	ON	off	off	ON	010
EW go	off	off	ON	ON	off	off	100
EW wait	off	off	ON	off	ON	off	101
EW delay	off	off	ON	off	off	ON	110

Rappresentazione Geometrica di Numeri Binari

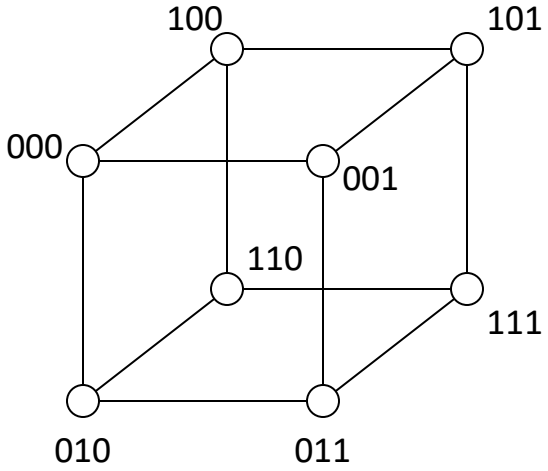
zero - cubo



1 - cubo



2 - cubo



3 - cubo

Distanza di Hamming

Si definisce **somma modulo-*2*** tra due bit, indicata con il simbolo $\oplus$, l'operazione:

$0 \oplus 0 = 0$ $\quad 1 \oplus 0 = 1$ $\quad 0 \oplus 1 = 1$ $\quad 1 \oplus 1 = 0$

0 se gli ingressi sono uguali, 1 se sono diversi

Si definisce somma modulo-2 di due punti dell'n-cubo Pi e Pj

$$P_k = P_i \oplus P_j = \left(a_{n-1} \oplus b_{n-1},\ a_{n-2} \oplus b_{n-2},\ \ldots,\ a_0 \oplus b_0 \right)$$

Si indica con |Pk| il numero di "1" di Pk

Si definisce distanza di Hamming (o metrica) tra due punti

$$\mathrm{D}\left(\mathrm{P_i}, \mathrm{P_j}\right) = \left| \mathrm{P}_i \oplus \mathrm{P_j} \right|$$ **distanza di Hamming**

Distanza di Hamming

Valgono le seguenti proprietà:

$D(P_i, P_j) = 0$ se e solo se $P_i = P_j$

$D(P_i, P_j) = D(P_j, P_i) > 0$ se $P_i \neq P_j$

$D(P_i, P_j) + D(P_j, P_k) \geq D(P_i, P_k)$ disuguaglianza triangolare

Esempio

a = 1 0 1 1 1 0 0 0

b = 0 1 0 1 0 1 0 1

a ⊕ b = 1 1 1 0 1 1 0 1 D(a,b) = 6

Due punti adiacenti su un n-cubo hanno distanza unitaria.

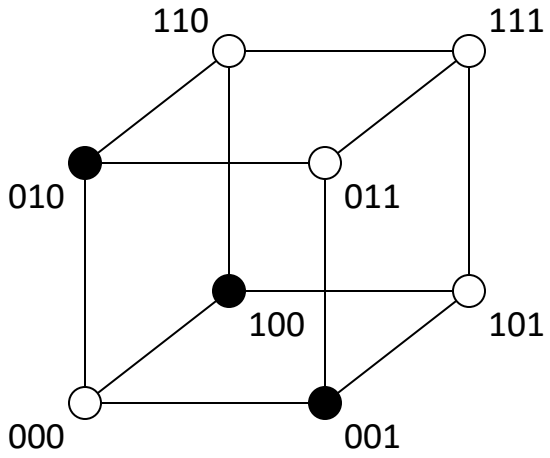
Il codice Gray è a distanza unitaria.

Codici a Rilevazione di Errore

Un codice che usa stringhe di n-bit non necessariamente contiene 2^n parole valide (*code word*).

Un codice rilevatore di errore ha la proprietà che un'alterazione della parola produce (con elevata probabilità) una stringa di bit che non è una parola del codice (*non coded word*).

Se una stringa è
- una code word $\Rightarrow$ **corretta**
- una non code word $\Rightarrow$ **errata**



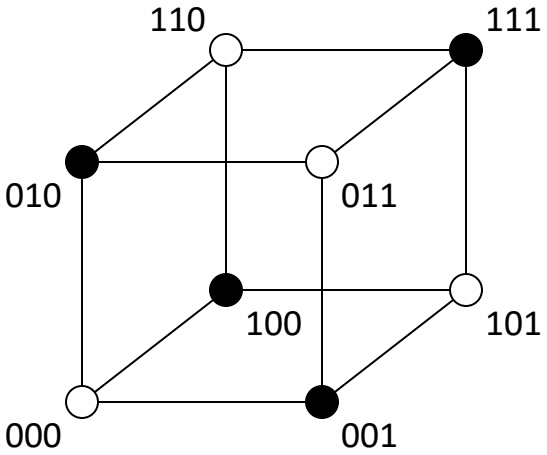
○ = code word

● = non code word

Un errore di un bit di 111 lo può cambiare in 110, 011, 101

Questa codifica non ha proprietà di rilevazione del singolo errore

Codici a Rilevazione di Errore



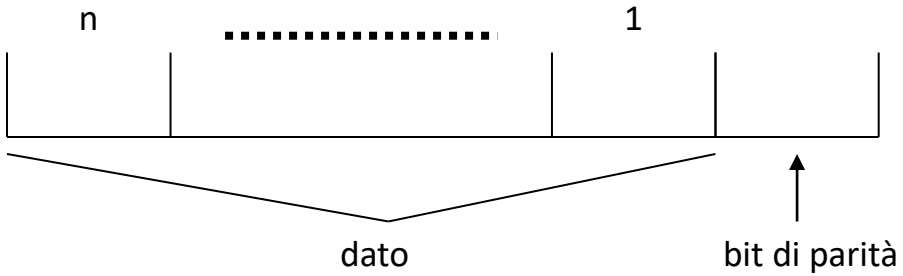
○ = code word

● = non code word

Ha proprietà di rilevazione dell'errore singolo

- Un codice rileva i singoli errori se la *minima distanza* = 2

Sono necessari n+1 bit per costruire un codice rilevatore di singolo errore con 2^n parole



Codici a Rilevazione di Errore

- **Bit di parità pari**:

  0 se il numero di 1 è pari

  1 se il numero di 1 è dispari

  Bit di parità dispari:

  1 se il numero di 1 è pari

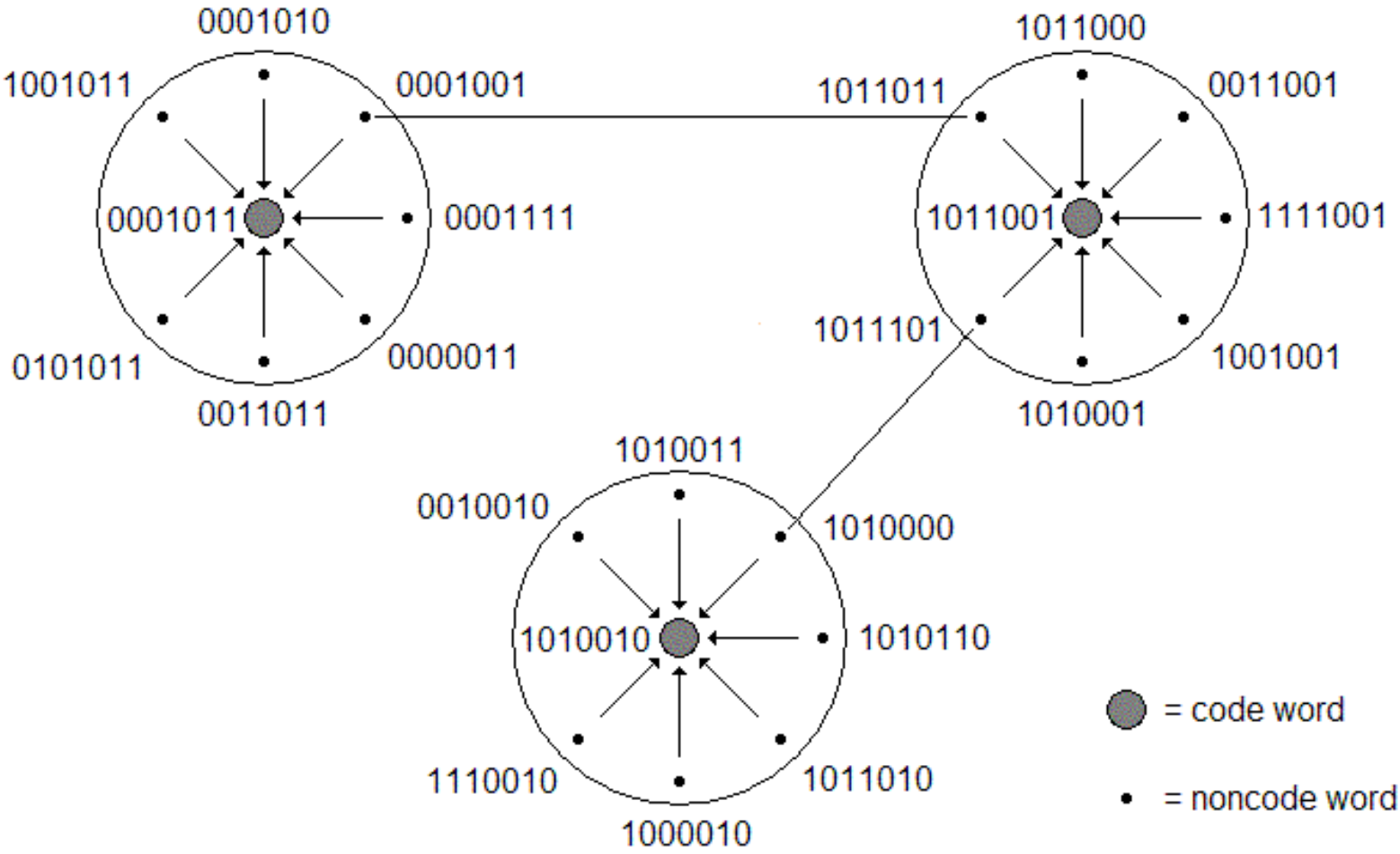
  0 se il numero di 1 è dispari

Es:	dato	cod. a parità pari	cod. a parità dispari
	0101110	01011100	01011101
	1011000	10110001	10110000

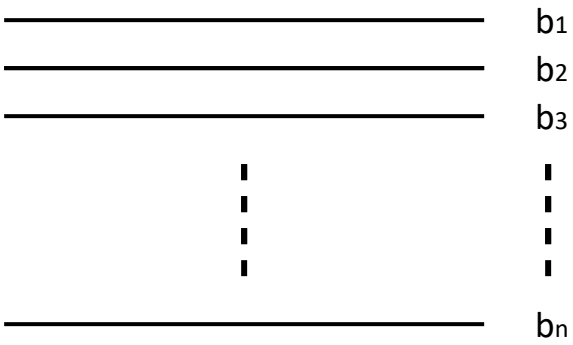
Codici Correttori di Errore

Esempio: codice a distanza 3

Anche con un errore di un bit è possibile riottenere il valore corretto: correzione di errore singolo

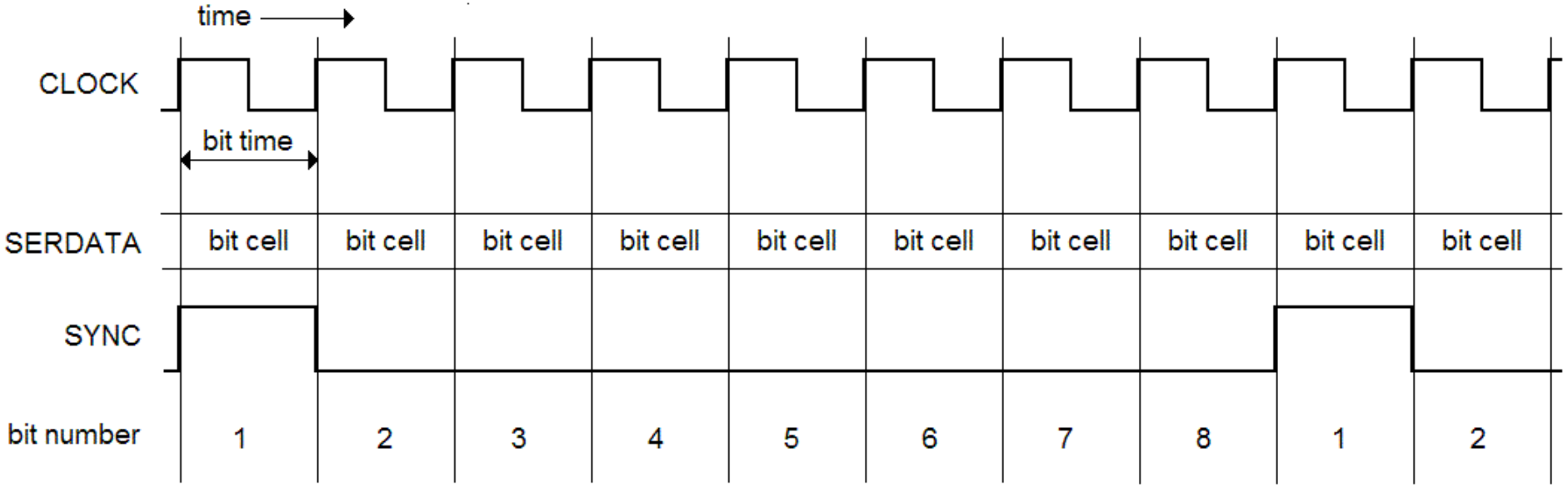


Trasmissioni Seriali e Parallele

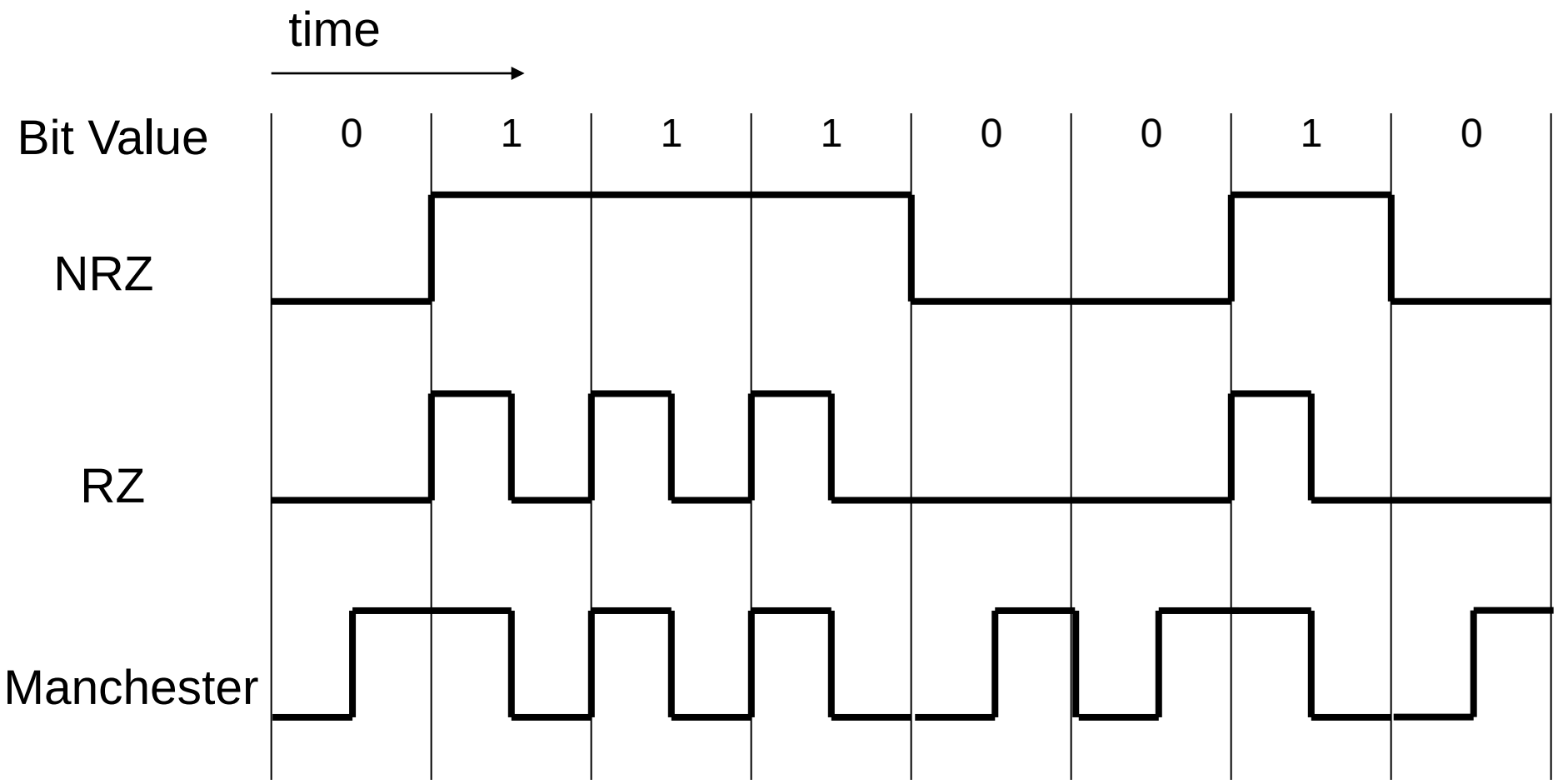


Trasmissioni seriali sincrone:

è necessario trasmettere un segnale di clock, uno di sincronismo e la linea dati



Codici Seriali



- **NRZ** (Non Return to Zero): il valore del bit è mantenuto per tutto il tempo di bit
- **Manchester**: una transizione low-high rappresenta uno 0, una transizione high-low rappresenta un 1